REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 May, 1995	Final Report: 8/1/91-3/31/95

4. TITLE AND SUBTITLE
Integration of Symbolic & Numerical Methods & Their Applications in Artificial Intelligence

5. FUNDING NUMBERS
AFOSR910361

6. AUTHOR(S)
Author: Deepak Kapur
Researchers: Deepak Kapur, Joseph Mundy, David Forsyth, Andrew Zisserman, Tushar Saxena, Lu Yang, Mohsin Ahmed, Xumin Nie

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Research Foundation of SUNY
University at Albany, SUNY
Office of Research, AD 218
1400 Washington Avenue
Albany, NY 12222

8. PERFORMING ORGANIZATION REPORT NUMBER
320-2258A

AFOSR-TR- 95-0462

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
AFOSR/PKA
110 Duncan Avenue, Suite B115
Bolling AFB
Washington, DC 20332-0001

nm

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DTIC ELECTE JUN 27 1995 F

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

An indexed-based object recognition system using geometric invariance techniques has been designed, and used to recognize buildings in an image of a military site and for recognizing curved planar objects including gaskets. New invariants and indexing techniques for polyhedral and curved objects with repetition or bilateral symmetry and objects with the imaged outline of a surface of revolution have been developed. A method to distinguish projectively equivalent but Euclidean distinct objects in an uncalibrated view has been investigated. A group-theoretic framework for relating quasi-invariants to invariants has been formulated.

Computing invariants can be formulated as an algebraic manipulation problem involving variable elimination and solving nonlinear polynomial equations. Based on Dixon's formulation of resultants, new methods for eliminating variables have been developed and implemented. These methods are much faster and superior than other elimination techniques. A branch and prune approach for numerically solving polynomial equations has been developed. A simple algorithm for separating invariant relations among object and image features to compute invariants of object features has been designed. These algorithms can serve as a basis for building an invariant work-bench that would enable researchers to experiment with geometric configurations and investigate their geometric invariants.

14. SUBJECT TERMS
Computer Vision, Object Recognition, Indexing, Invariants, Symbolic Computation, Elimination, Projection, Numerical Computation, Interval Methods.

15. NUMBER OF PAGES
12

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

DTIC QUALITY INSPECTED 3

19950626 021

Integrated Symbolic and Numeric Methods for Advanced Computer Vision

Deepak Kapur
Department of Computer Science
State University of New York
Albany, NY 12222

May, 1995

Abstract

An indexed-based object recognition system using geometric invariance techniques has been designed, and used to recognize buildings in an image of a military site and for recognizing curved planar objects including gaskets. New invariants and indexing techniques for polyhedral and curved objects with repetition or bilateral symmetry and objects with the imaged outline of a surface of revolution have been developed. A method to distinguish projectively equivalent but Euclidean distinct objects in an uncalibrated view has been investigated. A group-theoretic framework for relating quasi-invariants to invariants has been formulated.

Computing invariants can be formulated as an algebraic manipulation problem involving variable elimination and solving nonlinear polynomial equations. Based on Dixon's formulation of resultants, new methods for eliminating variables have been developed and implemented. These methods are much faster and superior than other elimination techniques. A branch and prune approach for numerically solving polynomial equations has been developed. A simple algorithm for separating invariant relations among object and image features to compute invariants of object features has been designed. These algorithms can serve as a basis for building an invariant work-bench that would enable researchers to experiment with geometric configurations and investigate their geometric invariants.

Accesion For
NTIS CRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

1 Introduction

Geometric and algebraic invariants have become an active research area in computer vision as they enable view-independent classification and recognition of objects.[1] This approach is particularly attractive because it is scalable and it does not rely on camera geometries. A serious deficiency of most recognition systems is their inefficiency at determining objects in a scene. Most systems attempt to match every object that could be in a scene (i.e. the entire model library) to all combinations of features in an image. Invariant based approach can enable a recognition system to work with a large model base, which has not been possible until now. An invariant based approach can be also used for model acquisition from images.

During this contract, a number of research activities related to computation and use of geometric and algebraic invariants in object recognition systems have been initiated. Mundy, Forsyth and Zisserman have developed an indexed-based object recognition system [20]. The system has been effectively used to recognize buildings in an image of a military site as well as for recognizing curved planar objects including gaskets. It was demonstrated using live images at the *Second European Conference on Computer Vision* in Italy. In order to make the recognition system to work on a larger class of objects including curved objects, new invariants and indexing techniques for polyhedral and curved objects with repetition or bilateral symmetry as well as objects with the imaged outline of a surface of revolution have been developed and integrated into the recognition system. Consistency conditions that allow objects that are Euclidean distinct but projectively equivalent to be distinguished in an uncalibrated view have been integrated into geometric invariance.

Another important development during the current contract has been the development of a theoretical framework for studying quasi-invariants, and analyzing their relationship with invariants [1]. Quasi-invariants were introduced by Binford as an alternative to the strict requirements of invariants.[2] Quasi-invariants are properties which need not be invariant for the whole set of imaging transformations, but are invariant locally for useful portions of the view space.

Discovering invariants is an art. Vision researchers have mostly used invariants in their work which were discovered by geometricists during the last century. Computing invariants can be formulated as an algebraic manipulation problem involving elimination of many variables from polynomial representations of algebraic relationship among object and image features, and extracting invariants from the results of elimination. Symbolic and numerical methods for manipulating nonlinear polynomial equations are helpful in identifying invariant relations for a geometric configuration and hypothesizing invariants based on them.

Based on Dixon's formulation of multivariate resultants, new methods for eliminating variables efficiently have been developed and implemented [15]. In comparison to other

[1] J.L. Mundy and A. Zisserman (eds.), *Geometric Invariance in Computer Vision,* (eds. Mundy and Zisserman), MIT Press, 1992, 1-39. J.L. Mundy, A. Zisserman, and D.A. Forsyth (eds.), *Applications of Invariance in Computer Vision*, LNCS 825, Springer-Verlag, 1994.

[2] T. Binford and T.S. Levitt, "Quasi-invariants: Theory and Exploitation," Proc. *DARPA Image Understanding* workshop, Washington D.C., 1993.

elimination methods including Macaulay and sparse resultants, Gröbner basis methods and triangulation methods, a method based on Dixon formulation is shown to be much faster and far superior in space and time, on a variety of problems from many application domains including vision, invariant computation, robotics and kinematics [13]. Recent theoretical investigations have revealed that methods based on Dixon's formulation are superior because Dixon's formulation implicitly exploits the sparsity of polynomial systems [14].

In collaboration with David McAllester and Pascal Van Hentenryck, a branch and prune algorithm has been developed to numerically find isolated solutions of polynomial constraints [19]. The algorithm has been implemented by Van Hentenryck as the *Newton* system, and has been tried on a many examples. Despite being so simple, the method outperforms conventional interval Newton methods and homotopy-based continuation methods.

A simple algorithm for separating invariant relations among object and image features has been recently designed. This algorithm checks whether a given invariant relation among object and image features is separable, and if so, computes an invariant expression in terms of object features.

We thus have the state-of-the-art methods and implementations for numerically as well as symbolically solving polynomial equations. These advances put us in an excellent position to work towards an invariant work-bench, a software system that would enable researchers to experiment with geometric configurations and investigate their geometric invariant and quasi-invariant relations with respect to different transformation groups. Future planned activities include the development of such an invariant work-bench, containing necessary tools for constructing and manipulating invariants and quasi-invariants.

In the next two sections, we provide some specific details about the main results achieved in geometric invariance and indexing techniques as well as in symbolic and numeric computation methods. Section 4 is the educational impact statement. Section 5 is a list of the published papers citing the contract. A copy of the papers discussing the results in detail are included in the Appendix.

2 Geometric Invariance and Object Recognition

Most of the work in object recognition focussed on expanding the range of objects that could be recognized using the techniques of geometric invariance. New indexing techniques for various families of curved surfaces were developed, particularly, proposing that repetition or symmetry in the geometry of an object yields an extended view of that object, and hence can produce an accurate reconstruction. Geometric invariance techniques have been extended to include a consistency argument that allowed objects that are Euclidean distinct but projectively equivalent to be distinguished in an uncalibrated view.

The ideas underlying object recognition using invariance are quite widely scattered across an increasing literature. In [20], the evolution of geometric invariant approach to object recognition is elaborated. The paper also discusses the relative usefulness of the techniques constructing invariants of three dimensional objects from a single perspective view, and assumptions under which they work. The design of a control architecture for

a system that can efficiently exploit and integrate currently available representational techniques, is presented. For example, there are classes for planar objects, symmetric point sets, canal surfaces, and surfaces of revolution. In [7], additional issues in recovering representations are discussed; it is argued that the importance of markings on objects has been underrated to date.

2.1 Indexing techniques for curved surfaces

It is traditionally harder to recognize curved objects in images because of the complex nature of the relationship between the geometry of the object and what appears in the image. In a variety of situations, constraints on the geometry of the surface make it possible to infer surface geometry from image information alone. It has been shown in a series of papers [5, 6] that in the case of extruded surfaces, the surface's projective geometry is in fact extremely sparse and can be captured by a plane curve, and a line on that curve's plane. For many surfaces, this configuration, which completely represents the surface, generates indexing functions. Most importantly, when surface geometry has been inferred, markings on the surface – edges, grey levels of color values (the property that distinguishes between different soft drink cans, for instance) – can be mapped back onto the recovered surface and used to generate a more distinctive representation.

It was shown that it is possible to recover the complete projective geometry of a generic algebraic surface from a single, uncalibrated image taken from a single, unknown viewpoint [3]. This constructive result is counterintuitive and is also mathematically intricate. Unfortunately, experimental work suggests that the practical consequences of this result are few, as the recognition process involves solving polynomial systems of very high degree with great accuracy and constructing a robust fitting technique.

New invariants for curved objects with bilateral symmetry, objects with the imaged outline of a surface of revolution and straight homogeneous generalized cylinders using constructions based on bi-tangents to the surface have been developed [16]. The class of surfaces are those generated as an envelope of a sphere of varying radius swept along an axis. The class includes canal surfaces, and surfaces of revolution. The representations are computed using only image information, from the symmetry set of the object's outline. They are invariant under weak-perspective imaging, and quasi-invariant to an excellent approximation under perspective imaging. Objects in the same class can be discriminated under perspective projection.

Multiple views of a single object lead to depth information through a process of matching points and triangulation. A single view of an object that has one of a variety of symmetries is conceptually equivalent to a series of views of part of that object; for example, a single view of an object with a twofold symmetry is equivalent to two views of half of that object. We have explored the interactions between object symmetries and the ambiguity in the reconstructed representation available in these kinds of situations.

2.2 Consistency techniques

Indexing by projective invariants, while efficient, has the unfortunate result that projectively equivalent objects cannot be distinguished, as they generate the same representa-

tion. This means that the techniques cannot distinguish between distinct objects related by a projective transformation, for example, boxes of different sizes. In [4], it is argued that this difficulty disappears when more than one object is in view. The ambiguity is resolved because a recognition hypothesis for a single object implicitly contains information about the internal parameters of the camera viewing that object. If two or more objects are present, these hypotheses must be consistent. A recognition system using this view can index the projective structure of objects using projective invariants. It can search over Euclidean distinct, projectively equivalent hypotheses for consistent systems of hypotheses. Euclidean ambiguities about objects are largely a property of the model library. So conditions that give rise to such ambiguities can be analyzed and utilized to distinguish correct systems of hypotheses about Euclidean distinct objects from incorrect hypotheses.

2.3 Quasi-invariants

Based on pragmatic considerations, Binford introduced quasi-invariants as an alternative to the strict requirements of invariants. Quasi-invariants are properties which need not be invariant for the whole set of imaging transformations, but are invariant locally for useful portions of the view space. The advantage is that many more quasi-invariants are available for a given set of geometric features than projective invariants.

Quasi-invariants are not well-understood, in particular, their relation to invariants is unclear. A group-theoretic framework for relating quasi-invariants to invariants as well as for studying the variability of quasi-invariants has been developed [1]. This research would enable to use invariants whenever they exist and are easy to compute, and augment them by quasi-invariants otherwise.

A quasi-invariant is defined in terms of three elements:

- a geometric configuration and associated property of the configuration,
- a distinguished set of image transformations, and
- a specific transformation in the set of image transformations.

Different properties of configurations can serve as quasi-invariant at different points in the transformation group. The set of image transformations need not be a group, but all known quasi-invariants are defined with respect to group invariants. All the examples of quasi-invariants used by Binford can be studied using this group-theoretic framework.

3 Symbolic & Numeric Computation Methods

3.1 Resurrecting Dixon Resultants

While writing an introductory article on elimination methods [10], we came across papers by Dixon written in the early 1900's in which he discussed his attempts to extend Bezout-Cayley's method for formulating resultants for eliminating a single variable from two polynomials. In one of the papers, Dixon was successful in giving a method for eliminating

two variables from three generic bi-degree polynomials.[3] However when polynomials are specialized to have particular coefficients, then Dixon's method does not work. Thus the main limitation of Dixon's approach was that it worked for a very small and restricted class of polynomial systems. This problem of Dixon's approach leading to singular resultant matrices almost always occurs for polynomial systems as number of variables increase.

We have been successful in extending Dixon's approach to work on a large class of polynomial systems. In [15], we showed how Dixon's formulation of multivariate resultants can be extended using *rank submatrix computation*, a simple linear algebra technique. This construction works also for other determinantal formulations of resultants in which matrices could be singular. As an example, the technique of rank submatrix computation can be used as an alternative to perturbation techniques for singular Macaulay matrices, and it has been found to be more efficient in practice than the generalized characteristic polynomial discussed by Canny as there is no need to introduce an extra variable.

We have implemented an interpolation-based approach for computing a projection operator based on the above discussed extension of Dixon's method. This implementation has been tried on many examples from application domains including robotics and inverse kinematics, geometric and solid modeling, engineering design, vision, computational biology as well as geometric theorem proving and problem solving. Using this extension, we have been able to solve many polynomial systems which could not be solved, using reasonable computational time and space, with any other methods including Macaulay resultants, sparse resultants, the Gröbner basis method, and Ritt-Wu's characteristic set method. We are very excited by this development since it appears to be the most efficient method for computing the projection operator in practice.

We have been able to show that in contrast to Macaulay resultants in which the size of the Macaulay matrix for a polynomial system is related to its Bezout bound, Dixon formulation is not governed by Bezout bound [14]. Instead, the size of the Dixon matrix for a polynomial system can be expressed in terms of the volume of polytopes of the polynomial system. The Dixon formulation implicitly exploits the sparsity of polynomial systems. In particular, it is shown that the size of the Dixon matrix of a system of polynomials with the same set of terms is bounded by the number of integral points inside the Minkowski sum of successive projections of the Newton polytopes of polynomials. Thus the size of the Dixon matrix is much smaller than the size of Macaulay matrix; further, the size of the Dixon matrix is also smaller (by an exponential factor) than the size of sparse resultant matrix, which is bounded by the number of integral points inside the Minkowski sum of the Newton polytopes (not their successive projections) of the polynomials. It is proved that even though entries in a Dixon matrix are somewhat complicated in contrast to entries in the Macaulay matrix and sparse resultant matrix where the entries are either 0 or coefficients of terms in polynomials, the Dixon matrix can be constructed fast using dense interpolation techniques. These results serve as a theoretical justification for the superiority of the method based on Dixon formulation in practice over other methods.

[3] A.L. Dixon, "The eliminant of three quantics in two independent variables," *Proc. London Mathematical Society,* 6, 1908, 468-478.

3.2 Branch and Prune Approach for Numerically Solving Polynomial Constraints

In collaboration with David McAllester and Pascal Van Hentenryck, we have developed a branch and prune based approach for numerically solving polynomial constraints (equalities as well as inequalities) using interval computations [19]. The approach can be viewed as a global search method which uses interval computations for numerical accuracy and pruning search space using local methods. As a first approximation, each polynomial is successively viewed as a univariate polynomial in each variable with interval coefficients, and the interval for each variable is narrowed to ensure that there is a solution on the boundary of the interval for each variable. When an interval for a variable cannot be narrowed using unary interval constraints, then branching is performed by splitting the interval into equal halves, and this process of narrowing the intervals is repeated until no solution can be found within an interval or a sufficiently small interval (determined by the desired level of accuracy) is identified.

The approach can be parameterized by different transformations for pruning the intervals. The Newton system currently uses three such transformations: (i) natural formulation of the constraints viewed as functions of a single variable by fixing the values of other variables as intervals available as current guesses, (ii) distributed formulation in which a normal form of a constraint is used, and (iii) finally for convergence and accuracy near a solution, Taylor series expansion of a constraint around the center of each interval. The first two transformations quickly prune the interval when it is too large and thus far away from a solution, whereas the third transformation is helpful in convergence and avoiding generation of small intervals around solutions.

The approach is quite simple, both from the viewpoints of mathematics and programming. It has been implemented and tried with remarkable success, on many benchmark examples from different application domains including robotics, kinematics, engineering, vision and chemistry.

3.3 Solving Parametric Polynomial Equations

We have developed a simple but powerful approach for solving a parametric system of polynomial equations (in which parameters are distinguished from variables) [9]. Gröbner basis algorithm and characteristic set triangulation algorithm have been extended to generate a (finite) family of solved forms. Each solved form can be used to generate solutions of parametric systems for a subset of parameter values expressed as a finite set of constraints. The solution sets are classified for different parameter values depending on whether there are no solutions, finitely many solutions, or infinitely many solutions. The extension is based on a constraints paradigm, in which parametric relations are viewed as constraints. The algorithms incrementally partition the parameter space and provide useful structural information about solutions of a nonlinear polynomial system for different ranges of parameter values.

An understanding of the solutions of a parametric system is useful in a wide variety of applications including computer vision, computer aided design and modeling. For different parameter subspaces, the solution set may behave differently because of which different

optimizations may be possible. The constraint-based approach for parametric systems may be useful for solving systems of polynomials with floating point coefficients using symbolic techniques. Algorithms for solving parameterized systems would be particularly relevant for constructing quasi-invariants where a goal is to construct rational functions of object parameters which do not vary locally for a large range of imaging parameters.

3.4 Other results

A method for converting a Gröbner basis with respect to a degree ordering to a Gröbner basis with respect to lexicographic order has been developed for positive dimension ideals using linear algebra techniques [12]. A main reason for interest in such a basis conversion algorithm is that Gröbner basis computation is much faster using the degree term ordering in contrast to lexicographic term ordering. A lexicographic Gröbner basis is, however, a lot more useful in gaining insight into the structure of zeros of a polynomial system.

We resurrected Ritt's concept of characteristic sets [8] and demonstrated that it is quite different from the concept of characteristic set that Wu has popularized in the context of geometry theorem proving as well as for equation solving. We proved many interesting characterization theorems for systems of polynomials to be characteristic sets. The applications of this approach for solving equations as well as computing structure of the zero sets of polynomials have been investigated.

Results about basis conversion as well as characteristic set conversion have been implemented in our experimental software GEOMETER, a system for algebraic and symbolic manipulation. Implementations of the Gröbner basis algorithm as well as characteristic set construction algorithm in GEOMETER run much more efficiently than the implementations of these algorithms in commercially available computer algebra systems including Maple, Mathematica, and Macsyma. We have paid considerable attention to efficiently coding the primitive operations in GEOMETER.

Distributed implementations of homotopy techniques based on continuation for numerically solving equations have also been experimented with.[4] Whereas conventional homotopy methods use Bezout bound for constructing an initial system whose solutions are traced to find solutions of a given system, the homotopy discussed by Verschelde et al used Bernstein bound which is dependent on the Newton polytopes of the polynomials in the system and is, thus, known to be much smaller for sparse polynomial systems. Our experience with the homotopy method based on Newton polytopes was not very positive; the method was found to be inefficient and slow since the construction of an initial system is recursive and quite complicated. We have instead used a homotopy method discussed by Morgan on a number of examples including examples from computer vision, invariants and robotics. The distributed implementation runs concurrently on a network of Sparc work-stations since computations for different paths corresponding to different solutions can proceed concurrently.

[4] A.P. Morgan, *Solving Polynomial Systems using Continuation for Scientific and Engineering Problems,* Prentice Hall, NJ, 1987. J. Verschelde, P. Verlinden, and R. Cools, "Homotopies exploiting Newton polytopes for solving sparse polynomial systems," *SIAM J. of Numerical Analysis,* 31(3), 915-930, 1994.

4 Educational Impact

One undergraduate student, Brian Levine, four graduate students – Tushar Saxena, Qing Guo, N. Sundaram, Sreenivasa Viswanadha, and two post doctorals – Dr. Xumin Nie and Dr. Mohsin Ahmed, were partially supported by this contract. The work on Dixon resultants supported under this grant will serve as a basis for Saxena's Ph.D. dissertation. A graduate level course on reasoning uses considerable material based on research in symbolic computation done for this contract. Another graduate level on advanced computer vision is based on results on geometric invariances and indexing techniques for model-based object recognition.

A number of papers have been presented at conferences and published in journals. Many invited presentations have been given at various conferences, as well as in academic institutions and research laboratories. A paper by Rothwell, Forsyth, Mundy and Zisserman got the *Marr Prize*, the most prestigious award in computer vision. Forsyth won the NSF Young Investigator Award. Kapur has been invited to give a tutorial on algorithmic elimination methods at *International Conference on Symbolic and Algebraic Computation (ISSAC)* to be held in Montreal in July 1995.

We participated in a seminar *on Invariance in Object Recognition for DOD Applications* at Wright Paterson Lab, Nov. 18-19, 1992. On the first day, Kapur and Mundy gave a tutorial on invariant theory and the use of invariants in image understanding. On the second day, Binford and Levitt of Stanford University discussed quasi-invariants and their use in object recognition. The seminar was attended by over 50 researchers from industry, government laboratories and universities. Interactions at the seminar led to an initiative based on thermal and geometric invariants for automatic target recognition. The discussion at the seminar about quasi-invariants led us to investigate a theoretical framework for quasi-invariants and relate them to invariants, which resulted in [1].

We have begun interacting with Dr. Robert Williams' group in Mission Avionics Technology Department at Naval Air Development Center at Warminster, PA. We have been discussing Dixon resultants and exploring its use for trajectory problems. Their group has been using our algorithms based on the Dixon formulation in their work.[5]

[5]McShane, Nakos and Williams, "The Kapur-Saxena-Yang Dixon Resultant with Maple and Mathematica," presented at *First Intl. IMACS Conf. on Applications of Computer Algebra,* Albuquerque, New Mexico, May 1995.

5 Publications citing the Contract

Below we list all the papers appearing in conferences, books and journals citing the contract. A copy each of these papers is included in the Appendix.

References

[1] T. Binford, D. Kapur, J.L. Mundy, "Relationship between Invariants and Quasi-invariants," Proc. *Asian Conference on Computer Vision,* Nov. 22-25, Osaka, Japan.

[2] B. Donald, D. Kapur, and J.L. Mundy (eds.) *Symbolic and Numerical Computation for Artificial Intelligence.* Academic Press, 1992.

[3] D.A. Forsyth, "Recognizing Algebraic Surfaces from their Outlines,"*International Conference on Computer Vision*, Berlin, 1993, as well as *Fifth IMA Conference on the Mathematics of Surfaces, Edinburgh,* Sept. 1992. An expanded version has been accepted for publication in *International 1Journal of Computer Vision.*

[4] D.A. Forsyth, J.L. Mundy, A. Zisserman, and C.A. Rothwell, "Using Global Consistency to Recognize Euclidean Objects with an Uncalibrated Camera," Proc. *Computer Vision and Pattern Recognition,* 1994.

[5] D.A. Forsyth and C.A. Rothwell, "Recovering extruded surfaces from their outlines," Proc. *Allerton Conference on Communication, Control, and Computing*, 1993.

[6] D.A. Forsyth and C.A. Rothwell, "Representations of 3 D objects that incorporate surface markings," Mundy J.L., Zisserman A., and Forsyth D.A., editors, *Applications of Invariance in Computer Vision*, LNCS 825, Springer-Verlag, 1994.

[7] D.A. Forsyth, A. Zisserman, and J. Malik, "Distinctive representations for the recognition of curved surfaces using outlines and markings," Proceedings *NSF Workshop on Representation in Computer Vision,* Springer Verlag, LNCS, 1995.

[8] D. Kapur, "Towards a Theory of Characteristic Sets" invited talk given at the *First International Workshop on Mechanization of Mathematics,* Chinese Academy of Sciences, Beijing, China, July 16-18, 1992. Draft Manuscript, Institute for Programming and Logics, State University of New York, Albany, NY 12222.

[9] D. Kapur, "An approach for solving systems of parametric polynomial equations," to appear in *Principles and Practices of Constraint Programming.* (eds. Saraswat and Van Hentenryck), MIT Press, 1995.

[10] D. Kapur and Lakshman Y.N, "Elimination Methods: An Introduction," in *Symbolic and Numerical Computation for Artificial Intelligence,* Donald, Kapur and Mundy (eds.), Academic Press, 1992.

[11] D. Kapur and X. Nie, "Reasoning about numbers in Tecton", Proceedings of the *8th International Symposium on Methodologies for Intelligent Systems, (ISMIS'94)*, Charlotte, North Carolina, October 1994, 57-70.

[12] D. Kapur and T. Saxena, "An Algorithm for Converting a Degree Gröbner Basis to Lexicographic Gröbner Basis" Draft Manuscript, Institute of Programming and Logics, State University of New York, Albany, NY, Oct. 1993. Presented at *East Coast Computer Algebra Day (ECCAD),* May 1994, Drexel Univ., Philadelphia.

[13] D. Kapur and T. Saxena, "Comparison of various multivariate resultant formulations," to appear in Proceedings of *Intl. Symp. on Symbolic and Algebraic Computation (ISSAC-95),* Montreal, Canada, July 1995.

[14] D. Kapur and T. Saxena, *Sparsity Considerations in Dixon Resultants.* Department of Computer Science, State University of New York at Albany, April 1995. Submitted to FOCS'95.

[15] D. Kapur, T. Saxena and L. Yang, "Algebraic and Geometric Reasoning using Dixon Resultants," Proceedings of *Intl. Symp. on Symbolic and Algebraic Computation (ISSAC-94),* Oxford, England, July 1994, 99-107.

[16] N. Pillow, S. Utcke and A. Zisserman, "Viewpoint-Invariant Representation of Generalized Cylinders Using the Symmetry Set," *Proc. British Machine Vision Conference,* 1994, 539-548. To appear in *Image and Vision Computing.*

[17] C.A. Rothwell, A. Zisserman, D.A. Forsyth, and J.L. Mundy, "Planar Object Recognition using Projective Shape Representation." To appear in *IJCV.*

[18] C.A. Rothwell, D.A. Forsyth, A. Zisserman and J.L. Mundy, "Invariants of Three Dimensional Point Sets from Single Images," Submitted to *IEEE PAMI.*

[19] P. Van Hentenryck, D. McAllester, and D. Kapur, "Solving polynomial systems using a branch and prune approach," to appear in *SIAM J. on Numerical Analysis.*

[20] A. Zisserman, D.A. Forsyth, J.L. Mundy, C.A. Rothwell and J. Liu, "3D object Recognition using Invariance," to appear in a special issue of the *Artificial Intelligence Journal* on Computer Vision.

[21] A. Zisserman, R. Hartley, J.L. Mundy and P. Beardsley, "Is Epipolar Geometry Necessary to Recover Invariants from Multiple Views?" Draft Manuscript, Robotics Research Group, Oxford University, Oxford, England.

Appendix

The Relationship Between Invariants and Quasi-Invariants*

T.O. Binford
Stanford University
Stanford CA

D. Kapur
State University of New York
Albany, NY

J.L. Mundy
General Electric CRD
Schenectady, NY

Abstract

Invariants have recently been found useful for object recognition and indexing into model-based vision systems. For many geometric configurations, invariants may not exist. Quasi-invariants are properties which need not be invariants for a whole set of imaging transformations, but are invariant locally. A framework for studying quasi-invariants is proposed, and is used to study relationships between invariants and quasi-invariants.

1 Introduction

There has recently been considerable interest in studying invariants and related properties of geometric configurations for object recognition and indexing into model-based vision systems. The main attraction of geometric invariants for computer vision is their independence of camera calibration and camera viewpoint. Thus, a data base of objects can be efficiently organized and indexed in terms of invariant values. Since the invariant index does not depend on viewpoint, the object description can be obtained from a single view of the object without manual model construction.

For a discussion of recent developments in the application of geometric invariance in vision, a reader may consult [4]. Fortunately, geometric invariants have been extensively studied by mathematicians and constructive algebraicists at least for last two centuries. Invariants are mathematically associated with a group of transformations, and invariants for different groups of planar transformations - Euclidean, orthographic, equi-form, affine, and projective, topological, have been investigated. Invariants for 3D surface classes have also been developed such as surfaces of revolution, symmetrical objects and algebraic surfaces.

Perspective transformations(central projection) do not constitute a group. The composition of two planar perspectivities is not necessarily a perspectivity. On the other hand, the theory of invariants is most fully developed for the case of group transformations. Perspective transformations are a subset of projective transformations which do constitute a group. Therefore projective invariants can be used to index a model database for recognition under perspective viewing. However, geometric configurations must be of sufficient complexity to support projective invariants. For example, projective invariants cannot be derived for 3 collinear points, 4 coplanar points, etc.

Quasi-invariants have been proposed by Binford [3] as an alternative to the strict requirements of invariants. The idea here is to identify properties which need not be invariant for the whole set of perspective transformations, but are invariant locally and remain reasonably constant over the entire view space. The advantage is that many more quasi-invariants are available for a given set of geometric features than projective invariants.

A goal of this work is to understand the relationship between quasi-invariants and invariants and to see whether invariants and quasi-invariants can be understood in a single framework. This will enable us to use invariants whenever they exist and are easy to compute, and augment them by quasi-invariants otherwise.

Given that quasi-invariants vary over imaging transformations, one obvious question is to analyze their variability over a practical range of view points. We are interested in studying the variability of quasi-invariants over group actions and classify quasi-invariants based on their group categories. While this is not a general way to analyze quasi-invariants we will demonstrate that many useful insights can be gained from the group invariant viewpoint.

2 Definitions

2.1 Invariants

The most well-known invariant for projective transformation is that of the cross-ratio for 4 collinear points (which is the ratio of length ratios). Five coplanar points which is a generalization of 4 collinear points to 2 dimensions, also have two projective invariants. Three points define a plane, so the ratio of area ratios of various triangles defined by 5 points,

$$\frac{\frac{Area(P_1,P_2,P_4)}{Area(P_1,P_3,P_4)}}{\frac{Area(P_1,P_2,P_5)}{Area(P_1,P_3,P_5)}},$$

is a projective invariant. A second, independent ratio can be constructed from another permutation of point

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

grouping. The concept of an invariant can be defined more generally.

Given a group G of transformations, a property, say I, expressed in terms of parameters P of a geometric configuration, is said to be invariant with respect to G if

$$I(P) = f(T) \;\; I(T(P)), \text{for any } T \in G,$$

where T is a transformation on a geometric configuration, $T(P)$ are the parameters of the transformed geometric configuration under T, and $f(T)$ is a function on the transformation parameters. For absolute invariants, $f(T)$ can be made 1.

2.2 Quasi-invariants

Four coplanar points do not have any projective invariant since any set of four points can be mapped onto any other set of four points by a projective transformation. It is however possible to associate quasi-invariants for 4 coplanar points as discussed by Binford and Levitt [3]. For example, the ratio of two areas is a quasi-invariant under the projective transformation between two planes, i.e.

$$\frac{Area(P_1, P_2, P_4)}{Area(P_1, P_2, P_3)}$$

A quasi-invariant can be defined more generally as follows.

Like an invariant, a quasi-invariant, QI is a property expressed in terms of parameters P of a geometric configuration. A property, QI, is *locally invariant* with respect to S, a set of transformations, at some distinguished element s in S, if for each transformation parameter t_i, the first derivative of QI with respect to t_i is 0 at s. The set S need not be a group, but all known quasi-invariants are defined with respect to group invariants.

A quasi-invariant is thus defined in terms of three elements,

- a geometric configuration and associated property of the configuration,
- a distinguished set of image transformations,
- a specific transformation in the set of image transformations.

In principle, there can be configurations and properties which are quasi-invariant at different points in the transformation group. However, the known quasi-invariants are locally invariant when the image plane is parallel to the geometric configuration and the image view point is at infinity.

A quasi-invariant QI is called *strong* at a distinguished element s in S if for each transformation parameter t_i, in addition to the first derivatives being 0, the second derivatives of QI with respect to t_i also vanish at s.

It is possible to further refine quasi-invariants by requiring that their first k derivatives with respect to transformation parameters be 0 at a distinguished element, for $k \geq 2$.

As a simple example, consider a point in 3-D; the coordinates of an image of this point under perspective viewing can be written as:

$$\frac{f\ x\ cos\theta}{z - x\ sin\theta},$$

where z is the depth and θ is rotation around y-axis. If the ratio of distances of two points equi-distant ($\pm r$) from the origin is computed, an affine invariant

$$\frac{z - r\ sin\theta}{z + r\ sin\theta}$$

is obtained. The derivatives of the above expression with respect to z and θ both vanish at $z = \infty$ provided r is finite (or $r << z$). As the reader can easily verify, the above expression is not locally invariant for values of z which are comparable to r. This is an example of a quasi-invariant that is locally invariant at $\theta = 0$ or $z = \infty$, but not at other values of z. In particular, the geometry of the object class (in the above example, r's value in comparison to z) may play a role.

3 A Group Framework for Quasi-Invariants

The above discussion suggests an alternate definition of quasi-invariants which can perhaps be useful for studying variability in quasi-invariants with respect to transformation parameters. Admittedly, this definition may rule out some quasi-invariants developed using Binford and Levitt's definition since there can be functions which have vanishing derivatives at some point in the group, but are not invariants of any subgroup.

All the examples of quasi-invariants discussed in Binford and Levitt [3] can be studied using a framework based on group invariants. First we observe that an invariant does not depend upon any transformation parameter, its first derivative with respect to any transformation parameter is 0. And, this is the case for every element in G.

Theorem: Given a group G of transformations, an invariant I is a (strong) quasi-invariant for G at every element in G.

3.1 Group Quasi-invariants

A key relationship between invariants and quasi-invariants is illustrated by the following observation.

Assume a group G of transformations which can be parameterized using a set T of parameters as well

as its various subgroups can be parameterized using subsets of T. Given two nontrivial subgroups G_i, G_j of G with transformation parameters T_i, T_j, such that $G_i \subset G_j, T_i \subset T_j$, a property QI is quasi-invariant at a distinguished point $g \in G_j$ for G_j if it is an invariant for G_i and for every additional parameter $t_j \in T_j - T_i$, the first derivative of QI with respect to t_j is 0 at g. Invariants defined in this way are called *group quasi-invariants.*

For perspective viewing there is a natural chain of group transformations, *equiform* $\subset$ *affine* $\subset$ *projective*, where each is more general (more group parameters) than the previous. In general there will be a hierarchy of transformations where

$$G_1 \subset G_2 ... \subset G_n.$$

4 Quantifying Quasi-invariant Variation

4.1 Variation Based on Derivatives

We can provide a basis for comparing quasi-invariant variation over the transformation group as follows.

Given two nontrivial subgroups G_i, G_j of G, and a distinguished element $g \in G_j$, a quasi-invariant QI *varies at least as much as, up to* k^{th} *order,* QI' if for every $k' \leq k$, the absolute value of every k'^{th} derivative QI with respect to transformation parameters $T_j - T_i$ is $\geq$ the absolute value of the corresponding k'^{th} derivative of QI'. We also say that QI' varies no more than QI, up to k^{th} order.

Theorem: Given a group G of transformations and $G_j \subset G$, for any k, an invariant of G_j varies no more than any quasi-invariant QI' of G_j, up to k^{th} order, at any element of G_j.

Theorem: Given a group G of transformations and $G_j \subset G$, a strong quasi-invariant of G_j varies no more than any other quasi-invariant QI' of G_j up to 2nd order at distinguished element $g \in G_j$.

It is also possible to compare quasi-invariants associated with two different pairs of subgroups for their applicability and variability. Given quasi-invariants QI with an associated pair of subgroups (G_i, G_j) and another QI' with an associated pair of subgroups (G'_i, G'_j) such that G_i is a subgroup of G'_i, and G'_j is a subgroup of G_j, QI *varies at least as much as* QI' at a distinguished element $g \in G'_j$ if the absolute value of every k^{th} derivative of QI with respect to parameters in $T'_j - T_i$ is $\geq$ the absolute value of the corresponding k^{th} derivative of QI'. Typically the larger subgroups G_j, G'_j are the same, and they are G itself, which is usually the group of projective transformations.

4.2 The Group Hierarchy Bound

For a given geometric configuration, appropriate quasi-invariants can be compared over the entire space of image viewpoints. If a quasi-invariant is an invariant of a more restricted group, its variation is likely to be larger over full set of image transformations than a quasi-invariant based on a more general group. For example, the angle between two lines varies more than the ratio of two lengths on a line which in turn varies more than the cross-ratio since these are invariants of a properly contained group hierarchy with respect to perspective.

Consider a configuration consisting of four equidistance collinear points (P_1, P_2, P_3, P_4) with another point, say P_5, not on the line but is equidistant from P_2 and P_3. It can be shown that cross-ratio $\frac{\frac{|P_4P_2|}{|P_4P_1|}}{\frac{|P_3P_2|}{|P_3P_1|}}$ is invariant with respect to perspective, but the collinear length ratio, say $\frac{|P_4P_2|}{|P_4P_1|}$, varies. However this length ratio varies less than the distance ratio $\frac{|P_5P_1|}{|P_5P_4|}$ or $\frac{|P_5P_2|}{|P_5P_3|}$. The variability depends upon (i) the distance between the camera and geometric configuration, and (ii) the coordinates of P_2 and P_5.

We believe that it should be possible to perform a global comparison of appropriate quasi-invariants by comparing their associated subgroups for which they are invariants. Since group invariants constitute an algebra (i.e. any rational function of invariants is also an invariant), a canonical form of group invariants may be needed in terms of some basic invariants so that basic invariants can be compared for variability. This aspect of variability of quasi-invariants is being currently investigated.

References

[1] S.A. Abhyankar, "Invariant theory and enumerative combinatorics of Young tableaux," in *Geometric Invariance in Computer Vision,* (eds. Mundy and Zisserman), MIT Press, 1992, 45-76.

[2] E.B. Barrett, P.M. Payton, N.N. Haag, and M.H. Brill, "General methods for determining projective invariants in imagery," *CVGIP: Image Understanding,* **53**, 1991, 46-65.

[3] T. Binford and T.S. Levitt, "Quasi-Invariants: Theory and exploitation, " *Proc. DARPA workshop on Image Understanding,* Washington, D.C., 1993.

[4] J.L. Mundy and A. Zisserman (eds), *Geometric Invariance in Computer Vision,* MIT Press, 1992.

[5] J.L. Mundy, P.M. Payton, M.H. Brill, E.B. Barrett, and R.P. Welty, "3-D Model alignment without computing pose," *Proc. DARPA workshop on Image Understanding,* 1992, 727-735.

Recognizing Algebraic Surfaces from their Outlines

D. A. Forsyth
Computer Science Division,
University of California at Berkeley,
Berkeley,
CA 94720

January 5, 1995

Abstract

The outline in a single picture of a generic algebraic surface of degree three or greater completely determines the projective geometry of the surface. The result holds for a generic perspective view of a generic algebraic surface, where the camera calibration parameters and the focal point are unknown. Known camera calibration appears not to reduce the projective ambiguity. The result is constructive. **Keywords:** Recognition, Computer Vision, Algebraic Surfaces, Invariant Theory, Outlines.

1 Introduction

Outlines, the points in an image where a surface turns away from the camera, are a potentially important source of information about the objects in a scene. Typically, image edges appear at most outline points, and image edges can be computed reasonably reliably. This potential has not been realised in the case of curved surfaces, because the complicated relationship between the outline of a curved surface and the surface makes outlines hard to interpret. This paper shows that, although the relationship between surface and outline is complicated, for a large class of surfaces the outline is sufficiently highly structured to determine the surface's projective geometry from a single view.

1.1 Recognising curved surfaces

There have been many approaches to recovering shape information for curved surfaces from images, including attempts to extend line labelling to curved shapes (e.g. [10, 20]), the development of constraint-based systems (e.g. [3]), the study of how the topology of a surface's outline changes as it is viewed from different points, formalised into a structure known as an *aspect graph* (for example, [16, 17, 26, 33, 34]), and studies of the relationship between the differential geometry of the outline and that of the surface, both for single images (e.g. [17, 18, 21]) and for motion sequences (e.g. [4, 11]).

Each approach has characteristic disadvantages; extensions to line labelling and aspect graphs can be extremely complicated for even simple curved surfaces (examples in [27, 33, 34]), constraint-based systems must search a model-base, and studies of differential properties seldom yield sufficient information to identify a surface. Recently, there have been attempts to represent the system of outlines of a curved surface as a linear combination of outlines (see, for example, [42]). This approach is represented as providing an approximation sufficiently accurate for some purposes, although it cannot capture all the complexities of the outline. There are two main difficulties: it is hard to specify correspondences between outline points, so that the linear combination is ill-defined; and, because the scheme is based on a local approximation, it cannot capture the global interactions between a surface and a focal point that produce the outline. However, the approximation can yield plausible outlines when views are taken from similar viewing positions.

Recovering surface geometry from a single outline is intractable if the surface is constrained only to be smooth or piecewise smooth, because significant changes can be made to the surface geometry without affecting the outline from a given viewpoint. As a result, an important part of the problem involves constructing as large a class of surfaces as possible that can either be directly recognised or usefully constrained, from their outline alone. In this context, studies have focused on rotationally symmetric surfaces (for example, [6, 8]), and straight homogenous generalised cylinders (for example, [2, 28, 43, 44, 47]).

Recently, Ponce and Kriegman [29, 30, 31, 32, 33] focussed attention on *algebraic surfaces*. Algebraic surfaces, which consist of all the points in space where a single polynomial vanishes, have numerous advantages as objects of study:

- Many man-made surfaces are made up of "patches" of algebraic surface, as most popular CAD/CAM surfaces are algebraic.
- The geometry of an algebraic surface is determined by a relatively small number of parameters (the coefficients of the polynomial that gives the surface). At the same

time, the surfaces have a rich and useful geometry.

- Algebraic surfaces have important "rigidity" properties. For example, one cannot add a local bump to an algebraic surface and obtain another algebraic surface; the whole surface must be deformed instead.

Ponce and Kriegman showed that elimination theory can be used to predict the outline of an algebraic surface viewed from an arbitrary viewing position. For a given surface a viewing position is then chosen using an iterative technique, to give a curve most like the curve observed. The object is then recognized by searching a database, and selecting the member giving the best fit to the observed outline. This work shows that outline curves strongly constrain the viewed surface, but has the disadvantage that it cannot recover surface parameters without solving an optimization problem, so that for a big model base, each model may have to be tested in turn against the image outline. Furthermore, camera parameters must be known to predict the outline correctly.

1.2 Indexing for recognition

A number of recent papers have shown how *indexing* can be used to avoid searching a model base (e.g. [7, 19, 35, 39, 45]). Objects are indexed by computing descriptions that are unaffected by the position and intrinsic parameters of the camera, and that differ from object to object. These descriptions, often known as *indexing functions*, have the same value for any view of a given object, and so can be used to index into a model base without search.

In a typical system that works for plane objects, projective invariants[1] are computed for a range of geometric primitives in the image. If the values of these invariants match the values of the invariants for a known model, we have good evidence that the image features are within a camera transformation of the model features. As a result, these invariants index into a model base directly. Object models consist of a system of invariant values and are therefore relatively sparse, meaning that hypothesis verification is required to confirm a model match. However, no searching of the model base is required because the hypothesised object's identity is determined by the invariant descriptors measured. Systems of this sort have been demonstrated for plane objects in a number of papers [7, 19, 36, 40, 46, 48]. These systems are attractive because, in the ideal case, an object description is computed from the image and identifies the object, without requiring that a model base be searched. As a result, systems with relatively large model bases can be constructed[2].

In the case of plane objects, indexing functions are easy to compute, because a view of a plane curve from an arbitrary focal point is within a projective transformation of the original curve. Constructing indexing functions for three dimensional objects is challenging, because a change in viewing position can lead to a profound change in the geometry of the outline. Furthermore, any indexing function should be both *invariant* and *computable from outline information alone.* Indexing functions with these properties have been demonstrated for polyhedra [37], and for rotationally symmetric surfaces [8].

This paper shows that such indexing functions can be computed for algebraic surfaces viewed in perspective using an uncalibrated camera, by establishing:

[1] A clear introduction to applying invariant theory in computer vision appears in [22].

[2] Current systems using indexing functions have model-bases containing of the order of thirty objects.

Theorem: *The equation of its outline in a perspective camera completely determines the projective geometry of an algebraic surface of degree 2 or greater, for a generic view of a generic algebraic surface.*

Here generic means "almost every"; precisely, the generic algebraic surfaces are all algebraic surfaces except those whose coefficients satisfy a non-trivial system of algebraic relations, to be determined later. At this point, we assume that the surface is smooth and irreducible. The result is similar to that independently obtained by [5], who demonstrated necessary and sufficient conditions for a curve to be an outline, but did not show that an outline determines a surface. Note that the theorem is trival for surfaces of degree two, as generic surfaces of degree 2 are all projectively equivalent. The main result is given by the following two properties of outlines:

- The contour generator of a generic algebraic surface is determined as a *space curve* uniquely (up to a projectivity of space), by a generic projection of the curve on to a plane. In particular, the outline of an algebraic surface in a given view contains sufficient information to compute both the contour generator of that surface and the focal point through which the contour generator was formed, together in some arbitrary projective frame.
- Given the contour generator of a generic surface viewed from a generic focal point, and that focal point, the surface can be uniquely determined.

2 The outline of an algebraic surface

Throughout the paper, we assume an idealised pinhole camera. These cameras possess a *focal point* and an *image plane.* Points in space appear in the image as the intersection between the image plane and a line through the focal point and the point in space. An orthographic view occurs when the focal point is "at infinity".

It is easy to see that if the focal point is fixed and the image plane is moved, the resulting distortion of the image is a *collineation*, a one-to-one map of the projective plane to itself that takes straight lines to straight lines. In what follows, it is assumed that neither the position of the image plane with respect to the focal point nor the size and aspect ratio of the pixels on the camera plane is known, so that the image presented to the algorithm is within some arbitrary collineation of the "correct" image. In this model, the image plane makes no contribution to the geometry, and its position in space is ignored.

The *outline* of a surface is a plane curve in the image, which itself is the projection of a space curve, known as a *contour generator*[3]. The contour generator is given by those points on the surface where the surface turns away from the image plane; formally, the ray through the focal point to the surface is tangent to the surface. As a result, at an outline point, if the relevant surface patch is visible, nearby pixels in the image will see vastly different points on the surface, and so outline points usually have sharp changes in image brightness associated with them. Figure 1 illustrates these concepts.

We shall study generic algebraic surfaces in projective space, otherwise written as P^3. A point in P^3 is given by four homogenous coordinates, where two sets of homogenous coordinates refer to the same point if they are within a scalar multiple of one another (Appendix

[3]There are a number of widely used terms for both curves, and no standard terminology has yet emerged.

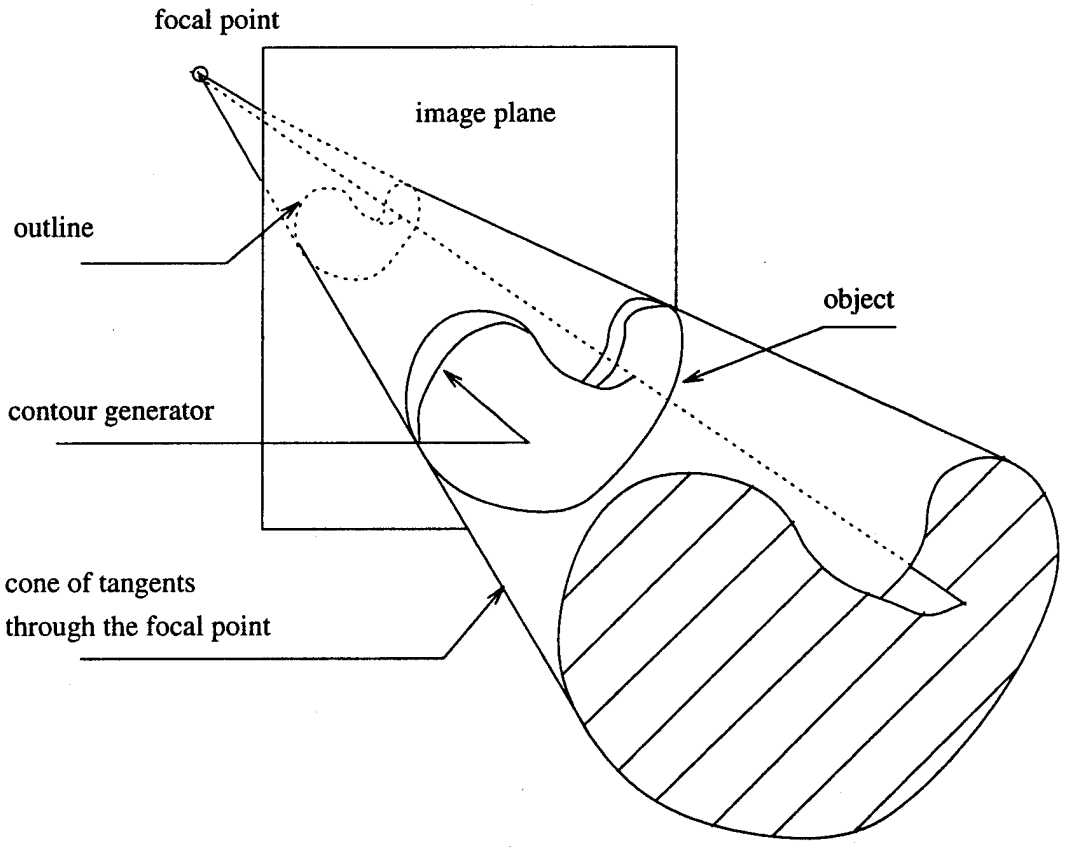


Figure 1: The outline and contour generator of a curved object, viewed from a perspective camera.

1 in [23] contains examples and discussions of the practical applications of homogenous coordinates). Projective three-space is similar to the three dimensional space in which we live, but contains provision for a plane of infinitely distant points as well. In P^3, an algebraic surface is given by the vanishing of a single homogenous polynomial in these four coordinates. We will assume that the camera has an infinite film plane as well, so that the image plane can be modelled by P^2, the projective plane. The points in space that project to points "at infinity" in the camera film plane lie on a plane parallel to the image plane and passing through the focal point.

We use the following notation:

- (u_0, u_1, u_2, u_3) are the coordinates of a point in P^3.
- (x_0, x_1, x_2) are the coordinates of a point in P^2, the image plane.
- (f_0, f_1, f_2, f_3) is the camera focal point.
- $S(u_0, u_1, u_2, u_3)$ is the homogenous polynomial that vanishes on the surface.
- d is the degree of S.

Since the surface is generic, it is irreducible, and so $S(u_0, u_1, u_2, u_3)$ does not factor. The contour generator lies on the surface, and so $S(u_0, u_1, u_2, u_3) = 0$ on the contour generator. The plane tangent to the surface at a point on the contour generator must pass through the focal point, by the definition of the contour generator. As a result, the expression

$$f_0 \frac{\partial S}{\partial u_0} + f_1 \frac{\partial S}{\partial u_1} + f_2 \frac{\partial S}{\partial u_2} + f_3 \frac{\partial S}{\partial u_3}$$

vanishes on the contour generator. This expression will be called T for short in what follows. We have immediately that:

- The contour generator is an algebraic space curve given by the vanishing of just two polynomials[4], S and T, and so is a *complete intersection.* Furthermore, for a generic choice of the focal point, T has degree $d-1$, and so the contour generator has degree $d(d-1)$. The surface given by $T = 0$ is known as the first polar of S.
- The *family* of contour generators on a surface is a family of curves *linearly* parametrised by focal points alone. Such families are known to algebraic geometers as linear systems, and are widely studied. The study of such systems makes general statements about contour generators possible. For example, a generic contour generator on a generic surface is smooth, by Bertini's theorem[5]. In fact, the contour generator is a plane section of the dual of the surface, where the sectioning plane depends on the focal point chosen. The simplicity of the system of contour generators stands in stark contrast to the complexity displayed by the family of outlines, as the focal point changes, information conventionally captured by an aspect graph.

[4]Some algebraic curves in space must be given by more than two equations - see, for example [24]

[5]A generic element of a linear system is smooth away from its base points (e.g. [13]); note that for a smooth surface, the system of contour generators has no base points, but if the surface is singular, the singularities of the surface are base points, and the contour generator may be singular.

- Contour generators are projectively covariant; that is, for a surface S, viewed from a focal point f, with contour generator C, if P is an arbitrary projectivity of space, then $P(C)$ is the contour generator of $P(S)$ viewed from $P(f)$. This is because the contour generator is defined by tangency and incidence conditions alone.

It is important to note that the treatment that follows assumes that the complex points of both the algebraic curve and the algebraic surface are meaningful. For example, when a count is given of the number of singular points on the outline of a given type, that count includes the complex singularities. It is conceivable that an algebraic surface could have an outline that consisted entirely of complex points; a natural example is the outline of a sphere viewed from a focal point lying inside the sphere. In this case, while in principle the outline constrains the surface just as effectively as if it had a large collection of real points, in practice the outline is difficult to observe. Another effect that can make the outline difficult to observe is self-occlusion, where sections of the outline are occluded by the surface and so are, in practice, invisible. Self occlusion is difficult to study given the methods here; visibility can change only at singularities, however.

This is why the statement of the theorem emphasizes the equation of the outline. In practice, if the outline has sufficient visible real points that a fitting process can determine its equation from the real points alone, its complex points and singularities follow. In principle, a fitting process should be robust to occlusions, as for irreducible algebraic curves (the genericity assumptions assure that the outlines covered in this paper are irreducible), only a finite number of points is necessary to determine the equation of the curve. This means that, to apply the result, we must assume that the view yields enough real points on the outline to determine its equation; this is not a particularly strong restriction in principle.

2.1 The singularities of the outline

Since the contour generator of a non-singular algebraic surface viewed through a generic focal point is smooth, the singularities of the outline must be a result of the projection from the contour generator to the outline. These singularities are the key to obtaining the contour generator from the outline; fortunately, they are highly structured. Generically there are only cusps and nodes. The following results have been known since at least the late 19th century (see, for example, [1]).

2.1.1 Cusps

A cusp in the outline is a local event on the contour generator, so that cusps are relatively easily studied; a cusp occurs when the contour generator is tangent to the ray through the focal point (see, for example, [16] for this widely known result).

Lemma 1: *Cusps in the outline are the projections of points on the contour generator where the second polar of the surface through the focal point vanishes; accordingly, there are* $d(d-1)(d-2)$ *cusps in the outline of a surface of degree* d.

> **Proof:** If p is a point on the contour generator that projects to a cusp on the outline, and f is the focal point, then the line pf is tangent to the contour generator at p. The line tangent to the contour generator at p is given by the intersection of the plane tangent to the surface at p and the plane tangent to

the first polar at p. Because this line passes through f, the plane tangent to the first polar at p must pass through f as well. Recall that the surface was written as S and the first polar was written as T. We have that the expression:

$$f_0\frac{\partial T}{\partial u_0} + f_1\frac{\partial T}{\partial u_1} + f_2\frac{\partial T}{\partial u_2} + f_3\frac{\partial T}{\partial u_3} = 0$$

must vanish at p. This expression, which is the first polar of T through f, is also known as the second polar of S through f, and has degree $d-2$ if d is the degree of S; call this expression P for convenience.

In turn, if S, T and P vanish at a point p, then the point is on the surface and on the contour generator by definition; furthermore, the plane tangent to the surface at p passes through f, and the plane tangent to T at p passes through f, so their intersection, which is tangent to the contour generator, passes through f. Thus, the contour generator cusps at exactly those points where S, T and P vanish; by Bézout's theorem there are $d(d-1)(d-2)$ such points, and so the outline has $d(d-1)(d-2)$ cusps. □

2.1.2 Double points

Double points (nodes) on the outline occur when a line through the focal point is tangent to S in two distinct points, and so are global events; determining the number of double points on the outline requires more complex reasoning.

Lemma 2: *There are* $\frac{1}{2}d(d-1)(d-2)(d-3)$ *double points on the outline of an algebraic surface of degree* d.

Proof: The contour generator is a complete intersection, and so its genus is given by the formula

$$\frac{1}{2}d(d-1)(2d-5)+1$$

where d is the degree of the surface (cf [14], p. 188, ex 8.4g). Project the contour generator into the image through the focal point; the resulting curve is birational to the contour generator, and so has the same genus. The singularities are stable (by the generic choice of surface and focal point), so the genus-degree formula for plane curves yields that

$$\frac{1}{2}d(d-1)(2d-5)+1 = (d(d-1)-1)(d(d-1)-2) - (n_c + n_d)$$

where n_c is the number of cusps and n_d is the number of double points. Rearranging the formula and substituting the above result on the number of cusps yields $n_d = \frac{1}{2}d(d-1)(d-2)(d-3)$. □

2.2 Global properties of the singularities of the outline

A property of the outline that will prove important later is that its singularities lie on the intersection of two plane curves, whose degree (which is relatively low for the number of points) can be determined using elimination theory. These curves can be studied, without loss of generality, by assuming that the focal point is the point $(0,0,0,1)$. This makes

computing the outline relatively simple; a point (u_0, u_1, u_2, u_3) projects through $(0, 0, 0, 1)$ to (u_0, u_1, u_2), because if (u_0, u_1, u_2) are fixed and the fourth coordinate varies, the locus of points obtained is a line, limiting to the origin as u_3 is large. Thus, (u_0, u_1, u_2) yield the line, and u_3 is a coordinate along the line.

The equation of a surface S of degree d can be rewritten as:

$$S(u_0, u_1, u_2, u_3) = H_0(u_0, u_1, u_2)u_3^d + H_1(u_0, u_1, u_2)u_3^{d-1} + ...H_d(u_0, u_1, u_2) = 0$$

where $H_i(u_0, u_1, u_2)$ is homogenous of degree i in u_0, u_1, and u_2. The focal point is $(0, 0, 0, 1)$, so that the first polar through the focal point is:

$$\frac{\partial S}{\partial u_3} = dH_0(u_0, u_1, u_2)u_3^{d-1} + (d-1)H_1(u_0, u_1, u_2)u_3^{d-2} + ...H_{d-1}(u_0, u_1, u_2)$$

and this vanishes on the contour generator too. Now the outline consists of those points (u_0, u_1, u_2) where both equations vanish; the equation of the outline is therefore obtained by eliminating u_3 between S and $\frac{\partial S}{\partial u_3}$. The singularities of the outline all have multiplicity two, and are those points (u_0, u_1, u_2) where S and $\frac{\partial S}{\partial u_3}$ have two common roots in u_3. The equations yielding these points can be obtained using a technique from Salmon [38].

Consider two polynomials in u_3,

$$F(u_3) = \sum_{i=0}^{i=d} a_{d-i}u_3^i$$

and

$$G(u_3) = \sum_{i=0}^{i=d-1} b_{d-1-i}u_3^i$$

If F and G have two common roots, then there must be some

$$M(u_3) = \sum_{i=0}^{i=d-3} A_{d-3-i}u_3^i$$

and

$$N(u_3) = \sum_{i=0}^{i=d-2} A_{2d-4-i}u_3^i$$

such that $FM + GN = 0$ identically. M is the product of all the factors of G that do not appear in F, and N is the product of all the factors of F that do not appear in G. The polynomial $FM + GN$ has degree $2d - 3$, and there are $2d - 3$ unknown A_i, and $2d - 2$ monomials in u_3. We can construct a $2d-3$ by $2d-2$ matrix $\mathbf{C}$, such that $FM+GN = \mathbf{a}^t\mathbf{C}\mathbf{u}$, where $\mathbf{a}$ is the vector

$$(A_0, A_1, ..., A_{d-3}, A_{d-2}, A_{d-1}, ..., A_{2d-5}, A_{2d-4})^t$$

$\mathbf{u}$ is the vector

$$(u_3^{2d-3}, u_3^{2d-2}, ..., u_3^2, u_3, 1)$$

and $\mathbf{C}$ is the $2d-3$ by $2d-2$ matrix whose entries are:

$$\begin{array}{cccccccccccc}
a_0 & a_1 & a_2 & .. & .. & .. & a_d & 0 & 0 & 0 & .. & 0 \\
0 & a_0 & a_1 & a_2 & .. & .. & .. & a_d & 0 & 0 & .. & 0 \\
0 & 0 & a_0 & a_1 & a_2 & .. & .. & .. & a_d & 0 & .. & 0 \\
.. & (d-2 & rows) & & & & & & & & & \\
0 & 0 & 0 & .. & .. & a_0 & a_1 & a_2 & .. & .. & .. & a_d \\
b_0 & b_1 & .. & .. & .. & b_{d-3} & b_{d-2} & b_{d-1} & 0 & 0 & .. & 0 \\
0 & b_0 & b_1 & .. & .. & .. & b_{d-3} & b_{d-2} & b_{d-1} & 0 & .. & 0 \\
.. & (d-1 & rows) & & & & & & & & & \\
0 & 0 & 0 & 0 & .. & b_0 & b_1 & .. & .. & b_{d-3} & b_{d-2} & b_{d-1}
\end{array}$$

Since, for an appropriate choice of A_i, $FM + GN = 0$ identically (i.e. all the coefficients vanish), there is some choice of $\mathbf{a}$ such that $\mathbf{a}^t\mathbf{C} = 0$. Hence, the $2d-3$ by $2d-3$ minors of $\mathbf{C}$ must vanish.

In our case, $a_j = H_j$, and $b_j = (d-j)H_j$. The method of construction of the matrix ensures that the minors are homogenous; the degree of a minor in (u_0, u_1, u_2) can be determined by computing the degree of a typical monomial in the minor. Such a monomial can be obtained by striking one column of $\mathbf{C}$, and multiplying $2d-3$ elements from the remaining square matrix using each row and each column only once. The resulting monomial will have the form $H_aH_bH_c...$, and its degree is the sum of the subscripts. This process shows the degrees of the minors are:

$$(d-1)(d-2), (d-1)(d-2)+1, (d-1)(d-2)+2, .., (d-2)(d-1)+2d-3$$

At a singularity of the outline, these minors must all vanish, so that there exists a family of curves, which intersect at most in points, of these degrees, which pass through the singular points. In particular, the singularities must lie on (though not necessarily exhaust) the intersection of a curve of degree $(d-1)(d-2)$ with a curve of degree $(d-1)(d-2)+1$, where these curves do not have a common component.

This means that the singularities are strongly constrained. A curve of degree s has $(1/2)(s+1)(s+2)$ coefficients, meaning that $(1/2)(s+1)(s+2)-1$ general points uniquely specify such a curve. There are in total $(1/2)((d^2-2d-1)+1)((d^2-2d-1)+2)$ singularities; if these were in general position, the lowest degree curve that would pass through all of them would have degree $(d^2-2d+1) = (d-1)^2$.

The matrix $\mathbf{C}$ yields a great deal of information about the structure of the problem. Write

$$\mathbf{C} = (\mathbf{c_0}, \mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_{2d-3}})$$

where the $\mathbf{c_i}$ are column vectors. Let

$$\mathbf{C_l} = (\mathbf{c_0}, \mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_{2d-4}})$$

By inspecting the diagonal elements, it can be seen that the determinant of $\mathbf{C_l}$, which is square, has degree $(d-1)(d-2)$. Let $\mathbf{D} = Adjoint(\mathbf{C_l})$ (where the adjoint is the transpose of the matrix of cofactors), and let

$$\mathbf{C_r} = (\mathbf{c_0}, \mathbf{c_1}, ..., \mathbf{c_{2d-6}}, \mathbf{c_{2d-5}}, \mathbf{c_{2d-3}})$$

Inspecting the diagonal elements shows that $Det(\mathbf{C_r})$ has degree $(d-1)(d-2)+1$. Write $\mathcal{P}$ for $Det(\mathbf{C_l})$ and $\mathcal{Q}$ for $Det(\mathbf{C_r})$. Now both $\mathcal{P}$ and $\mathcal{Q}$ are $2d-3$ by $2d-3$ minors of $\mathbf{C}$, and so must vanish on all the singular points of the outline.

Since $\mathbf{Cu}$ is a vector of polynomials, all of which vanish at every point on the contour generator, $\mathbf{DCu}$ must also consist of a vector of polynomials, each of which vanishes at every point on the contour generator. In particular, the last row of $\mathbf{DC}$ has the form:

$$(0, 0, 0, ..., 0, \mathcal{P}, \mathcal{Q})$$

and so the last element of $\mathbf{DCu}$ is the equation

$$\mathcal{P}u_3 + \mathcal{Q}$$

which must vanish at every point on the contour generator. This equation cannot vanish trivially; that is, at "almost every" point on the contour generator, both $\mathcal{P}$ and $\mathcal{Q}$ are non-zero, by the following argument:

> both $\mathcal{P}$ and $\mathcal{Q}$ are homogenous polynomials in the variables u_0, u_1 and u_2 and so they vanish on a cone passing through the point $(0,0,0,1)$. If $\mathcal{P}$ and $\mathcal{Q}$ were to vanish on the entire contour generator, this cone would contain the contour generator, and so $\mathcal{P}$ and $\mathcal{Q}$ would have to vanish on the projection of the contour generator through the point $(0,0,0,1)$ to any plane. However, a projection of the contour generator to a plane through $(0,0,0,1)$ must (by the generic choice of surface) be irreducible and have degree $d(d-1)$; neither $\mathcal{P}$ nor $\mathcal{Q}$ can vanish at every point of an irreducible curve of this degree, because their degrees are too low.

This means that, if $\mathcal{P}$ and $\mathcal{Q}$ can be determined from the image, *the contour generator can be reconstructed from the outline*, because the "missing" homogenous coordinate of the contour generator, u_3 (which can loosely be thought of as "depth") can be determined as

$$u_3 = \frac{-\mathcal{Q}}{\mathcal{P}}$$

This expression, though not strictly a function, is meaningful, because the degree of $\mathcal{Q}$ is one larger than the degree of $\mathcal{P}$; as a result, if (u_0, u_1, u_2) were to be scaled by λ, the expression for u_3 would be scaled by λ too.

In fact, $\mathcal{P}$ and $\mathcal{Q}$ can be determined from image information alone, up to an ambiguity which is a subgroup of the projective group; $\mathcal{P}$ is the only polynomial of degree $(d-1)(d-2)$ that vanishes on all the singularities of the outline, and hence can be determined from image information up to scale. In turn, $\mathcal{Q}$ is a polynomial of degree $(d-1)(d-2)+1$ that vanishes on all the singularities of the outline. There is a four dimensional space of such polynomials; section 3.1.3 shows that the ambiguity arising from choosing one of these polynomials to act as $\mathcal{Q}$ arbitrarily is just a projective transformation of the contour generator.

The constraints that singularities lie on curves of particular degrees determine the family of curves that are generic outlines of smooth surfaces, according to an result of [5] which states that:

> **Theorem:** *(D'Almeida) Let Γ be a plane curve of degree $n(n-1)$, $n \geq 3$. The necessary and sufficient condition that there exists a smooth surface $S \subset P^3$ and*

a generic point p of P^3 such that Γ is the curve of ramification of the projection of S through p, is as follows:

The curve Γ has $d = n(n-1)(n-2)(n-3)/2$ ordinary double points, $k = n(n-1)(n-2)$ cusps and no other singularities. There are two curves μ_0 and μ_1 of degrees n^2-3n+2 and n^2-3n+3 respectively, without a common component, that pass through the singular points of Γ. The minimal degree of a plane curve containing the singular points of Γ is n^2-3n+2

Note that the "curve of ramification" is equivalent to our outline. Any errors in translation are mine.

2.3 Further global properties of the outline

The study of outlines is quite rich in curious geometric properties; in particular, the form of a generic outline is strongly constrained, and the projective invariants of a generic outline must satisfy constraints. For example, note that there must exist a frame in which the outline of a cubic surface has the form $C^2 - Q^3 = 0$, where Q is quadratic and C is cubic.

This can be shown by representing the surface as

$$S(u_0,u_1,u_2,u_3) = H_0(u_0,u_1,u_2)u_3^3 + H_1(u_0,u_1,u_2)u_3^2 + H_2(u_0,u_1,u_2)u_3 + H_3(u_0,u_1,u_2)$$

By choice of frame, the focal point can be given coordinates $(0,0,0,1)$ and we can ensure $H_1(u_0,u_1,u_2) = 0$ identically; divide by H_0 (which is a constant), to get the form

$$S(u_0,u_1,u_2,u_3) = u_3^3 + H_2(u_0,u_1,u_2)u_3 + H_3(u_0,u_1,u_2)$$

The polar through the focal point is now

$$T(u_0,u_1,u_2,u_3) = 3u_3^2 + H_2(u_0,u_1,u_2)$$

The resultant with respect to u_3 has degree six, and consists of terms formed from H_3 (degree 3) and H_2 (degree 2), and so must have the form $C^2 - Q^3 = 0$, for an appropriate choice of C and Q.

Similar statements are possible about the outlines of surfaces of higher degree, but the form of the constraint becomes more complex; a possible benefit of such a result includes controlling the complexity of the fitting problem - most algebraic curves are not outlines.

3 Obtaining the contour generator from the outline

Determinining the contour generator from the outline requires knowledge of the "depth" to the contour generator at each point of the outline. The last sections indicated how this depth is to be found, by showing an expression that gives the homogenous coordinate u_3 as a rational function of the other three homogenous coordinates on the outline. In particular, this rational function can be determined from the singularities of the outline using the property that both numerator and denominator vanish the singularities of the outline. This means that the expression for u_3 is undetermined at these points. Surprisingly, this is a useful property, because it makes it possible to obtain a non-singular space curve from a singular plane curve. In particular, the process sketched above for determining the

contour generator from the outline is widespread in algebraic geometry, and is known as "blowing up." This section provides some simpler examples of blowing up to demonstrate how the process can "undo" singularities; it then shows that the reconstruction of the contour generator is correct by showing that it is the only possible such reconstruction, up to a projective transformation of space. This latter result requires some complicated machinery, which is briefly introduced.

3.1 Blowing up

The outline has only cusps and double points as singularities, by the assumption that both surface and viewing position are generic. This means that there is no need to blow up more complex singularities. In the case of blowing up cusps or double points, the central issue a depth function that can be evaluated along the plane span of the curve, giving the coordinates of the space curve in affine coordinates as

$$(x, y, \frac{f(x,y)}{g(x,y)})$$

or in homogenous coordinates as

$$(x_0, x_1, x_2, \frac{F(x_0,x_1,x_2)}{G(x_0,x_1,x_2)}) = (G(x_0,x_1,x_2)x_0, G(x_0,x_1,x_2)x_1, G(x_0,x_1,x_2)x_2, F(x_0,x_1,x_2))$$

Clearly, in the case of homogenous coordinates the degree of G is one less than the degree of F. The depth function must have two values at each double point (these are given as limits as a point on the curve approaches the double point), so as to construct two points at different depths in space that correspond to the single point in the image. As the following two examples show, this is achieved by having the depth function undefined $(0/0)$ at the singularities, with appropriate limiting properties close to the singularities; in the case of homogenous coordinates, all four homogenous coordinates vanish simultaneously, again with appropriate limiting properties.

3.1.1 Example: blowing up a double point in the affine plane:

Consider the curve given by $y^3 - x^2 + y^2 = 0$, which has a double point at the origin where the curve crosses itself transversally. The curve can be parametrised as

$$(x, y) = (t^3 - t, t^2 - 1)$$

where t is some complex parameter. The curve passes through the double point when $t = 1$ or $t = -1$.

The function

$$f(x, y) = (x, y, x/y)$$

which takes a point in the plane to a point in space, is undefined at the origin; furthermore,

$$\lim_{t \to 0} f(t\cos\theta, t\sin\theta)$$

depends on θ, so that when the function is applied to a curve approaching the origin, the value of the z-coordinate depends on the direction of the approach. In particular, applying

this function to the curve under consideration produces

$$(x, y, z) = (t^3 - t^2, t^2 - 1, t)$$

less the points $t = 1$ and $t = -1$, where the function is not defined. However, at these points the space curve has meaningful limits, which are $(0, 0, 1)$ and $(0, 0, -1)$. By attaching these limit points we obtain a smooth space curve from a singular plane curve.

3.1.2 Example: blowing up a cusp in the projective plane:

In the projective plane, points are given by three homogenous coordinates. In this case, a polynomial cannot be a function, because scaling each homogenous coordinate changes the value of the polynomial without changing the point at which the function is defined. Thus, functions are given by ratios of homogenous polynomials of the same degree in homogenous coordinates. In fact, a function that maps a curve in the projective plane to a curve in *projective* space can be given as *four* homogenous polynomials of the same degree in the homogenous coordinates of the plane; in this form, each polynomial represents a homogenous coordinate in space.

Consider the curve given by $x_0^3 - x_2 x_1^2 = 0$ in the projective plane; this curve has a cusp at $(0, 0, 1)$, and can be parametrised as $(r^2 s, r^3, s^3)$, where (r, s) are the homogenous coordinates of a point on the projective line. Consider the following function from the projective plane to projective three-space:

$$f(x_0, x_1, x_2) = (x_0^2, x_1 x_0, x_1 x_2, x_0 x_2)$$

Applied to the curve, this function yields the parametric space curve given in homogenous coordinates by:

$$(r^4 s^2, r^5 s, r^3 s^3, r^2 s^4)$$

which is equivalent to that given in homogenous coordinates by:

$$(r^2 s, r^3, r s^2, s^3)$$

This curve is a twisted cubic - this is perhaps easiest to see by dividing by the fourth coordinate, writing $r/s = t$ and ignoring the point at infinity, giving the curve in affine (non-homogenous) coordinates as (t^2, t^3, t); this space curve has no singularities.

3.1.3 Blowing up the outline

The key to blowing up a curve with double points and cusps, as the examples have shown, is to obtain a depth function that goes to $\frac{0}{0}$ at the double points and cusps of the curve. For the outline of an algebraic curve in affine coordinates, such a function is easily available. Recall from section 2.2 that there exists two equations $\mathcal{P}$ of degree $(d-1)(d-2)$, and $\mathcal{Q}$ of degree $(d-1)(d-2)+1$ (with no factor in common with $\mathcal{P}$), both of which vanish on the singularities of the outline. These equations yield the necessary depth function.

Because the reconstruction is proceeding up to a projective ambiguity, it is possible to choose a focal point; choose this focal point to be $(0, 0, 0, 1)$, to simplify the working. Now the contour generator is some curve $(u_0(t), u_1(t), u_2(t), u_3(t))$, and the outline in the image plane consists of the curve $(x_0(t), x_1(t), x_2(t)) = (u_0(t), u_1(t), u_2(t))$. Reconstructing the

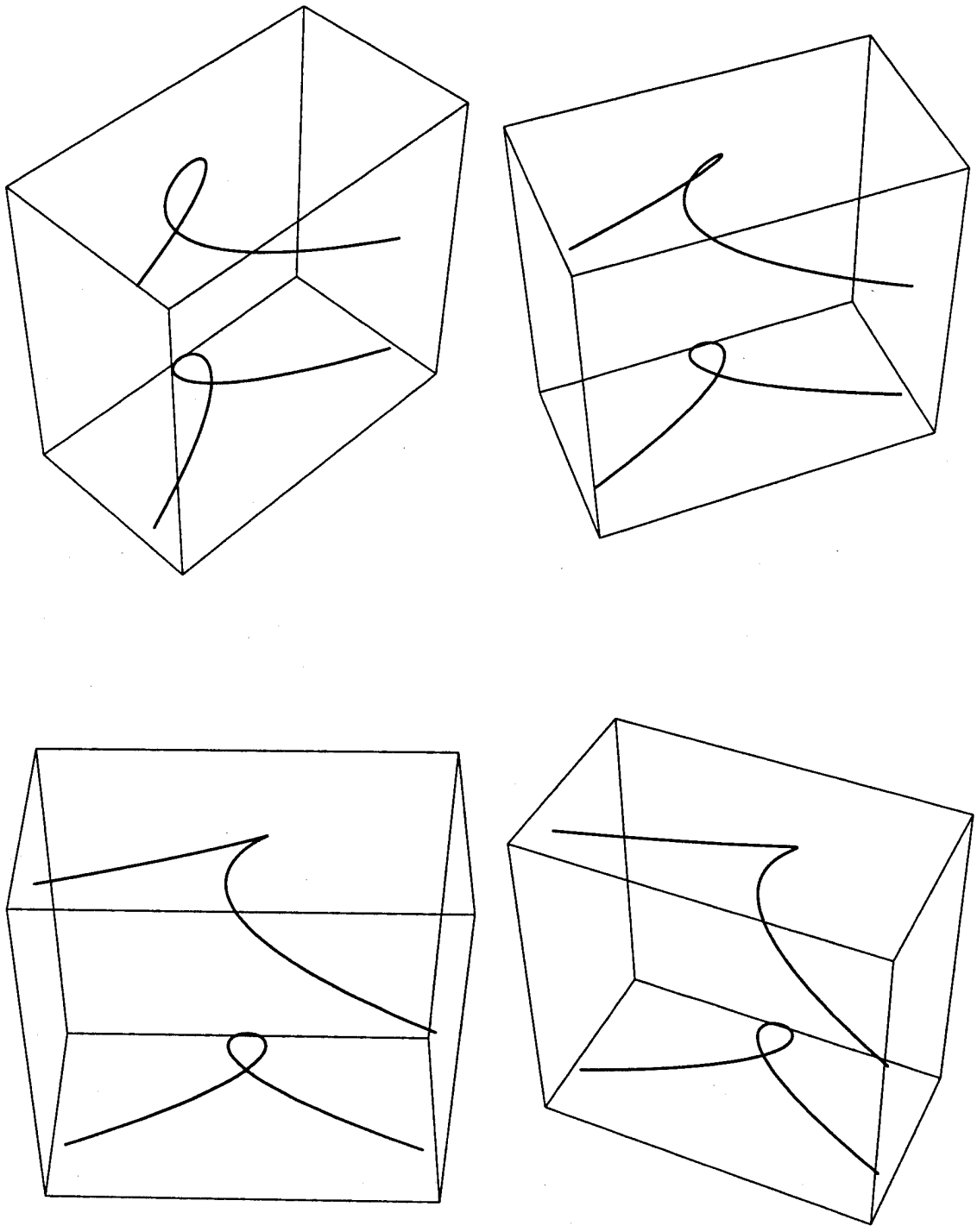


Figure 2: Four frames from a fly-by, showing a plane curve with a double point and its blow-up. The blown-up curve is a non-singular space curve, shown here lying above the plane curve. It projects to the plane curve under orthographic projection in this case.

contour generator consists, in effect, of supplying the missing $u_3(t)$. However, from the previous section, $\mathcal{P}u_3 + \mathcal{Q} = 0$ on the contour generator, and $\mathcal{P}$ and $\mathcal{Q}$ are expressions in u_0, u_1, and u_2 alone, *which can be determined from the image information*, so that u_3 can be determined at each point on the curve.

In particular, given an outline in the projective plane, apply the map

$$(x_0, x_1, x_2) \rightarrow (x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P}, \mathcal{Q})$$

taking every point on the outline to a point in space. For convenience, call this map the "lifting map". At the singular points of the outline, the lifting map degenerates (as both $\mathcal{P}$ and $\mathcal{Q}$ vanish at these points, the image of these points in the map given is $(0,0,0,0)$, which is not a meaningful point in projective space). The result of the following section shows that the closure (required to fill in the missing points where the map degenerates) of the image of the outline in the lifting map must be the contour generator.

The lifting map has further useful properties; in particular, it has the property that

$$\pi o Lift = Identity$$

where π is projection through the point $(0,0,0,1)$. In coordinates, drop the fourth homogenous coordinate, so that

$$\pi o Lift : (x_0, x_1, x_2) \rightarrow (x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P}) = (x_0, x_1, x_2)$$

This means that, if $Lift$ takes the outline to the contour generator, it does so with a notion of the appropriate focal point through which to project the contour generator back on to the outline - the particular lift constructed presumes that the focal point is $(0,0,0,1)$, which can be done without loss of generality by choice of coordinates. Any other particular focal point can be chosen as well, though the form of the resulting lift is slightly more complicated; the important thing is that, once the lifting process has been applied, both the contour generator and the focal point are available, *in a single coordinate system.* This data is sufficient to determine the surface.

The lifting map contains an intrinsic projective ambiguity, because $\mathcal{Q}$ cannot be determined uniquely. There are sufficient singularities for $\mathcal{P}$ to be known up to a scale - which is not a source of ambiguity, because we are working in homogenous coordinates - but there is a four-dimensional space of curves of degree $(d-1)(d-2)+1$ that vanish on the singularities, spanned by $(\mathcal{Q}, x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P})$. An element of this space is given by $\mathcal{Q}_a = a_0x_0\mathcal{P} + a_1x_1\mathcal{P} + a_2x_2\mathcal{P} + a_3\mathcal{Q}$ Now if the lifting map uses $\mathcal{Q}_a$ instead of $\mathcal{Q}$, the resulting curve is:

$$(x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P}, \mathcal{Q}_a) = (x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P}, \mathcal{Q})\mathbf{M}$$

where $\mathbf{M}$ is the matrix:

$$\begin{matrix} 1 & 0 & 0 & a_0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 1 & a_2 \\ 0 & 0 & 0 & a_3 \end{matrix}$$

This is clearly just a projective transformation, as long as $a_3 \neq 0$. Since both $\mathcal{P}$ and the whole space of possible $\mathcal{Q}_a$'s can be determined from the outline, satisfying the requirement that $a_3 \neq 0$ simply involves choosing a $\mathcal{Q}_a$ that does not share a factor with $\mathcal{P}$, which is easily done.

3.2 Uniqueness of the lift

The sections above have shown constructively that the outline can be lifted to yield the contour generator, and have demonstrated a lifting process that must yield the contour generator from the outline. It is also possible to show that this is the only process that will do so; the proof is not novel, and requires a certain amount of technical algebraic geometry; it is included here for completeness. Space does not allow a comprehensive introduction to the material required, but subsection 3.2.1 introduces the general approach, and sketches the direction that the mathematics in subsection 3.2.2 takes, as the form of argument used represents a powerful tool for solving questions about space curves. The reader is referred to [14], which is difficult but comprehensive, or to [12], which is much more approachable but less wide-ranging. The reader willing to accept that the lifting process in the previous section yields the contour generator may wish to skip both sections, or read only subsection 3.2.1.

3.2.1 Thrust of the mathematics

The central question is: given a projection of an abstract algebraic curve satisfying particular constraints, in how many projectively different ways could that curve be embedded in space, consistent with the image data? The result that will appear is that there is a natural choice of depth function to obtain the contour generator from the outline. This result is a statement about the possible embeddings of a curve in space that are consistent with the image data.

Embeddings of curves are generally attacked through a technical device called a *line bundle*, which consists of a collection of sets made up of the cartesian product of an open set on the curve and an affine line. These sets are pasted together in a precise way using *transition functions*. Transition functions are associated with the intersection of two of these sets; their domain is the open set on the curve, and their range is the line. Transition functions allow studies of *sections* of line bundles, which associate points on the line with points on the curve. Formally, a section is a map from the curve to the line bundle, so that the projection of the map onto the first factor is the identity; this means that, in some coordinate system, the map has the form $f : p \rightarrow (p, q)$, where p is a point on the curve and q is a point on the line. Where two sets intersect, there are two ways of writing each point on the curve and each point on the line - one set of coordinates for each set. Transition functions define the correspondence between points on the line in the coordinates associated with the first set, and those in the coordinates associated with the second set. In fact, the choice of transition functions yields the bundle.

The result is an object that locally looks like a piece of curve crossed with the affine line (c.f. the vector bundles of differential geometry). Line bundles in algebraic geometry have more rigidity properties than the bundles of differential geometry, for two reasons. Firstly, the topology used to define open sets is the Zariski topology, where all algebraic sets are closed; this means that an open set on a curve consists of the whole curve, less some finite number of points. Secondly, the bundles under consideration are typically *holomorphic* - this means that the transition functions are analytic in their domain.

The following example (which is a modified version of example 4.7 in [12]) displays a family of line bundles over the projective line. The first open set on the projective line will consist of the points given in coordinates as s, for s some complex number (henceforth, the complex numbers will be denoted by $\mathcal{C}$); call this set U. This is the projective line

less one point, the point at infinity. The second open set will consist of the points given in coordinates as t, for t some complex parameter; call this set V. Again, this is the projective line less a point (which would be the origin in s coordinates). Define the change of coordinates in crossing from U to V by $t = 1/s$. The two sets, pasted together in this way, give the whole of the projective line; figure 3 illustrates how the sets are assembled together to yield a line.

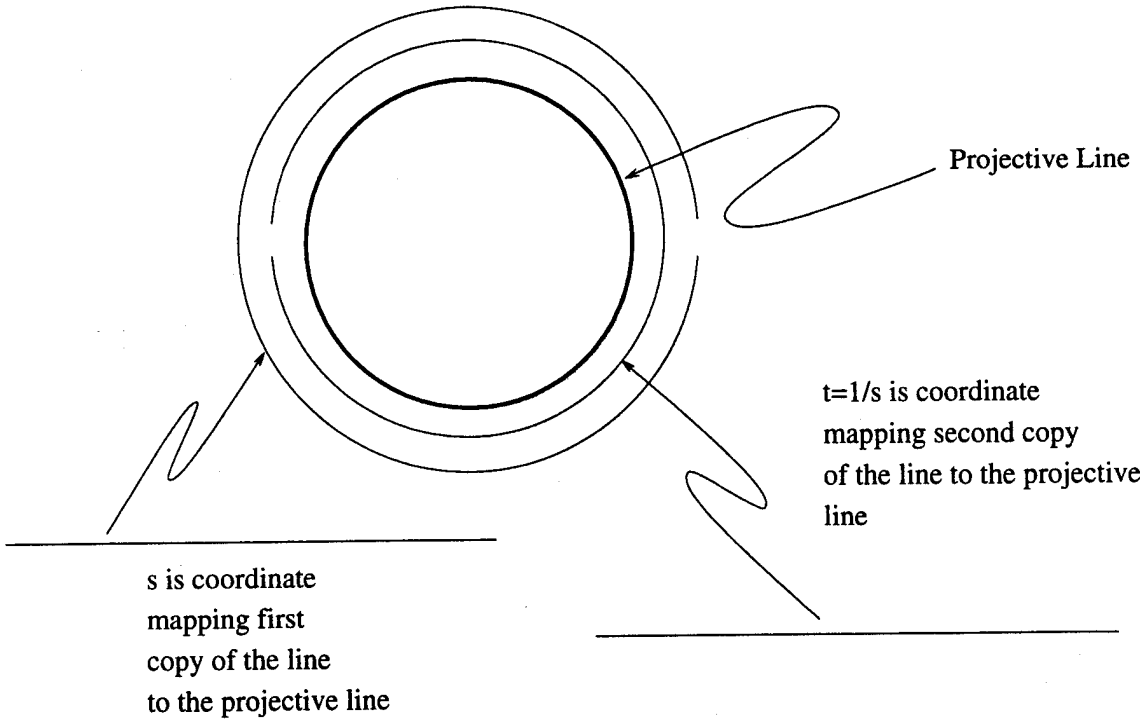


Figure 3: Pasting together two affine lines to get a projective line. The first copy of the affine line parametrizes all the points on the projective line save the point at infinity; the second copy parametrizes the point at infinity, but lacks the origin. These two copies intersect almost everywhere; they are pasted together by specifying how the coordinate of a given point on one set relates to the coordinate of the equivalent point on the other.

The line bundle will consist of the sets $U \times \mathcal{C}$ and $V \times \mathcal{C}$, pasted together in an appropriate way. The set $U \bigcap V$ consists of the whole line, less two points. There must be two transition functions: f_{VU}, which takes coordinates on the line in U's frame to those in V's frame, and f_{UV}, which takes coordinates on the line in V's frame to those in U's frame. Consider a point p in $U \bigcap V$. Write:

- p_U for the coordinates of p in U's frame;
- q_U for the coordinate in U's frame of a point on the line $\mathcal{C}$;
- p_V for the coordinates of p in V's frame;
- q_V for the coordinate in V's frame of the point on the line $\mathcal{C}$ that would be written q_U in U's frame.

Then the pair (p_U, q_U) corresponds to the pair $(p_V, q_V) = (p_V, f_{VU}(p_U)q_U)$. Clearly, we have that $f_{UV}f_{VU} = 1$, and that neither function vanishes on $U \bigcap V$. We can now define a family of line bundles by the transition functions $f_{VU} = s^{-n} = t^n$ and $f_{UV} = s^n = t^{-n}$. These functions specify how the coordinates of a holomorphic section change as we move from U to V and back.

In U, a holomorphic section of this bundle must have the form $(s, \sigma(s))$, where σ is a holomorphic function on $\mathcal{C}$. As a result, σ has a representation of the form

$$\sum_{i=0}^{i=\infty} a_i s^i$$

on U. In $U \bigcap V$, this section must also have the representation (in the coordinate t on V)

$$t^n \sum_{i=0}^{i=\infty} a_i t^{-i} = \sum_{i=0}^{i=\infty} a_i t^{n-i}$$

and this expression must also be holomorphic. In turn, this means that for $n < 0$, there are no holomorphic sections. For $n \geq 0$, there are holomorphic sections which have the form $(s, \sum_{i=0}^{i=n} a_i s^i)$ (for any choice of a_i) in U and in V, the form $(t, \sum_{i=0}^{i=n} a_i s^{n-i})$. Hence, for a given n, there is an $n+1$ dimensional vector space of holomorphic sections, corresponding to the choice of a_i.

Taken on its own, a section of a line bundle has no interest for us here; however, a ratio of holomorphic sections of a line bundle is a meromorphic (rational, with poles) function on the curve. Thus, in the example above, with $n = 3$, there is a four dimensional vector space of sections. The four sections given in U coordinates by $(s, 1)$, (s, s), (s, s^2), (s, s^3) can be thought of as a map, applied to the curve, taking it to the points given in homogenous coordinates in space as:

$$(1, s, s^2, s^3)$$

These four distinct holomorphic sections of this line bundle map the line to the twisted cubic in projective space, less one point; this missing point is the image of the point at infinity, and can be obtained by evaluating these sections at the one point in V that does not lie in $U \bigcap V$. In general, four distinct holomorphic sections of a line bundle on a curve represent a map taking the curve to a curve in P^3, and the resulting space curve is algebraic. Note

that sections can be added to one another or multiplied by constants, so that one usually considers the linear span of a set of sections. The attractive features of line bundles as tools are illustrated by our example:

- There are "few" holomorphic sections;
- it is very often possible to tell "how many" holomorphic sections there are;
- a bundle that has n independent holomorphic sections represents a map taking the curve to a curve in $P^{(n-1)}$.

As a result of these properties, line bundles are a central tool in studying embeddings of algebraic curves.

It can be seen from the example that different line bundles represent embeddings with different properties; we shall be concerned with a line bundle often represented as $\mathcal{O}_C(1)$, where C is the contour generator. In the case of the projective line given above, this would be the bundle that would result for $n = 1$. For a general plane curve C, a general section of $\mathcal{O}_C(1)$ would vanish either on a set of points where a line intersects the curve, or on a set of points that are functionally equivalent to a linear section. In this case, functional equivalence means that the points are given by the vanishing of $(p_1\pi)/p_2$, where p_1 and p_2 are homogenous polynomials of the same degree, π is the equation of a line, and the expression $(p_1\pi)/p_2$ has no poles, so that the zeros of p_2 must all lie on zeros of $p_1\pi$. Clearly, if p_1 is the equation of some arbitrary line and $p_2 = \pi$, this condition is satisfied. For some curves, there are other cases that will satisfy this condition. For example, if C is the outline of a surface, then $p_1 = \mathcal{Q}$, $p_2 = \mathcal{P}\pi$ will also satisfy this condition, where $\mathcal{P}$, $\mathcal{Q}$ are the equations vanishing on the singularities and defined above. This follows because the expression $\mathcal{P}u_3 + \mathcal{Q}$, which was shown above to vanish on the contour generator, demonstrates that all the zeros of $\mathcal{P}$ that lie on the contour generator conincide with zeros of $\mathcal{Q}$.

For a space curve, a choice of four linearly independent sections of the bundle $\mathcal{O}_C(1)$ gives an embedding of the curve in space; in particular, this bundle admits four sections that can be represented in coordinates as (u_0, u_1, u_2, u_3) (which basically just embeds the curve where it is in space). Four linearly independent sections chosen from the linear span of this family would yield an embedding of the curve that is projectively equivalent to the original curve. More interestingly, three linearly independent sections chosen from the linear span of this family would represent a projection of this curve onto a plane through some focal point; to recover the space curve, one would need to determine a fourth section in the family generated as the span of (u_0, u_1, u_2, u_3) (which would generate our "depth function"). Of course, if $\mathcal{O}_C(1)$ admits more than this four dimensional vector space of sections, the problem is hopeless, as it would not be possible to determine whether the fourth section chosen actually lies in the span of (u_0, u_1, u_2, u_3), and so one could not know without other sources of information whether the embedding chosen corresponded to the correct one. The crucial fact is that $\mathcal{O}_C(1)$ has only a four dimensional family of sections for C a contour generator (in fact, for C a complete intersection). This means that the fourth section can be determined from a projection of the curve up to at worst a projective ambiguity, so that the contour generator can be recovered from the outline.

3.2.2 Mathematical details

At issue is $\mathcal{O}_C(1)$, for C the contour generator; if $H^0(C, \mathcal{O}_C(1))$, which is the space of sections of $\mathcal{O}_C(1)$, is isomorphic to $H^0(P^3, \mathcal{O}_{P^3}(1))$, then $H^0(C, \mathcal{O}_C(1))$ has dimension four. Since the outline is birational to the contour generator, $\mathcal{O}_O(1)$ is the same as $\mathcal{O}_C(1)$, where O represents the outline; three linearly independent sections of $\mathcal{O}_O(1)$ are known (in coordinates, (x_0, x_1, x_2)). If a fourth can then be determined, then O can be embedded in space using these four sections, and the result must be projectively equivalent to C.

Lemma 3: *Given an algebraic curve C, which is a complete intersection in P^3 and is not a plane curve, $H^0(C, \mathcal{O}_C(1))$ is isomorphic to $H^0(P^3, \mathcal{O}_{P^3}(1))$.*

Proof: I am indebted to Prof. O. DeBarré, of the Mathematics Department, University of Iowa, for pointing out the following lemma, and showing me how it could be proven. This proof largely follows his; errors or inaccuracies are of my own addition. Note that a similar fact appears as an exercise in [14] (p. 188, ex. 8.4).

Consider the following exact sequence of sheaves associated with the curve:

$$0 \to \mathcal{I} \to \mathcal{O}_{P^3} \to \mathcal{O}_C \to 0$$

where the symbols have their usual meaning (see, for example, [14]). Taking the associated cohomology sequence, and twisting by 1, we obtain the following long exact sequence:

$$0 \to H^0(P^3, \mathcal{I}(1)) \to H^0(P^3, \mathcal{O}_{P^3}(1)) \to H^0(C, \mathcal{O}_C(1)) \to H^1(P^3, \mathcal{I}(1)) \to \dots$$

$H^0(P^3, \mathcal{I}(1))$ represents those homogenous linear expressions that vanish on the curve, and must be empty because the curve does not lie in any plane. $H^0(P^3, \mathcal{O}_{P^3}(1))$ represents the hyperplanes in P^3 and $H^0(C, \mathcal{O}_C(1))$ represents the space of sections of the line bundle given by a hyperplane section of C. If we can prove that $H^1(P^3, \mathcal{I}(1))$ is empty, we have that $H^0(C, \mathcal{O}_C(1))$ is isomorphic to the system of hyperplanes in P^3, and so that the sections of this bundle form a four-dimensional space.

The curve is a complete intersection, given (say) by $p = 0$, $q = 0$, for polynomials p and q. As a result, we have the following free resolution of its ideal:

$$0 \to R \to R \oplus R \to I \to 0$$

where R is the graded ring of homogenous polynomials in four variables over the complex numbers, and I is the curve's ideal. In this sequence, the injection $R \to R \oplus R$ is given by $f :\to (-pf, qf)$, and the surjection $R \oplus R \to I$ is given by $(a, b) :\to qa + pb$. Keeping track of the grading, we find:

$$0 \to R(1-m-n) \to R(1-m) \oplus R(1-n) \to I(1) \to 0$$

This free resolution yields the exact sequence of line bundles:

$$0 \to \mathcal{O}_{P^3}(1-m-n) \to \mathcal{O}_{P^3}(1-m) \oplus \mathcal{O}_{P^3}(1-n) \to \mathcal{I}(1) \to 0$$

Taking the associated cohomology sequence, and recalling the standard result that

$$H^i(P^n, \mathcal{O}_{P^n}(j)) = 0$$

for $0 < i < n$ and for all $j \in Z$ ([14], p. 225), gives that $H^1(P^3, \mathcal{I}(1))$ is empty, and so we have:

$$0 \rightarrow H^0(P^3, \mathcal{O}_{P^3}(1)) \rightarrow H^0(C, \mathcal{O}_C(1)) \rightarrow 0$$

that is, the two are isomorphic.

Lemma 4: *The expression*

$$\frac{\mathcal{Q}}{\mathcal{P}}$$

where $\mathcal{Q}$, $\mathcal{P}$ are the polynomials, given in section 2.2 that vanish on all the singular points of the outline O, represents in coordinates an element of $H^0(O, \mathcal{O}_O(1))$, and hence an element of $H^0(C, \mathcal{O}_C(1))$, for C the contour generator.

Proof: We have shown above that

$$\mathcal{P}u_3 + \mathcal{Q} = 0$$

on the contour generator; this is sufficient.

3.2.3 Summary

Given the outline O of a surface, the space curve C given by applying the map

$$(x_0, x_1, x_2) \rightarrow (x_0\mathcal{P}, x_1\mathcal{P}, x_2\mathcal{P}, \mathcal{Q})$$

where $\mathcal{P}$ and $\mathcal{Q}$ are polynomials that can be determined by an overconstrained fitting process from the singularities of the outline, is the contour generator of the surface when it is viewed from the point $(0, 0, 0, 1)$. Applying this map to a large number of points on the outline yields a set of points lying on the contour generator. As a result, the equations that vanish on the contour generator can be determined using a fitting process. Amongst this collection of equations lies the equation of a surface, projectively equivalent to the original surface.

4 Obtaining the surface from the contour generator

The previous sections showed that it is possible to take the image outline of an algebraic surface and obtain a point in space and a space curve, which are respectively the focal point and the contour generator that gave rise to the outline, and are in the same coordinate frame - that is, the outline is obtained by projecting the reconstructed contour generator through the reconstructed focal point. The contour generator and focal point resulting from this reconstruction are projectively equivalent to the original contour generator and focal point.

Once the contour generator through a particular focal point is known, it is a relatively simple matter to obtain the surface, because of the strong relationship between the polynomials that vanish on the contour generator. There is one equation of degree $d - 1$ that

vanishes on the contour generator; if there were more, its degree would be $(d-1)^2$ or less. This equation is the first polar of the surface through $(0,0,0,1)$; the coefficients of this equation can be determined from a set of points on the contour generator by a fitting process. Call this equation T_m.

There is a five-dimensional linear space of equations of degree d that vanish on the contour generator, and this space can be determined by a fitting process. The equations lie in the linear space spanned by $(u_0T, u_1T, u_2T, u_3T, S)$. The fitting process will yield a basis for this space; call the elements of this basis $(B_0, B_1, B_2, B_3, B_4)$. Since

$$T_m = \frac{\partial S}{\partial u_3}$$

and S lies in the span of this basis, it follows that

$$S = \sum_{i=0}^{i=4} \nu_i B_i \tag{1}$$

for some set of constants ν_i and that

$$T_m = \sum_{i=0}^{i=4} \nu_i \frac{\partial B_i}{\partial u_3} \tag{2}$$

Clearly, this equation is true in coefficients. The coefficients of T_m are known, as are the coefficients of B_i, and hence those of their partial derivatives. Since T_m must have at least 10 coefficients for the problem to be interesting (S must have degree 3 or greater for the result to be non-trivial), the terms ν_i can be determined from equation 2. Once ν_i are known, S can be reconstructed from equation 1. There must be at least one solution, because the curve is known to be a contour generator. In the general case, this is the only solution.

Lemma 5: *For S a generic surface viewed from a given focal point f, there is no other surface S', such that the contour generator of S' viewed from f is the same curve as the contour generator of S viewed from f.*

> **Proof:** The process that forms the contour generator is covariant. It is therefore sufficient to demonstrate that this lemma holds for a particular focal point. This focal point can be chosen to be $(0,0,0,1)$.
>
> If there are two different surfaces, whose equations are S and S', which have the same contour generator when viewed through this focal point, the tangency relation that defines this contour generator must be the same for both surfaces, as the contour generator has degree $d(d-1)$, and so only one form of degree $d-1$ can vanish on it. This can be written as:
>
> $$\frac{\partial S'}{\partial u_3} = \lambda_0 \frac{\partial S}{\partial u_3}$$
>
> where λ_0 is an unknown constant to allow for scaling the equations (which does not affect the geometry of the underlying curve).
>
> Furthermore, we have that the linear system of five degree d forms that vanishes on the contour generator is the same for each surface. Thus, in particular, there are constants λ_i such that
>
> $$S' = \lambda_1 u_0 \frac{\partial S}{\partial u_3} + \lambda_2 u_1 \frac{\partial S}{\partial u_3} + \lambda_3 u_2 \frac{\partial S}{\partial u_3} + \lambda_4 u_3 \frac{\partial S}{\partial u_3} + \lambda_5 S$$

As a result, we can write:

$$\frac{\partial S'}{\partial u_3} = \lambda_1 u_0 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_2 u_1 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_3 u_2 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_4 u_3 \frac{\partial^2 S}{\partial u_3 \partial u_3} + (\lambda_5 + \lambda_4) \frac{\partial S}{\partial u_3}$$

This can be rewritten as:

$$\lambda_0 \frac{\partial S}{\partial u_3} = \lambda_1 u_0 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_2 u_1 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_3 u_2 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \lambda_4 u_3 \frac{\partial^2 S}{\partial u_3 \partial u_3} + (\lambda_5 + \lambda_4) \frac{\partial S}{\partial u_3}$$

By rearranging terms, and setting $\mu_0 = \lambda_1$, $\mu_1 = \lambda_2$, $\mu_2 = \lambda_3$, $\mu_3 = \lambda_4$, $\mu_4 = \lambda_5 + \lambda_4 - \lambda_0$, we obtain:

$$\mu_0 u_0 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \mu_1 u_1 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \mu_2 u_2 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \mu_3 u_3 \frac{\partial^2 S}{\partial u_3 \partial u_3} + \mu_4 \frac{\partial S}{\partial u_3} = 0$$

For the case that $S' = \lambda_0 S$, we must have that $\lambda_1 = 0$, $\lambda_2 = 0$, $\lambda_3 = 0$, $\lambda_4 = 0$, and $\lambda_5 = \lambda_0$, so that all the μ_i must vanish, and the equation is trivially true. If we have S', S where $S' \neq \lambda_0 S$, then there must be some solution for the above equation where not all μ_i vanish. This yields an overdetermined system of equations *in the coefficients* of S, where the μ_i are unknown. For these equations to be satisfied, the determinants of the coefficient matrices, which are easily shown to be non-trivial expressions in the coefficients of S alone, must vanish. In turn, these determinants represent constraints that the coefficients of S must satisfy, and so S is not a general surface.

5 Geometric ambiguities

The discussion above assumed abstract projection. Because the focal point for the reconstruction and the sections of the line bundle used to lift the outline were chosen arbitrarily, it is not surprising that the best possible reconstruction is up to a projective transformation. However, this leaves a substantial ambiguity in the surface's geometry. It is often the case that the internal parameters of a camera are fully or partially known, and one might hope that a better reconstruction is possible in this case. Surprisingly, unless a modelbase is available, a better reconstruction appears impossible.

Consider a calibrated camera, where, without loss of generality, the focal point lies at $(0,0,0,1)$. The outline of an object is formed by a cone of rays through this point, and tangent to the object itself. The intrinsic ambiguity of the reconstruction process must include all transformations of the object that fix this cone of rays and the focal point - this is the group of dilations of space, written as:

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & d \end{matrix}$$

where $d \neq 0$. If there is no modelbase, then the geometry of the surface observed must be given as a set of invariants to some transformation group. In particular, in most conceivable applications the description must be invariant to Euclidean transformations of space. It is

easily verified[6] that the smallest subgroup of the projective group that contains both the Euclidean group and the dilations is the projective group itself. This means that to describe algebraic surfaces by invariants using only outline information and without reference to a modelbase, one must use projective invariants, whether the camera is calibrated or not.

However, a modelbase changes the ambiguity substantially. If, for example, there is a discrete modelbase with a small number of models, it is straightforward to extend the consistency approach of [9] to yield a Euclidean reconstruction of a system of surfaces, though the study of ambiguities in the reconstruction appears to become difficult. It is not known whether these ambiguities allow reconstructions when there are parametrised families of models.

6 Discussion

There is now a constructive path from observations of outline points to the full projective geometry of the surface, which goes as follows:

- Fit an algebraic curve that is an outline to the observations - this process will also yield the degree of the surface (by a search over degrees $d(d-1)$ for increasing d, if necessary).
- Determine the singularities of this fitted curve (using the coefficients of the curve).
- Compute the coefficients of the polynomials that would blow up these singularities, as described above in section 3.1.3.
- Use these coefficients to form a map from the plane to space (section 3.1.3). Apply this map to a large number of points on the outline, yielding a collection of points in space.
- Determine the unique surface of degree $d-1$ passing through these points in space.
- Determine the five-parameter family of surfaces of degree d passing through these points in space.
- Determine the ν_i of the previous section, using the methods given there.
- These ν_i can be substituted into the equations above, to give the coefficients of the surface.

Although a simple implementation that successfully identifies cubic surfaces from synthetic outline data, exists, there are real difficulties in constructing an implementation of this approach that works in a practical vision system:

- Computing the outline from image data requires fitting high degree algebraic curves to edge points. The degree goes up as the square of the degree of the surface.

[6]The most practical technique is simply to form the commutators for the Lie algebra of the group containing both Euclidean transformations and dilations, as described in [25], and then note that the span of the set of commutators and generators is the Lie algebra of the projective group.

- Computing the contour generator from the outline is tricky, as it requires finding singularities in the outline. Unfortunately, a small change in the coefficients of the outline can lead to substantial errors in the computed singularities, both in location and in multiplicity. Such errors are guaranteed by the fact that we are using a fitted curve. Furthermore, the singularities must have special properties, for the curve to be an outline at all. This has advantages and disadvantages: the curve can be chosen from a smaller, more specialised class of curves, which may make fitting more robust; at the same time, the curve produced by a general fitter cannot, in general, even be an outline.

- In practice, determining the surface from a system of points on the contour generator involves a process of fitting algebraic surfaces to points in space, and has the associated instabilities. Considerable precision in the points is required; in the experiments on synthetic data, this could be supplied, but it is doubtful whether such precision is available in practical situations.

Despite its present impracticality, this result is valuable, primarily because it shows that shape from outline is possible in the context of a very large and interesting range of surfaces, and thereby opens several promising avenues of research:

- *It is hard to be a contour generator.* In the case of algebraic surfaces, "most" curves are not contour generators, because either their degree, their genus, or the number and type of their singularities is wrong. There is good reason to believe that a similar result must hold for surfaces drawn from a "small" parametrised family of smooth surfaces, because the range of contour generators for a given surface is so small, although the mechanisms of proof and of computation may be more complex. This is the subject of active ongoing research.

- It is an example of a recognition algorithm that recognises an object drawn from a large, parametrised world (generic algebraic surfaces of degree three or greater) without searching a model-base. If this algorithm is presented with such a surface, it can (in principle) immediately describe the surface up to the intrinsic ambiguities of the viewing geometry. Given the way the algorithm is framed, verification appears to be either extremely difficult or impossible, and so the role of the model-base becomes uncertain.

- It suggests that the global properties of *systems* of contour generators are important objects of study. Compare the simple, neat structure of the family of contour generators on a projective algebraic surface with the extraordinary complexity of its aspect graph; in this case, the aspect graph is, in principle, redundant, because a single outline contains sufficient information to determine the entire surface. Furthermore, in the case of projective algebraic surfaces, the system of contour generators is one case of a well understood class of objects: a linear system of curves on a surface. A study of this system might yield a much more practical algorithm for recognising a surface from two or three uncalibrated views, with an unknown transformation between the views, by exploiting the fact that each outline represents a curve drawn from a "small" (three-dimensional) linear system of curves.

- It opens a number of curious geometric questions; for example, what is the relationship between the six cusps on the outline of a cubic surface, and the surface? The contour generator is obtained from the outline by blowing up these six cusps on the outline; but, by standard results, if we were to extend this blowing up process to the whole plane, we would obtain a surface passing through the contour generator, and the degree of this surface could not be greater than three. Note that this is *not* a general cubic surface, because the six points blown up are not in general position, but appears to be a surface bearing some substantial relationship to the original cubic surface.

Determining a class of surfaces that is plastic enough to be useful for modelling a wide range of real objects, yet rigid enough to allow strong statements about the shape of a particular surface from a single outline, is the central issue in studying shape from contour. We have shown that algebraic surfaces represent one extreme; a very large class of surface that is so rigid that an outline determines a surface. There is room for much future work.

Acknowledgements

Olivier De Barré, of the University of Iowa Mathematics Department pointed lemma 3 out to me, and showed me how it could be proven. This work has benefited from conversations with Tom Buchanan (who referred me to D'Almeida's work), Olivier Faugeras, Margaret Fleck, Peter Giblin, Richard Hartley, Steve Maybank, Joe Mundy, John Oliensis, Charlie Rothwell, Richard Weiss and Andrew Zisserman. Anonymous referees provided extensive and extremely helpful comments.

This work was supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361, in part by the National Science Foundation under award no. IRI-92-09729, in part by Magdalen College, Oxford, in part by the University of Iowa, and in part by a National Science Foundation Young Investigator Award with matching funds from GE, Tektronix, Rockwell and Eugene Rikel.

References

[1] Basset, A.B., *A treatise on the geometry of surfaces*, George Bell and Sons, London, 1910.

[2] Binford, T.O., Levitt, T.S., and Mann, W.B., "Bayesian inference in model-based machine vision," in Kanal, L.N., Levitt, T.S., and Lemmer, J.F., *Uncertainty in AI 3*, Elsevier, 1989.

[3] Brooks, R. A., "Model-Based Three-Dimensional Interpretations of Two Dimensional Images," *IEEE PAMI*, **5**, 2, p. 140, 1983.

[4] Cipolla, R. and Zisserman, A., "Qualitative Surface Shape from Deformation of Image Curves," *Int. J. Computer Vision*, **8** *1*, 1992.

[5] D'Almeida, J., (1992). Courbe de ramification de la projection sur P^2 d'une surface de P^3, *Duke Mathematical Journal*, **65**, 2, 229-233.

[6] Dhome, M., LaPreste, J.T, Rives, G., and Richetin, M. "Spatial localisation of modelled objects in monocular perspective vision," *Proc. First European Conference on Computer Vision*, O.D. Faugeras (ed.), Springer LNCS-x, 1990.

[7] D.A. Forsyth, J.L. Mundy, A.P. Zisserman, A. Heller, C. Coehlo and C.A. Rothwell, "Invariant Descriptors for 3D Recognition and Pose," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, **13**, 10, 1991.

[8] Forsyth, D.A., Mundy, J.L., Zisserman, A. and Rothwell, C.A., "Recognising rotationally symmetric surfaces from their outlines," *Proc. Second European Conference on Computer Vision*, G. Sandini (ed.), Springer LNCS-x, 1992.

[9] Forsyth, D.A., Mundy, J.L., Zisserman, A. and Rothwell, C.A., "Using global consistency to recognise Euclidean objects with an uncalibrated camera," *Proc. CVPR-94*, 1994.

[10] H. Freeman and R. Shapira, "Computer Recognition of Bodies Bounded by Quadric Surfaces from a set of Imperfect Projections," IEEE Trans. Computers, **C27**, 9, 819-854, 1978.

[11] Giblin, P. and Weiss, R, 1986. Reconstructions of surfaces from profiles, *Proc. ICCV-1*, London.

[12] Gomez-mont, X., "Meromorphic functions and cohomology on a Riemann surface," in Cornalba, M., Gomez-mont, X. and Verjovsky, A. (eds), *Lectures on Riemann Surfaces*, World Scientific, 1989.

[13] Griffiths, P. and Harris, J. *Methods of Algebraic Geometry*, John Wiley and Sons, 1986.

[14] Hartshorne, R. *Algebraic Geometry*, Springer Verlag Graduate Texts in Mathematics, 1977.

[15] Kapur, D. and Lakshman, Y.N., (1992). Elimination methods: an introduction, *Symbolic and Numerical Computation for Artificial Intelligence*, Donald, B.R, Kapur, D. and Mundy, J.L. (eds), Academic Press.

[16] Koenderink, J.J, *Solid Shape*, MIT Press, 1990.

[17] Koenderink, J.J. "What does the occluding contour tell us about Solid Shape," *Perception*, **13**, 1984

[18] Koenderink, J.J. and Van Doorn, A., "The Internal Representation of Solid Shape with respect to Vision," *Biological Cybernetics*, **32**, 1979.

[19] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR, p.335-344, 1988.

[20] Malik, J., "Interpreting line drawings of curved objects," *IJCV*, **1**, 1987.

[21] Marr, D., *Vision*, 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and co., San Francisco.

[22] J.L. Mundy and A.P. Zisserman, "Introduction," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[23] J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[24] Ohm, J. "Space curves as ideal-theoretic complete intersections," in Seidenberg, A. (ed), *Studies in algebraic geometry*, MAA Studies in Mathematics, 1980.

[25] Olver, P.J., *Applications of Lie Groups to Differential Equations*, Springer-Verlag Graduate Texts 107, 1986.

[26] Plantinga, H. and Dyer, C. " Visibility, Occlusion and the Aspect Graph," *CS TR 736*, U. Wisconsin, 1987.

[27] Petitjean, S., Ponce, J. and Kriegman, D., "Computing exact aspect graphs of curved algebraic surfaces", *Int. J. Computer Vision*, **9**, 3, 231-255, 1992.

[28] Ponce, J. "Invariant properties of straight homogenous generalised cylinders," *IEEE Trans. Patt. Anal. Mach. Intelligence*, **11**, 9, 951-965, 1989.

[29] Ponce, J. and Kriegman, D.J. "On Recognising and Positioning Curved 3 Dimensional Objects from Image Contours," *Proc. DARPA IU Workshop*, 1989

[30] Ponce, J. and Kriegman, D.J. "Computing exact aspect graphs of curved objects: parametric patches," *Proc. AAAI Conf.*, Boston, July, 1990.

[31] Ponce, J. and Kriegman, D.J. "New progress in prediction and interpretation of line-drawings of curved 3D objects," *Proc 5th IEEE Int. Symp. Intelligent Control*, 1990.

[32] Ponce, J., Hoogs, A. and Kriegman, D.J. "On using CAD models to compute the pose of curved 3D objects," *Proc IEEE workshop on Directions in Automated CAD-based Vision*, 1991.

[33] Ponce, J. and Kriegman, D.J., "Toward 3D curved object recognition from image contours," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[34] Rieger, J. "Global Bifurcation Sets and Stable Projections of Non-Singular Algebraic Surfaces," *Int. J. Computer Vision*, **7** *3*, 1992.

[35] Rothwell, C.A., Zisserman, A.P., Forsyth, D.A. and Mundy, J.L., "Using Projective Invariants for constant time library indexing in model based vision," *Proc. British Machine Vision Conference* , 1991.

[36] Rothwell, C.A., Zisserman, A.P, Forsyth, D.A. and Mundy, J.L., "Fast recognition using algebraic invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[37] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L., "Extracting projective structure from single perspective views of 3D point sets," *International Conference on Computer Vision*, Berlin, 573-582, 1993.

[38] Salmon, G. *Modern Higher Algebra*, Chelsea, New York.

[39] Stein, F. and Medioni, G., "Structural indexing: efficient 3D object recognition," *PAMI-14*, 125-145, 1992.

[40] Taubin, G. and Cooper, D.B., "Object recognition based on moment (or algebraic) invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[41] Terzopolous, D., Witkin, A. and Kass, M. "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion," *Artificial Intelligence*, **36**, 91-123, 1988.

[42] Ullman, S. and Basri, R. (1991). Recognition by linear combination of models, *IEEE PAMI*, **13**, 10, 992-1007.

[43] Ulupinar, F, and Nevatia, R. "Shape from Contour using SHGCs," *Proc. ICCV*, Osaka, 1990.

[44] Ulupinar, F, and Nevatia, R. "Recovering shape from contour for constant cross-section generalisd cylinders," *Proc. CVPR*, Mauii, 1991.

[45] Wayner, P.C. "Efficiently Using Invariant Theory for Model-based Matching," Proceedings CVPR, p.473-478, 1991.

[46] Weiss, I. "Projective Invariants of Shapes," Proceeding DARPA Image Understanding Workshop, p.1125-1134, April 1988.

[47] Zerroug, M. and Nevatia, R., "Volumetric Descriptions from a Single Intensity Image," *Int. J. Computer Vision*, to appear.

[48] Zisserman, A.P., Forsyth, D.A., Mundy, J.L and Rothwell, C.A., "Recognizing general curved objects efficiently," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.

Using Global Consistency to Recognise Euclidean Objects with an Uncalibrated Camera

D.A. Forsyth
Computer Science
University of Iowa
Iowa City, IA 52242

J.L. Mundy
General Electric
Center for Research and Development
Schenectady, NY 12345

A. Zisserman
Robotics Research Group
Oxford University
Oxford, UK

C.A. Rothwell
Robotics Research Group
Oxford University
Oxford, UK

Abstract

A recognition strategy consisting of a mixture of indexing on invariants and search, allows objects to be recognised up to a Euclidean ambiguity with an uncalibrated camera. The approach works by using projective invariants to determine all the possible projectively equivalent models for a particular imaged object; then a system of global consistency constraints is used to determine which of these projectively equivalent, but Euclidean distinct, models corresponds to the objects viewed. These constraints follow from properties of the imaging geometry. In particular, a recognition hypothesis is equivalent to an assertion about, among other things, viewing conditions and geometric relationships between objects, and these assertions must be consistent for hypotheses to be correct. The approach is demonstrated to work on images of real scenes consisting of polygonal objects and polyhedra. **Keywords:** *Recognition, Computer Vision, Invariant Theory, Indexing, Model-based Vision*

1 Introduction

Many recent object recognition systems model viewing with an uncalibrated camera or using an uncalibrated stereo pair as inducing either an affine or a projective transformation on figure. This approach allows invariants of the appropriate transformation to be used to index models to produce a selection of recognition hypotheses. These hypotheses are combined as appropriate, and the result is back-projected into the image, and verified by inspecting relationships between the back-projected outline and image edges [3, 5, 9, 12, 14]. Indexing using projective invariants has been demonstrated for plane objects and simple polyhedral objects, and has been extended with varying success to certain types of surfaces [1, 6, 8, 13, 15]. One main disadvantage of this approach is that objects are identified only up to either an affine or a projective ambiguity. This paper argues that this ambiguity is a consequence of considering recognition hypotheses in isolation, and is not intrinsic to the approach.

Systems based on indexing using projective invariants have not, to date, been able to distinguish between objects that are projectively equivalent, but not Euclidean equivalent, because such objects have the same projective invariants. In this paper, we show that a view of two or more coplanar objects, or polyhedra is enough to allow the objects to be recognised up to only a Euclidean ambiguity, if the objects can be recognised at all and if Euclidean models are available.

1.1 Frames and terminology

Much of the work we describe consists of reconciling different assertions about coordinate frames. As a result, the discussion can become confusing without an established terminology. The paper uses the following terms:

- **object:** an actual thing in the world.
- **model:** a collection of known measurements of the projective and Euclidean geometry of an object, which is stored in the system. A model could consist of a mixture of points, lines, planes, conics and more complicated curves or surfaces.
- **model frame:** the frame of reference in which the model measurements are taken; the reference points of an object in the world are within a Euclidean transformation of the reference points in this frame.
- **world frame:** a global frame of reference, in which objects exist. If the world consists of coplanar plane objects, then the world frame is the frame of reference within this plane; otherwise, the world frame is three-dimensional.
- **image frame:** a frame of reference constructed in the image plane, usually by reference to the pixel positions in the camera. For a view of a plane world, the image frame is within an unknown projective transformation of the world frame.
- **Euclidean transformation:** a projective transformation, equivalent to a rigid motion (rotation and translation), expressed in homogenous coordinates.

In this paper, the relationships between frames are emphasized; these relationships are usually determined by computing transformations between image features and model features. Such a transformation, although computed using

some specific set of features, *expresses the transformation between the model frame and the image frame.*

2 Plane objects

Consider a scene consisting of a set of distinct, coplanar plane objects, many of which are represented in a model-base. It is well known that any view of this scene with an uncalibrated camera can be obtained by applying an appropriate plane projective transformation to the scene. The goal of a recognition algorithm is from an image of the scene, label each object correctly up to Euclidean equivalence.

Indexing using projective invariants (as in [9]) associates with each group of image features a collection of object models (labels), which are projectively equivalent, but Euclidean inequivalent.

If only one known object is present, the task is possible only if there is just one possible label for that object. If two or more labels apply, the task can be considered in terms of constructing the largest possible consistent labelling, because implicit in each recognition hypothesis is information about the frame in which the objects lie. This information can be formalised to obtain possible contradictions between recognition hypotheses. The details of the idea appear below; an example that illustrates the reasoning appears in section 2.1.1.

2.1 Theory

Consider two coplanar plane objects, o_1 and o_2, for which we have models m_1 and m_2. Write the transformation from m_k to o_k as E_k, and the transformation from m_k to the image frame as P_k. E_k is Euclidean, and moves the model into its position in the world frame. Write the projective transformation from the world frame to the image frame as Q. Then $P_k = QE_k$, so that $P_1^{-1}P_2 = E_1^{-1}Q^{-1}QE_2 = E_1^{-1}E_2$ which is Euclidean. Since labelling an image curve with a particular model name determines the transformation from that model's frame to the image frame, the pairwise consistency of labellings can be checked by forming a system of matrices $P_1^{-1}P_2$, and checking whether they are Euclidean.

Objects can consist of points, or of some mixture of points, lines, conics, and other curves, as long as a projective transformation can be computed from the object frame to the image frame. This observation also justifies our emphasis on coordinate frames, rather than on particular geometric configurations. Section 2.2 details the ambiguities implicit in this scheme.

If a labelling is consistent it is possible to reconstruct the whole plane, in the frame of one given model, since $P_j^{-1}P_i$ gives the transformation from the configuration in m_i's frame to that in m_j's frame. To reconstruct the plane in, say, m_1's frame, for all m_k compute $P_1^{-1}P_k$ and then apply this map to m_k; this will give a collection of objects in m_1's frame

2.1.1 Example

Given an image of three objects, o_1, o_2 and o_3, which are plane and coplanar, and which are instances of known models, the recognition system would proceed as follows:

1. **Determine projective equivalence classes** by indexing the model-base using appropriate projective invariants. For each object, the indexing stage returns a collection of possible Euclidean models to which it might correspond. Assume that the response is: $o_1 \rightarrow (m_1, m_4, m_7)$, $o_2 \rightarrow (m_2, m_5, m_8)$ and $o_3 \rightarrow (m_3, m_6, m_9)$.
2. **Determine all image-model transformations** for every image-model correspondence using a least squares process. Call the transformations between o_i and m_j, P_{ij}. There is a total of nine transformations.
3. **Test consistency** between model hypotheses for each pair of objects. Thus, for o_1 and o_2, form the matrices:

$$P_{11}^{-1}P_{22}, P_{14}^{-1}P_{22}, P_{17}^{-1}P_{22}, P_{11}^{-1}P_{25}, P_{14}^{-1}P_{25},$$

$$P_{17}^{-1}P_{25}, P_{11}^{-1}P_{28}, P_{14}^{-1}P_{28}, P_{17}^{-1}P_{28}$$

   if, say, $P_{11}^{-1}P_{22}$ is very close to being a Euclidean matrix, then accept the pairing (o_1m_1, o_2m_2). For this example, assume that the pairs: (o_1m_1, o_2m_2), (o_1m_1, o_3m_3), (o_2m_2, o_3m_3), (o_1m_7, o_2m_8) are consistent.
4. **Form the longest possible consistent hypothesis** by merging consistent pairings. Thus, in this example, the longest consistent hypothesis is (o_1m_1, o_2m_2, o_3m_3). This is accepted as the correct labelling for the image; consistency is defined by ensuring that each object has at most one label, so that two pairings are consistent if they refer to distinct objects, or if they assign the same labels to objects that they share. The other possible consistent labelling is (o_1m_7, o_2m_8), which is the result of an ambiguity.

2.2 Ambiguities

An ambiguous image supports two or more consistent labellings that are indistinguishable, one of which will be correct. Ambiguities arise from quite complex interactions between properties of the image and of the modelbase; some modelbases may not admit ambiguities. We assume that projectively distinct objects receive distinct labels, and study inherent ambiguities in the consistency process.

> **Definition:** Two pairs of models, say $\{m_1, m_2\}$, $\{m_1', m_2'\}$, admit an ambiguous labelling if there is some image containing objects $\{o_1, o_2\}$ so that $\{o_1m_1, o_2m_2\}$, and $\{o_1m_1', o_2m_2'\}$ are both consistent labellings of the objects.

Admitting an ambiguous labelling poses a stringent constraint on the models in the modelbase. If two pairs of models, say $\{m_1, m_2\}$, $\{m_1', m_2'\}$, admit an ambiguous labelling, then there exist projectivities $P_{11'}$ and $P_{22'}$ such that $m_1' = P_{11'} m_1$ and $m_2' = P_{22'} m_2$

It can be shown that, for the configurations to be ambiguous, there are Euclidean transformations E_a, E_b such that $P_{11'} = E_a P_{22'} E_b$. This is an action of two copies of the Euclidean group on the space of projective transformations, and invariants can be obtained for this action. For example, writing the i, j'th component of a matrix Q as q_{ij}, the expression $(q_{20}^2 + q_{21}^2)^3 / Det(Q)^2$ is an invariant under this action. This means that this expression must take the same value for $Q = P_{11'}$ as it does for $Q = P_{22'}$. Thus, for a modelbase to admit ambiguities, it must contain at least two pairs of projectively equivalent models, where the projectivities between the ambiguous models *have special properties*. In turn, this statement suggests that ambiguities are unlikely. However, there is some reason to believe that modelbases containing man-made objects are likely to contain ambiguities; for example, a sequence of scaled versions of several different objects will certainly give rise to ambiguities.

2.3 Implementation details and experiments

A system implementing the approach described has been demonstrated on real images of simple scenes, using a stripped-down version of the system described in detail in [9] to perform early vision and fitting. Indexing, though performed in a reimplementation of that system, follows essentially the same pattern, but in the present system a successful indexing attempt returns a collection of Euclidean models. To focus attention on the Euclidean labelling properties of the system, the model base contains only one projective equivalence class of models, consisting of five projectively equivalent but Euclidean distinct models, so that for all known models the projective invariants are the same. The models all consist of polygons with five sides; objects were obtained by cutting these polygons out of black cardboard.

The success of this approach can be measured both by determining its effectiveness in labelling the scene, and by looking at Euclidean invariants of an unknown object, coplanar with the known objects in the scene, and reconstructed using the techniques described; stability in these invariants means that the Euclidean labelling was sufficiently successful to allow the Euclidean structure of other objects to be determined from the labelling. Some results are shown in figures 1 and 3.

3 3D objects

The situation is more difficult when the objects are three-dimensional. It is known that the projective geometry of a range of polyhedra can be recovered partially or completely from a single perspective view with an uncalibrated camera (see [10, 11]). In turn, projective invariants

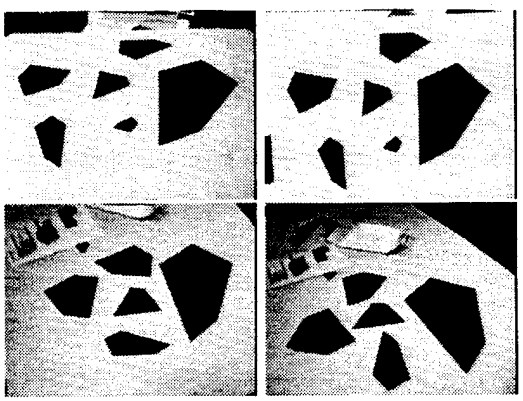


Figure 1: Six examples of scenes containing known coplanar plane objects (five sides), and an unknown object, imaged with a projective camera.

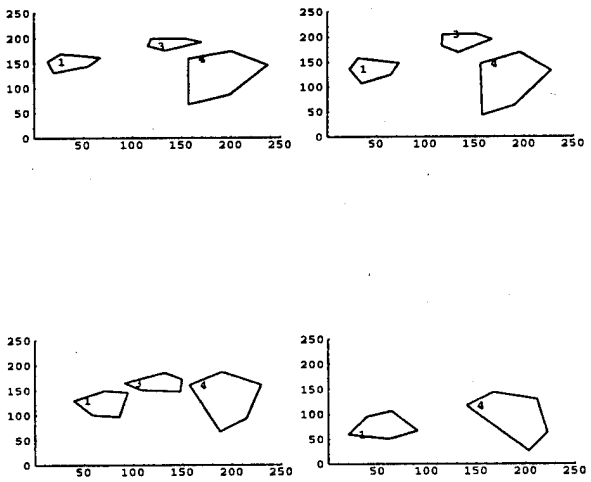


Figure 2: The Euclidean labels chosen by a global consistency analysis of the corresponding scenes in figure 1, superimposed on the backprojected outlines of the objects, which are five-sided plane polygons. Although the labellings are correct, the system consistently ignores one object (apparently as a result of a segmentation difficulty with one poorly cut corner).

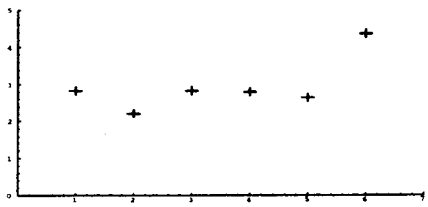


Figure 3: The graph shows the area of the unknown quadrilateral, measured in the six images shown above by computing a backprojection based on the Euclidean recognition hypotheses; note that the area is relatively stable.

can be computed and used to index the polyhedron in a model-base. Appropriate polyhedra are position-free in views, and tend to contain many faces with four or more vertices. However, for most situations, the resulting projective ambiguity is too great. To proceed, it is necessary to assume that that, for each generic view of every polyhedron in the modelbase, a distinctive projective structure can be recovered (details appear in [10]).

The perspective projection from 3D projective space, P^3, to the image plane, P^2, is modelled by a 3×4 projection matrix, $\mathbf{P}$, so that

$$\mathbf{x} \approx \mathbf{PX} \tag{1}$$

where homogeneous coordinates are used, $\mathbf{X} = (X, Y, Z, 1)^t$, $\mathbf{x} = (x, y, 1)^t$ and $\approx$ indicates equality up to a non-zero scale factor. Following Hartley [4], we partition $\mathbf{P}$ as

$$\mathbf{P} = (\mathbf{M}| - \mathbf{Mt}) \tag{2}$$

where $\mathbf{t}$ is the focal point (since the focal point projects as $\mathbf{PX} = \mathbf{0}$). Provided the first 3×3 matrix, $\mathbf{M}$, is not singular (i.e. the focal point is not on the plane at infinity), $\mathbf{P}$ can always be partitioned in this way.

To determine the position of the focal point in the model frame proceed as follows:

1. Compute the projection matrix $\mathbf{P}$ from the known model vertices and their corresponding image positions.
2. Partition $\mathbf{P}$ as above. This determines $\mathbf{t}$, which is the focal point in the object's frame.
3. The rays passing through other image outline points are given as the pre-image in $\mathbf{P}$ of the image points.

An alternative construction, due to Mohr, is also possible [7]. Labelling an image with a consistent Euclidean labelling proceeds as follows:

- Determine a set of projectively equivalent, Euclidean inequivalent labels for each polyhedron visible, using the indexing methods of [10].
- For each labelling of each item, compute the focal point and an appropriate cone of rays in the object's frame.
- Construct the largest pairwise consistent labelling of the scene, where pairwise consistency is checked by determining that the focal point and cone of rays constructed by assuming one object, is Euclidean equivalent to that constructed by assuming a second object.

As in the case of coplanar plane objects, although a correct labelling must be consistent in the sense given, a consistent labelling may not be correct. Thus, for particular scenes and particular model-bases, a unique labelling may not, in fact, be possible.

3.1 Ambiguities

The range of possible ambiguities in the case of polyhedral objects is wider than in the case of plane objects. Problems arise both as a result of viewing and self-occlusion issues and because the reconstructed polyhedra do not share the same projective frame. Ambiguities resulting from self-occlusion are not treated here, as the nature of the ambiguities depends in a complicated way on the structure of the recognition system. If the projective class of the object has been correctly recovered, and the cones through the object vertices and the focal point have been constructed correctly, the following, tractable question remains; given an image, and the projective structure of the polyhedra represented in that image, what ambiguities exist in the Euclidean labelling process described?

There is now a second source of ambiguity; many distinct objects can produce the same cone of rays through the focal point. It can be shown that, for a modelbase to admit an ambiguity, it must contain four elements p_1,p_2,p_1',p_2', with p_i and p_i' projectively equivalent, and where the transformations between the model frames satisfies:

$$P_{p_2p_2'} = ET^{-1}DTP_{p_1p_1'}$$

for some arbitrary translation T, elation D and Euclidean transformation E. Note that, since the total number of arbitrary degrees of freedom is 13, not every pair of transformations $P_{p_2p_2'}$, $P_{p_1p_1'}$ will have this property. As a result, not every modelbase admits an ambiguity, and unambiguous labellings appear possible for at least some modelbases. Many man-made objects yield modelbases that admit ambiguities (for example, a modelbase containing only cubes of different sizes).

3.2 Experiments

A system implementing the approach described has been demonstrated on real images of simple scenes. The modelbase contains five polyhedral objects, all projectively equivalent. Polygon vertices are marked in the image by hand; all further processing is automatic. Figure 4 shows

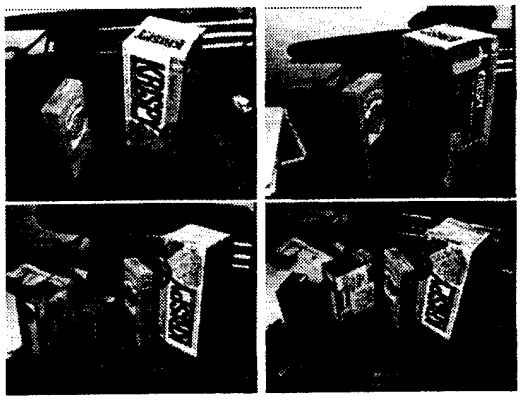


Figure 4: Four examples of scenes containing known polyhedral objects all of which are projectively equivalent, imaged using a perspective camera.

typical images; figure 5 shows the Euclidean labelling for corresponding images and figure 6 shows a reconstruction. The reconstruction techniques used at present can produce a reconstruction that is within an improper rotation (with negative determinant) of the original world; this appears to be an intrinsic ambiguity in the purely projective methods used, and may be overcome by considering the direction in which the camera is pointing. The techniques do not extend to reconstructing unknown objects in the way that the plane techniques do.

4 Discussion

We have shown methods for using geometric consistency in 2D from 2D recognition, and in 3D from 2D recognition; because the argument is based on frames and maps, the 2D from 2D argument carries over to the 3D from 3D case without modification (for example, on the output of "uncalibrated stereo"[2]).

Linking these ideas is the observation that *recognition hypotheses are frame hypotheses.* When a program asserts that some Euclidean model produced an image observation, it is making a statement about camera position and internal parameters. Such statements lead to global consistency constraints that must hold. If, in a world with many known objects, objects can be recognised effectively and unknown objects can be reconstructed up to a Euclidean ambiguity without camera calibration, there is no reason to calibrate the camera. Other constraints can be applied to the reconstruction (for example, using known objects, occlusion cues and up-vector estimates to bound the distance to the object). This paper has dealt with discrete modelbases; the case of parametrised systems of models bears further investigation.

More global consistency mechanisms are available - for example, the orderly and effective exploitation of the potential of t-junctions to explain occlusions. Many other sources of information, not necessarily primarily geometric, could be used to inform and strengthen recognition

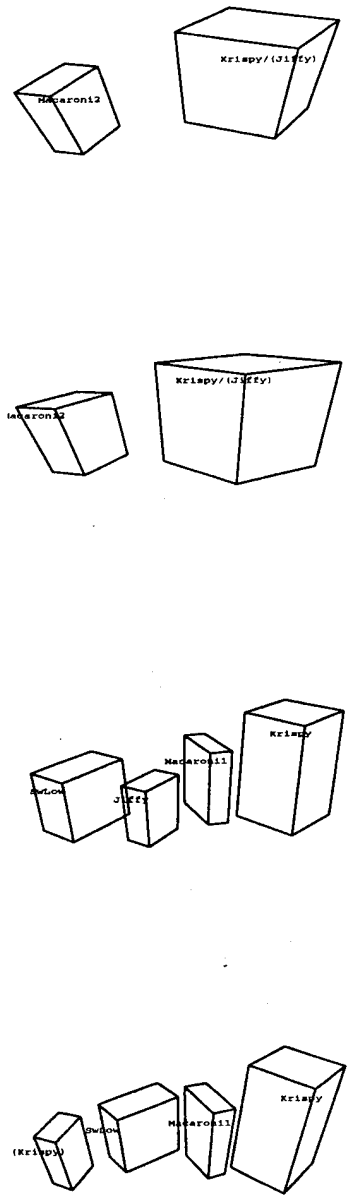


Figure 5: Euclidean labels chosen by a global consistency analysis of the scenes in figure 4, superimposed on the backprojected outlines of the objects. A label "i/j" means that there is a consistent interpretation where the object is either object i or object j. Incorrect labels are enclosed in parentheses.

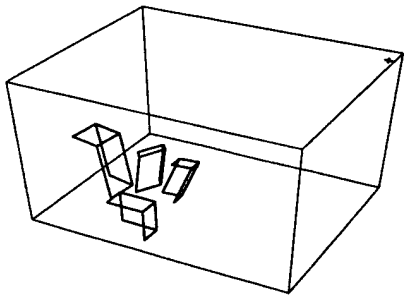


Figure 6: A Euclidean reconstructions of the polyhedral world shown in one image (bottom left image, all labels correct) taken from figure 4. The focal point is the marked point in the top right-hand corner of the figure. Note the distortions of the boxes, caused by mapping all boxes into the frame of one box; more sophisticated techniques might distribute error more evenly. The focal point is included so the reader can assess the effectiveness of the reconstruction by comparing with figure 4, which is seen from a different viewpoint; note that, for example, the bases of all boxes are near to coplanar.

hypotheses (and thereby shore up minor failures of the consistency mechanism).

Finally, consistency mechanisms of the type described are most effective in worlds well-populated with familiar objects. We believe that the tremendous potential power of a global consistency analysis will become most important in a system with a large modelbase, operating in a complex world.

Acknowledgements

DAF, JLM and AZ were all supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361. DAF was supported in part by the National Science Foundation under award no. IRI-9209729, and in part by a National Science Foundation Young Investigator Award with matching funds from GE, Eugene Rikel, Rockwell International and Tektronix. JLM was supported in part by by General Electric. AZ was supported in part by ESPRIT BRA project "VIVA". CAR was supported by a grant from General Electric.

References

[1] Binford, T.O., Levitt, T.S., and Mann, W.B., "Bayesian inference in model-based machine vision," in Kanal, L.N., Levitt, T.S., and Lemmer, J.F., *Uncertainty in AI 3*, Elsevier, 1989.

[2] Faugeras, O.D. "What can be seen with an uncalibrated stereo rig?", Proc. European Conference on Computer Vision, 1992.

[3] Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. "Invariant Descriptors for 3-D Object Recognition and Pose," *PAMI*-13, No. 10, p.971-991, October 1991.

[4] Hartley, R.I. "Cheirality invariants," *Proc. DARPA Image Understanding Workshop*, pages 745–753, 1993.

[5] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR88, p.335-344, 1988.

[6] Liu J., Mundy J.L., Forsyth D.A., Zisserman A. and Rothwell C.A., "Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders", *CVPR*, 1993.

[7] Mohr, R., Morin, L. and Grosso, E., "Relative positioning with uncalibrated cameras," in Mundy, J.L and Zisserman, A. (eds) *Geometric Invariance in computer vision*, MIT press, 1992.

[8] Ponce, J. "Invariant properties of straight homogenous generalised cylinders," *IEEE Trans. Patt. Anal. Mach. Intelligence*, **11**, 9, 951-965, 1989.

[9] Rothwell, C.A., Zisserman, A., Mundy, J.L. and Forsyth, D.A. "Efficient Model Library Access by Projectively Invariant Indexing Functions", Proceedings CVPR92, p.109-114, 1992.

[10] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L., "Extracting projective structure from single perspective views of 3D point sets," International Conference on Computer Vision, Berlin, 573-582, 1993.

[11] Sugihara, K., *Machine Interpretation of Line Drawings*, MIT Press, 1986.

[12] Taubin, G. and Cooper, D.B. "Recognition and Positioning of 3D Piecewise Algebraic Objects," Proceeding DARPA Image Understanding Workshop, p.508-514, September 1990.

[13] Ulupinar, F, and Nevatia, R. "Shape from Contour using SHGCs," *Proc. ICCV*, Osaka, 1990.

[14] Weiss, I. "Projective Invariants of Shapes," Proceedings DARPA Image Understanding Workshop, p.1125-1134, April 1988.

[15] Zerroug, M. and Nevatia, R., "Using invariance and quasi-invariance for the segmentation and recovery of curved objects," Proc 2'nd Joint Workshop on the Applications of Invariance in Computer Vision, Ponta Delgada, 1993.

Representations of 3D Objects that Incorporate Surface Markings

David Forsyth[1] and Charlie Rothwell[2]

[1] Department of Computer Science, University of Iowa, Iowa City, Iowa.
[2] Oxford University Robotics Research Group, Parks Rd, Oxford.

Abstract. In many cases, the geometric representation that a recognition system could recover is insufficient to identify objects. When object geometry is simple, it is not particularly distinctive; however, a rich representation can be obtained by mapping the surface markings of the object onto the geometry recovered. If edges are mapped, a representation that is relatively insensitive to the details of lighting can be recovered. Mapping grey levels or color values leads to a highly realistic graphical representation, which can be used for rendering. The idea is demonstrated using extruded surfaces, which consist of a section of a general cone cut by two planes. Such surfaces possess a simple geometry, yet are widespread in the real world. The geometry of an extruded surface is simple, and can easily be recovered from a single uncalibrated image. We show examples based on images of real scenes. **Keywords:** Object recognition, representation, surface markings, invariants.

1 Introduction

Efficient object recognition programs require distinctive object representations; in many applications, such as surveillance, video processing and image databases, only a single image is available. Much work has been done on recovering object geometry from single, uncalibrated images. This paper shows how surface patterns and markings can be recovered and associated with the geometry recovered, and demonstrates a representation, that captures both shape and pattern, for a large class of surfaces covering many man-made objects.

1.1 Recognition Using Indexing

In typical modern systems, recognition proceeds by:

- **Feature extraction:** Edges are extracted and the resulting geometric primitives are grouped into likely object groups (for example, a group of five lines, of two conics, or of a single "M-curve" in [11]).
- **Indexing and hypothesis merging:** Feature groups are used to compute geometric primitives that index the object in a model base (for example, the projective invariants of a pair of conics in [11]). Normally, one object leads to several groups of features indexing the appropriate model, so the resulting hypotheses must be merged using consistency criteria to obtain a single recognition hypothesis.

– **Verification:** Hypothesized objects are back-projected into the image; the results are used to obtain further information that may confirm the hypothesis.

Systems of this sort have been demonstrated for plane objects in a number of papers [3, 11, 4, 7, 13, 16]. Typically, object models consist of a system of invariant values and are therefore relatively sparse, meaning that hypothesis verification is required to confirm a model match. However, no searching of the model base is required because the hypothesised object's identity is determined by the invariant descriptors measured. These systems are attractive because, in the ideal case, an object description is computed from the image and identifies the object, without requiring that a model base be searched. As a result, systems with relatively large model bases can be constructed[3].

1.2 Recognising Curved Surfaces

The indexing systems described assume either that the model base contains only plane or polyhedral objects, or that depth data is available. Recognising curved surfaces in single images presents a particularly difficult problem, as the change in appearance of a curved surface as it is imaged from different viewing positions is not easily analysed. Throughout the paper, we assume an idealised pinhole camera. These cameras possess a *focal point* and an *image plane*. For each point in space, there is a line through that point and the focal point; the point in space appears in the image as the intersection of this line with the image plane. An orthographic view occurs when the pinhole is "at infinity".

If the focal point is fixed and the image plane is moved, the resulting distortion of the image is a collineation. In what follows, it is assumed that neither the position of the image plane with respect to the focal point nor the size and aspect ratio of the pixels on the camera plane is known, so that the image presented to the algorithm is within some arbitrary collineation of the "correct" image. In this model, the image plane makes no contribution to the geometry, and its position in space is ignored.

The *outline* of a surface is a plane curve in the image, which itself is the projection of a space curve, known as a *contour generator*[4]. The contour generator is given by those points on the surface where the surface turns away from the image plane; formally, the ray through the focal point to the surface is tangent to the surface. As a result, at an outline point, if the relevant surface patch is visible, nearby pixels in the image will see vastly different points on the surface, and so outline points usually have sharp changes in image brightness associated with them. Figure 1 illustrates these concepts.

It is generally accepted that the problem of recovering surface geometry from an outline alone is intractable if the surface is constrained only to be smooth, or

[3] Current systems using this approach have model-bases containing of the order of thirty objects.

[4] There are a number of widely used terms for both curves, and no standard terminology has yet emerged.

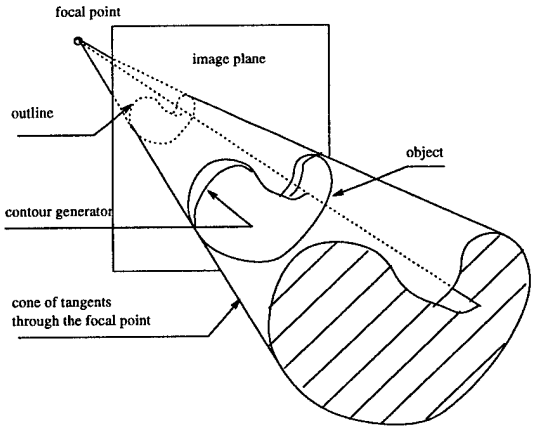


Fig. 1. The outline and contour generator of a curved object, viewed from a perspective camera.

piecewise smooth, as in this case significant changes can be made to the surface geometry without affecting the outline from a given viewpoint. As a result, an important part of the problem involves constructing as large a class of surfaces as possible that can either be directly recognised or usefully constrained from their outline alone. There have been advances in a number of special cases:[5, 8, 2] describe a systems for recognising rotationally symmetric surfaces from outlines, [1, 8, 10, 14, 15] treat straight homogenous generalised cylinders and [6] describes recognising algebraic surfaces from outlines. All these systems emphasize surface geometry in recognition.

Other systems, such as those of [12, 9] emphasize cues such as color and surface lightness over geometry. These cues are particularly effective in dealing with objects whose geometry is ill-defined or hard to describe (such as shirts or jerseys). Such cues cannot, however, be completely divorced from shape, as by themselves they are not particularly distinctive. Difficulties with these cues include perspective effects making it difficult to frame colour features in a truly shape independent fashion (foreshortening can have substantial effects on a colour histogram) and the profound sensitivity of colour and grey-level to illumination.

2 Recovering Extruded Surfaces

An *extruded surface* is a surface formed by a section cut from a general cone by two planes (see figure 2) in such a way that the section of surface does not include the vertex of the cone. This is the projective generalisation of the a surface formed by a system of parallel lines, with plane ends (such a surface can be extruded from a nozzle). Surfaces with this geometry have been the subject of a number of investigations (see, for example, [10, 14, 15], whose more general

theories cover these surfaces as special cases), as they are extremely common, either in themselves or as components of more complex objects - examples include most tin cans, boxes, books, and many plastic bottles.

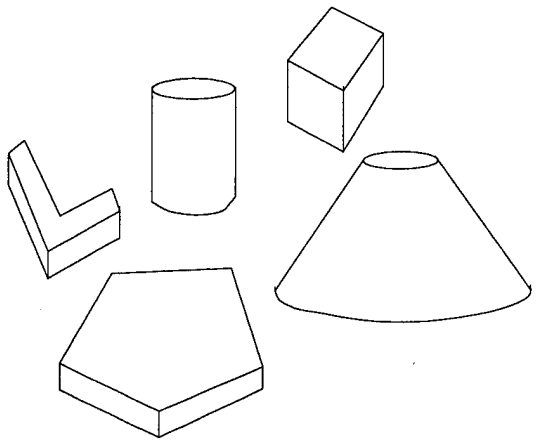


Fig. 2. A range of examples of extruded surfaces; note that for most examples, the vertex is at infinity.

The outline of an extruded surface consists of a system of line segments (possibly empty for some views), the projection of one plane section, and the (possibly occluded) projection of the other plane section; in particular, for all focal points outside a simple "forbidden zone", which lies between the top and bottom sectioning planes and to the object's side of their line of intersection, one or the other plane section is completely visible, and there are usually at least two line segments in the outline.

The projective geometry of an extruded surface can be completely reconstructed in a single image (given the focal point is generic), because it is so simple. Consider one of the plane sections, taken with the line on that plane where the two sectioning planes intersect; call the plane chosen the *defining plane* and the configuration of plane section and line a *defining plane section* or D.P.S. The complete projective geometry of the surface can be represented by a defining plane sections, as it is possible can choose a projective transformation of space that fixes the D.P.S., and transforms the vertex to any given point off the defining plane section and the other sectioning plane, which must pass through the line in the D.P.S., to any given plane not the defining plane nor containing the vertex. As a result, the surface can be reconstructed by taking a D.P.S, choosing a second sectioning plane through the line on the D.P.S. arbitrarily, and choosing an arbitrary vertex in space[5]. The section of surface that

[5] Of course, the second plane must not be a second copy of the defining plane, and the vertex may not lie on either plane.

lies between the planes and does not include the vertex is the reconstruction.

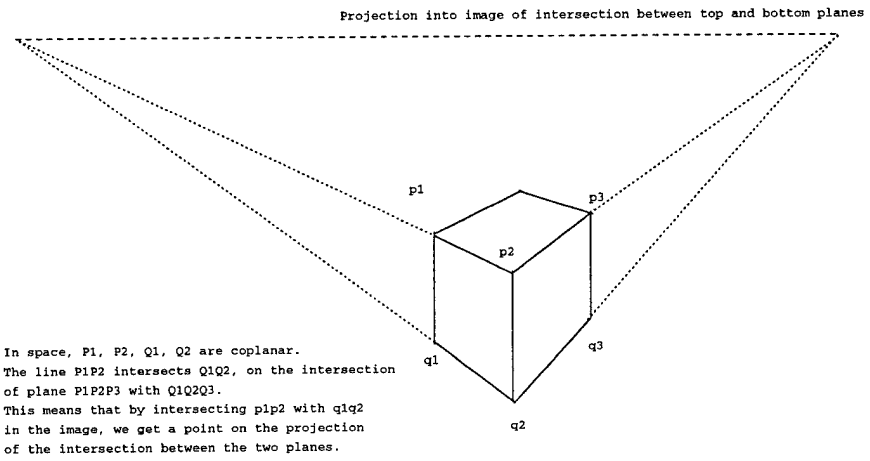


Fig. 3. Determining the projection into the image of the line of intersection of the two sectioning planes.

As at least one plane section is visible in most images, it is possible to recover the surface's projective geometry the projection of the line of intersection between the planes in space can be recovered in the image. This can be done, if the projection of the vertex is known, using the following approach (which is illustrated in figure 3):

- Mark three points, p_1, p_2 and p_3 on one plane section. These correspond to the three coplanar points P_1, P_2 and P_3 in space.
- Construct the corresponding points on the other plane section (for objects with a convex cross-section, self-occlusion is not, in fact, a problem here; see below). Call these points p'_1, p'_2, p'_3, and the corresponding coplanar points in space P'_1, P'_2, P'_3.
- Intersect the image lines p_1p_2, $p'_1p'_2$ to form q_{12} and the lines p_2p_3, $p'_2p'_3$ to form q_{23}. Since P_i, P'_i, P_j, P'_j must be coplanar (the lines $P_iP'_i$ and $P_jP'_j$ are rulings of the surface, and hence pass through the vertex), the intersections in the image are projections of actual intersections in space of the lines P_1P_2, $P'_1P'_2$, etc.
- Since P_1P_2 lies on the top plane, and $P'_1P'_2$ lies on the bottom plane, their intersection must lie on the intersection between the two planes. Thus, the projected line of intersection must pass through q_{12} and through q_{23}.

The aspect of this construction most likely to provide problems for early vision is the identification of corresponding points, which is complicated by self-occlusion. This problem is relatively easy to deal with if the cross-section of

the cone is convex[6]. For such cross-sections, which are well represented in applications, reasoning from a plane cross-section of the object is sufficient. It is apparent that only three possible cases of self-occlusion are significant:

1. one sectioning plane is occluded, and one is visible;
2. both sectioning planes are visible;
3. both are occluded.

Figure 4 shows how each case can occur. In case 3, it is possible to recover the projective geometry of the object only partially; as this case is "unusual" (from the position of the focal point), we concentrate on the other two cases.

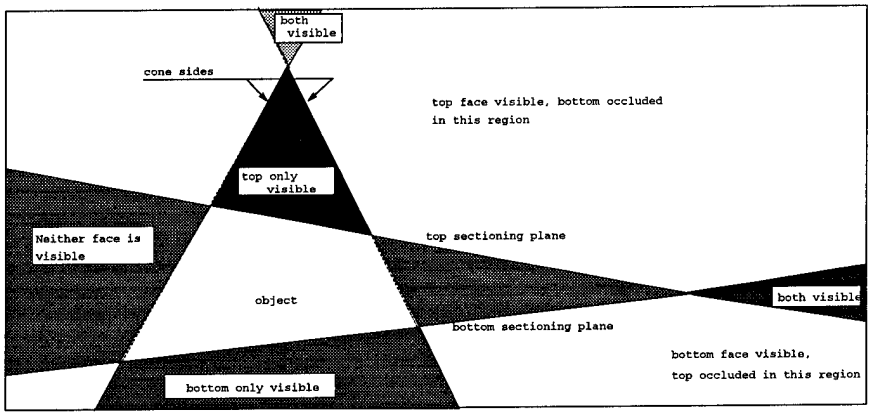
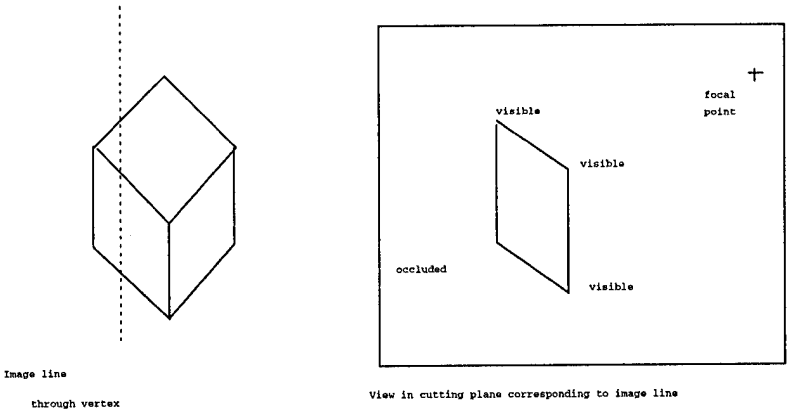


Fig. 4. The possible cases of occlusion of both top and bottom section curves depend on the position of the focal point with respect to the cone, and with respect to the top and bottom sectioning planes; this figure shows the possible cases, for an object viewed in section. Reasoning from a section is sufficient if the cross-section is convex.

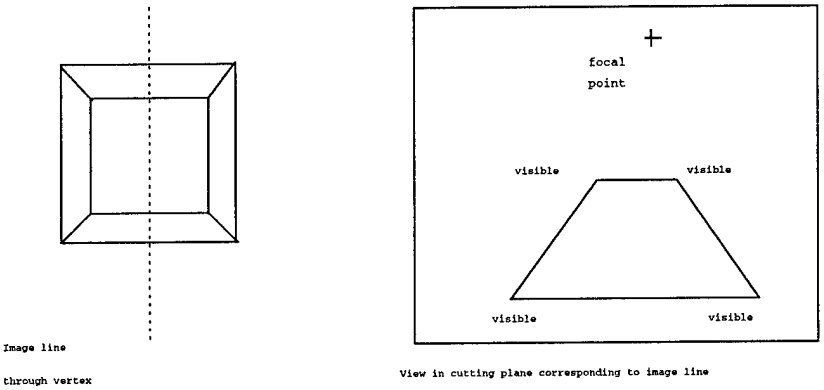
For a convex object, top-bottom correspondences are also easily obtained. Consider a general plane through the vertex and through the focal point; such a plane (which appears in the image as a line), and cuts the top and bottom sections in a number of points, some or all of which are visible in the image as points along a line. The cases are as follows:

– **One section visible:** if (say) the bottom section is occluded, all but one of the points of intersection between the given plane and the bottom section are occluded (figure 5). Hence, the number of visible intersections is odd, and points appear in a cyclic permutation of the order (u_1, u_2, v_2) (where u

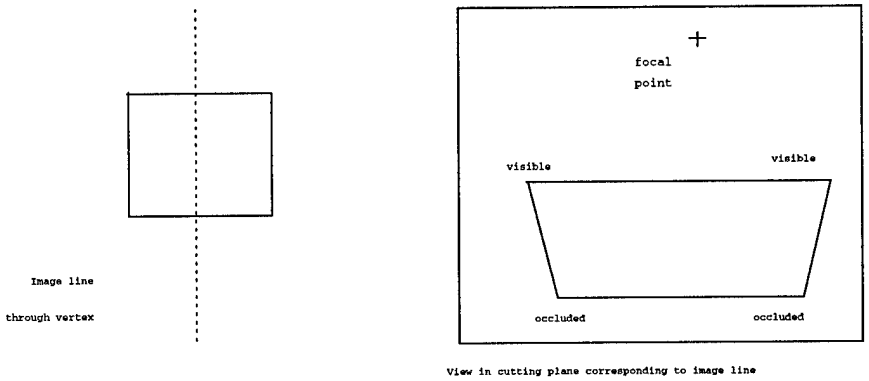
[6] For neatness' sake, we use a definition of convexity compatible with projective transformations; define a region to be convex if its intersection with any line is connected. This definition is broader than the usual definition, but the property is preserved under projective transformations.



Case 1: One section visible, one occluded



Case 2: Both sections visible



Case 3: Both sections occluded

Fig. 5. The effects of occlusion on determining top-bottom correspondences, for a convex cross section and the three cases given in the text.

and v are either top or bottom respectively, and u_i and v_i correspond) along the image line.

- **Both sections visible:** if neither section is occluded, all points of intersection will be visible and there will be an even number of intersections between the outline and the image line, which appear in a cyclic permutation of the order (u_1, u_2, v_2, v_1) along the image line.
- **Both sections occluded:** if both sections are occluded, one point of intersection will be visible for both top and bottom sectioning plane, and there will be two intersections between the outline and the image line, which clearly appear in a cyclic permutation of the order (u_1, v_1) along the image line.

2.1 Obtaining Surface Markings for Extruded Surfaces

To transfer surface markings from the image to the representation of the surface, it is convenient to break up the cross-section in the image into a polygonal approximation (the polygons can be arbitrarily small), and then note that this leads to a representation of the surface as a system of plane quadrilaterals. Finally, the positions of the vertices in the image to which the 3D vertices project, are known. This determines the projective transformation from the image quadrilateral to the surface quadrilateral (four points and their images determine a projective transformation), and this transformation applied to pattern points maps them onto the surface representation (figure 6). Clearly, if this texture mapping process works for grey-level surface markings, it will work for color surface markings as well. A number of color sequences exist, although production expense dictates that the examples given show results only for the grey level case.

3 Experimental Results and Discussion

This scheme has been implemented for images of real scenes. We use a simple manual process to mark outline points in images; in particular, the operator marks the two lines that determine a vertex, points on the top curve, and corresponding points on the bottom curve. This manual interface was used to allow a quick implementation to demonstrate the recovery process; we do not believe that it is essential. The representations extracted are 3D geometrical objects with associated surface markings, and therefore lend themselves well to display as a movie. Figure 7 shows typical images from which models are extracted; figures 8 and 9 show frames from movies of tumbling objects. Note that the texture on the object is stable as the object tumbles, indicating that the surface markings are being correctly extracted and placed; note further that, if the operator chooses a projective frame in which, for example, the soda cans have a circular cross section and roughly the right aspect ratio, the models are impressively realistic. Figure 10 shows two views of a simple 3D world constructed using a tool that places extruded models in space with respect to one another.

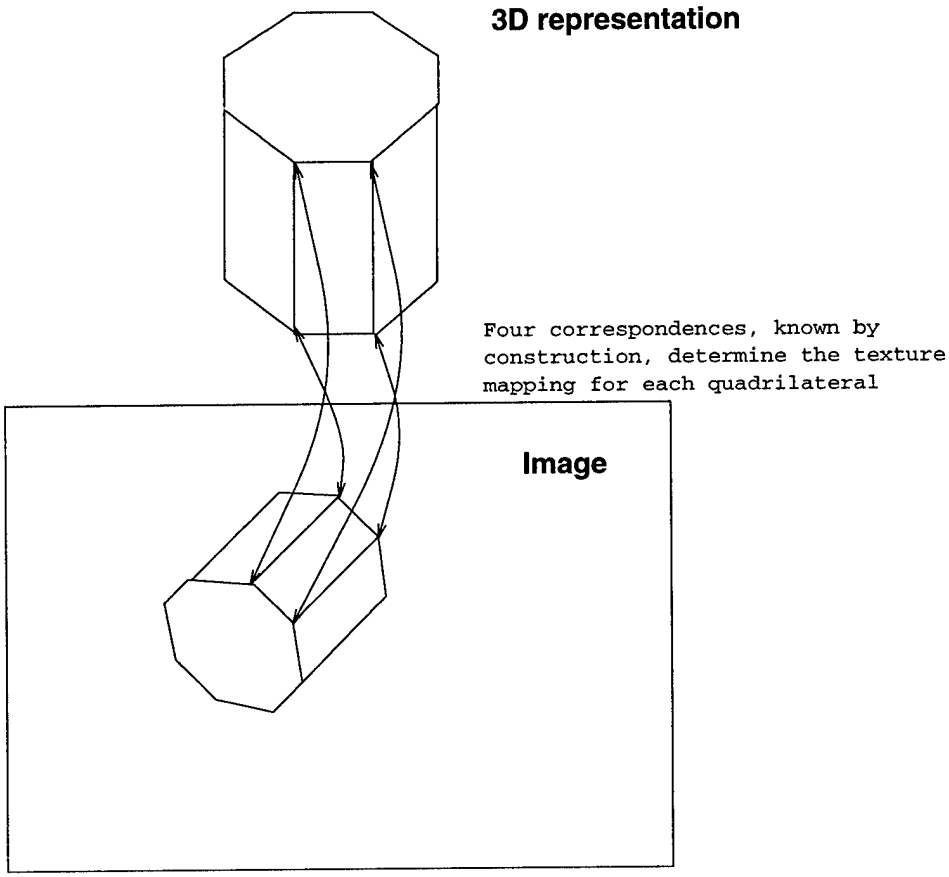


Fig. 6. Patterns and surface markings are transferred to the representation of the extruded surface, using the fact that four known correspondences determine a plane to plane projection. These correspondences are determined by construction.

3.1 Recovering Surface Markings in General

Ideally, a representation of a surface would contain information both about shape and about surface pattern features; in general, edges in surface pattern are likely to be more effective recognition cues due to their relative insensitivity to illumination, but for some applications such cues as pattern colour and lightness may be appropriate. The surface marking information should be provided as a pattern on a representation of the surface (i.e. referred to its position on the surface in some canonical frame[7]), so that foreshortening effects and the like can

[7] A canonical representation of the object's markings is one that is independent of the camera geometry. For instance, a canonical representation of a ruled surface such as a drinks can would be the unwrapped planar surface (modulo starting point), bounded by a rectangle.

Fig. 7. Four typical images, from which models were recovered.

be discounted. If the geometry of an object can be recovered from an image, then it is generally also possible to recover a representation that incorporates surface markings.

Typically, processes that recover geometry from outline information alone in an uncalibrated camera can only recover the projective geometry of an object. This means that an object is associated with a large equivalence class of possible recovered representations, each within a projective transformation of the original object geometry. The property required, that the surface markings be in the "right place" on the surface, is more properly referred to as *covariance*. Covariance in recovering surface markings would mean that, for any two projectively equivalent representations of an object recovered from two images of that object, the projective equivalence extends to the surface markings as well.

It is possible to produce algorithms with this property, given that one allows for occlusion of surface markings. The natural approach, is to compute a camera matrix that takes the inferred object geometry to the image outline; this matrix will be a three by four matrix (homogenous coordinates), whose kernel represents the focal point of the camera, in the model frame. The preimage of an image point in this matrix will be a line in space, passing through the focal point; the intersection, closest to the focal point, between this ray and the object must then be marked with the grey-level or colour in the camera. Thus, transferring surface markings from the image to the representation involves an intersection process, akin to ray-tracing. Because the points obtained by this process are defined by

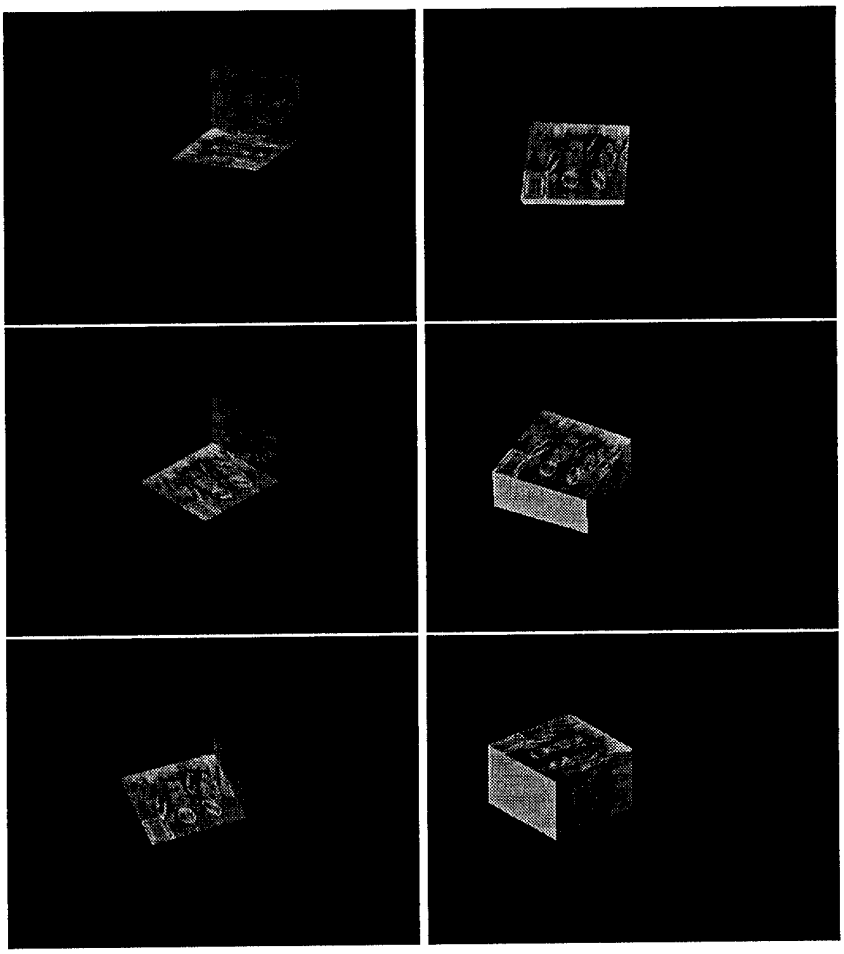


Fig. 8. Six frames from a motion sequence, showing a box for soda cans tumbling.

intersections, the construction is covariant.

The most significant ambiguities that will operate here will be the effects of illumination and self-occlusion. The traditional solution to illumination problems is to use edges, and it is clear that edge maps can be transferred to geometric representations as well as grey-levels. Self-occlusion represents a more interesting problem in determining feature properties.

3.2 Recognition Using Surface Markings

This form of representation can be used to support a hierarchical recognition system, that uses both shape and surface marking information to represent and recognise objects. As the geometry of objects of this form is so simple (as we have seen, it is completely determined by a plane curve and a line), objects can easily

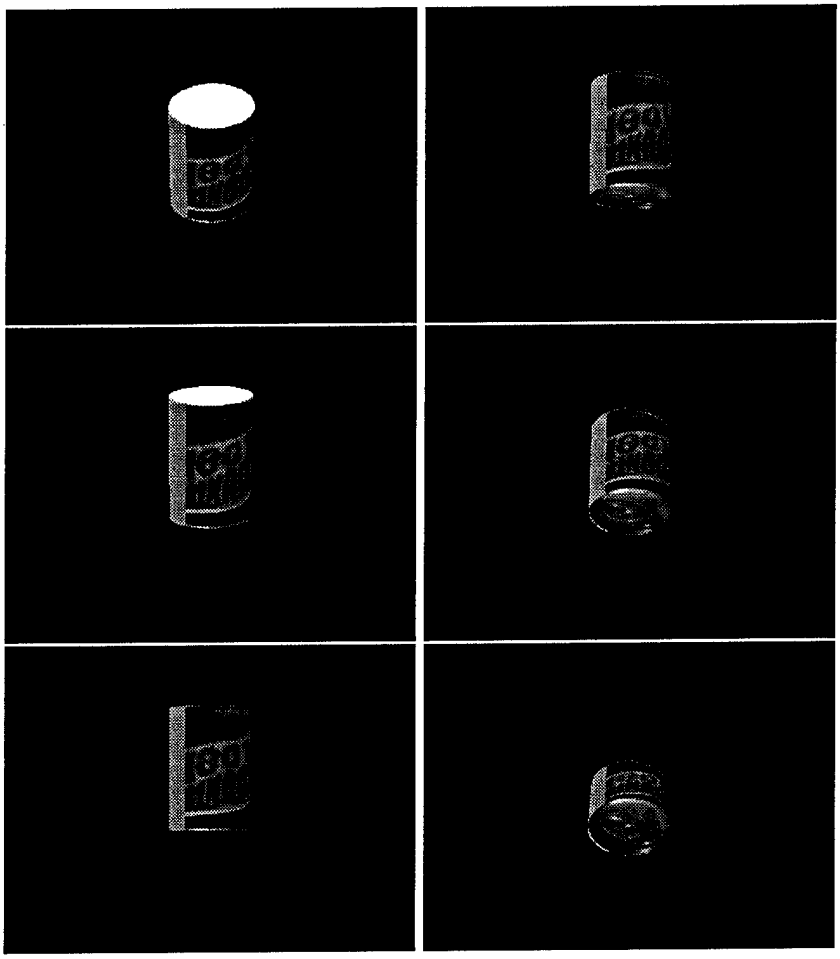


Fig. 9. Six frames from a motion sequence, showing a soda can tumbling backwards.

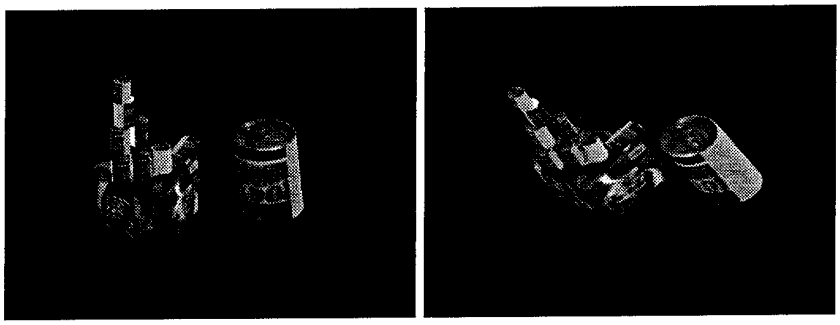


Fig. 10. Two views of a three dimensional world created and rendered using the representations described in the text.

be indexed using a geometrical description. At that point, the surface marking information, which is in a canonical frame, can be used to generate further indexing information for the surfaces using the methods of, for example, Nayar and Bolle [9]. A judicious use of surface marking information is likely to break the projective ambiguity implicit in the geometrical reconstruction - for example, a container for cans of grape soda and a matchbox are projectively equivalent, but have different markings on their faces. Figure 11 shows the canonical representations of face texture for three views of three different object faces.

Geometric representations of extruded surfaces can, therefore, be effectively texture mapped using image data. However, grey-level texture maps are extremely sensitive to illumination details, making them potentially ineffective for recognition. It is possible to texturemap the representation recovered with image edges to overcome this difficulty, and figures 12 to 14 show canonical representations of face edges for a range of surfaces. In general, these views look the same for different views of the same faces and different for views of different faces, and so should allow effective indexing. In fact, constructing an indexing process that works reliably for a large number of faces is not fully solved; much of the difficulty appears to stem from spatial quantisation noise, as when a face is strongly foreshortened, an image pixel may correspond to a large clump of pixels in the canonical representation. Constructing robust indexing techniques that use the surface marking information in these representations is the subject of active research.

Acknowledgements

DAF was upported in part by the National Science Foundation under award no. IRI-9209729, in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361, and in part by a National Science Foundation Young Investigator Award with matching funds from GE, Rockwell International and Tektronix. CAR was supported in part by a grant from GE. This idea has origins in both authors' collaborative work with Joe Mundy and Andrew Zisserman. DAF was encouraged to consider surface markings by a discussion with Shree Nayar. Thanks to David Mumford for helpful and informative conversations. Rodney André built the tool for modeling worlds of extruded surfaces.

References

1. Binford, T.O., Levitt, T.S., and Mann, W.B., "Bayesian inference in model-based machine vision," in Kanal, L.N., Levitt, T.S., and Lemmer, J.F., *Uncertainty in AI 3*, Elsevier, 1989.
2. Dhome, M., LaPreste, J.T, Rives, G., and Richetin, M. "Spatial localisation of modelled objects in monocular perspective vision," *Proc. First European Conference on Computer Vision*, 1990.
3. Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. "Invariant Descriptors for 3-D Object Recognition and Pose," *PAMI*-13, No. 10, p.971-991, October 1991.

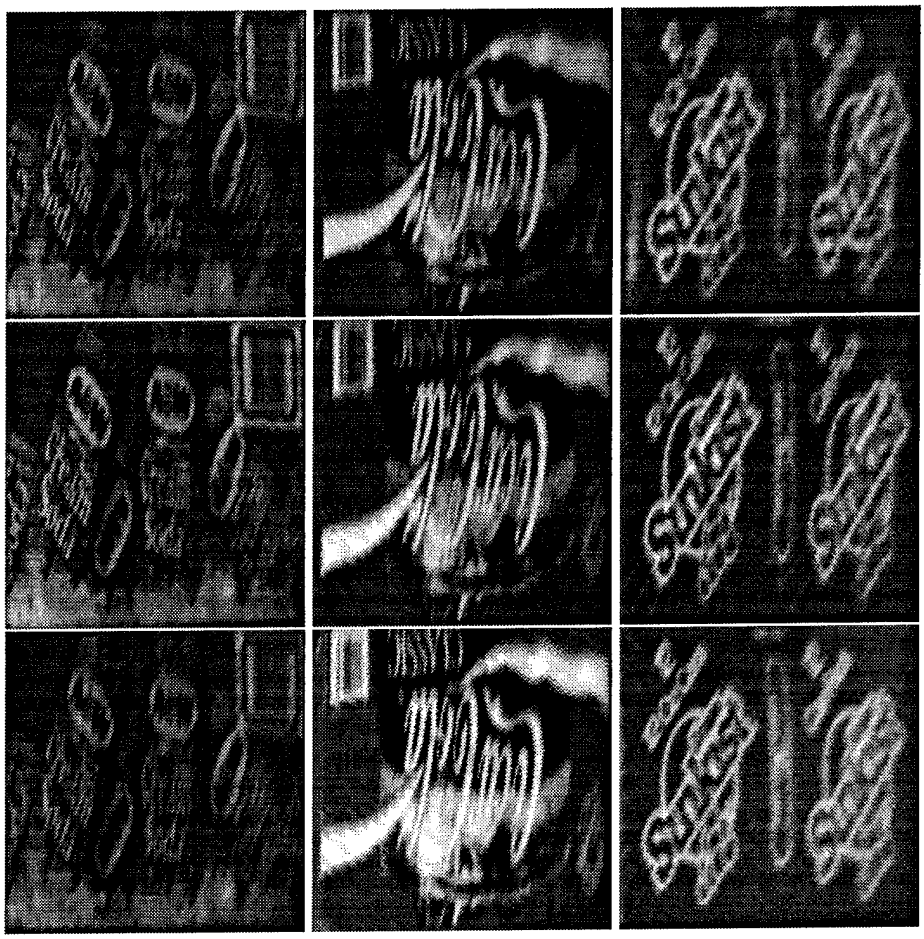


Fig. 11. Canonical representations of face markings for three views of three different object faces; note that different views of the same face look the same, and views of different faces look different.

4. Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. "Applications of invariant theory in vision," In Kapur, D. and Donald, B.R., (eds) *Proceedings Workshop on Integration of Symbolic and Numerical Methods, Saratoga N.Y.*, Academic Press, 1992.
5. Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. "Recognising Curved Surfaces from their Outlines," Proceedings ECCV2, p.639-648, 1992.
6. Forsyth, D.A., "Recognizing Algebraic Surfaces from their Outlines," *Accepted for Publication, International J. of Computer Vision*, 1993.
7. Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR88, p.335-344, 1988.
8. Liu J., Mundy J.L., Forsyth D.A., Zisserman A. and Rothwell C.A., "Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Gen-

Fig. 12. Canonical representations of face edges for three views of of a box in which soda cans are sold. Each column shows three different faces from a single view; different columns correspond to different views. These representations should be compared with those of figures 13 and 14; note that different views of the same face look the same, and views of different faces look different. This information is much richer than the geometry of the object, which is rather simple.

eralized Cylinders", *CVPR*, 1993.

9. Nayar, S.K. and Bolle, R. "Reflectance Ratio: A Photometric Invariant for Object Recognition," *Proc ICCV-4*, Berlin, 1993.
10. Ponce, J. "Invariant properties of straight homogenous generalised cylinders," *IEEE Trans. Patt. Anal. Mach. Intelligence*, **11**, 9, 951-965, 1989.
11. Rothwell, C.A., Zisserman, A., Forsyth, D.A. and Mundy, J.L. "Planar Object Recognition using Projective Shape Representation," to appear, IJCV 1994.
12. Swain, M.J. and Ballard, D.H., "Color Indexing," *International Journal of Computer Vision*, **7**, 1, 11-32, 1991.

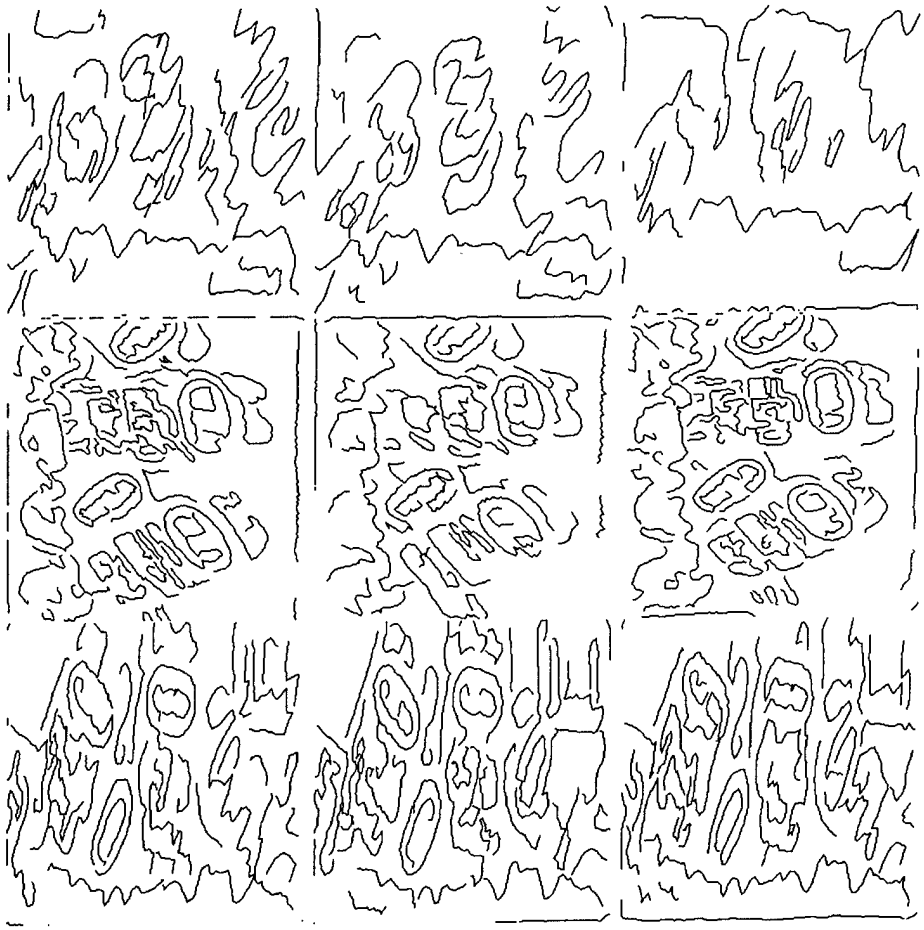


Fig. 13. Canonical representations of face edges for three views of of a box in which soda cans are sold. Each column shows three different faces from a single view; different columns correspond to different views. These representations should be compared with those of figures 12 and 14; note that different views of the same face look the same, and views of different faces look different.

13. Taubin, G. and Cooper, D.B. "Recognition and Positioning of 3D Piecewise Algebraic," Proceeding DARPA Image Understanding Workshop, p.508-514, September 1990.
14. Ulupinar, F, and Nevatia, R. "Shape from Contour using SHGCs," *Proc. ICCV*, Osaka, 1990.
15. Ulupinar, F, and Nevatia, R. "Recovering shape from contour for constant cross-section generalisd cylinders," *Proc. CVPR*, Mauii, 1991.
16. Weiss, I. "Projective Invariants of Shapes," Proceedings DARPA Image Understanding Workshop, p.1125-1134, April 1988.

This article was processed using the LaTeX macro package with LLNCS style

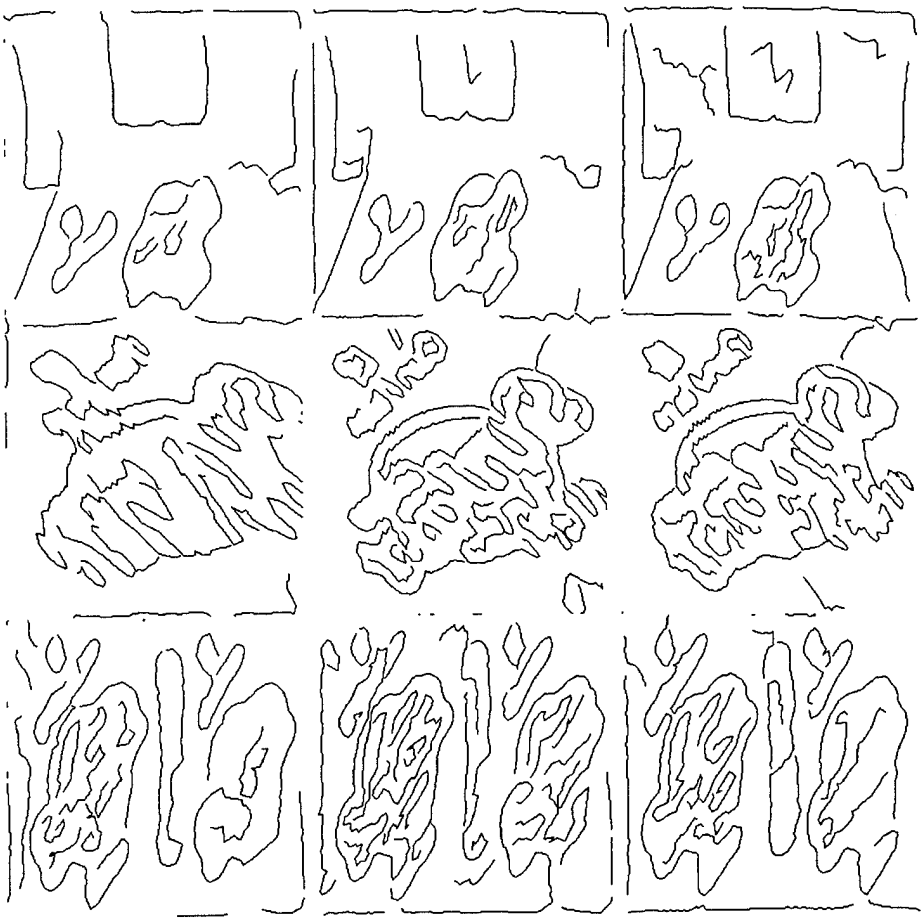


Fig. 14. Canonical representations of face edges for three views of of a box in which soda cans are sold. Each column shows three different faces from a single view; different columns correspond to different views. These representations should be compared with those of figures 12 and 13; note that different views of the same face look the same, and views of different faces look different.

Proc 31st Allerton Conf on Computing, Communication and Control

Recovering Extruded Surfaces from a Single Image

D.A. Forsyth[1] and C.A. Rothwell[2]

Abstract

Extruded surfaces consist of a section of general cone cut by two planes. Such surfaces possess a simple geometry, yet are widespread in the real world. The geometry of an extruded surface can be recovered from a single uncalibrated image. This geometric information can be fleshed out with the surface markings of the object, leading to a highly realistic 3D representation, which can be used for recognition or for rendering. We show examples based on images of real scenes.

1 Introduction

Efficient object recognition programs require distinctive object representations; in many applications, such as surveillance, video processing and image databases, only a single image is available. This paper shows how to recover an object representation that captures both shape and pattern for a large class of surfaces covering many man-made objects.

1.1 Recognition using indexing

In typical modern systems, recognition proceeds by:

- **Feature extraction:** Edges are extracted and the resulting geometric primitives are grouped into likely object groups (for example, a group of five lines, of two conics, or of a single "M-curve" in [11]).
- **Indexing and hypothesis merging:** Feature groups are used to compute geometric primitives to index the object in a model base (for example, the projective invariants of a pair of conics in [11]). Normally, one object leads to several groups of features indexing the appropriate model, so the resulting hypotheses must be merged using consistency criteria to obtain a single recognition hypothesis.
- **Verification:** Hypothesized objects are back-projected into the image; the results are used to obtain further information that may confirm the hypothesis.

Systems of this sort have been demonstrated for plane objects in a number of papers [3, 11, 4, 7, 13, 16]. Typically, object models consist of a system of invariant values and are therefore relatively sparse, meaning that hypothesis verification is required to confirm a model match. However, no searching of the model base is required because the hypothesised object's identity is determined by the invariant descriptors measured, so that systems with relatively large model bases can be constructed (currently, of the order of thirty).

[1]Department of Computer Science, University of Iowa, Iowa City, Iowa.
[2]Oxford University Robotics Research Group, Parks Rd, Oxford

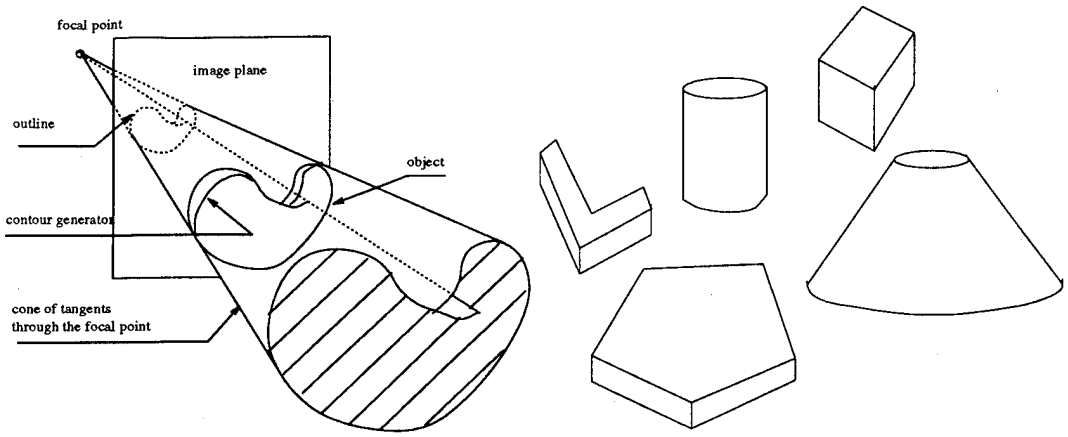


Figure 1: The left hand figure shows the outline and contour generator of a curved object, viewed from a perspective camera. The right hand figure shows a range of examples of extruded surfaces, most with vertex at infinity.

1.2 Recognising curved surfaces

The indexing systems described assume either that the model base contains only plane or polyhedral objects, or that depth data is available. Recognising curved surfaces in single images presents a particularly difficult problem, as the change in appearance of a curved surface as it is imaged from different viewing positions is not easily analysed. Throughout the paper, we assume an idealised pinhole camera. These cameras possess a *focal point* and an *image plane*. For each point in space, there is a line through that point and the focal point; the point in space appears in the image as the intersection of this line with the image plane - figure 1 illustrates such a camera. An orthographic view occurs when the pinhole is "at infinity".

If the focal point is fixed and the image plane is moved, the resulting distortion of the image is a collineation. In what follows, it is assumed that neither the position of the image plane with respect to the focal point nor the size and aspect ratio of the pixels on the camera plane is known, so that the image presented to the algorithm is within some arbitrary collineation of the "correct" image. In this model, the image plane makes no contribution to the geometry, and its position in space is ignored.

The *outline* of a surface is a plane curve in the image, which itself is the projection of a space curve, known as a *contour generator*. The contour generator is given by those points on the surface where the surface turns away from the image plane; formally, the ray through the focal point to the surface is tangent to the surface. As a result, at an outline point, if the relevant surface patch is visible, nearby pixels in the image will see vastly different points on the surface, and so outline points usually have sharp changes in image brightness associated with them. Figure 1 illustrates these concepts.

It is generally accepted that the problem of recovering surface geometry from an outline alone is intractable if the surface is constrained only to be smooth, or piecewise smooth, as in this case significant changes can be made to the surface geometry without affecting the outline from a given viewpoint. As a result, an important part of the problem involves constructing as large a class of surfaces as possible that can either be directly recognised or usefully constrained from their outline alone. There have been advances in a number of special cases:[5, 8, 2] describe a systems for recognising rotationally symmetric

surfaces from outlines, [1, 8, 9, 14, 15] treat straight homogenous generalised cylinders and [6] describes recognising algebraic surfaces from outlines. All these systems emphasize surface geometry in recognition.

Other systems, such as those of [12, 10] emphasize cues such as color and surface lightness over geometry. These cues are particularly effective in dealing with objects whose geometry is ill-defined or hard to describe (such as shirts or jerseys). Such cues cannot, however, be completely divorced from shape, as by themselves they are not particularly distinctive. Furthermore, perspective effects make it difficult to frame colour features in a truly shape independent fashion; for example, foreshortening can have substantial effects on a colour histogram.

Ideally, a representation of a surface would contain information both about shape and about surface pattern features, such as colour and surface lightness. The colour information should be provided as a pattern on a representation of the surface (i.e. referred to its position on the surface in some canonical frame), so that foreshortening effects and the like can be discounted. This paper shows how to construct such a representation from a single image of an extruded surface.

2 Recovering extruded surfaces

An *extruded surface* is a surface formed by a section cut from a general cone by two planes (see figure 1) in such a way that the section of surface does not include the vertex of the cone. This is the projective generalisation of the a surface formed by a system of parallel lines, with plane ends (such a surface can be extruded from a nozzle). Extruded surfaces, and surfaces made up from extruded components, are extremely common - examples include most tin cans, boxes, books, and many plastic bottles.

The outline of an extruded surface consists of a system of line segments (possibly empty for some views), the projection of one plane section, and the (possibly occluded) projection of the other plane section; in particular, for all focal points outside a simple "forbidden zone", which lies between the top and bottom sectioning planes and to the object's side of their line of intersection, one or the other plane section is completely visible, and there are usually at least two line segments in the outline.

The geometry of an extruded surface can be completely reconstructed in a single image (given the focal point is generic), because it is so simple. Consider one of the plane sections, taken with the line on that plane where the two sectioning planes intersect; call the plane chosen the *defining plane* and the configuration of plane section and line a *defining plane section* or D.P.S. The complete projective geometry of the surface can be represented by a defining plane sections, as it is possible can choose a projective transformation of space that fixes the D.P.S., and transforms the vertex to any given point off the defining plane section and the other sectioning plane, which must pass through the line in the D.P.S., to any given plane not the defining plane nor containing the vertex. As a result, the surface can be reconstructed by taking a D.P.S, choosing a second sectioning plane through the line on the D.P.S. arbitrarily, and choosing an arbitrary vertex in space. The section of surface that lies between the planes and does not include the vertex is the reconstruction.

As at least one plane section is visible in most images, it is possible to recover the surface's projective geometry the projection of the line of intersection between the planes in space can be recovered in the image. This can be done, if the projection of the vertex

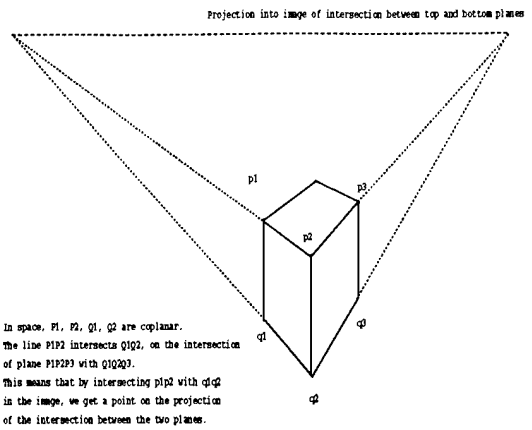


Figure 2: Determining the projection into the image of the line of intersection of the two sectioning planes.

is known, using the following approach (which is illustrated in figure 2):

- Mark three points, p_1, p_2 and p_3 on one plane section. These correspond to the three coplanar points P_1, P_2 and P_3 in space.
- Construct the corresponding points on the other plane section (for objects with a convex cross-section, self-occlusion is not, in fact, a problem here; see below). Call these points p'_1, p'_2, p'_3, and the corresponding coplanar points in space P'_1, P'_2, P'_3.
- Intersect the image lines p_1p_2, $p'_1p'_2$ to form q_{12} and the lines p_2p_3, $p'_2p'_3$ to form q_{23}. Since P_i, P'_i, P_j, P'_j must be coplanar (the lines $P_iP'_i$ and $P_jP'_j$ are rulings of the surface, and hence pass through the vertex), the intersections in the image are projections of actual intersections in space of the lines P_1P_2, $P'_1P'_2$, etc.
- Since P_1P_2 lies on the top plane, and $P'_1P'_2$ lies on the bottom plane, their intersection must lie on the intersection between the two planes. Thus, the projected line of intersection must pass through q_{12} and through q_{23}.

The aspect of this construction most likely to provide problems for early vision is the identification of corresponding points, which is complicated by self-occlusion. This problem is relatively easy to deal with if the cross-section of the cone is convex. For such cross-sections, which are well represented in applications, reasoning from a plane cross-section of the object is sufficient. It is apparent that only three possible cases of self-occlusion are significant:

1. one sectioning plane is occluded, and one is visible;
2. both sectioning planes are visible;
3. both are occluded.

Figure 3 shows how each case can occur. In case 3, it is possible to recover the projective geometry of the object only partially; as this case is "unusual" (from the position of the focal point), we concentrate on the other two cases.

For a convex object, top-bottom correspondences are also easily obtained. Consider a general plane through the vertex and through the focal point; such a plane (which

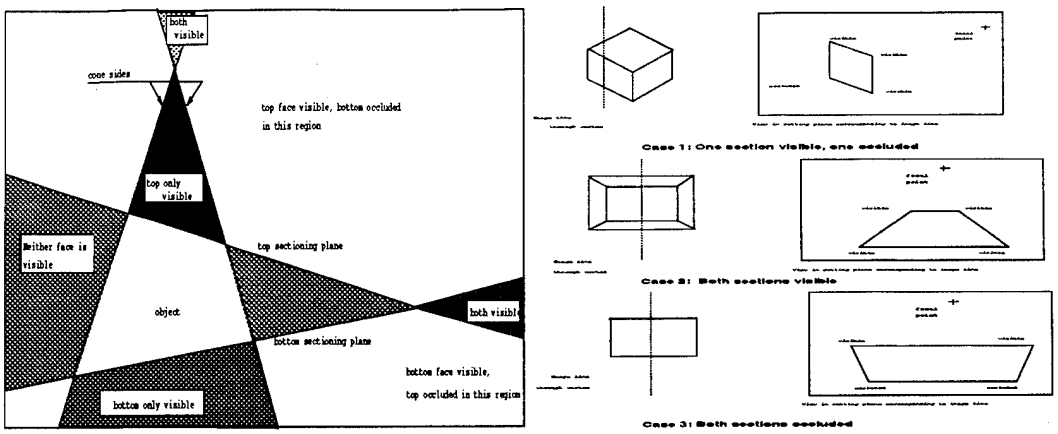


Figure 3: The left hand figure shows how the possible cases of occlusion of both top and bottom section curves depend on the position of the focal point with respect to the cone, and with respect to the top and bottom sectioning planes; this figure shows the possible cases, for an object viewed in section. Reasoning from a section is sufficient if the cross-section is convex. The right hand figure shows the effects of occlusion on determining top-bottom correspondences, for a convex cross section and the three cases given in the text.

appears in the image as a line), and cuts the top and bottom sections in a number of points, some or all of which are visible in the image as points along a line. The cases are as follows:

- **One section visible:** if (say) the bottom section is occluded, all but one of the points of intersection between the given plane and the bottom section are occluded (figure 3). Hence, the number of visible intersections is odd, and points appear in a cyclic permutation of the order (u_1, u_2, v_2) (where u and v are either top or bottom respectively, and u_i and v_i correspond) along the image line.

- **Both sections visible:** if neither section is occluded, all points of intersection will be visible and there will be an even number of intersections between the outline and the image line, which appear in a cyclic permutation of the order (u_1, u_2, v_2, v_1) along the image line.

- **Both sections occluded:** if both sections are occluded, one point of intersection will be visible for both top and bottom sectioning plane, and there will be two intersections between the outline and the image line, which clearly appear in a cyclic permutation of the order (u_1, v_1) along the image line.

In practice, reconstruction proceeds as follows:

1. The cross-section of the cone is determined, by finding the image curve corresponding to the intersection of the cone and the top plane. This curve gives a plane curve within a projective transformation of the original section, and is projected into some arbitrarily chosen plane in space

2. The projection of the vertex is determined in the image, by intersecting line segments in the outline. An arbitrary vertex is chosen in space.

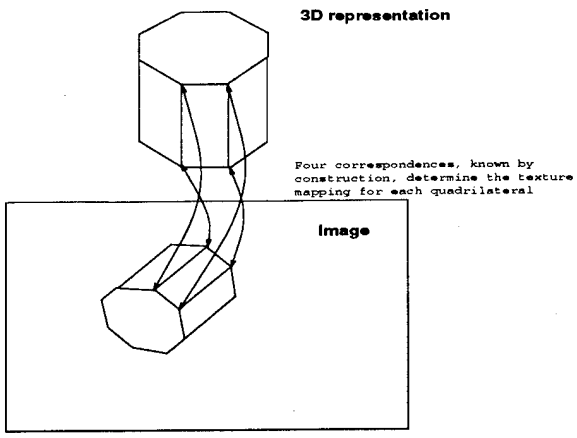


Figure 4: Patterns and surface markings are transferred to the representation of the extruded surface, using the fact that four known correspondences determine a plane to plane projection. These correspondences are determined by construction.

3. The image line that is the projection of the line where the top and bottom planes intersect, is determined (figure 2, and above), and projected onto the same plane as the cross-section, ensuring that the configuration of line and outline is projected as one.

4. The cone is constructed by constructing lines through the vertex and points on the plane cross-section.

5. The second plane is chosen; any plane passing through the line of intersection, and not through the vertex, will do. The lines passing through points on the top cross-section and the vertex mark out the bottom cross-section.

2.1 Recovering surface markings

To transfer surface markings from the image to the representation of the surface, it is convenient to break up the cross-section in the image into a polygonal approximation (the polygons can be arbitrarily small), and then note that this leads to a representation of the surface as a system of plane quadrilaterals. Finally, the positions of the vertices in the image to which the 3D vertices project, are known. This determines the projective transformation from the image quadrilateral to the surface quadrilateral (four points and their images determine a projective transformation), and this transformation applied to pattern points maps them onto the surface representation (figure 4). Clearly, if this texture mapping process works for grey-level surface markings, it will work for color surface markings as well. A number of color sequences exist, although production expense dictates that the examples given show results only for the grey level case.

3 Experimental results and discussion

This scheme has been implemented for images of real scenes. We use a simple manual process to mark outline points in images; in particular, the operator marks the two lines that determine a vertex, points on the top curve, and corresponding points on the bottom

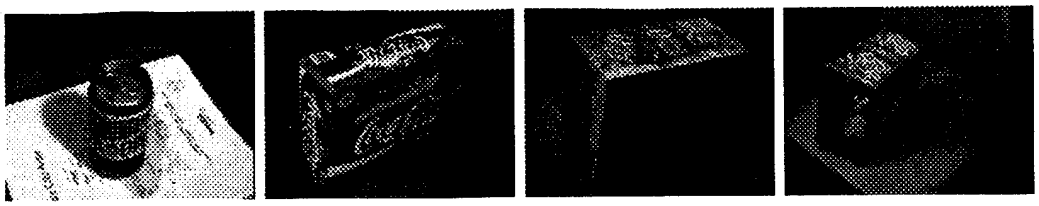


Figure 5: Four typical images, from which models were recovered.

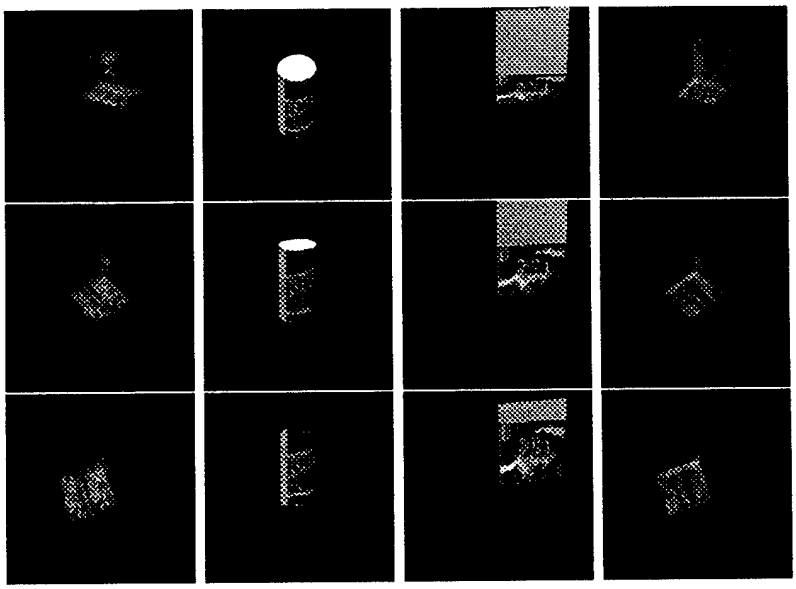


Figure 6: Frames from four motion sequences, created from the representations recovered using the techniques described in the text, showing boxes for soda cans tumbling, and a soda can tumbling. Note that the faces not visible in the image used for making models are rendered blank; constructing an integrated representation from a number of views of an object is an interesting question, at present open. Objects are scaled by hand, and rendered into a canonical frame. Note also the effects of image spatial quantization on one face of the box (which was severely foreshortened in the original image.

curve. This manual interface was used to allow a quick implementation to demonstrate the recovery process; we do not believe that it is essential. The representations extracted are 3D geometrical objects with associated surface markings, and therefore lend themselves well to display as a movie. Figure 5 shows typical images from which models are extracted; figure 6 shows frames from movies of tumbling objects. Note that the texture on the object is stable as the figure tumbles, indicating that the surface markings are being correctly extracted and placed; note further that, if the operator chooses a projective frame in which, for example, the soda cans have a circular cross section and roughly the right aspect ratio, the models are impressively realistic. Figure 7 shows two views of a simple 3D world constructed using a tool that places extruded models in space with respect to one another.

This form of representation can be used to support a hierarchical recognition system, that uses both shape and surface marking information to represent and recognise objects. As the geometry of objects of this form is so simple (as we have seen, it is completely determined by a plane curve and a line), objects can easily be indexed using a geometrical

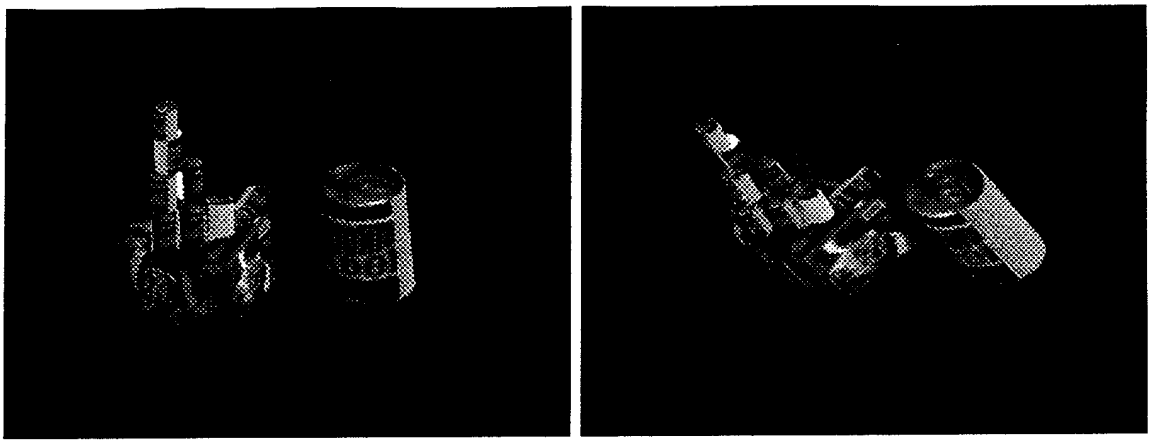


Figure 7: Two views of a three dimensional world created and rendered using the representations described in the text.

description. At that point, the surface marking information, which is in a canonical frame, can be used to generate further indexing information for the surfaces using the methods of, for example, Nayar and Bolle [10]. A judicious use of surface marking information is likely to break the projective ambiguity implicit in the geometrical reconstruction - for example, a container for cans of grape soda and a matchbox are projectively equivalent, but have different markings on their faces. Figures 8 show the canonical representations of face texture for three views of three different object faces. In general, these views look the same for different views of the same faces and different for views of different faces, and so should allow effective indexing. In fact, constructing an indexing process that works reliably for a large number of faces is not fully solved; much of the difficulty appears to stem from spatial quantisation noise, as when a face is strongly foreshortened, an image pixel may correspond to a large clump of pixels in the canonical representation. Constructing robust indexing techniques that use the surface marking information in these representations is the subject of active research.

Acknowledgements

Supported in part by the National Science Foundation under award no. IRI-9209729, in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361, and by a National Science Foundation Young Investigator Award with matching funds from GE, Rockwell International and Tektronix. This idea has origins in both authors' collaborative work with Joe Mundy and Andrew Zisserman. DAF was encouraged to consider surface markings by a discussion with Shree Nayar. Thanks to David Mumford for helpful and informative conversations. Rodney André built the tool for modeling worlds of extruded surfaces.

References

[1] Binford, T.O., Levitt, T.S., and Mann, W.B., "Bayesian inference in model-based machine vision," in Kanal, L.N., Levitt, T.S., and Lemmer, J.F., *Uncertainty in AI 3*, Elsevier, 1989.

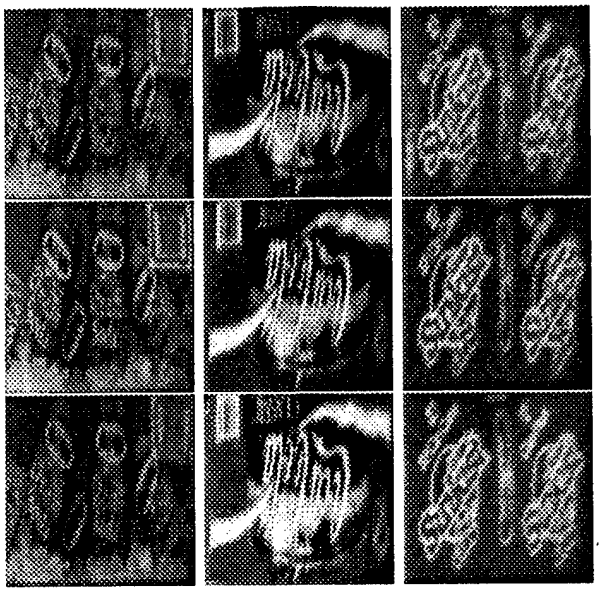


Figure 8: Canonical representations of face markings for three views of three different object faces; note that different views of the same face look the same, and views of different faces look different.

[2] Dhome, M., LaPreste, J.T, Rives, G., and Richetin, M. "Spatial localisation of modelled objects in monocular perspective vision," *Proc. First European Conference on Computer Vision*, 1990.

[3] Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. "Invariant Descriptors for 3-D Object Recognition and Pose," *PAMI*-13, No. 10, p.971-991, October 1991.

[4] Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. "Applications of invariant theory in vision," In Kapur, D. and Donald, B.R., (eds) *Proceedings Workshop on Integration of Symbolic and Numerical Methods, Saratoga N.Y.*, Academic Press, 1992.

[5] Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. "Recognising Curved Surfaces from their Outlines," Proceedings ECCV2, p.639-648, 1992.

[6] Forsyth, D.A., "Recognizing Algebraic Surfaces from their Outlines," *Accepted for Publication, International J. of Computer Vision*, 1993.

[7] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR88, p.335-344, 1988.

[8] Liu J., Mundy J.L., Forsyth D.A., Zisserman A. and Rothwell C.A., "Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders", *CVPR*, 1993.

[9] Ponce, J. "Invariant properties of straight homogenous generalised cylinders," *IEEE Trans. Patt. Anal. Mach. Intelligence*, **11**, 9, 951-965, 1989.

[10] Nayar, S.K. and Bolle, R. "Reflectance Ratio: A Photometric Invariant for Object Recognition," *Proc ICCV-4*, Berlin, 1993.

[11] Rothwell, C.A., Zisserman, A., Mundy, J.L. and Forsyth, D.A. "Efficient Model Library Access by Projectively Invariant Indexing Functions", Proceedings CVPR92, p.109-114, 1992.

[12] Swain, M.J. and Ballard, D.H., "Color Indexing," *International Journal of Computer Vision*, **7**, 1, 11-32, 1991.

[13] Taubin, G. and Cooper, D.B. "Recognition and Positioning of 3D Piecewise Algebraic," Proceeding DARPA Image Understanding Workshop, p.508-514, September 1990.

[14] Ulupinar, F, and Nevatia, R. "Shape from Contour using SHGCs," *Proc. ICCV*, Osaka, 1990.

[15] Ulupinar, F, and Nevatia, R. "Recovering shape from contour for constant cross-section generalisd cylinders," *Proc. CVPR*, Mauii, 1991.

[16] Weiss, I. "Projective Invariants of Shapes," Proceedings DARPA Image Understanding Workshop, p.1125-1134, April 1988.

Distinctive representations for the recognition of curved surfaces using outlines and markings

David Forsyth[1], Andrew Zisserman[2] and Jitendra Malik[1]

[1] Computer Science Division, U.C. Berkeley, Berkeley CA 94720, USA
[2] Robotics Research Group, Oxford University, Oxford, UK

Abstract. Recognising 3D objects from single images presents a range of significant problems, mostly to do with the nature and distinctiveness of the representations that can be recovered. In such special cases as polyhedra, surfaces of revolution, general cones, canal surfaces and algebraic surfaces, the geometry can be recovered with varying degrees of success and of ambiguity. We discuss these volumetric primitives, comparing their utility to that of surface primitives.

For a model based recognition system, representation is not simply concerned with particular geometric primitives, but the entire recognition process. In our view, representations should be motivated by the way quantities that are measurable in an image influence decisions throughout the recognition process.

When some geometric information is available, its potential distinctiveness can often be substantially enhanced by constructing representations that capture surface markings in an appropriate frame on the surface itself.

Keywords: Object Recognition, Computer Vision, Invariant Theory, Surface Representation, Surface Markings.

1 Introduction

The fundamental question in discussing representations is "What is the representation for?". Representations, such as depth maps and Bezier patches, that may be appropriate for tasks such as geometric modelling, graphics, navigation, or grasping may not be appropriate for model based recognition. In this paper we explore representations for object recognition, with a particular emphasis on representations for curved, 3D objects that can be extracted from a single image.

For recognition systems to be successful and useful, they will have to have large modelbases, containing a wide variety of objects. For systems with large modelbases, searching over object-image correspondences to estimate pose and then verify, is impractically expensive. Existing recognition schemes where recognition is phrased in this way include *pose-clustering* [35], *alignment* [16, 17], and *interpretation trees* [15]. Fortunately, *object recognition is not pose recovery*; representations can be defined which yield *viewpoint-invariant* descriptions from images. Such invariant descriptors directly identify models, without first computing pose, avoiding searching the model base. As a result, defining and managing appropriate object representations is the central question in building object recognition systems.

If the representation is too impoverished then it will not admit object-independent constructions that take image data alone and yield object identity, as recent papers have shown [4, 5, 22]. A number of the pose based schemes cited above are forced to search their model libary because the motivating notion of objects, as semi-coherent clouds of points or line-segments, is too parsimonious: clouds of points are a poor representation for most objects, because representing an object as a cloud of points wastes the available and potentially rich structure of outline and of markings.

It is convenient to distinguish between two basic types of representation:

- *Volumetric primitives*, which are drawn from globally constrained geometries such as generalized cylinders and algebraic surfaces, and can represent, for example, vases, cans and boxes. A great deal is known about extracting invariant representations of volumetric primitives.
- *Surface primitives*, which consist of patches of surface, defined by their curvatures for example, and subject to little or no constraint - they represent shirts, Henry Moore sculptures, and trees. Very little is known about what information should be extracted or what is available.

Typically, representations should (and occasionally do) include information both about the geometry of objects, and about the markings that lie on the objects. Geometry and markings are, to some extent, interdependent; measurements of markings are unreliable unless referred to some coordinate system on a curved surface itself (to avoid the effects of foreshortening), and markings generate image clutter that obstructs grouping. Admitting markings is essential, because it expands the scope of any representational scheme - it is hard to distinguish between soft-drink cans without exploiting marking information.

In what follows, we use the following terms:

- **markings:** patterns of reflectance changes on a surface - for example, the patterns on the cover of a book.
- **texture:** markings that are more structured or statistical patterns of reflectance changes - for example, the patterns on a wooden desktop.
- **outline:** the points in an image plane where the surface is tangent to the ray through the focal point and the image point.
- **contour generator:** the points on the surface that project down to the outline.

2 What is known

2.1 Indexing

A system that must handle large numbers of models requires *indexing*, where a process that is wholly or largely model-independent, is used to compute representations that are largely or wholly unaffected by the position and intrinsic parameters of the camera, and that differ from object to object. These descriptions, often known as *indexing functions*, have the same value for any view of

a given object, and so can be used to index into a model base without search (for example, [5, 7, 18, 28, 29, 31, 34, 39, 41]). Note that schemes where the model must be known to compute an index (e.g. [40]) do not escape searching the modelbase, and cannot be seen as solving the indexing problem in a useful way.

Invariant representations are a natural goal, and have a long tradition in vision. The evolution of technique in representation in the last decade or so can partly be charted by the extent to which the representation is invariant: the tangent angle vs. arc length representation [1] is invariant under *similarity* transformations, which correspond to presenting an object in a fronto-parallel plane - a very restricted viewing geometry; later, Lamdan *et al.* introduced *affine* invariant representations, corresponding to parallel projection with unknown intrinsic parameters [18]; and then *projectively* invariant representations [7] covered the most general transformation between a planar object and image, with no restrictions on pose or intrinsic parameters. Indexing using invariants yields an attractively simple architecture: in a typical system that works for *plane* objects, projective invariants are computed for a range of geometric primitives in the image; if the values of these invariants match the values of the invariants for a known model, we have good evidence that the image features are within a camera transformation of the model features, and that hypothesis can be either combined with other hypotheses, or verified directly. The efficiency of this indexing process means that systems with non-trivial sizes of model base can be constructed[3].

It must be possible to extract indexing functions from images under realistic conditions. For this reason they cannot be too local, as this compels the use of high derivatives which cannot be measured locally, or too global, and thus dependent on all features being present and grouped. There has been a trend, analogous to the invariance of representations described above, of moving away from global descriptions such as moments and Fourier descriptors (both which are hopelessly inadequate when features are missing - either due to occlusion, or to the ever-present problem of "drop out" with feature detectors) through to modern semi-local descriptors (e.g. [37]).

2.2 Volumetric primitives

In the computer vision literature, there is a long tradition in the use of volumetric primitives, generalized cylinders being the strongest current. Knowing that one is looking at a part of a volumetric primitive characterized by a small number of parameters is a very powerful constraint. An analogy with statistical inference is appropriate–if the data can be well modeled as arising from a parametric model of one of the standard types–the right strategy is to use the data to estimate the (small number of) model parameters and then use the model to answer the various questions one might have.

[3] Current systems using indexing functions have model-bases containing of the order of thirty objects.

In contexts when the objects are well modeled by volumetric primitives, not much benefit is derived from aspect graphs. Given that there exists a model with a small number of parameters, the multiple views are redundant.

Of course, just as parametric statistics is appropriate only for a subset of real world situations, simple volumetric primitives are not general enough to deal with the wide variety of man-made and natural objects - crumpled newspaper is one good example.

One approach to indexing a 3D object would be to determine a comprehensive reconstruction of its 3D geometry, and then abstract geometric invariants from that representation; in general, this is not possible. In some cases, for example, surfaces of revolution, complete reconstructions are definitely unavailable in a single image (some parameters are unresolved). However, for a number of volumetric primitives it is possible to extract index functions, from a single image, which measure some of the object's 3D geometric properties:

- **Polyhedra:** For polyhedra that are position free, and have "many" quadrilateral faces, projective invariants can be recovered from a single image ([30], after [32]); these invariants are measurable because of the rich incidence structure of these polyhedra.
- **Repeated structures:** A single view of an object with an n-fold reflectional symmetry is equivalent to n views of a section of that object; for polyhedral objects and space curves, this has been used to reconstruct the projective geometry of such objects [24, 30].
- **Surfaces of revolution (SOR):** Cross-ratio's of a series of points, defined by outline bitangent lines, on the imaged axis of a SOR, yield projective invariants of the surface [9, 19]. A further construction allows outlines to be transferred from view to view [36], but the information available in the image is insufficient to reconstruct the surface (two further parameters are required).
- **Straight homogeneous generalised cylinders:** It is possible to reconstruct some SHGC's from image information alone [42]; the ambiguity in the reconstruction is not specified, but appears to be at least an elation through the focal point.
- **Algebraic surfaces:** The complete projective geometry of a generic algebraic surface can be recovered from its outline in a single view through a generic focal point; the algorithm given is too complex for practical use [6].
- **Canal surfaces:** A canal surface is a generalised cylinder where the swept curve is a circle of constant radius, with the plane of the circle orthogonal to the axis curve. In the case of a planar axis, the axis curve can be recovered, from the outline alone, up to an affine ambiguity [26]. Consequently, affine invariants of the axis curve can be used as index functions, but the surface can only be recovered up to an affine ambiguity.

2.3 Surface primitives

For a representation to be effective for recognition, it must be an *abstraction* of the surface description produced by throwing away *irrelevant detail.* A pointwise

map of principal curvatures is not much more useful than a pointwise map of normals or depths.

A major program of research in trying to understand surfaces at a higher level of abstraction was initiated by Koenderink and Van Doorn, paralleling work in mathematics on singularity theory. Essentially they tried to understand visual events: how does the topology of the outline change for an observer moving around the object? This leads naturally to the notion of aspect graphs. This line of attack has been pursued to its logical conclusion– the development of algorithms and understanding the computational complexity is now complete for both polyhedra[14] and curved objects[27]. The computational complexity has proved to be very high, making the practical use of exact aspect graphs of general curved surfaces unlikely. An even more serious criticism is that the abstraction is the wrong kind of abstraction. The formalism doesn't take into account the *scale* of the different topological changes of the outline. For a human observer, and a computer vision system that has to start from a brightness image, the scale of the feature has a major impact on its detectability and localizability.

While the usefulness of multiple view representations remains debatable, it seems clear that a concern for what one can hope to reliably extract from early vision needs to inform the choice of representation formalism. From a single image, outline, texture and shading are the main cues available. Texture cues can be treated locally [21] so when we are lucky enough to have visually resolvable texture, we can locally get surface normals and curvatures. It is the shading cue that is more difficult to analyze; global interactions due to interreflections [8] make Horn style shape-from-shading theory unusable in general contexts. As of present writing, surface primitives derived from image data have not been successfully applied in recognition, and, as it is unlikely that this position will change in the short term, we do not treat surface primitives further.

2.4 Markings

There have been few attempts to exploit surface markings explicitly in recognition; most such attempts completely exclude geometry [25, 33]. Some of the difficulties usually cited include variations in colour caused by illuminant effects or interreflections, difficulty of segmenting marked objects, and the effects of foreshortening on markings. However, it is easily shown that, if the geometry of an object can be recovered up to a projective ambiguity from a single view, then the markings on that object can be covariantly recovered by mapping image grey-levels back on to the geometry recovered[4]. This observation has been used to produce representations that incorporate markings (in [11]), but no strategies for exploiting the markings were proposed.

Recognition of 2D textures is a well-studied problem with numerous successful applications in remote sensing, inspection etc. Usually it is approached with

[4] This follows because the process of mapping texture back involves intersecting lines through the focal point with the surface's geometry; since forming intersections is covariant (preserved) under projective ambiguities, backprojecting image texture must be covariant under projective ambiguities.

statistical classification techniques (see e.g. [12] for a review) given a suitable set of texture features. Various features have been used in the literature, including some derived from co-occurrence matrices, Fourier domain features, and those based on convolving the image with multiscale, multi-orientation linear filters.

3 Fundamental notions

Indexing is a concept so basic to recent work on representation that it had to be expounded (above, section 2.1) before the work was discussed. Clearly, one cannot build a large fast recognition system without some form of indexing, and this gives some clues to appropriate areas for future research. However, indexing alone will not answer all the difficulties of utilising a large model base with a wide variety of objects. Some form of *structuring* or organisation is required at every level of the recognition process. This organisation is centred on the notion of object **class**.

3.1 Class

The collection of known volumetric cases (SOR, algebraic ...) above induces a natural, utilitarian, *geometric* notion of class [43]: At early stages, constraints derived from the particular class are used to guide segmentation and grouping; At later stages viewpoint invariant or *viewpoint stable* descriptors for that class are computed from these groups; Index functions are computed from these descriptions and paired with appropriate model class sub-libraries, and so on. In this way an appropriately structured representational system can be used to control complexity at all levels in the recognition process.

Using model classes in this way can be thought of as a modern version of Shape from X, where X is now a model class. In an update of Binford's use of models [2, 3], we use model class to facilitate three distinct tasks:

1. **Grouping**
   Recent work by Zerroug and Nevatia shows that one can recover volumetric SHGC descriptions from fragments of the outline in the presence of occlusion, clutter and extraneous edges [42]. For a surface of revolution this grouping can be accomplished without requiring a cross-sectional curve in the image [43]. The grouped outline also enables simpler computation of the next two levels.
2. **Invariant Representation**
   Using the outline alone, 3D surfaces can often only be recovered up to a parameterised family of surfaces that could have projected to the outline. In contrast, an invariant representation can be obtained which is sufficient for recognition. For example, the axis cross-ratios obtained from corresponding outline bitangent for SORs are invariant to projective transformations, and can be used as index functions, but the surface is not determined.

3. **Recovery of 3D shape**
   With additional information, e.g. camera calibration, both pose and 3D shape can be determined. For example, for a SOR if a cross-sectional circle is visible and the camera is calibrated, pose is determined up to a one parameter family (distance from the camera), and the shape of the surface recovered from the outline up to an overall scaling.

This notion of class is organised around measurable image cues. There is no point in drawing class distinctions that cannot be measured in images; furthermore, an ideal notion of class involves a notion of *emerging object identity*, where the behaviour of later recognition modules is conditioned by class hypotheses established earlier. For example, given that an image line is likely to have come from a polyhedron rather than from a surface of revolution, it is more productive to organise the grouping strategy around constructing object faces.

3.2 Consistency

Typically, in systems that use indexing, recognition hypotheses are not monolithic; several image groups may index to object substructures. It is then essential to determine which of these hypotheses can be fused into an hypothesis about object identity, requiring a notion of consistency. As a second example, hypotheses about object identity are equivalent to hypotheses about such matters as camera internal parameters and illuminant colour, and some pairs of hypotheses about object identity may be mutually incompatible. By enforcing compatibility between these hypotheses, we may extract information about the camera [10] or the illuminant.

Consistency appears to offer mechanisms by which the geometry of unknown volumetric primitives can be constrained, using reconstructions of the known objects surrounding them. Without a modelbase, image data can constrain an object's Euclidean geometry to at best a four parameter ambiguity[5]. This ambiguity can be constrained further by the use of, say, such cues as occlusion, support, or the approximate size of typical objects.

3.3 Richer descriptors

It is a commonplace that richer descriptors should make recognition easier. The basis for this argument is simple; the number of cells in an indexing table goes up exponentially with the dimension, meaning that, in principle, cramped tables can be made spacious by making another measurement. However, success in constructing such descriptors has been generally poor. Cues such as colour and markings, which are widely recognised as having great potential, have not yet

[5] This is the family of projective transformations that fixes the focal point and the cone of rays joining the object to the focal point; as a result, if this transformation is applied to the world, the image is unaffected, which means it is a fundamental ambiguity.

been used effectively. The present generation of volumetric representations can be made richer in three ways: by increasing the geometric detail recovered, by recovering more distinctive geometric measurements, and by associating texture and marking cues with the geometry recovered.

Increasing geometric detail is not a promising strategy; as the history of 3D from 3D matching shows, a significant component of representation involves identifying and disposing of irrelevant information. For volumetric primitives, understanding how to recover more distinctive measurements and understanding which measurements are more stable and effective in indexing represents an important, largely open, problem. Referring markings to a surface coordinate frame (as in figure 1), and measuring descriptors of object patterns in that frame, will provide substantially increased richness in representation.

4 What should be done

Effective notions of class and of indexing are essential for building large model-based vision systems; class, because it can be used to organise both the modelbase and the segmentation process, and indexing, to restrict search of the modelbase. These areas represent those in which we believe the most concrete progress is possible. Progress in indexing is likely to be most concrete: it remains important to expand and strengthen notions of the volumetric primitives for which indexing functions can be constructed; to incorporate surface markings into the construction of indexing functions; and to generate a reasonable theory of how to choose amongst possible indexing options.

The extent to which surface markings have been neglected to date is surprising, as they present the best way to expand the size of modelbases possible with currently understood volumetric primitives. We see two approaches to managing the effects of foreshortening in images of markings: constructing pose-invariant features that describe markings; or, using the inferred geometry to refer the image detail back to a surface coordinate frame. The second approach is more general. The neglect of markings is particularly puzzling because the problems presented by markings appear to be concrete and accessible with present knowledge.

To conclude: We have argued in this paper that representation is not simply a matter of whether a particular primitive (super-quadric, SHGC, etc.) is used. Instead, representation in visual recognition is the entire *process* from the early stages of segmentation and grouping, through the extraction of viewpoint-invariant descriptions, hypothesis combination, to finally the recovery of 3D shape. The implementation of this process that we suggest, based on current achievements, is a *geometric* notion of class using volumetric primitives.

Acknowledgements

Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361, in part by the National Science Foundation under

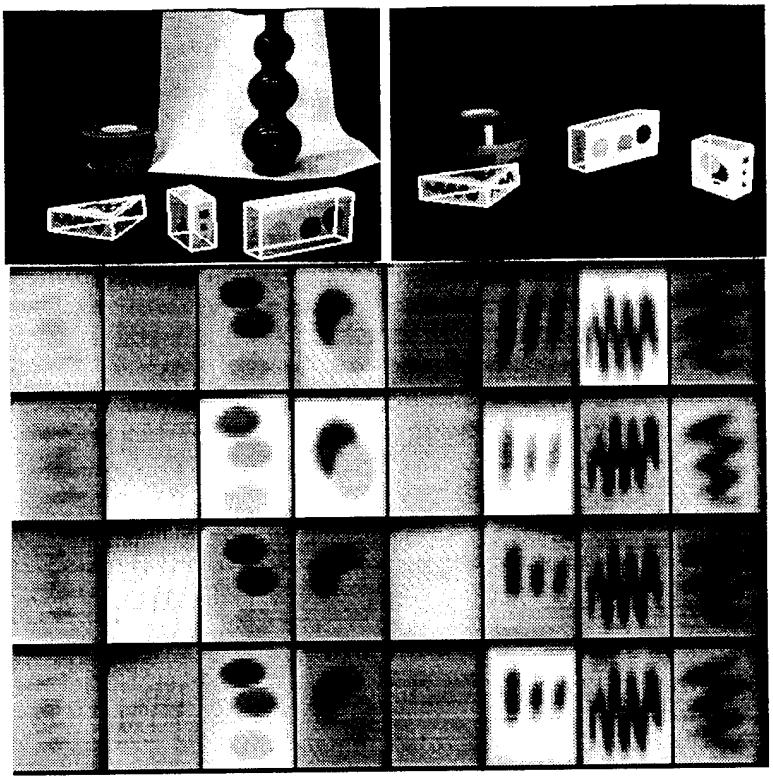


Fig. 1. *The figures on the top show groups of polyhedral objects with polyhedral snakes overlaid by the grouping process. Note the internal boundaries that have been inferred automatically. The markings within the quadrilateral faces are then projectively mapped to a canonical frame. The figure below shows faces in a canonical frame for four views each of eight distinct faces. Note that the resulting sets of markings are fixed (up to the symmetries of the square), meaning that iconic matching of markings is relatively easy. Furthermore, in each case the markings referred to a canonical frame are stable for changes in viewpoint, but largely distinct from face to face. Note the effects of illumination for different views of the same face. Geometrically stable marking data of this form is relatively easily matched using an iconic matcher. Not every object face needs to appear in the modelbase for a face match to be useful; faces are omitted from the modelbase when their markings are not distinctive.*

award no. IRI-92-09729, in part by the European ESPRIT Project 6448 'VIVA', in part by the National Science Foundation under award no. IRI-92-02129, and in part by a National Science Foundation Young Investigator Award with matching funds from GE, Tektronix, Rockwell and Eugene Rikel.

References

1. Asada, H. and Brady, M., "The curvature primal sketch," *PAMI-8*, 1, 2-14, 1986.

2. Binford, T.O. "Inferring Surfaces from Images," *Artificial Intelligence,* Vol. 17, p.205-244, 1981.
3. Binford, T.O., "Survey of model-based image analysis systems," *IJRR*, **1**, 1, 18-63, 1982.
4. Burns, J., Weiss, R. and Riseman, E., "View variation of point set and line segment features," *Proc DARPA IU workshop*, 1990.
5. Clemens, D.T. and Jacobs, D.W. "Model Group Indexing for Recognition," Proceedings CVPR, p.4-9, 1991, and *PAMI*-13, No. 10, p.1007-1017, October 1991.
6. Forsyth, D.A., "Recognising algebraic surfaces by their outlines," *Int. J. Computer Vision*, in press.
7. Forsyth, D.A., Mundy, J.L., Zisserman, A., Heller, A., Coehlo, C. and Rothwell, C.A., "Invariant Descriptors for 3D Recognition and Pose," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, **13**, 10, 1991.
8. Forsyth, D.A., and Zisserman, A., "Reflections on Shading," *Special Issue of I.E.E.E. Pattern Analysis and Machine Intelligence on physical modelling in computer vision*, July, **13**, 7, 671-679, 1991.
9. Forsyth, D.A., Mundy, J.L., Zisserman, A. and Rothwell, C.A., "Recognising rotationally symmetric surfaces from their outlines," *Proc. Second European Conference on Computer Vision*, G. Sandini (ed.), Springer LNCS-x, 1992.
10. Forsyth, D.A., Mundy, J.L., Zisserman, A. and Rothwell, C.A., "Using global consistency to recognise Euclidean objects with an uncalibrated camera," *Proc. CVPR-94*, 1994.
11. Forsyth, D.A. and Rothwell, C.A., "Representations of 3D objects that incorporate surface markings," in *Applications of invariance in computer vision*, J.L. Mundy, A. Zisserman and D.A.Forsyth, (ed.s), Springer LNCS 825, 1994.
12. Fukunaga, K. (1990) Introduction to Statistical Pattern Recognition 2nd Edition, London: Academic Press.
13. Garding, J. and T. Lindeberg, "Direct computation of shape cues by multi-scale retinotopic processing," *Int. J. Computer Vision*, to appear
14. Gigus, Z. and Malik, J., "Computing the aspect graph for line drawings of polyhedral objects," *PAMI-12*, 2, 113-122, 1990.
15. Grimson, W.E.L. and Lozano-Pérez, T. "Localizing Overlapping Parts by Searching the Interpretation Tree," *PAMI*-9, No. 4, p.469-482, July 1987.
16. Huttenlocher, D.P. and Ullman, S. "Object Recognition Using Alignment," Proceedings ICCV1, p.102-111, 1987.
17. Huttenlocher, D.P. and Ullman, S. "Recognizing Solid Objects by Alignment." *IJCV*-5, No. 2, p.255-274, 1990.
18. Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR, p.335-344, 1988.
19. J. Liu, J.L. Mundy, D.A. Forsyth, A.P. Zisserman and C.A. Rothwell, "Efficient Recognition of rotationally symmetric surfaces and straight homogeneous generalized cylinders," *IEEE conference on Computer Vision and Pattern Recognition '93*, 1993.
20. Malik, J. and Perona, P., "Preattentive texture discrimination with early vision mechanisms," *Journal of Optical Society of America A*, **7** (2), May 1990, pp. 923-932.
21. Malik, J. and Rosenholtz, R., "Recovering surface curvature and orientation from texture distortion; a least squares algorithm and analysis," *Proc. ECCV-94*, Springer Lecture Notes in Computer Science 801, 1994.

22. Moses, Y. and Ullman, S. "Limitations of Non Model-Based Recognition Systems," *Proc. ECCV*, LNCS 588, Springer-Verlag, p.820-828, 1992.
23. Mundy, J.L. and Zisserman, A., "Introduction," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
24. Mundy, J.L. and Zisserman, A., "Repeated structures: image correspondence constraints and 3D structure recovery," in *Applications of invariance in computer vision*, J.L. Mundy, A. Zisserman and D.A.Forsyth, (ed.s), Springer LNCS 825, 1994.
25. Nayar, S.K. and Bolle, R., "Reflectance Ratio: a photometric invariant for object recognition," *Proc. ICCV-4*, Berlin, 1993.
26. Pillow, N., Utcke, S. and Zisserman, A., 'Viewpoint-Invariant Representation of Generalized Cylinders Using the Symmetry Set', To appear, *Image and Vision Computing.*
27. Ponce, J. and Kriegman, D.J., "Toward 3D curved object recognition from image contours," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
28. Rothwell, C.A., Zisserman, A.P., Forsyth, D.A. and Mundy, J.L., "Using Projective Invariants for constant time library indexing in model based vision," *Proc. British Machine Vision Conference* , 1991.
29. Rothwell, C.A., Zisserman, A.P, Forsyth, D.A. and Mundy, J.L., "Fast recognition using algebraic invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
30. Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L., "Extracting projective structure from single perspective views of 3D point sets," *International Conference on Computer Vision*, Berlin, 573-582, 1993.
31. Stein, F. and Medioni, G., "Structural indexing: efficient 3D object recognition," *PAMI-14*, 125-145, 1992.
32. Sugihara, K., *Machine interpretation of line drawings*, MIT Press, 1986.
33. Swain, M.J. and Ballard, D.H., "Color Indexing," *Int. J. Computer Vision*, **7**, 1, 11-32, 1991.
34. Taubin, G. and Cooper, D.B., "Object recognition based on moment (or algebraic) invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
35. Thompson, D.W. and Mundy, J.L. "Three-dimensional Model Matching from an Unconstrained Viewpoint," Proceedings ICRA, p.208-220, April 1987.
36. Utcke, S. and Zisserman, A., 'Transfer and Invariants of Surfaces of Revolution', *OUEL Report*, Oxford, 1994.
37. Van Gool, L., Moons, T., Pauwels, E., and Oosterlinck, A., "Semi-differential invariants," in J.L. Mundy and A.P. Zisserman (ed.s) *Geometric Invariance in Computer Vision*, MIT Press, 1992.
38. Voorhees, H. and T.Poggio, "Computing texture boundaries from images," Nature 333, pp. 364-367, 1988
39. Wayner, P.C. "Efficiently Using Invariant Theory for Model-based Matching," Proceedings CVPR, p.473-478, 1991.
40. Weinshall, D., "Model-based invariants for 3-D vision," *Int. J. Computer Vision*, **10**, 27-42, 1993.
41. Weiss, I. "Projective Invariants of Shapes," Proceeding DARPA Image Understanding Workshop, p.1125-1134, April 1988.
42. Zerroug, M. and Nevatia, R., "Volumetric Descriptions from a Single Intensity Image," *Int. J. Computer Vision*, to appear.

43. Zisserman A., Forsyth D., Mundy J., Rothwell C., Liu J. and Pillow N., "3D Object Recognition using Invariance", *Oxford University Engineering Report*, OUEL 2027/94, 1994.

This article was processed using the LaTeX macro package with LLNCS style

Towards a Theory of Characteristic Sets

Deepak Kapur*
Institute for Programming and Logics
Department of Computer Science
State University of New York
Albany, NY 12222
kapur@cs.albany.edu

DRAFT-July, 1992

Abstract

Characteristic sets as defined by Ritt in his book **Differential Algebra** are resurrected. It is shown that Ritt's definition of a characteristic set is quite different from that used by Wu. A characterization theorem for Ritt's characteristic sets for polynomial ideals is proved. An algorithm for computing a characteristic set is given. It is shown how characteristic sets can be used to study the structure of ideals as well as their associated varieties. Many properties of ideals and associated varieties can be computed directly from their characteristic sets.

1 Introduction

Ritt, in his book, **Differential Algebra** (1950), introduced the concept of a *characteristic set* of a set of differential algebraic polynomials from a differential polynomial ring, and demonstrated its utility in studying properties of differential polynomial ideals as well as differential varieties.

In 1978, Wu Wen-Tsun [28, 29, 30] revived interest in characteristic sets by showing how they can be used for mechanizing geometry theorem proving and later, for studying zeros of polynomial equations. Wu made many significant contributions in making Ritt's constructions practical as well as extended Ritt's concepts to make them suitable for solving polynomial equations. It turns out, however, that Wu's definition of a characteristic set is quite different (in a fundamental sense) from Ritt's original definition.

In this paper, we critically examine Ritt's definitions and contrast them with Wu's definitions. We propose a framework for studying Ritt's characteristic sets. We give a necessary and sufficient condition for a set of polynomials to be a characteristic set. We also give an algorithm for computing a characteristic set from a basis of a polynomial ideal. We show how Ritt's characteristic set can be used for geometry theorem proving, equation solving, computing dimension as well as for decomposing a variety.

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361. The results reported in this paper were presented in an invited talk at an international workshop *Mechanization of Mathematics*, Chinese Academy of Science, Beijing, China, July 1992.

2 Ritt's Definition of Characteristic Set

Ritt, in his book, **Differential Algebra** (1950) (p. 5), defined the concept of a *characteristic set* of a set of polynomials from a polynomial ring $k[y_1, ..., y_l]$ as a *chain* of the lowest rank in the set (p. 5). A chain is a finite system of polynomials in *triangular form* (with each polynomial in the set introducing at least one new new variable). See the next subsection below for precise definitions.

After introducing the definition, Ritt remarked on p. 5 that a necessary and sufficient condition for a chain C in a set Σ of polynomials to be a characteristic set of Σ is that there is no non-zero polynomial p in Σ that is reduced with respect to C, or equivalently, every polynomial p in Σ can be reduced using C. Let us call this property as Ritt's first characterization theorem of characteristic sets.

Theorem 1 (Ritt) *A chain C in a set Σ of polynomials is a characteristic set of Σ if and only if every non-zero polynomial in Σ can be reduced by C.*

Ritt also gave a method for computing such a characteristic set from a finite Σ (p. 5). It is easy to see that even for infinite Σ, a characteristic set exists. Given a total ordering on variables, C is a finite maximal subset of minimal reduced polynomials in triangular form Σ. In other words, pick a minimal polynomial c_1 in Σ and include it in C. Delete from Σ all polynomials which can be reduced by c_1[1], and from the remaining subset of Σ, called Σ_1, pick a minimal polynomial c_2. The class of c_2 is necessarily higher than the class of c_1. Delete from Σ_1 all polynomials which can be reduced using c_1 and c_2, and from the remaining subset Σ_2, pick a minimal polynomial c_3, and so on. This process terminates since there are only finitely many indeterminates and a chain cannot have more than n elements.

In *Chapter 4* of his book (pp. 88-90), Ritt gave a characterization for characteristic sets of prime polynomial ideals.

Theorem 2 (Ritt) *A necessary and sufficient condition for a chain $C = p_1 \cdots p_n$ to be a characteristic set of a prime ideal over $k[u_1, \cdots, u_m, x_1, \cdots, x_n]$, where $\{u_1, \cdots, u_m, x_1, \cdots, x_n\} = \{y_1, \cdots, y_l\}, m + n = l, \{u_1, \cdots u_m\}$ are transcendentals, is that for each $0 < i < n$, p_{i+1} is irreducible (cannot be factored) over the extension field obtained by adjoining the zero of p_i to the extension field K_i. The polynomial p_1 should be irreducible over the field of rational functions $k(u_1, \cdots, u_m)$, and the extension field K_1 is obtained by adjoining the zero of p_1 to $k(u_1, \cdots, u_m)$.*

It is also the case that a polynomial p is in a prime ideal I if and only if it pseudo-divides to 0 using a characteristic set of I.

In the same chapter (pp. 95-96), Ritt also gave an algorithm for constructing from a finite Σ, characteristic sets of a finite number of prime ideals whose manifolds (irreducible varieties) make up the manifold (variety) of Σ. This was done by developing the above characterization theorem for characteristic sets associated with prime ideals.

[1] We cannot require that only those polynomials which pseudo-divide to 0 by c_1 be deleted, as there can be polynomials in Σ which neither pseudo-divide to 0 using c_1, nor are reduced with respect to c_1.

2.1 Background: Definitions

Below we first reproduce the following definitions from Ritt's book.

Let $y_1, \cdots, y_n$ be the set of indeterminates with the total ordering $y_1 < \cdots < y_l$. Consider the polynomial ring $R = k[y_1, \cdots, y_l]$. By the *class* of a polynomial p in R, we mean maximum i such that y_i appears in p. If p is in k, then the class of p is said to be 0. For a polynomial p with class $i > 0$, y_i is called its *leading* variable or indeterminate. The *initial* of p is the coefficient of the highest term in y_i when p is viewed as a univariate polynomial in y_i.

A polynomial p is said to be of higher rank than q with respect to y_i if the degree of y_i in p is greater than the degree of y_i in q. A polynomial p is said to be of higher rank than q, denoted as $p > q$, if p is of higher class than q or p and q are of the same class $i > 0$, and p is of higher rank than q with respect to y_i. Obviously, if p is of higher rank than q, then q is of lower rank than p. If neither p is of higher rank than q nor q is of higher rank than p, then p and q are said to be of the same rank. [2]

If p is of class $i > 0$, then q is said to be *reduced with respect to* p if q is of lower rank than p in y_i.

A sequence of polynomials $p_1, \cdots, p_k$ is called a *chain* (or equivalently in *triangular form*) if either

(i) k = 1, and $p_1 \neq 0$, or

(ii) $k > 1$, the class of p_1 is > 0, and for $j > i$, p_j is of higher rank than p_i and reduced with respect to p_i.

Of course, $k \leq n$. It is easy to see that every polynomial in a chain is introducing at least one new variable. That is, $C = p_1, \cdots, p_k$ is a chain if the leading variable of c_i is less than the leading variable of c_j, for all $1 \leq i < j \leq k$.

A chain $C_1 = p_1, \cdots, p_k$ is said to be of *higher rank* than another chain $C_2 = q_1, \cdots, q_l$, written as $C_1 > C_2$, if either

(i) there is a $i \leq k, l$ such that for all $j < i$, p_i and q_i are of the same rank, and p_j is of higher rank than q_j, or

(ii) $l > k$ and p_i and q_i are of the same rank for $i \leq k$.

Two chains such that neither is of higher rank than the other, are said to be of the same rank.

Given a set of chains, Ritt showed how to find a chain which is not of higher rank than any other chain in the set.

And, a characteristic set of Σ, as stated above, is defined to be a chain in Σ of a lowest rank.

A polynomial q is reduced with respect to a chain $C = p_1, \cdots, p_k$ if q is reduced with respect to p_i for each $1 \leq i \leq k$. This implies that for each $1 \leq i \leq k$, the degree of y_j in q is lower than than the degree of y_j in p_i, where the class of p_i is j.

A polynomial q reduces to q' using a chain $C = p_1, \cdots, p_k$ if there exist $e_1, \cdots, e_k$ and $a_1, \cdots, a_k$ such that

$$i_k{}^{e_k} \cdots i_1{}^{e_1} q = a_1 p_1 + \cdots + a_k p_k + q'.$$

It is good to take minimal $e_1, \cdots, e_k$. The polynomial q' is typically computed by pseudo-dividing q by p_k using minimal e_k, then pseudo-dividing the remainder by p_{k-1} and finally

[2] In case p and q are of the same class $i > 0$ and the degrees of y_i in p and q are the same, it is possible to further refine this relation by considering lower variables.

pseudo-dividing the result by p_1 using minimal e_1. It is easy to see that q' is reduced with respect to C. It is also easy to see that $p_j > a_i$ for all $j > i, 1 \leq i \leq k$. Also, q' is in the ideal generated by the basis consisting of C and q. We also call q' as the *normal form* of q with respect to C.

2.2 Characteristic Set a la Wu

Since 1984, Wu has popularized Ritt's concept of a characteristic set and Ritt's algorithm given on pp. 95-96 of his book, for constructing a characteristic set by showing its applicability to equation solving in general and geometry theorem proving, in particular. Wu made important contributions in computing characteristic sets efficiently as well as associating a characteristic set with quasi-varieties. Wu's papers have focussed on constructing characteristic sets from a finite set of polynomials with the crucial distinction being that the ideal generated by such a finite set need not be a prime ideal.

According to Wu's definition of a characteristic set, a characteristic set of Σ need not be in Σ. Thus Wu's definition of a characteristic set appears to be somewhat different from Ritt's definition of a characteristic set. Instead a characteristic set of Σ as defined by Wu is a chain of the lowest rank from a larger set containing Σ where the additional elements are obtained from Σ by pseudo-division. It turns out that all these additional elements are in the ideal generated by Σ. But, a characteristic set of Σ using Wu's algorithm is not necessarily a characteristic set of the ideal generated by Σ either since it does not always include minimal elements in the ideal.

Recently Wu (private communication, 1992) has suggested that a distinction ought to be made between two different notions of a characteristic set: one associated with a polynomial set and another associated with an ideal generated by a polynomial set. This distinction is somewhat unclear to us. Given a set of polynomials Σ, it is not clear what it means to associate a characteristic set with Σ. It is clearly not a chain of lowest rank in Σ as then not every polynomial in Σ pseudo-divides to 0 using this chain. It is a chain of the lowest rank from a set containing Σ (and perhaps, the set is closed under some other operations)? An interesting question comes up: What set is it? Is there an algebraic and/or geometric characterization of such a set?

3 Properties of Characteristic Sets

Henceforth, we associate a characteristic set with a polynomial ideal over $k[y_1, \cdots, y_l]$. Given a basis B, let $I = (B)$, the ideal generated by B. The set $\{y_1, \cdots, y_l\}$ can be classified into two subsets: the set of transcendentals, $\{u_1, \cdots, u_m\}$, and the remaining subset of indeterminates $\{x_1, \cdots, x_n\}$, such that $m + n = l$; obviously, no polynomial in $u_1, \cdots, u_m$ alone is in I, and for every x_i, there is a polynomial in x_i in I. We will also assume the ordering $u_1 < \cdots < u_m < x_1 < \cdots < x_n$. As defined by Ritt, a characteristic set of I is a chain of the lowest rank in I. Let $C = p_1 \cdots p_n$ be a characteristic set of I. It is easy to see that no non-zero polynomial in I is reduced with respect to C.

Theorem 3 *Every polynomial in I pseudo-divides to 0 using a characteristic set C of I.*

Proof. By contradiction. Let q be in I such that q does not pseudo-divide to 0 using C. Without any loss of generality we can assume that q is minimal. Let q' be the remainder

of q when reduced by C; obviously $q' \neq 0$. By the definition of pseudo-division, q' belongs to I since q and C are from I. Let the class of q' be j. Let p_i be the polynomial in C whose class is j; if there is no such polynomial in C, then $C' = p_1, \cdots, p_i, q', p_{i+1}, \cdots, p_n$ is a chain of lower rank than C in I, where the class of p_i is lower than j and the class of p_{i+1} is higher than j, which is a contradiction to the assumption C is a characteristic set of I. If such a p_i exists, then $C' = p_1, \cdots, p_{i-1}, q', p_{i+1}, \cdots, p_n$ is a chain of lower rank than C in I, which is a contradiction.

□

The converse is, however, not true. For prime ideals, it is the case that if a polynomial pseudo-divides to 0 using a characteristic set of a prime ideal, then the polynomial is in the prime ideal. For prime ideals, characteristic sets can be used for ideal membership test.

Below, we give examples of characteristic sets associated with ideals. It is assumed that $x < y$.

Example 1: Consider the ideal I_1 generated by $B_1 = \{(x-1)^2, (x-1)y + 1\}$ over $Q[x, y]$. The reader can verify that B_1 is a characteristic set of B_1 using Wu's definition. However, B_1 is not a characteristic set of I_1 since I_1 is the trivial ideal, and any non-zero element of Q is a characteristic set of I_1.

Example 2: The ideal I_2 generated by $B_2 = \{(x-1)^2, (x-1)y + (x-1)\}$ has B_2 as its characteristic set (B_2 is also a characteristic set in Wu's sense).

Example 3: Consider the ideal I_3 generated by $B_3 = \{(x^2-4), (y^2-9), (x+y+5)z^2+1\}$. B_3 is a characteristic set of B_3 using Wu's definition. But B_3 is not a characteristic set of I_3. Instead, $\{(x^2-4), (x-2)(y-3), (x-2)(6z^2+1)\}$ is a characteristic set of I_3.

A characteristic set as defined by Ritt should also serve as a characteristic set in Wu's sense. The requirement on a characteristic set as defined by Wu is that (i) it is a chain, (ii) every polynomial in the input set pseudo-divides to 0 using it, and (iii) the zeros (the variety) of the input set is the union of the the quasi-variety of the characteristic set (i.e., the zeros of the polynomials in the characteristic set on which the product of the initials does not vanish) and the variety of the input set augmented with the product of the initials.

We now develop a characterization theorem for characteristic sets of polynomial ideals analogous to the theorem that Ritt proved about characteristic sets for prime ideals. For that we need the following operations with respect to a chain. We assume that transcendental elements $u_1, \cdots, u_m$ are known.

3.1 Invertibility with respect to an ideal

Let I be an ideal generated by a basis $(p_1, \cdots, p_i)$. A polynomial q is said to be *invertible with respect to an ideal* I if there exist polynomials $a_1, \cdots, a_i$ and q^{-1} such that

$$q^{-1}q = a_1p_1 + \ldots + a_ip_i + r,$$

where r is a non-zero polynomial in the transcendentals. The polynomial q^{-1} is called an *inverse* of q with respect to I in the quotient ring $Q[u_1, \cdots, u_m, x_1, \cdots, x_n]/I$.

It is easy to see that q being invertible with respect to I means that q and I do not have any common zero in any algebraic closed extension of the field $k(u_1, \cdots, u_m)$ (follows from Hilbert's Nullstellensatz). Hence, a polynomial q is not invertible with respect to I if there do not exist $r, q^{-1}, a_1, \cdots, a_i$ satisfying the above properties. This also implies that q and I have a common zero (again follows from Hilbert's Nullstellensatz).

If a basis of I is a chain C, then the class of p_i above is lower than or equal to the class of q, and the inverse q^{-1} of q is of lower rank than p_i. Then, we will also say that q is invertible (or not invertible) with respect to C.

If a polynomial q is not invertible with respect to C, then it is possible to find a polynomial d such that

$$dq = a_1 p_1 + \ldots + a_i p_i,$$

where the class of p_i is lower than or equal to the class of q, and d is of lower rank than p_i. The polynomial d is a zero divisor in the quotient ring $k[x_1, \cdots, x_n]/(C)$, and will be called an *annihilator* of q with respect to C.

A polynomial r divides another polynomial s with respect to a I if and only if there exist q and a_i's such that

$$qr = a_0 s + a_1 p_1 + \ldots + a_i p_i,$$

where a_0 is a polynomial in the transcendentals, if any.

If a chain C is used as a basis for I and r divides another polynomial s with respect to C then the class of p_i is not higher than the class of s, and a_0 is a polynomial in the transcendentals, if any.

The invertibility check, the computation of q^{-1}, a_i's and r, as well as of d can be done by Collins' extended gcd (resultant) algorithm [7, 25]. This computation can also be performed using pseudo-division, as well as using D5 method [9].

A chain C is said to be *invertible* if the initial of every polynomial in C is invertible with respect to C. Such a chain has been called a *regular* chain by Kalkebrener [16].

A chain C is said to be in *canonical form* if every polynomial p in C, (i) p is reduced with respect to polynomials lower than p in C, and (ii) (a) the initial of p is either a polynomial in the transcendentals or (b) the zero set of the initial is a subset of the zero set of polynomials lower than p in C. The condition (ii) (b) is to ensure that the initial that is not invertible, does not have a zero that is not a zero of the polynomials lower than in it in the chain.

Example 4: A chain $\{(x-1)^3, (x-1)(x-2)y+(x-1)\}$ is not in canonical form, since the initial of the second polynomial has a zero $x = 2$ which is not a zero of $(x-1)^3$. An equivalent chain is, however, $\{(x-1)^3, (x-1)y - x(x-1)\}$ is in canonical form.

3.2 A Characterization Theorem for Characteristic Sets of Polynomial Ideals

Even if a characteristic set C of an ideal I is an invertible chain, it is not the case that if a polynomial p pseudo-divides to 0 using C, then p is in I. Consider, for example, the ideal I generated by $C = ux$, where u is an transcendental element. C is an invertible chain. The polynomial x pseudo-divides to 0 but x is not in the ideal I. However, this is true for zero-dimensional ideals.

3.3 Zero-dimensional Ideals

In this subsection, we prove a characterization theorem for zero-dimensional ideals. These results are then generalized to positive-dimensional ideals. Throughout this subsection, $m = 0$ and there are no transcendental elements.

Theorem 4 *Given an ideal I, if its characteristic set C is an invertible chain, then a polynomial q is in I if and only if q pseudo-divide to 0 with respect to C.*

Proof. Without any loss of generality, we can assume that the initial $initial(p_i)$ of p_i in C is an element of k (if $initial(p_i)$ is not an element of k, then p_i can be multiplied with the inverse of $initial(p_i)$ to obtain p_i' whose initial is in k).

The only-if part follows from the second characterization theorem about characteristic sets, i.e. every polynomial in I pseudo-divides to 0 with respect to a characteristic set C of I.

If q pseudo-divides to 0 using C, we can write

$$i_1{}^{e_1} \cdots i_k{}^{e_k} q = a_1 p_1 + \cdots + a_k p_k,$$

where i_j is the initial of p_j. Since $i_j \in k$, we have $q \in (C)$ and hence I.

□

A characteristic set of an ideal may have polynomials whose initials are not invertible with respect to the characteristic set.

Example 5: Let $I = ((x-1)^2, (x-1)y+(x-1))$. Under the ordering $y > x$, a Gröbner basis using lexicographic ordering on terms [2, 3, 14, 18], $GB = \{(x-1)^2, (x-1)y+(x-1)\}$, from which it follows that $C = GB$. The initial of the polynomial introducing y is $x-1$, and $x-1$ is not invertible with respect to C. However, using the ordering $x > y$, a characteristic set is $(y+1)x - (y+1)$.

There are ideals for which no matter what ordering on indeterminates is used, none of its characteristic sets has polynomials with invertible initials.

Example 6: Consider $I = ((x-1)(x-2), (x-1)(y-1), (y-1)(y+1))$. Using the ordering $x > y$, $C_1 = \{y^2 - 1, (y-1)x - (y-1)\}$ is a characteristic set of I. Using the ordering $y > x$, $C_2 = \{x^2 - 3x + 2, (x-1)y - (x-1)\}$ is a characteristic set of I.

Theorem 5 *If a polynomial p_i is in a characteristic set C and $initial(p_i)$ is not invertible with respect to C, then $initial(p_i)$ must divide p_i modulo the chain $\{p_1, \cdots, p_{i-1}\}, i > 1$.*

Proof Sketch. Let $p_i = initial(p_i) x_i{}^{d_i} + p'$, and $I_p = initial(p_i)$. Since I_p is not invertible with respect to C, there exist J_p and $a_1, \cdots, a_k, k < i$ such that

$$J_p I_p = a_1 p_1 + \cdots + a_k p_k.$$

So, $J_p p_i = J_p(I_p x_i{}^{d_i} + p') = J_p I_p x_i{}^{d_i} + J_p p'$. Clearly $J_p p'$ is in the ideal (C). Further J_p does not involve x_i thus implying that $J_p p'$ is of lower rank than p_i. The coefficient of every lower degree (> 0) terms in x_i in $J_p p'$ must pseudo-divide to 0 using C as otherwise a chain in (C) in which $J_p p'$ replaces p exists and is smaller than C contradicting the assumption that C is a characteristic set of (C). The coefficient of $x_i{}^0$ in $J_p p'$ must also pseudo-divide to 0 using C, as otherwise there exists a chain smaller than C in (C) in which a polynomial lower than p_i in C is replaced by a smaller polynomial.

Consider any such coefficient, say c, of a term in x_i in $J_p p'$. Since it pseudo-divides to 0 using elements lower than p in C, we have:

$$i_1{}^{e_1} \cdots i_k{}^{e_k} c = a_1 p_1 + \cdots + a_k p_k,$$

Without any loss of generality, we can assume that i_j is either an element of k or a polynomial that is not invertible with respect to C.

If every polynomial lower than p_i in C has an invertible initial, then each i_j above is in k, and hence c is in the ideal $(p_1, \cdots, p_k)$. Since $c = J_p c'$, where c' is the term in p_i without x_i, I_p divides c'.

In the general case, we can assume that for every polynomial smaller than p in C, its initial divides the polynomial. There are two possibilities for I_p to be not invertible with respect to C, I_p vanishes on a zero of some initial of a polynomial lower than p or I_p vanishes on a zero of polynomials lower than p which is not a zero of any of the initials, or both.

Any zero of an initial used on the left side is a zero of the right hand side. Since an initial that is is not invertible with respect to C, it is a zero of C, implying a zero of p, which means that the irreducible factors (primitive part) of the initial divide p any zero of c is also a zero of the right hand side.

□

3.4 Positive-Dimensional Ideals

Now we are able to state a characterization theorem for characteristic sets of polynomial ideals.

Lemma 1 *Given an ideal I and a characteristic set C of I such that C is an invertible chain, if p pseudo-divides to 0 then, p is in the ideal generated by $C' = p'_1, \cdots, p'_n$, where $p'_1 = p_1/content(p_1)$, the gcd of the coefficients of x_1 terms in p_1, and for $i > 1$, $p'_i = p_i/content(p_i)$, where the content of p_i is defined to be the gcd of the coefficients of x_i terms with respect to the chain $p'_1, \cdots, p'_{i-1}$.*

Theorem 6 *A chain C is a characteristic set of a polynomial ideal if and only if for every polynomial p in C, either its initial I_p is invertible with respect to C, or I_p divides p with respect to C.*

Proof. The if part follows from Theorem 5 above.

We prove the only if part by proving that C is a characteristic set of $I = (C)$. Without any loss of generality we can assume that the initial of every polynomial in C is either a polynomial in transcendentals or not invertible with respect to C. In either case, the initial of a polynomial p_i in C divides p_i with respect to polynomials lower than p_i in C. From C, we can generate an invertible characteristic set $C' = p'_1, \cdots, p'_k$ as follows: $p'_i = p_i/i_{p_i}$, $i_{p_i} = initial(p_i), 1 \leq i \leq k$.

The proof is by contradiction. Assume that C is not a characteristic set of I, then there must exist a non-zero polynomial, say q, in I such that q does not pseudo-divide to 0 using C as that would suggest that a chain smaller than C exists in I. Without any loss of generality, we can assume that q is reduced with respect to C (this is so because pseudo-division of a polynomial in an ideal by another polynomial in the ideal produces a polynomial in the ideal also).

Since q is in (C), there exist a_i's such that

$$q = a_1 p_1 + \cdots + a_k p_k.$$

so the above relation becomes

$$q = a_1 i_{p_1} p'_1 + \cdots + a_k i_{p_k} p'_k.$$

It is possible to pseudo-divide $a_j i_{p_j}$ by C' to obtain a remainder b_j such that

$$a_j i_{p_j} = a_{j,1} p'_1 + \cdots + a_{j,k} p'_k + b_j,$$

where b_j is reduced with respect to C' and $a_{j,i} < p'_l$ for all $1 \leq i < l \leq k$. Substituting for each $a_j i_{p_j}$, we get

$$q = d_1 p'_1 + \cdots + d_k p'_k,$$

where $d_i < p'_l$ for all $1 \leq i < l \leq k$. Let the class of q be $j \leq 0$; consider the coefficient of the highest term in x_j in q viewed as a univariate polynomial in x_j. The coefficients of all terms involving a variable $> x_j$ on the right hand side must be 0. From this, it follows that q is reduced with respect to C', implying that q is not reduced with respect to C, which is a contradiction.

So, C is a characteristic set of (C).

□

In fact, C serves as a characteristic set of many ideals containing I as a subideal.

Every characteristic set has a non-empty zero set.

For an invertible characteristic set C, its zero set is clearly the zero set of $(C)^e$.

4 An Algorithm for Computing a Characteristic Set of an Ideal

Since a characteristic set of an ideal consists of certain minimal elements in the ideal, it can be easily extracted from a Gröbner basis of the ideal. This is precisely what Kandri-Rody showed in his Ph.D. dissertation [17]. Kandri-Rody gave an algorithm for computing a characteristic set of a polynomial ideal using Buchberger's Gröbner basis algorithm. From a basis B of an ideal I and a total ordering on indeterminates, compute its reduced Gröbner basis using lexicographic ordering. From the Gröbner basis, extract a minimal chain of polynomials and call it C. Kandri-Rody showed that C is a characteristic set by showing that every polynomial in I pseudo-divides to 0 using C.

This algorithm is however inefficient in practice, since computing lexicographic Gröbner basis is inefficient in practice (except for the zero-dimensional case where the basis conversion algorithm of Gianni et al can be used to obtain a lexicographic Gröbner basis from a total degree Gröbner basis).

Below we give a direct algorithm for computing a characteristic set of a polynomial ideal specified by its basis B. The algorithm does not need to compute a Gröbner basis of a polynomial ideal. It is a modification of Ritt's algorithm given on p. 95 of his book.

Let $\Sigma = B$ and $C = \emptyset$.

Repeat the following steps until every polynomial in Σ pseudo-divides to 0 using C.

1. Find a minimal chain C in Σ. Let $S_0 = \Sigma; C = \emptyset, i = 0$. This is done by repeating the following two steps.

(a) Remove the smallest polynomial in S_i, say p. If p is a polynomial in the transcendentals and the lowest indeterminate x_1, then factor and check which factor is in the ideal I.

If not, check whether the initial *init* of p is invertible or not with respect to C. If *init* is invertible, then extend C to include p (p could be multiplied by the inverse of *init* and pseudo-divided by C to get a remainder which is included in C instead of p). If not, then check whether *init* is a factor of p with respect to C. If yes, then extend C to include p.

If *init* is not invertible and is also not a factor of p, then find a zero divisor *zd* of *init* with respect to C, and multiply p with it, pseudo-divide by C.

It can be easily shown that the initial of the result is invertible.

Augment C to include the result. Let c_i be the new polynomial added to C. Replace p in Σ by c_i also.

It is possible that in this step, we may get a polynomial in a lower variable which may result in removing an element from C that introduces the lower variable.

(b) Remove from S_i all polynomials which are not reduced with respect to the new polynomial c_i. Let S_{i+1} stand for the remaing subset of polynomials. Also increment i.

After this step, the initial of every polynomial in the chain is either invertible or a factor of the polynomial.

2. Once a minimal chain C is identified. Pseudo-divide every polynomial in Σ with respect to C. (This can be done using the division algorithm or Collins' subresultant algorithm [1, 6, 7, 25].) If all remainders are 0, then C is a characteristic set of (B). Otherwise, augment Σ to include the non-zero remainders.

3. Repeat steps 1 and 2.

Optimizations suggested by Wu and his colleagues (such as using subresultant computation to do multi-step pseudo-division as well as skipping some remainders) can be easily incorporated into the above algorithm.

Theorem 7 *The above algorithm gives a characteristic set C of $I = (B)$.*

Proof. If the algorithm stops after j iterations, then C is clearly a minimal chain in Σ.

In each iteration, Σ is modified by the following two operations. One, replace a polynomial p in Σ by another lower polynomial c_i obtained by multiplying p by a zero divisor with respect to a chain C at hand which annuls the initial of p. Second, augment Σ with non-zero remainders with respect to C. Each of these operations results in a polynomial in (Σ), and hence (B).

We need to prove that every polynomial in I pseudo-divides to 0 using C. This can be shown by proving that C constitutes a minimal chain and using Theorem 1. it is easy to see that except for the least polynomial in C, no polynomial in variables of class higher than $m+1$ in I in is of lower rank than the polynomial of the same class in C. For the least polynomial in C, we make an explicit check to ensure that it is a polynomial of smallest degree in the transcendentals and x_1 in I. Hence C is a characteristic set.

□

There exist an ideal I such that no matter what ordering is used, the above algorithm gives a characteristic set C of I in which initials of polynomials are not invertible. However, there exists a characteristic set of I consisting of polynomials with invertible initials.

Consider $I = ((x-1)(y-1), (x-2)(y-2))$. Using $y > x$, the above algorithm gives the characteristic set $C_1 = \{(x-2)y - 2(x-2), (x-2)(x-1)\}$. Using $x > y$, the characteristic

set computed by the algorithm is $C_2 = \{(y-1)x-(y-1),(y-1)(y-2)\}$. The polynomial $x+y-3$ belongs to I thus giving other characteristic sets in which $x+y-3$ is the smallest polynomial no matter what ordering is used.

It is an interesting question to determine additional computational steps to obtain an invertible characteristic set (if one exists) for an ideal with respect to an ordering.

One obvious way is to compute an invertible characteristic set, in case it exists, from a characteristic set using linear algebra techniques. The general structure of the polynomials in an invertible characteristic set is known (i.e. the degree of each variable in the polynomial). For the above example, it is known that the degree of the polynomial in y is 1, and the degree of the polynomial in x is < 2. So, the polynomial must of the form $y + a_1x + a_2$. This polynomial must also pseudo-divide to 0 using C_1, so $2(x-2)+(x-2)(a_1x+a_2)$ must pseudo-divide to 0 using $(x-2)(x-1)$, which implies that $a_1 = 1$ and $a_2 = -3$.

In general, we start with the smallest polynomial in the chain whose initial is not invertible. Let its class be i. Using the degree bounds on the variables, we assume a general polynomial satisfying the degree bounds with unknown coefficients of the terms. We set up linear equations from the fact that the polynomial must pseudo-divide to 0 using the known characteristic set, and solve for the coefficients. If there is a solution for the coefficients, the desired polynomial is determined; otherwise, there is no polynomial of class i with an invertible initial. In either case, we go to the next polynomial in the chain with an initial that is not invertible, and so on.

5 Decomposing a Variety into Simpler Varieties

The variety of a characteristic set C of an ideal I includes, in general, the variety of the ideal I. This is so because in general, (C) is a subideal of I.

If C is invertible, then $I^e = (C)$ when viewed over $k(u_1, \cdots, u_m)$. Then $V(I) = V(C)$.

If C is not invertible, then $V(I) \subseteq V(C)$.

A characteristic set C of an ideal I can be used to decompose $V(I)$ into subvarieties such that each subvariety has a invertible characteristic set associated with it.

Theorem 8 (Decomposition):
Given Σ, a system of invertible characteristic sets, $\{C_1, \cdots, C_k\}$ can be effectively constructed (without using factorization over algebraic extension fields) such that

$$Zeros(\Sigma) = Zeros(C_1) \cup \cdots \cup Zeros(C_k).$$

The decomposition theorem has many applications. For examples, it can serve as a basis for geometry theorem proving, determining zeros of polynomial equations, primary decomposition, determining zeros of parametric polynomial equations, computing dimension of a variety (ideal), etc.

Note that this decomposition theorem is of different character than a related theorem given by Wu in which decomposition of the zero set of a set of polynomials (variety) is given in terms of a finite union of quasi-varieties. In Wu's decomposition theorem, there is no guarantee that the quasi-variety is empty or not; extra computation is needed to ensure non-emptiness of quasi-varieties. In the above decomposition, every subvariety is non-empty since it is associated with an invertible characteristic set.

The following algorithm gives the above decomposition. The correctness proof of the algorithm serves as a proof of the decomposition theorem.

1. Compute a characteristic set C^0 of $I = (\Sigma)$. Let $F = \{C^0\}$

2. Check whether every element in F is an invertible characteristic set. If yes, terminate.

If not, pick a element of F that is not invertible, say C^i in F which includes a polynomial whose initial is not invertible.

3. Find the smallest polynomial, say p^i, in C^i whose initial is not invertible. Given that $initial(p^i)$ divides p^i with respect to C^i, a decomposition of C^i can be obtained using D5 algorithm (subresultant computation) which determines that $initial(p^i)$ is not invertible with respect to C^i. Let the decomposition give a family of chains $C^i_1, \cdots, C^i_j$, each one of them having exactly one of $initial(p^i)$ and $p^i/initial(p^i)$. Include in F, the characteristic sets of $\Sigma \cup C^i_g$, for each $1 \leq g \leq j$.

4. Repeat steps 2 and 3.

The algorithm stops when every element of F is an invertible characteristic set. The family F serves as the decomposition. The variety $V(I)$ is decomposed to the union of subvarieties $V(C^i), C^i \in F$.

The correctness of the algorithm is based on the fact that given a chain $C = p_1 \cdots p_{i+1}$ in an ideal I such that

(i) $initial(p_{j+1}), 0 \leq j \leq i$ is invertible with respect to $p_1, \cdots, p_j$,

(ii) $initial(p_{i+1})$ is not invertible with respect to $p_1, \cdots, p_i$ and

(iii) $initial(p_{i+1})$ divides p_{i+1} with respect to $p_1, \cdots, p_i$,

then there exist ideals $I_j = I \cup (p^j_1, \cdots, p^j_{i+1})$, $1 \leq j \leq g$ such that

(i) p^j_{i+1} is either $initial(p_{i+1})$ or $p_{i+1}/initial(p_{i+1})$,

(ii) for $1 \leq k \leq i$, p^j_k is a factor of p_k with respect to $p_1 \cdots p_{k-1}$, and there exist $j_1, \cdots, j_g$ such that $p_k = p^{j_1}_k * \cdots * p^{j_g}_k$ with respect to $p_1, \cdots, p_{k-1}$, and

(iii) p^j_k does not vanish on the zeros of $p^j_1, \cdots p^j_{k-1}$.

The termination of the algorithm follows from the property that a characteristic set of each of I_j is of lower rank than the characteristic set of I.

Examples: Consider the set $\Sigma = \{(x-1)(x-2), (x-1)(y-1), (y-1)(y+1)\}$. Assume $x > y$. A characteristic set $C_1 = \{y^2 - 1, (y-1)x - (y-1)\}$ can be constructed using the algorithm given in the previous section. Since the initial of the second polynomial, $y-1$, is not invertible. We decompose C_1 to consider new sets $\Sigma \cup \{y+1, x-1\}$ and the set $\Sigma \cup C_1 \cup \{y-1\}$, from which we can compute characteristic sets $C^1_1 = \{y+1, x-1\}$, and $C^2_1 = \{y-1, x^2 - 3x + 2\}$, giving us a decomposition of the variety of the above set.

Depending upon a variable ordering, we may get different decompositions with different number of components.

Steps 1 and 2 and 3 can be merged into the algorithm for computing a characteristic set discussed in the previous section. It is not essential to compute characteristic sets in step 3 (or even in step 1). At any point in the computation, if a polynomial is generated such that

(i) it is a candidate for inclusion in a characteristic set,

(ii) its initial is not invertible with respect to lower polynomials, and

(iii) its initial divides the polynomial,

decomposition can be performed in the process of determining noninvertibility of the initial.

6 Computing dimension of a polynomial ideal

The dimension of a polynomial ideal I (equivalently, variety) can be obtained from a decomposition of the variety associated with I as a family of invertible characteristic sets, as discussed above in the previous section.

Theorem 9 *The dimension of the unmixed ideal I (and the associated variety) generated by an invertible characteristic set $C = p_1, \cdots, p_k$ is the difference of the number of variables appearing in C and the size of the characteristic set.*

Proof. Let $m+1$ be the number of variables appearing in p_1. If $m > 0$, then no polynomial in any of the m variables appearing in p_1 is in I (since such a polynomial cannot be pseudo-divided to 0 by C). Let $m + i_2$ be the number of variables appearing in p_2; clearly no polynomial in

And, there are polynomials in $m + 1$ or more variables in I. So by the results in [22], the dimension of I is m.

□

If a characteristic set C is not invertible, nothing much can be said about the dimension of an ideal that has C as a characteristic set. However the following is true:

Theorem 10 *Given a characteristic set C, the dimension of the ideal (C) is the number of variables in C minus the size of C plus maximum number of polynomials in C whose initials simultaneously vanish.*

It is easy to see that the above theorem about the dimension of an invertible characteristic set C is an immediately corollary of this theorem since none of the initials of polynomials in C can vanish.

Proof. Let $\{initial_{i_1}, \cdots, initial_{i_j}\}$ be a maximal subset of initials of polynomials in C that simultaneously vanish. It is easy to see that the variety associated with (C) includes the subvariety $(C, initial_{i_1}, \cdots, initial_{i_j})$. Now the dimension of this variety is the number of variables plus j minus the size of C (a characteristic set corresponding to $(C, initial_{i_1}, \cdots, initial_{i_j})$ can be proved to be invertible and its size is the size of C minus j). It is easy to see that the variety of I does not include a subvariety of higher dimension.

□

The following property is helpful to determine the dimension of an ideal.

Lemma 2 *Given a chain $C = p_1, \ldots, p_i, p_{i+1}$ such that the initial of p_j, $j < i + 1$, is invertible with respect to C and $initial(p_{i+1})$ of p_{i+1} divides p_{i+1} with respect to C, the variety $V((p_1, \ldots, p_i, p_{i+1}))$ is either of the same dimension as the dimension of $V((p_1, \ldots, p_i, p_{i+1}/initial(p_{i+1}))$ or one dimension higher.*

Proof.
$V((p_1, \ldots, p_i, p_{i+1})) = V((p_1, \ldots, p_i, p_{i+1}/initial(p_{i+1}))) \cup V((p_1, \ldots, p_i, initial(p_{i+1})))$.

The dimension of $V((p_1, \ldots, p_i, p_{i+1}/initial(p_{i+1})))$ is $n - i + 1$, where n is the number of variables in C. The dimension of $V((p_1, \ldots, p_i, initial(p_{i+1})))$ is either 0 or the dimension of $V((p_1, \ldots, p_i))$ depending upon whether $initial(p_{i+1})$ is invertible with respect to $p_1, \ldots, p_i$. And, the dimension of $V((p_1, \ldots, p_i))$ is one more than the dimension of $V((p_1, \ldots, p_i, p_{i+1}/initial(p_{i+1})))$.

□

Putting all these together, we get:

Theorem 11 *Given an ideal I, let $C_1, \cdots, C_j$ be the invertible characteristic sets associated with unmixed ideals such that $V(I) = V(C_1) \cup \cdots V(C_j)$. Then, the dimension of I is maximum over the dimension of each (C_i).*

Proof. Follows from the definition of the dimension of an ideal and the above theorems.

Since many ideals can have a chain C satisfying the properties stated in section as a characteristic set, it is not possible to compute the dimension of a polynomial ideal I from its characteristic set C especially if C is not invertible. Let n be the total number of variables; let k be the size of the characteristic set C in which k_i are the polynomials whose initials are invertible. If C is an invertible chain, i.e., $k_i = k$, then the decomposition will have only one component, so the dimension of I is $n - k$. In the case when the initial of some polynomial in C is not invertible, i.e. $k_i < k$, we get upper and lower bounds on the dimension of I from the structure of C. The dimension cannot be lower than $n - k$, and cannot be more than $n - k_i$. This is so because it is possible to construct examples of ideals and characteristic sets for which the dimensions are any number between the lower and upper bounds obtained from their characteristic sets, including the lower and upper bounds.

Examples: Consider $\Sigma_1 = \{(x-1)(x-2), (x-1)(y-1), y^2-1\}, I_1 = (\Sigma_1)$. Assume $y > x$. Its characteristic set is: $C_1 = \{x^2 - 3x + 2, (x-1)(y-1)\}$. The initial $x - 1$ of the second polynomial is not invertible. So the upper and lower bounds on the dimension of I_1 are 1 and 0, respectively. We decompose the variety of I_1 and get two sets $\Sigma_1 \cup \{x-2, y-1\}$ and $\Sigma_1 \cup \{x-1\}$, from which we get two invertible characteristic sets $C_1^1 = \{x-2, y-1\}$ and $C_1^2 = \{x-1, y^2-1\}$. The variety is decomposed into these two varieties defined by C_1^1 and C_1^2. The dimension of the subvarieties of I_1 is 0 in each case, thus implying that the dimension of $I_1 = 0$.

If the characteristic set was computed using the ordering $x > y$, we get: $C_2 = \{y^2 - 1, (y-1)(x-1)\}$. The initial $y-1$ is not invertible, we decompose and get $C_2^1 = \{y+1, x-1\}$, and a new set $\{(x-1)(x-2), (x-1)(y-1), y^2-1, y-1\}$ from which another characteristic set $C_2^2 = \{y-1, (x-1)(x-2)\}$ is generated. The dimension of each of the subvarieties in this case is also 0.

Consider $I_2 = (x^2-1, (x-1)y, (x+1)z, yz)$. Assume $z > y > x$. A characteristic set $C_2 = \{x^2-1, (x-1)y, (x+1)z\}$. The initials of second and third polynomials are not invertible. The upper and lower bounds on the dimension of I_2 are 2 and 0, respectively. We decompose the variety of I_2 to get $C_2^1 = \{x+1, y\}$, $C_2^2 = \{x-1, z\}$, which gives the dimension of I_2 to be 1.

Consider $I_3 = (x^2-1, (x-1)y, (x-1)z)$. Assume $z > y > x$. A characteristic set $C_3 = \{x^2-1, (x-1)y, (x-1)z\}$. The initials of second and third polynomials are not invertible. The upper and lower bounds on the dimension of I_3 are 2 and 0, respectively. The variety of I_3 is decomposed to get $C_3^1 = \{x+1, y, z\}$, and $C_3^2 = \{x-1\}$, which gives the dimension of I_3 to be 2.

Consider $I_4 = (x^2-1, (x-1)y, (x-1)z, z^2-(x-1))$. Assume $z > y > x$. A characteristic set $C_4 = \{x^2-1, (x-1)y, (x+1)z\}$. The initials of second and third polynomials are not invertible. The upper and lower bounds on the dimension of I_4 are 2 and 0, respectively.

We decompose the variety of I_4 to get $C_4^1 = \{x+1, y, z^2+2\}$, and $C_4^2 = \{x-1, z\}$, which gives the dimension to be 1.

Acknowledgements: I am grateful to Professor Kandri-Rody for raising pertinent questions about Ritt's concept of characteristic sets which made me rethink about characteristic sets. Many ideas reported in this paper arose during conversations with Kandri-Rody while his visit to SUNY, Albany. I thank Lakshman Y.N. for many helpful discussions.

References

[1] J.S. Brown and J.F. Traub (1971), "On Euclid's algorithm and the theory of subresultants", *JACM,* **18**(4), 505-514.

[2] B. Buchberger (1965), *Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal*, Ph.D. Thesis (in German), Universitat Innsbruck.

[3] B. Buchberger (1985), "Gröbner bases: An Algorithmic method in Polynomial Ideal theory", in *Multidimensional Systems Theory,* N.K. Bose, ed., D. Reidel Publishing Co., 184-232.

[4] S.-C. Chou (1988), *Mechanical Geometry Theorem Proving,* D. Reidel Publishing Company, Netherlands.

[5] S.-C. Chou and X.-S. Gao (1990), "Ritt-Wu's decomposition algorithm and geometry theorem proving", Proc. of *CADE-10*, Kaiserslautern, Germany, Springer Verlag LNCS.

[6] G.E. Collins (1967), "Subresultants and polynomial remainder sequences", *JACM,* **14**(1), 128-142.

[7] G.E. Collins (1971), "The calculation of multivariate polynomial resultants", *JACM,* **18**(4), 515-532.

[8] C.I. Connolly, D. Kapur, J.L. Mundy, and R. Weiss (1989), "GeoMeter: A system for modeling and algebraic manipulation", Proc. *DARPA workshop on Image Understanding,* CA, 797-804.

[9] D. Duval (1991), "Computation with algebraic numbers: the D5 method", *J. Symbolic Computation,* to appear.

[10] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora (1989), *Efficient computation of zero-dimensional Gröbner bases by change of ordering*, Technical Report 89–52, LITP, Universite Paris.

[11] G. Gallo and B. Mishra, "Recent Progress in Characteristic Set Computation: Complexity and Open Problems," Proc. of the *1992 Intl. Workshop on Mathematics Mechanization,* Wu and Cheng (eds.), Beijing, China, July 16-18, 1992, Intl. Academic Publishers, 28-37.

[12] P. Gianni (1987), "Properties of Gröbner bases under specializations", Proc. *EUROCAL '87*, Leipzig, Springer Verlag LNCS 378, 293–297.

[13] W. Gröbner (1949), *Moderne algebraische Geometrie,* Wien.

[14] C.M. Hoffman (1989), *Geometric and Solid Modeling: An Introduction,* Morgan Kaufmann.

[15] M. Kalkbrener (1987), "Solving systems of algebraic equations by using Gröbner bases", Proc. *EUROCAL '87,* Leipzig, Springer Verlag LNCS 378, 282–292.

[16] M. Kalkbrener (1992), *A generalized Euclidean algorithm for solving systems of algebraic equations,* Technical Report, Mathematical Sciences Institute, Cornell University, Ithaca, NY.

[17] A. Kandri-Rody (1984), *Effective methods in the theory of polynomial ideals*, Ph.D. Thesis, Dept. of Mathematics, Rensselaer Polytechnic Institute, Troy, NY.

[18] D. Kapur and Lakshman Y.N. (1992), "Elimination Methods: An Introduction," in *Symbolic and Numerical Computation for Artificial Intelligence,* Donald, Kapur and Mundy (eds.), Academic Press, 1992.

[19] D. Kapur and J.L. Mundy (1988), "Wu's method and its application to perspective viewing", *Artificial Intelligence,* **37**, 15-36.

[20] D. Kapur and H. Wan (1990), "Refutational proofs of geometry theorems via characteristic set computation", Proc. of *the ACM-SIGSAM 1990 International Symposium on Symbolic and Algebraic Computation - ISSAC '90*, Japan.

[21] D.E. Knuth (1980), *Seminumerical algorithms: The art of computer programming*, 2, Second Edition, Addison Wesley, 407-408.

[22] H. Kredel and V. Weispfenning (1989), "Computing dimension and independent sets for polynomial ideals," *Computational Aspects of Commutative Algebra,* (ed. L. Robbiano), Academic Press, London, 97-113.

[23] D. Lazard (1992), "Solving zero–dimensional algebraic systems," *J. Symbolic Computation,* 13, 117-131.

[24] D. Lazard (1991), "A new method for solving algebraic systems of positive dimension," *Discrete Applied Math.,* 33, 147-160.

[25] R. Loos (1983), "Generalized polynomial remainder sequences", in *Symbolic and Algebraic Computation (Computing Supplement 4),* B. Buchberger, G.E. Collins, and R. Loos, eds., Springer Verlag, 2nd ed., 115-137.

[26] J.F. Ritt (1950), *Differential Algebra,* AMS Colloquium Publications.

[27] B.L. van der Waerden (1970), *Algebra*, **1** and **2**, Frederick Ungar Pub. Co.

[28] W. Wu (1984), "On the decision Problem and the mechanization of theorem proving in elementary geometry", *Scientia Sinica,* **21,** (1978), 150-172. Also in Bledsoe and Loveland, eds., *Theorem Proving: After 25 years, Contemporary Mathematics,* **29**, 213-234.

[29] W. Wu (1986), "Basic principles of mechanical theorem proving in geometries", *J. of System Sciences and Mathematical Sciences,* 4(3), (1984), 207-235. Also appeared in *J. of Automated Reasoning,* **2,** (1986), 221-252.

[30] W. Wu (1986), "On zeros of algebraic equations - an application of Ritt's principle", *Kexue Tongbao,* **31**(1), 1-5.

[31] W. Wu (1992), "On Char-set Method and Linear Equations Method of Polynomial Equation Solving," Proc. of the *1992 Intl. Workshop on Mathematics Mechanization,* Wu and Cheng (eds.), Beijing, China, July 16-18, 1992, Intl. Academic Publishers, 101-109.

1 An Approach for Solving Systems of Parametric Polynomial Equations

Deepak Kapur[1]

1.1 Abstract

An approach for solving nonlinear polynomial equations involving parameters is proposed. A distinction is made between parameters and variables. The objective is to generate from a system of parametric equations, solved forms from which solutions for specific values of parameters can be obtained without much additional computations. It should be possible to analyze the parametrized solved forms so that it can be determined for different parameter values whether there are infinitely many solutions, finitely many solutions, or no solutions at all. The approach is illustrated for two different symbolic methods for solving parametric equations – Gröbner basis computations and characteristic set computations. These methods are illustrated on a number of examples.

1.2 Introduction

Many complex phenomena can be modeled using nonlinear polynomial equations. Examples include imaging transformations in computer vision, computing geometric invariants, geometric and solid modeling, constraint-based modeling, reasoning about geometry problems, properties of chemical equilibrium, kinematics, robotics; an interested reader may consult [13, 14, 7, 5] for many examples. Variables in these equations can be classified into two subsets: *independent* variables and *dependent* variables. Independent variables often correspond to input to a phenomenon or the features of a physical subsystem in a phenomenon. Henceforth, independent variables will also be called *parameters*; dependent variables will be called just *variables* . For different parameter values, dependent variables have different behavior. A designer is usually interested in studying the phenomenon on a wide range of parameter

[1]Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361. This approach was first discussed in September 1990 in a proposal entitled *Integrated Symbolical and Numerical Methods for Advanced Computer Vision* to the United Air Force Office of Scientific Research. A preliminary version of this paper was presented at the Chinese Academy of Science, Chengdu, China, in July 1992.

values.

Given as input is a system of polynomial equations in variables and parameters, henceforth called a *system of parametric equations.* In certain cases, a set of constraints on parameters specifying parameter values of interest may be given separately as part of the input. An equation can be viewed as a nonlinear polynomial equation in variables, with the coefficient of a term in a polynomial being an expression (henceforth called a *parametric constraint expression* or simply a *constraint* purely in terms of parameters).[2] Instead of having to repeatedly solve polynomial equations for many different specific parameter values, the objective is to solve the equations once for all, and compute solved parametric forms. Solutions for specific parameter values can be obtained from a solved parametric form without much additional computations.

A simple approach for solving systems of parametric equations is proposed. The approach is used first to develop an algorithm for computing a *parametric Gröbner basis* from a system of parametric polynomials. The construction is illustrated using many examples. Then the approach is used to develop an algorithm for computing a *parametric characteristic set.* For this construction, the discussion is brief because of space limitations. The construction is illustrated using an example. More details can be found in an expanded version of this paper [9]. From a parametric Gröbner basis and characteristic set, solutions of parametric polynomial equations can be extracted.

The discussion in this paper is limited to parametric constraints expressed as polynomials over the parameters. When parameters are specialized to specific concrete values from some ground field, then a constraint expression evaluates to a value in the ground field also; the ground field used in this paper is that of the field of complex numbers. The proposed approach should be applicable for more general parametric constraints also (in particular, constraint expressions could be transcendental, trigonometric or radical expressions, or even differential polynomials) insofar as they constrain the coefficients of the polynomials, and constraints can be manipulated to determine whether they are consistent or not.

[2] We apologize for the abuse of the terminology since often by a constraint, we would mean a formula and not a constraint expression. We are hoping that the context would be able to disambiguate the use of the phrase constraint without causing any confusion.

1.2.1 Properties of solved parametric forms

Below are listed desirable features of a solver for a system of parametric equations and the output generated by the solver.

1. The output should be in a solved form with constraints on parameters. The output should be in sufficient detail so that for particular parameter values, minimal (preferably no) additional computation has to be performed to obtain a solved form from which solutions can be extracted. For example, a parametric Gröbner basis should remain a Gröbner basis under a *specialization*, i.e. when specific values for parameters are substituted. A parametric characteristic set should remain a characteristic set under a specialization.
2. The output should generate results for all possible values of parameters (that satisfy the constraints on parameters in case such constraints are specified as a part of the input).
3. Parametric constraints under which the system of equations does not have a solution should be clearly identified, and should not be mixed with parametric constraints for which the system has a solution.
4. Parametric constraints under which the system of equations has finitely many solutions should also be separately identified, and not mixed with parametric constraints for which the system has infinitely many solutions. It will be preferred that a solver does not mix parametric constraints leading to solution spaces of different dimensions.

1.2.2 Overview of the approach

The approach discussed here is straightforward and easy to understand. Polynomial computations are performed by making assumptions about the parametric constraint expressions arising in the leading coefficient of the polynomials. For Gröbner basis constructions, polynomials are manipulated in their distributed representation, whereas for characteristic set constructions, polynomials are manipulated in their recursive representation. Assumptions about parametric constraint expressions are collected as constraint sets. The concept of a constrained polynomial is introduced as a pair consisting of a finite set of constraints and a polynomial such that the coefficients of its terms are interpreted with respect to the constraint set.

The leading coefficient of a constrained polynomial may or may not be nonzero with respect to its associated constraint set. If the leading coefficient cannot be determined to be nonzero with respect to its constraint set, the constrained polynomial is considered ambiguous; otherwise, it is considered unambiguous. Additional constrained polynomials may be generated from an ambiguous constrained polynomial by extending its constraint set to make its leading coefficient nonzero. Apart from that, the basic structure of computations generated by algorithms remains the same.

For Gröbner basis construction, the concepts of reduction (simplification) of a constrained polynomial by another constrained polynomial and S-polynomial of constrained polynomials are defined. For characteristic set construction, the concept of a chain of constrained polynomials and pseudodivision of a constrained polynomial using the chain are defined. In the exposition here, algorithmic aspects have been emphasized rather than theoretical aspects because algorithmic constructions are viewed as the main contributions of the paper. Theoretical results are essentially obtained in a straightforward way by modifying proofs for Gröbner basis and characteristic set constructions on polynomial systems so that they work for parameterized systems. The reader is assumed to be familiar with Gröbner basis and characteristic set constructions; otherwise, the reader can consult an introductory paper by Kapur and Lakshman [10].

1.2.3 Organization

In the next section, assumptions about constraints on parameters are stated. Minimal requirements on a constraint solver are discussed. The notion of constrained polynomials is introduced. The definitions of degree, head term, head coefficient, and head monomial are extended to constrained polynomials in distributed representation. Unambiguous and ambiguous constrained polynomials are defined.

Section 3 extends Gröbner basis constructions to constrained polynomials. Key concepts such as S-polynomials and reduction are extended. A parametric Gröbner basis is defined for a set of parametric polynomials. Parametric Gröbner bases can be used for analyzing solutions of parametric polynomials, computing dimension, and other properties of solution sets. These constructions are illustrated on a number of examples.

Section 4 extends characteristic sets to constrained polynomials. De-

gree, head term, initial, class, etc. are defined for constrained polynomials in recursive representation. Definitions of pseudodivision and chains are extended to work on constrained polynomials. Parametric characteristic sets in Wu's sense are defined. They can be used to analyze solutions of parametric polynomial equations. An example illustrating the construction is briefly discussed.

During the development of this approach, we came to know about related approaches for linear equations by Sit [16, 17], and for nonlinear polynomial equations based on Wu's characteristic set method by Chou and Gao [4], and also for nonlinear polynomial equations based on the Gröbner basis method by Weispfenning [18]. Apart from the fact that the proposed approach is simpler and easier to understand, there appear to be considerable differences between the methods outlined here and in the above cited papers. Some of these differences are illustrated later using examples.

1.3 Constraints and constrained polynomials

1.3.1 Constraints

A constraint is, in general, viewed as a formula over parameters, which is true or false based on values substituted for free parameters appearing in the constraint. Let P, called a *parameter space*, stand for the set of values that a free parameter appearing in a constraint can take.[3] Given a substitution σ of parameters from P (also called a *specialization* or *instantiation*), written as $\{u_1 \leftarrow v_1, \cdots, u_m \leftarrow v_m\}$, where $u_1, \cdots, u_m$ are parameters, and $v_1, \cdots, v_m \in P$, a constraint c and a constraint expression cp specializes or evaluates under σ. If all free variables in c (cp respectively) are specialized, then $\sigma(c)$ ($\sigma(cp)$, respectively) specializes to a truth value (a value, respectively).

A constraint c is satisfiable (or consistent) if there is a specialization σ such that $\sigma(c)$ is true; otherwise c is unsatisfiable (or inconsistent). Similarly, a set C of constraints, called a *constraint set*, is *consistent* if and only if there is a specialization such that the constraints in C are satisfiable using that specialization; a constraint set C is thus a conjunction of constraints. C is *inconsistent* if the constraints in C

[3] In general, parameters may be typed, i.e. different parameters may range over different sets of values. The proposed approach generalizes to typed parameters.

cannot be simultaneously satisfied no matter what specialization is used. Let $spcl(C)$ be the set of all specializations satisfying constraints in C.

Given a constraint c, let $neg(c)$ stand for the negation of c to mean that for a given σ substituting for all free parameters in c, $\sigma(c)$ is true if and only if $\sigma(neg(c))$ is false, as well as that the set $\{c, neg(c)\}$ is inconsistent. Given a consistent constraint c and a consistent constraint set C, $C \cup \{c\}$ or $C \cup \{neg(c)\}$ is consistent; c is consistent with C iff $C \cup \{c\}$ is consistent. A constraint c is *dependent* with respect to a consistent C if and only if exactly one of $C \cup \{c\}$ and $C \cup \{neg(c)\}$ is consistent. A constraint c is *independent* of a constraint set C if both $C \cup \{c\}$ as well as $C \cup \{neg(c)\}$ are consistent. Further, $C \models c$ iff for every specialization σ, if $\sigma(C)$ is true, $\sigma(c)$ is true, i.e. $C \cup \{neg(c)\}$ is inconsistent.

Requirements on constraint solver

The main requirement on a constraint solver is that it should be a decision procedure for determining consistency of a constraint set C. For efficiency, it should be preferably incremental in the sense that the consistency check of $C \cup \{c\}$ as well as $C \cup \{neg(c)\}$ should be able to use intermediate results from the consistency check of C.

Polynomial equations and inequations as constraints

In the above discussion, a logical view of constraints has been adopted without fixing a language for expressing constraints. In the rest of the paper, however, constraints are assumed to be of the following forms: $cp = 0$ or $cp \neq 0$, where cp is a polynomial over the parameters. If c is an equation $cp = 0$, then $neg(c)$ is the inequation $cp \neq 0$ and similarly, if c is an inequation $cp \neq 0$, then $neg(c)$ is $cp = 0$. The parameter space P is assumed to be an algebraically closed field, for example, the field of complex numbers. The constraint $cp = 0$ (respectively, $cp \neq 0$) is *satisfiable* (or *consistent*) iff there exists a specialization σ such that $\sigma(cp)$ evaluates to 0 (respectively, does not evaluate to 0), and $cp = 0$ (respectively, $cp \neq 0$) is not satisfiable if for every possible specialization σ, $\sigma(cp)$ evaluates to a value different from 0 (respectively, evaluates to 0).

When constraints are polynomial equations and inequations, a Gröbner basis algorithm or a characteristic set algorithm can be used for consis-

tency check. However, in the discussion below, it is assumed that some constraint solver is available for querying whether $C \cup \{(cp = 0)\}$ and $C \cup \{(cp \neq 0)\}$ are consistent assuming that C is consistent.

1.3.2 Constrained Polynomials

A parameterized polynomial is interpreted in the context of a set of constraints on parameters. For different values of free parameters satisfying the set of constraints, a parameterized polynomial has different structure (i.e. different terms) since the coefficients of terms may become 0. A *constrained polynomial* is defined as a pair $< C, p >$, where p is a polynomial over parameters and variables, C is a finite set of constraints over parameters. To contrast, a polynomial (without any constraints) is *an unconstrained* polynomial p and is written as $< \emptyset, p >$. Similarly, a constrained polynomial $< C, p >$ such that for every specialization σ, $\sigma(C)$ is true, is the same as the unconstrained polynomial p. As a formula, $< C, p >$ can be viewed as $C \Rightarrow p = 0$. A constrained polynomial $< C, p >$ in which C is inconsistent (unsatisfiable), is said to be *undefined.* Such constrained polynomials will never be generated in a computation. Below, by a constrained polynomial $< C, p >$, we mean the one in which C is consistent, unless stated otherwise. Further $< \sigma(C), p >$ can also be written as $< C \cup E_\sigma, p >$, where E_σ are the equations $\{u_1 = v_1, \cdots, u_m = v_m\}$ corresponding to $\sigma = \{u_1 \leftarrow v_1, \cdots, u_m \leftarrow v_m\}$. A specialization σ can be applied on a set of constrained polynomials CP as well; $\sigma(CP) = \{< \sigma(C), \sigma(p) > \mid < C, p > \in CP\}$. If a specialization σ instantiates all parameters in $< C, p >$, then either $\sigma(C)$ is inconsistent or $\sigma(C)$ is consistent and equivalent to $\emptyset$ and in that case, $\sigma(< C, p >)$ is equivalent to $\sigma(P)$, an unconstrained polynomial without any parameters.

Two constrained polynomials $< C, p >$ and $< D, q >$ are equivalent iff for every specialization σ, either both $\sigma(C)$ and $\sigma(D)$ are unsatisfiable, or both $\sigma(C)$ and $\sigma(D)$ are true and $\sigma(p) = \sigma(q)$. A constrained polynomial $< C, p >$ is *nonzero* (written as $< C, p > \neq 0$) if there exists a satisfying specialization σ of C such that $\sigma(p) \neq 0$. By slightly abusing the notation, a constrained polynomial $< C, p >$ subsumes another constrained polynomial $< D, q >$ if for every specialization σ satisfying D, $\sigma(C)$ is true and $\sigma(p) = \sigma(q)$; this implies that $\{< C, p >, < D, q >\} = \{< C, p >\}$. Obviously, $< C, p >$ is equivalent to $< D, q >$ if $< C, p >$ subsumes $< D, q >$ and $< D, q >$ subsumes

$< C, p >$. Given $< C, p >$, its *simplified form* is $< D, q >$, where D is a simplified form of C and q only includes those terms of p whose coefficients cannot be deduced to be 0 from C.

A set of constrained polynomials CP can be simplified by removing any constrained polynomial from CP that is subsumed by another constrained polynomial in CP. Two sets of constrained polynomials CP and DP are equivalent if and only if for every specialization σ, $\sigma(CP) = \sigma(DP)$.

1.3.3 Degree, head term

The degree, head term, and head coefficient of a constrained polynomial are determined by its constraint set. As in the case of polynomials, a constrained polynomial can be viewed as a constrained univariate polynomial in one of its variables (recursive representation), or a constrained multivariate polynomials (distributed representation). Recursive representation is useful for univariate resultant computation and for computing characteristic sets, whereas distributed representation is useful for multivariate resultant computation and for Gröbner basis computations. In this subection and the section on Gröbner basis computations, distributed representation of polynomials is discussed. In the section on characteristic set computations, recursive representation of polynomials will be used.

In distributed representation, the coefficient of a term in a polynomial is a constraint expression. This coefficient may or may not be zero depending upon a constraint set associated with the polynomial. Let $>$ be a total admissible ordering on terms [2]. The *potential degree* of a constrained polynomial $< C, p >$ is defined as the degree of the highest term t in p such that (i) it is not the case that $C \models (c = 0)$, where c is the coefficient of t in p, and and (ii) for all terms $t' > t$, c', the coefficient of t' in p, is such that $C \models (c' = 0)$. Term t is then called the *potential head term* of $< C, p >$; c and ct are, respectively, called the *potential head coefficient* and *potential head monomial* of $< C, p >$. Let phm, pht, phc stand, respectively, for functions for computing the potential head monomial, potential head term and potential head coefficient of a constrained polynomial.

The *degree* of a constrained polynomial $< C, p >$ is defined as the degree of the highest term, say t, in p whose coefficient, say c, is such that $C \cup \{c = 0\}$ is inconsistent, i.e. c cannot be 0 with respect to C.

Head term, head coefficient and head monomial of a constrained polynomial are defined in a similar manner. Let hm, ht, hc stand, respectively, for functions for computing the head monomial, head term and head coefficient of a constrained polynomial.

The distinction between the potential degree and degree of a constrained polynomial $< C, p >$ is that its potential degree is greater than or equal to its degree because p may have terms whose coefficients may or may not be 0 with respect to C.

1.3.4 Unambiguous and ambiguous constrained polynomials

A constrained polynomial $< C, p >$ is *unambiguous* if the potential head monomial of $< C, p >$ is also the head monomial of $< C, p >$, i.e. (i) $phm(< C, p >) = a_i t_i$ (ii) $C \cup \{a_i = 0\}$ is inconsistent, and (iii) $C \cup \{a_i \neq 0\}$ is consistent. Otherwise, $< C, p >$ is called *ambiguous*. Further, as more constraints are added to C without destroying its consistency, the head term of an unambiguous $< C, p >$ does not change, i.e. for an unambiguous $< C, p >$, for each consistent constraint set D such that $spcl(D) \subseteq spcl(C)$, $pht(< C, p >) = pht(< D, p >)$. But for an ambiguous constrained polynomial, its potential head term and head coefficient may change as more constraints are added to C.

For example, consider $p = ux^2y + v^3x^2y + vy + ux = (u + v^3)x^2y + vy + ux$, where u, v are parameters and x, y are variables. In the distributed representation, in which p is viewed as a multivariate polynomial, under the total degree ordering induced by the variable ordering $y > x$, the degree of p relative to the empty constraint set is 0; the potential degree of $< \emptyset, p >$ is 3, and the potential head term is x^2y with $u + v^3$ being the potential head coefficient. The degree of p relative to $\{u + v^3 \neq 0\}$ is 3 and the head term is x^2y. The degree of p relative to $\{u + v^3 = 0\}$ is still 0 since it cannot be said whether $v = 0, u = 0$ or not. The degree of p relative to $\{u + v^3 = 0, v \neq 0\}$ is, 1, and the head term is y. The simplified form of $< \{u + v^3 = 0\}, p >$ is $< \{u + v^3 = 0\}, vy + ux >$.

Unambiguous polynomials from an ambiguous polynomial

From an ambiguous constrained polynomial $< C, p >$ with the potential head coefficient a_i, an equivalent finite set of unambiguous constrained polynomials can be generated based on the following property: $\{< C, p >\} = \{< C \cup \{a_i \neq 0\}, p >, < C \cup \{a_i = 0\}, p >\}$. The first

element is an unambiguous constrained polynomial, whereas the second element could be ambiguous for which the construction is repeated till all the constrained polynomials generated are unambiguous.

LEMMA 1 Given a finite set of ambiguous constrained polynomials, there exists an equivalent finite set of unambiguous constrained polynomials, and this set can be obtained using the above algorithm.

Compactification

It is also possible to define a transformation from unambiguous constrained polynomials to possibly ambiguous constrained polynomials, especially if a set includes constrained polynomials with the same polynomial component but possibly with different constraint sets. Using the property that $\{< C \cup \{a_i \neq 0\}, p >, < C \cup \{a_i = 0\}, p >\} = \{< C, p >\}$ if $phc(< C, p >) = a_i$, and neither $C \models (a_i = 0)$ nor $C \models (a_i \neq 0)$. In general, given two constrained polynomials $< C, p >$ and $< D, p >$, $\{< C, p >, < D, p >\} = \{< E, p >\}$ if it is possible to generate a constraint set E that corresponds to the disjunction of constraint sets C and D since $spcl(E) = spcl(C) \cup spcl(D)$. Often this may be possible because of the way constraints get associated with a constrained polynomial.

1.4 Gröbner Basis Computation

In this section, Gröbner basis computations are extended to constrained polynomials. The reader is assumed to be familiar with concepts and definitions related to Gröbner basis computations, in particular the notions of reduction of a polynomial by another polynomial, S-polynomials and completion algorithm. For an introduction to Gröbner basis, the reader may consult [1, 2, 7, 10]. In a constrained polynomial $< C, p >$, the constraint set C is assumed to be consistent. Further a specialization σ is assumed to substitute values for all parameters in a set of constrained polynomials.

Definition 1 A finite set GCB of constrained polynomials is a parametric Gröbner basis if and only if for every specialization σ of parameters in GCB, $\sigma(GCB)$ is equivalent to a finite set GB of polynomials that is a Gröbner basis.

The above definition captures the fact that for any specialization, a Gröbner basis can be obtained from a parametric Gröbner basis by substituting for the parameters and simplifying the constrained polynomials, thus satisfying one of the requirements for solved forms stated in the introduction.

Definition 2 A finite set GCB of constrained polynomials is a parametric Gröbner basis of a set CB of constrained polynomials if and only (i) GCB is a parametric Gröbner basis, and (ii) if for every specialization σ of parameters in CB, $\sigma(GCB)$ is equivalent to a finite set GB of polynomials, which is a Gröbner basis, generating the same ideal as the ideal generated by the set B of polynomials equivalent to $\sigma(CB)$.

1.4.1 S-polynomial computations

An important operation in a Gröbner basis computation is that of computing an *S-polynomial* of a pair of distinct polynomials (also called *critical pair* in the term rewriting literature). The simplification of a polynomial by another polynomial can be viewed as a special case of S-polynomial computation. Below we extend the concept of an S-polynomial to constrained polynomials.

For two nonequivalent constrained polynomials $< C, p >$ and $< D, q >$ such that $C \cup D$ is consistent, their *constrained S-polynomial* is defined as: Let $phm(< C \cup D, p >) = a_i t_i$ and $phm(< C \cup D, q >) = a_j t_j$ such that there exist smallest f_i, f_j, $f_i t_i = f_j t_j$, $f_i \neq t_j$, and $f_j \neq t_i$,

$$s - poly(< C, p >, < D, q >) =< C \cup D, a_j f_i p - a_i f_j q > .$$

For example, given

$$cp_1 =< \{v \neq 0\}, vxy + ux^2 + x >, \quad cp_2 =< \{u \neq 0\}, uy^2 + x^2 >,$$

since the combined constraint set $\{u \neq 0, v \neq 0\}$ is consistent, and assuming a lexicographic order on terms defined by the variable order $y > x$, $phm(< \{u \neq 0, v \neq 0\}, vxy + ux^2 + x >) = vxy$ and $phm(< \{u \neq 0, v \neq 0\}, uy^2 + x^2 >) = uy^2$; $a_1 = v, a_2 = u, t_1 = xy, f_1 = y, t_2 = y^2, f_2 = x$. The S-polynomial of cp_1 and cp_2 is

$$cp_3 = < \{u \neq 0, v \neq 0\}, u^2x^2y + uxy - vx^3 > .$$

The S-polynomial of two unambiguous constrained polynomials need not be unambiguous.

LEMMA 2 For any specialization σ satisfying $C \cup D$ and the constrained S-polynomial $< C \cup D, a_j f_i p - a_i f_j q >$ of $< C, p >$ and $< D, q >$, let p', q', r' be the polynomials equivalent to $\sigma(< C, p >), \sigma(< D, q >)$, $\sigma(< C \cup D, a_j f_i p - a_i f_j q >)$, respectively. Then, r' is the S-polynomial of p' and q'.

The above lemma establishes the soundness of S-polynomial computation. It is not necessary to assume that any of $< C, p >$ and $< D, q >$ be unambiguous, because even if a_i or a_j is 0 for a particular specialization, the result is still in the required ideal. It is perhaps better to define S-polynomials for pairs of unambiguous polynomials only; however, we do not have much experimental experience with these two different heuristics to say which one is better. All the calculations below are illustrated using unambiguous polynomials.

1.4.2 Reduction

For reduction (rewriting), it is more useful to have both the constrained polynomials to be unambiguous as otherwise polynomials may not be reducing!! If the constrained polynomial being reduced is ambiguous and its head coefficient can be 0, the result may not be smaller than the original constrained polynomial. If the constrained polynomial used to reduce another polynomial is ambiguous and its head coefficient can be 0, then again no reduction is taking place.

Reduction as a special case of S-polynomial Computation

Given two unambiguous constrained polynomial $< C, p >$ and $< D, q >$, $< C, p >$ can be *reduced* (*rewritten*) by $< D, q >$ if $C \cup D$ is consistent, $hm(< C \cup D, p >) = hm(< C, p >) = a_i t_i$, $hm(< C \cup D, q >) = hm(< D, q >) = a_j t_j$ and $t_i = f_j t_j$ for some f_j. In that case, $< C \cup D, a_j p >$ is said to *reduce to* (or, *rewrite to*) $< C \cup D, a_j p - a_i f_j q >$ in a single step.[4] It is easy to see that the result is smaller than the $< C \cup D, a_j p >$ in a well-founded order on the polynomial part of the constrained polynomials. This ensures that the reduction process always terminates. The reader would have noticed that this computation is a special case of S-polynomial computation discussed above.

[4] Since p is in an ideal implies $a_j p$ is also in the ideal, it is okay to reduce $a_j p$ instead of p.

Unlike in the case of the classical Gröbner basis construction, $< C \cup D, a_j p - a_i f_j q >$ cannot replace $< C, p >$ unless $C \models D$ as otherwise for a specialization σ satisfying C but not D, the ideal would not be the same. If $C \models D$, then $< C, p >$ can be replaced by $< C, a_j p - a_i f_j q >$ or deleted if $< C, a_j p - a_i f_j q >$ is equivalent to $< C, 0 >$.

Reduction as Replacement

It is possible to define reduction so that the constrained polynomial being reduced can always be replaced by a finite set of constrained polynomials. If it is not the case that $C \models D$, then $< C, p >$ can be replaced by $< C \cup D, a_j p - a_i f_j q >$ and the set $T = \{< C \cup \{neg(d)\}, p > \mid d \in D, C \cup \{neg(d)\}$ is consistent and p does not simplify to 0 with respect to $C \cup \{neg(d)\}\}$. The family of constraint sets $\{C \cup \{neg(d)\} \mid d \in D\}$ represents all specializations which satisfy C but do not satisfy D.[5] A constrained polynomial $< C, p >$ is said to reduce in a single-step to $\{< C \cup D, a_j p - a_i f_j q >\} \cup T$ in which at least one polynomial is smaller than p and all other constrained polynomials have constraint sets which are strictly implied by C.

A single-step reduction on a finite set S of constrained polynomials is defined as follows: S reduces to S' in one step by $< D, q >$ if there is a constrained polynomial $< C, p > \in S$ that reduces by $< D, q >$ in a single-step to a finite set T of constrained polynomials and $S' = S - \{< C, p >\} \cup T$. The multi-step reduction is then the transitive closure of the single-step reduction. Multi-step reduction terminates because at least one polynomial in the polynomial components of constrained polynomials is getting smaller in the well-founded order.

An unambiguous constrained polynomial $< C, p >$ is *reduced* (or, in a *normal form*) with respect to a set of unambiguous constrained polynomials CP if it cannot be reduced by any element in CP. Similarly, a set S of unambiguous constrained polynomials is reduced with respect to CP if every polynomial in S is reduced with respect to CP.

A constrained polynomial $< C, p >$ reduces to 0 with respect to a set of unambiguous constrained polynomials CP if $< C, p >$ reduces in finitely many steps to a finite set of constrained polynomials equivalent

[5] If we use complex constraints that are propositional formulas built from basic constraints of the form $ce = 0$, then an alternate and more compact representation would be $C \cup \{neg(d_1) \vee \cdots \vee neg(d_k) \mid d_1, \cdots, d_k \in D\}$. A disjunction of constraints that are equations can be replaced by the product of the polynomials in the equations, i.e. $ce_1 = 0 \vee ce_2 = 0$ can be replaced by $ce_1 * ce_2 = 0$.

to 0.

LEMMA 3 Given two nonequivalent unambiguous $< C, p >$ and $< D, q >$ such that $C \cup D$ is consistent, and $hm(< C \cup D, p >) = hm(< C, p >) = a_i t_i$, $hm(< C \cup D, q >) = hm(< D, q >) = a_j t_j$, $t_i = f_j t_j$ for some f_j, and $< C, a_j p >$ reduces by $< D, q >$ to $S = \{< C \cup D, a_j p - a_i f_j q >\} \cup \{< C \cup \{neg(d)\}, p > \mid d \in D, C \cup \{neg(d)\}$ is consistent, $< C \cup \{neg(d)\}, p >\neq 0, \}$. For every specialization σ satisfying C, if σ also satisfies D, then p' reduces by q' to r', where p', q', r' are equivalent to $\sigma(< C, p >)$, $\sigma(< D, q >)$, $\sigma(< C \cup D, a_j p - a_i f_j q >)$, respectively, and r' is in the ideal of p' and q'. Further, if σ does not satisfy D, then S includes a constrained polynomial $< D', p >$, where D' is satisfied by σ.

1.4.3 Test for a Parametric Gröbner Basis

A test for a set of unambiguous constrained polynomials to be a parametric Gröbner basis can be given using S-polynomial computation. (If the set includes ambiguous constrained polynomials, they can be replaced by an equivalent set of unambiguous constrained polynomials; the test applies to a set including ambiguous polynomials also.)

THEOREM 1 A set CP of constrained polynomials is a parametric Gröbner basis if and only if for every pair of nonequivalent unambiguous constrained polynomials $< C, p >$ and $< D, q >$ from CP such that $C \cup D$ is consistent, their S-polynomial (i.e. the set of unambiguous constrained polynomials equivalent to it) reduces to 0 by CP.

The proof of the theorem follows the pattern of a proof in the case of the classical Gröbner basis test for polynomials.

1.4.4 Parametric Gröbner Basis Algorithm

The above test can be used to design an algorithm for computing a parametric Gröbner basis from a set of constrained polynomials as in the case of a classical Gröbner basis algorithm on polynomials. A simple version of such an algorithm takes a finite set of polynomials as input, makes constrained polynomials out of them by associating the empty constraint set with each polynomial and repeatedly performs the following steps:

1. replace every ambiguous constrained polynomial by an equivalent set of unambiguous constrained polynomials,

2. for every pair of nonequivalent unambiguous constrained polynomials,
   (a) generate an S-polynomial, if any,
   (b) find an equivalent set of unambiguous constrained polynomials to it,
   (c) reduce each of these unambiguous constrained polynomials to a normal form, which itself is a finite set of constrained polynomials, and
   (d) if a nonzero normal form is generated, add all nonzero constrained polynomials in a normal form to the basis, and compute new S-polynomials with other polynomials in the basis.

This process is continued until all S-polynomials among pairs of constrained polynomials have been considered or reduce to 0. Suitable data structures and book keeping mechanisms can be developed to avoid unnecessary computations. Optimizations suggested in the literature for identifying unnecessary S-polynomials extend to parametric Gröbner basis computations also.

The termination of the above algorithm is based on Hilbert's basis condition and the following additional argument. Every term in a constrained polynomial could possibly become a head term. At any given iteration, there are only finitely many possibilities for generating different possible Gröbner bases assuming that all parameters are specialized. If the above algorithm does not terminate, there is an infinite path by König's lemma since the branching factor is finite in the tree of possible Gröbner bases. And, the existence of an infinite path is impossible since that contradicts Hilbert's basis condition.

The method presented here has been tried on many examples including those in [18] and [4]. Our outputs are different from the ones reported in [18]. Example 1 in [18], for instance, is a discussion of a set of two generic univariate quadratic polynomials; a Gröbner basis given in [18] includes two parametric cubic polynomials. In our computation, higher degree polynomials will not be computed.

From a parametric Gröbner basis consisting of unambiguous constrained polynomials, a subset which constitutes a parametric Gröbner basis with respect to a constraint set C can be extracted, and this subset serves as a parametric Gröbner basis with respect to C. For every constrained polynomial $< D, p >$ in a parametric Gröbner basis such that $C \cup D$ is consistent, a new constrained polynomial $< D \cup C, p >$ is included. For a specialization σ, the subset of constrained polynomials

whose constraints are satisfied by σ, is equivalent to a set of polynomials that constitute a Gröbner basis in the usual sense. A Gröbner basis thus obtained may have redundant elements; any polynomial in a Gröbner basis whose head monomial can be simplified using other polynomials is redundant.

Using Parametric Gröbner Basis for Analyzing Solutions

Just like Gröbner basis for polynomials, a parametric Gröbner basis can be used to study and analyze the solution space of a system of parametric polynomial equations with respect to a constraint set on parameters. It is possible to compute the dimension of different components of the variety, as well as compute other properties of the associated ideal and variety for different constraints on parameters. This is done in a way similar to computing dimension of a polynomial ideal from its Gröbner basis. For parametric polynomials in which for every parameter, the ideal generated is zero-dimensional, it is possible to use the basis conversion algorithm of [6] to obtain a parametric Gröbner basis with respect to a lexicographic ordering from a Gröbner basis constructed using another term ordering such as degree ordering or reverse lexicographic ordering. Computations of solutions from parametric Gröbner bases are briefly illustrated later.

Example 1: We illustrate the Gröbner basis construction on a simple example that models a chemical system and explains chemical equilibrium. The example is discussed in [18] and [4] so it also contrasts the proposed approach with methods discussed in [18] and [4] as well as illustrates differences in computational steps and outputs. The set of input polynomials is:

$\{p_1 = x_4 - (a_4 - a_2),\ p_2 = x_1 + x_2 + x_3 + x_4 - (a_1 + a_3 + a_4),$
$p_3 = x_1x_3x_4 - a_1a_3a_4,$
$p_4 = x_1x_3 + x_1x_4 + x_2x_3 + x_3x_4 - (a_1a_4 + a_1a_3 + a_3a_4)\},$

where a_1, a_2, a_3, a_4 are parameters, and x_1, x_2, x_3, x_4 are variables. The lexicographic ordering on terms defined by $x_1 > x_2 > x_3 > x_4$ is used.

The corresponding unambiguous constrained polynomials are:

$1 : < \emptyset, p_1 >,\ \ 2 : < \emptyset, p_2 >,\ \ 3 : < \emptyset, p_3 >,\ \ 4 : < \emptyset, p_4 > .$

Polynomials 3 and 4 can be simplified using 1 and 2 in the classical way since the associated constraint sets are empty. Polynomial 3 gets simplified to 3' and replaces 3. $3' : < \emptyset, q_3 >$, where $q_3 = (a_4 - a_2)x_2x_3 + (a_4 - a_2)x_3^2 - (a_1 + a_2 + a_3)(a_4 - a_2)x_3 + a_1a_3a_4$. Similarly, 4 can be replaced by: $4' : < \emptyset, q_4 >$, where $q_4 = (a_4 - a_2)x_2 + x_3^2 - (a_1 + a_2 + a_3)x_3 + (a_1a_2 + a_2^2 + a_2a_3 + a_1a_3 - a_2a_4)$. Neither 3' nor 4' is unambiguous; from 4' we get: $5 : < \{a_4 - a_2 \neq 0\}, q_4 >$ and $6 : < \{a_4 - a_2 = 0\}, q_4 >$, both of which are unambiguous. Similarly, from 3', we get: $7 : < \{a_4 - a_2 \neq 0\}, q_3 >$ and $8 : < \{a_4 - a_2 = 0\}, q_3 >$. Polynomial 8 is ambiguous and exhibits an interesting case since after simplification, it is equivalent to $< \{a_4 - a_2 = 0\}, a_1a_3a_4 >$. Unambiguous polynomials equivalent to it are: $9 : < \{a_4 - a_2 = 0, a_1a_3a_4 \neq 0\}, a_1a_3a_4 >$, and $10 : < \{a_4 - a_2 = 0, a_1a_3a_4 = 0\}, 0 >$.

This reveals that the original system is inconsistent for the constraint set $\{a_4 - a_2 = 0, a_1a_3a_4 \neq 0\}$. This also follows from the original system of parametric polynomials above; for parameter values satisfying these constraints, there is indeed inconsistency since if $a_4 = a_2$, then $x_4 = 0$ simplifying p_3 to a constant $a_1a_3a_4$.

Polynomial 5 can be used to simplify polynomial 7 and the result, $7' : < \{a_4 - a_2 \neq 0\}, q_7 >$, where $q_7 = x_3^3 - (a_1 + a_4 + a_3)x_3^2 + (a_1a_4 + a_3a_4 + a_1a_3)x_3 - a_1a_3a_4$, replaces 7.

The set $\{1, 2, 5, 6, 7', 9\}$ constitutes a parametric Gröbner basis.

It can be easily checked that every specialization of the above parametric Gröbner basis that satisfies one of these constraint set $\{a_4 - a_2 \neq 0\}$, $\{a_4 - a_2 = 0, a_1a_3a_4 = 0\}$ as well as $\{a_4 - a_2 = 0, a_1a_3a_4 \neq 0\}$ is a Gröbner basis. Further, the above constraint sets constitute a complete cover in the sense every specialization satisfies exactly one of the constraint sets. For each constraint set, its Gröbner basis can be used to analyze solutions of the original parametric system. For the constraint set $\{a_4 - a_2 = 0, a_1a_3a_4 \neq 0\}$, the parametric equations are inconsistent; the dimension of the associated solution set is -1 because for such specializations, there are no solutions. If a specialization satisfies $\{a_4 = a_2, a_1a_3a_4 = 0\}$, then $x_4 = 0$, and x_3 can be obtained by solving q_4, and x_1 and x_2 are related by p_2, resulting in a solution set of dimension 1. For specializations in which $a_4 \neq a_2$, the solution set is zero-dimensional since for every variable, there is a polynomial in the Gröbner basis whose leading term is a powerproduct in that variable alone: $x_4 = a_4 - a_2$, values of x_3 are given by q_7, from which the corre-

sponding values of x_2 and x_1 can be obtained using q_4 and p_2.

This result is different from the comprehensive Gröbner basis given in [18] or ascending sets given in [4].[6] In [18], many extra and complicated polynomials appear in the comprehensive basis given as the result. In [4], there are too many cases given which can be easily combined; in contrast the above result is compact, very much in Sit's sense [16].

Example 2: Let us consider another example from [4] to contrast the proposed approach with Chou and Gao's approach. Consider the following set of polynomials:

$$\{y^2 - zxy + x^2 + z - 1, xy + z^2 - 1, y^2 + x^2 + z^2 - r^2\},$$

in which r is a parameter. The computations are performed using the lexicographic term ordering defined by $y > x > z$.

Unambiguous constrained polynomials generated from the input are:

$1 : < \emptyset, y^2 - zxy + x^2 + z - 1 >$, $2 : < \emptyset, xy + z^2 - 1 >$,

$3 : < \emptyset, y^2 + x^2 + z^2 - r^2 >$.

Polynomial 1 can be simplified using 3 and 2 in the classical way (because of empty constraint sets) to give: $1' : < \emptyset, z^3 - z^2 + r^2 - 1 >$, which replaces 1. The S-polynomial of 2 and 3 is also computed in the classical way to produce: $4 : < \emptyset, z^2y - y - x^3 - z^2x + r^2x >$. The S-polynomial of 1' and 4 gives a new polynomial:

$$5 : < \emptyset, zy - 2y + r^2y + zx^3 - x^3 + z^3x - zr^2x - z^2x + r^2x >.$$

The S-polynomial of 2 and 4 gives a new polynomial:

$$6 : < \emptyset, x^4 + z^2x^2 - r^2x^2 - z^2 - r^2z + z + 2 - r^2 >.$$

The S-polynomial of 3 and 4 reduces to 0. The S-polynomial of 2 and 6 also reduces to 0. Polynomial 5 simplifies 4 to: $4' : < \emptyset, q_4 >$, where $q_4 = (r^4 - 4r^2 + 3)y - (z^2 - (r^2 - 1)z + r^2 - 1)x^3 + ((r^2 - 1)z^2 - (r^4 - 2r^2 + 1)z - 2 + 2r^2)x$; 4' replaces 4. Since 4' is ambiguous, equivalent unambiguous polynomials are: $7 : < \{r^4 - 4r^2 + 3 \neq 0\}, q_4 >$, and

[6] Because of space limitations, definitions of comprehensive and reduced Gröbner bases as given in [18] and ascending sets in [4] cannot be reproduced. The reader may consult the original papers for definitions and other details.

$8 : < \{r^4 - 4r^2 + 3 = 0\}, q_4 >$. Using polynomial 7, polynomials 2, 3, and 5 reduce to 0 under the constraint set $\{r^4 - 4r^2 + 3 \neq 0\}$.

The set $\{1', 2, 3, 5, 6, 7, 8\}$ constitutes a parametric Gröbner basis. A compactified Gröbner basis consists of $\{1', 2, 3, 4', 5, 6\}$.

For any specialization satisfying any of the constraint systems $\{r^4 - 4r^2 + 3 \neq 0\}$ and $\{r^4 - 4r^2 + 3 = 0\}$, the equivalent set of polynomials obtained from the above parametric Gröbner basis is a Gröbner basis. These constraint sets obviously constitute a complete cover.

Solution sets can be generated from the above parametric Gröbner basis. The dimension of the solution set is 0 irrespective of the value of r. Since $r^4 - 4r^2 + 3 = (r^2 - 1)(r^2 - 3)$, each factor can be considered to lead to a distinct case. For $r^2 - 1 = 0$, i.e. $r = 1, -1$, the Gröbner basis is: $\{ 1' : z^3 - z^2,\ 2 : xy + z^2 - 1,\ 3 : y^2 + x^2 + z^2 - 1,\ 4' : z^2x^3,\ 5 : zy - y + zx^3 - x^3 + z^3x - zx - z^2x + x,\ 6 : x^4 + z^2x^2 - x^2 - z^2 + 1\}$. By factoring 1', we get $\{z = 0, x^4 - x^2 + 1 = 0, y^2 + x^2 - 1 = 0\}$ giving one set of solutions, and $\{z = 1, x = 0, y = 0\}$ as another solution. Similarly, for $r^2 - 3 = 0$, the following solutions are obtained from its Gröbner basis: $\{z = -1, x = 0, y^2 = 2\}$, $\{z = -1, x^2 = 2, y = 0\}$; there are additional solutions for which the z component satisfies the $\{z^2 - 2z + 2 = 0\}$. Similarly, for $(r^2 - 1)(r^2 - 3) \neq 0$, solutions can also be computed.

The ascending sets given for this example in [4] are different from the above parametric Gröbner basis, in particular they are far too many; many of these ascending sets can be combined and described using a single ascending set. The above description is much more compact and comprehensive much in the spirit of [16, 17].

Example 3: Here is a slighly complicated example picked from [18]. Consider the set of two parametric polynomials: $\{f = vxy + ux^2 + x,\ g = uy^2 + x^2\}$, in which u, v are parameters and x, y are variables. From f, the unambiguous constrained polynomials are: $1 : < \{v \neq 0\}, f >$, $2 : < \{v = 0, u \neq 0\}, f >$; $3 : < \{v = 0, u = 0\}, f >$. From g, the unambiguous constrained polynomials are: $4 : < \{u \neq 0\}, g >$, $5 : < \{u = 0\}, g >$.

The S-polynomial of 1 and 4 is: $< \{u \neq 0, v \neq 0\}, -vx^3 + u^2x^2y + ux >$; its normal form is $< \{u \neq 0, v \neq 0\}, h >$ where $h = (u^3 + v^2)x^3 + 2u^2x^2 + ux$. This constrained polynomial is ambiguous; the unconstrained polynomials equivalent to it are: $6 : < \{v \neq 0, u \neq 0, u^3 +$

$v^2 \neq 0\}, h >$, $7 : < \{v \neq 0, u \neq 0, u^3 + v^2 = 0\}, h >$. The S-polynomial of 1 and 5 is: $< \{v \neq 0, u = 0\}, ux^3 + x^2 >$, which reduces to 0 using 5. Polynomial 5 reduces by polynomial 3 to $\{< \{v = 0, u = 0\}, 0 >, < \{v \neq 0, u = 0\}, x^2 >\}$. So we can replace 5 by $5' : < \{v \neq 0, u = 0\}, x^2 >$. The S-polynomial of 1 and 6 is: $< \{u \neq 0, v \neq 0, u^3 + v^2 \neq 0\}, (u^3 + v^2)ux^4 + (u^3 + v^2)x^3 - 2u^2vx^2y - uvxy >$, which simplifies to 0 by using constrained polynomials 1 and 6. Similarly, the S-polynomial of 1 and 7 also simplifies to 0.

It can be easily verified that the set of unambiguous constrained polynomials $\{1, 2, 3, 4, 5', 6, 7\}$ constitute a parameterized Gröbner basis because every S-polynomial reduces to 0. For the case when $\{u \neq 0, v \neq 0, u^3 + v^2 \neq 0\}$, $\{1, 4, 6\}$ is a Gröbner basis. Its solution set is zero-dimensional since the Gröbner basis includes polynomials whose head terms are powers of x and y. For $\{u \neq 0, v \neq 0, u^3 + v^2 = 0\}$, $\{1, 4, 7\}$ is a Gröbner basis, and its solution set is also zero-dimensional. In the case when $\{u \neq 0, v = 0\}$, $\{2, 4\}$ is a Gröbner basis and its solution set is also zero-dimensional. The set $\{1, 5'\}$ is a Gröbner basis for the case $\{u = 0, v \neq 0\}$, and for when $\{u = 0, v = 0\}$, $\{3\}$ is a Gröbner basis. In both cases, the solution set is of dimension 1 since the Gröbner bases do not include polynomials in y.

Constrained polynomials in the above parametric Gröbner basis can be compactified easily. For example, constrained polynomials 6 and 7 can be compactified to $< \{u \neq 0, v \neq 0\}, h >$; constrained polynomials 1, 2, 3 can be compactified to unconstrained polynomial f; similarly, 4 and 5' can be compactified to g. Compactification would result in $\{f, g, < \{u \neq 0, v \neq 0\}, h >\}$ as a Gröbner basis. The reader can check that for different constraints on parameters, the resulting Gröbner basis from the compactified parametric Gröbner basis, after deleting redundant elements, is the same as above.

The above parametric Gröbner basis of $\{f, g\}$ is different from a Gröbner system, a reduced Gröbner system, as well as a comprehensive Gröbner basis given in [18]. This suggests that even though there may be many similarities in the two approaches, they are different as they produce different answers. In particular, Weispfenning's algorithm produced the polynomial $2v(u^3 + v^2)x^3y + 2(u^3 + v^2)x^3 - ux$ in a Gröbner system as well as a comprehensive Gröbner basis which is not in the above answer. Further there are many additional polynomials in the result reported in [18].

1.5 Characteristic Set Construction

The concept of a characteristic set of a set of polynomials was introduced by Ritt in [15], and has been popularized by Wu because of its success in geometry theorem proving as well as solving systems of polynomial equations [19, 20]. In [8], we developed a characterization of characteristic sets as defined by Ritt, which is different from Wu's definition of a characteristic set. The characteristic set algorithm given by Wu transforms a system of polynomial equations into a triangular form whose zero set is "roughly equivalent" to the zero set of the original system. Below, we discuss characteristic sets in Wu's sense as the readers are likely to have better familiarity with them. But the approach discussed below extends to Ritt's characteristic sets discussed in [8].[7] We extend characteristic set construction to constrained polynomials. We assume the reader is familiar with the characteristic set method; for details, the reader may consult [15, 19, 10].

1.5.1 Definitions for recursive representation of polynomials

In characteristic set computations, recursive representation of polynomials is used. A total ordering on variables is assumed to be given. A polynomial is viewed as a univariate polynomial in the highest variable appearing in it. Degree, head term, head monomial and initial of a constrained polynomial are defined using recursive representation similar to the definitions for distributed representation.

Let $>$ be a total ordering on variables $x_1 < x_2 < \cdots < x_n$. A constrained polynomial $< C, p >$ is defined to be of *potential class* i if and only if (i) for every variable $x_j > x_i$, the coefficient a_j of a term $x_j^{d_j}$ in p is 0 with respect to C, where $d_j > 0$, i.e. $< C, a_j >=< C, 0 >$, and (ii) there is a term $x_i^{d_i}$, where $d_i > 0$, whose coefficient a_i in p is such that $< C, a_i >$ is not equivalent to $< C, 0 >$. The coefficient a_i is a polynomial in parameters as well as variables. The highest such term

[7] Ritt's definition of a characteristic set of a set Σ of polynomials is that in addition to being a chain, every polynomial in the ideal generated by Σ pseudodivides to 0 using the characteristic set. That is not necessarily the case for a characteristic set as defined by Wu's algorithm.

The major difference in the algorithm discussed in [8] for constructing Ritt's characteristic set is that when a chain is used for pseudodivision, it is ensured that the initial of a polynomial in the chain is invertible with respect to the polynomials lower in the chain. This may result in splitting of the system of polynomials into many subsystems.

$x_i^{d_i}$ is called the *potential* head term of $< C, p >$, and d_i its potential degree; the coefficient a_i is called the *potential* initial of $< C, p >$.

For example, consider $p = ux^2y + v^3x^2y + vy + ux$ discussed earlier assuming $y > x$. In recursive representation, p is viewed as a univariate polynomial in y and is written as: $((u+v^3)x^2+v)y+ux$; the degree of p relative to $\{u+v^3 \neq 0\}$ is 1, and the head term is y. The head coefficient of p, also called its *initial*, is $((u + v^3)x^2 + v)$. The degree of p relative to $\{v \neq 0\}$ is also 1, and the head term is y; the head coefficient of p remains to be $((u + v^3)x^2 + v)$ since $((u + v^3)x^2 + v) \neq 0$. The potential class of $< \emptyset, p >$ is 2; its potential degree is 1, and the potential initial is $((u + v^3)x^2 + v)$.

Given a constrained polynomial $< C, p >$ in recursive form, an equivalent simplified form $< C, q >$ can be defined in a way similar to the one for distributed representation: $< C, q >$ is obtained from $< C, p >$ by deleting terms whose coefficients are 0 with respect to C. For example, the simplified form of $< \{u + v^3 = 0, v \neq 0\}, p >$ is $<< \{u + v^3 = 0, v \neq 0\}, vy + ux >$, and the simplified form of $< \{u + v^3 = 0, v = 0\}, p >$ is $< \{u = v^3 = 0, v = 0\}, 0 >$.

1.5.2 Ambiguous and unambiguous constrained polynomials

If there is a consistent superset D of C such that $< D, a_i >$ is equivalent to $< D, 0 >$, then p is ambiguous. Otherwise, $< C, a_i >$ cannot be made equivalent to 0, and p is unambiguous in which case, i, $x_i^{d_i}$ d_i and a_i are, respectively, the class, head term, degree, and initial of p also. From an ambiguous constrained polynomial $< C, p >$ (implying that its initial a_i is such that $< C, a_i >$ is ambiguous), unambiguous constrained polynomials equivalent to it can be generated. Unlike in the distributed representation case, we cannot require that $a_i \neq 0$ since a_i is itself a polynomial in parameters and variables. In fact, it is necessary to keep checking for the head coefficient until a constraint expression is arrived at; then the constraint set C is appropriately extended as in the case of distributed representation. In general, the constraint set C is extended based on the structure of a_i to make a_i unambiguous with respect to an extended constraint set that includes C. This process can be repeated till all constrained polynomials thus obtained are unambiguous. Corresponding to an ambiguous constrained polynomial, there is a finite set of unambiguous constrained polynomials equivalent to it.

For example, assuming $y > x$, consider $p = ((u + v^3)x^2 + v)y + ux$.

The constrained polynomial $< \emptyset, p >$ is ambiguous because its initial, the polynomial $(u + v^3)x^2 + v$, is ambiguous with respect to the empty set of constraints. However, $(u + v^3)x^2 + v$ is nonzero with respect to $\{u + v^3 \neq 0\}$; so $< \{u + v^3 \neq 0\}, p >$ is unambiguous, its degree is 1, and its initial is $(u + v^3)x^2 + v$. Similarly, the initial remains nonzero with respect to the constraint set $\{u + v^3 = 0, v \neq 0\}$; the degree of $< \{u+v^3 = 0, v \neq 0\}, p >$ is also 1, and the initial remains $(u+v^3)x^2+v$. The initial $(u + v^3)x^2 + v$ becomes zero with respect to the constraint set $\{u + v^3 = 0, v = 0\}$, so the degree of $< \{u + v^3 = 0, v = 0\}, p >$ drops to 0, and the potential initial is ux, which is zero with respect to $\{u + v^3 = 0, v = 0\}$. So $< \{u + v^3 = 0, v = 0\}, p >$ is equivalent to 0.

Compactification of unambiguous constrained polynomials into ambiguous constrained polynomials can be defined for recursive representation also.

1.5.3 Pseudodivision

There are three important concepts in characteristic set computations: (i) a polynomial being reduced with respect to another polynomial, (ii) the reduction of a polynomial by another polynomial by pseudodivision [12], and (iii) the initial of a polynomial being invertible [8]. The first two must be extended for Wu's algorithm.

A constrained polynomial $< C, p >$ is *reduced* with respect to an unambiguous $< D, q >$ if and only if (i) the constraint set $C \cup D$ is consistent, and (ii) the class of $< D, q >$ is i and the potential degree of x_i in $< C \cup D, p >$ is $<$ the degree of x_i in $< D, q >$.

An unambiguous $< C, p >$ *reduces* (*pseudodivides*) by another unambiguous $< D, q >$ to $< C \cup D, r >$ if (i) $C \cup D$ is consistent, (ii) $< C, p >$ is not reduced with respect to $< D, q >$, and (iii) there exist a number k (preferably the smallest such number, but is $\leq (d - d_i) + 1$, where i is the class of $< D, q >$, d_i is the degree of $< D, q >$, and d is the degree of x_i in $< C, p >$), a polynomials b (preferably not involving parameters) and another polynomial r such that

$$< C \cup D, I_q{}^k p >=< C \cup D, bq > + < C \cup D, r >,$$

where I_q is the initial of $< C \cup D, q >$, and the potential degree of x_i in r is $< d_i$. This calculation can be done in the classical way by first simplifying p and q with respect to $C \cup D$. Note that $< C \cup D, r >$ need not be unambiguous.

It is easy to see that for every specialization σ satisfying $C \cup D$, the common zeros of $\sigma(p)$ and $\sigma(q)$ are also the zeros of the remainder $\sigma(r)$; further, if $r = 0$, then the common zeros of $\sigma(p)$ and $\sigma(q)$ are the same as the zeros of $\sigma(q)$ insofar as they are not the zeros of $\sigma(I_q)$. And, $\sigma(r)$ is in the ideal generated by $\sigma(p)$ and $\sigma(q)$. Unlike in the case of Gröbner basis computations, $< C \cup D, r >$ does not replace any of $< C, p >$ and $< D, q >$; instead, it gets added to the basis.

In order for the definitions to extend naturally, we define that an unambiguous constrained polynomial $< C, p >$ *reduces* by another unambiguous $< D, q >$ to 0 if $C \cup D$ is inconsistent. Thus an unambiguous constrained polynomial $< C, p >$ is *not reduced* with respect to another unambiguous $< D, q >$ if their constraint sets clash.

Definition 3 A set $\{< C_1, p_1 >, \cdots, < C_l, p_l >\}$ of unambiguous constrained polynomials is called a chain if (i) $l = 1$ and $< C_1, p_1 >$ is not equivalent to 0, or (ii) $0 < i_1 < i_2 < \cdots < i_l$, where i_j is the the class of $< C_j, p_j >$, and $D = C_1 \cup \cdots \cup C_l$ is consistent, and $< C_j, p_j >$ is reduced with respect to each $< C_i, p_i >$, $i < j \leq l$.

A chain is also called a *triangular* form; each element in the chain is said to introduce a new variable, i.e. $< C_j, p_j >$ introduces x_{i_j}. Also notice that the chain is relevant for only those parameter values that satisfy the constraint set D. The above definition can be weakened to define a chain in the loose sense or a chain in the weak sense as defined by Wu for polynomials, in which only the initial of $< C_j, p_j >$ is reduced with respect to $< C_i, p_i >$'s [19].

Pseudodivision of a constrained polynomial can be defined with respect to a chain in the obvious way. The constrained polynomial is pseudodivided by the polynomial of the largest class in the chain, then the result is pseudodivided by the next polynomial lower in the chain and so on, until the result that is reduced with respect to every polynomial in the chain is generated.

1.5.4 Computing Parametric Characteristic Set

A parametric characteristic set should cover all values of parameters satisfying parametric constraints if given as a part of the input. A parametric characteristic set may thus include many chains, but they are disjoint in covering different parameter values. A parametric characteristic set is defined based on the definition of a characteristic set.

Definition 4 A finite set CCP of unambiguous constrained polynomials is a parametric characteristic set if (i) for any two nonequivalent $< C, p >$ and $< D, q >$ of class i in CCP, $C \cup D$ is inconsistent, and (ii) for every specialization σ of parameters, $\sigma(CCP)$ is equivalent to a set CP of polynomials that constitutes a characteristic set.

Definition 5 A finite set CCP of unambiguous constrained polynomials is a parametric characteristic set of a finite set BCP of parametric polynomials if and only if (i) CCP is a parametric characteristic set, and (ii) for every specialization σ, $\sigma(CCP)$ is equivalent to a set CP of polynomials that constitutes a characteristic set of the set of polynomials equivalent to $\sigma(BCP)$.

A parametric characteristic set can be computed using a method similar to a method used to compute a characteristic set of polynomials as discussed in [19, 20, 3, 11]. A simple algorithm for computing a parametric characteristic set works as follows. Firstly, for every ambiguous polynomial, generate an equivalent set of unambiguous polynomials. Secondly, identify a minimal chain of constrained polynomials in the basis: From a given set BCP of constrained polynomials, let $S = BCP$, $B = \emptyset, CB = \emptyset$.

1. Pick a minimal element, say $< C, p >$, in S, i.e. a constrained polynomial of the smallest class, say i, and of the smallest degree, say d_i, in its largest variable x_i, and include it in B. Let $CB = CB \cup C$.
2. Remove from S any constrained polynomial $< D, q >$ such that the degree of x_i in $< D, q >$ is $\geq d_i$ or either $CB \cup D$ is inconsistent.
3. Repeat the above two steps until S is empty.

It can be easily shown that $B \subseteq BCP$ is a chain. Polynomials in BCP are pseudodivided by B. Let T be the nonzero remainders obtained. If T is $\emptyset$, then we have identified a chain belonging to a parametric characteristic set. Otherwise BCP is augmented to include T, and the above procedure of identifying a chain in BCP is repeated. This process terminates since T includes polynomials of lower degrees than those in B.

If a chain B in BCP is identified such that every constrained polynomial in BCP has a clashing constraint set with B or pseudodivides

to 0 using B, we *extract that chain out of* BCP and compute the remaining chains. B is a chain in a parametric characteristic set of BCP; the constraint set CB associated with B represents parameter values for which B is a characteristic set. For every specialization σ satisfying CB, every polynomial in the set Σ of polynomials equivalent to $\sigma(BCP)$ pseudodivides to 0 using B', the set of polynomials equivalent to $\sigma(B)$.

One simple (but naive) way to extract out B from BCP is to define

$$BCP' = \{< D, p > \mid < C, p > \in BCP, D = C \cup \{neg(cb)\} \text{ is consistent}, \\ cb \in CB, < D, p > \neq < D, 0 >\}$$

It is easy to see that for every specialization σ satisfying CB, $\sigma(BCP')$ is the empty set, and further, for every specialization σ not satisfying CB, $\sigma(BCP') = \sigma(BCP)$.[8]

If BCP' is not the empty set, assign BCP' to S and repeat the above steps of identifying and extracting out additional chains of a parametric characteristic set. The set of all chains collected in this way is a parametric characteristic set.

Many variations of the above simple algorithm are possible and many heuristics can be incorporated. Many examples, including those in [18, 4], have been successfully worked out using the above algorithm.

1.5.5 Example

Let us consider **Example 3** of the previous section used for illustrating Gröbner basis computations, and apply the characteristic set construction. The input is $\{f = vxy + ux^2 + x,\ g = uy^2 + x^2\}$ with $y > x$. Unambiguous constrained polynomials equivalent to f are: $1 : < \{v \neq 0\}, f >,\ \ 2 : < \{v = 0, u \neq 0\}, f >, 3 : < \{v = 0, u = 0\}, f >$. Unambiguous constrained polynomial equivalent to g are:
$4 : < \{u \neq 0\}, g >,\ \ 5 : <\ \{u = 0\}, g >$.

Applying the above algorithm, we get $\{3\}$ as a chain of a parametric characteristic set for the case when $\{u = 0, v = 0\}$. This chain is extracted giving a replacement of 5 by $5' : < \{u = 0, v \neq 0\}, g >$. From $\{1, 2, 4, 5'\}$, another chain $\{5', 1\}$ for the case when $\{u = 0, v \neq 0\}$ is generated. After extracting this chain, we have: $\{1', 2, 4\}$, where $1' : < \{u \neq$

[8] As in the case of Gröbner basis construction, a compact representation of BCP' can be constructed using constraints which are propositional formulas over basic constraints.

$0, v \neq 0\}, f >$. There is another chain $\{2, 4\}$ for $\{u \neq 0, v = 0\}$. After extracting this chain, we get: $\{1', 4'\}$, where $4' : < \{u \neq 0, v \neq 0\}, g >$.

Polynomial 1' is used to pseudodivide 4' and the result is: $< \{v \neq 0, u \neq 0\}, r >$, where $r = (u^3 + v^2)x^4 + 2u^2x^3 + ux^2$. Since the result is ambiguous, we get additional unambiguous polynomials equivalent to it: $6 : < \{v \neq 0, u \neq 0, u^3 + v^2 \neq 0\}, r >$, and $7 : < \{v \neq 0, u \neq 0, u^3+v^2 = 0\}, r >$. There are two more chains: $\{1', 6\}$ for the constraint set $\{v \neq 0, u \neq 0, u^3 + v^2 \neq 0\}$, and $\{1', 7\}$ for the constraint set $\{v \neq 0, u \neq 0, u^3 + v^2 = 0\}$.

A parametric characteristic set consists of:

$3 : < \{v = 0, u = 0\}, f >$, $5' : < \{u = 0, v \neq 0\}, g >$,

$2 : < \{v = 0, u \neq 0\}, f >$, $4'' : < \{u \neq 0, v = 0\}, g >$,

$1'' : < \{u = 0, v \neq 0\}, f >$, $1''' : < \{u \neq 0, v \neq 0, u^3 + v^2 \neq 0\}, f >$,

$1'''' : < \{u \neq 0, v \neq 0, u^3 + v^2 = 0\}, f >$,

$6 :< \{v \neq 0, u \neq 0, u^3 + v^2 \neq 0\}, r >$,

$7 : < \{v \neq 0, u \neq 0, u^3 + v^2 = 0\}, r >$.

In [20], Wu gave a structure theorem and showed how it could be used for computing zeros of polynomial equations from their characteristic sets. This construction can be extended to constrained polynomials. Below we illustrate how solutions can be obtained from some of the chains in the above parametric characteristic set.

The chain $\{1''', 6\}$ can be used to generate the zeros as follows: For $1'''$, the initial vx should be nonzero which requires that $x \neq 0$; this condition can be used to simplify $\{1''', 6\}$, to give a new set $\{1''', 6'\}$, where $6' : < \{v \neq 0, u \neq 0, u^3+v^2 \neq 0\}, (u^3+v^2)x^2+2u^2x+u >$. From constrained polynomials $1'''$ and $6'$, zeros can be computed by solving 6' and then substituting the values of x to get the corresponding values of y.

It is also necessary to consider the case when the initial $x = 0$. It is not enough to add the polynomial $x = 0$ to the chain $\{1''', 6\}$; instead this condition of an initial not being zero needs to be added to the whole basis from which the chain was generated. Alternately, $x = 0$ can be added to the original set and a parametric characteristic set can be computed for $\{u \neq 0, v \neq 0, u^3 + v^2 \neq 0\}$.

When $x = 0$ is added to the original basis, f becomes 0, and g becomes uy^2. We thus get a parametric characteristic set consisting $< \{u \neq$

$0\}, uy^2 >, < \emptyset, x >$, which gives the zero $x = 0, y = 0$ for the case when $u \neq 0$. For the case when $u = 0$, we get a 1-dimensional zero set, which is $x = 0$.

From the chain $\{1''', 7\}$, we get 7' under the condition that $x \neq 0$: $< \{u \neq 0, v \neq 0, u^3+v^2 = 0\}, 2u^2x+u >$, which gives a zero-dimensional zero set whose x component is $-\frac{1}{2u}$. Zero sets can be computed from other chains in a similar manner. Chain $\{5\}$ for the case $u = 0, v = 0$, also gives a 1-dimensional zero set, which is $x = 0$.

For computing zero sets, optimizations such that factoring and splitting can be performed to speedup the computations of chains insofar as the zero sets are preserved.

1.6 Concluding remarks and future research

A simple but powerful approach for solving systems of parametric polynomial equations has been discussed. The approach was applied to define and illustrate parametric Gröbner basis construction from a system of parametric polynomial equations. As was evident from examples discussed above, different values for parameters may result in different systems and associated solution sets, including cases when for some parameter values, system may be inconsistent and thus have no solution, as well as for other parameter values, the same system may have infinitely many solutions. The approach was also briefly illustrated for characteristic set constructions and generating solutions using characteristic sets. The same approach should extend to other symbolic elimination techniques including univariate resultants and multivariate resultants.

From the discussion, the reader must have noticed that sometimes unnecessary and excessive branching may be done because it is guided by considering the coefficient of the leading term in a parametric polynomial. Further research is necessary to study how branching can be avoided. In [16, 17], some ideas are discussed in the context of solving linear systems of parametric equations, which may be useful for nonlinear polynomial equations also.

Given that symbolic computation algorithms do not often finish on many large examples (often because of lack of space since large intermediate computations may be generated), numerical algorithms such as those based on homotopy and continuation techniques must be em-

ployed. It would be interesting to study how this approach can be used for solving systems of parametric polynomial equations using numerical methods.

Acknowledgements: Comments by Lakshman Y.N. and a referee on an earlier draft helped in improving the presentation of the paper.

Bibliography

[1] B. Buchberger (1965), *Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal*, Ph.D. Thesis (in German), Universitat Innsbruck.

[2] B. Buchberger (1985), "Gröbner bases: An Algorithmic method in Polynomial Ideal theory", in *Multidimensional Systems Theory*, N.K. Bose, ed., D. Reidel Publishing Co., 184-232.

[3] S.-C. Chou and X.-S. Gao (1990), "Ritt-Wu's decomposition algorithm and geometry theorem proving," Proc. *10th Intl. Conf. on Automated Deduction (CADE-10)*, Kaiserslautern, Germany, LNCS Springer Verlag, 207-220.

[4] S.-C. Chou and X.-S. Gao (1992), "Solving parametric algebraic systems," Proc. *ISSAC 1992*, Berkeley, CA. Also in Mathematics-Mechanization Research Reprints, No. 7, March 1992.

[5] B. Donald, D. Kapur, and J.L. Mundy (eds.) (1992), *Symbolic and Numerical Computation for Artificial Intelligence*, Academic Press.

[6] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora (1989), *Efficient computation of zero-dimensional Gröbner bases by change of ordering*, Technical Report 89–52, LITP, Universite Paris.

[7] C.M. Hoffman (1989), *Geometric and solid modeling: An introduction*, Morgan Kaufmann.

[8] D. Kapur (1992), A theory of characteristic sets, Unpublished Manuscript, Department of Computer Science, State University of New York, Albany, NY.

[9] D. Kapur (1992), Solving systems of parametric polynomial equations, Unpublished Manuscript, Department of Computer Science, State University of New York, Albany, NY.

[10] D. Kapur and Lakshman Y.N. (1992), "Elimination Methods: An Introduction," in *Symbolic and Numerical Computation for Artificial Intelligence*, Donald, Kapur and Mundy (eds.), Academic Press, 1992.

[11] D. Kapur and H. Wan (1990), "Refutational Proofs of Geometry Theorems via Characteristic Set Computation," Proc. of *the SIGSAM 1990 Intl. Symp. on Symbolic and Algebraic Computation - ISSAC '90*, Japan, August 1990.

[12] D.E. Knuth (1980), *Seminumerical algorithms: The art of computer programming*, 2, Second Edition, Addison Wesley, 407-408.

[13] A.P. Morgan (1987), *Solving Polynomial Systems using Continuation for Scientific and Engineering Problems*, Prentice Hall, Englewood Cliffs, NJ.

[14] V.-D. Nguyen, J.L. Mundy, and D. Kapur (1991), "Modeling Generic Polyhedral Objects by Constraints," Proc. of *Computer Vision and Pattern Recognition*, Hawaii, June 1991,

[15] J.F. Ritt (1950), *Differential Algebra*, AMS Colloquium Publications.

[16] W.Y. Sit (1991), "A theory for parametric linear systems," Proc. *ISSAC 1991*, Bonn, Germany, 112-121.

[17] W.Y. Sit (1992), *An algorithm for solving parametric linear systems.* Unpublished Manuscript. To appear in *J. Symbolic Computation.*

[18] V. Weispfenning (1990), *Comprehensive Gröbner bases*, MIP-9003, Fakultät für Mathematik und Informatik, Universität Passau, Germany.

[19] W. Wu (1986), "Basic principles of mechanical theorem proving in geometries", *J. of System Sciences and Mathematical Sciences*, **4**(3), (1984), 207-235. Also appeared in *J. of Automated Reasoning*, **2**, (1986), 221-252.

[20] W. Wu (1986), "On zeros of algebraic equations - an application of Ritt's principle", *Kexue Tongbao*, **31**(1), 1-5.

solutions to nonlinear analytic systems", submitted for publication.
A.P. Morgan, A.J. Sommese and L.T. Watson (1989), "Finding all isolated solutions to polynomial systems using HOMPACK", *ACM Trans. Math. Software*, **15**, 93–122.
A.P. Morgan and C.W. Wampler (1990), "Solving a planar four-bar design problem using continuation", *ASME J. Mechanical Design*, **112**, 544–550.
A.P. Morgan and L.T. Watson (1986), "Solving nonlinear equations on a hypercube", *Super and Parallel Computers and Their Impact on Civil Engineering*, M.P. Kamat, ed., ASCE Structures Congress '86, New Orleans, LA, 1–15.
A.P. Morgan and L.T. Watson (1987), "Solving polynomial systems of equations on a hypercube", *Hypercube Multiprocessors 1987*, M.T. Heath, ed., SIAM, Philadelphia, PA, 501–511.
A.P. Morgan and L.T. Watson (1989), "A globally convergent parallel algorithm for zeros of polynomial systems", *Nonlinear Analysis, Theory, Methods and Applications*, **13**, 1339–1350.
W.C. Rheinboldt and J.V. Burkardt (1983), "Algorithm 596: a program for a locally parameterized continuation process", *ACM Trans. Math. Software*, **9**, 236–241.
S. Richter and R. De Carlo (1984), "A homotopy method for eigenvalue assignment using decentralized state feedback", *IEEE Trans. Auto. Control*, **AC-29**, 148–158.
M.G. Safonov (1984), "Exact calculation of the multivariable structured-singular-value stability margin", *IEEE Control Decision Conf.*, Las Vegas, NV.
L.-W. Tsai and A.P. Morgan (1985), "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods", *ASME J. Mechanisms, Transmissions Automation in Design*, **107**, 189–200.
C.W. Wampler and A.P. Morgan (1991), "Solving the 6R inverse position problem using a generic-case solution methodology", *Mechanism Mach. Theory*, **26**, 91–106.
C.W. Wampler, A.P. Morgan and A.J. Sommese (1990a), "Numerical continuation methods for solving polynomial systems arising in kinematics", *ASME J. Mechanical Design*, **112**, 59–68.
C.W. Wampler, A.P. Morgan and A.J. Sommese (1990b), "Complete solution of the nine-point path synthesis problem for four-bar linkages", *ASME J. Mechanical Design*, to appear.
L.T. Watson (1979), "A globally convergent algorithm for computing fixed points of C^2 maps", *Appl. Math. Comput.*, **5**, 297–311.
L.T. Watson (1986), "Numerical linear algebra aspects of globally convergent homotopy methods", *SIAM Review*, **28**, 529–545.
L.T. Watson, S.C. Billups, and A.P. Morgan (1987), "HOMPACK: a suite of codes for globally convergent homotopy algorithms", *ACM Trans. Math. Software*, **13**, 281–310.
L.T. Watson and A.P. Morgan (1991), "Global optimization for polynomial programming problems using m-homogeneous polynomial homotopies", *J. Computational Appl. Math.*, to appear.
A.H. Wright (1985), "Finding all solutions to a system of polynomial equations", *Math. Comput.*, **44**, 125–133.
W. Zulehner (1988a), "A simple homotopy method for determining all isolated solutions to polynomial systems", *Math. Comput.*, **50**, 167–177.
W. Zulehner (1988b), "On the solutions to polynomial systems obtained by homotopy methods", *Numer. Math.*, **54**, 303–317.
W. Zulehner (1988c), "Numerical solution of polynomial equation systems by curve tracing", *Zeitschrift fur Angewandte Mathematik und Mechanik*, **68**, T504–T505.

Chapter 2

Elimination Methods: an Introduction

Deepak Kapur †

Institute for Programming and Logics, Department of Computer Science

State University of New York, Albany, NY 12222

kapur@cs.albany.edu

Yagiti N. Lakshman ‡

Department of Computer and Information Sciences

University of Delaware, Newark, DE 19716

lakshman@cis.udel.edu

This is an introductory paper on elimination methods applicable to a system of nonlinear polynomials equations. We discuss three different approaches for solving a system of nonlinear polynomial equations. The first approach of resultants is based on the theory of determinants, and was developed in the late 19th century and the early 20th century. The main idea is to generate from a system of nonlinear polynomial equations, a possibly larger system of independent polynomial equations such that there are as many equations as terms in the polynomials so that each term can be used as an unknown and the theory of linear system of equations can be applied. We describe the resultants of Sylvester, Bezout, Dixon's extension of Bezout's resultant, and Macaulay's resultant, followed by some recent extensions of Macaulay's resultant.

The second approach is based on polynomial ideal theory and generates special bases of polynomial ideals, called Gröbner bases. An algorithm for computing such bases was given by Buchberger and is described here. We review some recent developments in the Gröbner basis theory, by the use of which, nonlinear polynomial equations with finitely many solutions can be efficiently solved.

The third approach is based on Ritt's characteristic set construction, motivated by the analysis and decomposition of the zero sets of systems of polynomial equations. This approach has been recently popularized by Wu Wen-tsun who has impressively demonstrated its application to automated geometry theorem proving. Using Wu's method, it is possible to automatically prove, in a matter of seconds, nontrivial theorems in plane Euclidean geometry which human experts find difficult to prove.

† Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

‡ Supported by the Center for Advanced Technology in Automation and Robotics at Rensselaer Polytechnic Institute under a grant (Grant No. 90-39) from the New York State Science and Technology Foundation.

1. Introduction

In high school, we learn how to manipulate and solve a system of linear equations. In particular, we learn methods for determining whether a system of linear equations has a solution or not. With a little more effort, it is also possible to determine whether a system of equations has a single solution or infinitely many solutions. In the latter case, it is possible to study the structure of the solution space by classifying the variables into independent and dependent subsets and specifying the solutions in terms of independent variables.

A related problem of determining whether an equation c (or a system of equations) follows from a system $\mathcal{S}$ of equations can also be answered easily depending upon the number of solutions of $\mathcal{S}$. If $\mathcal{S}$ has no solution, i.e. $\mathcal{S}$ is inconsistent, then it is possible to take either of the two views – any equation follows from an inconsistent system of equations that has no solution, or alternatively, it is meaningless to ask whether an equation follows from an inconsistent set of equations. If $\mathcal{S}$ has a unique solution, then if the solution of $\mathcal{S}$ is also a solution of c, then c follows from $\mathcal{S}$. If $\mathcal{S}$ has infinitely many solutions, which can be represented by expressing each dependent variable in terms of the independent variables, the expression for each of the dependent variables can be substituted into the given equation c to check whether the equation holds. This is equivalent to checking whether the solution space of the given equation c includes the solution space of the system $\mathcal{S}$ of equations.

The above problems can also be studied in a slightly different setting using their formulation in terms of properties of matrices and determinants. That is what one often learns in a first course on linear algebra at a college level. What has been missing, of late, is a similar discussion for studying these problems for nonlinear polynomial equations. That is the case even though elegant generalizations were developed for solving these problems for nonlinear polynomial equations in the late 19th century and early 20th century. A number of books on the theory of equations were written which are now out of print. For an excellent discussion of the history of constructive methods in algebra, the reader is referred to a thought-provoking article by Professor Abhyankar (Abhyankar, 1976).

In this paper, we discuss these problems for nonlinear equations and discuss some of the approaches proposed in the late 19th century and early 20th century. We also discuss two additional constructive approaches – the approach proposed by Ritt and recently revived by Wu based on algebraic geometry, and another approach by Buchberger based on polynomial ideal theory.

We study two additional subproblems that turn out to be useful in their own right as well as intermediate steps while discussing the problems mentioned earlier. The first problem has to do with the equivalence of two systems of equations. The second problem is that of *projection* or *elimination*, which is to project the solution space of a system $\mathcal{S}$ of equations along certain dimensions and compute another system $\mathcal{S}'$ of equations in an appropriate subset of the variables, such that the solution space of $\mathcal{S}'$ coincides with the projection of the solution space of $\mathcal{S}$ on the chosen variables.

To summarize, here is a list of problems being considered. Given a system $\mathcal{S}$ of polynomial equations in n variables,

- does $\mathcal{S}$ have a solution?
- if $\mathcal{S}$ has a solution, what is the structure of the solution space? What is a good characterization of the structure of the solutions of $\mathcal{S}$?
- eliminate m variables from $\mathcal{S}$ to obtain a system of equations $\mathcal{S}'$ in the remaining $n-m$ variables such that the solutions of $\mathcal{S}'$ are the projections of the solutions of $\mathcal{S}$ to the coordinates corresponding to these $n-m$ variables.
- Given two systems of polynomial equations $\mathcal{S}$ and $\mathcal{S}'$, do they have the same solutions, or, is the set of solutions of $\mathcal{S}'$ properly contained in the set of solutions of $\mathcal{S}$?

1.1. Overview

We begin with the basic definitions of zero sets of polynomial systems and ideals. This is followed by an exposition on resultants. We discuss Dixon's formulation of the resultant of two polynomials in one variable as well as three polynomials in two variables, followed by a presentation of Macaulay's construction of the multivariate multi-polynomial resultant. The concept of the u-resultant for determining the common zeros of polynomials is discussed.

Section 3 is about Gröbner bases. The concepts of an ordering on power products and viewing polynomial equations as simplification rules are discussed. A Gröbner basis of an ideal is a basis with many useful properties. Among other things, Gröbner bases can be used to solve systems of equations. After an overview of the basic properties of Gröbner bases, we pay special attention to systems of polynomials that have finitely many common solutions. For computing the common zeros of such systems, we describe a recent algorithm due to Faugère *et al.* (1989).

Section 4 discusses Ritt's characteristic sets. Our presentation is based on Wu's treatment of characteristic sets and an algorithm for computing a characteristic set. A breadth-first algorithm, which is Ritt's original algorithm, as well as a depth-first algorithm for computing characteristic sets are described.

An expanded version of this article includes illustrations of the applications of different elimination techniques to curve and surface implicitization, detection of unfaithful parameterizations and geometry theorem-proving. The expanded version will be available as a technical report from the Department of Computer and Information Science of the University of Delaware or from Institute for Programming and Logics, the Department of Computer Science, State University of New York at Albany.

1.2. Preliminaries

Let $\mathcal{Q}$ denote the field of rational numbers and $\mathcal{C}$ denote the field of complex numbers. Unless specified otherwise, by a polynomial, we mean a multivariate polynomial with rational or integer coefficients. A univariate polynomial $p(x)$ is an element of the polynomial ring $\mathcal{Q}[x]$.

A multivariate can be viewed in one of two ways: as an element of the ring $\mathcal{Q}[x_1, \ldots, x_n]$ or as an element of the ring $\mathcal{Q}[x_1, \ldots, x_{n-1}][x_n]$. Under the first view, the polynomial $p(x_1, \ldots, x_n)$ is seen as a sum of products with nonzero coefficients, where each product

$x_1^{d_1}x_2^{d_2}\cdots x_n^{d_n}$, is called a *term* or *power product*; together with its coefficient, it is called a *monomial*; the degree of a term $x_1^{d_1}x_2^{d_2}\cdots x_n^{d_n}$ is $d_1+d_2+\cdots+d_n$.

Under the second view, the polynomial $p(x_1,\ldots,x_n)$ is seen as a univariate polynomial in x_n (also known as the *main* or *leading* variable) with coefficients that are themselves multivariate polynomials in the remaining variables. For example, the polynomial

$$q(x_1,x_2,x_3) = 2x_1^2x_2x_3^2 + x_2^2x_3^2 - 2x_1 + x_2^2 + x_3$$

can be viewed as a sum of monomials appearing in it, or can be viewed as the polynomial

$$\widehat{q}(x_3) = (2x_1^2x_2 + x_2^2)x_3^2 + x_3 + (-2x_1 + x_2^2)$$

in the variable x_3 (q could be also considered as a univariate polynomial with x_1 as the main variable or x_2 as the main variable).

The degree of a univariate polynomial $p(x)$ is the maximum degree, say d, of x in $p(x)$; the leading term of $p(x)$ is then x^d, and the leading coefficient (also called the *initial*) of $p(x)$ is the coefficient of x^d in $p(x)$. For a multivariate polynomial, the leading term and the leading coefficient can be determined only after an ordering on terms is chosen. If a multivariate polynomial is considered as a polynomial in one of its variables, say x_3 in the case of $\widehat{q}$ above, then its degree is the maximum degree in that variable. For $\widehat{q}$, the degree of the polynomial is 2. Its leading term is x_3^2 and the leading coefficient is the polynomial $2x_1^2x_2 + x_2^2$.

A univariate polynomial $p(x)$ is said to vanish at $x = a$ if the polynomial p evaluates to zero when a is uniformly substituted for x in p. The value of p at a is denoted by $p(a)$; if $p(a) = 0$, then a is called a *zero* of $p(x)$. Equivalently, the polynomial equation $p(x) = 0$ is said to have a solution a. The domain from which the values are picked to be checked for zeros of p is very important. It is possible for p to have a zero in one domain and not in another domain. For instance, x^2+1 does not have a zero in the reals, but it does have a zero in the complex numbers. A univariate polynomial of degree n with complex coefficients has exactly n complex roots. They may or may not be distinct.

The above definitions extend to multivariate polynomials also. Given $p(x_1,\ldots,x_n)$, an n-tuple $(a_1,a_2,\ldots,a_n) \in \mathcal{C}^n$, the affine n-space over complex numbers, is a zero of p if p evaluates to zero when, for each $1 \le i \le n$, a_i is uniformly substituted for x_i, i.e. $p(a_1,a_2,\ldots,a_n) = 0$. Equivalently, $(a_1,a_2,\ldots,a_n)$ is called a solution of the multivariate polynomial equation $p(x_1,\ldots,x_n) = 0$ if $p(a_1,\ldots,a_n) = 0$.

Given a system $\{f_1(x_1,x_2,\ldots,x_n),\ f_2(x_1,x_2,\ldots,x_n),\ \ldots,\ f_r(x_1,x_2,\ldots,x_n)\}$ of polynomials (equivalently a system $\{f_i(x_1,x_2,\ldots,x_n) = 0,\ i = 1,\ldots,r\}$ of polynomial equations), $(a_1,a_2,\ldots,a_n)$ is a common zero (respectively, a common solution) of the system, if, for each $1 \le i \le r$, $f_i(a_1,a_2,\ldots,\ldots a_n) = 0$, i.e. $(a_1,a_2,\ldots,a_n)$ is a zero of every polynomial in the system. The n-tuple $(a_1,a_2,\ldots,a_n)$ is also called a common root of these polynomials. Henceforth, we will abuse the terminology; by a system, we will mean either a set of polynomials or a set of polynomial equations, with the hope that the context can resolve the ambiguity.

Given a system $\mathcal{S}$ of polynomials, the set

$$\text{Zero}(\mathcal{S}) = \{(a_1,a_2,\ldots,a_n) \in \mathcal{C}^n \mid \forall f \in \mathcal{S}, f(a_1,a_2,\ldots,a_n) = 0\}$$

is called the *zero set* of the system $\mathcal{S}$. The zero set defined by a system of polynomials

is also called an *algebraic* set. Throughout this paper, we have focused on zeros in the field of complex numbers (in general, an algebraically closed field). For details about zero sets of polynomials over the reals (these are called semi-algebraic sets), the reader may consult Tarski (1948), Arnon *et al.* (1984), Coste and Roy (1988) and Renegar (1989).

Example: Let $f(x,y) = x^3 - y^2$, $g(x,y) = 2x^2 + (y-1)^2 - 2$, $\mathcal{S} = \{f = 0, g = 0\}$,

$$\begin{aligned}\text{Zero}(\mathcal{S}) = \{&(1,1),\\ &(-2.253012931 + 1.322062452i, 3.016885573 + 2.953686458i),\\ &(-2.253012931 - 1.322062452i, 3.016885573 - 2.953686458i),\\ &(-.4604205396 + .3623712296i, -.3774386961 - .2422512995i),\\ &(-.4604205396 - .3623712296i, -.3774386961 + .2422512995i),\\ &(.4268669419, -.2788937516)\}\end{aligned}$$

(the values shown here are floating point approximations). Geometrically, Zero($\mathcal{S}$) consists of the coordinates of all the points of intersection of the cusp $f(x,y) = 0$ and the ellipse $g(x,y) = 0$.

The zero set of a system of polynomials could be infinite. In that case, it is possible, as in linear algebra, to talk about the dimension of the zero set. If the zero set is finite, the system is called *zero-dimensional*. If the zero set is empty, then the system is said to be of dimension -1. If the zero set is infinite, then the system has positive dimension.

A zero a of a univariate polynomial $p(x)$ is said to be of multiplicity $k+1$ if $p^{(i)}(a)$ for $i = 0,\ldots,k$ all evaluate to zero, where $p^{(i)}$ is the i-th derivative of p with respect to x. This is equivalent to saying that $(x-a)^{k+1}$ divides $p(x)$.

1.2.1. Affine Zeros and Projective Zeros

The familiar complex n-space is known as the affine n-space over the complex numbers. It consists of all n-tuples of the complex numbers. In cartesian coordinates, each n-tuple $(c_1,c_2,\ldots,c_n) \in \mathcal{C}^n$ represents a point in the affine n-space. By Zero($\mathcal{S}$), we mean the set of affine zeros of $\mathcal{S}$. We now introduce the notion of projective space and projective zeros.

The projective n-space over the complex numbers, denoted by $\mathcal{P}^n$, consists of all $(n+1)$-tuples over the complex numbers except $(0,0,\ldots,0)$. Each $(n+1)$-tuple in $\mathcal{P}^n$ is said to represent a point in the projective n-space. However, the tuples $(a_0,a_1,\ldots,a_n)$ and $(\lambda a_0,\lambda a_1,\ldots,\lambda a_n)$ are said to represent the same point for any non-zero complex number λ and the two $(n+1)$-tuples are said to be equivalent. Thus, each point in $\mathcal{P}^n$ is represented by any member of an infinite set of proportionate (equivalent) n-tuples. We can embed the affine n-space into the projective n-space as follows. To each point $(c_1,c_2,\ldots,c_n) \in \mathcal{C}^n$, we associate the $(n+1)$-tuple $(1,c_1,c_2,\ldots,c_n)$. As mentioned earlier, any other $(n+1)$-tuple of the form $(\lambda,\lambda c_1,\ldots,\lambda c_n)$ represents the same point as long as $\lambda \neq 0$. Any $(n+1)$-tuple $B = (b_0,b_1,b_2,\ldots,b_n)$ in which $b_0 \neq 0$ is associated with the unique point in affine n-space whose coordinates are

$$\bar{B} = (b_1/b_0, b_2/b_0, \ldots, b_n/b_0).$$

Note that any other $(n+1)$-tuple equivalent to B gives rise to the same point $\bar{B}$ in the affine n-space. Those $(n+1)$-tuples which have $b_0 = 0$ are said to represent *points at infinity*. The set of all points at infinity in $\mathcal{P}^n$ is known as the *hyper plane at infinity*. To summarize what we have said so far, projective n-space consists of the affine n-space together with the hyper-plane at infinity.

A polynomial $f(x_0, x_1, \ldots, x_n)$ is said to be homogeneous of degree d if each *term* in f has degree d.

Example: $f(x_0, x_1, x_2) = x_0^2 x_1 - 2x_1 x_2^2 + x_1^3$ is a homogeneous polynomial of degree 3. $g(x_1, x_2) = x_1^3 + x_2 - x_1 x_2$ is *not* homogeneous.

If $(a_0, a_1, \ldots, a_n)$ is a zero of the homogeneous polynomial $f(x_0, x_1, \ldots, x_n)$, i.e.

$$f(a_0, a_1, \ldots, a_n) = 0,$$

then any other $(n+1)$-tuple $(\lambda a_0, \lambda a_1, \ldots, \lambda a_n)$ is also a zero of $f(x_0, x_1, \ldots, x_n)$.

Given a non-homogeneous polynomial $f(x_1, x_2, \ldots, x_n)$, of degree d, it can be *homogenized* as follows. Consider

$${}^h f(x_0, x_1, x_2, \ldots, x_n) = x_0^d f(x_1/x_0, x_2/x_0, \ldots x_n/x_0)$$

where x_0 is a new variable. ${}^h f(x_0, x_1, \ldots, x_n)$ is a homogeneous polynomial of degree d such that

$${}^h f(1, x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots x_n).$$

Let $(a_1, a_2, \ldots, a_n) \in \mathcal{C}^n$ be a zero of f, i.e. $f(a_1, a_2, \ldots a_n) = 0$. Then $(1, a_1, a_2, \ldots, a_n) \in \mathcal{P}^n$ (or any $(n+1)$-tuple equivalent to it) is a zero of ${}^h f$. Conversely, if $(a_0, a_1, a_2, \ldots, a_n) \in \mathcal{P}^n$ is a zero of ${}^h f$, and $a_0 \neq 0$, then $(a_1/a_0, a_2/a_0, \ldots, a_n/a_0) \in \mathcal{C}^n$ is a zero of f. If $a_0 = 0$, then there is no corresponding point in the affine space that is a zero of f. Such zeros of ${}^h f$ are called *zeros at infinity* of f.

Most of the resultant calculations work over projective space with homogeneous polynomials.

1.2.2. Ideals

Consider a commutative ring $\mathcal{R}$. Let $\mathcal{A} \subseteq \mathcal{R}$. $\mathcal{A}$ is called an *ideal* in $\mathcal{R}$ iff:

- for all $f, g \in \mathcal{A}$, $f + g \in \mathcal{A}$, and,
- for all $f \in \mathcal{A}$, $gf \in \mathcal{A}$ for any $g \in \mathcal{R}$.

Let $f_1, f_2, \ldots, f_r \in \mathcal{R}$. Consider an ideal $\mathcal{J}$ that contains all of $f_1, \ldots, f_r$. By the above definition, the element

$$f = g_1 f_1 + g_2 f_2 + \ldots + g_r f_r \in \mathcal{J}$$

for *any* $g_1, g_2, \ldots, g_r \in \mathcal{R}$. Indeed, the set

$$\mathcal{I} = \{\sum_{i=1}^{r} g_i f_i \mid g_i \in \mathcal{R}\}$$

is an ideal in $\mathcal{R}$ and it is the *smallest* ideal in $\mathcal{R}$ containing the set $\{f_1, f_2, \ldots, f_r\}$. $\mathcal{I}$ is called the *ideal generated by* $f_1, f_2, \ldots, f_r$ and denoted by $(f_1, f_2, \ldots, f_r)$. The set $\{f_1, f_2, \ldots, f_r\}$ is called a *generating set* or a *basis* for the ideal $\mathcal{I}$.

Examples:

1. $\mathcal{R}$ = ring of integers, $\mathcal{A}$ = set of all multiples of 3.
2. $\mathcal{R}$ = ring of integers, and $\mathcal{A}$ = the set of integers of the form $9a + 30b$, where a, b are integers.
3. $\mathcal{R} = \mathcal{Q}[x, y]$, and $\mathcal{A}$ is the set of all polynomials $f(x, y) \in \mathcal{Q}[x, y]$ such that $f(a, b) = 0$ for some fixed constants $a, b \in \mathcal{Q}$.
4. $\mathcal{R} = \mathcal{Q}[x_1, x_2, \ldots, x_n]$ and $\mathcal{A} = (x_1 - a_1, x_2 - a_2, \ldots, x_n - a_n)$ where $a_1, a_2, \ldots, a_n \in \mathcal{Q}$.

An ideal can have many bases. For example, the ideal in the first example has $\{3\}$ as its basis whereas the ideal in the second example has $\{9, 30\}$ as it basis, but the two ideals are the same.

Let

$$\mathcal{I} = (f_1, f_2, \ldots, f_r) \subseteq \mathcal{Q}[x_1, x_2, \ldots, x_n].$$

Let $(a_1, a_2, \ldots, a_n) \in \mathcal{C}^n$ be a *common zero* of $f_1, f_2, \ldots, f_r$, i.e.

$$f_i(a_1, a_2, \ldots, a_n) = 0, \quad i = 1, \ldots, r.$$

Since, for any $f \in \mathcal{I}$, there exist $g_i \in \mathcal{Q}[x_1, x_2, \ldots, x_n]$ such that $f = \sum_1^r g_i f_i$, it follows that $f(a_1, a_2, \ldots, a_n) = 0$, i.e. $(a_1, a_2, \ldots, a_n)$ is a *zero of every polynomial* in the ideal. The set

$$\text{Zero}(\mathcal{I}) = \{(a_1, a_2, \ldots, a_n) \in \mathcal{C}^n \mid \forall f \in \mathcal{I}, f(a_1, a_2, \ldots, a_n) = 0\}$$

is called the *zero set* of the ideal $\mathcal{I}$.

Earlier we considered a polynomial equation defining a cusp, $f(x, y) = x^3 - y^2$, and another polynomial equation defining an ellipse, $g(x, y) = 2x^2 + (y-1)^2 - 2$. Let $\mathcal{I} = (f, g)$. The location of the points of intersection of the two curves is not evident from the equations $f(x, y) = 0, g(x, y) = 0$. It will be shown later that the set

$$\begin{aligned} G \;=\; & \{g_1(x) = x^6 + 4x^5 + 4x^4 - 6x^3 - 4x^2 + 1, \\ & g_2(x, y) = y + 1/2(-x^3 - 2x^2 + 1)\} \end{aligned}$$

is another basis for the ideal $\mathcal{I}$. Notice that G has one polynomial that depends only on x, namely $g_1(x)$, and one that depends on both x and y, i.e. $g_2(x, y)$. The roots of $g_1(x)$ are the x-coordinates of the points of intersection of the cusp $f(x, y) = 0$ and the ellipse $g(x, y) = 0$. For each root α of $g_1(x)$, the y-coordinates of the corresponding intersection points are found by computing the roots of $g_2(\alpha, y)$. As in linear algebra, we say that the above two polynomials in G are in *triangular form*.

2. Resultants

Given two polynomials $f(x), g(x) \in \mathcal{Q}[x]$ of degrees m and n respectively, i.e.

$$\begin{aligned} f(x) &= f_n x^n + f_{n-1} x^{n-1} + \ldots f_1 x + f_0, \quad \text{and,} \\ g(x) &= g_m x^m + g_{m-1} x^{m-1} + \ldots + g_1 x + g_0, \end{aligned}$$

when do f and g have common roots? The question leads naturally to a condition that has to be satisfied by the coefficients of f and g. This condition was discovered by Euler and is now commonly referred to as the *vanishing of the Sylvester resultant of f and g*. The Sylvester resultant of f, g is the determinant of the following matrix:

$$R = \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 & g_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 & g_1 & g_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 & g_2 & g_1 & g_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_n & f_{n-1} & f_{n-2} & \dots & f_{n-m+1} & g_n & g_{n-1} & g_{n-2} & \dots & g_0 \\ 0 & f_n & f_{n-1} & \dots & f_{n-m+2} & g_{n+1} & g_n & g_{n-1} & \dots & g_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & f_n & 0 & 0 & 0 & \dots & g_m \end{pmatrix}$$

Assuming that at least one of f_n, g_m is non-zero, *the vanishing of the Sylvester resultant is a necessary and sufficient condition for f and g to have common roots.*

Resultants are most commonly used for computing projections and for successive elimination of variables. The Sylvester resultant has been studied extensively in the past. We refer the interested reader to the beautiful subresultant theory developed simultaneously and independently by G.E. Collins and W.S. Brown. For an exposition of the theory, see Knuth (1980, pp. 407–408), Loos (1983), Collins (1967, 1971) and Brown and Traub (1971). Efficient implementations of algorithms for computing resultants are available in most computer algebra systems including REDUCE, MACSYMA, MAPLE and MATHEMATICA.

2.1. Dixon's Formulation

In 1779, Bezout had already developed a method for computing the resultant of two univariate polynomials. We describe Cayley's reformulation of Bezout's method. It is simple to explain and extends naturally to the bivariate case as shown by Dixon. Cayley proposed viewing the resultant of $f(x)$ and $g(x)$ as follows. Replace x by α in both $f(x)$ and $g(x)$ and we get polynomials $f(\alpha)$ and $g(\alpha)$. The determinant $\Delta(x, \alpha)$ of the matrix

$$\begin{vmatrix} f(x) & g(x) \\ f(\alpha) & g(\alpha) \end{vmatrix}$$

is a polynomial in x and α and it obviously is equal to zero if $x = \alpha$. This implies that the determinant has $(x - \alpha)$ as a factor. The polynomial

$$\delta(x, \alpha) = \frac{\Delta(x, \alpha)}{(x - \alpha)}$$

is an $n - 1$ degree polynomial in α and is symmetric in x and α. It vanishes at every common zero x_0 of $f(x)$ and $g(x)$ no matter what values α has. So, at $x = x_0$, the coefficient of every power product of α in $\delta(x, \alpha)$ is 0. This gives n equations which are polynomials in x, and the maximum degree of these polynomials is $n - 1$. Any common zero of $f(x)$ and $g(x)$ is a solution of these polynomial equations, and these polynomial equations have a common solution if the determinant of their coefficients is

0. Unlike in Sylvester's formulation, where the resultant of f and g is the determinant of an $(m+n) \times (m+n)$ matrix, in the Cayley-Dixon formulation, the resultant is obtained as the determinant of a $n \times n$ matrix.

For example, consider a generic cubic polynomial:

$$p = ax^3 + bx^2 + cx + d.$$

The discriminant of p is the resultant of p and dp/dx. The polynomial p has multiple roots if and only if its discriminant is zero. Let us compute the discriminant of p by Cayley's method. We have

$$dp/dx = 3ax^2 + 2bx + c.$$

The determinant of the matrix:

$$\begin{vmatrix} ax^3 + bx^2 + cx + d & 3ax^2 + 2bx + c \\ a\alpha^3 + b\alpha^2 + c\alpha + d & 3a\alpha^2 + 2b\alpha + c \end{vmatrix}$$

when divided by $x - \alpha$ gives the polynomial:

$$(3a^2x^2+2abx+ac)\alpha^2+(2abx^2+(2b^2-2ac)x+(bc-3ad))\alpha+(acx^2+(bc-3ad)x+(c^2-2bd).$$

We get three equations by equating the coefficients of the power products of α above to 0:

$$\begin{array}{rrrcl} 3a^2 & x^2 + 2ab & x + ac & = & 0, \\ 2ab & x^2 + (2b^2 - 2ac) & x + (bc - 3ad) & = & 0, \\ ac & x^2 + (bc - 3ad) & x + (c^2 - 2bd) & = & 0. \end{array}$$

Treating x^0, x^1, x^2 as unknowns, we have three homogeneous equations in three unknowns; they have a common solution if and only if the determinant of the coefficient matrix is 0, i.e.

$$\begin{vmatrix} 3a^2 & 2ab & ac \\ 2ab & (2b^2 - 2ac) & (bc - 3ad) \\ ac & (bc - 3ad) & (c^2 - 2bd) \end{vmatrix} = -a^2(-c^2b^2 + 4ac^3 + 4b^3d - 18abdc + 27a^2d^2) = 0.$$

The reader may want to compare this determinant with the Sylvester resultant given by the determinant

$$\begin{vmatrix} a & b & c & d & 0 \\ 0 & a & b & c & d \\ 3a & 2b & c & 0 & 0 \\ 0 & 3a & 2b & c & 0 \\ 0 & 0 & 3a & 2b & c \end{vmatrix} = -a(-c^2b^2 + 4ac^3 + 4b^3d - 18abdc + 27a^2d^2).$$

The Dixon resultant has an extraneous factor of a as compared to the Sylvester resultant. This factor arises because Cayley's formulation assumes that both the polynomials are of the same degree. In general, the Bezout resultant computed using the Cayley-Dixon formulation will have an extraneous factor $I_f^{(degree(f)-degree(g))}$, where I_f is the initial of $f(x)$.

Dixon (1908) showed how to extend this formulation to three polynomials in two variables. Consider the following three generic bi-degree polynomials which have all the

power products of the type $x^i y^j$, where $0 \leq i \leq m, 0 \leq j \leq n$, i.e.

$$f(x,y) = \sum_{i,j} a_{ij} x^i y^j, \quad g(x,y) = \sum_{i,j} b_{ij} x^i y^j, \quad h(x,y) = \sum_{i,j} c_{ij} x^i y^j.$$

Just as in the earlier case, Dixon observed that the determinant

$$\Delta(x,y,\alpha,\beta) = \begin{vmatrix} f(x,y) & g(x,y) & h(x,y) \\ f(\alpha,y) & g(\alpha,y) & h(\alpha,y) \\ f(\alpha,\beta) & g(\alpha,\beta) & h(\alpha,\beta) \end{vmatrix}$$

vanishes when α or β are substituted for x or y, respectively, implying that $(x-\alpha)(y-\beta)$ is a factor of the above determinant. The expression

$$\delta(x,y,\alpha,\beta) = \frac{\Delta(x,y,\alpha,\beta)}{(x-\alpha)(y-\beta)}$$

is a polynomial of degree $2m-1$ in α, $n-1$ in β, $m-1$ in x and $2n-1$ in y. Since the above determinant vanishes when we substitute $x = x_0, y = y_0$ where (x_0, y_0) is a common zero of $f(x,y), g(x,y), h(x,y)$, into the above matrix, $\delta(x_0, y_0, \alpha, \beta)$ must vanish no matter what α and β are. The coefficients of each power product $\alpha^i \beta^j, 0 \leq i \leq 2m-1, 0 \leq j \leq n-1$, have common zeros which include the common zeros of $f(x,y), g(x,y), h(x,y)$. This gives $2mn$ equations in power products of x, y, and the number of power products $x^i y^j, 0 \leq i \leq m-1, 0 \leq j \leq 2n-1$ is also $2mn$. The determinant of the coefficient matrix from these equations is a multiple of the resultant. Using a simple geometric argument, Dixon proved that in this case, the determinant is in fact the resultant up to a constant factor. For three arbitrary polynomials, Dixon developed some special methods which selected some coefficients of $\alpha^i \beta^j$ from δ, and used dialytic expansion of $f(x,y), g(x,y), h(x,y)$ to come up with a system of k linearly independent polynomial equations expressed using k power products in x, y.

As an example, consider the following two bi-quadratics and a linear form.

$$\begin{aligned} f_1 &= a_1 x_1^2 x_2^2 + a_2 x_1^2; \\ f_2 &= b_1 x_1^2 x_2^2 + b_2 x_2^2; \\ f_3 &= u_1 x_1 + u_2 x_2 + u_3 \end{aligned}$$

The polynomial $\delta(x_1, x_2, \alpha, \beta)$ can be given as:

$$\begin{pmatrix} 1 & \alpha & \beta & \alpha^2 & \alpha^2\beta & \alpha\beta & \alpha^3 & \alpha^3\beta \end{pmatrix} \quad \mathbf{D} \quad \begin{pmatrix} 1 & x_2^3 & x_1 & x_2 & x_2^2 & x_1x_2^3 & x_1x_2^2 & x_1x_2 \end{pmatrix}^{\mathbf{Tr}}$$

where $\mathbf{D}$ is the 8 by 8 matrix :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & a_1u_3b_2 & 0 & a_2u_3b_2 \\ 0 & a_1u_3b_2 & 0 & a_2u_3b_2 & 0 & a_1u_1b_2 & 0 & a_2u_1b_2 \\ 0 & 0 & a_2u_3b_2 & 0 & 0 & a_1u_2b_2 & a_1u_3b_2 & a_2u_2b_2 \\ 0 & a_1u_1b_2 & 0 & 0 & 0 & 0 & a_2u_2b_1 & a_2u_3b_1 \\ 0 & 0 & a_2u_3b_1 & 0 & a_1u_1b_2 & 0 & 0 & a_2u_2b_1 \\ a_2u_3b_2 & a_1u_2b_2 & a_2u_1b_2 & a_2u_2b_2 & a_1u_3b_2 & 0 & a_1u_1b_2 & 0 \\ 0 & 0 & 0 & a_2u_3b_1 & a_2u_2b_1 & 0 & 0 & a_2u_1b_1 \\ a_2u_3b_1 & 0 & a_2u_1b_1 & a_2u_2b_1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The determinant of $\mathbf{D}$,

$$a_1^2 a_2^4 b_1^2 b_2^4 u_3^4 (a_1^2 b_1^2 u_3^4 + 2a_1^2 b_1 b_2 u_1^2 u_3^2 + a_1^2 b_2^2 u_1^4 + 2a_1 a_2 b_1^2 u_2^2 u_3^2 - 2a_1 a_2 b_1 b_2 u_1^2 u_2^2 + a_2^2 b_1^2 u_2^4),$$

is the resultant up to a constant factor.

Chionh's thesis (Chionh, 1990) gives a good summary of Dixon's approach to resultants. He also discusses how Dixon's resultants for the two variable case can be used for implicitization and finding base points.

2.2. Multivariate Resultants

Macaulay constructed a resultant (henceforth referred to as the Macaulay resultant) for n homogeneous polynomials in n variables (Macaulay, 1916). It simultaneously generalizes the Sylvester resultant and the determinant of a system of linear equations (in the sense that the Macaulay resultant for two homogeneous polynomials in two variables is the same as their Sylvester resultant, and the Macaulay resultant for a system of n homogeneous linear equations in n variables is the same as the determinant of the system). Macaulay's resultant disappeared from the literature for several decades until it was used in a slightly different form by Lazard (1981) for equation-solving. More recently, Canny (1988) resurrected the Macaulay resultant and used it in his roadmap algorithm for the robot motion-planning problem.

The Macaulay resultant can be used to eliminate several variables at once from a system of polynomial equations. Macaulay used his resultant construction to actually determine the solutions of a system of homogeneous polynomial equations. If one wishes to solve non-homogeneous polynomial equations using the Macaulay resultant, one has to homogenize the polynomials first. Methods based on Macaulay's matrix give out zeros in $\mathcal{P}^n$ for the homogenized system of equations and they can include zeros at infinity, which will have to be dealt with if one is interested only in affine common zeros.

2.3. Macaulay's Matrices

In this section, we describe Macaulay's construction. The key idea is to show which power products are sufficient in the dialytic method to be used as multipliers for the polynomials, so that we get a square system of l linear equations in l power products which can be considered as the unknowns.

Let $f_1, f_2, \ldots, f_n$ be n *homogeneous* polynomials in $x_1, x_2, \ldots, x_n$. Let $d_i = \deg(f_i)$ and

$$d_M = 1 + \sum_1^n (d_i - 1).$$

Let T denote the set of all terms of degree d_M in the n variables $x_1, x_2, \ldots, x_n$, i.e.

$$T = \{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} | \alpha_1 + \alpha_2 + \ldots + \alpha_n = d_M\}$$

and

$$|T| = \binom{d_M + n - 1}{n - 1}.$$

The polynomials are multiplied with appropriate power products to generate $|T|$ equations in $|T|$ unknowns which are power products of degree d_M. The order in which the polynomials are considered for selecting multipliers results in different systems of linear equations.

Example: We will use a generic system of one linear and two quadratic polynomials in three variables to illustrate various aspects of Macaulay's construction. Let

$$\begin{aligned} f_1 &= a_{1,1}x_1^2 + a_{1,2}x_1x_2 + a_{1,3}x_1x_3 + a_{2,2}x_2^2 + a_{2,3}x_2x_3 + a_{3,3}x_3^2,\ d_1 = 2, \\ f_2 &= b_{1,1}x_1^2 + b_{1,2}x_1x_2 + b_{1,3}x_1x_3 + b_{2,2}x_2^2 + b_{2,3}x_2x_3 + b_{3,3}x_3^2,\ d_2 = 2, \\ f_3 &= c_1x_1 + c_2x_2 + c_3x_3,\ d_3 = 1 \text{ and } d_M = 3. \end{aligned}$$

The number of terms in three variables of degree 3 is 10. To determine the power products to be used as multipliers to obtain a 10 by 10 system, we will use the ordering (f_1, f_2, f_3) to illustrate the construction. The first three rows are obtained by multiplying f_1 by the power products of degree 1, the difference of d_M and the degree of f_1; these power products are: x_1, x_2, x_3. The next three rows are obtained also by multiplying f_2 by the power products of degree 1 that are not multiples of x_1^2. In this case, they are x_1, x_2, x_3. The last four rows are obtained by multiplying f_3 by the power products of degree 2, the difference of d_M and the degree of f_3, that are not multiples of either x_1^2 or x_2^2. These power products are: $x_1x_2, x_1x_3, x_2x_3, x_3^2$.

Macaulay's matrix in this case is:

	x_1^3	$x_1^2x_2$	$x_1^2x_3$	$x_1x_2^2$	$x_1x_2x_3$	$x_1x_3^2$	x_2^3	$x_2^2x_3$	$x_2x_3^2$	x_3^3
x_1	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{2,2}$	$a_{2,3}$	$a_{3,3}$	0	0	0	0
x_2	0	$a_{1,1}$	0	$a_{1,2}$	$a_{1,3}$	0	$a_{2,2}$	$a_{2,3}$	$a_{3,3}$	0
x_3	0	0	$a_{1,1}$	0	$a_{1,2}$	$a_{1,3}$	0	$a_{2,2}$	$a_{2,3}$	$a_{3,3}$
x_1	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{2,2}$	$b_{2,3}$	$b_{3,3}$	0	0	0	0
x_2	0	$b_{1,1}$	0	$b_{1,2}$	$b_{1,3}$	0	$b_{2,2}$	$b_{2,3}$	$b_{3,3}$	0
x_3	0	0	$b_{1,1}$	0	$b_{1,2}$	$b_{1,3}$	0	$b_{2,2}$	$b_{2,3}$	$b_{3,3}$
x_1x_2	0	c_1	0	c_2	c_3	0	0	0	0	0
x_1x_3	0	0	c_1	0	c_2	c_3	0	0	0	0
x_2x_3	0	0	0	0	c_1	0	0	c_2	c_3	0
x_3^2	0	0	0	0	0	c_1	0	0	c_2	c_3

The general construction is given below. Let

$$\begin{aligned} T^{(0)} &= \{\text{terms of degree } d_M - d_1\}, \\ T^{(1)} &= \{\text{terms of degree } d_M - d_2 \text{ and not divisible by } x_1^{d_1}\}, \\ T^{(2)} &= \{\text{terms of degree } d_M - d_2 \text{ and not divisible by } x_1^{d_1}, \text{ or by } x_2^{d_2}\}, \\ &\vdots \\ T^{(n-1)} &= \{\text{terms of degree } d_M - d_n \text{ and not divisible by} \\ &\qquad x_1^{d_1} \text{ or } x_2^{d_2} \text{ or } \ldots \text{ or } x_{n-1}^{d_{n-1}}\}. \end{aligned}$$

Macaulay refers to $T^{(i)}$ as the *reduced* set of terms with respect to $x_1, \ldots, x_i$ (so a term is reduced with respect to $x_1, \ldots, x_i$ if it is not divisible by any of $x_1^{d_1}$, $x_2^{d_2}$, ..., $x_i^{d_i}$). Now, construct a matrix A with $|T|$ columns and $\sum_i |T^{(i)}|$ rows. The columns of A are labeled by the terms in T in some order. The first $|T^{(0)}|$ rows are labeled by the terms in $T^{(0)}$, the next $|T^{(1)}|$ rows are labeled by the terms in $T^{(1)}$ and so on. In the row labeled by the term $t \in T^{(i)}$, arrange the coefficients of tf_{i+1}, with the coefficient of a term t' in tf_{i+1} appearing under the column labeled by t' (note that tf_{i+1} has degree d_M).

We reproduce below Macaulay's argument showing that the matrix A thus constructed is square. Let a_i denote the coefficient of $x_i^{d_i}$ in f_i for $i = 1, 2, \ldots, n$ (note that this coefficient could possibly be 0). Since each row contains the coefficients of some f_i alone (shifted appropriately), every row contains exactly one a_i.

Every column contains at least one a_i. Suppose not. Let the label of a column not containing a_i be t. So, there is no term t' and no i such that $t'x_i^{d_i} = t$ which implies that t is not divisible by $x_i^{d_i}$ for any i. This implies that $\deg(t) \leq \sum_i (d_i - 1) < d_M$, which is a contradiction since $\deg(t) = d_M$.

Each column contains at most one a_i. The proof is again by contradiction. Suppose a column labeled by t has a_i, a_j with $i < j$; this means that there are terms $t_1 \in T^{(i-1)}$ and $t_2 \in T^{(j-1)}$ such that $t_1x_i^{d_i} = t_2x_j^{d_j} = t$. This implies that $x_i^{d_i}$ divides t_2; but $t_2 \in T^{(j-1)}$, which means that it is not divisible by $x_1^{d_1}$ or $x_2^{d_2}$, so on up to $x_{j-1}^{d_{j-1}}$; in particular, t_2 is *not divisible* by $x_i^{d_i}$ since $i < j$, which is a contradiction.

We have thus shown that each row and each column has exactly one a_i and this establishes a one-to-one correspondence between rows and columns, which implies that

$$|T| = \sum_i |T^{(i)}|$$

and A is a square matrix.

Let $\det(A)$ denote the determinant of A, which is a polynomial in the coefficients of the f_i's. It is homogeneous in the coefficients of each f_i and the degree of $\det(A)$ in the coefficients of f_n is $d_1d_2 \ldots d_{n-1}$ (this is the number of rows of A labeled by the terms in $T^{(n)}$; these rows contain the coefficients of f_n).

The construction of A depends on how the polynomials are ordered. A different order produces a different matrix. The ideal generated by the determinants of all such matrices is the so-called *ideal of inertial forms*, a concept that goes back to Hurwitz (van der Waerden, 1950) and this ideal is known to be principle. Let $\det(A_\sigma)$ denote the determinant of a matrix constructed using a permutation σ of the polynomials $f_1, f_2, \ldots, f_n$. The greatest common divisor of the $\det(A_\sigma)$'s ($\sigma \in S_n$, the symmetric group on n letters) regarded as polynomials in the indeterminate coefficients of the f_i is defined to be the resultant (denoted R) of the system $\{f_1, f_2, \ldots, f_n\}$.

For the above example, the resultant is the greatest common divisor of the determinants of all such matrices ($3! = 6$ of them in this case).

We list some important properties of the resultant below.

1. $R = 0$ if and only if the f_i's have a non-trivial common zero.
2. R is absolutely irreducible and invariant under linear coordinate transformations. The

vanishing of R is thus a necessary condition for the system to have a common zero and it is the smallest such condition.

3. R is homogeneous in the coefficients of each f_i and has degree $(\prod_{j=1}^{n} d_j)/d_i$ in coefficients of f_i. For instance, in the above example, the resultant is a polynomial in which each term has degree 2 in $a_{j,k}$, degree 2 in $b_{j,k}$ and degree 4 in c_j.
4. If $f_n = gh$, then the resultant of $f_1, f_2, \ldots, f_n$ is a product of two resultants R_1 (of $f_1, f_2, \ldots, g$) and R_2 (of $f_1, f_2, \ldots, h$).

Macaulay also constructed the following formula relating R and $\det(A_\sigma)$:

$$R \det(B_\sigma) = \det(A_\sigma)$$

where $\det(B_\sigma)$ is the determinant of a submatrix of A_σ. The submatrix B_σ is obtained by deleting all columns labeled by terms *reduced* (in Macaulay's sense) in *any* $n-1$ *of the variables*, and, those rows which contain one of the a_is in the deleted columns.

Example: In the earlier example, the submatrix B for the Macaulay matrix is

$$\begin{array}{c} \\ x_3 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1^2 x_3 & x_2^2 x_3 \end{array} \\ \begin{pmatrix} a_{1,1} & a_{2,2} \\ b_{1,1} & b_{2,2} \end{pmatrix} \end{array}.$$

The matrix B was obtained by deleting columns that are *reduced* (in Macaulay's sense) in any two variables (e.g. x_1^3 is not divisible by x_2^2 or by x_3, so it is *reduced* in x_2, x_3; hence, the column labeled x_1^3 was deleted, a similar reason for deleting the other columns). The surviving columns are reduced in fewer than $n-1$ variables; for example, $x_1^2 x_3$ is divisible by x_1^2, x_3 but not by x_2^2. Hence, it is reduced in x_2 only. The rows that contained an a_i (in this example, this means one of $a_{1,1}, b_{2,2}, c_3$) in the deleted columns are also deleted. For example, the first row was deleted because it contained $a_{1,1}$ in a deleted column, namely, the column labeled x_1^3.

The crucial point here is that Macaulay's formula works in "general", or when the coefficients are taken to be indeterminates. If one wants to compute R for a specific system of polynomials using this formula, one may encounter the problem of having $\det(B_\sigma) = 0$. A similar problem can arise if we try to compute R as the gcd of $\det(A_\sigma)$'s due to the vanishing of some or all of the $\det(B_\sigma)$'s. The computation of the resultant as a *generic polynomial* in the indeterminate coefficients of f_i is infeasible, as it is very large, even for low degree input polynomials.

2.4. The U-Resultant

Suppose we have a system of n polynomials in n variables, i.e.

$$f_1(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n),$$

and we wish to find their common zeros. Let d_i denote the total degree of f_i. Introduce a new homogenizing variable x_0 and let R_u denote the Macaulay resultant of the $(n+1)$ homogeneous polynomials ${}^h f_1, {}^h f_2, \ldots, {}^h f_n, f_u$ in $(n+1)$ variables $x_0, x_1, \ldots, x_n$ where f_u is the linear form

$$f_u = x_0 u_0 + x_1 u_1 + \ldots + x_n u_n$$

and $u_0, u_1, \ldots, u_n$ are *new unknowns*. R_u is a polynomial in $u_0, u_1, \ldots, u_n$. It is homogeneous in the u_i of degree $B = \prod_{i=1}^{n} d_i$ (these observations follow from the properties of the Macaulay resultant listed earlier). R_u is known as the u-resultant of the given system of polynomials. It can be shown that R_u factors into *linear factors* over the complex numbers, i.e.

$$R_u = \prod_{j=1}^{B} (u_0 \alpha_{0,j} + u_1 \alpha_{1,j} + \ldots + u_n \alpha_{n,j})$$

and if $(u_0 \alpha_{0,j} + u_1 \alpha_{1,j} + \ldots + u_n \alpha_{n,j})$ is a factor of R_u, then $(\alpha_{0,j}, \alpha_{1,j}, \ldots, \alpha_{n,j})$ is a common zero of of ${}^h f_1, {}^h f_2, \ldots, {}^h f_n$. The converse can also be proved, i.e. if $(\beta_{0,j}, \beta_{1,j}, \ldots, \beta_{n,j})$ is a common zero of ${}^h f_1, {}^h f_2, \ldots, {}^h f_n$ then $(u_0 \beta_{0,j} + u_1 \beta_{1,j} + \ldots + u_n \beta_{n,j})$ divides R_u. This gives an algorithm for finding all the common zeros of ${}^h f_1, {}^h f_2, \ldots, {}^h f_n$.

Example: Consider the unit circle $x_1^2 + x_2^2 - 1 = 0$ and the pair of straight lines $(x_1 - x_2 - 1)\ (x_1 - x_2 + 1)$. To find their intersection, we compute their u-resultant which is the Macaulay resultant of

$$\begin{aligned} f_1 &= x_1^2 + x_2^2 - x_0^2 \\ f_2 &= x_1^2 - 2x_1 x_2 + x_2^2 - x_0^2 \\ f_u &= u_0 x_0 + u_1 x_1 + u_2 x_2. \end{aligned}$$

The u-resultant is computed to be the polynomial

$$u_1^2 u_2^2 - u_1^2 u_0^2 - u_2^2 u_0^2 + u_0^4.$$

which factors as

$$(0.u_1 - 1.u_2 + 1.u_0)(0.u_1 + 1.u_2 + 1.u_0)(-1.u_1 + 0.u_2 + 1.u_0)(1.u_1 + 0.u_2 + 1.u_0).$$

We can read off the four intersection points from the linear factors as

$$(0, -1), (0, 1), (-1, 0), (1, 0).$$

Constructing the full u-resultant, however, is an almost impossible task (it is a polynomial of degree $\prod_{i=1}^{n} d_i$ in n variables). So, if one is interested in computing the common zeros of a set of polynomials, one does so by computing specializations of R_u. For example, the univariate polynomial $R_1(u_0)$ obtained by substituting $u_i = 0$ for $i = 2, \ldots, n$ and $u_1 = -1$ has as its roots the x_1 coordinates of the common zeros. $R_1(u_0)$ can be computed from the Macaulay matrices by evaluation of determinants with rational (or complex) entries and interpolation and without constructing the full u-resultant. For more details, see Canny (1988), Lakshman (1990a), Lakshman and Lazard (1991) and Manocha and Canny (1991).

This method does not always work, however. Since for each common zero $(\beta_{1,j}, \ldots, \beta_{n,j})$ of the f_i's, the linear form $(u_0 + u_1 \beta_{1,j} + \ldots + u_n \beta_{n,j})$ divides the u-resultant R_u, if the given system of polynomials $f_1, f_2, \ldots, f_n$ has infinitely many common zeros, then the u-resultant R_u of the system is identically zero and one cannot compute the common zeros by this method. Therefore, we assume that the given system of polynomials $f_1, f_2, \ldots, f_n$ has only finitely many common zeros. However, even this is not sufficient since the u-resultant vanishes whenever there are infinitely many common zeros of the *homogeneous polynomials* ${}^h f_1, {}^h f_2, \ldots, {}^h f_n$. It may happen that $f_1, f_2, \ldots, f_n$ have only finitely many

common zeros but ${}^hf_1, {}^hf_2, \ldots, {}^hf_n$ have infinitely many common zeros – all but a finite number of them at infinity (when this happens, the zero set is said to have an excess component at infinity).

Example: The u-resultant of

$$f_1 = x_1^2 - x_2^2 + 2x_2 + 1, \text{ and } f_2 = x_1^2 - 2x_1x_2 + x_2^2 + x_1 + 2$$

is zero because they have a common component at infinity given by $x_1 = x_2$.

Often, one has a system of non-homogeneous polynomials that are known to have only finitely many common zeros but the corresponding homogeneous system may have excess components at infinity. The u-resultant algorithm cannot be used as it is in this situation. Grigoriev and Chistov (1983) and Canny (1990) suggest a modification of the algorithm that will give all the affine zeros of the original system (as long as they are finite in number) even in the presence of excess components at infinity. We now briefly describe their approach.

Let $f_1, f_2, \ldots, f_n$ be as before. Let

$$g_i = {}^hf_i + \lambda x_i^{d_i}, \quad \text{for } i = 1, \ldots, n,$$

and

$$g_u = (u_0 + \lambda)x_0 + u_1x_1 + \ldots + u_nx_n$$

where λ is a new unknown. Let $R_u(\lambda, u_0, \ldots, u_n)$ be the Macaulay resultant of $g_1, g_2, \ldots, g_n$ and g_u, regarded as homogeneous polynomials in $x_0, x_1, \ldots, x_n$. $R_u(\lambda, u_0, \ldots, u_n)$ is called the *generalized characteristic polynomial* of $f_1, \ldots, f_n$. Now, look at $R_u(\lambda, u_0, \ldots, u_n)$ as a polynomial in λ whose coefficients are polynomials in $u_0, u_1, \ldots, u_n$, i.e.

$$R_u(\lambda, u_0, \ldots, u_n) = \lambda^\delta + R_{\delta-1}\lambda^{\delta-1} + \ldots + R_k\lambda^k$$

where $k \geq 0$ and the R_i are polynomials in $u_0, u_1, \ldots, u_n$ (if $k = 0$, R_k will be the same as the u-resultant R_u; however, if there are excess components at infinity, then $k > 0$). The trailing coefficient R_k shares a very useful property with the u-resultant, namely, it can be shown that R_k factors into *linear factors* over the complex numbers, i.e.

$$R_k = \prod_{j=1}^{B}(u_0\alpha_{0,j} + u_1\alpha_{1,j} + \ldots + u_n\alpha_{n,j})$$

and if $(u_0\alpha_{0,j} + u_1\alpha_{1,j} + \ldots + u_n\alpha_{n,j})$ is a factor of R_k, then $(\alpha_{0,j}, \alpha_{1,j}, \ldots, \alpha_{n,j})$ is a common zero of of ${}^hf_1, {}^hf_2, \ldots, {}^hf_n$. The converse can also be proved, i.e. if $(\beta_{1,j}, \ldots, \beta_{n,j})$ is an *affine common zero* of $f_1, f_2, \ldots, f_n$ then $(u_0 + u_1\beta_{1,j} + \ldots + u_n\beta_{n,j})$ divides R_k. This gives us a way to recover all the affine common zeros of $f_1, f_2, \ldots, f_n$ even in the presence of excess components at infinity. Again, in practice, one never constructs the complete generalized characteristic polynomial of a system of polynomials. As with the u-resultant, one can recover the affine common zeros by computing specializations of the generalized characteristic polynomial.

2.5. Implementations of Multivariate Resultants

Multivariate resultant algorithms (both Dixon's method for the bivariate case as well as Macaulay's method) can be easily implemented on computer algebra systems since the main calculation needed is that of determinant of a matrix. Sederberg (1983) presents an implementation of Dixon's method. Chionh (1990) reports experimenting with the implementations of Dixon's method and Macaulay resultants in MAPLE for implicitization, parameterization and surface intersection problems. Since the matrices that arise are large with polynomial entries, special techniques such as interpolation and modular methods are needed to do slightly nontrivial examples.

Manocha and Canny (1991) have recently reported impressive results in using multivariate resultant-based methods for implicitization problems using interpolation and modular methods. This is the first and very impressive illustration of multivariate resultants outperforming Gröbner basis methods and characteristic set methods for the implicitization problem.

3. Gröbner Bases Computations

We first establish some of the basic notions needed for our exposition of Gröbner basis computations. The discussion below assumes the coefficient field to be Q. However, the development of Gröbner basis theory as discussed below carries over to polynomial rings in a finite number of variables over most fields (field of complex numbers, finite fields, field of rational functions in k variables over the complex numbers, ...).

3.1. Term Orderings

As said before, a term or power product is any product of powers $x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n}$ of the variables $x_1, x_2, \ldots, x_n$ with $\alpha_j \geq 0$. For a term $t = x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n}$, $\deg(t)$ denotes the total degree of the term t, i.e. $\deg(t) = \alpha_1 + \alpha_2 + \ldots + \alpha_n$. We are interested in *total orderings* (denoted by $\prec$) on terms that satisfy the following properties.

1. *Compatibility with multiplication:* if t, t_1, t_2 are terms, then, $t_1 \prec t_2 \Longrightarrow t\,t_1 \prec t\,t_2$.
2. *Termination:* there can be no strictly decreasing infinite sequence of terms such as

$$t_1 \succ t_2 \succ t_3 \succ \ldots.$$

Such term orderings are called *admissible* orderings and they play a key role in the development of Gröbner basis theory. Commonly used term orderings are

(i) the *Lexicographic Order*, $\prec_l$, in which terms are ordered as in a dictionary i.e. for terms t_1, t_2 with $t_1 = x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n}$ and $t_2 = x_1^{\beta_1}x_2^{\beta_2}\ldots x_n^{\beta_n}$ then $t_1 \prec_l t_2$ iff $\exists i \leq n$ such that $\alpha_j = \beta_j$ for $i < j \leq n$ and $\alpha_i < \beta_i$. For example, for terms made up of two variables x_1, x_2, where $x_1 \prec x_2$, we have

$$1 \prec_l x_1 \prec_l x_1^2 \prec_l x_1^3 \ldots \prec_l x_2 \prec_l x_1x_2 \prec_l x_1^2x_2 \ldots \prec_l x_2^2 \prec_l x_1x_2^2 \prec_l x_1^2x_2^2 \ldots$$

(ii) the *Degree Order*, $\prec_d$, in which terms are compared first by their degrees, and equal degree terms are compared lexicographically i.e.

$$t_1 \prec_d t_2 \text{ iff } \deg(t_1) < \deg(t_2) \text{ or } \deg(t_1) = \deg(t_2) \text{ and } t_1 \prec_l t_2.$$

For example, in the bivariate case, assuming $x_1 \prec x_2$ we have

$$1 \prec_d x_1 \prec_d x_2 \prec_d x_1^2 \prec_d x_1x_2 \prec_d x_2^2 \prec_d x_1^3 \prec_d x_1^2x_2 \prec_d x_1x_2^2 \prec_d x_2^3 \cdots$$

3.2. Head Terms and the Notion of Reduction

Given an admissible term order $\prec$, for every polynomial f in $Q[x_1, x_2, \ldots, x_n]$, we call the largest term (under $\prec$) in f that has a non-zero coefficient as the head term of f, denoted by head(f). By ldcf(f), we denote the leading coefficient of f, i.e. the coefficient of head(f) in f. Clearly, for every polynomial f, we can write

$$f = \text{ldcf}(f)\ \ \text{head}(f) + g \quad \text{where} \quad \text{head}(g) \prec \text{head}(f).$$

We write tail(f) for g. For example, if

$$f(x, y) = x^3 - y^2, \text{ then}$$

$$\text{head}(f) = x^3 \text{ and } \text{tail}(f) = -y^2$$

under the total degree ordering $\prec_d$, and

$$\text{head}(f) = y^2, \qquad \text{tail}(f) = x^3$$

under the purely lexicographic ordering with $x \prec_l y$.

Let f and g be two polynomials; suppose g has a term t with a non-zero coefficient that is a multiple of head(f), i.e.

$$g = at + \hat{g} \text{ where } a \in Q \text{ and } t = t'\ \ \text{head}(f)$$

for some term t'. We say that g is *reducible* with respect to f and denote a *reduction by* f as

$$g \xrightarrow{f} h$$

where

$$h = g - \hat{a}\ \ t'\ \ f = \hat{a}\ \ t'\ \ \text{tail}(f) + \hat{g}, \qquad \text{and} \qquad \hat{a}\ \text{ldcf}(f) \ = \ a.$$

The polynomial g is said to be *reducible* with respect to a set (or basis) of polynomials $F = \{f_1, f_2, \ldots, f_r\}$ if it is reducible with respect to one or more polynomials in F; else we say that g *is reduced* or g *is a normal form* with respect to F.

Given a polynomial g and a basis $F = \{f_1, f_2, \ldots, f_r\}$, through a *finite sequence of reductions*

$$g = g_1 \xrightarrow{F} g_2 \xrightarrow{F} g_3 \ldots \xrightarrow{F} g_s,$$

we can obtain a polynomial g_s that is reduced with respect to F. Two things to note:

- Clearly, any sequence of reductions has to end after a finite number of reductions; if not, we can create an infinite decreasing chain of terms from this sequence which contradicts the assumption that the term ordering being used is terminating.

- For every g_i in the above reduction sequence,

$$g_i - g \in (f_1, f_2, \ldots, f_r).$$

Let us consider an example. Let

$f_1 = x_1^2x_2 - 2x_2x_3 + 1$, $f_2 = x_1x_2^2 - x_3^2 + 2x_1$, $f_3 = x_2^2x_3 - x_1^2 + 5$, and, $g = 3x_1^2x_2^2 + x_1^3 - 1$.

Let $F = \{f_1, f_2, f_3\}$. Under $\prec_d$,

$$\text{head}(f_1) = x_1^2x_2, \qquad \text{head}(f_2) = x_1x_2^2, \qquad \text{head}(f_3) = x_2^2x_3$$

and g is reducible with respect to F. One possible reduction sequence is:

$$g = g_1 \xrightarrow{f_1} g_2 = 6x_2^2x_3 - 3x_2 + x_1^3 - 1 \xrightarrow{f_3} g_3 = 6x_1^2 - 3x_2 + x_1^3 - 31$$

and g_3 is a normal form with respect to F. It is possible to reduce g in another way that leads to a different normal form! For example, we have

$$g = g_1 \xrightarrow{f_2} g_2' = 3x_1x_3^2 + {x_1}^3 - 6x_1^2 - 1$$

and the normal form g_2' is different from g_3. For an arbitrary set of basis polynomials, we cannot expect to avoid this phenomenon. A Gröbner basis has the following special property:

Definition 3.1. *A basis $G \in Q[x_1, x_2, \ldots, x_n]$ is called a Gröbner basis for the ideal it generates if and only if every polynomial in $Q[x_1, x_2, \ldots, x_n]$ has a unique normal form with respect to G.*

Gröbner bases were introduced by Buchberger (1965, 1976). One of his fundamental contributions was to show that every ideal in $Q[x_1, x_2, \ldots, x_n]$ has a Gröbner basis. He designed an algorithm to construct a Gröbner basis for any ideal I in $Q[x_1, x_2, \ldots, x_n]$ starting from an arbitrary basis for I.

Let us consider the earlier example for a moment. We found at least two normal forms (g_3 and g_2') for the polynomial g with respect to F. The reason was that g had a monomial that was reducible by two polynomials in the basis F, i.e. head(g) was a common multiple of head(f_1) and head(f_2). The ambiguity concerning the normal form of g with respect to F can be resolved by augmenting the basis F by the polynomial $g_3 - g_2'$ (the augmented basis still generates the same ideal since $g_3 - g_2' \in (F)$). Buchberger's algorithm attempts to resolve the situation for *all terms that have more than one normal form* with respect to F. The key insight in Buchberger's algorithm is to show that we need to consider only a finite set of terms. To this end, we define an s-polynomial of two polynomials f_1, f_2. Let

$$m = \text{lcm}(\text{head}(f_1), \text{head}(f_2)) = \ m_1\ \ \text{head}(f_1) = \ m_2\ \ \text{head}(f_2)$$

where m_1, m_2 are terms. Define

$$s\text{-poly}(f_1, f_2) = m_1\ \ \text{ldcf}(f_2)f_1 - m_2\ \ \text{ldcf}(f_1)f_2.$$

In the following description of Buchberger's algorithm, by $\mathcal{NF}_G(f)$, we denote *any* normal form of f with respect to the *current elements* of the basis G. The basis G is augmented until $\mathcal{NF}_G(s\text{-poly}(g_i, g_j))$ is zero for the s-polynomial of every pair of polynomials in G.

Given a basis F for an ideal $\mathcal{I}$ and an admissible term ordering $\prec$, the algorithm returns a Gröbner basis for $\mathcal{I}$ for the term ordering $\prec$.

$G := F$;
$B := \{\langle f_i, f_j\rangle \mid f_i, f_j \in F, i < j\}$;
while B **is non-empty do**
  $h := \mathcal{NF}_G(s\text{-poly}(f_i, f_j))$; % for some $\langle f_i, f_j\rangle \in B$
  if $h \neq 0$ **then**
    $B := B \cup \{\langle h, g\rangle | g \in G\}$;
    $G := G \cup \{h\}$;
  fi;
  $B := B \setminus \{\langle f_i, f_j\rangle\}$;
od;

For the well-known proofs of termination and correctness of the algorithm, the reader is referred to Buchberger (1965, 1976). We now list some of the most important properties of Gröbner bases in the form of a theorem.

THEOREM 3.1. *The following properties are equivalent:*

1. *G is a Gröbner basis for the ideal I with respect to a term order $\prec$.*
2. *For every pair of polynomials $g_1, g_2 \in G$, the normal form of the s-polynomial of g_1, g_2 with respect to G is zero.*
3. *Every polynomial f in $Q[x_1, x_2, \ldots, x_n]$ has a unique normal form with respect to G.*
4. *A polynomial f is a member of the ideal I if and only if its normal form with respect to G is zero.*

Examples: Consider the ideal $\mathcal{I}$ generated by

$$f(x,y) = x^3 - y^2,\ g(x,y) = 2x^2 + (y-1)^2 - 2$$

that we saw earlier (f defines a cusp and g defines an ellipse).

$$G_1 = \{y^2 + 2x^2 - 2y - 1, x^3 + 2x^2 - 2y - 1\}$$

is a Gröbner basis for $\mathcal{I}$ under the degree ordering with $x \prec y$.

$$\begin{aligned} G_2 \quad = \quad & \{g_1(x) = x^6 + 4x^5 + 4x^4 - 6x^3 - 4x^2 + 1, \\ & g_2(x,y) = y + 1/2(-x^3 - 2x^2 + 1)\} \end{aligned}$$

is a Gröbner basis for $\mathcal{I}$ under the lexicographic ordering with $x \prec y$.

$$\begin{aligned} G_3 \quad = \quad & \{h_1(y) = y^6 - 6y^5 + 17y^4 + 4y^3 - 9y^2 - 6y - 1, \\ & h_2(x,y) = x + 1/4(2y^5 - 13y^4 + 40y^3 - 10y^2 - 18y - 5)\} \end{aligned}$$

is a Gröbner basis for $\mathcal{I}$ under the lexicographic ordering with $y \prec x$.

These examples illustrate the fact that, in general, an ideal has different Gröbner bases depending on the term ordering that we choose. But, can an ideal have different Gröbner bases for the same term ordering? The answer is *no*, provided we restrict our attention to the so-called reduced Gröbner bases.

DEFINITION 3.2. *A Gröbner basis is called a* reduced *Gröbner basis iff for every $g \in G$,*

$$\mathcal{NF}_{G'}(g) = g$$

where $G' = G \setminus \{g\}$, i.e. each polynomial in the basis G is reduced with respect to all the other polynomials in G.

In the above example, G_1, G_2, G_3 are all reduced Gröbner bases. The basis

$$G_0 = \{f\} \cup G_1$$

is a Gröbner basis for $\mathcal{I}$ under the degree lexicographic ordering. However, it is not a reduced basis since f is reducible with respect $G_0 \setminus \{f\}$. From here on, by a Gröbner basis, we mean a reduced Gröbner basis unless specified otherwise. The choice of the term ordering depends on what we wish to use a Gröbner basis for. Also, the time needed to compute a Gröbner basis is very sensitive to the term ordering used.

In the Gröbner basis G_2 for the above example, there is a polynomial $g_1(x)$ that depends only on x and one that depends on both x, y (in general, there can be several polynomials in a Gröbner basis that depend on x, y). We say that the *variables are separated in the basis G_2*. A basis in which variables are separated is similar to a triangular form; in a triangular form, there is at most one polynomial in $x_1, x_2, \cdots, x_i$ for each $1 \leq i \leq n$, but in a basis in which variables are separated, there can be more than one polynomials in $x_1, x_2, \cdots, x_i$ for each $1 \leq i \leq n$. Such a separation of variables can be used to compute all the common zeros of the ideal $\mathcal{I}$. We first find all the roots of the univariate polynomial $g_1(x)$. These give the x-coordinates of the common zeros of the ideal $\mathcal{I}$. Similar to the case of a linear system of equations, we can perform back substitution. For each root α of g_1, we can find the common roots of $g_2(\alpha, y), \ldots$ which give the y-coordinates of the corresponding common zeros of $\mathcal{I}$. This algorithm is mentioned only to indicate the kind of uses that one can derive from Gröbner bases.

Ideal bases in which the variables are separated are very useful in solving a variety of problems. In fact, the separation observed in Gröbner bases G_2, G_3 above is not accidental. It was observed by Trinks (and Buchberger) that such a separation of variables exists in Gröbner bases whenever the term ordering used is a lexicographic one. We now illustrate the observations of Trinks (1978) and Buchberger (1985).

Let $\mathcal{I}$ be an ideal in the polynomial ring $Q[x_1, x_2, \ldots, x_n]$ and let $j < n$. The set of all the polynomials in $\mathcal{I}$ that depend only on $x_1, \ldots, x_j$ constitute a sub-ideal of $\mathcal{I}$. More precisely, let

$$\mathcal{I}_j = \mathcal{I} \cap Q[x_1, \ldots, x_j].$$

The ideal $\mathcal{I}_j$ is called the *contraction of the ideal $\mathcal{I}$ to the subring $Q[x_1, \ldots, x_j]$*. Some authors refer to it as the j-th elimination ideal of $\mathcal{I}$.

THEOREM 3.2. *Let G be the reduced Gröbner basis for an ideal $\mathcal{I} \subseteq Q[x_1, x_2, \ldots, x_n]$ under the lexicographic order on terms with $x_1 \prec x_2 \prec \ldots \prec x_n$. Then,*

$$\mathcal{I} \cap Q[x_1, \ldots, x_j] = (G \cap Q[x_1, \ldots, x_j])Q[x_1, \ldots, x_j]$$

for each $i = 1, 2, \ldots, n$.

Here, by $(G \cap Q[x_1, \ldots, x_j])Q[x_1, \ldots, x_j]$, we mean the ideal generated by $(G \cap Q[x_1, \ldots, x_j])$ in the ring $Q[x_1, \ldots, x_j]$, i.e.

$$(G \cap Q[x_1, \ldots, x_j])Q[x_1, \ldots, x_j] = \{\sum g_i h_i \mid g_i \in (G \cap Q[x_1, \ldots, x_j]), h_i \in Q[x_1, \ldots, x_j]\}.$$

Example: Let $F = \{f_1, f_2, f_3\}$ where

$$\begin{aligned} f_1 &= x_3^4 - 6x_1x_3^3 + 13x_3^2x_1^2 - 12x_3x_1^3 + 4x_1^4, \\ f_2 &= x_2^2 - 2x_2x_1 - 2x_2x_3 + x_1^2 + 2x_1x_3 + x_3^2, \\ f_3 &= x_1^2 + 4x_1 + 3, \\ f_4 &= x_1x_3^3 - 3x_3^2x_1^2 + 3x_3x_1^3 - x_1^4 \\ &\quad + x_3^3 - 3x_3^2x_1 + 3x_3x_1^2 - x_1^3 + x_1x_3 - 2x_1^2 + 3x_3 - 6x_1 \end{aligned}$$

Let G denote the reduced Gröbner basis for (F) under the lexicographic ordering with $x_1 \prec x_2 \prec x_3$. We have

$$G = \{g_1, g_{2,1}, g_{2,2}, g_{3,1}, g_{3,2}, g_{3,3}\}$$

where

$$\begin{aligned} g_1 &= x_1^2 + 4x_1 + 3, \\ g_{2,1} &= x_1x_2^2 + 3x_2^2 + 9x_1 + 27 + 6x_2x_1 + 18x_2, \\ g_{2,2} &= 2x_2^3 + 36x_2^2 - 27x_2x_1 + 135x_2 - 108x_1 + 108, \\ g_{3,1} &= x_1x_3 + 2x_1 + 3x_3 + 6, \\ g_{3,2} &= 4x_2x_3 + 24x_3 - 2x_2^2 + 4x_2x_1 + 15x_1 + 45, \\ g_{3,3} &= 2x_3^2 + 12x_3 - x_1 + 15, \end{aligned}$$

and,

$$\mathcal{I}_1 = (g_1)Q[x_1],\ \mathcal{I}_2 = (g_1, g_{2,1}, g_{2,2})Q[x_1, x_2],\ \mathcal{I}_3 = (G)Q[x_1, x_2, x_3] = \mathcal{I}.$$

In principle, any system of polynomial equations can be solved using a lexicographic Gröbner basis for the ideal generated by the given polynomials by the algorithm outlined above. The fact that Gröbner bases under lexicographic term orderings exhibit separation of variables is useful in many situations. In geometric modeling, this property has been used

- to compute intersections of curves and surfaces,
- to find an implicit equation for a curve or surface given parametrically, and
- to determine whether a given rational parameterization of a curve or surface is faithful.

For further details and illustrations of the above applications, we refer the reader to Manocha and Canny (1990), Hoffman (1989, 1990) and Hoffman and Vermeer (1991).

If a set of polynomials does not have a common zero, i.e. its ideal is the whole ring, then it is easy to see that a Gröbner basis of such a set of polynomials includes 1 no matter what term ordering is used. Gröbner basis computations can thus be used to check for the consistency of a system of nonlinear polynomial equations.

THEOREM 3.3. *A set of polynomials in $Q[x_1, \cdots, x_n]$ has no common zero in C if and only if their reduced Gröbner basis with respect to any admissible term ordering is* $\{1\}$.

A refutational method based on this result is proposed for automatically proving geometry theorems in Kapur (1986, 1988). A refutational approach using Gröbner basis computations for propositional calculus as well as for first-order predicate calculus is discussed in Kapur and Narendran (1985).

3.3. Gröbner Bases in Practice

In general, Gröbner bases are hard to compute. That it is inherently so was shown by Mayr and Meyer (1982) whose result can be used to exhibit ideals for which doubly exponential degree explosions during Gröbner basis computations are inevitable. In addition, the coefficients of polynomials that get generated during Gröbner bases computations can get extremely large. Another problem can arise due to the choice of the term ordering used in a Gröbner basis computation. We have seen that lexicographic bases or those similar to lexicographic bases are the most useful. However, in practice, lexicographic bases are known to be the hardest to compute.

Despite these difficulties, highly non-trivial Gröbner bases computations have been performed. Computations with ideals in polynomial rings over the rational numbers with 8-10 variables with degrees of polynomials in the initial basis about 5 are feasible. If the coefficients belong to a finite field (typically $\mathcal{Z}_p$ where p is a word sized prime), much larger computations are possible. Macaulay (Bayer and Stillman, 1989) and CoCoA (Giovini and Niesi, 1990) are specialized computer algebra systems built for performing large computations in algebraic geometry and commutative algebra; they are quite easy to use and provide a variety of built-in functions for computing with Gröbner bases. Most general computer algebra systems (such as MAPLE, MACSYMA, MATHEMATICA, REDUCE) also provide the basic Gröbner basis functions. An implementation of Gröbner basis algorithm also exists in GeoMeter (Cyrluk *et al.*, 1988; Connolly *et al.*, 1989), a programming environment for geometric modeling and algebraic reasoning. This implementation has been used for proving nontrivial plane geometry theorems using a refutational approach discussed in Kapur (1988).

Most Gröbner basis implementations use several modifications to Buchberger's algorithm in order to speed up the computations. We now briefly describe some of the common improvements.

Recall that in Buchberger's algorithm, one computes a normal form of an s-polynomial with respect to the current basis and, if it is non-zero, augments the current basis with the normal form. The s-polynomial reductions are repeated until all s-polynomials have normal form zero. An s-polynomial reduction is said to be ***useless*** if it does not produce a new polynomial to augment the current basis with (i.e. a reduction that produces a zero normal form). It has been observed that a lot of time is spent in performing useless reductions and one would like to avoid as many useless reductions as possible. In order to facilitate this, Buchberger proposed some simple conditions for predicting useless reductions. Since then, several researchers have invented variations and extensions to Buchberger's criteria. We only present Buchberger's criteria.

- If the head terms of g_i and g_j are co-prime, then s-polynomial of g_1, g_2 can be reduced to zero by the current basis. Hence, there is no need to perform the reduction.
- If g, g_1, g_2 are such that head(g) divides lcm(head(g_1), head(g_2)) and reductions of s-poly(g, g_1) and s-poly(g, g_2) are already done, then it is not necessary to perform the reduction of s-poly(g_1, g_2).

The implementation of these criteria for predicting useless reductions along with the so-called ***normal selection strategy*** (each time, the s-polynomial of the pair g_i, g_j for which

lcm(head(g_1), head(g_2)) is the smallest among all the untried pairs in the current basis) is known to improve the running time of Buchberger's algorithm significantly. For complete details, we refer the reader to Gebauer and Möller (1988) and Giovini *et al.* (1991).

3.4. Zero Dimensional Ideals and Basis Conversion

We mentioned earlier the sensitivity of Gröbner basis computations to the term ordering being used. It is observed in practice that lexicographically ordered Gröbner bases (and lexicographic-like ordered bases, namely, the block ordered bases) are much harder to compute than the total degree ordered bases. However, for a number of applications, as we have already seen, one needs to compute a Gröbner basis under a lexicographic or lexicographic-like ordering. This raises the following question:

- *Suppose we are given the reduced Gröbner basis G_1 for an ideal $\mathcal{I}$ under a degree ordering, can one compute the reduced Gröbner basis for $\mathcal{I}$ under a lexicographic ordering much faster than by a direct computation using Buchberger's algorithm on G_1?*

Faugère *et al.* (1989) provided an elegant answer to this question for a special class of ideals called *zero-dimensional ideals.*

Recall that an ideal $\mathcal{I} \in \mathcal{Q}[x_1, x_2, \ldots, x_n]$ is said to be zero-dimensional if Zero($\mathcal{I}$) is finite. In other words, there are only finitely many common zeros of the polynomials in $\mathcal{I}$. The following property of Gröbner bases, observed first by Buchberger, characterizes 0-dimensional ideals:

THEOREM 3.4. *Let G be the reduced Gröbner basis for an ideal $\mathcal{I} \in \mathcal{Q}[x_1, x_2, \ldots, x_n]$ under* any *admissible term ordering. $\mathcal{I}$ is zero-dimensional iff, for each i, $1 \leq i \leq n$, G contains a polynomial whose head term is a pure power of x_i, i.e. of the form $x_i^{d_i}$ for some integer d_i.*

A term t is said to be reduced with respect to G if t is not divisible by the head term of any polynomial in G. The condition just mentioned amounts to saying that the number of terms reduced with respect to G is finite iff the ideal $\mathcal{I}$ is zero dimensional. The number of terms reduced with respect to G is an important invariant (we denote it by D) of the ideal $\mathcal{I}$. It is the same as the cardinality of Zero($\mathcal{I}$) or the number of common zeros, *counted with multiplicities.* The algorithm of Faugère, Gianni, Lazard and Mora does the following:

- *Given a Gröbner basis G_1 for a zero-dimensional ideal $\mathcal{I}$ under some term ordering $\prec_1$, compute a reduced Gröbner basis G_2 for $\mathcal{I}$ under a second term ordering $\prec_2$.*

The original intent of Faugère *et al.* (1989) was to find all the common zeros of $\mathcal{I}$ quickly. Their algorithm, which we refer to as the *basis conversion algorithm*, is typically used as follows for solving zero-dimensional systems of equations:

- Compute a Gröbner basis G_1 for the ideal generated by the given system of polynomials under a total degree ordering $\prec_1$.
- Use the *basis conversion algorithm* to obtain a Gröbner basis G_2 under a lexicographic ordering $\prec_2$.

- Apply the *back substitution* phase (described earlier) on G_2 to obtain the common zeros.

The authors report spectacular success for this approach on several bench-mark systems of equations including cases where a direct computation of a lexicographic basis had never been carried out (usually the machine would run out of space on these examples!). Since then, variations of this technique have been developed to solve several problems related to zero-dimensional ideals (Lakshman, 1990).

The basis conversion algorithm enumerates terms starting from 1, in the *increasing order with respect to the second (new) term ordering $\prec_2$* . As it considers a term t, it classifies t as one of the following using the Gröbner basis G_1.

- t is *reduced* with respect to G_2, or
- t is *a lead term* of some polynomial in G_2, or
- t is *a multiple* of some lead term with respect to G_2.

Note that G_2 is the *desired basis* and hence the above classification is non-trivial. In fact, this classification is what leads to the construction of G_2. The complete description of the basis conversion algorithm follows:

- Let $\mathcal{NF}(t)$ denote the normal form of the term t with respect to G_1.
- Newbasis: Gröbner basis being built.
- ReducedTerm: Set of monomials that are known to be reduced with respect to Newbasis; Initialized to $\{1\}$.
- NextTerm: Function that returns the smallest monomial (under the desired admissible term ordering) that is neither in ReducedTerm nor is a multiple of some lead term in Newbasis. Returns false if no such monomial exists.

```
ReducedTerm:= {1};
Newbasis:= { };
while (t := NextTerm()) do
   If there exist t_1, ..., t_s in ReducedTerm, and λ_j ∈ Q
  such that NF(t) + Σ_{j=1}^{s} λ_j NF(t_j) = 0, then,
       Newbasis := Newbasis ∪ {t + Σ_{j=1}^{s} λ_j t_j}
   else
       ReducedTerm := ReducedTerm ∪ {t};
       Save NF(t);
   fi
od end;
```

Example: Consider the ideal $\mathcal{I} = (f, g)$ where $f = x^3 - y^2$, $g = (y-1)^2 + 2x^2 - 2$. The basis $G = \{f_1 = y^2 - 2y - 1 - 2x^2, f_2 = x^3 + 2x^2 - 2y - 1\}$ is the reduced Gröbner basis for $\mathcal{I}$ under the degree ordering with $x \prec y$. Note that under this ordering, x^3 is the head term of f_2 and y^2 is the head term of f_1. Indeed, $\mathcal{I}$ is a zero-dimensional ideal since G has a polynomial whose lead term is a pure power of x and one whose lead term is a pure power of y (we have already seen that there are only finitely many common zeros of $\mathcal{I}$). Suppose we wish to compute the reduced Gröbner basis G_2 for $\mathcal{I}$ under the lexicographic term order with $y \prec_l x$. We proceed thus:

We begin by looking at the term y (the smallest term under $\prec_l$ that is not yet classified). At this point, we only know that the term 1 is reduced with respect to G_2. We note that

$\mathcal{NF}(y) = y$. Our attempt to find a rational constant λ_1 such that

$$\mathcal{NF}(y) + \lambda_1(\mathcal{NF}(1)) = 0$$

ends in failure. We therefore conclude that y is *reduced* with respect to G_2 and consider y^2. We find that $\mathcal{NF}(y^2) = 2y + 1 - 2x^2$ and this time, we look for rational constants λ_1, λ_2 such that

$$\mathcal{NF}(y^2) + \lambda_2(\mathcal{NF}(y)) + \lambda_1(\mathcal{NF}(1)) = 0. \qquad (iv)$$

Since rational constants λ_1, λ_2 satisfying the above relation do not exist, we conclude that y^2 is also reduced with respect to G_2. We then consider y^3, y^4, y^5 (all of which fail to produce a linear relation of the type (iv)) and finally, y^6. At this point, we are looking for rational constants $\lambda_1, \lambda_2, \lambda_4, \lambda_4, \lambda_5, \lambda_6$ such that

$$\mathcal{NF}(y^6) + \lambda_6(\mathcal{NF}(y^5)) + \lambda_5(\mathcal{NF}(y^4)) + \lambda_4(\mathcal{NF}(y^3)) + \lambda_3(\mathcal{NF}(y^2)) + \lambda_2(\mathcal{NF}(y)) + \lambda_1(\mathcal{NF}(1)) = 0.$$

Substituting the appropriate normal forms, we have

$$\begin{aligned}
(72yx^2+ \quad & 176xy - 570y - 259 + 506x^2 + 76x) \\
+ \quad & \lambda_6(-12yx^2 + 52xy - 107y - 52 + 96x^2 + 24x) \\
+ \quad & \lambda_5(-4y - 8yx^2 - 3 + 4x^2 + 8xy + 4x) \\
+ \quad & \lambda_4(5y - 2yx^2 + 2 - 4x^2) \\
+ \quad & \lambda_3(2y + 1 - 2x^2) \\
+ \quad & \lambda_2(y) \\
+ \quad & \lambda_1(1) = 0.
\end{aligned}$$

Equating the coefficients of like terms, we have

$$\begin{aligned}
-12\lambda_6 - 8\lambda_5 - 2\lambda_4 + 72 &= 0, && \{\text{coeff. of } yx^2\} \\
52\lambda_6 + 8\lambda_5 + 176 &= 0, && \{\text{coeff. of } xy\} \\
96\lambda_6 + 4\lambda_5 - 4\lambda_4 - 2\lambda_3 + 506 &= 0, && \{\text{coeff. of } x^2\} \\
-107\lambda_6 - 4\lambda_5 + 5\lambda_4 + 2\lambda_3 + \lambda_2 &= 0, && \{\text{coeff. of } y\} \\
24\lambda_6 + 4\lambda_5 + 76 &= 0, && \{\text{coeff. of } x\} \\
-52\lambda_6 - 3\lambda_5 + 2\lambda_4 + \lambda_3 + \lambda_1 &= 0, && \{\text{constant term}\}
\end{aligned}$$

This is a system of linear equations in the variables λ_i and can be solved easily. The unique solution is

$$\lambda_6 = -6,\ \lambda_5 = 17,\ \lambda_4 = 4,\ \lambda_3 = -9,\ \lambda_2 = -6,\ \lambda_1 = -1$$

(the uniqueness of the solution can be deduced from the uniqueness of the reduced Gröbner basis for $\mathcal{I}$ with respect to the term order $\prec_l$). Therefore, we classify y^6 as a head term with respect to G_2 and add the polynomial

$$y^6 - 6y^5 + 17y^4 + 4y^3 - 9y^2 - 6y - 1$$

to the basis G_2. The next term that is considered by the algorithm is x (at this point, x is the *smallest unexamined* term according to $\prec_l$ that is not a multiple of any term known to be a lead term with respect to G_2). We now look for a linear relation among $\mathcal{NF}(x), \mathcal{NF}(y^5), \mathcal{NF}(y^4), \mathcal{NF}(y^3), \mathcal{NF}(y^2), \mathcal{NF}(y)$ and 1. Such a relation exists and we

find that

$$x + 1/4(2y^5 - 10y2 - 13y4 + 40y3 - 18y - 5)$$

to be the polynomial resulting from the linear relation. Therefore, we add it to the basis G_2. We now know two lead terms in G_2, namely, y^6, x. The next term that we examine must not be divisible by either y^6 or x. But we have already examined all such terms and classified them. Hence, the algorithm terminates, and we have the basis

$$G_2 = \{x + 1/4(2y^5 - 10y2 - 13y4 + 40y3 - 18y - 5), y^6 - 6y^5 + 17y^4 + 4y^3 - 9y^2 - 6y - 1\}.$$

Note that the λ_i are determined by solving a linear system of equations. For the purposes of illustration, we wrote down a complete linear system. The linear systems of equations that arise in this algorithm have a nice structure (a consequence of the way they are generated) and in practice, one takes advantage of the structure to find the λ_i efficiently. It is shown in Faugère *et al.* (1989) that the number of rational arithmetic operations performed by the basis conversion algorithm is $O(n^2D^3 + n^2D^2\log(nD))$ (recall that D is the number of reduced terms with respect to G_1; n is the number of variables). In order to achieve the above bound for the running time of the algorithm, it is necessary to save all the normal forms of the terms computed by the algorithm (in the step save $\mathcal{NF}(t)$).

The termination of the basis conversion algorithm follows from the property that zero-dimensional ideals have only finitely many terms that are in normal form with respect to its Gröbner basis constructed using an admissible ordering. The fact that the basis generated from the above construction is a Gröbner basis is a corollary of the following property of Gröbner bases.

DEFINITION 3.3. *Given a basis F, and a term order $\prec$, define $init_\prec(F)$ to be the set of all head terms of the polynomials in F. For an ideal I, define the* initial ideal *with respect to $\prec$ to be the ideal generated by the set $init_\prec(\mathcal{I})$.*

It can be shown that a basis G of an ideal $\mathcal{I}$ is a Gröbner basis with respect to $\prec$ if and only if the initial ideal of $\mathcal{I}$ with respect to $\prec$ is generated by the head terms of the polynomials in the basis G. The basis conversion algorithm has been used for computing the implicit equation of a parametrically given surface (see Hoffman, 1989).

3.5. TRIANGULAR SETS

We have seen earlier how lexicographic Gröbner bases can be used to solve systems of polynomial equations. A Gröbner basis contains all the information about the zeros of the ideal it generates, including multiplicities (this fact is essential when one wants to compute the primary decomposition of an ideal; see Lakshman, 1990, for instance). However, if one is interested merely in the location of the zeros of an ideal (let us assume for the moment that we are dealing with zero dimensional ideals), then the multiplicities can slow down the computation of the coordinates of the zeros of the ideal. It is possible to obtain sets of polynomials in which all the variables are separated as in a lexicographic Gröbner basis but the sets are much *simpler* than a lexicographic Gröbner basis.

Kandri-Rody (1984) showed how to construct such a set from a lexicographic Gröbner basis and called it an *extracted characteristic* set following Ritt. Extracted characteristic sets were used by Kandri-Rody for testing the primality of an ideal as well as for computing the dimension of an ideal. Lazard (1989a, 1989b) also defined triangular sets

for studying zero sets ideals. In the rest of this section, we describe Lazard's triangular sets for zero dimensional ideals and present an algorithm due to Lazard for computing triangular sets from a lexicographic Gröbner basis.

A triangular set is any set of polynomials

$$f_1, f_2, \ldots, f_n \in Q[x_1, x_2, \ldots, x_n]$$

such that

$$f_1 \in Q[x_1],\ f_2 \in Q[x_1, x_2],\ f_3 \in Q[x_1, x_2, x_3],\ \ldots, f_n \in Q[x_1, x_2, \ldots, x_n].$$

The highest variable under the ordering $x_1 \prec x_2 \prec \ldots \prec x_n$ appearing in a polynomial f is called the *main variable* of f. By $\deg(f_i)$, we mean the degree of f_i in its main variable x_i, and, by $\deg_j(f_i)$, we mean the degree of f_i in x_j. The triangular set is called *reduced* if $\deg_j(f_i) < \deg(f_j)$ for all $j < i \leq n$. In this subsection, by a triangular set, we mean a reduced triangular set in which every polynomial is monic in its main variable, i.e. its initial is 1. It can be shown that *the zero set of every zero-dimensional ideal $\mathcal{I}$ is the union of the zero-sets of finitely many distinct triangular sets.* We say that two sets of polynomials are *equivalent* if they have the same zero set.

Example: Consider the ideal $\mathcal{I}$ given by the basis G below. In fact, G is the reduced Gröbner basis for $\mathcal{I}$ under the pure lexicographic ordering with $x \prec y \prec z$.

$$\begin{aligned} G = \{g_6 &= 4z^2 + 4xz + 13x^3 - 88x^2 + 173x - 94, \\ g_5 &= yz - y - z + 1, \\ g_4 &= x^2z - x^2 - 3xz + 3x + 2z - 2, \\ g_3 &= y^2 - 1, \\ g_2 &= yx^2 - 6yx + 9y - x^2 + 6x - 9, \\ g_1 &= x^4 - 9x^3 + 29x^2 - 39x + 18\} \end{aligned}$$

Given below is one possible triangular set decomposition of the zero set of $\mathcal{I}$,

$$\{(x^2 - 3x + 2, y - 1, z^2 + xz + 1),\ (x^2 - 6x + 9, y^2 - 1, z - 1)\}$$

whose structure is simpler than that of the Gröbner basis G.

We now sketch Lazard's algorithm for computing a triangular set decomposition of a zero set informally using the above example. In a reduced lexicographic Gröbner basis of a zero dimensional ideal, there is always a single polynomial in the lowest variable; this polynomial goes into a triangular set. However, there can be many polynomials in the other variables. For the above example, there are two polynomials, g_2, g_3, in x, y and there are three polynomials, g_4, g_5, g_6, in x, y, z. We attempt to make the smallest polynomial in x, y, i.e. g_2, monic. When considered as a polynomial in y, it has $x^2 - 6x + 9$ as the leading coefficient. We attempt to compute the inverse of $x^2 - 6x + 9$ modulo the polynomial g_1, i.e. try to compute a polynomial h such that

$$(x^2 - 6x + 9)h \equiv 1 \pmod{g_1}.$$

But $g_1 = x^4 - 9x^3 + 29x^2 - 39x + 18 = (x^2 - 6x + 9)(x^2 - 3x + 2)$, implying that $x^2 - 6x + 9$ cannot be inverted modulo g_1. Therefore, we try to split the triangular set constructed thus far into two triangular sets: $T^{(1)}$ containing $x^2 - 6x + 9$ and $T^{(2)}$ containing the $x^2 - 3x + 2$. For each triangular set, we repeat the above operation of making the remaining polynomials in the Gröbner basis monic.

With respect to $T^{(1)}$, the second polynomial $g_2 = yx^2 - 6yx + 9y - x^2 + 6x - 9$ in the Gröbner basis simplifies to 0. The third polynomial g_3 is already monic, so it is added to $T^{(1)}$. Consider g_4, the smallest polynomial in x, y, z, whose leading coefficient is $x^2 - 3x + 2$. We attempt to invert it with respect to $T^{(1)}$; using the extended Euclidean algorithm, as discussed below, its inverse with respect to $x^2 - 6x + 9$ is $-3/4x + 11/4$, i.e.

$$(x^2 - 3x + 2)(-3/4x + 11/4) = 1 \ \ mod \ \ x^2 - 6x + 9.$$

When we multiply g_4 by $-3/4x + 11/4$ and simplify by $T^{(1)}$, we get $z - 1$, which is added to $T^{(1)}$; we have now completed the computation of one triangular set. It is easy to see that g_5 and g_6 simplify to 0 with respect to $T^{(1)}$.

The computation of the second triangular set $T^{(2)}$ containing $x^2 - 3x + 2$ is done in the same way. The inverse of the leading coefficient of g_2 can be computed with respect to $T^{(2)}$ using which g_2 is made monic, and $y - 1$ is added to $T^{(2)}$. Polynomial g_3 simplifies to 0 using $T^{(2)}$. Polynomials g_4, g_5 also simplify to 0 using $T^{(2)}$. Simplifying g_6 gives $z^2 + xz + 1$ which is added to $T^{(2)}$.

As the reader might have noticed, two operations are needed with respect to a triangular set: *simplification/reduction* and *inversion*. They are discussed next.

3.5.1. Reduction with respect to a Triangular Set

Given a triangular set $T = \{f_1, f_2, \ldots, f_n\}$ with x_i being the main variable in f_i and a polynomial $f \in Q[x_1, \ldots, x_n]$. Let d_i be the degree of f_i in its main variable x_i. The *remainder* of f with respect to T is defined as follows. We *divide* f by f_n with a remainder, i.e.

$$f = Q_n f_n + r_n$$

where Q_n is the quotient, and r_n is the remainder. The degree of r_n in x_n is less than the degree of f_n in x_n. We can now divide r_n by f_{n-1} treating them both as a polynomials in x_{n-1}. The coefficients of f_{n-1} are polynomials in $x_1, \ldots, x_{n-2}$ and the coefficients of r_n are polynomials in $x_1, \ldots, x_{n-2}, x_n$, i.e. we have

$$r_n = Q_{n-1} f_{n-1} + r_{n-1}$$

where the degree of the remainder r_{n-1} in $x_n < d_n$ and its degree in $x_{n-1} < d_{n-1}$. We can compute successive remainders $r_{n-2}, \ldots, r_1$ with respect to $f_{n-2}, \ldots, f_1$. The last remainder r_1 is such that $\deg_i(r_1) < d_i$ for $1 \leq i \leq n$ and it is called the *remainder* of f with respect to T. The process of obtaining r_1 from f and T is called *reducing* f by T.

Example: Let $f = z^2 + y^2 + x^2$. Its remainder with respect to the triangular set $(x^2 - 3x + 2, y - 1, z^2 + xz + 1)$ with $x \prec y \prec z$ is computed as follows:

$$\begin{aligned} \text{the remainder of } f \text{ using } (z^2 + xz + 1) &= x^2 + y^2 - xz - 1 \\ \text{the remainder of } (x^2 + y^2 - xz - 1) \text{ using } (y - 1) &= x^2 - xz \\ \text{the remainder of } (x^2 - xz) \text{ using } (x^2 - 3x + 2) &= -xz + 3x - 2 \end{aligned}$$

The polynomial $-xz + 3x - 2$ is the remainder of f with respect to the triangular set. Notice that the reduction of a polynomial f with respect to a triangular set T is the same as computing the normal form of f with respect to T (which also happens to be a reduced Gröbner basis).

3.5.2. Inversion with respect to a Triangular Set

Let $T = \{f_1, f_2, \ldots, f_n\}$ be a triangular set with x_i being the main variable of f_i. Without any loss of generality, we can assume that f is already reduced with respect to T (if f is not already reduced with respect to T, reduce it using T). By inverting a polynomial f with respect to T, we mean finding a polynomial g such that the remainder of gf with respect to T is 1. Not every polynomial has an inverse with respect to a triangular set T. In case f does not have an inverse with respect to T, the process of trying to invert f *may* lead to a splitting of T. Let x_i be the highest variable appearing in f. Treating f and f_i as polynomials with coefficients that are polynomials in $x_1, \ldots, x_{i-1}$, i.e. $f, f_i \in \mathcal{Q}[x_1, \ldots, x_{i-1}][x_i]$, compute their greatest common divisor using the extended Euclidean algorithm (see Knuth, 1980, pp. 407–408). The crucial point here is that we proceed as though the leading coefficients of the remainders that appear in performing Euclid's algorithm on f and f_i are *invertible* in T. Let g be their gcd. We have

$$g = fp + f_i q$$

where p, q are the multipliers produced by the extended Euclidean algorithm. There are three possibilities:

- if $g = 1$, then p is the inverse of f with respect to T;
- if $g = f_i$, then f_i divides f and f is *not* invertible with respect to T.
- if $f_i \neq g \neq 1$, then f_i has factors $g, f_i/g$ and the triangular set T is now *equivalent* to the union of the two triangular sets

  $$T^{(1)} = \{f_1, f_2, \ldots, g, f_{i+1}, \ldots, f_n\}, \text{ and, } T^{(2)} = \{f_1, f_2, \ldots, f_i/g, f_{i+1}, \ldots, f_n\}.$$

  f is not invertible with respect to $T^{(1)}$ and the inverse of f with respect to $T^{(2)}$ is given by $g^{-1}p$ where g^{-1} denotes the inverse of g with respect to $T^{(2)}$.

The crucial point is that the triangular set T can split at *lower levels*. This is because the extended Euclidean algorithm used to compute the gcd of f, f_i needs to compute *inverses* of polynomials with respect to f_j, $j < i$, which is done recursively.

To summarize, the operation of inverting a polynomial f with respect to a triangular set T produces a family of triangular sets $T^{(1)}, T^{(2)}, \ldots, T^{(k)}$ and polynomials $h_1, \ldots, h_k$ such that

- T is *equivalent* to the union of the triangular sets $T^{(1)}, T^{(2)}, \ldots, T^{(k)}$.
- if $h_i \neq 0$, then h_i is the inverse of f with respect to $T^{(i)}$, i.e. the remainder of fh_i with respect to $T^{(i)}$ is 1; if $h_i = 0$, then f is not *invertible* with respect to $T^{(i)}$.

What we have sketched here is, in essence, the *D5 method* of Duval for handling algebraic numbers. We illustrate the operation of inversion with respect to a triangular set below. For complete details on the D5 technique, we refer the reader to Duval (1991).

Example: Let us invert the polynomial $f = (x-1)z + 1$ with respect to the triangular set $T = (x^2 - 3x + 2, y - 1, z^2 + xz + 1)$ of the previous example with the ordering $x \prec y \prec z$.

The first step is to compute the gcd of $f, z^2 + xz + 1$, treating them as polynomials in z. At this point, we are required to invert $x - 1$ (the leading coefficient of f) with respect to T. Since $x - 1$ is independent of y, z, we try to compute the gcd of $x^2 - 3x + 2, x - 1$ treating them as polynomials in x; we find that $x - 1$ is their *gcd*. This leads to a split of T into $T^{(1)} = (x - 1, y - 1, z^2 + xz + 1)$ and $T^{(2)} = (x - 2, y - 1, z^2 + xz + 1)$. The polynomial $x - 1$ is not invertible with respect to $T^{(1)}$; it has an inverse with respect to $T^{(2)}$, which is 1 (since $(x-1).1 \equiv 1 \bmod (x-2)$). $T^{(1)}, T^{(2)}$ can be reduced to give $T^{(1)} = (x - 1, y - 1, z^2 + z + 1)$ and $T^{(1)} = (x - 1, y - 1, z^2 + 2z + 1)$. Note that we may have to *reduce* the triangular sets as we propagate the split upwards.

Our task now is to invert f with respect to $T^{(1)}$ and $T^{(2)}$ separately. The polynomial f has remainder 1 with respect to $T^{(1)}$ and 1 is its own inverse; f has remainder $z + 1$ with respect to $T^{(2)}$. We find that the gcd of $z + 1, z^2 + 2z + 1$ is $z + 1$. Therefore, f is not invertible with respect to $T^{(2)}$.

3.5.3. Lazard's Algorithm for Computing a Family of Triangular Sets

The algorithm that we present is called *D5Lextriangular* in Lazard (1989a). It uses the D5 method to compute a triangular set decomposition of the zero set of a zero dimensional ideal given a lexicographic Gröbner basis for the ideal. We would like to point out that there are algorithms for computing triangular sets that do not need a lexicographic Gröbner basis for input. We refer the interested reader to Lazard (1989a) and Lakshman (1990a).

Algorithm D5Lextriangular

Input: A reduced Gröbner basis G for a zero dimensional ideal $\mathcal{I}$ under the lexicographic ordering with $x_1 \prec x_2 \prec \ldots \prec x_n$; it is assumed that G is presented as a list with the head terms of the polynomials sorted in increasing order under $\prec$.

Output: A list of triangular sets equivalent to G.

Functions used:

Reduce(f, T): reduces the polynomial modulo the triangular set T.

Inverse(f, TL): f is a polynomial and TL is a list of triangular sets $U_1, \ldots, U_l$ (the list can be empty). The function returns a list of pairs, $[(h_1, T_1), \ldots, (h_k, T_k)]$, $k \geq 1$, such that the union of the triangular sets in TL is equivalent to the union of triangular sets $T_1, \ldots, T_k$; if $h_i \neq 0$, then, h_i is the inverse of f modulo T_i, else f is not invertible modulo T_i.

Lcoeff(f, x): returns the leading coefficient of f treated as a univariate polynomial in the variable x.

$TL :=$ [[*first*(G)]]; % start with the univariate polynomial in G.
for i **from** 2 **to** n **do**
  $H :=$ *sublist of polynomials in G that depend on x_i but not on $x_{i+1}, \ldots, x_n$.* **repeat**
  $f :=$ first(H);
  $H :=$ rest(H);

```
        g := Lcoeff(f, x_i);
        L := Inverse(g, TL);
        TL := [];
        for each pair (h_j, T_j) in L
          if (h_j ≠ 0) then
              add Reduce(f h_j, T_j) to T_j;
              append T_j to the list TL;
          fi od;
      until H is empty;
    od;
    return(TL);
```

The above algorithm can be further optimized as follows:

- In the inner for-loop, if a $T_j \in TL$ already contains a polynomial that depends on x_i (for the current i), then, it is not necessary to perform the body of the for-loop for that T_j as Reduce(fh_j, T_j) will turn out to be zero.

We illustrate the algorithm and the optimization using the previous example. For y, H includes two polynomials, g_2 and g_3; Inverse is invoked on the initial of g_2 with respect to g_1. This leads to a split of the triangular set consisting of g_1 giving two triangular sets $T^{(1)} = \{x^2 - 6x + 9\}$ and $T^{(2)} = \{x^2 - 3x + 2\}$. and $h_1 = 0$ and $h_2 = -3/4x + 11/4$. At the end of the first iteration of the second inner loop, $T^{(1)} = \{x^2 - 6x + 9\}$ and $T^{(2)} = \{x^2 - 3x + 2, y - 1\}$. Now, g_3 reduces to 0 using $T^{(2)}$. The inverse of the initial of g_3 is computed with respect to $T^{(1)}$ (since the initial is 1, the inverse is also 1) which results in the second element $y^2 - 1$ added to $T^{(1)}$. Similarly, for z there are three polynomials g_4, g_5, g_6 in ascending order. The inverse of the initial of g_4 is computed with respect to $T^{(1)}$ as well as $T^{(2)}$. Using $T^{(1)}$, the inverse is $-3/4x + 11/4$ which is multiplied with g_4 to give $z - 1$ and that is added to $T^{(1)}$. For $T^{(2)}$, the inverse of the initial of g_4 does not exist. Polynomials g_5 and g_6 reduce to 0 using $T^{(1)}$. The inverse of the initial of g_5 does not exist with respect to $T^{(2)}$ either, but the inverse of the initial of g_6 with respect to $T^{(2)}$ can be computed thus giving the third element $z^2 + xz + 1$ for $T^{(2)}$. This gives the decomposition in terms of triangular sets.

The correctness of the algorithm is based on a theorem due to Gianni and Kalkbrenner (see Lazard, 1989a; Gianni, 1987; Kalkbrenner, 1987). These ideas are extended to positive dimensional ideals in Lazard (1989b).

4. Characteristic Set Construction

In this section, we discuss Ritt's characteristic set construction. The discussion is based on Ritt's presentation in his book *Differential Algebra* (Ritt, 1950, Chapter 4) and Wu's exposition of Ritt's approach as discussed in Wu (1986a). We first give an informal explanation of Ritt's characteristic set method and then give the technical details. For many applications of the characteristic set construction, the reader can consult Wu (1984, 1986a, 1986b), Chou (1988), Chou and Gao (1990a, 1990b, 1990c, 1990d), Kapur and Mundy (1988) and Kapur and Wan (1990).

Given a system $\mathcal{S}$ of polynomial equations, the characteristic set algorithm transforms $\mathcal{S}$ into *a triangular form* $\mathcal{S}'$ so that the zero set of $\mathcal{S}$ is "roughly equivalent" to the zero

set of $\mathcal{S}'$. A total ordering on variables is assumed. Unlike in Gröbner basis computations, polynomials are treated as univariate polynomials in their highest variable. The primitive operation used in the transformation is that of pseudo-division of a polynomial by another polynomial. The algorithm is similar in flavor to Gaussian elimination for linear equations and computing a lexicographic Gröbner basis.

If the number of equations in $\mathcal{S}$ is less than n, the number of variables, then the variable set $\{x_1, \cdots, x_n\}$ is typically classified into two subsets: *independent* variables (also called *parameters*) and *dependent* variables, and a total ordering on the variables is so chosen that the dependent variables are all higher in the ordering than the independent variables. We denote the independent variables by $u_1, \cdots, u_k$ and dependent variables by $y_1, \cdots, y_l$, and the total ordering is $u_1 \prec \ldots \prec u_k \prec y_1 \prec \ldots \prec y_l$, where $k + l = n$. Choosing independent variables and defining an ordering on dependent variables is similar to choosing an ordering on variables for defining a lexicographic term ordering for computing a Gröbner basis.

To check whether an equation $f = 0$ follows from $\mathcal{S}$, f is pseudo-divided using the polynomials in a triangular form $\mathcal{S}'$ of $\mathcal{S}$. If the remainder of pseudo-division is 0, then $f = 0$ is said to follow from $\mathcal{S}$ under the condition that the initials of polynomials in $\mathcal{S}'$ are not zero. We discuss later why this condition on the initials is needed.

4.1. Ritt-Wu's Theorem

First, we formally define concepts used above in the characteristic set construction. Then, we give the definition of a characteristic set and the main theorem about characteristic sets.

Given a polynomial p, the *highest variable* of p is y_i if $p \in Q[u_1, ..., u_k, y_1, ..., y_i]$ and $p \notin Q[u_1, ..., u_k, y_1, ..., y_{i-1}]$ (there is a similar definition for u_i). The *class* of p is then called i.

A polynomial p is $\geq$ another polynomial q if and only if

1. the highest variable of p, say y_i, is $\succ$ the highest variable of q, say y_j, i.e. the class of p is higher than the class of q, or
2. the class of p = the class of q, and the degree of p in y_i is $\geq$ the degree of q in y_i.

If $p \geq q$ and $q \geq p$, then p and q are said to be *equivalent*, written as $p \cong q$; this means that p and q are of the same class i, and the degree of y_i in p is the same as the degree of y_i in q.

A polynomial p is *reduced with respect to* another polynomial q if

(a) the highest variable, say y_i, of p is $\prec$ the highest variable of q, say y_j (i.e. $p \prec q$), or

(b) $y_i \succ y_j$ and the degree of the y_j in q is $>$ the degree of y_j in p.†

If p is not reduced with respect to q, then p *reduces* to r using q by pseudo-dividing p by q giving r as the remainder of the result of pseudo-division.

A list C of polynomials, $(p_1, \cdots, p_m)$ is called a *chain* if either

† As we shall see later, this definition of reduction will be weakened in order to improve the efficiency of various algorithms for computing a characteristic set.

(i) $m = 1$ and $p_1 \neq 0$, or
(ii) $m > 1$ and the class of p_1 is > 0, and for $j > i$, p_j is of higher class than p_i and reduced with respect to p_i; we thus have $p_1 \prec p_2 \prec \ldots \prec p_m$. (Wu, 1986a).

(A chain is the same as an *ascending set* defined by Wu.)

A polynomial p reduces to p' with respect to a chain $C = \langle p_1, \cdots, p_m \rangle$ if there exist nonnegative integers $i_1, \cdots, i_m$ such that

$$I_1^{i_1} \cdots I_m^{i_m} p = q_1 p_1 + \cdots + q_m p_m + p'$$

where p' is reduced with respect to each of $p_i, 1 \leq i \leq m$. Typically pseudo-division is successively done using p_i's starting with p_m, the polynomial in the highest variable.

Given two chains $C = \langle p_1, \cdots, p_m \rangle$ and $C' = \langle p'_1, \cdots, p'_{m'} \rangle$, $C \succ C'$ if (i) there is a $j \leq m$ as well as $j \leq m'$ such that $p_i \cong p'_i$ for all $i < j$ and $p_j \succ p'_j$, or (ii) $m' > m$ and for $i \leq m$, $p_i \cong p'_i$.

As stated earlier, a set $G = \{g_1, \cdots, g_m\}$ of polynomials is said to be in *triangular form* if and only if $g_1, g_2, \cdots, g_m$ are, respectively, polynomials in $\{u_1, \cdots, u_k, y_1\}$, $\{u_1, \cdots, u_k, y_1, y_2\}$, $\cdots$, $\{u_1, \cdots, u_k, y_1, \cdots, y_m\}$. If $m = l$, then G is said to be fully triangular. It is easy to see that every chain is in triangular form.

4.1.1. Characteristic Set

Ritt was apparently interested in associating characteristic sets only with *prime* ideals. A prime ideal is an ideal with the property that if an element h of the ideal can be factored as $h = h_1 h_2$, then, either h_1 or h_2 must be in the ideal. For a prime ideal Σ, Ritt defined a subset of Σ that forms the lowest chain to be a *characteristic set* of Σ. In chapter 4 in the section *Components of Finite Systems* of his book *Differential Algebra* (Ritt, 1950, p. 95), Ritt describes an algorithm for computing characteristic sets for all the minimal prime ideals containing an ideal $\mathcal{I}$, given a basis for $\mathcal{I}$.

Wu (1986a) altered Ritt's notation somewhat to make it more useful. Wu associated a characteristic set with the zero set of an arbitrary set of polynomials. He called a characteristic set associated with a prime ideal (equivalently, an irreducible zero set) as *irreducible*. A zero set is irreducible if it cannot be expressed as a union of proper algebraic subsets. †

A characteristic set $\{g_1, \cdots, g_l\}$ as defined by Wu could be irreducible or reducible, whereas a characteristic set defined by Ritt is always irreducible. It is interesting to note that in his first paper on geometry theorem-proving (Wu, 1984), Wu defined a characteristic set to be a triangular set in which for $i = 1$ to l,

1 the initial of g_i is a polynomial in the parameters only, and
2 g_i is irreducible over Q_{i-1} where $Q_0 = Q(u_1, \cdots, u_k)$ and $Q_j = Q_{j-1}(\alpha_j)$ is an algebraic extension of Q_{j-1} obtained by adjoining a root α_j of $g_j = 0$ to Q_{j-1}, i.e. $g_j(\alpha_j) = 0$ in Q_j for $1 \leq j < i$.

Wu called such a triangular set as a *privileged basis* associated with a prime ideal. He

† Recall that a zero set is algebraic if it is the zero set of a set of a polynomials.

credited this definition to Gröbner 's book on algebraic geometry (Gröbner , 1949), where it is called a *prime basis*, and to Ritt for "his intimately related concept of a characteristic set" (see Wu, 1984, p. 219). In his subsequent papers, Wu relaxed this requirement on initials on pragmatic grounds, and required a polynomial in a characteristic set to be reduced with respect to other polynomials. In the sequel, we have followed Wu's definitions as they seem to be more useful.

Wu (1986a) attributes the following definition and theorem to Ritt.

DEFINITION 4.1. *Given a finite set Σ of polynomials in $u_1, \ldots, u_k, y_1, \ldots, y_l$, a* characteristic set *Φ of Σ is defined to be either*

- *$\{p_1\}$, where p_1 is a polynomial in $u_1, \ldots, u_k$, or*
- *a chain $\langle p_1, \ldots, p_l \rangle$, where p_1 is a polynomial in $y_1, u_1, \ldots, u_k$ with initial I_1, p_2 is a polynomial in $y_2, y_1, u_1, \ldots, u_k$, with initial I_2, ..., p_l is a polynomial in $y_l, \ldots, y_1, u_1, \ldots, u_k$ with initial I_l, such that*
  - *any zero of Σ is a zero of Φ, and*
  - *any zero of Φ that is not a zero of any of the initials I_i, is a zero of Σ.*

THEOREM 4.1. (RITT) *Given a finite set Σ of polynomials in $y_l, \ldots, y_1, u_1, \ldots, u_k$, there is an algorithm which computes a* characteristic set *Φ of Σ.*

The algorithm discussed in the next subsection involves augmenting Σ with additional polynomials from the ideal generated by Σ obtained through pseudo-division until we have a set Δ such that

1 $\Sigma \subseteq \Delta$,
2 Σ and Δ generate the same ideal, and
3 a minimal chain Φ of Δ pseudo-divides every polynomial in Δ to 0,

then Φ is a characteristic set of Σ as well as Δ. We will call Δ to be a *saturation* of Σ.

Since every polynomial q in Δ pseudo-divides to 0 using Φ, i.e.

$$I_1^{i_1} \cdots I_l^{i_l} q = q_1 p_1 + \cdots + q_m p_l.$$

This implies:

$$\mathsf{Zero}(\Sigma) = \mathsf{Zero}(\Delta) \subseteq \mathsf{Zero}(\Phi),$$

and

$$\mathsf{Zero}(\Sigma) = (\mathsf{Zero}(\Phi) \setminus \mathsf{Zero}(\prod_{i=1}^{l} I_i)) \cup \bigcup_i \mathsf{Zero}(\Sigma \cup \{I_i\}).$$

Using Wu's notation, $\mathsf{Zero}(\Phi/I)$ to stand for $\mathsf{Zero}(\Phi) \setminus \mathsf{Zero}(I)$, the above can be rewritten as:

$$\mathsf{Zero}(\Sigma) = \mathsf{Zero}(\Phi / \prod_{i=1}^{l} I_i) \cup \bigcup_i \mathsf{Zero}(\Sigma \cup \{I_i\}).$$

THEOREM 4.2. *Given a finite set Σ of polynomials, if its characteristic set Φ includes a constant (in the case $l = n$), then Σ does not have a common zero.*

If Φ does not include a constant, it does not mean that Σ has common zeros. For example, consider $\Sigma = \{(x^2-2x+1)=0, (x-1)z-1=0\}$. Under the ordering $x \prec z$, Σ is a characteristic set. The two polynomials do not have a common zero (however, under the ordering $z \prec x$, the characteristic set of Σ includes 1).

The converse of the above theorem holds only for irreducible characteristic sets. This is discussed in a later subsection.

4.2. Algorithms for Computing a Characteristic Set

A characteristic set Φ is computed from a set Σ of polynomials by successively adjoining Σ with remainder polynomials obtained by pseudo-division. Starting with $\Sigma_0 = \Sigma$, we extract a minimal chain (called a *basic set* by Wu) from Σ_i, and compute non-zero remainders of polynomials in Σ_i with respect to the minimal chain.† If this remainder set is nonempty, we adjoin it to Σ_i to obtain Σ_{i+1} and repeat the computation until we have Σ_n such that every polynomial in Σ_n pseudo-divides to 0 with respect to its minimal chain. The set Σ_n is a saturation of Σ and the minimal chain of Σ_n is a characteristic set of Σ_n as well as Σ. This algorithm is given in chapter 4 in the section *Components of Finite Systems* in Ritt's *Differential Algebra* (Ritt, 1950, p. 95). The above construction terminates since the minimal chain of Σ_i is $\succ$ the minimal chain of Σ_{i+1} and the ordering on chains is well-founded (the maximum size of a chain is l, and the degree of at least one polynomial in Σ_{i+1} is lower than the degree of the corresponding polynomial in the same variable in Σ_i).

In the process of computing a characteristic set from Σ, if an element of the coefficient field (a rational number if $l = n$ or a rational function in $Q(u_1, \cdots, u_k)$) is generated as a remainder, this implies that Σ does not have a solution, or is *inconsistent*.

Below, we give two algorithms for computing characteristic sets as implemented in GeoMeter for geometry theorem proving (Connolly *et al.*, 1989; Kapur and Wan, 1990). These algorithms differ on the order in which remainders are computed and processed. The first one is Ritt's algorithm.‡

Char-Set-Breadth-first$(H, \prec)$

Input: A set of polynomials H and a variable ordering $\prec$.
Output: A characteristic set for H under the variable ordering $\prec$.
Functions used:
Basic-set$(E, \prec)$: Described below.
pseudo-divide-reduction$(p, B, \prec)$: successively reduces (pseudo-divides) the polynomial p with respect to the polynomials in the basic set B starting with the largest polynomial with respect to $\prec$.

$E := \emptyset$; $R := H$;
while $R \neq \emptyset$ do
$E := E \cup R$;
$B :=$Basic-set$(E, \prec)$;
$R := \{q \mid q =$pseudo-divide-reduction$(p, B, \prec), q \neq 0, p \in E \setminus B\}$;
od;
Return B;

Basic-set$(S, \prec)$

Input: A set of polynomials S and a variable ordering $\prec$.
Output: A basic set contained in S with respect to $\prec$.

$B := \emptyset$; $T := S$;
while $T \neq \emptyset$ do
$p :=$ a smallest polynomial in T;% "smallest" with respect to $\prec$.
$B := B \cup \{p\}$;
$T := \{q \mid q \in T \setminus \{p\}, q$ is reduced with respect to $p\}$;
od;
Return B;

The correctness of the above algorithm is based on the fact that q, the remainder from pseudo-division of p by a basic set B, is in the ideal generated by $B \cup \{p\}$, which is a subideal of the ideal generated by H. An invariant of the while loop in the procedure Char-Set-Breadth-first is that the ideal generated by E is the same as the ideal generated by H which implies that the zero sets of H and E are the same. If B is the result of Char-Set-Breadth-first, then every polynomial in E of the last iteration (which is a saturation of Σ as defined in the last subsection) pseudo-divides to 0 using B. Consequently, the zero set of B includes the zero set of H as a subset; in particular, the zero set of B minus the zero set of the initials of polynomials in B is the zero set of H.

The breadth-first strategy is quite inefficient for computing a characteristic set for detecting inconsistency. In each stage, all possible remainders are computed first before the next basic set is computed. It is much better to use a depth-first strategy given below. In this strategy, the basic set is updated each time a new remainder is obtained; in this way, remainders of lower classes can be obtained more quickly.

Char-Set-Depth-first$(H, \prec)$

Input: A set of polynomials H and a variable ordering $\prec$.
Output: A characteristic set for H with respect to $\prec$.
Functions used:
Basic-set$(E, \prec)$: same as before.
pseudo-divide-reduction$(p, B, \prec)$: same as before.
Update-Basic-set$(p, B, \prec)$: described below.

$E := H$;
$B :=$Basic-set$(E, \prec)$;
$ER := E \setminus B$;
while ($ER \neq \emptyset$ and $1 \notin B$) do
$S := E \setminus B$;
$ER := \emptyset$;
for each p (starting from the smallest polynomial) $\in S$ do
$ER := ER \cup$ Update-Basic-set$(p, B, \prec)$ od;

† Note that a minimal chain extracted from a set need not be unique.

‡ Recall that by the smallest polynomial in a set S, we mean a polynomial p whose highest variable, say y_i, is not greater than the highest variable of any other polynomial in S and furthermore, among the polynomials with y_i as the highest variable, the degree of p in y_i is the least.

$E := E \cup ER$;
od;
Return B;

Update-Basic-set$(p, B, \prec)$

Input: A polynomial p, a basic set B with respect to a variable ordering $\prec$.
Output: The set of remainder polynomials generated when p is added to B. As a side effect, B is a new basic set.

r :=pseudo-divide-reduction$(p, B, \prec)$;
if $r = 0$ then Return $\emptyset$
else if r is a constant then
$B := \{1\}$;
Return $\emptyset$;
fi;
$R := \{r\}$;
$T := \{q \mid q \in B, q$ is not reduced with respect to $r\}$;
$B := B \cup \{r\} \setminus T$;
for each q (starting from the smallest polynomial) $\in T$ do
$R := R \cup$ Update-basic-set$(q, B, \prec)$ od;
Return R;

The breadth-first and depth-first strategies do not always produce the same result. For the breadth-first strategy, in generating remainders during each stage, one remainder obtained does not in any way affect the computation of the next remainder. For the depth-first strategy however, since one remainder may affect what the next remainder might be, it is possible that some remainders computed in the breadth-first strategy may never be computed in the depth-first strategy.

4.3. Proving Conjectures from a System of Equations

A direct way to check whether an equation $c = 0$ follows (under certain conditions) from a system $\mathcal{S}$ of equations is to compute a characteristic set $\Phi = \{p_1, \cdots, p_l\}$ from $\mathcal{S}$ and check whether c pseudo-divides to 0 with respect to Φ. If c has a zero remainder with respect to Φ, then the equation $c = 0$ follows from Φ under the conditions that none of the initials used to multiply c is 0. The algebraic relation between the conjecture c and the polynomials in Φ can be expressed as:

$$I_1^{i_1} \cdots I_l^{i_l}\, c = q_1 p_1 + \cdots + q_l p_l,$$

where I_j is the initial of $p_j, j = 1, \cdots, l$.

This approach is used by Wu and Chou for geometry theorem proving. A characteristic set is computed from the hypotheses of a geometry problem; then a conjecture is pseudo-divided by the characteristic set to check whether the remainder is 0. If the remainder is 0, the conjecture is said to be generically valid from the hypotheses.

A refutational way to check whether $c = 0$ follows from $\mathcal{S}$ is to compute a characteristic set of $S \cup \{cz - 1 = 0\}$, where z is a new variable. This approach has also been used for geometry theorem proving and discussed in Kapur and Wan (1990).

4.4. Wu's Structure Theorem

If a polynomial c representing a geometric conjecture does not pseudo-divide to 0 with respect to a characteristic set Φ, it cannot always be said that $c = 0$ does not follow from Φ or, for that matter, from Σ from which Φ is computed. Some additional properties need to be checked. We need to make sure that the characteristic set is *irreducible*.

Definition 4.2. *A characteristic set $\Phi = \{p_1, \cdots, p_l\}$ is* irreducible *over $Q[u_1, ..., u_k, y_1, ..., y_l]$ if for $i = 1$ to l, p_i is irreducible over Q_{i-1} where $Q_0 = Q(u_1, \cdots, u_k)$ and $Q_j = Q_{j-1}(\alpha_j)$ is an algebraic extension of Q_{j-1}, obtained by adjoining a root α_j of $p_j = 0$ to Q_{j-1}, i.e. $p_j(\alpha_j) = 0$ in Q_j for $1 \leq i < j$.*

Similarly, if Σ does not have a solution, either a characteristic set Φ of Σ includes an element $Q(u_1, \cdots, u_k)$, or Φ is reducible and each of the irreducible characteristic sets includes an element of $Q(u_1, \cdots, u_k)$.

Theorem 4.3. *Given a finite set Σ of polynomials, if (i) its characteristic set Φ is irreducible, (ii) Φ does not include a constant, and (iii) the initials of the polynomials in Φ do not have a common zero with Φ, then Σ has a common zero.*

To deal with a reducible characteristic set, Ritt and Wu advocated the use of factorization over algebraic extensions of $Q(u_1, \cdots, u_k)$, which is an expensive operation. With reducibility check by factorization over extension field, the check for consistency is complete using the characteristic set method.†

In the case that any of the polynomials in a characteristic set can be factored, there is a branch for each irreducible factor as the zeros of p_j are the union of the zeros of its irreducible factors. Suppose we compute a characteristic set $\Phi = \{p_1, \cdots, p_l\}$ from Σ such that for $i > 0$, $p_1, \cdots, p_i$ are irreducible over $Q_0, \cdots, Q_{i-1}$, respectively, but p_{i+1} can be factored over Q_i. It can be assumed that

$$g\ \ p_{i+1} = p^1{}_{i+1} \cdots p^j{}_{i+1},$$

where g is in $Q[u_1, \cdots, u_k, y_1, \cdots, y_i]$, and $p^1{}_{i+1} \cdots p^j{}_{i+1} \in Q[u_1, \cdots, u_k, y_1, \cdots, y_i, y_{i+1}]$ and these polynomials are reduced with respect to $p_1, \cdots, p_i$. Wu (1986a) proved that

$$\mathsf{Zero}(\Sigma) = \mathsf{Zero}(\Sigma^{11}) \cup \cdots \cup \mathsf{Zero}(\Sigma^{1j}) \cup \mathsf{Zero}(\Sigma^{21}) \cup \cdots \cup \mathsf{Zero}(\Sigma^{2i}),$$

where $\Sigma^{1h} = \Sigma \cup \{p^h{}_{i+1}\}, 1 \leq h \leq j$, and $\Sigma^{2h} = \Sigma \cup \{I_h\}$, where I_h is the initial of $p_h, 1 \leq h \leq i$. So characteristic sets are instead computed from new polynomials sets to give a system of characteristic sets. (To make full use of the intermediate computations already performed, Σ above can be replaced by a saturation of Σ.) The final result of this decomposition is a system of irreducible characteristic sets:

$$\mathsf{Zero}(\Sigma) = \bigcup_i \mathsf{Zero}(\Phi_i / J_i),$$

† We would like to remind the reader that we are only considering zeros over the complex numbers whereas, strictly speaking, for geometry theorem-proving one must check whether real zeros of the hypotheses are contained in the real zeros of the conclusion.

where Φ_i is an irreducible characteristic set and J_i is the product of the initials of all the polynomials in Φ_i. For further details, the reader should consult Wu (1986a).

Example: Consider the polynomial set

$$\begin{aligned} H = \{&13z^3 + 12x^2 - 23xz + 16z^2 - 49x + 39z + 40,\\ &13xz^2 - 12x^2 + 10xz - 29z^2 + 49x - 26z - 53,\\ &13x^3 - 88x^2 + 4xz + 4z^2 + 173x - 94,\\ &yz - y - z + 1, x^2z - x^2 - 3xz + 3x + 2z - 2,\\ &y^2 - 1, yx^2 - 6yx + 9y - x^2 + 6x - 9\}. \end{aligned}$$

The following characteristic set can be computed from it using the ordering $x \prec y \prec z$.

$$\begin{aligned} C = \{f_1 &= (x^2 - 3x + 2)z - x^2 + 3x - 2,\\ f_2 &= (x^2 - 6x + 9)y - x^2 + 6x - 9,\\ f_3 &= x^6 - 12x^5 + 58x^4 - 144x^3 + 193x^2 - 132x + 36\}. \end{aligned}$$

The polynomial f_3 in C can be factored as $(x-1)^2(x-2)^2(x-3)^2$, giving the following characteristic sets which can be further decomposed:

$$\begin{aligned} C_1 &= \{z^2 + z + 1, y - 1, x - 1\},\\ C_2 &= \{z^2 + 2z + 1, y - 1, x - 2\},\\ C_3 &= \{z - 1, y^2 - 1, x - 3\}. \end{aligned}$$

For a system of polynomials in which no variable has degree more than 2, such as in problems arising in Euclidean plane geometry, Chou developed an algorithm to check irreducibility and perform factorization in the case of reducibility to generate irreducible characteristic sets; the reader may consult Chou (1988).

4.5. Implementation

To our knowledge, none of the computer algebra systems supports an implementation of characteristic set construction. Characteristic set algorithms can be, however, easily implemented in any computer algebra system that supports an efficient implementation of pseudo-division. Wu and Chou have reported implementations using which they got extremely impressive results for plane geometry problems. GeoMeter has an implementation of the algorithms reported earlier in this section. Considerable work is needed to further improve characteristic set algorithms.

Based on their extensive experience in using the characteristic set construction, Wu and Chou have proposed modifications in the definitions of reduction as well as a polynomial being reduced with respect to a chain. In Nguyen *et al.* (1991), a lazy approach to the evaluation of the coefficients of a polynomial represented in recursive form is discussed.

As in Gröbner basis computations, the performance of algorithms to compute a characteristic set is sensitive to the classification of variables into dependent and independent variables, as well as the order of dependent variables. Several heuristics have been proposed for deducing a good ordering from a problem formulation in the application of geometry theorem proving.

5. Conclusion

When we started writing this introductory article, we had planned to discuss many additional topics. As evident, we will need a lot more space. We had to omit many issues, in particular a detailed discussion of the applications pointing out limitations of these approaches as they are currently practiced. We are, however, quite optimistic about the potential use of these methods.

One of the areas that we feel should be investigated is the apparent close relationship between the three approaches in the zero-dimensional case. We believe a lot can be learned from the study of a relationship among various approaches. In particular, it might be possible to achieve better bounds on the worst case complexity of Gröbner basis and characteristic set computations. We may also have a better idea about why these methods work better for sparse systems. It might be possible to apply results about redundant computations in Gröbner basis theory to resultant computation. On the other hand, identification of a close relationship between these approaches might give a clue about how techniques that work well for resultant computations can be used in Gröbner basis and characteristic set methods. Most importantly, there is a need for studying techniques for efficiently implementing these algorithms for polynomials arising in various application domains.

References

S.S. Abhyankar (1976), "Historical ramblings in algebraic geometry and related algebra", *American Math. Monthly*, 83(6), 409–448.

D.S. Arnon, G.E. Collins and S. McCallum (1984), "Cylindrical algebraic decomposition I: the basic algorithm, II: an adjacency algorithm for the plane", *SIAM J. Comput.*, 13, 865–889.

D. Bayer and M. Stillman (1989), *Macaulay User's Manual*, Cornell University, Ithaca, NY.

J.S. Brown and J.F. Traub (1971), "On Euclid's algorithm and the theory of subresultants", *J. ACM*, 18(4), 505–514.

B. Buchberger (1965), *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes Nach Einem Nulldimensionalen Polynomideal*, Ph.D. Thesis (in German), Universitat Innsbruck, Austria.

B. Buchberger (1976), "A theoretical basis for the reduction of polynomials to canonical form", *ACM SIGSAM Bull.*, 10(3), 19–29.

B. Buchberger (1983), "A note on the complexity of constructing Gröbner bases", *Proc. EUROCAL '83*, London, Lecture Notes in Comput. Sci. 162, Springer-Verlag, NY, 137–145.

B. Buchberger (1985), "Gröbner bases: an algorithmic method in polynomial ideal theory", *Multidimensional Systems Theory*, N.K. Bose, ed., D. Reidel Publishing Co., Netherlands, 184–232.

J.F. Canny (1988), *The Complexity of Robot Motion Planning*, ACM Doctoral Dissertation Series, MIT Press, Cambridge, MA.

J.F. Canny (1990), "Generalized characteristic polynomials", *J. Symbolic Computation*, 9, 241–250.

J.F. Canny, E. Kaltofen and Y.N. Lakshman (1989), "Solving systems of nonlinear polynomial equations faster", *Proc. Int. Symp. Symbolic Algebraic Computation (ISSAC-89)*, Portland, OR, 121–128.

A. Cayley (1865), "On the theory of elimination", *Cambridge and Dublin Math. J.*, III, 210–270.

E. Chionh (1990), *Base Points, Resultants, and the Implicit Representation of Rational Surfaces*, Ph. D. Thesis, Dept. of Comput. Sci., University of Waterloo, Waterloo, Canada.

S.-C. Chou (1988), *Mechanical Geometry Theorem Proving*, D. Reidel Publishing Co., Netherlands.

S.-C. Chou and X.-S. Gao (1990a), "Ritt-Wu's decomposition algorithm and geometry theorem proving", *Proc. 10th Int. Conf. Automated Deduction (CADE-10)*, Kaiserslautern, Germany, Lecture Notes in Comput. Sci., Springer-Verlag, NY, 207–220.

S.-C. Chou and X.-S. Gao (1990b), *On the Parameterization of Algebraic Curves*, Technical Report 90-18, Dept. of Comput. Sci., University of Texas, Austin.

S.-C. Chou and X.-S. Gao (1990c), *On the Normal Parameterization of Curves and Surfaces*, Technical Report 90-19, Dept. of Comput. Sci., University of Texas, Austin.

S.-C. Chou and X.-S. Gao (1990d), *Independent Parameters, Inversions and Proper Parameterization*,

REASONING ABOUT NUMBERS IN TECTON

– Preliminary Version —

Deepak Kapur*and Xumin Nie†
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, New York 12222, U.S.A.

Abstract

We discuss algorithms and heuristics for reasoning about numbers - rationals, integers, and naturals, and the associated operations $+$, $=$, $\geq$, in *Tecton*, a specification and verification system currently under development. A main objective is to make Tecton more efficient for tasks of reasoning about specification and programs that involve numbers. We extend Fourier's algorithm for deciding satisfiability of linear constraints over the rationals to linear constraints over the integers. This algorithm serves as a building block for a complete decision procedure for universally quantified Presburger arithmetic, in which a special emphasis is placed on deducing equalities and using them as rewrite rules for simplification and elimination. We discuss how this decision procedure has been integrated with definitions and properties of interpreted function symbols specified as terminating rewrite rules in the Tecton system.

1 Introduction

In the summer of 1991, we implemented a set of algorithms and heuristics to automatically reason about numbers – rationals, naturals and integers, into a verification and specification system *Tecton* [11] which is being developed on top of our theorem prover *Rewrite Rule Laboratory* (*RRL*) [15]. This paper is a belated report on these algorithms and our approach for implementing these algorithms into a rewrite rule based theorem prover. In particular, we extend Fourier's method for linear inequalities over the rationals, as explained in [20], to consider linear inequalities over the integers, and show its completeness as a decision procedure for satisfiability problem of linear inequalities over the integers.[1] The extended algorithm is used as a building block for a decision procedure for universally quantified Presburger arithmetic with uninterpreted symbols. A particular emphasis is placed on deducing equalities so that they can be used as rewrite rules for simplification and for detecting unsatisfiability.

We discuss how the decision procedure for Presburger arithmetic has been integrated with terminating (conditional) rewrite rules defining interpreted function symbols and their inductive properties. In particular, we discuss the interaction between the decision procedure and conditional

*Partially supported by National Science Foundation Grant No. CCR–9303394 and United States Air Force Office of Scientific Research Grant No. AFOSR-91-0361.

†Current address: Department of Computer Science, The Wichita State University, Wichita, Kansas 67208.

[1] After writing an earlier draft of this paper, we recently learned (May 1994) that Williams had already reported such an extension in [28]. A careful study of Williams's paper and Cooper's paper [4] would reveal that Williams was rediscovering Cooper's results and Presburger's procedure as reported in [6]. This is not very surprising because Fourier's method has been rediscovered many times in the mathematics literature.

rewriting while normalizing a formula to prove its validity in the Tecton system. Many examples are given to illustrate different decision procedures and their combinations.

A main objective in designing the Tecton system is to support construction of large complex proofs, such as those that typically are necessary in reasoning about specification and descriptions of generic components useful in software and hardware design [11]. Often, one needs to reason about natural numbers and integers, particularly in the context of linear data structures such as arrays and sequences, where indices are needed, and totally-ordered enumerative sets such as natural numbers and integers are natural choices for indices. Reasoning about numbers is also useful in reasoning about measures including size, length, depth, etc, defined on data structures. Using Presburger arithmetic, we have recently developed a method for checking completeness of function definitions defined on numbers using $0, s, +, \geq$ as terminating rewrite rules [10]. Using this method, it is possible to prove completeness of the function definitions used in an inductive proof of unique prime factorization theorem done on *RRL* using the cover set method for automating induction [31]; this method can also be used to prove the completeness of cover sets. In another paper, we have developed methods using Presburger arithmetic for generating induction schemes and other heuristics for enhancing the inductive theorem proving capabilities of *RRL* and Tecton [14].

Algorithms for deciding properties of numbers were extensively discussed in the literature in the mid-70's and early 80's when there was considerable interest in program verification. A particular focus has been on a decision procedure for a subclass of *Presburger arithmetic* (in particular, universally quantified theory of Presburger arithmetic). Cooper [4] gave an algorithm which improved upon on his previous algorithms as well as an algorithm given in logic text-books, e.g. [6]. Shostak [25] proposed a method for proving formulas in Presburger arithmetic using a procedure for solving linear inequalities over the rationals using the *sup-inf* method proposed by Woody Bledsoe [1, 2]. Subsequently, Shostak [26] gave an algorithm for Presburger arithmetic with uninterpreted function symbols, formulas handled by Cooper's algorithm; this algorithm was based on integer linear programming. Nelson and Oppen [23] proposed an elegant framework of cooperating decision procedures with a simplex based algorithm for solving linear inequalities over the rationals as one of the decision procedures in the Stanford Pascal Verifier. This framework was adopted in the Eves system and has been implemented in its theorem prover [5].

In the mid-80's, Boyer and Moore [3] incorporated a rational-based procedure in their theorem prover for automating proofs by induction.[2] They extensively discussed their implementation, in particular representations used for building a data base for rewriting using contexts, as well as specific techniques used to manage the interaction between the rewriter and the decision procedure.

Lassez and his group have been investigating efficient methods for linear constraints over the rationals in the context of constraint logic programming [19, 20]. In particular, they have revived interest in Fourier's algorithm for checking the satisfiability of linear constraints over the reals and the rationals. In [20], Lassez and Maher discussed Fourier's algorithm in detail and showed how it could be used to deduce implicit equalities in inequality constraints. The discussion of Fourier's algorithm below is based on [20].

We have recently learned of another interesting application of reasoning methods over numbers in the area of data dependency analysis for compilers for supercomputers. Presburger arithmetic has been used to study dependency among reads and writes of array elements in loop programs. For more details, the reader can consult [24].

2 Algorithms for Reasoning about Linear Inequalities

The main algorithm for solving linear inequalities that we implemented is that of Fourier [20], which as Boyer and Moore remarked [3], "is just a formalization of the high school idea of "cross multiplying and adding" equalities to eliminate variables." In this section, we first review Fourier's algorithm

[2] We believe that Hodes' method [7] implemented in Boyer and Moore's prover is essentially Fourier's algorithm (perhaps with minor variations).

for checking satisfiability of linear inequalities over the rationals, and then describe how Fourier's algorithm can be extended to derive *implicit equalities* [20]. Later, we discuss our extension of Fourier's algorithm to linear integer inequalities. This extension serves as a basis for a decision procedure for universally quantified Presburger arithmetic.

Henceforth, by *Presburger arithmetic*, we mean the universally quantified theory built using natural number constants (or integer constants whenever that is evident from the context), variables over the natural numbers (or integers), addition (and subtraction), the usual arithmetical relations (such as $=, \geq, >, \leq, <$), and the usual first order logical connectives; no other symbols are assumed. By a *Presburger formula*, we mean a quantifier-free formula in Presburger arithmetic. Later we consider extended Presburger arithmetic in which function symbols are introduced; if a function symbol is not constrained using any additional properties, it is considered to be *uninterpreted*; otherwise it is considered to be *interpreted.* Following [20], we now introduce some terminology.

Definition 2.1 An (integral) *inequality* is an expression of the form $\sum_{j=1}^{n} a_j x_j \leq b$ where a_j and b are integers. Let C be an inequality $\sum_{j=1}^{n} a_j x_j \leq b$, we use $C^=$ to denote the equation $\sum_{j=1}^{n} a_j x_j = b$. For $\alpha > 0$, αC is the inequality obtained by multiplying C by α, i.e. $\sum_{j=1}^{n} \alpha a_j x_j \leq \alpha b_k$. The sum $C_1 + C_2$ of inequalities C_1 ($\sum_{j=1}^{n} a_{1j} x_j \leq b_1$) and C_2 ($\sum_{j=1}^{n} a_{2j} x_j \leq b_2$) is the inequality $\sum_{j=1}^{n} (a_{1j} + a_{2j}) x_j \leq (b_1 + b_2)$. If an inequality H can be expressed as $H = \sum_{k=1}^{m} \alpha_k C_k$ and $\alpha_k \geq 0$ $(1 \leq k \leq m)$, then H is called as a *non-negative linear combination* of inequalities $C_1, ..., C_m$. When each α_k is strictly positive, then H is a positive linear combination.

We define in the usual way the concepts of satisfiability of formulas over a domain and of logical consequence ($\models$). We define an *implicit equality* in a domain in a set $\mathcal{P}$ of inequalities to be an equation $C^=$ where C is an inequality in $\mathcal{P}$ and $\mathcal{P} \models C^=$ over the domain. A domain could be rationals, integers or naturals. Often, the domain would be evident from the context.

2.1 Fourier's algorithm for rationals

Let $\mathcal{P}$ be a set of inequalities, and let x be a variable appearing in it. Let C_i be an inequality in $\mathcal{P}$ with a negative coefficient $-c_i$ of x, $c_i > 0$, and C_j be an inequality in P with a positive coefficient c_j of x, $c_j > 0$. Then $c_i C_j + c_j C_i$ is an inequality which does not contain x. Such an operation is called *elimination*, since it eliminates an occurrence of the variable x from C_i, C_j. Every common solution of C_i, C_j is also a solution of $c_i C_j + c_j C_i$; further, every solution of $c_i C_j + c_j C_i$ can be extended to get a common solution of C_i, C_j. Because of such relationship among the solution of the derived inequality to C_i, C_j, i.e. a common solution of C_i, C_j can be projected by forgetting the value of x to get a solution of $c_i C_j + c_j C_i$, this operation is also called *projection.*

Given $\mathcal{P}$, we can derive from $\mathcal{P}$, all inequalities that do not have any occurrence of x by pairing inequalities containing x with negative coefficients with inequalities containing x with positive coefficients. The set of all such inequalities, together with the inequalities of $\mathcal{P}$ which do not contain x, is the result of a *macro Fourier step* eliminating x completely from $\mathcal{P}$. Typically many redundant inequalities are generated which can be detected and deleted using the test that $C \leq c$ implies $\alpha C \leq d$ for any $\alpha > 0, \alpha c \leq d$.

If x does not have a positive coefficient in any inequality of $\mathcal{P}$ or x does not have a negative coefficient in any inequality of $\mathcal{P}$, then the Fourier step eliminating x from $\mathcal{P}$ simply deletes all inequalities containing x. This is because an appropriate value of x can be constructed to satisfy all such inequalities. Such a Fourier step is called *trivial.*

Fourier's algorithm consists of repeatedly performing macro Fourier steps, eliminating one variable at a time, until either the set contains a contradictory inequality $0 \leq c$ where c is negative, implying that the original set of inequalities is unsatisfiable, or all variables are eliminated without yielding any contradictory inequality, thus implying that the original set of inequalities is satisfiable.

It is easy to see that any inequality C produced in Fourier's algorithm is a non-negative linear combination of inequalities $C_1, \ldots, C_m \in \mathcal{P}$. The following theorems from Käufl ([16]) and Lassez

and Maher ([20]) serve as a basis for deriving implicit equalities from a set $\mathcal{P}$ of inequalities.

Theorem 2.2 *If $\mathcal{P}$ implies an equality e, there is a finite subset $\{C_1, ..., C_k\}$ of $\mathcal{P}$ such that $\mathcal{P} \models e$ iff $\{C_1^=, ...C_k^=\} \models e$ ([16]).*

Theorem 2.3 *If $\sum_{k=1}^{m} \alpha_k C_k = 0 \leq 0, C_k \in \mathcal{P}, \alpha_k > 0, 1 \leq k \leq m$, then each $C_j, 1 \leq j \leq m$, is an implicit equality ([20]).*

Theorem 2.4 *An inequality C_k in a set of consistent inequalities $\mathcal{P}$ is an implicit equality in rationals iff Fourier's algorithm produces an inequality $0 \leq 0$ using C_k ([20]).*

2.2 Extending Fourier's algorithm to integers

Fourier's algorithm can be extended to form a complete decision procedure for checking satisfiability of linear inequalities over the integers. It is easy to see that if a set of inequalities is unsatisfiable over the rationals, it is unsatisfiable over integers. It is however possible for a set of inequalities to be unsatisfiable over the integers, but it may have a satisfying assignment over the rationals. A simple example is that of $\{2x \leq 1, -2x \leq -1\}$.

In the extended Fourier's algorithm the basic Fourier step is the same, i.e. given an inequality C_i in $\mathcal{P}$ with a negative integral coefficient $-c_i$ of x, $c_i > 0$, and C_j, another inequality in $\mathcal{P}$, with a positive integral coefficient c_j of x, $c_j > 0$, $d_i C_j + d_j C_i$ is an inequality wich does not contain x, where $d_i = \frac{lcm(c_i,c_j)}{c_j}, d_j = \frac{lcm(c_i,c_j)}{c_i}$. It would have been okay to multiply C_j by c_i and C_i by c_j, but one is likely to see smaller integers if we remove the gcd of c_i, c_j from the multipliers.

If during the algorithm, while eliminating a variable x, an inequality of the form $0 \leq c_i$, where c_i is positive is generated, then in contrast to the rational case, there is no guarantee any more that inequalities leading to this inequality can be satisfied. Unlike in the rational case, it may not be possible to extend an integer solution of the derived inequalities (which does not have any assignment for x) to an integer solution of the original inequalities, as the following simple example illustrates. Let $C_i = 4x \leq 7, C_j = -6x \leq -8$. We deduce the inequality: $3 * 4x - 2 * 6x \leq 3 * 7 - 2 * 8$, which is equivalent to $0 \leq 5$. The two inequalities being considered are: $12x \leq 21$ and $-12x \leq -16$, i.e. $16 \leq 12x \leq 21$. But there is no x that lies in this interval. So if after eliminating all the variables, no contradictory inequality is generated, there is no guarantee yet that the inequalities are satisfiable. It becomes necessary to check whether intervals for variables are large enough to produce a satisfying assignment. For unsatisfiability, an additional constraint can be added: if C_i is equal to $A \leq c_i x$, C_j is equal to $c_j x \leq B$, add $d_i B < d_j A + lcm(c_i, c_j) - d_i - d_j$.

We now give the details with examples.

2.2.1 Preprocessing equalities and inequalities

Equalities are processed first. For each equality, it is checked whether the gcd of the coefficients of the variables divides the constant in the equality; if it does not, then the equality cannot be satisfied. In an equality, a variable with a unit coefficient is preferred for elimination. A system of equalities is solved using the algorithm given in [17]. Variables solved for can be eliminated from the inequalities.

A constraint $c_1 x_1 + \cdots c_l x_l \leq d$, where d is an integer, can be simplified by dividing it by $g = gcd(c_1, \cdots, c_l)$ to give $c_1' x_1 + \cdots c_l' x_l \leq d'$, where $c_i' = \frac{c_i}{g}, d' = \lfloor \frac{d}{f} \rfloor$.

Below, we assume that equalities have been processed and inequalities have been simplified. Any new equality or inequality generated during the extended Fourier procedure is preprocessed in the same way.

2.2.2 Processing inequalities

As in [20], given a set $\mathcal{P}$ of integer inequalities, classify all inequalities based on whether x appears with a negative coefficient, positive coefficient, or does not appear at all.

$$\begin{array}{ll} l_i \leq c_i x & i = 1, 2, \ldots, p, \\ d_j x \leq r_j & j = 1, 2, \ldots, q, \\ g_l \leq 0 & l = 1, 2, \ldots, s. \end{array}$$

Let $\mathcal{P}'$ be the set of inequalities deduced from $\mathcal{P}$ by eliminating x by pairwise resolution of an inequality with a negative coefficient of x and an inequality with a positive coefficient of x.[3] Assuming that $\mathcal{P}'$ is satisfiable and σ is a satisfying assignment for $\mathcal{P}'$, it is possible to extend this assignment to include a value of x to have a satisfying assignment for $\mathcal{P}$ provided the value lies in the intervals specified by $\mathcal{P}$. This check is delayed until the end; however, it is necessary to keep track of intervals for possible values that x can take.

Let $L = lcm(c_1, \ldots, c_p, d_1, \ldots, d_q)$. Then inequalities in $\mathcal{P}'$ imply these inequalities[4]

$$\begin{array}{ll} \frac{L}{c_i} l_i \leq \frac{L}{d_j} r_j & i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q, \\ g_l \leq 0 & l = 1, 2, \ldots, s. \end{array}$$

If $\mathcal{P}'$ has a solution σ, then σ can be extended to include a value of x that would be a solution of $\mathcal{P}$ provided an integral multiple of L lies in the interval

$$\left[max\{\sigma(\frac{L}{c_i} l_i)\}, min\{\sigma(\frac{L}{d_j} r_j)\} \right], i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q.$$

Let a finite set of inequalities, $\{\frac{L}{c_i} l_i \leq L\ x \leq \frac{L}{d_j} r_j,\ \ i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q\}$ be called the *defining constraint* on x, denoted as Def_x. If for every solution σ of $\mathcal{P}'$, no integral multiple of L lies in the interval defined by instantiating the defining constraint of x by σ, then $\mathcal{P}$ is unsatisfiable even if $\mathcal{P}'$ is satisfiable. For checking satisfiability, it then becomes necessary to generate an assignment at the end of the algorithm and extend it by making sure that every variable satisfies its defining constraint. While generating a satisfying assignment, we may find that it is not possible to extend a partial assignment further, thus detecting unsatisfiability.

If a Fourier step involves deleting inequality constraints because a variable x being eliminated appears only with negative coefficients (or positive coefficients), say, $l_i \leq c_i x,\ c_i > 0,\ 1 \leq i \leq p$ ($d_j x \leq r_j,\ d_j > 0,\ 1 \leq j \leq q$, respectively), then $max\{\frac{L}{c_i} l_i \mid 1 \leq i \leq p\} \leq Lx$ ($Lx \leq min\{\frac{L}{d_j} r_j \mid 1 \leq i \leq p\}$, respectively) serves as the defining constraint for x.

Theorem 2.5 *Let $\mathcal{P}$ be a finite set of inequalities as defined above. Let $\mathcal{P}'$ be the inequalities generated from $\mathcal{P}$ after eliminating x. Let Def_x, the defining constraint for x, be*

$$\frac{L}{c_i} l_i \leq L\ x \leq \frac{L}{d_j} r_j,\ \ i = 1, 2, \ldots, p;\ j = 1, 2, \ldots, q.$$

$\mathcal{P}$ is satisfiable if and only if $\mathcal{P}'$ is satisfiable using an assignment σ and there exists a multiple of L in the interval $\left[max\{\sigma(\frac{L}{c_i} l_i)\},\ min\{\sigma(\frac{L}{d_j} r_j)\} \right]$.

[3] For simplicity, we assume in the discussion below that exactly one variable x is eliminated in a macro Fourier step. The case of more than one variables getting eliminated while eliminating one variable can be handled in a similar manner.

[4] Because of pairwise resolution, inequalities generated in $\mathcal{P}'$ are likely to have smaller numbers appearing as coefficients. Cooper [4] instead multiplied every inequality with L/c_i, where c_i is the absolute value of the nonzero coefficient of x in an inequality C_i, replaced Lx by a new variable x' and required x' to be divisible by L.

Example 2.6 Let $\mathcal{P} = \{4x + 4y \leq 3, 2x + 2y \geq 1\}$. After a Fourier step to eliminate x, one gets $2 - 4y \leq 3 - 4y$, where $L = lcm(2,4) = 4$, which gives the inequality $2 \leq 3$, which is satisfiable. The defining constraint for x is $2 - 4y \leq 4x \leq 3 - 4y$, and there is no integral multiple of 4 in the interval $[2 - 4y, 3 - 4y]$. Hence $\mathcal{P}$ is unsatisfiable.

In the above example, it could be deduced quickly that no value of x can satisfy the inequalities in $\mathcal{P}$. In general, many macro Fourier steps may have to be done, and then solutions tried to get to an interval using which a solution may not be extensible.

There are many refinements and optimizations for checking satisfiability. For example, it is not necessary to search through all possible assignments of a variable in the whole interval of values given by its defining constraint. We will not discuss them here for lack of space. Neither will we discuss the more general case of a linear combination of variables possibly getting eliminated in an elimination step and its implications for checking satisfiability. An interested reader may consult [13]. We would however like to mention that delaying the interval check to the end is useful for unsatisfiable formulas. Although we have not performed an experimental comparison, it is likely to be much more efficient than Cooper's method in which the defining constraint on a variable is expressed using *mod*.

Recently, Jaffar et al [9] discussed a transitive closure algorithm for processing integer inequalities that have at most two variables; this algorithm is based on Shostak's method for computing loop residues. They showed that if the coefficients of variables in these inequalities are units (TVPI), then satisfiability can be checked in $O(n^3)$ time and $O(n^2)$ space, where n is the number of inequalities. Fourier's algorithm can be used for unit $TVPI$ problems, and it has the same complexity.

2.3 Using implicit equalities in detecting unsatisfiability

In contrast to Cooper's approach as well as Shostak's approach, a rewrite rule based prover is always on the lookout for deducing equalities so that they can be used for simplification, rewriting, and eliminating variables. *RRL* uses equalities (implicit as well as explicit) as rewrite rules. As soon as such an equality is identified, it can be used as a rewrite rule to simplify other inequalities. This becomes especially important while using the extended Fourier's algorithm for deciding linear arithmetic with uninterpreted function symbols as well as in the presence of function symbols defined using rewrite rules. Below we discuss how implicit equalities are derived using the extended Fourier's algorithm for integer inequalities.

The method of Lassez and Maher discussed earlier for the case of linear inequalities over the rationals extends to integers as the following theorem states.

Theorem 2.7 *An inequality C_k in a set of consistent inequalities $\mathcal{P}$ is an implicit equality in integers if the extended Fourier's algorithm produces an inequality $0 \leq 0$ using C_k.*

Example 2.8 Consider the set of integer linear inequalities

$$\begin{array}{lll} C_1: \ a - b \leq 0, & C_2: \ b - f(a) \leq 0, & C_3: \ f(a) \leq 1, \\ C_4: \ -a - b \leq -2, & C_5: \ -b - f(b) \leq -2, & C_6: \ f(a) - f(f(b)) \leq -1. \end{array}$$

It is easy to verify that

$$C_1 + 2C_2 + 2C_3 + C_4 = 0 \leq 0$$

Thus we have

$$\begin{array}{ll} C_1^{=}: \ a = b, & C_2^{=}: \ b = f(a), \\ C_3^{=}: \ f(a) = 1, & C_4^{=}: \ a + b = 2. \end{array}$$

As will be clear from subsequent discussion, the above equalities are used to deduce that each of $a, b, f(a), f(b), f(f(b))$ take the value 1. From this, the unsatisfiability of the above set of inequalities follows since C_6 cannot be satisfied.

The reader should note two crucial properties which are different from the case of inequalities over the rationals. Firstly, there is no guarantee that the deduced implicit equality is satisfiable, as the inequalities from which the equality is deduced may not be satisfiable. The second is that the converse of the above theorem does not hold. That is, there are equalities (implicit as well as derived) of $\mathcal{P}$ which cannot be deduced directly by the extended Fourier's algorithm. It is necessary to do some extra work, as illustrated by the following examples.

Example 2.9 Consider the following set of integer linear inequalities

$$\begin{array}{llll} C_1: & x - 4y \leq -1, & C_2: & -x + 4y \leq 2, \\ C_3: & x \leq 2, & C_4: & -x \leq -1. \end{array}$$

Eliminating y from C_1, C_2, we obtain $0 \leq 1$; the defining constraint for y is that $x + 1 \leq 4y \leq x + 2$. Eliminating x from C_3, C_4, we obtain $0 \leq 1$; the defining constraint for x is that $1 \leq x \leq 2$. So the extended Fourier's algorithm does not deduce any equality. However, the above set of inequalities is satisfiable, and the only solution is $x = 2, y = 1$ since for $x = 1$, there is no y satisfying the defining constraint of y. This implies that $4y = x + 2$ as well as $x = 2$ are implicit equalities. This example illustrates that even though the constraint on x suggests two assignments, one of those assignments cannot be extended as there are no values of y satisfying the defining constraint of y.

Example 2.10 Consider the following set of integer linear inequalities

$$\begin{array}{llll} C_1: & x - 4y \leq -1, & C_2: & -x + 4y \leq 2, \\ C_3: & x \leq 3, & C_4: & -x \leq -2. \end{array}$$

The defining constraint for y is that $x+1 \leq 4y \leq x+2$, and the defining constraint for x is $2 \leq x \leq 3$. For both values of x, $y = 1$ which is an equality that follows but cannot be obtained directly from any inequality by replacing $\leq$ by $=$. This example illustrates that a finite set of linear inequalities may not have any inequality which is really an equality but it may imply other equalities.

Extending Fourier's algorithm to deduce all implicit equalities from linear inequalities over the integers is an interesting challenge.

2.4 A decision procedure for pure linear arithmetic

A quantifier-free formula without any (uninterpreted or interpreted) function symbols can be decided using the extended Fourier's algorithm. Given a formula F, its proof can be done by refutation. An easy (but inefficient) way is to transform the negation of F into a disjunctive normal form,

$$\neg F \equiv G_1 \vee G_2 \vee \cdots \vee G_n$$

where each G_i is a conjunction of integer inequalities.[5] Henceforth, a disjunctive normal form will be used for the sake of simplicity. The formula $\neg F$ is satisfiable iff one of G_i is satisfiable, and so F is valid iff none of G_i is satisfiable.

Every literal is transformed into a disjunction of conjunction of inequalities using only $\leq$ predicate. For example, $A \neq B$ is replaced by $(A > B \vee A < B)$; in case of naturals and integers, $A < B$ is replaced by $A + 1 \leq B$, whereas for rationals, a new variable, called a *surplus variable*, is introduced, so we have $A + z \leq B$ with the requirement that z is strictly positive. Firstly, equalities are solved and used to eliminate variables. Inequalities are simplified. Then Fourier's extended procedure is applied on inequalities. Any equalities and inequalities deduced are first preprocessed with redundant inequalities removed.

[5] As pointed by Cooper [4], it is not necessary to transform a formula into a disjunctive normal form; instead once negation is pushed down to the literals, the resulting formula can be checked for satisfiability as it is. Another approach toward avoiding generating a disjunctive normal form is to use *if-then-else* notation used in the AFFIRM system as well as Boyer and Moore's theorem prover.

It should be noted that one need not have three separate implementations of Fourier's algorithms - one for rationals, another for integers and third for naturals. Instead, it suffices to have an implementation of the extended Fourier's algorithm. In the case of integers or naturals, satisfiability is declared only if no contradictory inequality is generated and a satisfying assignment for variables can be generated from their defining constraints. In the case of rationals, satisfiability is declared if no contradictory inequality is generated. Further surplus variables are eliminated at the end, and before elimination, it is checked that each surplus variable can indeed be assigned a positive value (which is ensured by checking that the interval for the surplus variable includes a positive value); one way to check this is to analyze the minimum of upper bounds for positiveness.

For natural numbers, an additional constraint of nonnegativeness for every variable over the naturals is added. But more importantly, the semantics of subtraction, to be denoted by $\ominus$ (also known as Peano subtraction), is different. Given $x \ominus y$, there are two cases to be considered: (i) x is no smaller than y, and (ii) x is smaller than y, in which case the result is 0.

3 Linear Arithmetic with Uninterpreted Function Symbols

Given a Presburger formula G with occurrences of uninterpreted function symbols, a formula G_a is generated, which is a conjunction of $GE \wedge G'$, such that G' is a pure Presburger formula without any uninterpreted symbols and GE is a conjunction of equalities relating functional terms to new symbols, called *abstraction constants.* GE does not include any occurrence of any arithmetic operator other than $=$. G_a is unsatisfiable if and only if G is unsatisfiable.

Equalities in G' are used to eliminate variables from inequalities in G' as well as from GE. The resulting equalities in GE can be processed using a congruence closure algorithm [23] or a Knuth-Bendix completion procedure on ground terms which is guaranteed to terminate [18, 8, 12]. In *RRL* , the Knuth-Bendix completion procedure on ground terms is used which gives a complete decision procedure for ground equalities as a finite set of rewrite rules. The rewrite rules are used to normalize G'.

The formula G' is subsequently processed using the extended Fourier's algorithm. Any equality deduced from G' is added to GE and a new canonical ground rewrite system is incrementally generated by adding the new equalities. This may result in additional equalities relating functional terms and hence abstraction constraints, which may further simplify the inequalities currently being considered in the extended Fourier algorithm (see example 2.11 for instance in which implicit equalities are used to relate terms).

This repeated interaction between the decision procedure for equality on ground terms, equalities deduced from Fourier's algorithm and normalization of inequalities using equalities terminates. This is so because (i) the ground completion procedure always terminates and (ii) Fourier's algorithm terminates since in each macro step, at least one variable appearing in the inequalities is eliminated (and number of variables appearing in the inequalities never increases; the number may decrease because of equalities).

The soundness of the procedure follows from the soundness of the ground completion and Fourier's algorithm. So, if unsatisfiability is detected by Fourier's algorithm, then G is unsatisfiability. Otherwise, if the procedure terminates giving a satisfying assignment for G_a, then G can be shown to be satisfiable by building a model for it provided all equalities that can be deduced from GE and G' have been made explicit using Fourier's algorithm and ground completion. Assuming that there is a way to extend Fourier's algorithm to deduce all implicit equalities, this approach gives a complete decision procedure for Presburger arithmetic with uninterpreted symbols. In the absence of a procedure for deducing all implicit equalities from integer inequalities, a method proposed in [23] can be used. All possible equalities among variables appearing in G_a are deduced using a complete decision procedure for unsatisfiability of integer inequalities.

Example 3.1 Consider an example from [27]:

$$z = f(x - y) \wedge x = z + y \wedge -y \neq -(x - f(f(z))).$$

The functional terms $f(x - y)$, $x - y$, and $f(f(z))$ are abstracted to be u_1, u_2 and u_3, respectively. We thus have:

$$(f(x - y) = u_1 \wedge x - y = u_2 \wedge f(f(z)) = u_3) \wedge (z = u_1 \wedge x = z + y \wedge -y \neq -(x - u_3)).$$

Ground completion on equalities would give: $\{z \rightarrow u_1, x \rightarrow y + u_1, u_2 \rightarrow u_1, u_3 \rightarrow u_1, f(u_1) \rightarrow u_1\}$. From this and $u_1 \neq u_3$, a contradiction follows.

4 Integration of Fourier's Algorithm in a Rewrite System

Most work on Presburger decision procedure assumed formulas in pure Presburger arithmetic [1, 6] or formulas with uninterpreted function symbols [4, 2, 25, 26]. In [27, 23], methods for combining decision procedures are described which enable handling some interpreted function symbols, for example the quantifier-free theory of *lists* with *cons, car, cdr.* In [3], Boyer and Moore described how Hodes' procedure can be integrated with interpreted symbols defined as lisp functions. In this section, we discuss the integration of a complete decision procedure for Presburger arithmetic with interpreted function symbols defined as a finite set of canonical unconditional rewrite systems.

Definition 4.1 A *rewrite rule* is an oriented equation of the form $lhs \rightarrow rhs$, where lhs is its left-hand side and rhs is its right-hand side. A *rewrite system* $\mathcal{R}$ is a finite set of rewrite rules.

Given a rewrite system $\mathcal{R}$ and a term $t[t']$, where t' is a subterm of t, $\mathcal{R}$ reduces t to another term s if there is a rewrite rule $l \rightarrow r$ and a substitution γ such that $\gamma(l) = t'$; s is t with the subterm t' replaced by $\gamma(r)$.

Definition 4.2 A rewrite system $\mathcal{R}$ is *canonical* if and only if it is *terminating* and *confluent*, i.e. it does not admit any infinite rewrite sequence and every rewrite sequence from a term t can be extended to give a unique normal form, called the *canonical form* of t.

It is easy to see that $\mathcal{R}$ is a decision procedure for equations. Given an equation $s = t$, we compute the canonical forms of s and t, and check for equality. If they are equal, then $\mathcal{R} \models s = t$; otherwise, $\mathcal{R} \models (s = t)$ does not follow. However, it is not possible to get, in general, a decision procedure for the quantifier-free theory of $\mathcal{R}$ even though $\mathcal{R}$ is canonical. For example, consider $\mathcal{R}$ to be the associativity rule (oriented in any way), which constitutes a canonical rewrite system and serves as a decision procedure for the equational theory of free semi-groups. If it was possible to get a decision procedure for the quantifier-free theory of $\mathcal{R}$, then we would be able to solve the word problem of any finitely presented semi-group (including the ones with unsolvable word problems) by specifying it as a conditional equation, in which the conditions are the finite presentation of the semi-group, and the conclusion is an equation relating two words.

The Knuth-Bendix completion procedure and its extensions can be used as a semi-decision procedure for the quantifier-free theory of $\mathcal{R}$. Whether

$$\mathcal{R} \models F = ((s_1 = t_1 \wedge \cdots \wedge s_k = t_k) \supset (s = t))$$

can be checked as follows: let s', t', s'_i, t'_i be Skolem forms of s, t, s_i, t_i, respectively, obtained by introducing Skolem constants for the variables in F. The completion procedure can be attempted on $\mathcal{R} \cup \{s'_i = t'_i\}$. If the conclusion $s' = t'$ in the conditional equation indeed follows, while attempting to generate an augmented canonical system from $\mathcal{R}$, the conclusion can be proved.

If $\mathcal{R}$ is such that for any finite set GE of ground equations expressed using symbols in $\mathcal{R}$ and constants, completion terminates producing a finite canonical rewrite system $\mathcal{R}'$, then $\mathcal{R}'$ can be used

to decide whether the conclusion of a conditional equation whose conditions constitute GE, follows from $\mathcal{R}$ or not. We will call a canonical $\mathcal{R}$ with this property as an *admissible* rewrite system $\mathcal{R}$. For an admissible $\mathcal{R}$, its quantifier-free theory is decidable using completion. (The above statement is true even when the conjecture above is generalized to $(L_1 \wedge \cdots \wedge L_k) \supset L$, where L as well as each $L_i, 1 \leq i \leq k$, is an equation or a disequation.)

Example 4.3 Consider an interpreted symbol defined using the equation $f(f(x)) = x$. The rewrite system $\{f(f(x)) \rightarrow x\}$ can be shown to be admissible. The conjecture is $f(x) = f(y) \supset x = y$. The Skolem form of its negation is $f(a) = f(b) \wedge a \neq b$. The ground equality $f(a) = f(b)$ is oriented to $f(a) \rightarrow f(b)$. Its left side $f(a)$ superposes with $f(f(x))$ giving a ground superposition $f(f(a))$ from which $a = b$ follows, and this gives a contradiction, implying that the conjecture follows.

This example indicates that it is not sufficient to normalize literals in a conjecture using $\mathcal{R}$; instead it is necessary to superpose ground equalities of the negation of the conjecture with rules in $\mathcal{R}$.

Example 4.4 As another example, consider a theory of *lists* with *cons, car, cdr* as discussed in [23, 27]. The defining rules for the interpreted function symbols are:

$$\begin{array}{lll} 1. & cons(car(x),\ cdr(x)) & \rightarrow\ x, \\ 2. & car(cons(x, y)) & \rightarrow\ x, \\ 3. & cdr(cons(x, y)) & \rightarrow\ y. \end{array}$$

These three rules can be shown to constitute an admissible rewrite system. The formula $F \equiv cons(car(x), cdr(car(y))) = cdr(cons(y, x)) \supset cdr(car(y)) = cdr(x)$ can be shown to follow from the rules as follows.

$$4.\quad cons(car(a), cdr(car(b))) \rightarrow cdr(cons(b, a)).$$

Rule 4 superposes with rule 3 to produce:

$$5.\quad cdr(car(b)) \rightarrow cdr(a),$$

from which the contradiction follows.

This complete decision procedure for the quantifier-free theory of an admissible rewrite system $\mathcal{R}$ can be combined with the complete decision procedure for Presburger arithmetic with uninterpreted symbols to give a complete decision procedure in the case of both uninterpreted and interpreted symbols. Similar to the case of Presburger arithmetic with uninterpreted symbols, functional terms are abstracted in a conjecture using new abstraction constants. These ground equalities and other ground equalities in a conjecture are then completed using ground completion; additional equalities are generated by superposition of ground equalities with $\mathcal{R}$. New equalities are used to normalize linear inequalities, from which additional implicit equalities are deduced. This interplay similar in the case of uninterpreted symbols, gives a complete decision procedure when interpreted symbols are axiomatized by an admissible rewrite system.

There exist equational theories which have a decidable quantifier-free theory but they do not admit admissible rewrite systems. Furthermore, there are equational theories with a decidable quantifier-free theory which have a canonical rewrite system, but the rewrite system is not admissible.

5 Fourier's Algorithm and Conditional Rewriting

Functions on commonly used data structures such as lists, sequences, arrays, records, etc., are typically expressed using conditional rewrite rules; unconditional rewrite rules are not sufficient. Further lemmas and theorems about functions are also typically conditional rewrite rules. Tecton and its theorem prover *RRL* support definitions and lemmas given as conditional rewrite rules.

Definition 5.1 A *conditional rewrite rule* is a rule of the form

$$lhs \rightarrow rhs \quad if \quad p_1 \wedge p_2 \cdots \wedge p_k,$$

where *lhs* is the left-hand side of the rewrite rule, *rhs* is the right-hand side and $p_1, p_2, \ldots, p_k$ are the *conditions*.

A term $t[t']$, where t' is a subterm of t, rewrites to another term s using a conditional rewrite rule $l \rightarrow r \quad if \quad p_1 \wedge \ldots \wedge p_n$, if there is a substitution γ such that $\gamma(l) = t'$ and each of the conditions $\gamma(p_i)$ reduces to *true* recursively also by rewriting using rules; s is t with the subterm t' replaced by $\gamma(r)$. If $\mathcal{R}$ does not include such a rule, then t is said to be in a normal form. As should be evident, conditional rewriting is a recursive process since in order to apply a rewrite rule, its conditions must reduce to true, which itself is determined by rewriting. Termination of this process is guaranteed by using a termination ordering $>_t$ in which $l >_t r$ as well as $l >_t p_i$. For a detailed treatment of conditional rewriting used in *RRL* , see [29, 30].

For combining a complete decision procedure for Presburger arithmetic with a decision procedure for interpreted symbols given as a canonical (conditional) rewrite system, it is possible to identify conditions on a conditional rewrite system similar to the one discussed for unconditional rewrite rules in the previous section. The effectiveness of such an approach is unclear because firstly, it may not be possible to generate a canonical conditional rewrite system, and then secondly, generating additional ground equalities from conditional rewrite rules and equalities can be expensive. In Tecton, we have attempted to make a compromise. We do not assume that interpreted symbols are defined using a canonical conditional rewrite system. We also do not perform any superpositions between the conditional rewrite system and the ground equalities obtained from the conditions of the conclusion. So the implementation is incomplete, but our experience in using it suggests that it works in most situations. Before giving all the steps performed in our preliminary implementation, we will use a streamlined example to illustrate the main features of the implementation.

Example 5.2 A goal to prove is:

$$(p(x) \wedge (z \leq f(max(x,y))) \wedge (0 < min(x,y)) \wedge (x \leq max(x,y)) \wedge (max(x,y) \leq x)) \supset (z < g(x) + y)$$

Among the rewrite rules in the rewrite system are the following two rewrite rules for the interpreted symbols max, f, g, p.

$$\begin{array}{ll} R_1: & min(x,y) \rightarrow y \quad if \quad max(x,y) = x \\ R_2: & f(x) \leq g(x) \rightarrow true \quad if \quad p(x) \end{array}$$

The goal is negated and Skolemized to give:

$$\begin{array}{l} p(A) \wedge (L \leq f(max(A,B))) \wedge (0 < min(A,B)) \\ \wedge (A \leq max(A,B)) \wedge (max(A,B) \leq A)) \wedge \neg(L < g(A) + B). \end{array}$$

(To save space, we do not use abstraction constants to abstract functional terms.) The literal $p(A)$ is the only one not belonging to Presburger language. The set of linear inequalities are:

$$\begin{array}{llll} C_1: & -f(max(A,B)) + L \leq 0, & C_2: & -min(A,B) \leq -1, \quad C_3: \quad -max(A,B) + A \leq 0, \\ C_4: & max(A,B) - A \leq 0, & C_5: & g(A) + B - L \leq 0 \quad . \end{array}$$

Because $C_3 + C4 = 0 \leq 0$, the implicit equality

$$E_1: \quad max(A,B) = A$$

is derived by Fourier's algorithm. This equality is used to simplify the inequality set to produce the following inequality set

$$C_1': \quad -f(A) + L \leq 0, \quad C_2: \quad -min(A,B) \leq -1, \quad C_5: \quad g(A) + B - L \leq 0.$$

Now L can be eliminated to give:

$$C_2: \quad -min(A,B) \leq -1, \quad C_5': \quad -f(A) + g(A) + B \leq 0.$$

Since $min(A,B)$ matches the left side of rule R_1 using the substitution $\{x \leftarrow A,\ y \leftarrow B\}$, its condition $max(A,B) = A$ must be established. This equality is already known, so $min(A,B)$ is reduced to B; the equality $min(A,B) = B$ is also added to the equality set. Assuming an ordering $f > g$, term $f(A)$ also matches a maximal term in the linear rule R_2, so the condition $p(A)$ must be established, which is already in the equality set. So, the instance of the linear rule, $f(A) \leq g(A)$, is added to the inequality set, giving:

$$\begin{aligned} C_2': &\quad -B \leq -1, \\ C_5': &\quad -f(A) + g(A) + B \leq 0, \\ C_6: &\quad f(A) - g(A) \leq 0. \end{aligned}$$

Fourier's algorithm detects a contradiction because an inequality $0 \leq -1$ is generated, implying that the original goal is proved.

5.1 Implementation

In this subsection, we give a brief sketch of our preliminary implementation. Many design decisions were greatly influenced by the discussion in Boyer and Moore's report [3] in which they extensively discussed data structures for representing a data base of contexts and the interaction between the rewriter and the decision procedure.

Our preliminary implementation works as follows. Given a goal G, it is negated and Skolemized. The negated Skolemized goal may be divided into many subgoals by splitting at the top most level if there is a disjunction. Each of the subgoals which is of the form $L_1 \wedge \cdots \wedge L_k$ where each L_i is a literal, is attempted for unsatisfiability. From L_i's, a set GE of ground equalities (including literals such as $p(a)$ or $p(a) = false$) is collected. Below, we give the steps:

1. Ground completion is performed to generate a canonical rewrite system for ground equalities. If a contradiction is detected, the subgoal is unsatisfiable. Otherwise, the canonical rewrite system is used to normalize inequalities.

2. The normalized inequalities are passed to Fourier's algorithm for elimination. (a) Whenever an implicit equality is generated, it is passed to step 1, which is repeated. (b) If a contradiction is detected, then the subgoal is unsatisfiable. After steps 1 and 2, if no progress can be made, go to the next step.

3. A maximal literal (term) among all the ground inequalities and ground equalities generated is selected. This literal is checked for matchability with the left side of a rule defining an interpreted symbol. In the case of a linear rule (such as R_2 above relating f, g, p), a maximal term in the left side of a rule is used for checking a possible match.

   Suppose θ is a substitution under which a match is possible with a rule; it is then checked whether the condition in the rewrite rule, if any, instantiated by θ follows from the remaining literals (ground equalities and inequalities generated so far) and other rewrite rules defining interpreted symbols appearing in the condition. This is done using contextual rewriting [29, 30]. If the condition in the rewrite rule is established, then the subterm in the goal matching the left side is replaced by its right hand side instantiated with θ. The above steps are then repeated on the result.

   This process of checking for matchability is itself recursive, as it may involve doing steps 1, 2, and 3 on conditions (on a smaller input in a well-founded order).

   If at any step, it is not possible to reduce a maximal literal, then the procedure terminates declaring that it is unable to prove the subgoal and hence, the original goal.

6 Further Extensions

We have used the Tecton proof system to verify properties of many problems, including sorting algorithms such as insertion sort and quicksort, string matching algorithms, the termination of Takeuchi's function [22], many efficient iterative programs for computing arithmetic functions using the so-called Russian peasant algorithm. Recently, we have also verified properties of parallel programs expressed using a powerlist data structure and recursion as proposed by Misra [21]. The use of the procedure for Presburger arithmetic has made the proofs compact and relatively easier to automate and understand in contrast to proofs generated without using Presburger arithmetic.

In our initial implementation, Fourier's algorithm detected inconsistency if an inequality $0 \leq c$ is derived where $c < 0$. No attempt was made to check for unsatisfiability of integral equalities, or to check whether a satisfying assignment could be generated from defining constraints on the variables eliminated. In the case of natural numbers, we limited the amount of case analysis by considering only some of the conditions generated from Peano's subtraction operation. This was again based on efficiency considerations. We are currently extending our implementation to include additional checks discussed in this paper. Further, some aspects of computing superposition of ground equalities with conditional rewrite rules defining interpreted function symbols that can be performed efficiently, would also be included, and this is likely to extend the class of formulas that the Tecton system would be able to handle automatically.

Acknowledgement: We thank Mahadevan Subramaniam for helpful comments on earlier drafts of the paper.

References

[1] W.W. Bledsoe, A new method for proving certain Presburger formulas, *Proc. 4th IJCAI,* Tbilisi, Georgia, 1975, 15-21.

[2] W.W. Bledsoe and R. Shostak, A prover for general inequalities, 6th IJCAI, 1979, 66-69.

[3] R.S. Boyer and J S. Moore, Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic, *Machine Intelligence* 11 (1988) 83–157.

[4] D.C. Cooper, Theorem proving in arithmetic without multiplications, *Machine Intelligence* 6 (1972) 43–59.

[5] D. Craigen, S. Kromodimoelijo, I. Meisels, W. Pase, and M. Saaltink, Eves system description, *Proc. Automated Deduction - CADE 11,* LNAI 607, Springer Verlag (1992), 771-775.

[6] H. Enderton, *A Mathematical Introduction to Logic.* Academic Press, 1977.

[7] L. Hodes, Solving problems by formula manipulation, *Proc. 2nd Intl. Conf. on AI,* The British Computer Society, 1971, 553-559.

[8] G. Huet and D. Lankford, On the uniform halting problem for term rewriting systems, INRIA Report 283, March 1978.

[9] J. Jaffar, M.J. Maher, P.J.Stuckey, and R.H.C. Yap, "Beyond finite domains," Proc. *Second Workshop on Principles and Practices of Constraint Programming* (*PPCP '94*), Orcas Island, Washington, May 1994, 77-84.

[10] D. Kapur, An automated tool for analyzing completeness of equational specifications, Proc. *Intl. Symp. on Software Testing and Analysis,* Seattle, August 1994.

[11] D. Kapur, D.R. Musser, and X. Nie, "An Overview of the Tecton Proof System," accepted for publication in *Theoretical Computer Science* Journal, special issue on *Formal Methods in Databases and Software Engineering,* (ed. V. Alagar), Vol. 133, October, 1994.

[12] D. Kapur and P. Narendran, The Knuth-Bendix completion procedure and Thue systems, *SIAM Journal on Computing* 14(4), (Nov. 1985), 1052-1072.

[13] D. Kapur and X. Nie, Reasoning about Numbers in Tecton. Tech. Report, Dept. of Computer Science, State University of New York, Albany, NY 12222, 1994.

[14] D. Kapur and M. Subramaniam, Using linear arithmetic procedure for generating induction schemes in mechanizing induction, submitted to *FSTTCS, 1994.*

[15] D. Kapur, and H. Zhang, An overview of Rewrite Rule Laboratory (RRL), to appear in a special issue of *Computers in Math. with Applications,* 1994. Earlier descriptions appeared in CADE-88 and RTA-89.

[16] T. Käufl, Reasoning about systems of linear inequalities, *Proceedings of 9th International Conference on Automated Deduction* 563–572, Argonne, Illinois (1988).

[17] D. Knuth, *The art of computer programming: Seminumerical algorithms.* Vol. 2, Second Edition, 4.5.2, 326-328, Addison-Wesley. 1981.

[18] D. Knuth and P. Bendix, Simple word problems in universal algebras, in *Computational Problems in Abstract Algebra* (ed. Leech), Pergamon Press (1970), 263-297.

[19] J.-L. Lassez, T. Huynh and K. McAloon, Simplification and elimination of redundant linear arithmetic constraints, *Proc. North American Conf. on Logic Programming,* 37-51, 1989.

[20] J.-L. Lassez and M.J. Maher, On Fourier's algorithm for linear arithmetic constraints, *J. of Automated Reasoning,* 9, 1992, 373-379.

[21] J. Misra, "Powerlist: A structure for parallel recursion," *A Classical Mind: Essays in Honor of C.A.R. Hoare,* Prentice Hall, Jan. 1994.

[22] J S. Moore, A mechanical proof of the termination of Takeuchi's function. *Information Processing Letters* 9(4): 176 – 181 (1979).

[23] G. Nelson and D.C. Oppen, Simplification by cooperating decision procedures, *ACM Transactions on Programming Languages and Systems* 1 (2) (1979) 245 – 257.

[24] W. Pugh, A practical algorithm for exact array dependence analysis, *Communications of the ACM,* 35(8), (1992), 112-114.

[25] R.E. Shostak, On the SUP–INF method for proving presburger formulas, *Journal of ACM* 24 (4) (1977) 529–543.

[26] R.E. Shostak, A practical decision procedure for arithmetic with function symbols, *Journal of ACM* 26(2) (1979) 351–360.

[27] R.E. Shostak, Deciding combination of theories, *Journal of ACM* 31 (1), (1984) 1-12.

[28] H.P. Williams, Fourier-Motzkin elimination extension to integer programming problems, *Journal of Combinatorial Theory (A),* 21, (1976), 118-123.

[29] H. Zhang, Reduction, Superposition, and Induction: Automated Reasoning in an Equational Logic. Ph.D. Thesis. Rensselaer Polytechnic Institute, Computer Science Department (1988).

[30] H. Zhang, Implementing contextual rewriting. In: Proc. *Third International Workshop on Conditional Term Rewriting Systems,* J. L. Remy and M. Rusinowitch (eds.), Lecture Notes in Computer Science, Vol. 656, Springer-Verlag, Berlin, 1992, pp. 363–377.

[31] H. Zhang, D. Kapur, and M.S. Krishnamoorthy, "A mechanizable induction principle for equational specifications," *Proc. of Ninth International Conference on Automated Deduction (CADE-9)*, Argonne, IL. Springer-Verlag LNCS 310, 250-265, 1988.

Comparison of Various Multivariate Resultant Formulations*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222
{*kapur, saxena*}*@cs.albany.edu*

Abstract

Three most important resultant formulations are the Macaulay, Dixon and sparse resultant formulations. For most polynomial systems, however, the matrices constructed in these formulations become singular and the projection operator vanishes identically. In such cases, perturbation techniques for Macaulay formulation such as *generalized characteristic polynomial* (*GCP*) and a method based on *rank submatrix computation* (*RSC*), applicable to all three formulations, can be used, giving four methods, Macaulay/*GCP*, Macaulay/*RSC*, Dixon/*RSC* and Sparse/*RSC*, for computing nontrivial projection operators.

In this paper, these four methods are compared. It is shown that the Dixon matrix is (by a factor up to $\mathbf{O}(e^n)$ for a certain class) smaller than the sparse resultant matrix which is (by a factor up to $\mathbf{O}(e^n)$ for a certain class) smaller than the Macaulay matrix. Empirical results confirm that Dixon/*RSC* is the most efficient, followed by Sparse/*RSC* then Macaulay/*RSC* and finally Macaulay/*GCP*, which is found to be almost impractical. All four methods are found to generate extraneous factors in the projection operator.

Efficient heuristics for interpolation, used to expand the resultant matrices, are also discussed.

1 Introduction

Solving a set of nonlinear polynomial equations or deriving conditions for the existence of their solutions is a fundamental problem in many areas of mathematics, engineering, physical and computer sciences. These methods sometimes involve elimination of variables from a given set of polynomial equations to obtain a set of polynomials whose vanishing is a necessary (and sometimes sufficient) condition for the existence of solutions. Elimination of n variables from $n+1$ equations results in a single such polynomial known as the *resultant*. In recent years, as computational power has increased, a lot of research has been devoted to eliminating variables symbolically and computing the resultant.

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

The resultant of a system of polynomials is an element of a Gröbner basis of its ideal [3] if an elimination monomial ordering is used, but this is an expensive way to compute the resultant. Until recently, a classical method by Sylvester was used to compute the resultant [9]. This method eliminates one variable from two polynomials, so to eliminate n variables from $n+1$ polynomials, it has to be applied successively. It turns out that more efficient methods, which eliminate all variables together from the set of polynomials, had been developed at the beginning of this century by mathematicians such as Cayley [7], Dixon [10] and Macaulay [20]. These multivariate resultant formulations were recently resurrected by many researchers, and have numerous applications [21, 8, 17].

Typically, these multivariate resultant formulations try to express resultants as formulas involving determinants. Ideally, one wants a single matrix whose determinant is the resultant, however this is not always possible[1]. In general, one has to settle for formulations which either express the resultant as a function of more than one determinants or give a single matrix whose determinant is a nontrivial multiple of the resultant. In this paper, we concentrate on such approaches.

Three major multivariate resultant formulations are the Macaulay [20, 4], Dixon [10, 18] and sparse [22, 5] resultant formulations. Given a set of polynomials, these formulations construct matrices, called the Macaulay matrix, the Dixon matrix and the sparse resultant matrix, respectively. In Macaulay formulation, the ratio of the determinants of the Macaulay matrix and one of its submatrices gives the resultant or some multiple of the resultant[2] (called a *projection operator*). In Dixon and sparse resultant formulations, the determinant of the respective matrices gives a projection operator. Unfortunately, sometimes while working with nongeneric, nonhomogeneous polynomial systems, some or all these three matrices can become singular and the projection operator vanishes identically. Such trivial projection operators are useless and give no information about the affine solutions of the system.

One way to extract a nontrivial projection operator is by perturbation of the polynomial system [15, 4]. Ad hoc perturbations can be used with all three formulations, but they may not work. For Macaulay formulation, Canny [4]

[1] Weyman and Zelevinsky [24] classify certain cases in which a pure determinantal formulation is possible. We have been informed by a referee that Jouanolou's work also discusses cases which permit pure determinantal formulation.

[2] In the case of generic homogeneous polynomials, Macaulay formulation results in the exact resultant.

gave a general method to perturb any polynomial system, compute its *Generalized Characteristic Polynomial* (*GCP*) and extract a nontrivial projection operator from the *GCP*. We call this the **Macaulay**/*GCP* method.

Another method for extraction of a nontrivial projection in the face of singular resultant matrices was outlined in our previous paper [18]. This method uses rank subdeterminant computation (*RSC*) which, though developed in conjunction with Dixon formulation, is applicable to all three formulations. This gives three more methods, **Macaulay**/*RSC*, **Dixon**/*RSC* and **Sparse**/*RSC*, giving a total of four methods for computing nontrivial projection operators. However, the choice between these four methods is not clear, as they have never been compared.

The purpose of this paper is to compare these four methods. We give theoretical bounds on the relative sizes of the three resultant matrices in the worst case for two different measures, viz., total degree of the polynomials and the degree of polynomials in individual variables. It is found that the Dixon matrix is the smallest, followed by the sparse resultant matrix, and the Macaulay matrix is the largest. In fact, in the worst case under the individual variable degree measure, the Dixon matrix is smaller by an exponential ($O(e^n)$) factor than the sparse resultant matrix, which in turn is smaller by an exponential (again, $O(e^n)$) factor than the Macaulay matrix. Moreover, we show by example that all four methods suffer with *extraneous factors*.

The comparison is evidenced by empirical results on 11 problems (implicitization, geometry formula derivation, equilibrium of Lorentz system, Vision etc.) and 3 random examples. Dixon/*RSC* seems most efficient and was able to solve all 14 problems, followed by Sparse/*RSC* which also solved all problems, but took much (up to 50 times) longer. Next was Macaulay/*RSC* which could not solve 3 problems, and on others, took much longer than both, Sparse/*RSC* and Dixon/*RSC*. Macaulay/*GCP* was almost impractical due to the introduction of an extra perturbation variable, and could solve only 2 problems. The software we used is described in section 5. For comparison purposes, we also give timings for Gröbner basis construction.

In section 2, we outline the three multivariate resultant formulations and point out their limitations. In section 3, we give two ways to get around these limitations and extract a nontrivial projection operator. In section 4, we give theoretical bounds on the relative sizes of the three resultant matrices. In section 5, implementational details of the four methods and some improvements on Zippel's interpolation algorithm are outlined. In section 6, we compare the four methods and give empirical evidence. Section 7 concludes giving some future research problems.

2 Multivariate Resultant Formulations

2.1 Notation and Preliminaries

Let $X = \{x_1, \ldots, x_n\}$ be a set of n variables and $A = \{a_1, \ldots, a_m\}$, a set of m parameters distinct from X. Let $\mathcal{F} = \{p_1, \ldots, p_{n+1}\} \subset \mathbb{Q}[X, A]$ be a set of $n+1$ nonhomogeneous polynomials. Let $tdeg(p, X)$ be the total degree of the polynomial p in the variable set X and $deg(p, x_j)$, the degree of the polynomial p in the variable x_j.

The objective is to eliminate the variables X from $\mathcal{F}$ and obtain a polynomial in the parameters A whose vanishing is a necessary condition for the existence of solutions to $\mathcal{F}$. Note that there may not exist a condition which is also sufficient in the nonhomogeneous case. However, there *is* the *smallest* necessary condition, which we call the *resultant*. Multiples of the resultant are called *projection operators* and their vanishing is also a necessary condition for $\mathcal{F}$ to have a common solution. The resultant may be reducible[3], but, if a projection operator is irreducible, it must be the resultant. All factors in a projection operator besides the resultant are *extraneous factors*. If a polynomial system has solutions in $\mathbb{C}^n$ under all possible specializations of the parameters (i.e., the system has an affine variety of dimension $> m-1$ in $\mathbb{C}^{n+m}$), then the resultant is identically zero. Also, an identically zero polynomial trivially qualifies as a projection operator for any system of polynomials since it is a multiple of the resultant with an extraneous factor of zero. Such trivial projection operators are obviously useless as they give no information about the solutions of polynomial systems. For polynomial systems whose affine variety is of dimension $m-1$ in $\mathbb{C}^{n+m}$, we are interested in computing projection operators which are not identically zero, at least when the system has a nontrivial common solution (i.e., a solution in the *algebraic torus* $(\mathbb{C}-\{0\})^n$ for some specialization of the parameters), and hopefully have few extraneous factors.

Let $\mathcal{I} \subseteq \mathbb{Q}[X, A]$ be the ideal generated by $\mathcal{F}$. A straightforward way to compute the resultant is by computing the Gröbner basis of $\mathcal{I}$ using an appropriate elimination ordering. The resultant is then an element of the basis. However, Gröbner basis methods are inefficient. Consider the following alternative approach.

Find another set of polynomials $\hat{\mathcal{F}} \subset \mathcal{I}$ with the following two properties:

1. Set of solutions of $\hat{\mathcal{F}}$ is preferably equal to or a superset (as small as possible) of the set of solutions of $\mathcal{F}$.
2. Number of different terms (in X) contained in the polynomials of $\hat{\mathcal{F}}$ is exactly as many as the number of polynomials in $\hat{\mathcal{F}}$.

If such an $\hat{\mathcal{F}}$ can be found, then it can be viewed as a linear system of homogeneous equations by treating each term in X (including 1) as an independent variable. A projection operator can then be found as the determinant of the coefficient matrix of this linear system.

The three most efficient methods for computing the resultant, viz. Macaulay, Dixon and sparse resultant formulations, try to find such $\hat{\mathcal{F}}$s. In this section we briefly outline these formulations. Our objective is not to describe methods in full detail, rather to give just enough information so that the reader can get an idea of the resultant matrix sizes and limitations of these formulations. Before we do that, we give an example and its resultant. This example is used throughout the paper to illustrate the applicability or limitations of different methods.

Example: (Resultant) Let $\mathcal{F} = \{p_1, p_2, p_3\} \subset \mathbb{Q}[a, b, c, x, y]$ where

$$\begin{aligned} p_1 &= ax^2 + bxy + (b + c - a)x + ay + 3(c-1), \\ p_2 &= 2a^2x^2 + 2abxy + aby + b^3, \\ p_3 &= 4(a-b)x + c(a+b)y + 4ab. \end{aligned}$$

The variables x and y are to be eliminated to obtain the resultant, a polynomial in the parameters a, b and c. From

[3] For generic polynomials the resultant is always irreducible, however under specializations, the resultant can be reducible. Eg. the resultant of $ax + b = 0$ and $bx + a = 0$ is $(a-b)(a+b)$.

the Gröbner basis of this system, the resultant is found to be the following irreducible polynomial with 75 terms:

$$
\begin{aligned}
\rho(a,b,c) = {} & 144ca^2b^3 + 28a^2c^2b^5 - 144a^4b^4 + 36c^2a^3b^2 - 72c^3a^3b^2 + 4cb^9 \\
& + a^3b^6c^2 - 12c^2a^3b^5 - 240a^4b^2 - 264a^4c^3b + 336a^4c^2b - 144a^4cb \\
& - 72a^5cb - 384a^4cb^3 + 36a^5c^2 + 112a^5b^3 - 8ca^6b^2 - 80a^5b^4 - 32a^6b^2 \\
& + 20ca^4b^4 + 96ba^5 + 48a^3b^6 + 288a^4b^3 - 96a^3b^4 + 192a^3b^3 - 16a^2b^7 \\
& + 40a^3c^3b^3 - 192a^5b^2 + 16a^5b^4c + 16a^4b^5c - 8a^3b^6c - 24a^3c^2b^3 \\
& - 8a^2b^6c^2 + 2a^2c^2b^7 - 12a^2b^7c + 80a^3b^5 + 20ca^2b^6 - 264c^2a^2b^3 \\
& - 96a^6c^2b - 80a^5b^2c^2 + 320a^5cb^2 - 40a^5cb^3 - 16a^4c^3b^3 + 44a^4c^2b^3 \\
& - 216a^3cb^3 + 2a^3c^4b^3 + 16a^3cb^4 + 108a^3b^4c^2 - 20a^3b^5c - 28a^3c^3b^4 \\
& - 8a^2c^3b^5 + 2a^5c^2b^3 + 72a^4c^4b + 4a^2c^4b^4 + 36a^3c^4b^2 + 2ac^4b^5 \\
& - 24c^2a^2b^4 + 120c^3a^2b^3 + 64a^7b^2 + 72ca^2b^4 - 52c^2ab^6 + 96a^6cb \\
& - 4cab^7 + 2c^2ab^7 + 48cab^6 - 16a^2b^6 + ac^2b^8 + 4c^3ab^6 - 24a^5c^2b \\
& - 24a^4b^2c^2 + 216a^4cb^2 + 40a^4c^3b^2 - 48a^2b^4 + 36a^5c^4 - 72a^5c^3. \quad \square
\end{aligned}
$$

2.2 The Macaulay Formulation

For $1 \leq i \leq n+1$, let $d_i = tdeg(p_i, X)$ and $d_M = 1 + \sum_1^{n+1}(d_i - 1)$. Let the set of terms $T = \{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \mid \alpha_1+\alpha_2+\ldots+\alpha_n \leq d_M\}$. Each polynomial, p_i, in $\mathcal{F}$ is multiplied with certain terms to generate $\hat{\mathcal{F}}$ with $|T|$ equations in $|T|$ unknowns (which are terms in T). The coefficient matrix of $\hat{\mathcal{F}}$ so constructed is the **Macaulay matrix**. Columns of the Macaulay matrix are labeled by the terms in T in some order, and each row corresponds to a polynomial in $\mathcal{F}$ multiplied by a certain term.

The order in which polynomials are considered for selecting multipliers results in different (but equivalent) $\hat{\mathcal{F}}$s. Let M_σ denote the Macaulay matrix for a permutation σ of polynomials in $\mathcal{F}$. Macaulay also defined N_σ, a certain square submatrix of M_σ, such that the projection operator is the ratio of the determinants of M_σ and N_σ. See [20, 4, 17] for details of the construction and proofs of these properties.

Example: (Macaulay) For the example at the beginning of this section, $T = \{x^3, x^2y, x^2, xy^2, y^3, y^2, xy, x, y, 1\}$. $\hat{\mathcal{F}}$ is obtained by multiplying p_1 by x, y and 1, p_2 by x, y and 1 and p_3 by xy, x, y and 1. The Macaulay matrix, M_σ is the following 10×10 matrix:

$$
\begin{array}{c}
\\
x \times p_1 \\ y \times p_1 \\ p_1 \\ x \times p_2 \\ y \times p_2 \\ p_2 \\ xy \times p_3 \\ x \times p_3 \\ y \times p_3 \\ p_3
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} x^3 & x^2y & x^2 & xy^2 & y^3 & y^2 & xy & x & y & 1 \end{array} \\
\left(\begin{array}{cccccccccc}
a & b & (b+c-a) & 0 & 0 & 0 & a & 3(c-1) & 0 & 0 \\
0 & a & 0 & b & 0 & a & (b+c-a) & 0 & 3(c-1) & 0 \\
0 & 0 & a & 0 & 0 & 0 & b & (b+c-a) & a & 3(c-1) \\
2a^2 & 2ab & 0 & 0 & 0 & 0 & ab & b^3 & 0 & 0 \\
0 & 2a^2 & 0 & 2ab & 0 & ab & 0 & 0 & b^3 & 0 \\
0 & 0 & 2a^2 & 0 & 0 & 0 & 2ab & 0 & ab & b^3 \\
0 & 4(a-b) & 0 & c(a-b) & 0 & 0 & 4ab & 0 & 0 & 0 \\
0 & 0 & 4(a-b) & 0 & 0 & 0 & c(a-b) & 4ab & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & c(a-b) & 4(a-b) & 0 & 4ab & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4(a-b) & c(a-b) & 4ab
\end{array}\right)
\end{array}
$$

and N_σ is

$$
\begin{array}{c}
\\ p_1 \\ p_2
\end{array}
\begin{array}{c}
\begin{array}{cc} x^2 & y^2 \end{array} \\
\left(\begin{array}{cc} a & 0 \\ 2a^2 & 0 \end{array}\right)
\end{array}.
$$

Unfortunately, here both $det(M_\sigma)$ and $det(N_\sigma)$ are 0, so the resultant or a nontrivial projection operator cannot be computed directly. $\square$

Limitation: As is demonstrated in this example, this formulation may be unsuccessful in computing a nontrivial projection operator. One of the reasons is that $det(N_\sigma)$ can be identically zero, and the division cannot be carried out. Even if that is not the case, $det(M_\sigma)$ can be identically zero, hence giving only a trivial projection operator.

2.3 The Dixon Formulation

Let $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ be a new set of variables and

$$
\delta(X, \bar{X}) = \begin{vmatrix} P_{1,1} & \cdots & P_{1,n+1} \\ P_{2,1} & \cdots & P_{2,n+1} \\ \vdots & \vdots & \vdots \\ P_{n,1} & \cdots & P_{n,n+1} \\ p_1(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) & \cdots & p_{n+1}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) \end{vmatrix},
$$

where for $1 \leq j \leq n$ and $1 \leq i \leq (n+1)$,

$$
P_{j,i} = \frac{p_i(\bar{x}_1, \ldots, \bar{x}_{j-1}, x_j, \ldots, x_n) - p_i(\bar{x}_1, \ldots, \bar{x}_j, x_{j+1}, \ldots, x_n)}{(x_j - \bar{x}_j)},
$$

and $p_i(\bar{x}_1, \ldots, \bar{x}_k, x_{k+1}, \ldots, x_n)$ stands for uniformly replacing x_j by $\bar{x}_j$ for $1 \leq j \leq k$ in p_i. The polynomial δ is known as the **Dixon polynomial**. Now, $\hat{\mathcal{F}}$ is the set of all coefficients (which are polynomials in X) of terms in δ, when viewed as a polynomial in $\bar{X}$. The coefficient matrix of $\hat{\mathcal{F}}$ is known as the **Dixon matrix**. It can be proven that for *generic ndegree*[4] polynomials, the Dixon matrix is square - hence its determinant is a projection operator [10, 18].

Example: (Dixon) The Dixon matrix for the previous example is of dimension 2×3:

$$
\begin{array}{c}
\\ x \\ 1
\end{array}
\begin{array}{c}
\begin{array}{ccc} 1 & x & y \end{array} \\
\left(\begin{array}{ccc}
\begin{array}{l} 4ca^2b^2 - c^2b^4 + 12cab^2 \\ -12ca^2b + 12a^2b - 12ab^2 \\ -c^2b^3a + cb^3a^2 - 4ab^4 \\ -cb^5 - 4a^3b^2 + 8a^2b^3 \end{array} &
\begin{array}{l} 4a^3b^2 - 8a^4b \\ -6a^3c + 6a^3c^2 \\ +6a^2c^2b - 6ca^2b \\ -cb^3a^2 - cb^4a \end{array} &
\begin{array}{l} 6a^2c^2b - 6cab^2 \\ +4a^2b^3 - 8a^3b^2 \\ -cb^5 - cb^4a \\ -6ca^2b + 6ac^2b^2 \end{array} \\
\begin{array}{l} 8ca^2b^2 - 4b^5 + 24cab^2 \\ -30ca^2b + 24a^2b - 24ab^2 \\ -6a^3c + 6a^3c^2 - cb^3a^2 \\ -8a^4b - 4a^3b^2 - cb^4a \\ +8a^2b^3 + 6a^2c^2b + 4ab^4 \end{array} &
\begin{array}{l} 12a^3b - 4a^2b^2 \\ -8a^4 + 2a^3c^2 \\ +2ca^2b^2 - 2a^4c \\ +2a^2c^2b \end{array} &
\begin{array}{l} 2ac^2b^2 + 12a^2b^2 \\ +2a^2c^2b - 2a^3cb \\ -8a^3b - 4b^3a \\ +2cb^3a \end{array}
\end{array}\right)
\end{array}.
$$

Since the Dixon matrix is rectangular, its determinant cannot be computed. Hence the resultant cannot be computed by this formulation either. $\square$

Limitation: Again this example demonstrated the limitation of Dixon method. The Dixon matrix can be rectangular, hence eliminating the possibility of computing the determinant. Even if it is square, it may be singular, resulting in a trivial projection operator.

2.4 The Sparse Resultant Formulation

This approach is based on the recent results pertaining to sparse polynomial systems (Bernshtein [2], Gelfand et. al. [16] and Sturmfels [22]). Sparse resultants appeared in Sturmfels & Zelevinsky [23], its matrix formulation was given by Canny & Emiris [5] and was successively improved in [6, 11, 12] using better heuristics. The Bezout bound on the number of solutions to a set of polynomials is quite loose. Instead, this formulation uses the recently developed, and significantly tighter, *BKK bound* [16] to construct a smaller resultant matrix than Macaulay.

Newton Polytope of a polynomial p is the convex hull of the set of exponents (treated as points in $\mathbb{R}^n$) of all terms in p. *Minkowski sum* of two Newton polytopes Q and S is the set of all vector sums, $q+s$, $q \in Q$, $s \in S$. Let $Q_i \subset \mathbb{R}^n$ be the Newton polytope (wrt. X) of the polynomial p_i in

[4] $n+1$ nonhomogeneous polynomials $p_1, \ldots, p_{n+1}$ in $x_1, \ldots, x_n$ are called *generic ndegree* if all coefficients are independent parameters and there exist nonnegative integers $m_1, \ldots, m_n$ such that each $p_j = \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} a_{j,i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n}$ *for* $1 \leq j \leq n+1$.

$\mathcal{F}$. Let $Q = Q_1 + \cdots + Q_{n+1} \subset \mathbb{R}^n$ be the Minkowski sum of Newton polytopes of all the polynomials in $\mathcal{F}$. Let $\mathcal{E}$ be the set of exponents (lattice points of $\mathbb{Z}^n$) in Q obtained after applying a small perturbation to Q to move as many boundary lattice points as possible outside Q.

Construction of $\hat{\mathcal{F}}$ here is similar to Macaulay formulation - each polynomial, p_i in $\mathcal{F}$ is multiplied with certain terms to generate $\hat{\mathcal{F}}$ with $|\mathcal{E}|$ equations in $|\mathcal{E}|$ unknowns (which are terms in $\mathcal{E}$), and its coefficient matrix is the **sparse resultant matrix**. A projection operator for $\mathcal{F}$ is simply the determinant of this matrix. Columns of the sparse resultant matrix are labeled by the terms in $\mathcal{E}$ in some order, and each row corresponds to a polynomial in $\mathcal{F}$ multiplied by a certain term.

Recall that the size of the Macaulay matrix is $|T|$. Size of the sparse resultant matrix, $|\mathcal{E}|$, is typically smaller than $|T|$, especially when the BKK bound is tighter than the Bezout bound. Algorithms in [11, 12, 6] construct matrices using some greedy heuristics which may result in smaller matrices, but in the worst case, the size can still be $|\mathcal{E}|$.

Example: (Sparse) Continuing with the same example, construction of the Newton polytopes and their Minkowski sum (after a perturbation in the positive direction) reveals that $\mathcal{E} = \{xy, xy^2, xy^3, x^2y, x^2y^2, x^2y^3, x^3y, x^3y^2, x^4y\}$. The 9×9 sparse resultant matrix is:

$$
\begin{array}{c}
 \\
xy \times p_1 \\
xy^2 \times p_1 \\
x^2y \times p_1 \\
xy \times p_2 \\
xy^2 \times p_2 \\
x^2y \times p_2 \\
xy \times p_3 \\
xy^2 \times p_3 \\
x^2y^2 \times p_3
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
xy & xy^2 & xy^3 & x^2y & x^2y^2 & x^2y^3 & x^3y & x^3y^2 & x^4y
\end{array} \\
\begin{pmatrix}
3(c-1) & a & 0 & b-a+c & b & 0 & a & 0 & 0 \\
0 & 3(c-1) & a & 0 & b-a+c & b & 0 & a & 0 \\
0 & 0 & 0 & 3(c-1) & a & 0 & b-a+c & b & a \\
b^3 & ab & 0 & 0 & 2ab & 0 & 2a^2 & 0 & 0 \\
0 & b^3 & ab & 0 & 0 & 2ab & 0 & 2a^2 & 0 \\
0 & 0 & 0 & b^3 & ab & 0 & 0 & 2ab & 2a^2 \\
4ab & c(a+b) & 0 & 4(a-b) & 0 & 0 & 0 & 0 & 0 \\
0 & 4ab & c(a+b) & 0 & 4(a-b) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4ab & c(a+b) & 0 & 4(a-b) & 0
\end{pmatrix}
\end{array}.
$$

The determinant of the above matrix is a projection operator with 205 terms (in expanded form):

$$\varrho = -2a^2\,(a-b-c)\left(a^2c + abc - 4ab + 4b^2\right)\rho\,(a,b,c)\,.$$

Notice that there are 3 extraneous factors. □

Limitation: Even though, in this particular example we are able to obtain a nontrivial projection operator, it may not always be the case. Although it seems rare, the sparse resultant matrix can be singular (Example 2 in Table 3), in which case the determinant will be a trivial projection operator.

3 Two Ways to Extract the Projection Operator

There are two ways to deal with degenerate resultant matrices encountered in all three formulations. One involves perturbation of the original set of polynomials and the other involves *rank submatrix computation.*

3.1 Perturbation (*GCP*)

The main problem in all three formulations is that resultant matrices can be rectangular, or even if square, they may be singular. Perturbation of polynomials can result in square and nonsingular resultant matrices [15, 4, 8]. One can try ad hoc perturbations of $\mathcal{F}$ in the hope of obtaining a nontrivial projection operator as follows. Randomly perturb the system using a perturbation variable, compute the projection operator for this perturbed system, factor it, remove any extraneous factors and finally, substitute zero for the perturbation variable to get the projection operator of the original system. Such ad hoc perturbations can be used in conjunction with all three resultant formulations, but they may not work. The following general perturbation mechanism applicable to Macaulay formulation was given by Canny in [4].

Let s be a new perturbation variable. Create a new set of polynomials $\mathcal{F}(s)$ by replacing each $p_j \in \mathcal{F}$ (for $1 \le j \le n$) by $p_j - sx_j^{d_j}$ and p_{n+1} by $p_{n+1} - s$. Compute the projection operator, $\varrho(s)$, of $\mathcal{F}(s)$ using Macaulay formulation. $\varrho(s)$ is known as the *Generalized Characteristic Polynomial* (*GCP*) and is a polynomial in s. The trailing coefficient (wrt. s) of *GCP* is a nontrivial projection operator of $\mathcal{F}$. Note that the size of the Macaulay matrix is same for $\mathcal{F}$ and $\mathcal{F}(s)$. For a proof of the fact that the projection operator so derived is not identically zero and it does vanish on all the affine zeros of the system, see [4]. We call this method **Macaulay/***GCP*. We now finish the previous example using this method.

Example: (Macaulay/*GCP*) For the example, $det(M_\sigma)$ and $det(N_\sigma)$, both are identically zero, so we compute the *GCP*. Construct $\mathcal{F}(s)$ and letting $M_\sigma(s)$ and $N_\sigma(s)$ stand for its Macaulay and the denominator matrix, $det(N_\sigma(s)) = det(N_\sigma - Is) = s(s-a)$. $det(M_\sigma(s)) = det(M_\sigma - Is)$ is a polynomial with 705 terms. $GCP = det(M_\sigma(s))/det(N_\sigma(s))$, is a polynomial with 462 terms. Finally, the projection operator, which here is the constant term of *GCP*, has 159 terms (in expanded form):

$$\varrho = c\,(a+b)\left(a^2c + abc - 4ab + 4b^2\right)\rho\,(a,b,c)$$

Notice that there are 3 extraneous factors. □

3.2 Rank Submatrix (*RSC*)

Another way to extract a nontrivial projection operator from degenerate resultant matrices is through their submatrices. Following method is applicable to all three formulations:

1. Set up the Resultant matrix (R) of $\mathcal{F}$.
2. If any column in R is linearly independent of the other columns, then return the determinant of any *rank submatrix* of R.
3. Else, this heuristic fails.

The resultant matrix, R, can be any of the three resultant matrices, giving three methods for computing a nontrivial projection operator, viz. **Macaulay/***RSC*, **Dixon/***RSC* and **Sparse/***RSC*. By *rank submatrix* we mean a maximal nonsingular submatrix, and its determinant can be computed by performing Gaussian elimination and returning the product of the (nonzero) pivots [18, 14]. In an earlier paper, we required a more complicated algorithm since we were deriving conditions for existence of solutions in $\mathbb{C}^n$, but for solutions in the *algebraic torus* $(\mathbb{C} - \{0\})^n$, this simple algorithm suffices. See [18] for details. Though there exist examples where the condition in step (2) is not true, they seem rare. In all examples we tried, R was degenerate many times, but the condition of step (2) was always satisfied. For the continuing example in this paper, we had not been able to compute a nontrivial projection operator using the original Macaulay and Dixon formulation. We do that now using Macaulay/*RSC* and Dixon/*RSC* respectively.

Example: (Macaulay/*RSC*) It is easily confirmed that the last column of the singular Macaulay matrix, M_σ, is linearly independent of the other columns, hence the rank of

n	d	Dix	Sparse[5]	Mac
2	2	5	15	15
2	3	12	33	36
2	4	22	66	66
3	2	14	54	56
3	3	55	198	220
4	2	42	179	210

Table 1: Matrix Sizes, Worst Total Degree Case.

n	d	Dix	Sparse	Mac
2	2	8	32	66
2	3	18	72	153
2	4	32	144	276
3	2	48	488	2024
3	3	162	1263	7140
4	2	384	*	91390

Table 2: Matrix Sizes, Worst Ind. Var. Deg. Case.

the matrix (9) decreases (becomes 8) on its removal. Projection operator is the product of the 9 pivots after Gaussian elimination and has 159 terms (in expanded form):

$$\varrho = ac(a+b)\left(a^2c + abc - 4ab + 4b^2\right)\rho(a,b,c).$$

Notice that there are 4 extraneous factors. □

Example: (Dixon/*RSC*) It is easily confirmed that the first column of the rectangular Dixon matrix is linearly independent of the other columns, hence the rank of the matrix (2) decreases (becomes 1) on its removal. We do Gaussian elimination on the Dixon matrix, and the Projection operator is the product of the 2 pivots and has 75 terms (in expanded form):

$$\varrho = -b\rho(a,b,c).$$

Again there is 1 extraneous factor but among all four methods for this example, the reader might observe that this method generates the least number of extraneous factors. □

4 Theoretical Comparison of Matrix Sizes

The size of the resultant matrix is critical in determining the efficiency of a given method. We compare the sizes of the three resultant matrices in the worst case using two measures: (i) the total degree of the input polynomials and (ii) the degree of input polynomials in each variable.

Total Degree based: For $1 \leq i \leq (n+1)$, let $tdeg(p_i, X) = d$. It is easily established that the sizes of Dixon, Macaulay and sparse resultant matrix are bounded by $\binom{nd}{n}$, $\binom{(n+1)d}{n}$ and $\binom{(n+1)d}{n}$ respectively. These formulas show that the Dixon matrix is smaller[6] than the other two. See Table 1 for typical worst case sizes in practice.

The total degree measure is not very fair to both Dixon and sparse resultants. Better bounds are obtained for these two if the measure for the degree of each polynomial in individual variables is considered.

Variable Degree based: For $1 \leq i \leq (n+1)$ and $1 \leq j \leq n$, let $d = deg(p_i, x_j)$. Sizes of the Dixon, Macaulay and sparse resultant matrices are bounded by $n!d^n$, $\binom{(n+1)nd}{n}$ and $((n+1)d)^n$ respectively. Using these formulas and Stirlings approximation for factorial (for asymptotically large n and d), it can be proven that Dixon matrix is smaller, by an exponential $(\mathbf{O}\left(e^{n+1}\right))$ factor, than sparse resultant matrix, which is smaller, also by an exponential $(\mathbf{O}\left(e^{n}\right))$ factor, than Macaulay, and transitively the Dixon matrix is smaller, by an exponential $(\mathbf{O}\left(e^{2n+1}\right))$ factor, than the Macaulay matrix. Typical worst case sizes are shown in Table 2. A (*) entry means the program ran out of memory.

The two worst case scenarios sketched above provide convincing arguments that the Dixon matrix is smallest of the three, followed by sparse and Macaulay is the largest. That these bounds are closely followed even in the non-worst case, real-world examples is evident from the columns labeled **Matrix Size** in Table 3. We will use this knowledge to carry out a comparison of various methods in a later section. For a comprehensive analysis of matrix sizes and comparison for multi-homogeneous systems, the reader may wish to consult [19].

5 Implementational details

Since the resultant matrices are symbolic (each entry being a polynomial in the parameters), their determinants (or rank subdeterminants) are computed using polynomial interpolation rather than direct Gaussian elimination.

Implementation of Macaulay/*GCP*: $M_\sigma(s)$ and $N_\sigma(s)$ (see section 3.1) for $\mathcal{F}(s)$ were constructed using our implementation in MAPLE. We used our C^{++} implementation of Zippel's sparse polynomial interpolation algorithm [25] to compute their determinants. Division of the two determinants and extraction of the projection operator from the *GCP* was done on MAPLE.

Implementation of *RSC* methods: We used our own MAPLE programs to construct the Macaulay and Dixon matrices, and Emiris' C program based on the algorithm in [11] to construct the sparse resultant matrix. The linear independence of any column in these matrices was checked probabilistically by substituting random numbers for parameters and performing the check on numeric (rather than symbolic) matrices. Finally, the rank subdeterminant was interpolated using our C^{++} implementation of Zippel's sparse polynomial interpolation algorithm.

Implementation of Zippel's interpolation algorithm: Given a resultant matrix, a *blackbox* is constructed which returns the value of the polynomial (determinant or the rank subdeterminant) for a particular value assignment to the parameters. This is done by (i) evaluating each entry of the symbolic resultant matrix after substituting parameter values then (ii) performing Gaussian elimination on the matrix (which now has integer entries) and finally (iii) returning the product of diagonal entries (for *GCP*) or nonzero pivots (for *RSC*).

[5] It should be pointed out that "sparse" is really a misnomer. Though the algorithm for sparse resultant is not completely insensitive to zero terms, the worst case is easily achieved for relatively sparse polynomials. Eg., for $n = 2, d = 3$, a set of three polynomials with the supports $(x^3, x^2, xy^2, y^3, 1)$ are relatively sparse (only 5 terms as opposed to possible 10), but the sparse matrix still has the worst case size.

[6] The formula for Dixon matrix size above is loose and the exact size is always smaller. Also, more efficient algorithms for sparse resultants use heuristics which may construct matrices of smaller size than the formula above indicates. However, such heuristics are not considered here because the order of the bound does not change.

Ex	Application	No. of Polys	Dixon			Macaulay			Sparse			Gröbner
			Matrix Size	Cnstr. Time	*RSC* Time	Matrix Size	*GCP* Time	*RSC* Time	Matrix Size	Cnstr. Time	*RSC* Time	*Macaulay* Time
1	Geom. Reason.	3	13	0.18	342	55	*	4608	34	0.3	566	65863
2	Formula Deriv.	5	14	0.36	76	330	*	*	86	9.8	4094	585
3	Formula Deriv.	3	5	0.08	*[7]	15	N/R	*[7]	14	0.2	*[7]	*
4	Geometry	3	4	0.08	1021[7]	15	*	*[7]	14	0.2	*[7]	15429
5	Random	4	6	0.06	*[7]	84	N/R	*[7]	27	0.7	*[7]	2207
6	Implicitization	3	10	0.1	0.58	45	3741	13.15	16	0.2	0.71	1
7	Implicitization	3	18	0.45	47	66	N/R	604	55	2	519	12530
8	Implicitization	3	18	9.33	126	153	*	94697	54	1.9	1369	70485
9	Random	4	7	0.21	220	56	*	9972	24	1.2	412	13508
10	Random	5	36	1.1	8130	1365	*	*	151	40	201558	*
11	Implicitization	3	11	0.06	3.01	36	N/R	32.58	33	0.7	31.04	57
12	Equilibrium	4	5	0.05	0.24	20	N/R	1.53	20	0.7	2.36	5
13	Vision	6	6	3.3	0.05	792	*	*	60	248	10.53	6
14	Comput. Bio.	3	8	0.067	0.27	28	11.18	0.68	16	0.2	0.33	1

Table 3: Empirical Data

The interpolation algorithm then loops through m stages (where m is the number of parameters), each time lifting one parameter by making a number of calls to the *blackbox*. At the end of the m^{th} stage, the polynomial has been completely interpolated. See [25] for details.

We performed all computations in a finite field modulo a large prime. If the exact projection operator is required, the computations can be performed in various finite fields modulo different primes until the product of the primes is larger than a predetermined coefficient bound. Finally exact coefficients can be lifted from their images in various finite fields using Chinese remainder theorem. If the coefficient bound is unknown (or too loose), lift the coefficients successively after each finite field computation until they stabilize.

In practice, the most expensive operation during interpolation is the *blackbox* computation, and the possibilities for improvement lie in reducing the number of calls to *blackbox*. Two main heuristics we used are as follows:

1. **Estimating individual variable degree bounds:** Zippel's algorithm assumes the same degree bound (say D) for each parameter at any stage. This assumption is costly for later stages if the actual degree of the parameter being lifted is less than D. Fortunately, when the first parameter is being lifted, this assumption is not too costly. Based on this observation, using a crude bound D, we interpolate univariate polynomials in all the parameters before starting multivariate interpolation. From these univariate polynomials in each parameter, we determine individual degree bound for each parameter. Later, when interpolating the multivariate polynomial, we use these individual degree bounds at each stage. This results in a speedup of the algorithm since it reduces the number of calls to *blackbox*.

2. **Order of interpolation:** Once the individual degree bound for each parameter is known, it turns out that the number of calls to *blackbox* is sensitive to the order in which the parameters are lifted. If parameters are lifted in increasing order of their individual degree bounds, fewer calls are made to the *blackbox* which resulted in a speedup.

Ex	Dix/ *RSC*	Mac/ *GCP*	Mac/ *RSC*	Sparse/ *RSC*
3	6.56	N/R	26.5	76.2
4	1.6	*	84	290
5	0.9	N/R	850	23.4

Table 4: Timings - direct determinant expansion in MAPLE.

Using these two heuristics, we are able to obtain a substantial speedup over a straightforward implementation of Zippel's algorithm.

6 Empirical Results and Comparison

All empirical results are listed together in Table 3. The 14 examples used in Table 3 can be obtained in MAPLE format by anonymous ftp from `ftp://ftp.cs.albany.edu /pub/saxena/examples.14`. All timings are in seconds on a 64Mb SUN Sparc10. A (*) in a *Time* column either means that the resultant cannot be computed even after running for more than a day or the program ran out of memory. An N/R in the GCP column means there exists a polynomial ordering for which the Macaulay and the denominator matrix are nonsingular and therefore GCP computation is not required. Besides resultant formulation timings, we also present timings for computing resultant using Gröbner basis construction with block ordering (variables in one block and parameters in another) using the *Macaulay* [1] system. Gröbner basis computations are also done in a finite field, in fact, with a much smaller characteristic (31991) than in the resultant computations. We also tried Gröbner basis using *GB* system of Faugere [13], but it does not support block ordering, and lexicographic ordering resulted in much inferior performance.

Examples 3, 4 and 5 consist of generic polynomials with numerous parameters and dense resultants. Interpolation methods are not appropriate for such examples and timings using straightforward determinant expansion in MAPLE are in Table 4.

[7] See Table 4 for timing using direct determinant expansion.

In all four methods, the time taken to compute the resultant depends on the cost of the two steps - (i) constructing the resultant matrix and (ii) interpolating the resultant from the resultant matrix.

Cost of Matrix Construction is insignificant (see the columns labeled Cnstr. Time in Table 3) for all formulations on all examples. It should be noted that the construction of Dixon matrices was done on MAPLE using a straightforward implementation without any improvements. An efficient implementation in C would further reduce the already insignificant Dixon matrix construction time. So, the total time to compute the resultant is dominated by the interpolation step.

Interpolation Cost (see section 5) depends on the cost of an individual *blackbox* call and the total number of calls to *blackbox*. The cost of an individual call to *blackbox* includes the cost of substituting values for parameters in the matrix and the cost of performing Gaussian elimination on the resulting matrix. Even though the cost of evaluating an individual entry in Dixon matrix is higher (because the entries are polynomials of degree at most $n+1$ in the coefficients of the original polynomials), the number of such evaluations is determined by the matrix size and is much smaller than for Macaulay and sparse matrices. Consequently, the time taken for substituting values for parameters in the Dixon matrix is smaller than the time taken for substitution in the Macaulay and sparse matrices.

The time taken by a call to *blackbox* is however totally dominated by Gaussian elimination after substitutions have been made into a matrix. Below we compare the four methods based on time taken to perform Gaussian elimination in a *blackbox* call and number of calls to the *blackbox*.

6.1 Dixon/*RSC* vs. Macaulay/*RSC* vs. Sparse/*RSC*

1. **Cost of Gaussian Elimination:** Once the substitution phase is over, the matrix entries are only entries from some finite field, and the cost of Gaussian elimination depends solely on the size of the matrices. Due to the much smaller size of the Dixon matrix, time taken for Gaussian elimination is much less than for the other two. Since Gaussian elimination dominates the *blackbox* call, the *cost of an individual blackbox call is the least for Dixon matrix*, followed by the sparse and finally the Macaulay matrix.
2. **Number of calls to *blackbox*:** The number of calls to *blackbox* depends on the structure of the polynomial being interpolated. If there are no extraneous factors, the polynomial being interpolated in all three cases is the same, and so the number of calls to *blackbox* is also the same in all three methods. If extraneous factors are present, then the cost depends on the number of terms in the projection operator. Usually the number of terms are less if there are less extraneous factors (note that this is not true in general).

Since the interpolation cost depends mostly on the two factors discussed above, *total time taken to interpolate the resultant is smallest for Dixon/RSC followed by Sparse/RSC and largest for Macaulay/RSC.*

This gradation of efficiency is reflected in Table 3. All the examples were completed using the Dixon/*RSC* method. Sparse/*RSC* also solved all the examples, but always took much longer than Dixon/*RSC*. Macaulay/*RSC* could not complete 2 problems and took much longer than Sparse/*RSC* (for most problems) and Dixon/*RSC* (for all problems).

6.2 *RSC* vs. *GCP*

1. **Cost of Gaussian Elimination:** Size of the resultant matrix of a polynomial system after perturbation is the same as that before perturbation (see section 3). So, the cost of Gaussian elimination is the same in *GCP* and *RSC* methods.
2. **Number of calls to *blackbox*:** Due to the introduction of a new perturbation variable, one more iteration (or stage) is need during interpolation for that variable. This results in more calls to all the procedures including *blackbox*. Secondly, due to the new variable, the number of terms in the polynomials at each stage of interpolation also becomes larger. This raises the number of calls to *blackbox* even further[8].

In practice, this rise in the number of calls to *blackbox* almost makes the *GCP* method impractical. On the other hand, cost of the *RSC* method remains the same. These observations about the *GCP* method are confirmed by comparing the entries of column labeled Macaulay/*GCP* against the columns Dixon/*RSC*, Macaulay/*RSC* and Sparse/*RSC*.

While working with Macaulay formulation, it is often (9 out of the 14 examples in Table 3) the case that one of the two methods, *RSC* or *GCP*, need to be resorted to, and analysis suggests that *RSC* should be chosen due to its efficiency.

The *GCP* method does have a distinct merit in that it can always extract the proper affine components of the variety, whereas other methods may fail. However, the failure of *RSC* methods seems to occur very rarely and they have been successfully applied to all nontrivial examples in the paper.

7 Conclusion

While working with nongeneric, nonhomogeneous polynomials, all four elimination methods generate extraneous factors in the projection operators (see the example used in this paper). So the only factor in making a choice between the four methods is efficiency. To this end, we find that the Dixon formulation coupled with rank submatrix computation is the fastest way to compute a nontrivial projection operator of a set of polynomials. The main reason is that Dixon matrix can be smaller, by a factor that is up to exponential in the number of variables, than the other two resultant matrices, viz., Macaulay and sparse.

The comparison of matrix sizes was done in the worst case scenario under two different measures. Being worst cases, they don't take into account the sparsity of polynomials encountered in real-world. Even though the real-world examples we presented show that the Dixon matrix is indeed much smaller than the other two, a theoretical analysis which takes into account sparsity needs to be done. Our preliminary work in this direction [19] shows that, like the sparse resultant formulation, Dixon formulation also exploits the Newton polytopes and takes into account the sparsity of the input polynomials, though in a different fashion.

Extraneous factors occur in all methods in this paper, however, it is not clear which method generates least factors. Any bounds on the number of extraneous factors generated by the four methods will be very useful. Another important research direction is to discover ways of eliminating extraneous factors from these methods without compromising their efficiency.

[8] It may be possible to avoid the *GCP* and directly compute its trailing coefficient, but this is likely to require as many *blackbox* calls.

Acknowledgement: We thank Ioannis Emiris for giving us a preliminary version of his sparse resultant program.

References

[1] Bayer D., Stillman M., *Macaulay User's Manual*, Cornell University, Ithaca, NY.

[2] Bernshtein D.N., The Number of Roots of a System of Equations, *Funktsional'nyi Analiz i Ego Prilozheniya*, 9(3):1-4.

[3] Buchberger B., Gröbner bases: An Algorithmic Method in Polynomial Ideal Theory, *Multidimensional Systems Theory*, N.K. Bose, ed., D. Reidel Publ. Co., 1985.

[4] Canny J., Generalized Characteristic Polynomials, *Journal of Symbolic Computation*, 9:241-250.

[5] Canny J. and Emiris I., An Efficient Algorithm for the Sparse Resultant, *Proc. of AAECC 93*, LNCS 263, Springer-Verlag, May 1993, pp. 89-104.

[6] Canny J. and Pedersen P., An Algorithm for Newton Resultant, Technical Report, *Cornell University*, CornellCS:TR93-1394, 1993.

[7] Cayley A., On the Theory of Elimination. *Cambridge and Dublin Mathematical Journal*, III, 1865, 210-270.

[8] Chionh E., *Base Points, Resultants and Implicit Representation of Rational Surfaces.* Ph.D. thesis, Dept. Comp. Sci., Univ. of Waterloo, 1990.

[9] Collins G.E., The Calculation of Multivariate Polynomial Resultants. *JACM*, 18 (4), (1971), 515-532.

[10] Dixon A.L., The Eliminant of Three Quantics in Two Independent Variables. *Proc. London Mathematical Society*, 6, 1908, 468-478.

[11] Emiris I. and Canny J., A Practical Method for the Sparse Resultant, *Proc. ACM ISSAC*, 183–192, Kiev, July 1993.

[12] Emiris I. and Canny J., Efficient Incremental Algorithms for the Sparse Resultant and the Mixed Volume, Technical Report, *Univ. of California, Berkeley*, 1994.

[13] Faugère J.C., Gb Reference Manual, PoSSo project, *Institut Blaise Pascal*, France.

[14] Gantmatcher F.R., *Matrix Theory*, vol. 1. Chelsea Publishing, 1959, Chap. 2.

[15] Grigoryev D.Yu. and Chistov A.L., *Sub-exponential Time Solving of Systems of Algebraic Equations.* LOMI Preprints E-9-83 and E-10-83, Leningrad, 1983.

[16] Gelfand I.M., Kapranov M.M. and Zelevinsky A.V., *Discriminants, Resultants and Multidimensional Determinants*, Birkhäuser, Boston, 1994.

[17] Kapur D. and Lakshman Y.N., Elimination Methods: an Introduction. *Symbolic and Numerical Computation for Artificial Intelligence* B. Donald et. al. (eds.), Academic Press, 1992.

[18] Kapur D., Saxena T. and Yang L., Algebraic and Geometric Reasoning using Dixon Resultants, *Proc. ACM ISSAC*, Oxford, England, July 1994.

[19] Kapur D and Saxena T., Sparsity Considerations in the Dixon Resultant Formulation, manuscript under preparation.

[20] Macaulay F.S., *The Algebraic Theory of Modular Systems*, Cambridge Tracts in Math. and Math. Phys., 19, 1916.

[21] Sederberg T. and Goldman R., Algebraic Geometry for Computer-Aided Design, *IEEE Computer Graphics and Applications*, Vol. 6, No. 6, pp. 52-59, 1986.

[22] Sturmfels B., Sparse Elimination Theory, *Proc. Computat. Algebraic Geom. and Commut. Algebra*, D. Eisenbud and L. Robbiano, eds., Cortona, Italy, June 1991.

[23] Sturmfels B. and Zelevinsky A., Multigraded resultants of the Sylvester type, *Journal of Algebra*, 1992.

[24] Weyman J. and Zelevinsky A., Determinantal Formulas for Multigraded Resultants, *Journal of Algebraic Geometry*, pp 569-597, 1994.

[25] Zippel R. *Effective Polynomial Computation*, Kluwer Academic Publishers, Boston, 1993.

An Algorithm for Converting a Degree Gröbner basis to a Lexicographic Gröbner basis*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222

Rough Draft

Abstract

An algorithm for converting a Gröbner basis, G_1 of an arbitrary ideal from one ordering $<_1$ to a Gröbner basis G_2 with respect to another target ordering $<_2$ is presented. Its behavior does not depend upon the dimension of the input ideal and it works for any positive or zero dimensional ideal. The target ordering can be any ordering including elimination orderings such as pure lexicographic.

The algorithm is similar to the *FGLM* algorithm for basis conversion of zero dimensional ideals in the following sense: it incrementally constructs (i) a set T of terms, (ii) a set G of polynomials by computing the normal forms of the terms in T with respect to $<_1$ and solving some linear equations similar to those in *FGLM*. However the order in which terms are generated in T is not completely determined by the target ordering $<_2$. If $<_2$ does not have the property that for every term t, there are only finitely many terms smaller than t, then an enumeration ordering $<_e$ is built using $<_1$ and $<_2$ such that it has that property. For termination condition, it is first checked whether all the polynomials in G_1 reduce to 0 modulo G, and if so, then check if G is a Gröbner basis. We prove that there is a T which will be generated within a finite number of steps for which the termination condition would succeed.

The algorithm has been implemented in GEOMETER and has been successfully tried on a number of examples. It seems to work better than computing a Gröbner basis directly for ideals with small dimensions. Planned extensions include development of heuristics for developing good enumeration orderings as well as an efficient termination check. For the latter, the use of a Hilbert function of an ideal is being investigated.

1 Introduction

A Gröbner basis algorithm is considered as one of the most efficient elimination algorithms in practice for solving many problems related to polynomial ideals as well as the algebraic varieties defined by polynomial ideals. There has been a great deal of interest to obtain efficient algorithms for computing Gröbner bases. This is especially true for computations under an

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

elimination ordering of the variables as a Gröbner basis with respect to an elimination ordering is the most useful computational object for many problems. It is generally agreed that Gröbner basis computations are the hardest for elimination orderings of the variables like pure lexicographic ordering. For zero-dimensional polynomial ideals, this problem was tackled in a novel manner in [1]. Instead of computing a Gröbner basis with respect to an elimination ordering directly, Faugere et al computed a Gröbner basis w.r.t. some other ordering, like graded lexicographic (also popularly known as the degree ordering) or graded reverse lexicographic ordering, which is easy to compute, and the basis was transformed into a basis w.r.t. the desired elimination ordering. This transformation was performed using linear algebra techniques. Impressive results were reported using this technique of computing Gröbner bases of ideals w.r.t. elimination orderings in [1]. We will refer to it as the FGLM algorithm.

It has not been clear how the FGLM algorithm can be generalized to positive dimensional ideals. If the algorithm is applied on a Gröbner basis of a positive dimensional ideal, then it does not terminate. The termination of the algorithm depends upon the property that for a zero-dimensional ideal, the set of all reduced terms with respect to a Gröbner basis in its monomial ideal is finite. For a positive dimensional ideal, this set of reduced terms in its monomial ideal is infinite, consequently the FGLM algorithm goes on indefinitely enumerating terms from the reduced set forever.

We present an algorithm whose termination does not depend upon the finiteness property of the set of reduced terms with respect to a Gröbner basis in a monomial ideal. Our algorithm, instead, depends on the property that a Gröbner basis of a polynomial ideal with respect to any admissible ordering is finite. Hence the algorithm works for ideals of any dimension. This algorithm can be viewed a generalization of the algorithm developed in [1] for zero dimensional ideals.

The paper is organized as follows. The second section first reviews the algorithm for zero dimensional ideals as developed in [1]. We introduce the main new idea, using which the FGLM algorithm can be generalized so that it no longer depends on the finiteness of the set of reduced terms with respect to the target Gröbner basis in a monomial ideal. The third section discusses a naive, simple basis conversion algorithm which works for arbitrary ideals. In the fourth section, an efficient version of the algorithm is discussed. The fifth section gives the correctness argument for the algorithm. In the sixth section, heuristics are discussed to improve the performance of an implementation of the algorithm. In section seven, we present two simple examples illustrating how the algorithm transforms Gröbner bases w.r.t. graded lexicographic orders (total degree orders) to the bases for pure lexicographic orders. Finally, in section eight, some preliminary empirical results are reported.

2 A Generalization of the Zero Dimensional Algorithm

The basis conversion algorithm for zero dimensional ideals in [1] mainly relied on the fact that there are only a finite number of reduced terms w.r.t. any Gröbner basis of a zero dimensional ideal. This gives rise to a staircase like volume which encloses all the reduced terms. We give below the main steps of the algorithm given in [1]. The algorithm is called *FGLM* and it takes as input, a Gröbner basis G_1 of the ideal with respect to an ordering $<_1$. Its output is G_2, the reduced Gröbner basis w.r.t. a target ordering $<_2$. In the following, the subscript of $<_i$ means that the comparision is done using the ordering $<_i$. For example, larger$_{<_2}$ means that larger

with respect to $<_2$. Also, $\mathcal{NF}_{<_1}(t)$ is a function that computes the normal form of a term t modulo $G_1 and <_1$.

FGLM(G_1)

Input : A Gröbner basis w.r.t. $<_1$.

Output : The reduced Gröbner basis G_2 w.r.t. $<_2$.

Algorithm :

begin
$B := \phi$
$InitialTerms := \phi$
$t_0 := 1$
$i := 1$
while there exists a term larger$_{<_2}$ than t_{i-1} which is not
divisible by any element of $InitialTerms$ do
    $t_i :=$ the smallest$_{<_2}$ term larger$_{<_2}$ than t_{i-1} which is
    not divisible by any element of $InitialTerms$
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j's$ be the solutions
        $B := B \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $InitialTerms := InitialTerms \cup \{t_i\}$
    else
        $i := i + 1$
    fi
od
return(B)
end.

The algorithm is guaranteed to terminate because for zero-dimensional ideals, the number of reduced terms for its Gröbner basis w.r.t. any admissible ordering is finite.

This algorithm cannot be used for positive dimensional ideals because in general, it does not terminate; there could potentially be infinitely many reduced terms with respect to a Gröbner basis of a positive dimensional ideal. The enumeration process could go on for ever and we may never get a Gröbner basis. For example, consider the case when the reduced Gröbner basis w.r.t. the pure lexicographic order (with $x > y > z$) is $\{x + yz, y^2z - z\}$. If $FGLM$ was run on a Gröbner basis of the same ideal with a degree ordering, terms purely in z would be enumerated and this enumeration would go on forever since there is no polynomial purely in z in this one dimensional ideal!

We will now present a generalization of the $FGLM$ algorithm and use this generalization to get an algorithm for the positive dimensional case. The main idea is to parameterize the $FGLM$ algorithm by enumerating terms only from a certain finite set T. So the inputs to this algorithm (which we call Solve) are a Gröbner basis w.r.t. $<_1$ and also an additional input which is a set of terms T. We will not specify what the output of this program is, at this point, but we will discuss some properties of it after the description.

Solve(G_1, T)

Input : A Gröbner basis w.r.t. $<_1$ and a set of terms T.

Algorithm :

begin
$B := \phi$
$InitialTerms := \phi$
$t_0 :=$ smallest$_{<_2}$ term in T
$i := 1$
while there exists a term larger$_{<_2}$ than t_{i-1} in T which is not
divisible by any element of $InitialTerms$ do
    $t_i :=$ the smallest$_{<_2}$ term larger$_{<_2}$ than t_{i-1} in T which is
    not divisible by any element of $InitialTerms$
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j's$ be the solutions
        $B := B \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $InitialTerms := InitialTerms \cup \{t_i\}$
    else
        $i := i + 1$
    fi
od
return(B)
end.

This algorithm is very similar to the *FGLM* algorithm outlined above except that it enumerates *only* those terms which are in a predetermined set T. We refer to this algorithm as *Solve*. Some useful properties of this algorithm are:

- *If* T is the set of all the terms in the reduced Gröbner basis with respect to the target ordering, then when algorithm returns G_2, the desired Gröbner basis w.r.t. $<_2$. Obviously, having T to be a *superset* of all the terms in the reduced Gröbner basis with respect to the target ordering would also suffice.
- In case T does *not* contain all the terms in the reduced Gröbner basis w.r.t. $<_2$, then the algorithm may not give us a Gröbner basis, but B generates a subideal of the ideal of the input Gröbner basis.

These properties of this algorithm enables us to develop an algorithm for positive dimensional ideals in the following sections.

3 The positive dimensional case

As discussed earlier, the *FGLM* algorithm would not terminate for ideals of positive dimensions because its termination relies on the fact that the set of reduced terms with respect to the

Gröbner basis is finite. In fact, the algorithm would not give any useful information for a lexicographic ordering if the ideal did not include purely a polynomial in the lowest variable in the ordering.

On the other hand, the utility of the output as well as the efficiency of the generalized algorithm *Solve* depends upon the finiteness of T as well as an appropriate choice for T. Since the reduced Gröbner basis is finite for any admissible ordering, if T can be built by enumerating terms in such a way that eventually all terms appearing in the reduced Gröbner basis w.r.t. the desired target ordering can be included in T, *Solve* would then return a Gröbner basis. So the parameter to *Solve*, the set of terms T may be incrementally generated by an *enumeration ordering.* Instead of requiring that the terms be enumerated in the same order as in the target ordering, we only require that terms are enumerated in such an order that we eventually generate enough terms to include terms appearing in the reduced Gröbner basis with respect to the target ordering.

A good *enumeration ordering*, $<_e$ guarantees that eventually, T will be a superset of all the terms in the reduced Gröbner basis w.r.t. $<_2$. But how do we know that we have generated enough terms so that the set of terms that we have is a superset of the set of terms in the reduced Gröbner basis? In other words, how do we reconize that T is *good enough*, even though we are guaranteed that within a finite amount of time, it would be? This is the termination condition. Clearly it cannot be that every term greater than the terms generated so far in the enumeration ordering is reducible (as in 0-dimensional $FGLM$). For positive dimensional ideals, there are infinitely many reduced terms. This is an interesting research issue to be further investigated. Currently, there are two ideas: (i) Run the algorithm *Solve* each time a term from the enumertation ordering is produced and test if the output of *Solve* is a Gröbner basis and also if it reduces all the polynomials of G_1 to zero. (ii) the use of Hilbert function (polynomial) for checking the number of reduced terms of various degrees. Below we sketch the algorithm that we have proposed here. A more efficient and detailed algorithm will be given in the next section. Assume that we have a good *enumeration ordering*, $<_e$. Notice that any sequential (or degree compatible) ordering would suffice.

Algorithm(G_1)

1. $T := \{1\}$
2. $B := \phi$
3. while *Gröbner ?*(B) is false do
   (a) $t :=$ the smallest term larger$_{<_e}$ than all the terms in T
   (b) $T := T \cup \{t\}$
   (c) $B := Solve(G_1, T)$
4. return(B)

Here, the predicate *Gröbner ?*(B) returns true if B is a Gröbner basis, else it returns false. Also, $reduces_B(G_1)$ returns true if all the polynomials in G_1 reduce to zero modulo B. We now discuss the two main steps in the general algorithm:

- How to generate T systematically and efficiently based on the input Gröbner basis and the target ordering so that eventually all terms in the output Gröbner basis would be included?

- How can the two tests – whether a given basis generates the same ideal as the input basis, and whether a given basis is a Gröbner basis, be performed efficiently?

Incrementing T

If the target ordering is degree compatible, then it could be used as an enumeration ordering. A degree compatible ordering has the nice property that there are finitely many terms smaller than any term in the ordering. For arbitrary ideals, elimination orders such as lexicographic orderings are not likely to work as enumeration orderings because in an elimination ordering, there can be infinite terms below the smallest head-term of a Gröbner basis with respect to the target order. For the example above, there are infinitely many terms lower than the smallest head term y^2z. In general, it should be possible to make use of the structure of the input Gröbner basis to pick an appropriate ordering which is a mixture of a degree compatible and elimination orderings such as block-orderings. This issue is under further investigation. If the target ordering is degree compatible, then it is used as an enumeration ordering. If the target ordering is not degree compatible but the input Gröbner basis is computed using a degee compatible ordering, then the input ordering is used for enumeration. Otherwise, a degree ordering (with input order to break ties) is used. We call this order $<_e$. So the strategy for incrementing T is as follows:

Start with an empty T and increment it by the terms in an ascending $<_e$ order.

Assume that T_s is the set of all terms which are smaller, w.r.t. $<_e$, than a term s. Let the largest term, w.r.t. $<_e$, which appears in a reduced Gröbner basis with respect to the target ordering be q. Since T_q is finite, it is easy to see that T *will* equal T_q within a finite number of steps. This guarantees that the general algorithm will terminate.

Because of the relaxation of this requirement about an enumeration ordering being different from a target ordering, when a new term is enumerated, it is not necessarily greater in the target ordering than previously enumerated terms. So when we solve for reducible enumerated terms, we may get more than one polynomials to be potential candidates for being elements of a Gröbner basis with respect to the target ordering.

Termination test

Two tests must succeed for the algorithm to terminate. The basis B generated by the general algorithm must generate the same ideal as the input basis, and B should be a Gröbner basis with respect to the target ordering. One way to perform the first test is to check whether all polynomials in the input Gröbner basis reduce to 0 using B; this would ensure that B generates the same ideal as the input Gröbner basis since every element in B is in the ideal of the input Gröbner basis.

For the second test, one possibility is to check whether all nontrivial S-polynomials (critical pairs) of B reduce to 0. This test can be performed incrementally making use of the information from previous unsuccessful attempts of the test.

There are perhaps other ways of testing whether a Gröbner basis of the same ideal has already been generated. One possibility is to use the Hilbert function of an ideal using its Gröbner basis given as the input. The main idea is to count the number of irreducible terms for every degree, and then use this information to decide whether a Gröbner basis with respect to the target ordering has been generated or not. This approach has not yet been fully studied

in this framework, but it may have interesting applications. We are particularly concerned about the difficulty in computing a Hilbert function of positive dimensional ideals.

In the next section, we present a more efficient variant of this algorithm. In another section, we present heuristics using which this algorithm can be made faster.

4 Basis conversion algorithm

The following algorithm, *BasisConvert*, generates a Gröbner basis G_2 w.r.t. a target ordering $<_2$, given a Gröbner basis G_1 w.r.t. a term order $<_1$ The *Solve* algorithm has been slightly changed in this section for efficiency considerations.

BasisConvert(G_1)

Input : A Gröbner basis G_1 w.r.t. $<_1$.

Output : A Gröbner basis, G_2 w.r.t. $<_2$ of the ideal generated by G_1.

Algorithm :

```
begin
G2 := φ;
InitialTerms := φ;
ReducedTerms := φ;
t := 1;
While not(TerminationTest(G1, G2)) do
    t := NextTerm(t, InitialTerms);
    ReducedTerms := ReducedTerms ∪ {t};
    G' := Solve(G1, ReducedTerms);
    if G' ≠ φ then
        G2 := G2 ∪ G';
        InitialTerms := InitialTerms ∪ HeadTerms(G');
        ReducedTerms := RemoveMultiples(ReducedTerms, HeadTerms(G'));
    fi;
od;
Return(G2);
end;
```

The input/output specifications of some of the procedures used in the above algorithm are given below. Procedure *Solve* is described fully. The enumeration order, $<_e$ is defined to be $<_2$ if $<_2$ is degree compatible, $<_1$ if $<_1$ is degree compatible, else it is defined as follows:

$$a <_e b \; if \; degree(a) < degree(b) \; or (degree(a) = degree(b) \; and \; a <_1 b)$$

NextTerm(t, S)

Input : A term t and a set of terms S.

Output : The smallest term larger than t w.r.t. $<_e$ and not divisible by any term in the set S.

HeadTerms(S)

Input : A set of polynomials S.

Output : The set of headterms of all polynomials in S w.r.t. $<_2$.

RemoveMultiples(T, S)

Input : A set of terms T and another set of terms S.

Output : The set $T - multiples(S)$ where $multiples(S)$ is the set of all multiples of the terms in S.

Solve(G_1, T)

Input : A set of terms T and any term order $<$.

Output : The set B of all polynomials p which satisfy the following properties:

1. p is in the ideal generated by G_1,
2. p is a linear combination of the elements of T and
3. There is no polynomial, p' which is in the ideal generated by G_1 and is a linear combination of terms in T, but $HeadTerm(p')$ divides some term of p.

Algorithm :

begin
$B := \phi$
$t_0 := 1$
$i := 1$
while there exists a term larger than t_{i-1} in T w.r.t. $<$ do
    $t_i :=$ the smallest term larger than t_{i-1} in T (w.r.t. $<$)
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j's$ be the solutions
        $S := S \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $T := RemoveMultiples(T, t_i)$
    else
        $i := i + 1$
    fi
od
return(S)
end.

TerminationTest(G_1, G_2)

Input : A Gröbner basis w.r.t. $<_1$ viz. G_1 and the set of polynomials G_2.

Output : $True$ if $G_{<_2}$ is a Gröbner basis of the ideal generated by $G_{<_1}$ w.r.t. $<_2$, else $False$.

Algorithm : Test if every polynomial in $G_{<_1}$ normalizes to zero w.r.t. $G_{<_2}$. If they don't then return false else test if $G_{<_2}$ is a Gröbner basis. More details and heuristics for this procedure will be outlined in section 6.

5 Proof of correctness of *BasisConvert*

5.1 Proof of correctness of *Solve*

For a fixed T and an ordering $<_2$, the set of all polynomials which satisfy all the conditions in the output of *Solve* is unique (call it $P_{T,<_2}$). Let $P_{T,<_2} = \{p_1, p_2, ..., p_n\}$. Also let this be an ordered set, i.e., $ht_{<_2}(p_i) <_2 ht_{<_2}(p_j)$ if $i < j$. The following lemma on $P_{T,<_2}$ proves the correctness of procedure *Solve*:

Theorem 5.1 *For a given T and an ordering $<$, $Solve(G_1, T)$ produces all the polynomials in $P_{T,<_2}$, in that order.*

Proof: We use mathematical induction on i (where p_i is the i^{th} polynomial in $P_{T,<_2}$) to prove this lemma.

Base case Let q_1 be the first polynomial that *Solve* produces. q_1 satisfies the first requirement in the output of *Solve* because of the fact that the linear equation which resulted in q_1 was solvable implies that q_1 is in the ideal generated by G_1. The second condition is also satisfied since the linear equation which resulted in q_1 was only in terms of the normal-forms of the terms in T. The third condition is satisfied because in the procedure *Solve* we are enumerating terms from T in $<_2$ order. Hence if there was any polynomial smaller than q_1 which was in the ideal and could be formed using terms in T, it's corresponding linear equation would have been solvable so, we would already have it and q_1 wouldn't be the first one. For any polynomial's head term to divide some term in q_1, that polynomial must be smaller than q_1. Since we don't *have* any polynomial with smaller headterm w.r.t. $<$, the third condition is also satisfied. Hence q_1 is the smallest polynomial which satisfies all the three conditions in the output specification of *Solve* so $q_1 = p_1$.

Induction hypothesis Let us assume that the first i polynomials produced by the algorithm are $p_1 \cdots p_i$.

We now prove that the $(i+1)^{th}$ polynomial produced by *Solve* (q_{i+1}) is the same as the $(i+1)^{th}$ polynomial of $P_{T,<_2}$ i.e., p_{i+1}. This can be proven in the same way as the base case by replacing T by T_i and $P_{T,<_2}$ by $P_{T_i,<_2}$ in the proof, where

$$T_i = T - \{multiples(ht_{<_2}(p_1)), multiples(ht_{<_2}(p_2)), \cdots, multiples(ht_{<_2}(p_i))\}$$

and realizing the simple fact that

$$P_{T_i,<_2} = \{p_{i+1}, p_{i+2}, \cdots, p_n\}$$

□

This concludes the proof of correctness of *Solve*.

5.2 Proof of correctness of *BasisConvert*

We will now prove that within a finite number of steps, *BasisConvert* will return a Gröbner basis of the ideal w.r.t. the target ordering. We prove a lemma which establishes that within a finite number of steps, T will contain all the terms of the reduced Gröbner basis and then the Gröbner basis will be computed by invoking *Solve* at this point.

Lemma 5.2 *Let s be a term. In a finite number of steps of the algorithm, all the terms which are smaller than s w.r.t. $<_e$ will either be in the set $ReducedTerms$ or be divisible by some term in the set $InitialTerms$.*

Proof : We say that a term is being considered by the procedure $NextTerm$ when it is either produced by $NextTerm$ or it is skipped by $NextTerm$. Notice that a term is said to have been produced by $NextTerm$ if it is actually returned by the procedure.

Since the enumeration is being done in $<_e$ order which is degree compatible, s will be considered in a finite number of steps. At that point, all the terms smaller that s would already have been considered by $NextTerm$. For each smaller term s_i (all of which were considered), we can do the following analysis:

1. s_i was not produced. Since it was not produced, it must be divisible by some term in $InitialTerms$, and will continue to be so since nothing is ever removed from $InitialTerms$.
2. s_i was produced, hence one of the following will be true:
   (a) s_i was put in the set $InitialTerms$ (and then will continue to be in that set) or
   (b) s_i was put in $ReducedTerms$. Hence, one of the following will be true:
      i. s_i will continue to be in $ReducedTerms$, or
      ii. s_i will become divisible by some term in $InitialTerms$ at a later point of time, and be removed from $ReducedTerms$.

Hence, when s is being considered, all terms smaller than s would either be in $ReducedTerms$ or be divisible by some element of $InitialTerms$. □

We will denote the set of all terms smaller than or equal to a term s w.r.t. $<_e$ to be T_s. Also $RG_<$ denotes the reduced Gröbner basis of an ideal w.r.t. the ordering $<$. If S is a set of polynomials, then $tail(S)$ is the difference of the set of all terms in all the polynomails of S and the set of head terms of all polynomials in S..

Lemma 5.3 *Let $T(RG_{<_2})$ be the set of all terms in the reduced Gröbner basis of the ideal generated by the original Gröbner basis G_1 w.r.t. the target term ordering. In a finite number of steps of the algorithm, all the following become true together:*

1. $T(RG_{<_2}) \subseteq ReducedTerms \cup InitialTerms$
2. $tail(RG_{<_2}) \subseteq ReducedTerms$
3. $\forall p \epsilon RG_{<_2}, ht(p) \epsilon InitialTerms \Rightarrow \exists q \epsilon G_2 ht(p) = ht(q)$

4. $\forall p \epsilon RG_{<_2}, ht(p) \epsilon ReducedTerms \Rightarrow p \epsilon Solve(G_1, ReducedTerms)$

Proof: From the properties of $Solve$, we can assume that at all the polynomials in G_2 are from the ideal generated by G_1.

We first prove that within a finite number of steps, the first condition will be satisfied. Then we prove that the first condition implies all the remaining conditions, hence proving the lemma.

1. First condition becomes true in a finite number of steps: Let the largest term in $T(RG_{<_2})$ w.r.t. $<_e$ be s. Then by previous lemma, within a finite number of steps, every term of $T_s(\supseteq T(RG_{<_2}))$ will either be in $ReducedTerms$ or be divisible by some term in $InitialTerms$. But all the terms in $InitialTerms$ are head terms of some polynomial in the ideal. Since there is no term $t \epsilon T(RG_{<_2})$ which can be divisible by the head term (say h) of another polynomial in the ideal unless $h = t$, all the terms of $T(RG_{<_2})$ must either be in $ReducedTerms$ or be in $InitialTerms$ (as opposed to simply being divisible by some term in $InitialTerms$).

2. First condition implies the second condition: Previous condition implies that all the terms in the tail of every polynomial in $RG_{<_2}$ must either be in $ReducedTerms$ or be in $InitialTerms$. But any term in the tail of a polynomial in the reduced Gröbner basis can never be the head term of another polynomial in the ideal (since this contradicts the fact that $RG_{<_2}$ is reduced). Hence all the terms in the tail of all the polynomials of $RG_{<_2}$ must be in the set $ReducedTerms$.

3. Third condition is always true: The third condition is an invariant of the main loop of the algorithm. If any term is in $InitialTerms$, it is clear from the algorithm that it must be the case that there is a polynomial in G_2 whose head term is the same as this term (since we only put head terms in $InitialTerms$).

4. Previous conditions imply the fourth condition: If the head term of any polynomial in $RG_{<_2}$ is in $ReducedTerms$, then, since the tail of this polynomial must be in $ReducedTerms$ (by the second condition), this polynomial would satisfy the second condition in the output of $Solve$. It will also satisfy the first condition of the output of $Solve$ since it is obviously in the ideal. Moreover, since this polynomial is a member of the reduced Gröbner basis, there is no polynomial at all in the ideal whose head term divides any term in this polynomial (unless it is larger than this polynomial w.r.t. $<_2$), it also satisfies the third condition of the output of $Solve$. Since it satisfies all the three conditions of the output of $Solve$, it must be produced by $Solve$ when invoked with $ReducedTerms$.

□

Theorem 5.4 *In a finite number of steps, G_2 would be a Gröbner basis for the ideal generated by G_1.*

Proof : By the previous corollary, after a finite number of steps, the head terms of all polynomials in the reduced Gröbner basis are either in $ReducedTerms$, and then by the fourth condition of the previous corollary they will be produced (and hence be put in G_2), or in

InitialTerms, in which case, by the third of the previous corollary, there already exists a polynomial in G_2 which has the same head term as this one. So for all polynomials in $RG_{<_2}$, there exists a polynomial in G_2 which has the the same head term as this one. Hence by Proposition 5.38 of [2], G_2 must be a Gröbner basis. □

This last theorem proves the correctness of *BasisConvert*. Notice that *BasisConvert does not* necessarily produce the reduced Gröbner basis w.r.t. $<_2$. All that we have proved is that it produces a Gröbner basis. The reduced Gröbner basis can easily be obtained from this Gröbner basis by deleting all polynomial whose head terms are multiples of the head terms of other polynomials and then interreducing them.

6 Optimizations

The most time-consuming steps in the algorithm are:

- normal form computations of terms,
- solving linear equations to generate polynomials expressed using (possibly) reduced terms with respect to G_2 in the ideal,
- testing for ideal equality, and
- testing for the output to be a Gröbner basis.

6.1 Generating normal forms of terms

Computing the normal form of a term t is done in two steps.

1. Generate a polynomial (which we call a *subnormal form*) whose normal form is the same as that of t. An objective here is to use normal forms already computed and saved in a hash-table.
2. Compute the normal form of a subnormal form.

- **Generating a subnormal form.** Since terms are being generated in a degree compatible order by the procedure $NextTerm$, when a term t is generated, the set $ReducedTerms$ contains terms, all of which are smaller than t w.r.t. $<_e$. For every variable x_i which divides t, it must be the case that the term $\frac{t}{x_i} \in ReducedTerms$ (as otherwise , there must exist a variable x_k such that $\frac{t}{x_k}$ is divisible by some term in $InitialTerms$, and hence, so is t so t should not have been generated anyway). Let x_s be the smallest variable which divides the term t. The normal form of t, $\mathcal{NF}(t)$, is computed as follows:

$$\mathcal{NF}(t) = Normalize(x_s \times \mathcal{NF}(\frac{t}{x_s}))$$

The polynomial $x_s \times \mathcal{NF}(\frac{t}{x_s})$ is called a *subnormal form* of t.

- **Computing the normal form of a subnormal form.** The procedure *Normalize* is invoked on a polynomial such that all the terms in it are smaller than t w.r.t. $<_1$. Since the summation of the normal forms of terms gives the normal form of the whole polynomial, some of the terms of this polynomial which are smaller than t w.r.t $<_e$ can be replaced by their normal forms, which have already been computed and stored. It turns out that in practice, if the normal forms of the terms *not* smaller than t w.r.t $<_e$ are also computed and stored, they can be used in subsequent computations making the normalization process more efficient. If $<_1$ coincides with the enumeration term order $<_e$, i.e. $<_1$ is degree compatible, then we will have normal forms of all the terms of the polynomial which are not divisible by any term in *InitialTerms*, and this further simplifies the computation of normal forms.

6.2 Solving Linear Equations

Instead of invoking *Solve* after every term is generated, it is more efficient to invoke only after all the terms of a certain degree have been generated. For every degree, we invoke *Solve*, and if new polynomials are generated to be included in the new basis, normalize G_1 w.r.t. the new basis collected till that point, and if all the polynomials in G_1 reduce to zero, test the new basis for a Gröbner basis. For solving a system of equations, the optimization used for the zero-dimensional case ([1]) can be used.

6.3 Normalizing G_1 and testing G_2 for Gröbner basis

- **Partial normalization of G_1.** As should be obvious, the test for checking whether every polynomial in G_1 reduces to 0 using the new basis is done incrementally when new polynomials are added to the new basis. If during this test, a polynomial in G_1 does not reduce to 0, *false* is returned. Next time the same test is invoked, there is no need to check again the polynomials in G_1 which earlier reduced to 0 since polynomials are not being deleted from the new basis. The test can resume from a normal form of the last polynomial in G_1 that did not reduce to 0.

- **Testing G_1 for Gröbner Basis.** This test is also done incrementally. Firstly we need to consider only those polynomials in the new basis whose head-terms do not reduce by the other polynomials in the new basis. For any distinct pair of such polynomials in the new basis, if the associated S-polynomial (critical pair) does not reduce to 0, *false* is returned. Next time this test is invoked, we can resume from this S-polynomial itself (assuming the head term of any of the polynomials in the new basis from which the S-polynomial was generated does not reduce by any new polynomial added to the new basis). Finally, the criteria (like Buchberger's two criteria [4]) which minimize the number of S-polynomials needed to be reduced in the classical algorithm can be used to optimize the number of S-polynomials which must be considered for a Gröbner basis test.

Because of these optimizations, polynomials in G_1 are reduced exactly once.

7 Examples

Below we illustrate the main steps of the general algorithm on a couple of very simple examples.

Example 1: Consider the following Gröbner basis of a one dimensional ideal w.r.t. the total degree order with variables $x > y > z$.

$$G_1 = \{yz + x, xy + z, x^2 - z^2\}.$$

The general algorithm can be used to transform G_1 into G_2 where $<_2$ is the pure lexicographic order with $x > y > z$. In the following trace, we present the snapshots of the values of of the sets $ReducedTerms$, $InitialTerms$, G_2 after all the terms of a certain degree have been generated and the call to $Solve$ has been made. Finally it is indicated how much of G_1 has been reduced by the current G_2. In the last case, we start with G_1 at degree 1 and keep reducing at each step until this set reduces to ϕ and then test G_2 for Gröbner basis.

After all the terms of degree one have been produced

$$ReducedTerms = \{z, y, x\},$$

$$InitialTerms = \phi,$$

$$G_2 = \phi,$$

$$G_1 = \{yz + x, xy + z, x^2 - z^2\} \textit{ reduced to } \{yz + x, xy + z, x^2 - z^2\}.$$

After all the terms of degree two have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2\},$$

$$InitialTerms = \{x\},$$

$$G_2 = \{x + yz\},$$

$$G_1 = \{yz + x, xy + z, x^2 - z^2\} \textit{ reduced to } \{y^2z - z, x^2 - z^2\}.$$

After all the terms of degree three have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2z, y^3\},$$

$$InitialTerms = \{x, y^2z\},$$

$$G_2 = \{x + yz, y^2z - z\},,$$

$$\{y^2z - z, x^2 - z^2\} \textit{ reduced to } \phi.$$

Since, at this point, every polynomial from the original Gröbner basis was reduced to zero, G_2 is tested for a Gröbner basis. The test succeeds. Hence G_2 is the Gröbner basis of the ideal generated by G_1 w.r.t. the pure lexicographic order $<_2$.

Example 2: Consider another Gröbner basis of a one dimensional ideal w.r.t. the total degree order with variables $x > y > z$:

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\}.$$

The goal is to transform G_1 into G_2 where $<_2$ is the pure lexicographic order with $x > y > z$. Notice that in this example, when the polynomials in the original Gröbner basis G_1 reduce to 0, G_2 is still not a Gröbner basis. One more step is needed. Also, after degree 2, some polynomials are obtained, which belong to the ideal, but later other polynomials are produced whose headterms divide the headterms of these polynomials.

After all the terms of degree one have been produced

$$ReducedTerms = \{z, y, x\},$$

$$InitialTerms = \phi,$$

$$G_2 = \phi,$$

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\} \textit{ reduced to } \{xy - x, x^2 + xz, y^2z + x\}.$$

After all the terms of degree two have been produced

$$ReducedTerms = \{z, y, x, z^2, yz, y^2, xz\},$$

$$InitialTerms = \{xy, x^2\},$$

$$G_2 = \{xy - x, x^2 + xz\},$$

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\} \textit{ reduced to } \{y^2z + x\}.$$

After all the terms of degree three have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2z, y^3\},$$

$$InitialTerms = \{x\},$$

$$G_2 = \{x + y^2z, xy - x, x^2 + xz\},$$

$$\{y^2z + x\} \textit{ reduced to } \phi.$$

At this point, since all polynomials in G_1 reduce to 0, G_2 must be tested for a Gröbner basis. This test fails because the S-polynomial of $\{x + y^2z, xy - x\}$ does not reduce to 0. So we enumerate terms of degree 4.

After all the terms of degree four have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2z, y^3, z^4, yz^3, y^2z^2, y^4\},$$

$$InitialTerms = \{x, y^3z\},$$

$$G_2 = \{x + y^2z, y^3z - y^2z, xy - x, x^2 + xz\}.$$

Again the test for a Gröbner basis is performed and this time it succeeds. Hence G_2 is a Gröbner basis for the ideal generated by G_1. The Gröbner basis from G_2 is obtained by deleting polynomials whose head terms can be reduced by the head terms of other polynomials.

8 Empirical results

The general algorithm has been implemented in GEOMETER system [3]. The heuristics outlined in section 6 have been tried. A degree ordering was used for enumeration of terms. Polynomials in the input Gröbner basis were incrementally reduced to 0 to ensure that the output generated the same ideal. At the end, a Gröbner basis test was incrementally run in which S-polynomials are reduced to 0. The performance of the general basis conversion algorithm is compared with the classical Buchberger's Algorithm using the normal selection strategy [4]. Some of the examples tried for comparisons were given below. The polynomials for each set are written in the infix notation and a Gröbner basis was computed w.r.t. the term order $A > B > C > D > E > F$.

1.
```
(+ A B C D E F)
(+ (* A B) (* B C) (* C D) (* D E))
(+ (* A B C) (* B C D) (* C D E))
(+ (* A B C D) (* C B D E))
(+ (* A B C D E) -1)
```

2.
```
(+ A B C D E F)
(+ (* A B) (* B C) (* C D) (* D E))
(+ (* A B C) (* B C D) (* C D E))
(+ (* A B C D) (* B C D E))
(+ (* A B C D E F) -1)
```

3.
```
(+ A B C D E F)
(+ (* A B) (* B C) (* C D) (* D E) (* E F))
(+ (* A B C) (* B C D) (* C D E))
(+ (* A B C D) (* B C D E))
(+ (* A B C D E F) -1)
```

4.
```
(+ A B C D E F -1)
(+ (* A B) (* B C) (* C D) (* D E) -1)
(+ (* A B C) (* B C D) (* C D E))
(+ (* A B C D) (* B C D E))
(+ (* A B C D E F) -1)
```

In Figure 1 above, timings are given for computing a degree basis on GeoMeter (a), converting to a lexicographic basis from the degree basis on GeoMeter (b), total time taken to obtain the lexicographic basis by conversion (c) (= a + b), obtaining the lexicographic basis directly on GeoMeter (d) and finally, for obtaining the lexicographic basis directly on Maple (e). A star (*) entry indicates that either the program ran out of space or it ran for more than 24 hours.

It was found that for most nontrivial examples, *BasisConvert* performed better than the classical algorithm as implemented in GEOMETER and MAPLE. Comparisons with other systems are difficult to make because their implementations and handling of data structures could possibly be totally different.

There is considerable room for improving the performance of the *BasisConvert* algorithm. We earlier mentioned issues related to enumeration orderings as well as tests for termination.

Example	a	b	c	d	e
1	16	221	237	570	2010
2	27	365	382	811	*
3	35	3850	3885	8894	*
4	368	19213	19581	*	*

Figure 1: Comparison of *BasisConvert* with direct computation of Gröbner basis

Further, currently we are using a naive Gaussian elimination procedure in *Solve* using the *kcl* infinite precision rational arithmetic. A better implementation of a Gaussian elimination method is likely to result in much better timings.

References

[1] J.C. Faugere, P. Gianni, D. Lazard and T. Mora Efficient computation of zero-dimensional Gröbner bases by change of ordering. *To appear in: J. Symb. Comp.* 1990.

[2] T. Becker, V. Weispfenning in cooperation with H. Kredel Gröbner Bases - A Computational Approach to Commutative Algebra. *Springer Verlag* 1993.

[3] C.I. Connolly, D. Kapur, J.L. Mundy and R. Weiss GeoMeter: A System for Modeling and Algebraic Manipulation. *Proc. of DARPA Workshop on Image Understanding.* Palo Alto, CA, pp 797-804 1989.

[4] B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory. In: (ed. N.K. Bose) *Multidimensional System Theory,* D. Reidel Publishing Co, 1985, 184-232.

Sparsity Considerations in Dixon Resultants*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222

April 26, 1995

Abstract

New results relating the sparsity of non-homogeneous polynomial systems and computation of their projection operator (i.e., a non-trivial multiple of multivariate resultant) using Dixon's method are developed. It is proved that the Dixon formulation of resultants, despite being classical, implicitly exploits the structure of the Newton polytopes of input polynomials; the complexity of computing Dixon resultant is not determined by the total degree of the polynomial system. Bound on the size of the Dixon matrix of unmixed polynomial systems is derived in terms of their Newton polytopes. This bound is used to prove that for a multi-homogeneous system, the size of its Dixon matrix is of a smaller order than its n-fold mixed volume. Using dense multivariate polynomial interpolation techniques, it is shown that for a fixed number of variables, Dixon matrix of multi-homogeneous polynomial systems can be constructed using $\mathcal{O}(\mathcal{M}^3)$ arithmetic operations, where $\mathcal{M}$ is the n-fold mixed volume of the input system. The Dixon matrix is found to be smaller than the sparse and Macaulay resultant matrices by a factor exponential in the number of variables, and the Dixon formulation yields a faster algorithm to compute the resultant. This work links the classical Dixon formulation (developed in 1908) to the modern line of sparsity analysis based on Newton polytopes.

1 Introduction

Eliminating n variables from $n+1$ multivariate polynomials results in the *resultant* – a quantity which has proved useful in many applications [8]. Most efficient ways to compute the resultant express it as a determinant. However, a pure determinantal expression for the resultant is rare (see [13] for some classes of polynomials which allow a pure determinantal formulation), and in these cases, one can still express some nontrivial multiple of the resultant (called a *projection operator*) as a determinant.

There are three major methods to compute the resultant or a projection operator – the classical formulations by Dixon [4] and Macaulay [12] developed early this century, and the recently developed sparse resultant formulation [11, 6]. All three methods construct matrices whose determinant is either the resultant, or a projection operator.

The Macaulay formulation uses the traditional Bezout bound on the number of solutions of a polynomial system to construct the Macaulay matrix. The size of this matrix depends on the total degree of the input polynomials. However, recently, sharper bounds have been derived on the number

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

of solutions, known as BKK bounds [1], which exploit the structure and sparsity of the polynomials as determined by their Newton polytopes. The sparse resultant formulation is a modification of the Macaulay formulation which exploits the modern BKK bound and Newton polytopes of the input polynomials to construct smaller matrices [7]. The size of the sparse resultant matrix depends on the volume of the polytopes of input polynomials, and is not governed by their total degree. Size of sparse resultant matrix is small because it does not assume any monomials, besides the ones inside the Newton polytope, have non-zero coefficients – unlike the Macaulay matrix, which is much larger because it is constructed under the crude assumption that all monomials of total degree less than or equal to the input polynomial have non-zero coefficients, without regard to the more refined structure of the polynomial in terms of its Newton polytope.

Methods based on the Dixon formulation have been empirically found to be more efficient than the Macaulay and the sparse resultant formulations [9, 10]. However, being a classical approach (developed in 1908), the relationship between the Dixon formulation and the modern BKK bounds, or the structure of the input polytopes, is not well understood. In [11], Sturmfels gave the bracket representation of the Dixon resultant for bi-homogeneous systems; however the Dixon resultant was not analyzed in its full generality. Moreover, it has been unclear whether the Dixon formulation exploits the sparsity of input polynomials, and if so, to what extent (quantitatively). Little has been known about the size of the Dixon matrix and its relationship to the Newton polytopes of the input polynomials.

This paper makes three contributions. The first contribution is to show that the Dixon formulation of multivariate resultants implicitly exploits the structure of Newton polytopes of input polynomials and is, thus, not governed by their total degree (or the classical Bezout bound). Using this relationship, an explicit bound on the size of Dixon matrix is derived in terms of the volume of polytopes of input polynomials, and this is the second and main contribution of the paper. It is shown that the size of the Dixon matrices of unmixed systems is bounded by the number of integral points inside the Minkowski sum of *successive projections* of their Newton polytopes (Theorem 3.4 and Corollary 3.5). And, the third contribution of the paper is an upper bound on the complexity of constructing Dixon matrices. This is especially significant since entries in a Dixon matrix are more complex in contrast to entries in a Macaulay matrix or a sparse resultant matrix, which are simply the coefficients of terms in polynomials. This bound on the construction of Dixon matrix shows that despite more complicated entries, it can be constructed fast. These contributions are further elaborated below.

It is shown in this paper that, much like the sparse resultant formulation, the Dixon formulation assumes that all monomials outside the polytopes have zero coefficients. If the polytopes of two polynomial systems have the same volumes and projections, then no matter how different their total degrees, their Dixon matrices, like their sparse resultant matrices, would be of the same size. The size of the Dixon matrix is much smaller than the size of Macaulay matrix. It is also shown that the size of the Dixon matrix is smaller than the size of sparse resultant matrix, which is bounded by the number of integral points inside the Minkowski sum of the Newton polytopes (not their successive projections) of the input polynomials. This is analogous to the Dixon matrix whose size is bounded by the number of integral points inside the Minkowski sum of the *successive projections* of Newton polytopes of input polynomials. However, since the projections are of successively lower dimension than the polytopes themselves, the Dixon matrix is smaller. Specifically, the size of the Dixon matrix of multi-homogeneous polynomials is proved to be of smaller order than their n-fold mixed volume (Theorem 3.8), whereas the size of the sparse resultant matrix is larger than the n-fold mixed volume by an exponential multiplicative factor (Theorem 3.9). If the n-fold mixed volume of a multi-homogeneous polynomial system is $\mathcal{M}$ and the number of variables being eliminated is n

then for asymptotically large number of partitions of variables (approaching n),

$$\text{Size of its Dixon matrix} = \mathcal{O}\left(\frac{1}{n+1}\mathcal{M}\right), \quad (1)$$

$$\text{Size of its Sparse Resultant matrix} = \mathcal{O}\left(e^{n+1}\mathcal{M}\right). \quad (2)$$

This is actually congruent with the intuition Dixon discussed in his paper, because he mainly developed this formulation as an efficient method for n-degree polynomial systems[1], and for asymptotically large number of partitions, multi-homogeneous systems approach the n-degree case.

Regarding the construction of the Dixon matrix, it is shown that it can be constructed fast using a variant of the dense multivariate polynomial interpolation algorithm, which exploits the bounds on the size of the Dixon matrix. Specifically, for multi-homogeneous systems with a fixed number of variables, the Dixon matrix can be constructed using $O(\mathcal{M}^3)$ arithmetic operations (Theorem 4.1).

Putting these results together, the cost of computing a projection operator using Dixon formulation is the sum of (i) the cost of constructing the Dixon matrix and (ii) the cost of computing its determinant. The number of arithmetic operations required for (ii) is $\mathcal{O}(\mathcal{M}^3)$ (see Equation 1). Consequently, for a fixed number of variables, the computation of a projection operator using the Dixon formulation takes $\mathcal{O}(\mathcal{M}^3)$ arithmetic operations. This is in contrast to the sparse resultant formulation which, if the polynomials are *full*[2], then for a fixed number of variables, takes $\mathcal{O}(\mathcal{M}^{6.5})$ arithmetic operations [7]. If the polynomials are *hollow*[3], the construction time of the sparse resultant reduces considerably and the determinant computation dominates, but, it still takes longer than the Dixon formulation because the matrix size is larger (Equation 2).

To the best of our knowledge, this is the first time such a relationship has been shown between the classical Dixon formulation and the modern line of analysis based on Newton polytopes of polynomials. The analysis based on this relationship reveals that the Dixon formulation does take into account the sparsity of input polynomials, creates smaller resultant matrices, and has lower complexity than the sparse resultant formulation.

2 Dixon Method for Computing Projection Operators

Let $X = \{x_1, \ldots, x_n\}$ and $\bar{X} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ be two sets of n variables each. Given a polynomial $p \in \mathbb{Q}[X]$, $p(\bar{x}_1, \ldots, \bar{x}_k, x_{k+1}, \ldots, x_n)$ stands for uniformly replacing x_j by $\bar{x}_j$ for $1 \leq j \leq k$ in p. For a given polynomial $q \in \mathbb{Q}[X, \bar{X}]$, $V_X(q)$ and $V_{\bar{X}}(q)$, the **monomial vectors** of q wrt. X and $\bar{X}$ respectively, are defined to be the vectors of all monomials appearing in q in the variables X and $\bar{X}$ respectively. The **exponent vector** of the monomial $x_1^{e_1} \cdots x_n^{e_n} \bar{x}_1^{\bar{e}_1} \cdots \bar{x}_n^{\bar{e}_n}$ wrt. the set X of variables is defined to be the n-tuple $(e_1, \ldots, e_n) \in \mathbb{Z}^n$. The **support**, $supp(q) \subset \mathbb{Z}^n$, of a polynomial q is the set of exponent vectors of all the monomials, wrt. X, with non-zero coefficients in q. The determinant of a square matrix A is denoted by $|A|$.

Example: If $X = \{x_1, x_2\}$ and $p = 2x_1^3 x_2 \bar{x}_1^3 + x_2^4 \bar{x}_1 \bar{x}_2^2 + \bar{x}_1^2 + \bar{x}_1 \bar{x}_2 + 1 \in \mathbb{Q}[X, \bar{X}]$, then $V_X(p) = [x_1^3 x_2,\ x_2^4,\ 1]$ and $V_{\bar{X}}(p) = [\bar{x}_1^3,\ \bar{x}_1 \bar{x}_2^2,\ \bar{x}_1^2,\ \bar{x}_1 \bar{x}_2,\ 1]$. $supp(p) = \{(3,1), (0,4), (0,0)\}$. □

[1] $n+1$ nonhomogeneous polynomials $p_1, \ldots, p_{n+1}$ in $x_1, \ldots, x_n$ are called n-degree there exist nonnegative integers $m_1, \ldots, m_n$ such that each
$p_j = \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} a_{j,i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n}$ *for* $1 \leq j \leq n+1$.

[2] In a full polynomial, all terms inside its Newton polytope have non-zero coefficients.

[3] In a hollow polynomial, only the terms on the boundary of its Newton polytope have non-zero coefficients.

Definition 2.1 *Let* $P = \{p_1, \ldots, p_{n+1}\} \subset \mathbb{Q}[X]$ *be a set of* $n+1$ *polynomials in* n *variables* $x_1, \ldots, x_n$. *The* **cancellation matrix** *of* P, *denoted* C_P, *is*

$$C_P = \begin{bmatrix} p_1(x_1, x_2, x_3, \ldots, x_n) & \cdots & p_{n+1}(x_1, x_2, x_3, \ldots, x_n) \\ p_1(\bar{x}_1, x_2, x_3, \ldots, x_n) & \cdots & p_{n+1}(\bar{x}_1, x_2, x_3, \ldots, x_n) \\ p_1(\bar{x}_1, \bar{x}_2, x_3, \ldots, x_n) & \cdots & p_{n+1}(\bar{x}_1, \bar{x}_2, x_3, \ldots, x_n) \\ \vdots & \vdots & \vdots \\ p_1(\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n) & \cdots & p_{n+1}(\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n) \end{bmatrix}.$$

The **Dixon polynomial** *of* P, *denoted* $\delta_P \in \mathbb{Q}[X, \bar{X}]$, *is*

$$\delta_P = \frac{|C_P|}{\prod_{i=1}^{n}(x_i - \bar{x}_i)},$$

and the **Dixon matrix** *of* P, *denoted* D_P, *is the matrix for which* $\delta_P = V_{\bar{X}}(\delta_P)\, D_P\, V_X(\delta_P)$.

If the Dixon matrix is square, its determinant is a projection operator whose vanishing is a necessary condition for the system P to have a common solution in the *algebraic torus* $(\mathbb{C} - \{0\})^n$. However, if the Dixon matrix is rectangular then the determinant (and the projection operator) cannot be directly computed. Even if it is square, it may be singular, resulting in a trivial projection operator, which is undesirable. The following theorem given in [9], provides a method to extract a non-trivial projection operator in such instances.

Theorem 2.2 *If* D_P *has a column which is linearly independent of all other columns, then the determinant of any non-singular rank submatrix of* D_P *is a non-trivial projection operator.*

This method of computing the projection operator is quite efficient in practice as demonstrated in [10]. As can be noticed from the theorem, this method may fail if there is no column in the Dixon matrix which is linearly independent of all other columns. However, such failure seems very rare; in fact, it has never occurred on the numerous problems that we have tried from different application domains. (For systems with two variables, it can be proved that this method always works [4].)

Below we establish a relationship between the Newton polytopes of the input polynomials and the size of the Dixon matrix, and compare it with resultant matrices in other formulations.

3 Newton Polytopes and the Dixon Matrix

Let the convex hull of a set $\mathcal{A} \subset \mathbb{Z}^n$ be denoted by $conv(\mathcal{A}) \subset \mathbb{R}^n$. The **Newton polytope** of a polynomial p is $\mathcal{N}(p) = conv(supp(p)) \subset \mathbb{R}^n$, where as defined above, $supp(p)$ is the set of exponent vectors of all the monomials wrt. X in p. For a polynomial p, $supp(p) \subseteq \mathcal{N}(p) \cap \mathbb{Z}^n$.

Given a set of points $\mathcal{A} \subset \mathbb{R}^n$ and $0 \leq i \leq n$, $\pi_i(\mathcal{A}) \subset \mathbb{R}^n$ stands for the set obtained by replacing the first i coordinates of all points in $\mathcal{A}$ by 0 – a projection of $\mathcal{A}$ to $n-i$ dimensions. We will abuse the terminology and call this projection as the $\mathbf{i^{th}}$ **projection** of $\mathcal{A}$. $\pi_0(\mathcal{A}) = \mathcal{A}$ and $\pi_n(\mathcal{A}) = \{0, \ldots, 0\}$.

The **Minkowski sum** of sets $\mathcal{A}, \mathcal{B} \in \mathbb{R}^n$ is $\mathcal{A} + \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\} \subset \mathbb{R}^n$, or equivalently,

$$\mathcal{A} + \mathcal{B} = \bigcup_{a \in \mathcal{A}} (a + \mathcal{B}). \tag{3}$$

Fact 3.1 *Given a set* $\mathcal{A} \subset \mathbb{R}^n$ *and two polynomials* p *and* q, *if* $\mathcal{N}(p) \subseteq \mathcal{A}$ *and* $\mathcal{N}(q) \subseteq \mathcal{A}$, *then* $\mathcal{N}(p+q) \subseteq \mathcal{A}$.

Fact 3.2 *Given two polynomials, p and q, $\mathcal{N}(pq) = \mathcal{N}(p) + \mathcal{N}(q)$.*

For simplicity, it is assumed from now onwards that the input system is **unmixed**, i.e., the Newton polytopes of all polynomials are equal. For such a set, P, let $\mathcal{N}(P)$ denote the Newton polytope of any polynomial in P. Following lemma relates the Newton polytope of the determinant of cancellation matrix to the polytopes of input polynomials.

Lemma 3.3 *For an unmixed set, $P \subset \mathbb{Q}[X]$, of $n+1$ polynomials, $\mathcal{N}(|C_P|) \subseteq \sum_{i=0}^{n} \pi_i(\mathcal{N}(P))$, where C_P is the cancellation matrix of P.*

Proof: The Newton polytope of all entries in row $i+1$ of the cancellation matrix is $\pi_i(\mathcal{N}(P))$. If $|C_P|$ is expanded as a sum of products, then each summand is a product of exactly $n+1$ polynomials, one from each row. Form Fact 3.2, the polytope of this summand is $\sum_{i=0}^{n} \pi_i(\mathcal{N}(P))$, so the result follows from Fact 3.1. □

We can relate the Dixon polynomial to the input polytopes as follows:

Theorem 3.4 *For an unmixed set, $P \subset \mathbb{Q}[X]$, of $n+1$ polynomials*[4]*,*

$$supp(\delta_P) \subseteq \sum_{i=0}^{n} \pi_i(\mathcal{N}(P)) \cap \mathbb{Z}^n.$$

Proof: Let $q = \prod_{i=1}^{n}(x_i - \bar{x}_i)$. Polynomial q contains the term $\bar{x}_1 \cdots \bar{x}_n$ which is a constant wrt. the set X of variables, so $(0, 0, \ldots, 0) \in \mathcal{N}(q)$. Since $|C_P| = q \times \delta_P$,

$$\begin{aligned}
\sum_{i=0}^{n} \pi_i(\mathcal{N}(P)) &\supseteq \mathcal{N}(|C_P|) = \mathcal{N}(q \times \delta_P) = \mathcal{N}(q) + \mathcal{N}(\delta_P) && \text{(by Lemma 3.3 and Fact 3.2)} \\
&= (\mathcal{N}(q) \cup \{(0, 0, \ldots, 0)\}) + \mathcal{N}(\delta_P) = (\mathcal{N}(q) + \mathcal{N}(\delta_P)) \cup \mathcal{N}(\delta_P) && \text{(by Equation 3)} \\
&\supseteq \mathcal{N}(\delta_P) \supseteq supp(\delta_P).
\end{aligned}$$

□

The number of columns in the Dixon matrix equals the cardinality of support of the Dixon polynomial. The cardinality of an integer point set is asymptotically bounded by the volume of their convex hull [5]. Let $vol(\mathcal{A})$ denote the n-dimensional **volume** of a convex set $\mathcal{A} \subset \mathbb{R}^n$. We get the following bound on the size of Dixon matrix as a function of the Newton polytopes of input polynomials.

Corollary 3.5 *The number of columns in the Dixon matrix of an unmixed set, $P \subset \mathbb{Q}[X]$, of $n+1$ polynomials is*

$$\mathcal{O}\left(vol\left(\sum_{i=0}^{n} \pi_i(\mathcal{N}(P))\right)\right).$$

We believe this is the first time that such a bound on the size of the Dixon matrix has been derived in terms of the input polytopes. We now compare this to similar results on the size of the Macaulay and sparse resultant matrices.

[4] A slightly stronger version can be proved, viz., support of the Dixon polynomial is, in fact, the number of integer lattice points inside the polytope obtained *after perturbing* $\sum_{i=0}^{n} \pi_i(\mathcal{N}(P))$ just a little bit in the negative octant. Note that this number is strictly less than the number determined by Theorem 3.4 because it moves some of the boundary points out of the polytope. But this decrease does not change the analysis and bounds in the rest of the paper, so we adhere to the theorem above.

3.1 Comparison with Macaulay Matrix

If the total degree of each input polynomials is d, then the size of the Macaulay matrix is $\mathcal{O}\left(\frac{(n+1)^n d^n}{n!}\right)$. So the total degree completely determines the size of Macaulay matrix, no matter how many terms are missing from the polynomials. If a polynomial system has the same number of variables, but polynomials of higher total degree than another, the Macaulay matrix of the former is larger, even though their Newton polytopes may have the same shape and volume.

Corollary 3.5 shows that for the Dixon matrix, this is not the case at all. In fact, the size of the Dixon matrix has no dependence on the total degree of the polynomials, only on their Newton polytopes. Sizes of the Dixon matrices of two different polynomial systems are the same as long as their Newton polytopes have the same shape, no matter what the total degrees of the polynomials.

The Dixon formulation *does* consider the sparsity in the input polynomials and implicitly exploits the structure of the Newton polytopes of input polynomials.

3.2 Comparison with Sparse Resultant Matrix

A result analogous to Corollary 3.5 exists for the sparse resultant matrix [7]:

Fact 3.6 *Size of the sparse resultant matrix of an unmixed set, $P \subset \mathbb{Q}[X]$, of $n+1$ polynomials is*

$$\mathcal{O}\left(vol\left(\sum_{i=0}^{n}\mathcal{N}(P)\right)\right).$$

The reader will notice the surprising similarity between Fact 3.6 and Corollary 3.5. While the sparse resultant matrix considers integral points inside the Minkowski sum of the polytopes, the Dixon matrix considers integral points inside the Minkowski sum of *successive projections* of polytopes. Since the projections are of successively lower dimensions than the polytopes themselves, the Dixon matrix is smaller than the sparse resultant matrix.

As an example of how these various polytopes look, see Figure 1. P is the polytope of the unmixed system $p_i = a_i\,x^2y + b_i\,x^3y + c_i\,x^2y^2 + d_i\,x^3y^2 + e_i\,x^3y^3$ for $1 \leq i \leq 3$. The number of integral points in D and S are the sizes of the Dixon and sparse matrices, respectively. The Macaulay matrix is so large that the region corresponding to terms in its columns cannot be shown in this figure – both, the sparse and Dixon polytopes, are subsets of this region. To see just how much the quantitative difference in sizes of the three resultant matrices is, we analyze the case of multi-homogeneous polynomial systems, which forms an important class with a large number of applications such as robotics, vision etc.

3.3 Multi-Homogeneous Systems

Definition 3.7 *A set of polynomials is called* **multi-homogeneous of type** $(\mathbf{l_1}, \ldots, \mathbf{l_r}; \mathbf{d_1}, \ldots, \mathbf{d_r})$ *if it is unmixed and the set of variables can be partitioned into r subsets, $x_1, \ldots, x_r$ such that, for $1 \leq i \leq r$, the number of variables in the set x_i is l_i and the total degree of each polynomial in P in the variable set x_i is d_i.*

In the rest of the paper, a multi-homogeneous system should be understood to mean a multi-homogeneous polynomial set of type $(l_1, \ldots, l_r;\, d_1, \ldots, d_r)$ with $n+1$ polynomials, where $n = \sum_{i=1}^{r} l_i$ is the total number of variables. We also assume that, for $1 \leq i \leq r$, the number of partition variables $l_i = \mathcal{O}(l)$, and the partition degree $d_i = \mathcal{O}(d)$, and consequently, $n = \mathcal{O}(r\,l)$.

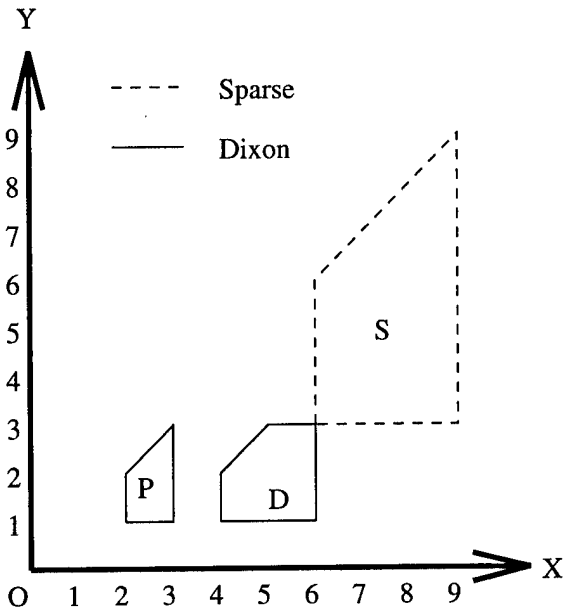


Figure 1: Various polytopes for the system P

The concept of an n-fold mixed volume provides a good quantitative basis to compare different resultant formulations. Given an unmixed set $P \subset \mathbb{Q}[X]$, its **n-fold mixed volume**, denoted $mvol(P)$, is $mvol(P) = n!\ vol(\mathcal{N}(P))$. Following theorem relates the size of the Dixon matrix to the n-fold mixed volume. Its proof is in Appendix A.

Theorem 3.8 *For asymptotically large r, the size of the Dixon matrix of a multi-homogeneous system P is $\mathcal{O}\left(\frac{mvol(P)}{n+1}\right)$.*

Following can be analogously derived for the sparse resultant matrix from Fact 3.6:

Theorem 3.9 *For asymptotically large r, size of the sparse resultant matrix of a multi-homogeneous system P is $\mathcal{O}\left(e^{n+1}\ mvol(P)\right)$.*

For multi-homogeneous systems, for asymptotically large r, the Dixon matrix is smaller than the sparse resultant matrix by a factor exponential in n. It can be shown that the sparse resultant matrix is, in turn, smaller than the Macaulay matrix by a factor exponential in n.

The entries in the Dixon matrix are however complicated in contrast to the entries in the sparse resultant matrix and Macaulay matrix, for which the entries are either 0 or coefficients of the monomials in the polynomials. In the next section, we analyze the complexity of constructing a Dixon matrix.

4 Construction of the Dixon Matrix

Dixon matrix can be constructed by expanding the determinant of the cancellation matrix and dividing it by $\prod_{i=1}^{n}(x_i - \bar{x}_i)$. Since the entries of the cancellation matrix are multivariate polynomials,

direct expansion can be quite expensive, and give rise to large intermediate expressions. This can be avoided by using dense multivariate polynomial interpolation [14].

Interpolation works in stages. In each stage, polynomial is interpolated in one more variable using bounds on the number of terms in the polynomial (when viewed as a polynomial in the variables which have already been lifted) and its degree in the variable being lifted. Since the Dixon polynomial has $2n$ variables, namely X and $\bar{X}$, $2n$ stages are needed. Using analysis in Section 3 (Corollary 3.5), tight bounds on the number of terms in the Dixon polynomial, and its degree in different variables, can be derived and used in each stage of interpolation. The complexity of our variant of interpolation depends on the volume of the polytopes since that is the quantity which determines the number of terms in the Dixon polynomial, and hence, the cost of each stage. The complexity of construction for multi-homogeneous systems is given by the following theorem whose proof is in Appendix B.

Theorem 4.1 *The construction of the Dixon matrix of a multi-homogeneous system, P, requires $\mathcal{O}\left(\frac{n^3\ l!^{3r}}{n!}\ mvol\,(P)^3\right)$ arithmetic operations.*

The use of interpolation to construct Dixon matrices is possible because its entries are coefficients of a polynomial, which can be interpolated from input polynomials. This is not possible for sparse resultant matrix since its entries do not have such a relationship with the input polynomials.

4.1 Total Cost of Computing the Resultant using Dixon Formulation

Computing the resultant using Dixon formulation involves two steps: (i) constructing the Dixon matrix, and (ii) computing the determinant of the Dixon matrix. For a fixed number of variables, the size of the Dixon matrix is proportional to the n-fold mixed volume, thus, computing its determinant takes at most $\mathcal{O}(mvol(P)^3)$ arithmetic operations. From the above theorem, the construction cost, for a fixed number of variables, is $\mathcal{O}(mvol(P)^3)$. Consequently, for a fixed number of variables, the total cost of computing the resultant of a multi-homogeneous system using the Dixon formulation is $\mathcal{O}(mvol(P)^3)$.

4.2 Comparison with Sparse Resultant Formulation

The construction cost of sparse resultant matrix involves two measures of sparsity. First is the mixed volume, which we have used throughout this paper, and the other is the number of terms with non-zero coefficients in the polynomials. The following theorem from [7] gives the cost of constructing the sparse resultant matrix.

Fact 4.2 *If the number of monomials with non-zero coefficients in each polynomial of a multi-homogeneous system P is m, then the complexity of constructing its sparse resultant matrix is $\mathcal{O}\left(e^n\,(m\,n)^{5.5}\ mvol\,(P)\right)$.*

The sparse resultant formulation has an advantage that its construction cost is sensitive to both, the mixed volume, as well as the cardinality of the support of input polynomials (the traditional measure of sparsity). The construction of sparse resultant matrix can, in certain cases, take less time than the construction of the Dixon matrix. However, even in these cases, total cost of computing the resultant using sparse resultant formulation is more than Dixon formulation because the size of sparse resultant matrix is unaffected by the cardinality of support.

We consider two extreme cases: (i) *full* polynomials, in which all terms inside the Newton polytope have non-zero coefficients, and (ii) *hollow* polynomials, in which only the terms on the boundary of the polytopes have non-zero coefficients.

Ex	Application	No. of Polys	Dixon			Macaulay			Sparse		
			Matrix Size	Cnstr. Time	RSC Time	Matrix Size	GCP Time	RSC Time	Matrix Size	Cnstr. Time	RSC Time
1	Geom. Reason.	3	13	0.18	342	55	*	4608	34	0.3	566
2	Formula Deriv.	5	14	0.36	76	330	*	*	86	9.8	4094
3	Formula Deriv.	3	5	0.08	*	15	N/R	*	14	0.2	*
4	Geometry	3	4	0.08	1021	15	*	*	14	0.2	*
5	Random	4	6	0.06	*	84	N/R	*	27	0.7	*
6	Implicitization	3	10	0.1	0.58	45	3741	13.15	16	0.2	0.71
7	Implicitization	3	18	0.45	47	66	N/R	604	55	2	519
8	Implicitization	3	18	9.33	126	153	*	94697	54	1.9	1369
9	Random	4	7	0.21	220	56	*	9972	24	1.2	412
10	Random	5	36	1.1	8130	1365	*	*	151	40	201558
11	Implicitization	3	11	0.06	3.01	36	N/R	32.58	33	0.7	31.04
12	Equilibrium	4	5	0.05	0.24	20	N/R	1.53	20	0.7	2.36
13	Vision	6	6	3.3	0.05	792	*	*	60	248	10.53
14	Comput. Bio.	3	8	0.067	0.27	28	11.18	0.68	16	0.2	0.33

Table 1: Empirical Data

Full Systems: In a full system P, $m = \mathcal{O}(vol(P))$. Using this, the complexity of constructing its sparse resultant matrix is $\mathcal{O}\left(\frac{e^{6.5n}}{n^{5.5(n-1)}}\ (mvol(P))^{6.5}\right)$. For a fixed number of variables, since the cost of computing the determinant of the sparse resultant matrix is $\mathcal{O}(mvol(P))^3$ (from Theorem 3.9), the construction cost dominates the total cost of computing the resultant. Hence, the total cost of computing the resultant of a *full* multi-homogeneous system using sparse resultant formulation is $\mathcal{O}(mvol(P)^{6.5})$. However, the Dixon formulation (Section 4.1) takes $\mathcal{O}(mvol(P)^3)$ arithmetic operations, which is better than sparse.

Hollow Systems: In this case, we can assume that $m = \mathcal{O}(1)$ in Fact 4.2. So the construction cost of the sparse resultant matrix, in this case, is proportional to the n-fold mixed volume. This is better than the construction cost of the Dixon matrix, and if only a resultant matrix is desired, the sparse formulation is preferable. However, while computing the resultant, the determinant computation step dominates. Because of larger size of the sparse resultant matrix, the Dixon formulation again computes the resultant faster.

In both cases, more so in the full case, the Dixon formulation is less expensive than the sparse resultant formulation. To see typical timings in practice, see Table 4.2 (from [10]) in which the timings of Dixon, sparse and Macaulay formulations are given on 14 examples, mostly from robotics, vision, geometry and graphics. The coefficients in these systems are polynomials in parameters and not just integers. For sparse and Dixon formulations, the matrix construction times as well as the determinant computation time (called RSC, which uses interpolation to compute the determinant of the resultant matrices) are separately given. We wish to point out that for implementational simplicity, we used direct matrix expansion of the cancellation matrix for constructing Dixon matrix, and this has worse complexity than the interpolation method discussed in this paper. For Macaulay formulation, we give two timings, one using the Generalized Characteristic Method of Canny [2], and the other using a method similar to one in Theorem 2.2. As can be observed, Dixon matrix is much smaller and the Dixon formulation performs much better than others and was able to solve all 14 problems reasonably well. This work is the result of an effort to seek theoretical justification for these empirical observations.

References

[1] Bernshtein D.N., The Number of Roots of a System of Equations, *Funktsional'nyi Analiz i Ego Prilozheniya,* 9(3):1-4.

[2] Canny J., Generalized Characteristic Polynomials, *Journal of Symbolic Computation*, 9:241-250.

[3] Canny J., Pedersen P., An Algorithm for Newton Resultant, Technical Report, *Cornell University*, CornellCS:TR93-1394, 1993.

[4] Dixon A.L., The eliminant of three quantics in two independent variables. *Proc. London Mathematical Society,* 6, 1908, 468-478.

[5] Ehrart E. Sur un problème de géométrie diophantienne, I. Polyèdres et réseaux., *J. Reine Angew. Math.*, 226:1-29, 1967.

[6] Gelfand I.M., Kapranov M.M., Zelevinsky A.V., *Discriminants, Resultants and Multidimensional Determinants*, Birkhäuser, Boston, 1994.

[7] Emiris I., *Sparse Elimination and Applications in Kinematics*, Doctoral dissertation thesis, Department of Computer Science, U of Calif., Berkeley, 1994.

[8] Hoffman C.M., Algebraic and Numerical Techniques for Offsets and Blends, *Computation of Curves and Surfaces* W. Dahmen et. al. (eds.), Kluwer Acad. Pub., 1990, 499-528.

[9] Kapur D., Saxena T., Yang L., Algebraic and Geometric Reasoning using Dixon Resultants, Proc. *ACM ISSAC 94,* Oxford, England, July 1994.

[10] Kapur D., Saxena T., Comparison of Various Multivariate Resultants, To appear in Proc. *ACM ISSAC 95*, Montreal, Canada, July 1995.

[11] Sturmfels B., Sparse Elimination Theory, *Proc. Computat. Algebraic Geom. and Commut. Algebra*, D. Eisenbud and L. Robbiano, eds., Cortona , Italy, June 1991.

[12] Macaulay F.S., *The Algebraic Theory of Modular Systems*, Cambridge Tracts in Math. and Math. Phys., 19, 1916.

[13] Weyman J. and Zelevinsky A., Determinantal Formulas for Multigraded Resultants, *Journal of Algebraic Geometry*, pp 569-597, 1994.

[14] Zippel R., *Effective Polynomial Computation*, Kluwer Academic Publishers, Boston, 1993.

A Proof of Theorem 3.8

Let P be a multi-homogeneous polynomial set of type $(l_1, \ldots, l_r; d_1, \ldots, d_r)$ with $n+1$ polynomials, where $n = \sum_{i=1}^{r} l_i$ is the total number of variables. Then,

$$\begin{aligned} vol\left(\mathcal{N}(P)\right) &= \prod_{i=1}^{r} \frac{d_i^{l_i}}{l_i!}, \\ mvol\left(P\right) &= n!\ vol\left(\mathcal{N}(P)\right) \\ \text{Size of Dixon matrix} &= \mathcal{O}\left(\frac{\prod_{i=1}^{r}\left(\sum_{j=1}^{i} l_j + 1\right)^{l_i}}{n+1}\, vol\left(\mathcal{N}(P)\right)\right). \end{aligned}$$

Assuming $l_i = \mathcal{O}(l)$, using the Stirling approximation for asymptotically large r, the proof of Theorem 3.8 follows:

$$\begin{aligned} \frac{\text{Size of Dixon matrix}}{mvol\left(P\right)} &= \frac{\prod_{i=1}^{r}(i\,l+1)^{l}}{(rl+1)\,(r\,l)!} = \mathcal{O}\left(\frac{e^{rl}}{rl+1}\left(\prod_{i=1}^{r}\frac{il+1}{rl}\right)^{l}\right) = \mathcal{O}\left(\frac{e^{rl}}{rl+1}\left(\prod_{i=1}^{r}\frac{i}{r}\right)^{l}\right) \\ &= \mathcal{O}\left(\frac{e^{rl}}{rl+1}\left(\frac{r!}{r^r}\right)^{l}\right) = \mathcal{O}\left(\frac{e^{rl}}{rl+1e^{rl}}\right) = \mathcal{O}\left(\frac{1}{n+1}\right). \end{aligned}$$

B Proof of Theorem 4.1

The polytopes of all multi-homogeneous systems with n variables with partition degree d are subsets of the polytope of the system $(\underbrace{1, \ldots, 1}_{n}; \underbrace{d, \ldots, d}_{n})$. Since the size of the Dixon matrix depends only on the input polytopes, the Dixon matrix is the largest for $(1, \ldots, 1; d, \ldots, d)$. Consequently, the cost of construction the Dixon matrix for such a system bounds the cost of constructing the Dixon matrix for all other systems with n variables and partition degree d.

Let the n variables of the system $(1, \ldots, 1; d, \ldots, d)$ be $x_1, \ldots, x_n$ and the corresponding substitution variables be $\bar{x}_1, \cdots, \bar{x}_n$. For $1 \leq i \leq n$, one can easily derive from the analysis in section 3 that for the system $(1, \ldots, 1; d, \ldots, d)$:

1. The degree of δpin x_i is $id - 1$, and hence, degree in $\bar{x}_{n-i+1}$ is also $id - 1$.
2. The number of terms in the variables $x_1, \ldots, x_i$ (regarding all other variables as constants) are bounded by $i!\, d^i$.

We use dense multivariate interpolation to compute the Dixon polynomial (which will directly give us the Dixon matrix). Given $D+1$ values of a univariate polynomial, interpolating them to produce the polynomial takes $I(D) = \mathcal{O}(D^2)$ time [14]. Let the cost of computing a single value of the univariate polynomial from the cancellation matrix be B. Computing a value requires (i) substituting values for each variable in each entry of the cancellation matrix which takes $\mathcal{O}(n^3 d^n)$ arithmetic operations and (ii) computing its determinant, which takes $\mathcal{O}(n^3)$ operations. Hence, $B = \mathcal{O}(n^3 d^n)$.

Multivariate interpolation works in stages; in each stage one variable is lifted. Since the Dixon polynomial has $2n$ variables, namely X and $\bar{X}$, we need $2n$ stages to interpolate it. We lift the variables in the order $x_1, \bar{x}_n, x_2, \bar{x}_{n-1}, \ldots, x_n, \bar{x}_1$. We call the k^{th} variable in this sequence, y_k, eg.

$y_3 = x_2$ and $y_4 = \bar{x}_{n-1}$. Let N_k be the complexity of the first k stages. Now, the k^{th} stage involves (i) interpolating the polynomial in $k-1$ variables as many times as the $deg(\delta_P, y_k) + 1$ and (ii) interpolating as many univariate polynomials (in y_k) as the number of terms in δ_P in variables $y_1, \ldots, y_{k-1}$. Since, in the first stage, one needs to compute the values of the Dixon polynomial directly by computing the determinant of the Dixon matrix, we get the following recurrence relations for our $2n$ stages:

$$\begin{aligned} N_1 &= d\,B + I(d), \\ N_k &= k_c\,d\,N_{k-1} + \left(k_f!\,d^{k_f}\right)\left((k_c-1)!\,d^{k_c-1}\right)\,I\left(k_c d\right), \end{aligned}$$

where, $k_c = \lceil k/2 \rceil$ and $k_f = \lfloor k/2 \rfloor$. The solution to this recurrence is

$$N_{2n} = n!^2 d^{2n}\left(B + \frac{2}{d}\sum_{i=1}^{n}\frac{I(id)}{i}\right) = \mathcal{O}\left(n!^2 d^{2n}\left(n^3 d^n + \frac{2}{d}\sum_{i=1}^{n}\frac{i^2 d^2}{i}\right)\right) = \mathcal{O}\left(n^3 n!^2 d^{3n}\right),$$

therefore,

$$\frac{N_{2n}}{mvol(P)^3} = n^3 n!^2 d^{3n}\frac{l!^{3r}}{n!^3 d^{3n}} = \frac{n^3 l!^{3r}}{n!},$$

and the theorem is proved.

Algebraic and Geometric Reasoning using Dixon Resultants*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222
{kapur, saxena}@cs.albany.edu

Lu Yang
Centre for Mathematical Sciences
Chengdu Institute of Computer Applications
Academia Sinica
610041 Chengdu, China

Abstract

Dixon's method for computing multivariate resultants by simultaneously eliminating many variables is reviewed. The method is found to be quite restrictive because often the Dixon matrix is singular, and the Dixon resultant vanishes identically yielding no information about solutions for many algebraic and geometry problems. We extend Dixon's method for the case when the Dixon matrix is singular, but satisfies a condition. An efficient algorithm is developed based on the proposed extension for extracting conditions for the existence of affine solutions of a finite set of polynomials. Using this algorithm, numerous geometric and algebraic identities are derived for examples which appear intractable with other techniques of triangulation such as the successive resultant method, the Gröbner basis method, Macaulay resultants and Characteristic set method. Experimental results suggest that the resultant of a set of polynomials which are symmetric in the variables is relatively easier to compute using the extended Dixon's method.

1 Introduction

There exist many different techniques for solving a system of algebraic polynomial equations. Resultant computations are still perhaps the most popular way to get information about solutions of polynomial equations. Sylvester resultant method is the most studied and used technique for determining a common solution of two polynomial equations in one variable. Implementations of Sylvester resultants are supported in all computer algebra systems including Mathematica, Maple, Reduce and Macsyma.

Successive Sylvester resultant computations can be used to solve a system of polynomial equations in many variables by eliminating variables one at a time. However, the performance of successive resultant techniques is very sensitive to the ordering of variables. Human intervention is required to determine the most efficient ordering and so they are not automatic methods. Successive elimination methods are also very inefficient, in fact, it is more efficient to directly compute a condition for the existence of common zeros without eliminating variables one at a time. Macaulay's multivariate resultant method is one such method, and it has recently gotten popularized [3]. Unfortunately, efficient implementations of Macaulay's method are not available in any computer algebra method. Other alternatives such as Grobner basis [2] and Characteristic set methods [14][6] don't seem to work well either.

Dixon's method is an efficient method to simultaneously eliminate variables from a system of nonhomogeneous polynomial equations, but unfortunately, it does not work for most algebraic and geometric problems. The main result reported in Dixon's original paper [7] was a method to obtain the resultant of a set of three *generic bidegree* polynomials. This is a generalization of Cayley's formulation [4] of Bezout's efficient method for computing the resultant of two univariate polynomials. Dixon wrote that his method generalized to any $n+1$ *generic ndegree*[1] polynomials in n variables. That is, his method gives the resultant of $n+1$ generic ndegree polynomials. For arbitrary set of $n+1$ nonhomogeneous polynomials in n variables (i.e. not necessarily generic and ndegree), his method gives a necessary condition (henceforth called the *Dixon resultant*) for the existence of a common affine zero.

For most problems which arise in geometry, the matrix set up in Dixon's method, the *Dixon matrix*, becomes singular. As a consequence, the Dixon resultant vanishes identically, without providing any information about the common solutions of equations. This is perhaps the reason that Dixon's method has not been widely used, even though it is quite efficient. Chionh in [5] suggested using perturbation of certain coefficients to obtain nonzero conditions, but this is a nonautomatic method and requires human expertise. Canny in [3] defined the Generalized Characteristic Polynomial for Macaulay resultants which is a systematic way of perturbing a polynomial system so that one gets nonzero conditions for the existence of affine solutions. This can also be achieved for Dixon resultants by mechanically perturbing the polynomials to be ndegree, but this leads to a larger size Dixon matrix with larger entries too. Computing the determinant of such a matrix is cumbersome and leads to

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

[1] $n+1$ nonhomogeneous polynomials $p_1, \cdots, p_{n+1}$ in $x_1, \cdots, x_n$ are said to be *generic ndegree* if there exist nonnegative integers $m_1, \cdots, m_n$ such that each

$$p_j = \Sigma_{i_1=1}^{m_1} \cdots \Sigma_{i_n=1}^{m_n} a_{j,i_1,\cdots,i_n} x_1^{i_1} \cdots x_n^{i_n} \text{ for all } 1 \leq j \leq n+1.$$

inefficiency.

In this paper, we overcome these restrictions by extracting nonzero conditions directly from the Dixon matrix of the system of polynomials. The proposed method does not involve any perturbation, instead we identify and prove a condition on singular Dixon matrices under which, *nonzero* necessary conditions for the existence of a common solution for a system of equations can be extracted. Based on this result, an algorithm for computing the nonzero necessary conditions for the existence of affine common zeros of a system of polynomials directly from its Dixon matrix is developed. It is found that most of the problems for which Dixon's original method was inapplicable, can be solved using our extension of it. Moreover, unlike perturbation techniques [3], the proposed method does not introduce any new variable and terms into the system of polynomials. Because of this, computation is not made any more difficult. Finally, unlike successive elimination techniques and perturbation techniques of [5], our method is fully automatic and does not require any human intervention.

We successfully proved many nontrivial algebraic and geometric identities and theorems within a few minutes (the maximum time for any example was less than 9 minutes) using a naive implementation of our algorithm in MAPLE. We mention five such examples in this paper. Two of these five examples seem intractable by implementations of most other techniques of elimination. The other three examples also take too long to compute using other techniques. Implementations of both, Sylvester's resultant and Bezout's resultant in MAPLE were tried to perform successive resultant computations, but they ran up to a day before running out of memory on all examples. We also tried to compute the lexicographic Gröbner basis on MACAULAY (Bayer and Stillman [1]) as well as MAPLE, but they too ran out of memory after running upto a day. Under block orderings, MACAULAY could not compute the resultant of two examples. For the other three that it could solve, it took substantially longer time than our algorithm. Macaulay resultants (GCP if the Macaulay matrix is singular [3]) were also implemented and tried on MAPLE, but they failed for three examples and took longer time on the other two. Finally, perturbation techniques similar to the GCP of Canny were tried to obtain nonsingular Dixon matrices, but perturbation of polynomials resulted in a blowup of the order of the Dixon matrix due to additional terms and consequently, the determinant computation could not be successfully performed on MAPLE. These preliminary findings, though far from complete, are very encouraging and suggest that further studies are needed to examine methods based on Dixon resultants.

In the next section, we review Cayley's formulation of Bezout's method for two univariate polynomials, and Dixon's method for any $n+1$, generic ndegree polynomials in n variables. Section 2.3 outlines the shortcomings of Dixon's method and the reason for its unapplicability to most geometric problems. In section 3, an algorithm is developed which overcomes this restriction and extracts a nonzero necessary condition for the existence of affine zeros from the Dixon matrix, *even* when it is singular. In section 4, two detailed examples are presented to illustrate the application of our method. In section 5, the advantages of this method over other techniques are discussed, and finally, in section 6, some empirical results are reported.

2 Review of Dixon Resultants

In the next subsection, we describe Cayley's method to determine the resultant for two univariate polynomials. In the subsection following that, we describe an extension by Dixon which gives a general method to determine resultant of $n+1$ *generic ndegree* polynomials in n variables. A detailed exposition of Dixon's method can be found in [12].

2.1 Cayley's Method for Two Polynomials in One Variable

Even though the analysis in this section was developed by Cayley in [4], we will mostly use Dixon's name while developing the notation so that a uniform notation can be carried to the generalization that is presented in the next subsection.

Consider a set $\mathcal{F}$ of two univariate polynomials $\{p_1(x_1), p_2(x_1)\}$. Let $d_{max} = max(degree(p_1, x_1), degree(p_2, x_1))$, where by $degree(p_1, x_1)$, we mean the maximum degree of x_1 in p_1. Consider the polynomial

$$\Delta(x_1, \alpha_1) = \begin{vmatrix} p_1(x_1) & p_2(x_1) \\ p_1(\alpha_1) & p_2(\alpha_1) \end{vmatrix},$$

where α_1 is a new variable and $p_1(\alpha_1)$ stands for uniformly replacing x_1 by α_1 in p_1. Making $x_1 = \alpha_1$ would make $\Delta = 0$ which means that $x_1 - \alpha_1$ divides Δ. Let,

$$\delta(x_1, \alpha_1) = \frac{\Delta(x_1, \alpha_1)}{(x_1 - \alpha_1)} = \frac{p_1(x_1)p_2(\alpha_1) - p_2(x_1)p_1(\alpha_1)}{(x_1 - \alpha_1)}$$

We call δ the **Dixon Polynomial**. The polynomial δ is a $d_{max} - 1$ degree polynomial in α_1 and is symmetric in x_1 and α_1. Every common zero of $p_1(x_1)$ and $p_2(x_1)$ is a zero of $\delta(x_1, \alpha_1)$ no matter what value α_1 has; thus at a common zero of p_1 and p_2, the coefficient of every power product of α_1 in $\delta(x_1, \alpha_1)$ must be 0. This gives a set (say $\mathcal{E}'$) of d_{max} equations corresponding to the coefficients of all the power products of α_1 (viz. α_1^i for each $0 \leq i \leq d_{max} - 1$), each of which is a $d_{max} - 1$ degree polynomial in x_1. So, if D is the $d_{max} \times d_{max}$ coefficient matrix of $\mathcal{E}'$, then

$$\mathcal{E}' \equiv D \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^{d_{max}-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If each power product of x_1 (including $x_1^0 = 1$) is viewed as a new variable, v_i, then we get a set $\mathcal{E}$ of d_{max} homogeneous linear equations in d_{max} variables:

$$\mathcal{E} \equiv D \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{d_{max}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If a common affine zero exists for $\mathcal{F}$ (say $x_1 = c_1$), then this is a solution for $\mathcal{E}'$ also. This results in a nontrivial solution for $\mathcal{E}$ viz., $v_1 = 1, v_2 = c_1, v_3 = c_1^2, \cdots, v_{d_{max}} = c_1^{d_{max}-1}$. Hence if $\mathcal{F}$ has a common affine zero, then $\mathcal{E}$ has a nontrivial solution, implying that the determinant of D is zero. This gives a necessary condition on the coefficients of p_1 and p_2 for them to have a common zero.

It was proved by Cayley in [4] that vanishing of the determinant of D, the Dixon resultant of $\mathcal{F}$, is a necessary and sufficient condition for $\mathcal{F}$ to have a nontrivial common projective zero. In fact, Bezout matrix coincides with the Dixon matrix for the univariate case.

2.2 Dixon's Generalization to Two and More Variables

Dixon explicitly generalized Cayley's method presented in the previous subsection to the two variable case in [7], but it can be easily generalized to any number of variables (and Dixon alluded to this), so we present this generalization under Dixon's name too.

Let $\mathcal{F} = \{p_1(x_1, \cdots, x_n), \cdots, p_{n+1}(x_1, \cdots, x_n)\}$ be the set of $n+1$ *generic ndegree* polynomials in n variables. Let,

$$d_{max_i} = max(degree(p_1, x_i), \cdots, degree(p_{n+1}, x_i)),$$

for all $1 \leq i \leq n$.

We form an $(n+1) \times (n+1)$ determinant similar to the determinant in the last section. Let this determinant be defined as follows:

$$\Delta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_n) = \begin{vmatrix} p_1(x_1, x_2, \cdots, x_n) & \cdots & p_{n+1}(x_1, x_2, \cdots, x_n) \\ p_1(\alpha_1, x_2, \cdots, x_n) & \cdots & p_{n+1}(\alpha_1, x_2, \cdots, x_n) \\ p_1(\alpha_1, \alpha_2, \cdots, x_n) & \cdots & p_{n+1}(\alpha_1, \alpha_2, \cdots, x_n) \\ \cdots & \cdots & \cdots \\ p_1(\alpha_1, \alpha_2, \cdots, \alpha_n) & \cdots & p_{n+1}(\alpha_1, \alpha_2, \cdots, \alpha_n) \end{vmatrix},$$

where $\alpha_1, \cdots, \alpha_n$ are new variables and $p_i(\alpha_1, \cdots, \alpha_k, x_{k+1}, \cdots, x_n)$ stands for uniformly replacing x_j by α_j for $1 \leq j \leq k$ in p_i.

Each of $x_i = \alpha_i$, for all $1 \leq i \leq n$, is a zero of Δ, so they can be removed by dividing Δ by $\Pi_{i=1}^{n}(x_i - \alpha_i)$. Let

$$\delta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_2) = \frac{\Delta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_n)}{(x_1 - \alpha_1) \cdots (x_n - \alpha_n)}.$$

The polynomial δ, which is the **Dixon polynomial**, is of degree $((n+1-i) \times d_{max_i}) - 1$ in α_i and $(i \times d_{max_i}) - 1$ in x_i for all $1 \leq i \leq n$.

Any common zero of $\mathcal{F}$ (say $x_1 = c_1, \cdots, x_n = c_n$) makes the Dixon polynomial vanish, no matter what the values of $\alpha_1, \cdots, \alpha_n$, hence all the coefficients of the various power products of $\alpha_1, \cdots, \alpha_n$ in the Dixon polynomial vanish. Let $\mathcal{E}'$ be the set of all the polynomials in $x_1, \cdots, x_n$ which are coefficients of the power products of $\alpha_1, \cdots, \alpha_n$ in δ. This set then has exactly

$$\Pi_{i=1}^{n}((n+1-i) \times d_{max_i}) = n! \times \Pi_{i=1}^{n} d_{max_i} = s$$

equations (one for each power product of $\alpha_1, \cdots, \alpha_n$), each of which is of degree $(i \times d_{max_i}) - 1$ in x_i. Also, there are

$$\Pi_{i=1}^{n} i \times d_{max_i} = n! \times \Pi_{i=1}^{n} d_{max_i} = s$$

power products in $x_1, \cdots, x_n$ in the equations of $\mathcal{E}'$. Let D be the $s \times s$ coefficient matrix of $\mathcal{E}'$. Then

$$\mathcal{E}' \equiv D \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ \vdots \\ x_n^2 \\ \vdots \\ \Pi_{i=1}^{n} x_i^{i \times d_{max_i} - 1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If each power product of $x_1, \cdots, x_n$ (including $x_1^0 \cdots x_i^0 = 1$) is viewed as a new variable, v_i, then we get a set $\mathcal{E}$ of s homogeneous linear equations in s variables:

$$\mathcal{E} \equiv D \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If a common affine zero exists for $\mathcal{F}$ (say $x_1 = c_1, \cdots, x_n = c_n$), then this is a solution for $\mathcal{E}'$ also. This results in a nontrivial solution for $\mathcal{E}$ viz., $v_1 = 1, v_2 = c_1, v_3 = c_2, \cdots, v_s = c_1^{d_{max_1}-1} c_2^{2d_{max_2}-1} \cdots c_n^{nd_{max_n}-1}$. Hence, as before, if $\mathcal{F}$ has a common affine zero, then $\mathcal{E}$ has a nontrivial solution, implying that the determinant of D is zero. This gives a necessary condition on the coefficients of $p_1, \cdots, p_{n+1}$ for them to have a common zero. As before, D the **Dixon matrix** and its determinant the **Dixon resultant.**

It was proved by Dixon in [7] that for $n+1$ *generic ndegree* polynomials in n variables, vanishing of the Dixon resultant is a necessary and sufficient condition for them to have a nontrivial projective zero, or a necessary condition for the existence of an affine zero. Moreover, $det(D)$ is not identically zero. Recall that *generic ndegree* means that there exists a set of n integers, $k_1, \cdots, k_n$ such that all polynomials, $p_1, \cdots, p_{n+1}$, are of the kind:

$$p_j = \Sigma_{i_1=1}^{k_1} \cdots \Sigma_{i_n=1}^{k_n} a_{j,i_1,\cdots,i_n} x_1^{i_1} \cdots x_n^{i_n}, \ 1 \leq j \leq n+1.$$

where each a is a distinct indeterminate. So for such a set of polynomials, the Dixon resultant is a polynomial in all of a's that is not identically zero but vanishes for those particular values of a's (from the algebraic closure of the field of rational numbers, say) for which $\mathcal{F}$ has a common affine zero. Next we discuss the case when the polynomials are not generic and ndegree.

2.3 Case of Arbitrary Polynomials

Consider the following three polynomials in two variables.

$$\begin{aligned} p_1 &= x^2 + axy - y + b \\ p_2 &= xy + (a+b)y - c \\ p_3 &= bx + y + 2ac \end{aligned}$$

Here a, b and c are parameters. This set of polynomials is not generic (because for example, the coefficient of x^2 in p_1,

1, is not an independent parameter, and the coefficient of y in p_2 is not independent since it depends on the coefficients of p_1). This set is also not ndegree (because for example, even though the maximum degree of x and y is two and one respectively, there is no term x^2y in any of the polynomials. So what must one do in this case? The analysis in the previous section was developed for generic ndegree polynomials, so it would not work directly for this set of polynomials. It is possible to write down the generic polynomials of ndegree (2,1) in x and y:

$$\begin{aligned} p_1' &= a_1x^2y + a_2x^2 + a_3xy + a_4x + a_5y + a_6 \\ p_2' &= a_7x^2y + a_8x^2 + a_9xy + a_{10}x + a_{11}y + a_{12} \\ p_3' &= a_{13}x^2y + a_{14}x^2 + a_{15}xy + a_{16}x + a_{17}y + a_{18} \end{aligned}$$

Now construct its 4×4 Dixon matrix D. Compute its determinant $det(D)$ and substitute the values $a_1 = 0, a_2 = 1, a_3 = a, a_4 = 0, a_5 = -1, \cdots, a_{18} = 2ac$ into $det(D)$. This would give a polynomial in a, b and c (call it p_{dix}). The polynomial p_{dix} vanishes at all those values of a, b and c at which p_1, p_2 and p_3 has an affine zero. This same approach could be followed for any arbitrary set of polynomials $\mathcal{F}$ as follows:

1. Construct generic ndegree polynomials $\mathcal{F}'$ which when specialized in a certain way (say a mapping ψ from the parameters of $\mathcal{F}'$ to the parameters of $\mathcal{F}$), becomes $\mathcal{F}$.
2. Find the Dixon resultant of $\mathcal{F}'$.
3. Specialize the parameters in this Dixon resultant polynomial using ψ to get the polynomial p_{dix}.

When this approach is followed, often p_{dix} becomes identically zero. This is something which was avoided while working with generic ndegree polynomials. More importantly, this approach also results in inefficiency because $\mathcal{F}'$ has larger polynomials (substantially in some cases) than $\mathcal{F}$.

We have adopted a different approach in this paper. Remove the restriction of generic ndegree from the construction of the previous section. So now, we are given a set of any arbitrary polynomials. Construct Δ as in the last subsection. Divide its determinant by $(x_1 - \alpha_1) \cdots (x_n - \alpha_n)$ to get the Dixon polynomial δ. This polynomial now may have a degree less than or equal to $((n+1-i) \times d_{max_i}) - 1$ in each α_i and less than or equal to $(i \times d_{max_i}) - 1$ in each x_i for all $1 \leq i \leq n$. Construct the set of linear equations $\mathcal{E}$ from the Dixon polynomial as before, except that now $\mathcal{E}$ may have less than or equal to s equations in less than or equal to s variables. We continue to call the coefficient matrix of $\mathcal{E}$ to be the **Dixon matrix** D *which may be an* $s_1 \times s_2$ *matrix where* $s_1 \leq s$ *and* $s_2 \leq s$.

Notice though that the determinant of the Dixon matrix may not give us a necessary condition for the existence of affine zeros of $\mathcal{F}$ because $\mathcal{E}$ may not have a nontrivial solution even when $\mathcal{F}$ has a common affine zero. This is because the Dixon matrix may not contain the column corresponding to the monomial $x_1^0 \cdots x_n^0$, and hence if $\mathcal{F}$ has only $x_1 = 0, \cdots, x_n = 0$ as the common solution, then $\mathcal{E}$ only has a trivial solution. Even if the Dixon matrix *does* contain the column corresponding to $x_1^0 \cdots x_n^0$, it could be singular. Worst of all, the Dixon matrix could be a rectangular matrix (because s_1 and s_2 may not be same), in which case, there does not even exist the possibility of computing the determinant. We deal with all of these possibilities together in the next section.

3 Dealing with Singular Dixon Matrices

First let us formalize the notion for this new framework of arbitrary polynomials. Let there be m parameters $a_1, \cdots, a_m$ and $\mathcal{P} = \mathcal{Q}[a_1, \cdots, a_m]$, the ring of all polynomials in them ($\mathcal{Q}$ is the field of rational numbers). The problem being solved is, given a set $\mathcal{F}$ of $n+1$ polynomials from $\mathcal{P}[x_1, \cdots, x_n]$, give a polynomial from $\mathcal{P}$ that is not identically zero, but which must vanish for all those particular values of the parameters $a_1, \cdots, a_m$ from $\bar{\mathcal{Q}}^m$ ($\bar{\mathcal{Q}}$ is the algebraic closure of $\mathcal{Q}$) for which $\mathcal{F}$ has a common affine zero in $\bar{\mathcal{Q}}^n$. We will call any such polynomial a **Projection operator,** ϱ.

Consider a slightly different problem from the one posed in the previous paragraph as follows: A set of constraints $\mathcal{C}$ on the variables $x_1, \cdots, x_n$ of the form $x_{i_1} \neq 0 \wedge \cdots \wedge x_{i_k} \neq 0$ are also given, and we are looking for a polynomial in $\mathcal{P}$ that is not identically zero, but vanishes on all those particular values of parameters from $\bar{\mathcal{Q}}^m$ for which $\mathcal{F}$ has a common affine zero in $\bar{\mathcal{Q}}^n$ *which also satisfies the set of constraints* $\mathcal{C}$. We will call this the **Projection operator modulo** $\mathcal{C}$, $\varrho_{\mathcal{C}}$.

If one is looking for a necessary condition, ϱ for the existence of affine solutions, then $\mathcal{C}$ can be ϕ, and $\varrho = \varrho_\phi$. But the problem of computing $\varrho_{\mathcal{C}}$ in itself is important because in areas such as geometric theorem proving, one occasionally encounters constraints under which solutions are sought. So, we will now give a method which, given the set of $n+1$ polynomials $\mathcal{F} \subset \mathcal{P}[x_1, \cdots, x_n]$ and the set of constraints $\mathcal{C}$ on the variables, gives the required necessary condition on the parameters which is not identically zero *provided* the Dixon matrix satisfies a certain condition.

Recall that the Dixon matrix is of dimension $s_1 \times s_2$ where $s_1 \leq n! \Pi_{i=1}^n d_{max_i}$ rows (corresponding to the coefficients of all power products of αs in the Dixon polynomial) and $s_2 \leq n! \Pi_{i=1}^n d_{max_i}$ columns (corresponding to all power products of x_is in the Dixon polynomial). Let the columns of the Dixon matrix be denoted by m_i (m_1 being the first column of the Dixon matrix and so on). For any column indexed by m_i, let $monom(m_i)$ denote the monomial (or power product in $x_1, \cdots, x_n$) corresponding to that column. Also, given a set of constraints $\mathcal{C}$, let $nvcol(\mathcal{C})$ denote the set of all columns m_i such that $\mathcal{C} \Rightarrow monom(m_i) \neq 0$.

First let us establish a lemma about polynomials in which all coefficients are from the algebraic closure of the field of rational numbers $\bar{\mathcal{Q}}$. Given a set of $n+1$ polynomials $\mathcal{G} \subset \bar{\mathcal{Q}}[x_1, \cdots, x_n]$, let N be the $s_1 \times s_2$ Dixon matrix of $\mathcal{G}$. Also let,

$$\mathcal{N}_1 = \{X | X \textit{ is an } s_1 \times (s_2 - 1) \textit{ submatrix of } N \textit{ obtained by deleting a column which belongs to } nvcol(\mathcal{C}) \textit{ from } N\}$$

Notice that if $\mathcal{C}$ is $x_1 \neq 0 \wedge \cdots \wedge x_n \neq 0$ then $\mathcal{N}_1$ contains all the $s_1 \times (s_2 - 1)$ submatrices of N. Now we prove a lemma relating the rank of the Dixon matrix N with the ranks of all the matrices which are elements of $\mathcal{N}_1$ as follows.

Lemma 3.1 *If* $\mathcal{G}$ *has a common affine zero which satisfies* $\mathcal{C}$, *then*

$$\forall X \epsilon \mathcal{N}_1, rank(X) = rank(N)$$

Proof : Let $x_1 = c_1, \cdots, x_n = c_n$ be the common affine zero of $\mathcal{F}$ which satisfies $\mathcal{C}$. Since each of the s_2 variables in the set of s_1 homogeneous linear equations, $\mathcal{E}$, stands for a monomial in $x_1, \cdots, x_n$, this solution for $\mathcal{F}$ generates a solution for $\mathcal{E}$, say $v_1 = C_1, v_2 = C_2, \cdots, v_{s_2} = C_{s_2}$. So,

if $m_1, \cdots, m_{s_2}$ are the $s_1 \times 1$ column vectors of the Dixon matrix N then,

$$C_1 m_1 + C_2 m_2 + \cdots + C_i m_i + \cdots + C_{s_2} m_{s_2} = 0.$$

Let X be any element of $\mathcal{N}_1$. Then X was obtained from N by deleting some column of N (say m_i) which belonged to $nvcol(\mathcal{C})$. But this means that $monom(m_i)$ cannot vanish on any solution which satisfies $\mathcal{C}$, or in other words, $C_i \neq 0$. So dividing the above equation by C_i and rearranging terms on both sides, we get the equation:

$$m_i = \frac{-C_1}{C_i} m_1 + \frac{-C_2}{C_i} m_2 + \cdots + \frac{-C_{i-1}}{C_i} m_{i-1} + \frac{-C_{i+1}}{C_i} m_{i+1} + \cdots + \frac{-C_{s_2}}{C_i} m_{s_2}$$

In other words, the column vector m_i is a linear combination of the column vectors $m_1, \cdots, m_{i-1}, m_{i+1}, \cdots, m_{s_2}$. Hence, the dimension of the column vector space spanned by $m_1, \cdots, m_{i-1}, m_{i+1}, \cdots, m_{s_2}$ is the same as the dimension of the column vector space spanned by $m_1, \cdots, m_{s_2}$. As a consequence, the rank of the matrix formed by the columns $m_1, \cdots, m_{i-1}, m_{i+1}, \cdots, m_{s_2}$ (viz., X) must be the same as that of the matrix formed by all the column vectors (viz., N), i.e., $rank(X) = rank(N)$. □

Now let's get back to the case when the polynomials have coefficients from $\mathcal{P}$. Let $\mathcal{F} \subset \mathcal{P}[x_1, \cdots, x_n]$ be a set of $n+1$ polynomials and D the Dixon matrix of $\mathcal{F}$ (with entries now from $\mathcal{P}$). Let $r = rank(D)$. Also, let $\mathcal{D}_1$ be the set of all $s_1 \times (s_2 - 1)$ matrices obtained from D by deleting a column which is an element of $nvcol(\mathcal{C})$ (just as $\mathcal{N}_1$ was obtained from N previously).

Let $\phi : \{a_1, \cdots, a_m\} \mapsto \bar{\mathcal{Q}}$ be a mapping which gives values to the parameters from the algebraic closure of $\mathcal{Q}$. By abuse of notation $\phi(\mathcal{F})$,$\phi(D)$, and $\phi(\mathcal{D}_1)$ are the results of substituting those values for the parameters in $\mathcal{F}$, D and $\mathcal{D}_1$ respectively and removing all zero rows and columns from D and elements of $\mathcal{D}_1$. Finally, let

$$\mathcal{R} = \{Y | Y \text{ is an } r \times r \text{ nonsingular submatrix of } D\}$$

Note that $\mathcal{R}$ is never an empty set (because $rank(D) = r$) and moreover, for all $Y \epsilon \mathcal{R}$, $det(Y) \not\equiv 0$ (because elements of $\mathcal{R}$ are nonsingular). Now we have the following theorem.

Theorem 3.2 *If $\exists X \epsilon \mathcal{D}_1$ s.t. $rank(X) < rank(D)$ then for all $Y \epsilon \mathcal{R}$, $\phi(det(Y))$ vanishes if $\phi(\mathcal{F})$ has a common affine zero which satisfies $\mathcal{C}$.*

Proof: Let $\mathcal{G} = \phi(\mathcal{F})$, then it is easy to see that $N = \phi(D)$ is the Dixon matrix of $\mathcal{G}$ and $\mathcal{N}_1 = \phi(\mathcal{D}_1) - \{N\}$ is the set of all submatrices of N obtained by deleting a column which is an element of $nvcol(\mathcal{C})$ from N. So by the previous lemma, if $\mathcal{G} = \phi(\mathcal{F})$ has a common affine zero which satisfies $\mathcal{C}$ then

$$\forall X \epsilon \mathcal{D}_1, rank(\phi(D)) = rank(\phi(X)). \quad (1)$$

Also, the rank of any matrix with entries from $\mathcal{P}$ cannot increase when values from $\bar{\mathcal{Q}}$ are substituted for the parameters, so

$$\forall X \epsilon \mathcal{D}_1, rank(\phi(X)) \leq rank(X). \quad (2)$$

If there is an $X \epsilon \mathcal{D}_1$ which satisfies the premise of the lemma then

$$rank(X) < rank(D) = r, \quad (3)$$

and from equations (1), (2) and (3) we get,

$$rank(\phi(D)) = rank(\phi(X)) \leq rank(X) < rank(D) = r$$

$$i.e., \; rank(\phi(D)) < r.$$

This means that once the values of parameters are substituted in D, its rank becomes less than r i.e., all the $r \times r$ submatrices of D (which are exactly the elements of $\mathcal{R}$) become singular, which means that their determinants vanish at these values of the parameters. □

Theorem 3.2 suggests the following algorithm to obtain $\varrho_{\mathcal{C}}$. Check if $\exists X \epsilon \mathcal{D}_1$ *s.t.* $rank(X) < rank(D)$. If this is true, then the determinant of any element of $\mathcal{R}$ gives the required polynomial. But we need to be able to efficiently perform the check, obtain an element of $\mathcal{R}$ and finally compute its determinant. To perform the check, the following lemma is of help.

Let $\bar{w} = (w_1, \cdots, w_{s_2})^T$ be the $s_2 \times 1$ vector which satisfies the matrix equation $D\bar{w} = \bar{0}$, then

Lemma 3.3 *$\exists X \epsilon \mathcal{D}_1$ s.t. $rank(X) < rank(D)$ if and only if there exists some $0 \leq i \leq s_2$ such that the $w_i = 0$ and $\mathcal{C} \Rightarrow monom(m_i) \neq 0$.*

Proof: First of all, let us write the expanded version of the matrix equation $D\bar{w} = \bar{0}$:

$$m_1 w_1 + m_2 w_2 + \cdots + m_{s_2} w_{s_2} = \bar{0}$$

Only if: Let $X \epsilon \mathcal{D}_1$ *s.t.* $rank(X) < rank(D)$. Let m_i be that row of D whose deletion resulted in X. Then, by definition of $\mathcal{D}_1$, $\mathcal{C} \Rightarrow monom(m_i) \neq 0$. Also, $rank(X) < rank(D)$, implies that m_i is linearly independent. This means that $w_i = 0$ because otherwise, the above equation can be divided by w_i and then by rearranging terms, m_i can be expressed as a linear combination of other columns, hence implying that m_i is linearly dependent and resulting in a contradiction.

If: Assume that there exists some $0 \leq i \leq s_2$ such that $w_i = 0$ and $\mathcal{C} \Rightarrow monom(m_i) \neq 0$. The latter assumption means that the matrix obtained by deleting m_i from D (say X) must be in $\mathcal{D}_1$. But also since $w_i = 0$, it follows that column m_i is linearly independent of other columns of D, hence implying that $rank(X) < rank(D)$. □

Finally, in accordance with theorem 3.2, we would like to get the determinant of any element in $\mathcal{R}$ which will give the required necessary condition on the parameters. This is achieved due to the following lemma. Let D_{row} have the following properties:

1. D_{row} is row-reduced, i.e., each column of D_{row} which contains the leading non-zero entry of some row has all its other entries 0.
2. D_{row} is row-equivalent to D, i.e., D_{row} can be obtained from D by a finite sequence of the following two steps:
   (a) **Elimination step:** Replacement of i^{th} row of D by the i^{th} row plus d times j^{th} row, d is any rational function in the parameters, and $i \neq j$.
   (b) **Pivoting step:** Interchange two rows (say the i^{th} and the j^{th} rows) of D.

Notice that D_{row} can be constructed from D by simple Gaussian elimination and many computer algebra systems already support such an operation ($D_{row} = gausselim(D)$ in MAPLE).

Lemma 3.4 *There exists some $Y \epsilon \mathcal{R}$ such that the product of all the pivot elements of D_{row} is equal to $det(Y)$.*

Proof: Assign to the j^{th} row of D, a label l_j. Now obtain the matrix D_{row} which is row-equivalent to D by successively applying a permutation of the above two steps, except that whenever the pivoting step is applied, interchange the labels of the rows too. In the elimination step, the labels remain the same. Since the rank of D is r, D_{row} will have r pivots. Let $m_{i_1}, \cdots, m_{i_r}$ be the pivot columns, and $l_{j_1}, \cdots, l_{j_r}$ the labels of pivot rows of D_{row}. Then it is easy to show that the product of all pivot elements of D_{row} is the determinant of the $r \times r$ submatrix (say Y) of D obtained by the rows labelled by $l_{j_1}, \cdots, l_{j_r}$ and the columns indexed by $m_{i_1}, \cdots, m_{i_r}$ in D. Since the product of pivot elements is not identically zero, Y is not singular so it is in $\mathcal{R}$ by definition. □

Based on theorem 3.2 and lemmas 3.3 and 3.4, we can get the following algorithm. Assume that the set of $n+1$ polynomials $\mathcal{F} \subset \mathcal{P}[x_1, \cdots, x_n]$ and the constraints $\mathcal{C}$ are given. The following algorithm checks if the precondition in theorem 3.2 is true in which case it returns ϱ_C. If the precondition of theorem 3.2 is not true, then this heuristic fails.

1. Set up the $s_1 \times s_2$ Dixon matrix (D) of $\mathcal{F}$.
2. Solve the matrix equation $D\bar{w} = \bar{0}$. Let $\bar{w} = (w_1, \cdots, w_{s_2})$ denote the solution.
3. Find out if there exists an w_i in $\bar{w}$ such that $w_i = 0$ and also $\mathcal{C} \Rightarrow monom(m_i) = 0$. If such an w_i exists then
   (a) Compute D_{row}.
   (b) Return the product of all the pivots of D_{row}.
4. Else, this heuristic fails.

Theorem 3.5 *If the Dixon matrix of a set of polynomials satisfies the precondition of theorem 3.2, then the above algorithm computes ϱ_C.*

Proof : Follows directly from theorem 3.2 and lemmas 3.3 and 3.4. □

There are two points to be noted here. First, though there exist examples where the condition in step (3) is not true, they are rare. In all the examples from geometry that we tried, D was singular many times, but the condition of step (3) always held. Secondly, once the polynomial in the parameters is obtained from the algorithm, it can usually be factored. Since the vanishing of this polynomial is only a necessary condition, occasionally it will be the case that some of the factors do not vanish on any of the solutions. These factors can be removed to obtain a smaller ϱ_C. This is done manually as suggested by specific algebraic or geometric problem. Our experience has been that it is usually easy to identify such extraneous factors, but automatic method to accomplish this need to be further explored. In the next section, we discuss some examples.

4 Geometric Reasoning using Dixon resultants

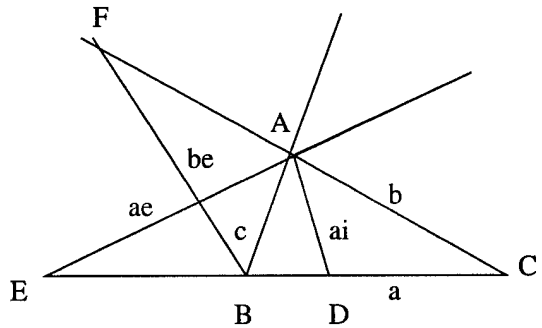
We implemented the proposed algorithm on a SPARCstation 10 in MAPLE using the primitive operations such as *linsolve, gausselim*, etc., available in its linear algebra package. No attempt was made to optimize Maple code. We believe the algorithm can be made much faster using interpolation techniques for computing determinants as reported by Canny and Manocha in [13].

Two geometric identities derived using the algorithm are presented in this section. A few more identities will be presented without details in a later section.

Example 1: Side-Bisector Relation

Let ABC be a triangle as in the following figure, a, b and c the length of the sides BC, AC and AB, a_i and a_e the length of internal (AD) and external (AE) bisectors of angle A, and b_e the length of the external angle bisector (BF) of angle B. The objective is to express a in terms of a_i, a_e and b_e.

This problem was first posed by Heymann in [11]. He wanted to determine if, given *general* values of the three angle bisectors, can one draw the triangle using only a compass and a ruler? This is possible if and only if the expression involving a, a_i, a_e and b_e is of degree 2^m in a for some integral m (see corollary 2 of Theorem 5.4.1 in [10]). This example was again posed in [8] where they solved this problem, and we have presented it exactly in the same way.



It is a standard result of Euclidean geometry that:

$$a_i^2 = \frac{cb(c+b-a)(c+b+a)}{(b+c)^2}$$

$$a_e^2 = \frac{cb(a+b-c)(c-b+a)}{(c-b)^2}$$

$$b_e^2 = \frac{ac(a+b-c)(c+b-a)}{(c-a)^2}$$

Hence, it is easy to express the length of the bisectors in terms of the length of the sides. The challenge is to express the length of the sides in terms of the length of these three bisectors. In principle, this is a simple elimination problem, i.e., if we can eliminate the variables b and c from any two equations, then we can plug the expressions for these two variables into the third equation to obtain an expression for a in terms of only a_i, a_e and b_e.

This can be achieved by computing the resultant of these three polynomials w.r.t. the two variables, b and c. To this effect, let us first represent these equations as polynomials by transforming them to the following:

$$q_1 = a_i^2(b+c)^2 - cb(c+b-a)(c+b+a)$$

$$q_2 = a_e^2(c-b)^2 - cb(a+b-c)(c-b+a)$$

$$q_3 = b_e^2(c-a)^2 - ac(a+b-c)(c+b-a)$$

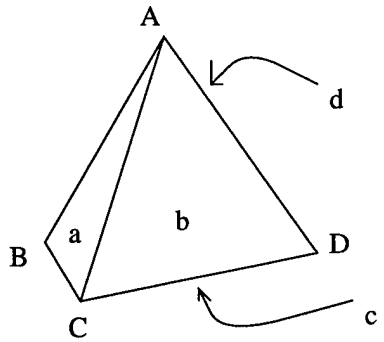
The objective is to eliminate b and c without any constraints on them. We discuss the trace of our algorithm on this set in the next paragraph.

First the Dixon polynomial was computed, followed by the Dixon matrix (M) of this set which turned out to be 13×14. Solving the matrix equation $M\bar{x} = \bar{0}$ resulted in a vector whose component corresponding to the monomial $b^0c^0 = 1$ is 0 and this took 198 seconds. Hence the condition in step (2) of the algorithm is true. Now Gaussian elimination was performed. Gaussian elimination took 284 seconds. The product of the pivot elements of the matrix after Gaussian elimination is the necessary condition that a, a_i, a_e and b_e must satisfy for any triangle ABC. The total computation took 501 seconds. After removing extraneous factors, the smallest necessary condition was obtained and it contains 330 terms. The result is the same as the one reported in [8]. Since the expression is of degree 20 in a, the triangle cannot be constructed using a compass and a ruler, given a_i, a_e and b_e (see corollary 2 of Theorem 5.4.1 in [10]).

Gao and Wang in [8] reported that they solved the problem by successively eliminating b and c using a combination of pseudo division and Sylvester's resultant computation. They took about 19 hours on their implementation in lisp on a SUN 4 workstation. Our algorithm took less than 17 minutes on a SUN 4, and less than 9 minutes on a SPARCstation 10. □

Example 2: Maximum Volume of a Tetrahedron

Consider a tetrahedron as below:



The objective is to determine the maximum volume that a tetrahedron can have, given that the squares of the areas of the four faces, ABC, ACD, BCD and ABD are a, b, c and d respectively.

It has been established in [9] that if there exist parameters x, y, z and w which satisfy the following four equations:

$$\begin{aligned} yz + zw + wy - a &= 0 \\ zx + xw + wz - b &= 0 \\ wx + xy + yw - c &= 0 \\ xy + yz + zx - d &= 0 \end{aligned}$$

then the tetrahedron is an orthocentric tetrahedron and hence the one with the maximum volume for these surface areas. Moreover, the square of the volume (T) of this tetrahedron is:

$$T = \frac{2}{9}(xyz + yzw + zwx + wxy)$$

How does one get the value of T purely in terms of a, b, c and d? This problem translates to eliminating x, y, z and w from the above mentioned five equations, i.e., computing the resultant of the following five polynomials:

$$\begin{aligned} q_1 &= yz + zw + wy - a \\ q_2 &= zx + xw + wz - b \\ q_3 &= wx + xy + yw - c \\ q_4 &= xy + yz + zx - d \\ q_5 &= 2(xyz + yzw + zwx + wxy) - 9T \end{aligned}$$

with respect to x, y, z and w. It turns out that any of these variables x, y, z and w being zero is a degeneracy, hence we can work under the constraint $C = x \neq 0 \wedge y \neq 0 \wedge z \wedge w \neq 0$. We now discuss the trace of the program.

The Dixon matrix was set up and it was found to be of dimension 16×16. Solving the equation $M\bar{x} = \bar{0}$ took 10 seconds and it was found that the component of $\bar{x}$ corresponding to the monomial x was 0. So the condition of step 3 is satisfied and Gaussian elimination was performed which took 84 seconds. The necessary condition for the existence of an affine zero satisfying C was computed as the product of the pivots entries of the matrix which was obtained after Gaussian elimination. It was found that there was no extraneous factor in the necessary condition, hence it is the smallest necessary condition and it contained 434 terms. The total time taken was 110 seconds. □

5 Advantages of Dixon resultants

In our experimentation with this technique, we found that this technique is faster than successive Sylvester resultant computation for computing resultants for both cases - (1) two polynomials and single variable to be eliminated, and (2) more than two polynomials with more than one variable to be eliminated. In the case when the Dixon matrix is singular, this technique was also found to be faster than perturbation techniques. The reasons are as follows.

1. **Smaller Determinant :** In the one variable case, it was seen in section 2.1 that the resultant is the determinant of an $max(m,n) \times max(m,n)$ Dixon matrix (where m and n are the degrees of the two polynomials). This is a much smaller matrix than the $(m+n)\times(m+n)$ matrix that one gets using Sylvester's formulation. The smaller matrix leads to **reduced time.**

2. **Uniform Approach :** When more than one variables need to be eliminated, Dixon's method works better because all polynomials and variables are treated uniformly. The method particularly works best in the case of problems (or polynomials) which are symmetric in the variables which need to be eliminated. One possible reason is that Dixon's method adopts a symmetric approach and a single Dixon matrix is set up for all the polynomials together. This is in contrast to the successive resultant computation techniques which eliminate variables one by one, and compute numerous intermediate resultants before successively computing the resultant of the whole set. This ordering among the variables breaks the symmetry of the problem because of which huge intermediate polynomials before the (relatively small) resultant is obtained. This usually turns out to be costly. Dixon's method avoids such intermediate polynomials and hence is **much faster** and also **saves space.** This is the main reason that our algorithm is able to prove theorems and derive identities substantially faster than other methods.

3. **Faster than Perturbation :** Perturbation usually is an expensive operation as opposed to the approach presented in this paper. Dixon resultants (and in fact most resultant formulations in general) are sensitive to the number of variables, the degree of each polynomial and also the distribution of variables (a variable occurring in a few polynomials is better than it occurring in a lot of polynomials). If the Dixon matrix is singular, then a generic perturbation (a la [3]) can also be performed to obtain a projection operator which is not identically zero. However, the perturbation variable is introduced at the highest possible degree, in every polynomial. Our method, on the other hand, avoids perturbation in many cases. In the proposed method, the projection operator is extracted from the Dixon matrix of the original system, **without any extra computation.** This results in a method which is more efficient than perturbations and this is reflected in all the examples of this paper.

4. **Automatic Method :** Methods based on variable orderings or which employ successive elimination also suffer with human intervention. A good ordering must be specified or the order in which elimination is performed must be given. This seems unavoidable since the time taken by successive elimination techniques is sensitive to the variable ordering used. Dixon's method, on the other hand, does not eliminate the variables in any particular order. Instead, this method directly computes the resultant; thus being **fully automatic.** In fact, never once did we have to interfere during the proofs of the geometry theorems and algebraic identities mentioned in this paper.

6 Empirical results

We present more examples, and give the following characteristics about each example in table 1:

(a) Sing : Whether the Dixon matrix was singular (s) or nonsingular (n).

(b) Terms : Number of terms in the smallest necessary condition for affine zeros.

(c) Dix : Time taken by an implementation of our algorithm in MAPLE on a SPARCstation 10.

(d) Gröb : Time taken by MACAULAY system [1] on a SPARCstation 10 for computing the Gröbner basis using block ordering where the first block contains all the variables and the second all the parameters. Variables among the same block are degree ordered, and across the blocks, they are lexicographically ordered.

(e) Mac/GCP : Time taken to compute the Macaulay resultant (GCP when Macaulay matrix is singular [3]) using MAPLE on a SPARCstation 10.

A (*) indicates that either the program went on for more than a day, or it ran out of space.

We tried successive resultant computation using the preexisting implementations of Sylvester's resultants and Bezout's resultants in MAPLE for various variable orderings, but the computation ran out of memory for every example. We also tried to compute the lexicographic Gröbner bases to obtain the conditions for the common zeros. Neither MAPLE, nor MACAULAY (Bayer and Stillman [1]) could compute the Gröbner basis of any of the examples in this paper. For lexicographic ordering among all the variables, MACAULAY ran for upto a day on some of these examples and then ran out of memory. When block ordering (as described in (d) above) was used, MACAULAY successfully terminated after a long time on three of the examples (2, 4 and 5), but the remaining two (1 and 3) went on for a day and still did not terminate.

Example	Sing	Terms	Dix	Gröb	Mac/GCP
1	s	330	501s	*	*
2	s	434	110s	585s	*
3	n	2424	9.6s	*	39s
4	s	990	154s	15429s	*
5	n	781	2.4s	2207s	850s

Table 1: Comparision of Various Methods

Attempt was made to compute the Macaulay resultant (GCP in case the Macaulay matrix was singular [3]) on MAPLE, but none of the GCP computations (examples 1, 2 and 4) ever finished. In all those cases, the computation ran for upto a day before running out of space. The resultant for the examples in which GCP was not required (examples 3 and 5) successfully terminated, but took longer time than our method.

For all the examples in which the Dixon matrix is singular, perturbation techniques ([3]) were also tried. One can perturb the polynomials so that they become ndegree and the Dixon matrix is no longer singular. This technique also failed for each example in this paper because the Dixon matrix blows up after perturbation, hence resulting in a substantially larger Dixon matrix with larger polynomial entries. The determinant computation for these matrices ran out of space for all the examples on MAPLE.

For each example, the variables are one or more of x, y and z. The polynomials whose resultant needs to be computed are two or more of q_i's. The numbering of the examples follows after the previous two examples.

Example 3: Expression for the distance of the intersection of two general conics from the origin.

$$\begin{aligned} q_1 &= a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \\ q_2 &= b_1x^2 + b_2xy + b_3y^2 + b_4x + b_5y + b_6 \\ q_3 &= x^2 + y^2 - T \end{aligned}$$

Example 4: Conditions for perpendicular intersection of a general conic and a general circle.

$$\begin{aligned} q_1 &= a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \\ q_2 &= x^2 + y^2 + b_1x + b_2y + b_3 \\ q_3 &= \frac{dq_1}{dx}\frac{dq_2}{dx} + \frac{dq_1}{dy}\frac{dq_2}{dy} \end{aligned}$$

Example 5: Conditions for the following four equations to have a common solution.

$$\begin{aligned} q_1 &= a_1x + a_2y + a_3z + a_4 \\ q_2 &= b_1xy + b_2yz + b_3zx + b_4 \end{aligned}$$

$$q_3 = c_1xyz + c_2$$
$$q_4 = d_1xyz + d_2x + d_3y + d_4z$$

7 Conclusion

Dixon's method for simultaneously eliminating several variables is discussed. The method is extended for the case when Dixon matrix is singular. An algorithm is given to extract a nonzero projection operator for a subclass of the systems of multivariate polynomials from its singular Dixon matrix. This algorithm avoids perturbation.

A great deal of work needs to be done for further investigating Dixon's method. In particular, there is a need to further understand Dixon's method in the general case. Determinant computations of matrices with polynomial entries are a major bottleneck in the method. Fast methods based on interpolation, similar to those in [13], need to be investigated in order to make Dixon's method more widely applicable.

Acknowledgements : We would like to thank Lakshman Y. N. for useful discussions.

References

[1] Bayer D., Stillman M., *Macaulay User's Manual*, Cornell University, Ithaca, NY.

[2] B. Buchberger, Gröbner bases: An Algorithmic method in Polynomial Ideal theory, in *Multidimensional Systems Theory*, N.K. Bose, ed., D. Reidel Publishing Co., 184-232, 1985.

[3] Canny J., Generalized Characteristic Polynomials, *Journal of Symbolic Computation*, 1990, 241-250.

[4] Cayley, A., On the theory of elimination. *Cambridge and Dublin Mathematical Journal*, III, 1865, 210-270.

[5] Chionh E., *Base points, resultants, and the implicit representation of rational Surfaces.* Ph. D. thesis, Dept. Comp. Sci., Univ. of Waterloo, 1990.

[6] Chou, S.-C., Gao, X.-S., Methods for Mechanical Geometry Formula Deriving. *Proc. ISSAC 1990*, ACM press, 1990.

[7] Dixon, A.L., The eliminant of three quantics in two independent variables. *Proc. London Mathematical Society*, 6, 1908, 468-478.

[8] Gao, X.-S., Wang, D.-K., On the Automatic Derivation of a Set of Geometric Formulae. *Mathematics Mechanization Research Preprints* No. 8, 1992, 26-37, Academia Sinica, China.

[9] Gerber, L., The Orthocentric Simplex as an Extreme Simplex. *Pacific Journal of Mathematics* 56, 1975, 97-111.

[10] Herstein, I. N., *Topics in Algebra - Second Edition* John Wiley & Sons, Inc., USA, 1975.

[11] Heymann, W. Problem der Winkehalbierenden *Ztschr. f. Math. and Phys.* No. 35, 1890.

[12] Kapur, D., Lakshman, Y., Elimination Methods: an Introduction. *Symbolic and Numerical Computation for Artificial Intelligence* Donald, Kapur and Mundy (eds.), Academic Press, 1992.

[13] Manocha D., Canny J.F., Multipolynomial Resultant Algorithms. Tech. Report, *University of California, Berkeley*, 1991.

[14] W. Wu, On the decision Problem and the mechanization of theorem proving in elementary geometry, *Scientia Sinica*, 21, (1978), 150-172. Also in Bledsoe and Loveland, eds., *Theorem Proving: After 25 years, Contemporary Mathematics*, 29, 213-234. 1984.

Viewpoint-Invariant Representation of Generalized Cylinders Using the Symmetry Set

Nic Pillow, Sven Utcke and Andrew Zisserman
Robotics Research Group, Department of Engineering Science
Oxford University, OX1 3PJ.

Abstract

We demonstrate that viewpoint-invariant representations can be obtained from images for a useful class of 3D smooth object. The class of surfaces are those generated as the envelope of a sphere of varying radius swept along an axis. This class includes canal surfaces and surfaces of revolution.

The representations are computed, using only image information, from the symmetry set of the object's outline. They are viewpoint-invariant under weak-perspective imaging, and quasi-invariant to an excellent approximation under perspective imaging. To this approximation, the planar axis of a canal surface is recovered up to an affine ambiguity from perspective images.

Examples are given of the representations obtained from real images, which demonstrate stability and object discrimination, for both canal surfaces and surfaces of revolution. Finally, the representations are used as the basis for a model-based object recognition system.

1 Introduction

The aim of this work is to extract viewpoint-invariant descriptions of 3D smooth objects from single images. These descriptions will be used as shape descriptors in a model-based recognition system. For a completely general object, and with no other information, it is not possible to recover shape or invariant descriptions from a single image (see for example [7, 9, 16] for 3D point sets). However, if the 3D structure is *constrained*, then invariant descriptions can be obtained. Here we consider surfaces belonging to a particular class of generalized cylinder (GC) [1]. The class consists of surfaces generated as the envelope of a *sphere* of varying radius swept along the cylinder axis (which need not be straight). Examples include pipes or tubes ('canal surfaces') where the sphere radius is constant, and surfaces of revolution, where the axis is a line. This class generates a large number of commonly occurring manufactured objects.

The key idea here is that since the 'envelope of the profile[1] is the profile of the envelope' [25] the image projection of this class of surface is an envelope of circles.

[1]The image outline of the surface.

By inverting this process—recovering circles from the profile—the projection of the axis and the scaling function can be extracted from the image.

Previous work on extracting GC's from images has had different or more limited goals: first, rather than perspective, the weak perspective imaging approximation has generally been used (see [20, 30], where earlier references are given); second, the goal has been reconstruction rather than representation, and this requires a reference cross-section to be visible in the image [28]; third, those methods that have produced invariants under perspective [11] have only utilized a number of points on the image outline—not the entire curve. In this paper, in common with the above, only the profile is used; no use is made of surface markings or texture.

The tool employed here is the *symmetry set* [5], studied by Giblin & Brassett [12], which is the locus of centres of circles bitangent to a plane curve. Previous symmetry analysis has largely concentrated on extracting bilateral (reflection) or rotational symmetries from images of planar objects, or from a single silhouette of 3D objects, viewed in a fronto-parallel plane [3, 4, 26]. The methods developed for those cases can not be applied if the viewpoint is not fronto-parallel since, for a planar object, reflectional symmetries are then *skewed* by imaging [13, 17, 29]. For a 3D smooth object additional distortions occur—the contour generator[2] moves on the surface as viewing position changes. In this case it is not a fixed space curve which is projected, and the image profile can change radically with viewpoint, defeating any simple application of skewed symmetry.

2 Theory

In the following we consider two types of image projection: perspective and weak perspective. In both cases the camera aspect ratio must be known, though no other intrinsic parameters are required. However, much of the construction uses only affine or projective properties, and this is made explicit.

2.1 Weak Perspective

Consider sweeping a sphere of varying radius along the axis curve: the resulting surface is the envelope of the swept sphere. At each point the profile of the sphere is a circle, and the profile of the surface is the envelope of the circles. Under affine projection the centre of the sphere projects to the centre of the circle. Consequently, the circle's centre sweeps out the projection of the axis. This construction is illustrated schematically in figure 1.

Now consider two such circles: as the scaling arising from weak perspective is the same in both cases, the ratio of circle radii equals the ratio of radii of the generating spheres. The usefulness of these results is that the circles can be recovered from the profile by constructing the symmetry set, the locus of centres of circles bitangent to the profile. To summarize:

[2]The curve on the surface which projects to the image profile.

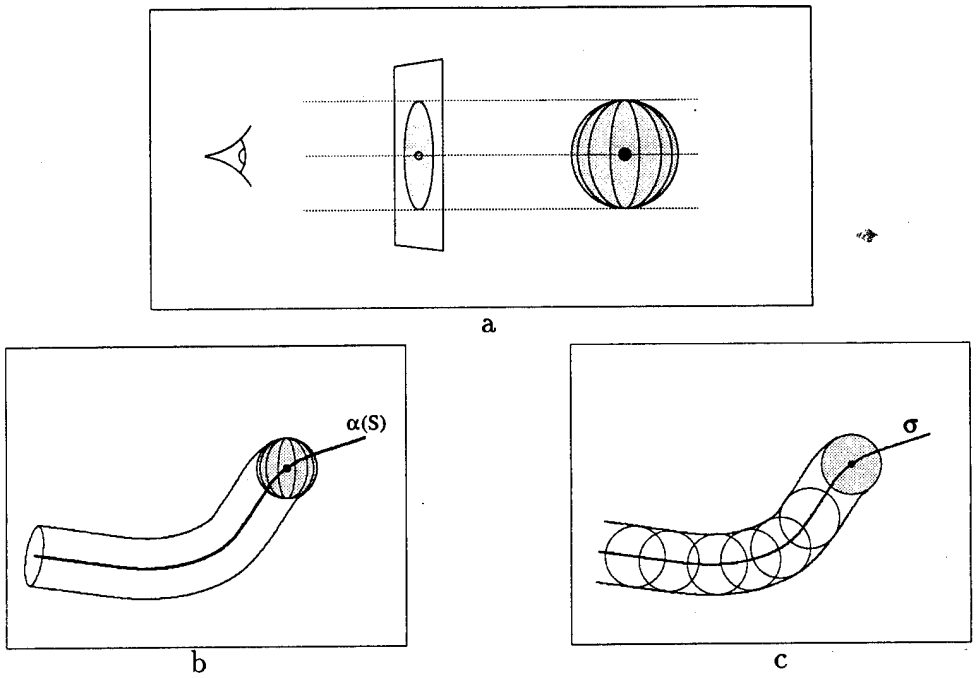


Figure 1: *(a) Under weak perspective projection the profile (image outline) of a sphere is a circle, and the sphere centre is projected to the circle centre. (b) The surface is generated as the envelope of a sphere swept along an axis α. (c) The surface profile is generated as the envelope of a circle swept along σ, where the image curve σ is the projection of the axis curve α. The figure illustrates the case for a constant radius sphere — a canal surface. The profile is still generated (locally) as the envelope of circles when the sphere radius varies.*

Given a weak perspective image of a surface generated (locally) as the envelope of a sphere of varying radius $R(S)$, with centres on a plane curve $\alpha(S)$ (the axis); the symmetry set, computed from the image profile, has the following properties:

1. *The contact of image circles with the profile identifies corresponding points on the surface, i.e. points which lie on the same circular cross-section.*

2. *The curve covered by circle centres, $\sigma(s)$, is within a plane affine transformation of the curve $\alpha(S)$.*

3. *The scaling function for radii of image circles, $r(s)$, equals the scaling function for radii of the generating spheres $R(S)$ for corresponding points, $\alpha(S)$ and $\sigma(s)$, up to an overall scale ambiguity.*

Note, only the second item requires that the axis be planar. The other properties hold if the axis is a space curve.

The curves $\{\sigma(s), r(s)\}$ are a viewpoint-invariant representation. $\alpha(S)$ is determined up to an affine ambiguity, and consequently affine invariants of $\alpha(S)$ are equal to affine invariants of $\sigma(s)$, and these are viewpoint-invariant. $R(S)$ is determined up to scale. The parametrization of $\sigma(s)$ is described in section 2.4.

2.2 Perspective

Under perspective projection the profile of a sphere is an ellipse, and the centre of the sphere does not project to the ellipse centre. However, in practice, for finite image planes this effect is extremely small: the aspect ratio of a sphere's profile is at worst 0.94, and its centre is displaced at most by 1.2% of its diameter, even at the border of a wide-angle lens with a 46° field of view. This is an example of a *quasi-invariant* [2]. Thus, the symmetry set will still be an excellent approximation of the projected axis. However, the transformation between the axis and image curves will be projective, rather than affine. To summarize:

Given a perspective image of a surface generated (locally) as the envelope of a sphere of varying radius $R(S)$, with centres on a plane curve $\alpha(S)$ (the axis); the symmetry set, computed from the image profile, (approximately) has the following properties:

1. *The contact of image circles with the profile identifies corresponding points on the surface (i.e. points on the same circular cross-section).*

2. *The curve covered by circle centres, $\sigma(s)$, is within a plane projective transformation of the curve $\alpha(S)$.*

The curve $\{\sigma(s)\}$ is a viewpoint-invariant representation. $\alpha(S)$ is determined up to a projective ambiguity, and consequently projective invariants of $\sigma(s)$ are viewpoint-invariant. Furthermore, it is shown in section 4.2 that for canal surfaces with at least two inflections, $\alpha(S)$ is determined up to an affine ambiguity, even under perspective distortion.

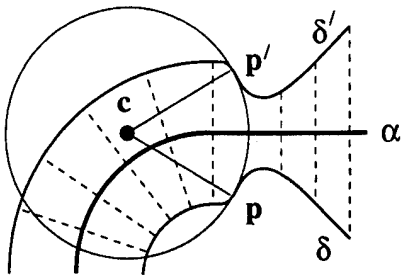


Figure 2: *The ribbon* $\{\delta(s), \delta'(s)\}$ *is generated by sweeping a line of varying length, such that the line's mid-point lies on the axis* α*, and the line is orthogonal to the axis. In general the symmetry set of this type of ribbon will not coincide with the axis. For example, the centre* $\mathbf{c}$ *of the symmetry set at* $\mathbf{p}, \mathbf{p}'$ *does not lie on* α*.*

2.3 Surface Class and Special GCs

The surface class, $\mathcal{S}$, considered here consists of those surfaces generated as the envelope of a sphere swept along a plane curve. For these surfaces, by construction, a sphere with a circle of points tangent to the surface is one of the generating spheres, and will have its centre on the axis. Consequently, the projected axis and profile symmetry set coincide.

Generalized cylinders are often defined in terms of sweeping a planar cross-section along an axis [1]. However, if a surface is generated by sweeping circles of varying radius orthogonal to a planar axis, then the centre of a sphere with a circle of points tangent to the surface will *not*, in general, coincide with the axis. To see this, consider the intersection of such a surface with the plane of the axis. The intersection is a pair of plane curves, $\{\delta(s), \delta'(s)\}$ which are generated by the ends of a swept line. The line is orthogonal to the axis, with its mid-point on the axis, and the line length varies according to the radius of the circle. In general, the symmetry set of $\{\delta(s), \delta'(s)\}$ will *not* coincide with the axis. Consequently, the centre of a sphere with a circle of points tangent to the surface can not lie on the axis. An example is shown in figure 2. A full discussion of the 2D case is given by Ponce [21] where the line and circle sets are termed Brooks and Blum ribbons respectively.

Two special cases of surfaces in $\mathcal{S}$, which are defined by additional constraints, are considered throughout the rest of the paper. In the following we list a number of properties for these cases which do not hold for an unconstrained member of $\mathcal{S}$. These properties are used during the image processing (grouping and symmetry set extraction), and in the construction of a canonical frame. Both these special cases can also be generated by sweeping circles.

2.3.1 Canal surfaces

Here there is an additional constraint which is that the scaling function is a constant. These surfaces are also called Circular Planar Right Generalized Cylinders with constant cross-section.

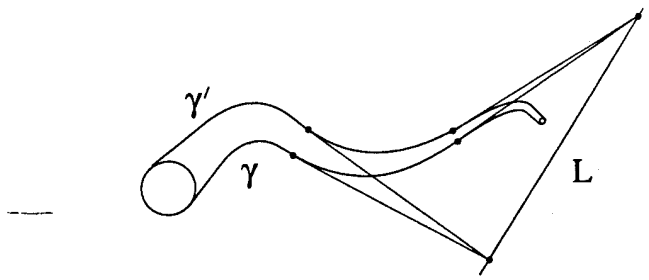


Figure 3: *For a canal surface with a planar axis, inflections in the profile occur in pairs for each inflection of the axis. The intersection of a pair of inflection tangents determines the vanishing point of the tangent line at the axis inflection. Two such vanishing points determine the vanishing line, L, of the plane of the axis. This line can be used to group further profile inflection tangents (i.e. the pairs of tangents must intersect on this line).*

1. Under weak perspective projection, the two sides of the profile are parallel curves of the symmetry set (the projection of the axis). This follows directly from the profile curves being the envelope of constant-radius circles swept along the symmetry set.

2. Inflections in the axis produce inflection pairs on the profile. Tangents at corresponding profile inflections (on each side of the profile) intersect on the vanishing line of the axis curve's plane. Two such intersections determine the vanishing line, and hence the extraction of affine curve measurements; see figure 3. This relationship is exact—it is not a quasi-invariant. Note: this constraint also applies to line segments on the profile (a line is a 'degenerate' inflection), so can be applied to a piecewise linear axis.

2.3.2 Surfaces of revolution

Here the additional constraint is that the axis is straight. Consequently, the scaling function is the most informative component of the viewpoint-invariant representation. A surface of revolution is a special case of a Straight Homogeneous Generalized Cylinder.

1. Tangents at corresponding points on the two sides of the profile (i.e. images of points on the same circular cross-section) intersect on the projected symmetry axis [20].

2. In particular, corresponding profile bitangents intersect at an image point $\mathbf{p}$, which is the image of $\mathbf{P}$, the point of intersection between the object axis and planes bitangent to the surface. $\mathbf{P}$ is viewpoint-independent [11], and $\mathbf{p}$ can be identified in any image.

3. The profile can be separated into two 'sides', which are related by a planar harmonic homology [17, 18] (i.e. a projective transformation $\mathtt{T}$ such that $\mathtt{T}^2 =$

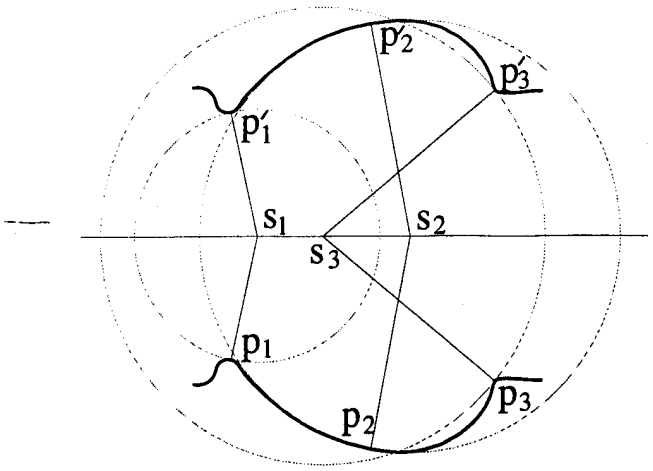


Figure 4: *Three of the fitted circles for a surface of revolution. Note that the centre's position on the axis of symmetry, s, is not a monotonic function of distance along the profile, p.*

I [27]). T provides point to point correspondence between the sides of the profile, thus greatly simplifying the processing. Provided the field of view is not too large, T is approximately affine [15]; this is another example of a quasi-invariant.

2.4 Curve Parametrization

Consider the motion of circle centres for a point progressing along the profile. If the circle radius increases, then the centre can reverse direction and 'double back' on itself. This does not occur for canal surfaces (constant radius) but will occur in general, and is clearly demonstrated in figure 4. Correspondingly, the centres of the spheres generating the surface (a 3D symmetry set) double back. There are thus two parametrizations to be considered: first, the parametrization of the axis, S, (which is a monotonic function of distance along the axis of, for example, a surface of revolution); and second, the parametrization of the symmetry set. This latter parametrization is not a monotonic function of S, in general. The figures give results parametrized by S. If, instead, symmetry set parametrization were used, it would avoid the doubling back in figures 9 and 10, though there would still be cusps in the graph.

2.5 Self-Occlusion

The discussion to this point has not considered cases where the surface occludes itself under projection. Since the axis α is planar, the points on the imaged axis σ are, at worst, a plane projective transformation of points on the axis. Provided the axis is a smooth curve, then its image is also smooth since a plane projective

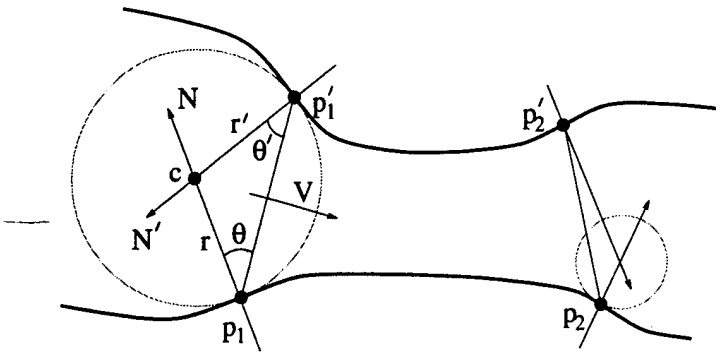


Figure 5: $\mathbf{N}$ *and* $\mathbf{N}'$ *are normals to the profile curves at* $\mathbf{p}$ *and* $\mathbf{p}'$ *respectively.* $\mathbf{V}$ *is the unit vector orthogonal to* $\mathbf{p}-\mathbf{p}'$. *A circle bitangent to the curves at those points has centre* $\mathbf{c}$ *at the intersection of* $\mathbf{N}$ *and* $\mathbf{N}'$, *with* $r = r'$ *and* $\theta = \theta'$. $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *illustrate corresponding and non-corresponding points respectively. One possible measure of correspondence is* $|r - r'|$, *but this is insensitive and requires normalization, since it varies with* $||\mathbf{p} - \mathbf{p}'||$. *The implemented measure is* $|\sin\theta - \sin\theta'|$.

transformation does not introduce cusps. Consequently, the symmetry set of the profile will be smooth, even if the profile contains cusps. (It is assumed that the symmetry set is determined from correctly corresponding points of the profile even when the profile self-intersects, for example, in a swallowtail.)

For an opaque object, parts of the contour generator are occluded, and consequently parts of the profile are 'missing' (compared to the profile of a semi-transparent object). Again, provided the symmetry set is determined for correctly corresponding points on the profile, the symmetry set will be smooth, but some parts of the axis may not appear in the image (i.e. the symmetry set may contain gaps corresponding to the 'missing' parts of the profile).

For a surface of revolution self-occlusion occurs simultaneously (i.e. at the same circular cross section) on both sides of the profile. This is not the case, in general, for a canal surface.

3 Extracting the Symmetry Set

3.1 Initial Processing

Edges are extracted to sub-pixel accuracy using a local implementation of the Canny edge operator [8]. These are linked into edgel-chains by a sequential linker which extrapolates over small gaps. Accurate curve normals are computed at each point of the edgel-chain by locally fitting a quadratic using least squares regression with central Gaussian weighting on a thirteen point neighbourhood.

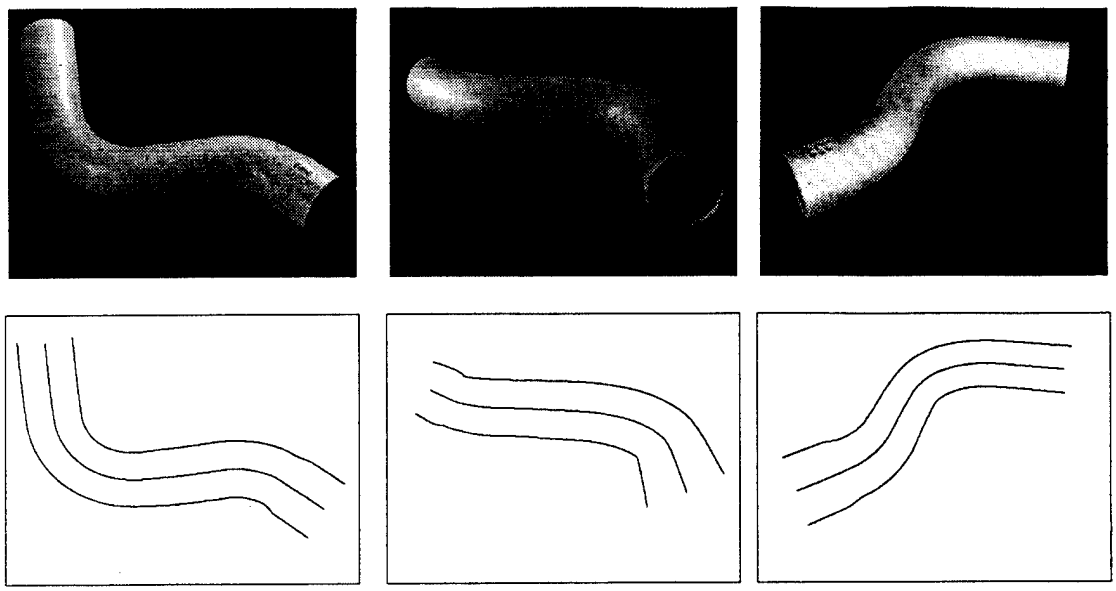


Figure 6: *Upper: images from substantially different viewpoints of the same canal surface (pipe 1). Lower: the profiles and extracted symmetry sets. Note the radically different shapes of the profiles and symmetry sets. However, in a canonical frame the three symmetry sets are virtually identical—see figure 7c.*

3.2 Canal Surfaces

3.2.1 Bitangent circles

The procedure for generating the symmetry set involves working along two curves, γ and γ', simultaneously. The aim is to fit a circle which is tangent simultaneously to points p and p' on each curve. If such a circle exists then its centre lies on the intersection of the normals at p and p', and is equally distant from p and p'. To determine such points, a 'correspondence measure' is computed as

$$\mathcal{M}(p,p') = |\mathbf{V}\cdot\mathbf{N} - \mathbf{V}\cdot\mathbf{N}'| = |\sin\theta - \sin\theta'|;$$

see figure 5.

For a given pair of points, the nearer to zero the measure, the closer those points are to corresponding. A point on the symmetry set is generated by selecting a point, p, on one curve, and then determining the point on the other curve, p', which matches most closely according to the value of $\mathcal{M}(p,p')$. The symmetry set point is calculated analytically by intersecting the normals of the least squares quadratics associated with points p and p'.

The construction of the whole symmetry set proceeds by selecting successive points of both curves and generating a symmetry set point from each as described above. Checks are included to ensure that symmetry set points are properly ordered, and that no pairing of profile points generates more than one symmetry set point.

Examples of extracted symmetry sets for canal surfaces are shown in figure 6. The pipes used here, and in subsequent images, have piecewise constant radius rather

than constant radius. These belong to a larger class of surface, but provided the radius is constant in the neighbourhood of axis inflections (so that vanishing lines can be determined) the theoretical results of section 2.3.1 are still valid. However, extracting a symmetry set is more complicated in this case, since the process must cope with abrupt changes in radius.

3.3 Surfaces of Revolution

The extra constraints available in this case are used to simplify and improve the processing. The imaged axis is computed directly from the profile, without fitting circles. The scaling function is obtained from radii of bitangent circles with centres constrained to the axis.

3.3.1 Calculation of the axis

Under weak perspective the two sides of the profile are related by an affine transformation with three degrees of freedom [17]. These represent the skewed symmetry line (2 DOF) and the correspondence direction (1 DOF)[3].

The affine transformation $\mathbf{x}' = \mathtt{A}\mathbf{x}+\mathbf{b}$ which relates corresponding points $\mathbf{x}'$ and $\mathbf{x}$ on the two profile sides is determined by: first, obtaining an approximate solution by determining point correspondences from bitangent contact points; second, numerical minimization of the squared distances between one side of the profile and the other transformed by $\{\mathtt{A}, \mathbf{b}\}$, measured at a number of points along the profile length. Ten points give an excellent match for the entire profile. This determines point correspondences between the two sides of the profile. It can be shown [17] that the symmetry axis is given by

$$2a_y x - 2a_x y - a_y b_x + a_x b_y = 0$$

where $\mathbf{a}$ and $\mathbf{b}$ ($\mathbf{b}$ as above) are the eigenvectors of $\mathtt{A}$ (and have eigenvalues $+1$ and -1 respectively).

3.3.2 Calculation of the scaling function

For each point, $\mathbf{p}$, on one side of the profile, the corresponding point, $\mathbf{p}'$, on the other side is determined using the affine transformation $\{\mathtt{A}, \mathbf{b}\}$. Circles, with centres constrained to the symmetry axis, are fitted to the thirteen point profile neighbourhood of $\mathbf{p}$ and $\mathbf{p}'$, using a modified version of Pratt's circle fitting algorithm [22]. Example circles are shown in figure 4. A method for measuring the scaling function is described in section 4.1.2.

[3] If the image aspect ratio is correct, then under weak perspective the profile sides are related by a mirror symmetry [19]. The restricted affine transformation is used (only one more parameter in this case than a mirror symmetry) because the process is then tolerant to perspective effects as described in section 2.3.2.

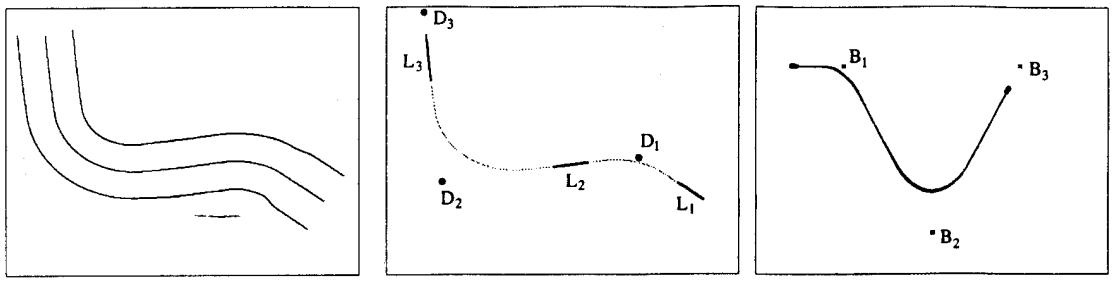


Figure 7: *The transformation of a symmetry set into an affine canonical frame. (a) Profile curves and symmetry set obtained from a weak perspective image. (b) Distinguished lines and their intersections, the three distinguished points. The transformation is found which maps the distinguished points,* $\{\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3\}$*, onto the basis points,* $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$*. (c) The result of applying this transformation to the whole symmetry set. The basis points are shown as crosses. There are six symmetry sets superimposed here. These are extracted from weak perspective images with varying viewpoint of pipe 1 (the three from figure 6 and three similar). They are virtually identical, demonstrating the stability of the affine frame.*

4 Viewpoint-Invariant Representation: The Canonical Frame Construction

A canonical frame is a method of affine [14] or projective [23] curve normalization. The normalization is achieved in two stages. First, a number of *distinguished points* (or lines) are selected on the curve. Distinguished points are ones which can be located before and after the transformation, such as corners (tangent discontinuities), inflections (zeros of curvature), and bitangent contact points. Second, the canonical frame is defined by selecting positions for a number of basis points or lines, for instance the three vertices of an equilateral triangle in the affine case. The curve is then transformed such that the distinguished points map to the basis points. All curves which are equivalent up to an affine transformation map to the same curve. In the projective case, four points or lines are required.

4.1 Affine Frame

4.1.1 Canal surfaces

The distinguished features used to define the canonical frame are the three straight segments of the symmetry set ('extended inflections'). The symmetry sets for the objects in the model-base have at most four such lines: for any given pipe, the same three must be selected. Ordering is provided simply by the symmetry set. The intersection of these three lines generates three distinguished points, which are mapped to vertices of an isosceles triangle in the canonical frame, as illustrated in figures 7a–c. The basis points chosen do not differ much from the axis curves of our

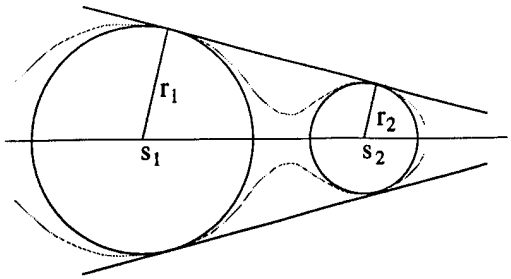


Figure 8: *The affine basis for a surface of revolution is defined on the image axis by two points, s_1 and s_2, which are the centres of circles tangent at the bitangent contact points. These basis points are the projection of the centres of spheres tangent to the surface at the circle of contact of the bitangent cone. Affine coordinates on the axis defined with respect to these points are viewpoint-invariant.*

example pipes; this avoids disproportionate enlargement or contraction of sections of any symmetry set in the canonical frame. For weak perspective images, stability over viewpoint is excellent; see figure 7c.

4.1.2 Surfaces of revolution

A canonical frame in this case simply requires an affine parametrization for the axis, and a normalization for the scaling function. The following construction requires only one bitangent on the profile. An affine parameter on the axis requires two basis points; these are the centres of the circles bitangent at the bitangent line contact points. The normalization is determined by setting the sum of the circle radii to a constant value. See figure 8. Stability over viewpoint and discrimination between objects are demonstrated in figures 9 and 10 respectively.

The utility of the scaling function as a representation for a general surface of revolution is clearly limited: first, because self-occlusion progressively erases the function; second, because the representation is biased against 'spherical surfaces'— as the surface of revolution approaches a sphere, the symmetry set reduces to a point, and the scaling function to a single value.

4.2 Projective Frame

In general, an axis curve can only be recovered up to a projective transformation from a perspective image. However, for canal surfaces, the vanishing line of the axis curve plane can be determined from profile inflection tangents by the construction of figure 3. If the vanishing line is sent to infinity, the canonical frame curve is again within an affine transformation of the axis, i.e. affine properties can be measured from the perspective image. The necessary projective transformation is computed from the following two requirements: first, the vanishing line maps to infinity, (i.e. the third homogeneous line coordinate is zero); second, the three distinguished points are mapped to the affine canonical frame basis points.

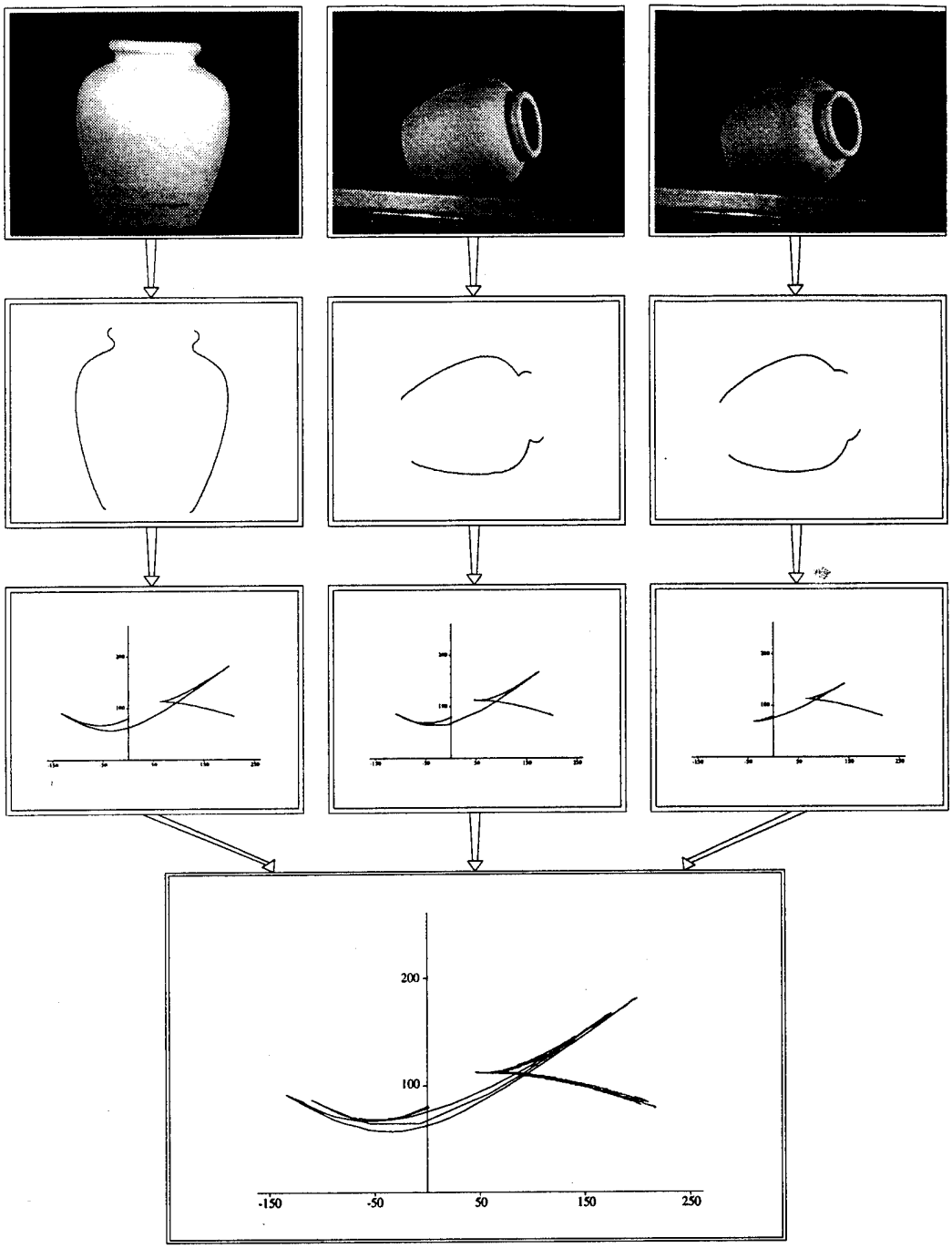


Figure 9: *Top row: weak perspective images of the same vase taken from different viewpoints. Self-occlusion increases left to right. Second row: extracted profiles. Third row: scaling plotted against an affine axis parameter. Cusps in the graph are generated at points where the profile evolute [6] crosses the symmetry axis. The comparison of the three canonical curves demonstrates stability over viewpoint. Differences arise from self-occlusion which progressively erodes the function from the left.*

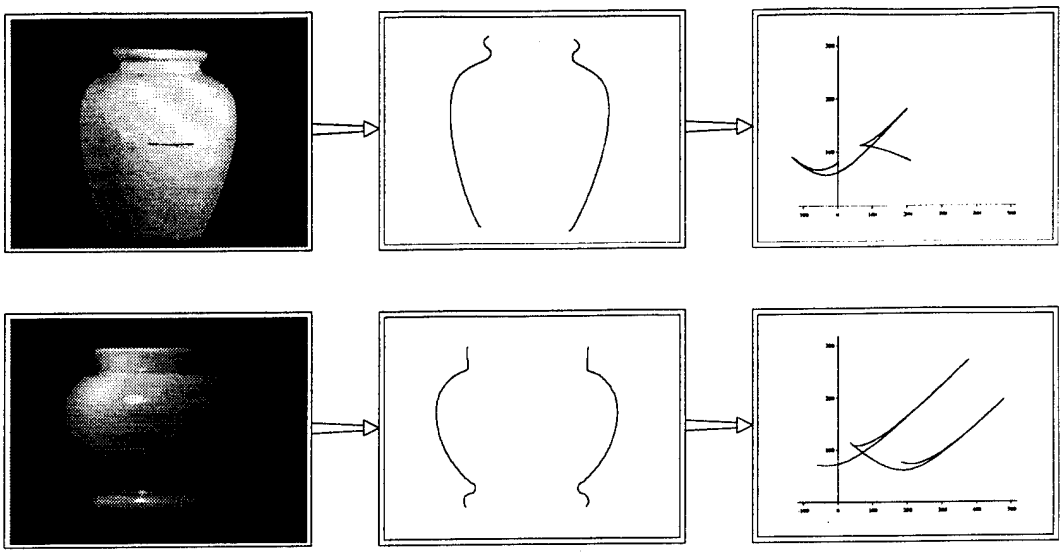


Figure 10: *Graphs of scaling against an affine axis parameter. The representation clearly discriminates between the two vases.*

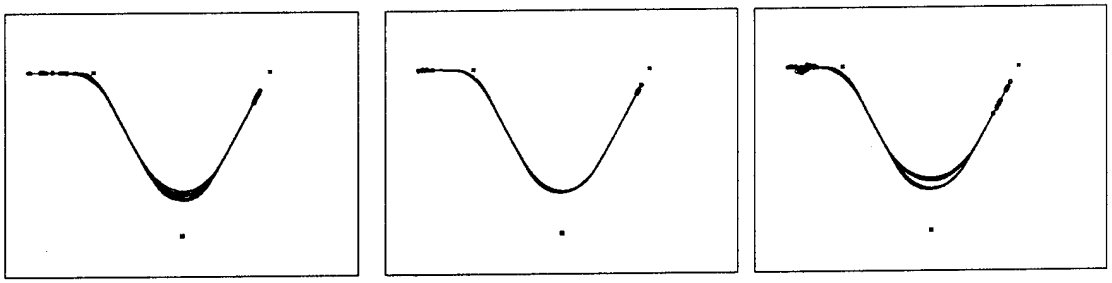


Figure 11: *(a) Affine canonical frame for symmetry sets from six weak perspective images (as in figure 7c) and seven images with significant perspective effects. Note the distortion for the latter curves. (b) Projective canonical frame for the thirteen symmetry sets for the same pipe used in (a). The curves are virtually identical, eliminating the perspective distortion shown in the affine frame. (c) Superimposed symmetry sets of pipes 1 and 4 (pipe 4 is shown in figure 12) in a projective canonical frame. In each case there are twelve curves. This demonstrates discrimination between objects from their canonical frame curves. The difference between the two curve sets is measured using a statistical classifier.*

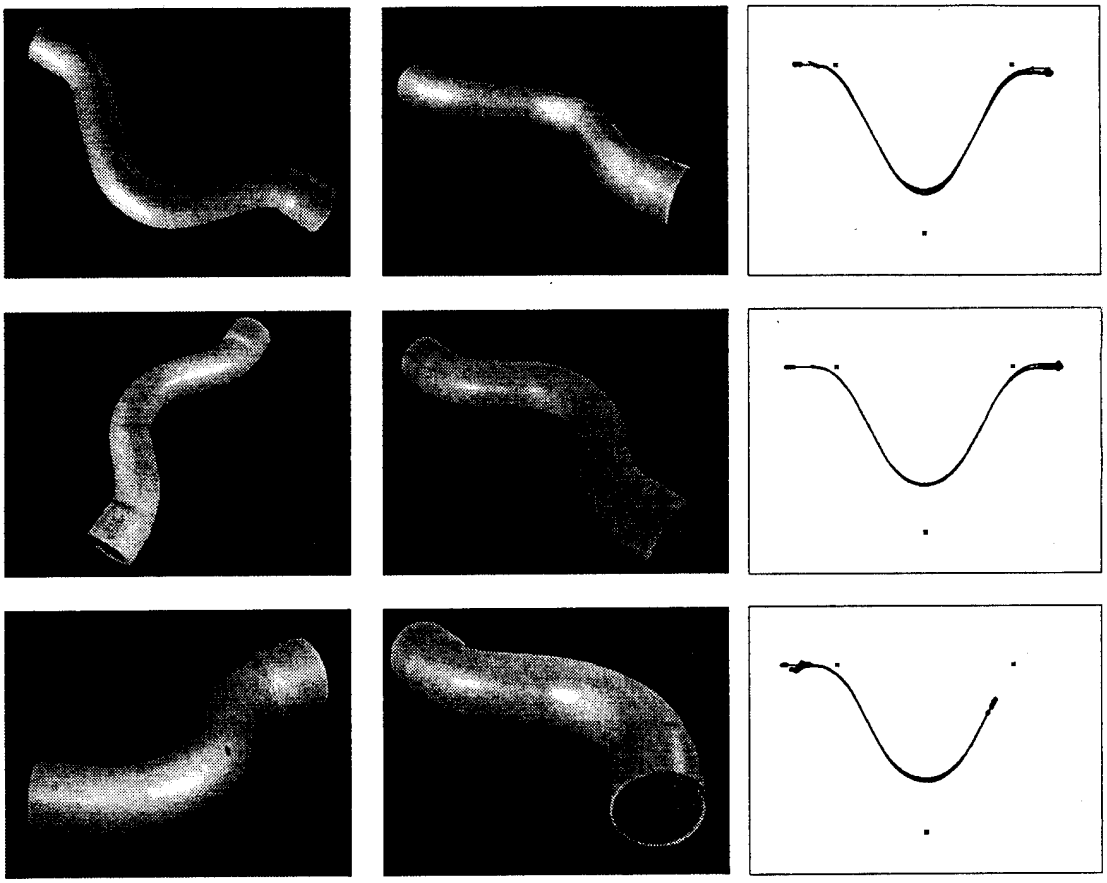


Figure 12: *Example images and projective canonical frames for three canal surfaces, pipes 2, 3 and 4. Note the wide variation in viewing position. From top to bottom, the canonical frames contain symmetry sets generated from six, five and twelve images respectively. At least half of each set show significant perspective distortions—note the variation in pipe width in the middle column due to perspective. The end-diameter of pipe 2 is 15mm—the same diameter as pipe 1; pipes 3 and 4 both have an end-diameter of 22mm. The slight instability present towards the ends of the canonical frame curves are due to errors in the extracted symmetry set which occur where the pipe radius changes.*

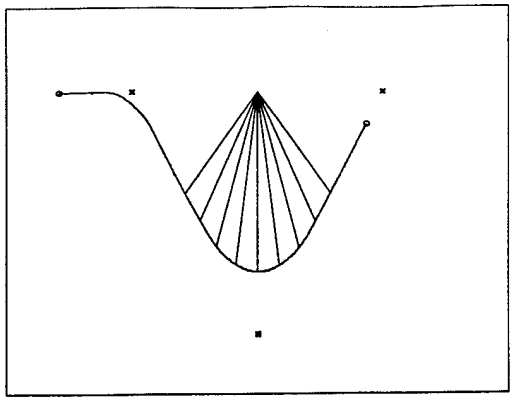


Figure 13: *The construction used to obtain the invariant line lengths* **L** *in the canonical frame. Nine rays are cast from a point mid-way between two of the distinguished points. The value recorded is the length of the ray. The rays have equal angular separation.*

	Invariant 1		*Invariant 2*		*Invariant 3*	
	mean	*s.d.*	*mean*	*s.d.*	*mean*	*s.d.*
pipe 1 (13)	−21.988	0.1800	130.513	0.1671	−22.531	0.1417
pipe 2 (6)	−21.459	0.0998	129.530	0.1514	−22.410	0.1938
pipe 3 (5)	−23.981	0.1383	130.336	0.1564	−22.089	0.1631
pipe 4 (12)	−24.442	0.1879	130.042	0.1773	−22.592	0.1781

Table 1: *Invariants computed using a Fisher linear discriminant based on the canonical frame measures shown in figure 13. The bracketted numbers after each pipe give the number of images contributed for that pipe to the computation of the Fisher Discriminant matrix.*

Stability is poor if an affine frame (three symmetry set lines) is used for perspective images; see figure 11a. However, if a projective frame (three symmetry set lines and the vanishing line) is used, stability is excellent; see figure 11b. This stability of canonical frame curves is an indirect indication of the planarity of the axis curve. Examples of projective frames for other pipes and discrimination between them are given in figures 11c and 12.

5 Viewpoint-Invariant Measurements: Invariants

The previous section described the computation of viewpoint-invariant *curves*. Here, we compute *invariants* from the curves. An invariant is a number whose value is unaffected by viewpoint.

Invariants are obtained from measurements on the canonical frame curve illustrated in figure 13. This construction [24] is similar to the *footprints* of Lamdan, *et al.* [14], although here lengths are measured rather than areas. The vector of invari-

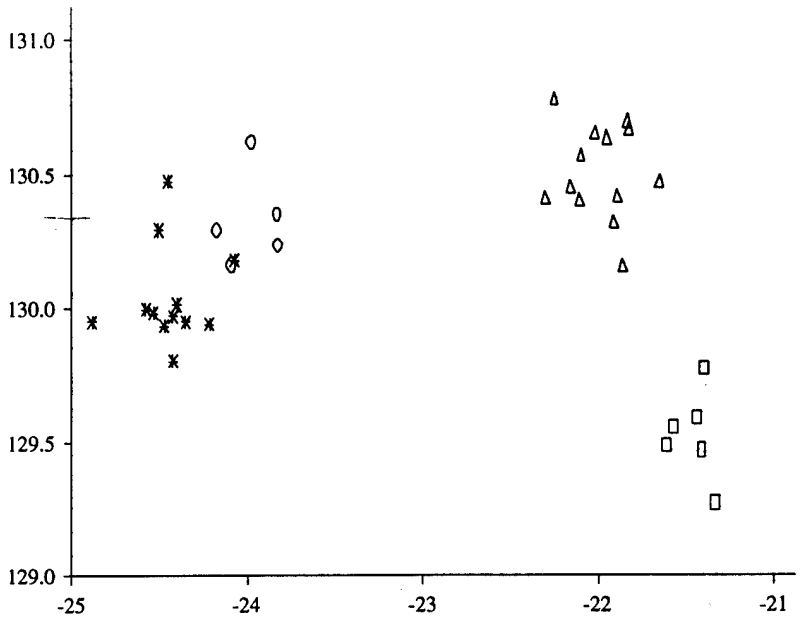


Figure 14: *Scatter plot of the first two invariants computed from the pipe images. Pipes 1–4 are represented by triangles, squares, circles and stars respectively. This demonstrates the sensitivity of the discriminant—the canonical frame curves for pipes 1 & 2 are almost identical.*

ant line lengths $\mathbf{L}$ is not used directly for discrimination. Instead, an *index vector* $\mathbf{M}$ is constructed from $\mathbf{L}$ using a statistical classifier over all extracted canonical frame curves. There are two advantages of this: first, the index is more discriminating than 'raw' lengths; second, the dimension of the index will generally be less than the dimension of $\mathbf{L}$, simplifying its use in a recognition system.

The Fisher linear discriminant [10], which is an optimal linear classifier, is used for the computation of the index. The discriminant minimises the 'intra-class variance' (that is over several examples of the same pipe) and maximises the 'inter-class separation' (that is for different pipes). It does so by transforming to a new, orthogonal basis, $\mathbf{M} = \mathtt{E}\,\mathbf{L}$, where the matrix $\mathtt{E}$ is computed from feature measurements taken over all the canonical frames. The invariants are the components of the vector $\mathbf{M}$.

Values of these invariants and their variances for the four pipes of figures 6 and 12 are given in table 1. The scatter plot of figure 14 demonstrates that the four pipes could almost be distinguished solely using the first two invariants. In practice three invariants reliably (up to two standard deviations) distinguish all the examples of figures 11b and 12.

6 Recognition

Object representation up to a linear transformation is sufficient for recognition. Indeed, a model-based recognition system has been built for planar objects using only projective properties [24]. The same system architecture can be used to build

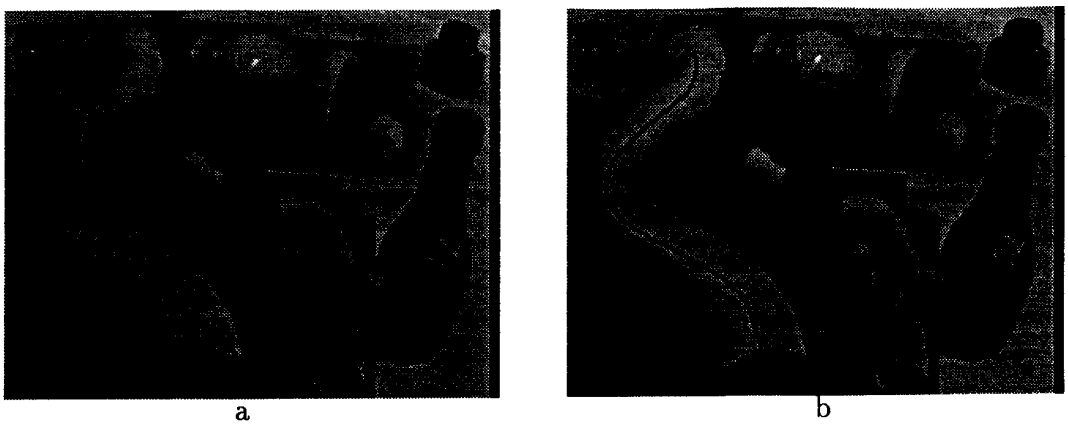


Figure 15: *Viewpoint-invariant recognition. (a) The image contains one pipe in the model library (the large pipe on the left), and two others that are not in the library, as well as other objects. (b) The pipe in the library is correctly recognised and identified as pipe 2. The black curve shows the projected model symmetry set. The other pipes are not recognised. All processing is automatic. Note the significant perspective distortion of the left pipe.*

a recognition system based on the GC invariants.

There are two stages in building a recognition system: first, acquisition, where canonical frame curves, and their invariants, are stored in a model library; second, recognition, where the system identifies which model (if any) is in a perspective image.

Recognition proceeds by: (1) curves are selected as potentially belonging to pipes by locating consistent sets of straight line pairs, other curves are discarded; (2) computing symmetry sets for all selected curve pairs; (3) computing invariants for each symmetry set curve; (4) using the invariants as an *index* to access the model library—if an invariant value corresponds to one of the library values, a recognition hypothesis is generated; (5) verifying the recognition hypothesis by comparing the target symmetry set to a library curve.

A recognition system has been built which can identify a pipe from a perspective image. The image can contain several objects from the library as well as other unmodelled objects ('clutter'), and the viewpoint is unconstrained. At present the model library contains four pipes. Recognition examples are shown in figures 15 and 16.

7 Discussion

We have demonstrated a viewpoint-invariant representation for GCs of a certain class $\mathcal{S}$. The representation is stable over viewpoint, discriminates between objects, and can be reliably extracted from images. Invariants based on the representation have been successfully used as index functions in a model-based recognition system.

Figure 16: *(a) The image contains two pipes in the model library as well as other objects; pipe 1 on the left, pipe 4 on the right. (b) Both pipes are correctly recognised.*

Although theoretically correct only for precise envelopes of spheres, the methods are not too sensitive to deviations from the ideal. For example, the cross-section of pipe 1 is actually elliptical, with an aspect ratio varying between 0.89 and 1.0, rather than uniformly circular. Clearly, because of self-occlusion, the representation is less useful for surfaces of revolution.

We are currently enlarging the model library, and investigating invariants which are less sensitive to missing contours. Typically contour segments are missing due to: feature 'drop out', occlusion from other objects, and self-occlusion. There are a number of alternative methods for extracting affine invariants from the remaining symmetry set curve portions. In particular, semi-differential curve invariants [29] do not require such a rich curve geometry as that required for a canonical frame, since fewer inflections are needed.

Acknowledgements

The authors are very grateful for discussions with Peter Giblin, who made a number of improvements to an earlier draft of this paper, and also to Bill Bruce, Andrew Blake, Roberto Cipolla and Joe Mundy. Financial support was provided by the US Air Force Office of Scientific Research grant no. AFOSR-91-0361, EPSRC, ESPRIT Project 6448 'VIVA', and the University of Oxford.

References

[1] Binford T.O., 'Inferring Surfaces from Images'. *Artificial Intelligence*, vol. 17, pp. 205–244, 1981.

[2] Binford T.O. and Levitt T.S., 'Quasi-Invariants: Theory and Explanation'. *Proc. Darpa IUW*, pp. 819–829, 1993.

[3] Blum H., 'Biological Shape and Visual Science (Part I)'. *J. theor. Biol.*, vol. 38, pp. 205–287, 1973.

[4] Brady M. and Asada H., 'Smoothed Local Symmetries and their Implementation'. *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 36–61, 1984.

[5] Bruce J.W., Giblin P.J. and Gibson C.G., 'Symmetry sets', *Proc. of Royal Soc. of Edinburgh*, 101A, 163-186, 1985.

[6] Bruce J.W. and Giblin P.J. *Curves and Singularities.* Cambridge University Press, Second Edition, 1992.

[7] Burns J.B., Weiss R.S. and Riseman E.M., 'The Non-existence of General-case View-Invariants'. In [18].

[8] Canny J., 'A Computational Approach to Edge Detection'. *IEEE Trans. PAMI*, vol. 8, no. 6, pp. 679–698, 1986.

[9] Clemens D.T. and Jacobs D.W., 'Model Group Indexing for Recognition'. *IEEE Trans. PAMI*, vol. 13, no. 10, pp. 1007–1017, 1991.

[10] Duda R.O. and Hart P.E. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[11] Forsyth D.A., Mundy J.L., Zisserman, A. and Rothwell C.A., 'Recognising rotationally symmetric surfaces from their outlines'. *Proc. ECCV*, LNCS 588, Springer-Verlag, 1992.

[12] Giblin P.J. and Brassett S.A., 'Local Symmetry of Plane Curves'. *American Mathematical Monthly*, vol. 92, no. 10, pp. 689–707, 1985.

[13] Kanade T., 'Recovery of Three Dimensional Shape of an Object from a Single View'. *Artificial Intelligence*, vol. 17, nos. 1–3, pp. 409–460, 1981.

[14] Lamdan Y., Schwartz J.T. and Wolfson H.J., 'Object Recognition by Affine Invariant Matching'. *Proc. CVPR*, pp. 335–344, 1988.

[15] Liu J.S., Mundy J.L., Forsyth D.A., Zisserman A. and Rothwell C.A., 'Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders', *Proc. CVPR*, 1993.

[16] Moses Y. and Ullman S., 'Limitations of Non Model-Based Recognition Systems'. *Proc. ECCV*, LNCS 588, Springer-Verlag, 1992.

[17] Mukherjee D.P., Zisserman A. and Brady J.M., 'Shape from symmetry—detecting and exploiting symmetry in affine images'. To appear, *Proc. Royal Soc.*, 1994.

[18] Mundy J.L. and Zisserman A.P., *Geometric Invariance in Computer Vision.* MIT Press, 1992.

[19] Nalwa V., 'Line drawing interpretation: Bilateral symmetry' *IEEE PAMI*, vol 11, 10, 1117-1120, 1989.

[20] Ponce J., 'Invariant properties of straight homogeneous generalized cylinders'. *IEEE PAMI*, vol. 11, no. 9, pp. 951–965, 1989.

[21] Ponce J., 'On characterizing ribbons and finding skewed symmetries'. *Proc. Robotics and Automation*, vol. 1, 49-54, 1989.

[22] Pratt V., 'Direct Least-Squares Fitting of Algebraic Surfaces'. *Computer Graphics*, vol. 21, no. 4, pp. 145–151, 1987.

[23] Rothwell C.A., Zisserman A., Forsyth D.A. and Mundy J.L., 'Canonical Frames for Planar Object Recognition'. *Proc. ECCV*, LNCS 588, Springer-Verlag, 1992.

[24] Rothwell C.A., Zisserman A., Forsyth D.A. and Mundy J.L., 'Planar Object Recognition using Projective Shape Representation'. To appear, *IJCV*, 1994.

[25] Rycroft J.E. *A Geometrical Investigation into the Projections of Surfaces and Space Curves*, Ph. D. Thesis, University of Liverpool, 1992.

[26] Scott G.L., Turner S. and Zisserman A., 'Using a Mixed Wave/Diffusion Process to Elicit the Symmetry Set'. *Image and Vision Computing*, vol. 7, pp. 63–70, 1989.

[27] Springer C.E., *Geometry and Analysis of Projective Spaces.* Freeman, 1964.

[28] Ulupinar F., 'Perception of 3-D Shape from 2-D Image of Contours'. *PhD thesis, University of Southern California*, 1991.

[29] Van Gool L.J., Moons T., Pauwels E. and Oosterlinck A., 'Semi Differential Invariants'. In [18].

[30] Zerroug M. and Nevatia R., 'Using invariance and quasi-invariance for the segmentation and recovery of curved objects'. In Mundy J.L., Zisserman A., and Forsyth D.A., editors, *Applications of Invariance in Computer Vision*, LNCS 825, Springer-Verlag, 1994.

Planar Object Recognition using Projective Shape Representation

C.A. Rothwell*, A. Zisserman*, D.A. Forsyth* and J.L. Mundy†

(*) Robotics Research Group, Department of Engineering Science, University of Oxford, Parks Rd, Oxford OX1 3PJ, UK.

(*) Department of Computer Science, University of Iowa, Iowa City, IA 52242.

(†) GE Center for Research and Development, 1 River Rd, Schenectady, NY 12345.

August 1993

Abstract

We describe a model based recognition system, called LEWIS, for the identification of planar objects based on a projectively invariant representation of shape. The advantages of this shape description include simple model acquisition (direct from images), no need for camera calibration or object pose computation, and the use of index functions. We describe the feature construction and recognition algorithms in detail and provide an analysis of the combinatorial advantages of using index functions. Index functions are used to select models from a model base and are constructed from projective invariants based on algebraic curves and a canonical projective coordinate frame. Examples are given of object recognition from images of real scenes, with extensive object libraries. Successful recognition is demonstrated despite partial occlusion by unmodelled objects, and realistic lighting conditions.

1 Introduction

1.1 Overview

In the context of this paper, recognition is defined as the problem of assigning the correct label to an object seen in a perspective view. Recognition is considered successful if the 2D geometric configuration of an object in an image can be explained as a perspective projection of a geometric model of the object. In this paper we restrict ourselves to planar objects, although many man-made 3D objects can be decomposed into recognisable planar patches.

A key aspect of the system is the use of the projective transformation group to represent perspective image projections. Most object recognition systems use approximations to perspective, such as affine or orthographic camera models. Such approximations are often valid, but viewing conditions where depth variation of the object is significant compared to viewing distance, or those that consider a wide viewing

angle, require a complete representation of the effects of perspective image formation.

Perspective, or central projection, does not exhibit the full range of geometric transformation possible under the projective model. For example, convexity is preserved under perspective projection (so long as the imaged object does not intersect the image plane), but not under full projective transformation. However, the convenience of homogeneous coordinates, the consequent linearity of projective transformations, and the associated group properties motivate our use of projective geometry throughout. The restrictions associated with perspective are introduced in the recognition process as part of hypothesis confirmation. It should also be noted that the parameters associated with internal camera calibration are implicit in projective projection, so object description and recognition is not dependent on camera geometry.

The recognition system, which is called LEWIS (Library Entry Working through an Indexing Sequence), is designed around the use of invariant indexing functions to represent each object class. An invariant is defined as a function which measures some geometric properties of an object but whose value is independent of projective frame. These indexing functions are computed from the geometric coordinates or coefficients of a small group of image features such as points, lines and conics. The emphasis is on efficiently indexing a large model library, where the index keys are constructed from invariant function values. In practice, the index derived from one view (a model acquisition view) can be used to access the object model in any subsequent view.

Since these indexes can be derived from any viewpoint, it follows that any unoccluded view of the object can serve as a model. We derive the invariant values for the library from a typical view and also include information which is needed for verification, such as the main geometric features of the object and the bounding box of the features. It is beneficial to acquire the model directly from an image since the resulting geometry reflects the actual shape of the object, including rounded corners and other manufacturing artifacts.

Invariant indexing functions are derived according to two approaches. In the first approach, *algebraic invariants* are based on classical results derived from the projective geometry of algebraic curves [Semple52]. The fundamental invariant in projective geometry is the cross ratio, which is defined for four collinear points in terms of ratios of distances between the points. A similar invariant can be defined for four lines concurrent at a single point. More general algebraic invariants can be derived from configurations of conics, points and lines. For example, a cross ratio can be generated from two points and a conic; this arises because the line passing through the two points intersects the conic in two other collinear points. These and other algebraic invariants will be discussed in detail in section 2.2.

The second approach to the construction of invariant indexing functions is the use of projective coordinate frames. In the projective plane, four points, no three of which are collinear, define unique projective coordinates for any other point in the plane. These projective coordinates are invariant to any projective transformation of the plane. We can define a particular frame, usually a fronto-parallel view, which we call the *canonical frame*. Invariant indexes are constructed from a sample of points on the boundary of the object when projected onto the canonical frame. The advantage of the canonical frame construction is that the

object boundary does not have to be an algebraic curve.

These ideas have been incorporated into a complete recognition system over the past four years. The system, LEWIS, has been tested on a large set of images and under varying levels of occlusion and clutter. The major issues which have been examined in the evaluation of LEWIS are:

1. The dependence of recognition complexity on the number of models in the database.
2. The discrimination power of projective invariant descriptions, particularly in the presence of clutter and occlusion.
3. The effect of illumination, object surface properties and feature segmentation on invariant values.
4. The practicality of constructing object models directly from an example object view.

We will explore these issues in later sections, but first it will prove useful to establish the framework for object recognition. In particular, we establish the benefits of a model library accessed by invariant keys.

1.2 Recognition Framework

Recognition consists of two process: the first is the identification of which object is potentially present in the scene; and the second is the establishment of a correspondence between the image and the identified model features. Often, these processes are not distinct, though together they can be partitioned into three stages that should be contained within any recognition system (these are similar to those defined in [Grimson90], p.33):

Grouping: what subset of the data belongs to a single object?

Indexing: which object model projects to this data subset?

Verification: how much image support is there for this correspondence?

Naturally, these stages represent an idealised decomposition; robust recognition generally requires numerous interactions between the stages. However, this structure yields a productive framework for defining and measuring the general characteristics of recognition systems.

The aim of grouping (also called *perceptual organisation* [Lowe85], *selection*, or *figure-ground discrimination*) is to provide an association of features that are likely to have come from a single object in a scene. Image features which are exploited in grouping cover all levels of image segmentation, for instance: edgels; corners; algebraic features such as lines and conics; smooth curves represented as splines; and feature descriptions based on regions, such as texture. These features are typically grouped together using cues such as proximity, parallelism [Binford81, Lowe85] collinearity and approximate continuity in curvature [Cox92, Shaashua88]. In the work reported here, we exploit many of these techniques to generate feature groups from which invariant index functions are constructed.

Indexing addresses the problem of model hypothesis generation. For a small number of models, for example two or three, it is reasonable to try simply to find image feature support for each model. This approach is typical of many existing systems [Ayache86, Ayache87, Grimson90, Huttenlocher88, Lowe87, Murray87, Pollard89]. As the size of the model library increases, this approach becomes computationally too expensive. It is then more effective to choose potential models from the library based on the observed image features. That is, image feature measurements are used to *index* into the model base. The work presented in this paper demonstrates that efficient indexing strategies can be constructed, and that through using them, dramatic improvements in hypothesis generation efficiency can be achieved.

The final stage is verification. Grouping and indexing have hypothesised a match between an object and a small number of image features. This match is used to project the model onto the image. The validity of the model hypothesis and model-to-image feature correspondences is determined by searching for image features that have not been used in the construction of indexes. These are features that, for instance, have been missed by the grouping stage. The more features that can be found which are close to the projected model boundaries, the more likely it is that the initial hypothesis is correct. Once all possible correspondences have been accepted or ruled out, a conclusion as to the identity of an object can be made. Generally, a hypothesis is considered successful if the error between projected model features and corresponding image features is below some threshold and a reasonable fraction of the object outline is covered by image features.

There are three distinct algorithms that have been used to compute correspondence. In the first approach, *interpretation trees* [Ayache87, Brooks83, Ettinger88, Fisher89, Grimson87, Murray87, Pollard89, Reid91], the set of correspondences is grown incrementally according to a branch and bound search algorithm. Features are added according to their consistency with a model hypothesis associated with each node in the graph. Consistency is also a function of the specific set of features defined by the path from the root of the search tree to the current node.

The second approach, *hypothesise and test* [Ayache86, Bolles87, Goad83, Huttenlocher87, Lowe87], generates model hypotheses exhaustively from the library, although properties of small feature groups can be used to suggest initial trial feature correspondences. These hypotheses are tested by establishing model-dependent and priority ordered checklist of other features which must be present to satisfy the hypothesis. A set of *focus features* are defined for each object which are easily extracted and also provide maximum discrimination among object classes.

The third approach, *pose clustering* [Cass92, Stockman87, Thompson87], uses the concept of pose consistency to generate hypotheses. An object is projected onto an image under a single transformation acting on all points of the object. The image projection of an object is composed of a 3D Euclidean transformation (called pose), followed by a perspective mapping. The 3D pose can be computed from various model-to-image feature correspondences and should be the same for all correct correspondences from a single object. The search for correct correspondences is then the problem of finding clusters in pose space.

The recognition system reported here shares many characteristics with these approaches, particularly in the stages of feature grouping and hypothesis testing, but differs considerably in how model hypotheses are generated and feature correspondences are established. Our approach to these stages centres on the use of index functions that we now define more formally.

1.3 Indexing Functions

The concept of the indexing function can be developed formally as follows: the index is considered to be a vector, $\mathbf{M}$, which selects a particular model from the library. Each model consists of the set of significant geometric features of the object boundary as well as ancillary information required for hypothesis confirmation such as, the bounding box of the features, pixel chains from which the boundary features are constructed, and perhaps texture or other details of the object surface properties.

The model index is a function only of a set of *projected* model features, $\mathbf{F}$, that is $\mathbf{M}$ can be computed from any image projection of the model features. The practical consequence is that models can be constructed simply by acquiring one or a few image views of the object in isolation. If $\mathbf{F}_{model}$ is the set of features actually on an object, and T is the transformation from the object in an arbitrary pose onto the camera, then:

$$\mathbf{M}(T(\mathbf{F}_{model})) = \mathbf{M}(\mathbf{F}_{model}).$$

This equation states that the index function is *(scalar) invariant* [Forsyth91] to transformations of the object which result from different viewpoints. In the results reported here, the index functions are invariant to projective transformations of the image plane. Each element of the index vector $\mathbf{M}$ is an invariant measure computed from a group of model features such as conics, lines, points and plane curve segments. Ideally, the index function should uniquely retrieve a model from the library, but in practice it is likely that a small number of models are retrieved with the same index. Even so, the search cost is considerably reduced below that of testing every member of the library.

The concept of an indexing function described above assumes that both the indexes for the model and for the object can be measured perfectly in a scene. In practice, the measurements are imprecise due both to modeling and imaging errors[1]. It is therefore necessary to provide a range of invariant values in the construction of the index function. In LEWIS, the range is established by quantising the index space according to the observed variation in invariant values due to the effects just mentioned. The quantised index value is denoted by $\mathbf{Q}$ and a quantisation is selected so that,

$$\mathbf{Q}(\mathbf{M}(\mathbf{F}_{model}) + \mathbf{E}_{model}) = \mathbf{Q}(\mathbf{M}(\mathbf{F}_{image}) + \mathbf{E}_{image}).$$

[1] This is not just due to random image noise which is often considered to be the sole cause of error, but also due to events in the image which are not modeled or expected. Examples of unmodeled image events are: specularities; surface texture; and impinging objects.

Note that the quantisation function **Q** is the same for both the model and the image. This is a direct result of being able to acquire models from images. The error characteristics $\mathbf{E}_{model}$ and $\mathbf{E}_{image}$ can also be assumed to be the same.

Other recognition systems have also exploited index functions based on invariants. A system using projective invariants is described by Nielsen for identifying and tracking mobile robots [Nielsen88]. Early versions of the system described here are reported by Forsyth, *et al.* [Forsyth91]. Indexing functions based on affine invariants have formed the basis for a number of planar object recognition systems, for instance the series of papers by Kalvin *et al.* [Kalvin86, Schwartz87, Lamdan88], Wayner [Wayner91], Clemens and Jacobs [Clemens91], Huttenlocher [Huttenlocher91], Taubin and Cooper [Taubin91]. Other avenues of invariant research have been covered by Weiss [Weiss88], Stein and Medioni [Stein92], Califano and Mohan [Califano92], Gueziec and Ayache [Gueziec93], and Rigoutsos and Hummel [Rigoutsos91].

It is also possible to gain some of the advantages of indexing without using index functions that are strictly invariant. For example, Jacobs describes an approach to indexing 3D objects using one parameter families of index values in image transform space. One can then select models based on proximity to these index sets [Jacobs92]. Another approach is the use of quasi-invariants [Binford93], where functions are constructed that are not invariant under general perspective viewing, but are reasonably constant over most practical viewing conditions. The quasi-invariants that have been suggested are *invariants* of more restricted transformation groups such as affine and equiform (scaled Euclidean). Affine transformations apply when the depth change along the object plane is small compared to the distance from the center of perspective. The equiform case occurs when the object plane is parallel to the image plane.

1.4 Outline of the Paper

Section 2 introduces the notation used in the rest of the paper, defines the algebraic and canonical frame invariants, and describes the segmentation and grouping procedures used in LEWIS. Section 3 surveys the recognition architecture with results and statistics given for the systems working on real images. Finally, section 4 highlights weaknesses in the current approach and suggests directions for future research.

2 Invariant Indexing Functions

2.1 Notation

When homogeneous coordinates are used points on the plane are represented by a triple $\mathbf{x} = (x_1, x_2, x_3)^T = (\lambda x, \lambda y, \lambda)^T$ where $(x, y)^T$ are the standard Euclidean plane coordinates of the point and λ is an arbitrary (non-zero) projective scale factor. Points in the projective plane are equivalent for all values of λ. Lines are defined by $\mathbf{l} = (a, b, c)^T = (\mu \sin\theta, -\mu \cos\theta, \mu d)^T$, where θ is the orientation of the line with respect to the x axis and d is the perpendicular distance of the line from the origin. μ is the projective scale factor for lines.

The incidence of a point and a line in the projective plane is given by, $ax_1 + bx_2 + cx_3 = 0$, or in vector notation, $\mathbf{l} \cdot \mathbf{x} = 0$.

A conic is the set of points $(x_i, y_i, 1)^T$ that satisfy:

$$ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0. \tag{1}$$

A more convenient representation of a conic uses a planar point $\mathbf{x}$ and a quadratic form $\mathtt{C}$:

$$\mathbf{x}^T \mathtt{C} \mathbf{x} = 0, \tag{2}$$

where:

$$\mathtt{C} = \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix}. \tag{3}$$

From now on, typewriter font denotes matrices, bold letters denote vectors, large letters denote model objects and small letters denote image objects. For example, $\mathtt{C}$ is a model conic, $\mathbf{X}$ a model point, and $\mathtt{c}$ and $\mathbf{x}$ their images in a view where recognition is to be achieved.

2.1.1 Projective Transformations

A projective transformation $\mathtt{T}$ between two planes is represented as a 3×3 matrix acting on homogeneous coordinates of the plane. It is a linear mapping on homogeneous points. A homogeneous representation means that only ratios of matrix elements are significant, and consequently the transformation has 8 degrees of freedom. Under imaging, this transformation models the *composed* effects of 3D rigid rotation and translation of the world plane (camera extrinsic parameters), perspective projection to the image plane, and an affine transformation of the final image which covers the effects of camera intrinsic parameters. The effects of radial distortion due to the camera lens are not modeled.

All of the parameters of these separate transformations cannot be recovered uniquely from a single 3×3 matrix, since there are 6 unknown pose parameters, and 5 unknown internal camera parameters (these are camera centre, focal length, aspect ratio and the angle between the coordinate axes of the image plane). For plane to plane perspective transformations, there are therefore 11 unknowns but only 8 constraints. Fortunately, the invariant description, and model projection used in the recognition system, do not require explicit knowledge of either the pose or the internal camera parameters. We need solve only for the independent parameters of the projective transformation $\mathtt{T}$. Note that projectivities form a group, and so most notably every action has an inverse and the composition of two projectivities is also a projectivity. Consequently, two images from different viewpoints of the same planar object are always related by a projectivity.

The mapping of four points between two planes, of which no three points are collinear, is sufficient to determine the transformation matrix $\mathtt{T}$. Each point provides two linear constraints on the transformation parameters, therefore four independent points provide the required $4 \times 2 = 8$ constraints. Corresponding

points (x_i, y_i) and (X_i, Y_i) are represented by homogeneous 3 vectors $(x_i, y_i, 1)^T$ and $(X_i, Y_i, 1)^T$. The projective transformation $\mathbf{x} = \mathtt{T}\mathbf{X}$, $(|\mathtt{T}| \neq 0)$ is:

$$k_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix},$$

where k_i is an arbitrary non-zero scalar. Note that in using this formulation we are unable deal with plane points lying on the ideal line; this is, however, unimportant as in practice all of the points to be transformed lie within the finite and bounded image plane. For $N \geq 4$ points, singular value decomposition can be used to compute $\mathtt{T}$. The computation can be formulated as minimising $||\mathtt{At}||$ subject to $||\mathtt{t}|| = 1$, where $\mathtt{t}$ is the nine-element vector of transform parameters and $\mathtt{A}$ is a $N \times 9$ element matrix of elements formed from the coordinates of the matched image and model points.

Using similar algorithms, projectivities can be computed between sets of lines, or as shown in [Rothwell94] for different combinations of points, lines and conics. The projective transformation of lines is closely related to that for points. Given the transformation matrix for points, $\mathtt{T}$, lines transform according to

$$\mathbf{l} = \mathtt{T}^{-T}\mathbf{L},$$

where $\mathtt{T}^{-T}$ is the inverse transpose of $\mathtt{T}$. The transformation of conics is as follows: given $\mathtt{C}$ and its respective image conic $\mathtt{c}$, and point transformation matrix $\mathtt{T}$, is constrained by:

$$\mathtt{c} = \kappa \,.\, \mathtt{T}^{-T}\mathtt{C}\,\mathtt{T}^{-1}. \tag{4}$$

2.2 Algebraic Invariants

There are three different algebraic invariants used within the recognition system for coplanar algebraic features: five lines; a pair of conics; and a conic and two lines. Their derivation is given, for example, in [Mundy92]. There are many other possible configurations (with points, cubics, etc.) that could also be used to generate invariants. The particular configurations used in LEWIS have been chosen because the constituent geometric features can be produced directly and accurately from segmentation. In contrast, points are extracted most accurately *indirectly* by intersecting lines.

2.2.1 Five Coplanar Lines

Given five coplanar homogeneous lines $\mathbf{l}_i$, where $i \in \{1, .., 5\}$, two functionally independent projective invariants are defined using determinants

$$I_1 = \frac{|\mathtt{M}_{431}||\mathtt{M}_{521}|}{|\mathtt{M}_{421}||\mathtt{M}_{531}|} \qquad \text{and} \qquad I_2 = \frac{|\mathtt{M}_{421}||\mathtt{M}_{532}|}{|\mathtt{M}_{432}||\mathtt{M}_{521}|}, \tag{5}$$

where $\mathtt{M}_{ijk} = (\mathbf{l}_i, \mathbf{l}_j, \mathbf{l}_k)$.

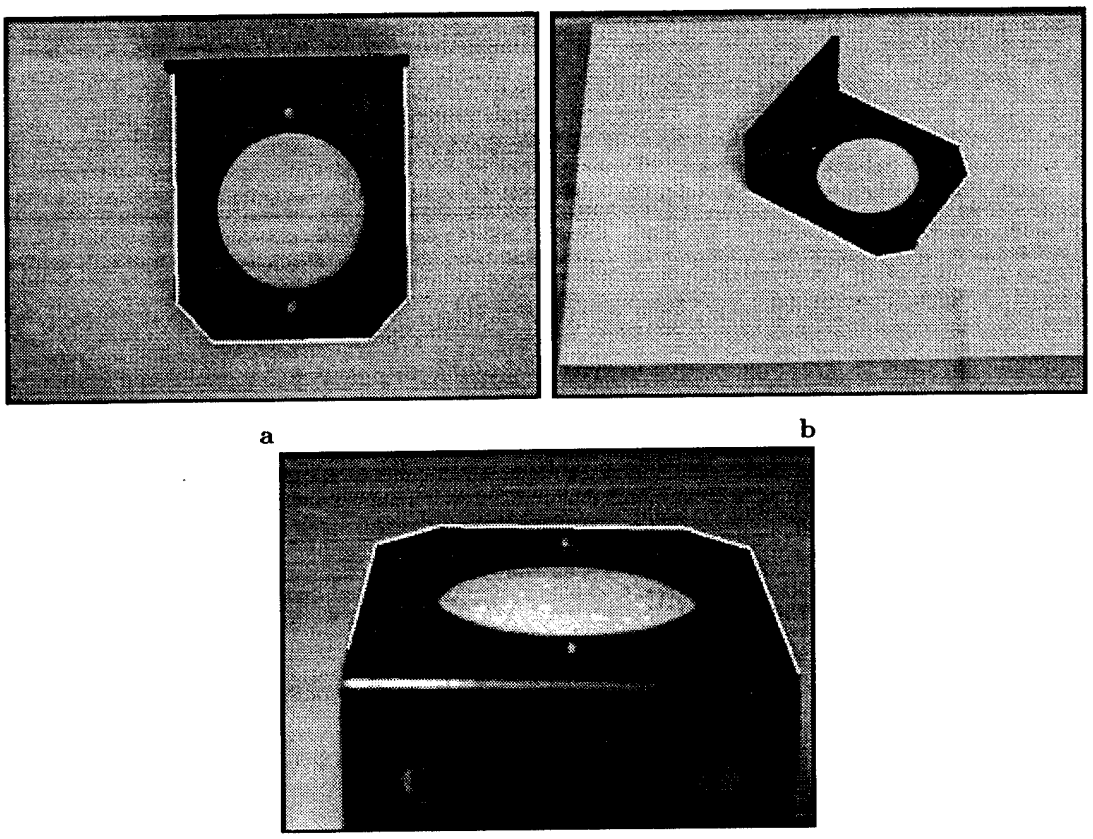


Figure 1: *Examples of similarity, affine and perspective images of a bracket. For each view the lines used to compute the invariants are marked in white. The pair of five-line invariants are computed using the determinant formulae given in this section. The invariant values for the images, and those actually measured on the object are shown in table 1. The fact that they remain essentially invariant demonstrates the stability of the invariants under real imaging conditions. For reference, the values of affine invariants computed from area ratios are also given in the table.*

Table 1: *I_1 and I_2 are five-line invariants computed for the similarity, affine and perspective views of the bracket shown in figure 1. I_{a1} and I_{a2} are affine invariants defined by the ratio of areas of triangles constructed from the points of intersection of the lines. The values of I_1 and I_2 are consistent with those measured on the object and vary only slightly with viewpoint which demonstrates the practicality of deriving invariant measures from image features. Note that particularly for the image with substantial perspective distortion, the affine invariants I_{a1} and I_{a2}, vary considerably more than I_1 and I_2.*

	I_1	I_2	I_{a1}	I_{a2}
object	0.840	1.236	0.739	1.083
similarity	0.842	1.234	0.706	1.051
affine	0.840	1.232	0.743	1.066
perspective	0.843	1.234	0.623	0.949

Table 2: *The conic-pair invariants computed for the similarity, affine and perspective views of the computer tape shown in figure 2. Note the stability of the measured values with respect to change in viewpoint.*

	I_1	I_2
object	3.073	3.082
similarity	3.074	3.082
affine	3.072	3.080
perspective	3.070	3.078

A major problem with the determinant formulae given in equation 5 is that the invariants can become undefined for certain geometric configurations. The determinant, $|\mathtt{M}_{ijk}|$ vanishes when the lines $\mathbf{l}_i$, $\mathbf{l}_j$ and $\mathbf{l}_k$ are concurrent. In LEWIS, grouping is used to eliminate configurations where both invariants are undefined so that one of the values of I_1 and I_2 can always be used. The grouping algorithm, described later, ensures that only the lines $\mathbf{x}_i, i \in \{1,3,5\}$ are allowed to be concurrent. Since there is no determinant of $\mathtt{M}_{135}$ in I_2, it will always be well formed, though I_1 will sometimes fail.

Examples of the invariants computed for real image distortions are demonstrated in figure 1, and the invariant values given in table 1. The fact that the values remain constant over a change in viewpoint demonstrates the stability of the invariants under image noise.

2.2.2 Two Coplanar Conics

A pair of conics $\mathtt{C}_i, i \in \{1,2\}$ has two independent projective invariants. These can be expressed in terms of ratios of eigenvalues [Quan91], or equivalently

$$I_1 = \frac{Trace[\mathtt{C}_1^{-1}\mathtt{C}_2].|\mathtt{C}_1|^{1/3}}{|\mathtt{C}_2|^{1/3}} \qquad \text{and} \qquad I_2 = \frac{Trace[\mathtt{C}_2^{-1}\mathtt{C}_1].|\mathtt{C}_2|^{1/3}}{|\mathtt{C}_1|^{1/3}}.$$

If the conics are *normalised* so that $|\mathtt{C}_i| = 1$ the invariants take on the simpler form of:

$$I_1 = Trace[\mathtt{C}_1^{-1}\mathtt{C}_2] \qquad \text{and} \qquad I_2 = Trace[\mathtt{C}_2^{-1}\mathtt{C}_1].$$

These invariants have been tested extensively during the development of the system reported in this paper, and have been found to have good noise characteristics. A simple example showing the measured invariants for similarity, affine and perspective views of the computer tape shown in figure 2 are given in table 2. The small deviation of the invariants demonstrates their stability, more complete results are given in [Forsyth91].

2.2.3 A Conic and Two Lines

For a conic $\mathtt{C}$ and two lines $\mathbf{l}_i, i \in \{1,2\}$, a single invariant can be computed:

$$I = \frac{(\mathbf{l}_1^T\mathtt{C}^{-1}\mathbf{l}_2)^2}{(\mathbf{l}_1^T\mathtt{C}^{-1}\mathbf{l}_1)(\mathbf{l}_2^T\mathtt{C}^{-1}\mathbf{l}_2)}.$$

The invariant computed for the similarity, affine and perspective image sequence of the bracket is shown in figure 3. The corresponding invariant values in table 3. Again the stability of the invariant form is demonstrated over a large range of viewpoints.

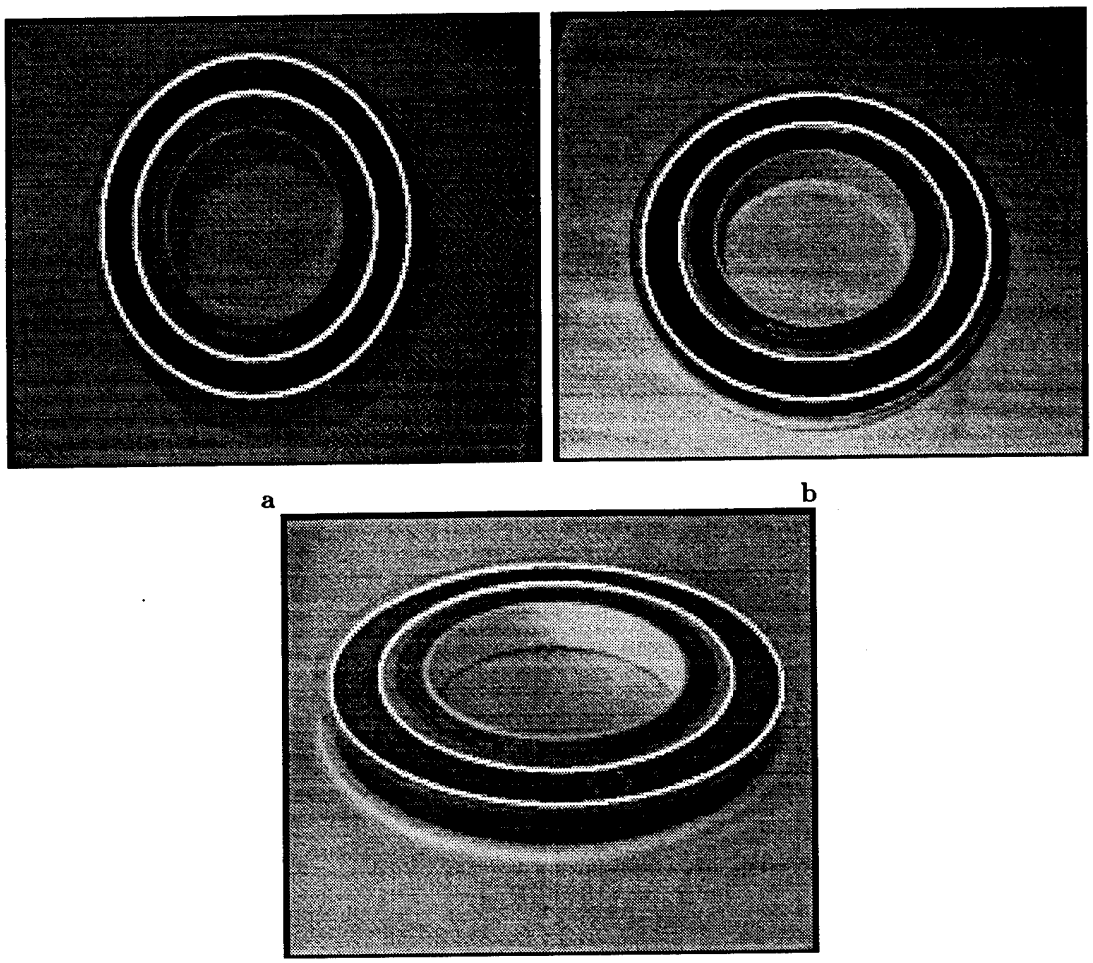


Figure 2: *Similarity, affine and perspective images of a computer tape with the conics used to compute the invariants marked in white. The invariant values are given in table 2.*

Table 3: *The conic and line-pair invariants computed for the similarity, affine and perspective views of the bracket shown in figure 3. Note the stability of the measured values with respect to change in viewpoint. For comparison, an affine invariant is also tabulated. In this case, I_a is defined by the ratio of the areas of a triangle and of the conic itself. One vertex of the triangle is located at the intersection point of the two lines; the other two vertices are defined by the points of tangency to the conic of the pair of lines through the first point that touch the conic. Note that I_a is significantly less stable than I.*

	I	I_a
object	1.33	0.398
similarity	1.33	0.389
affine	1.31	0.403
perspective	1.28	0.437

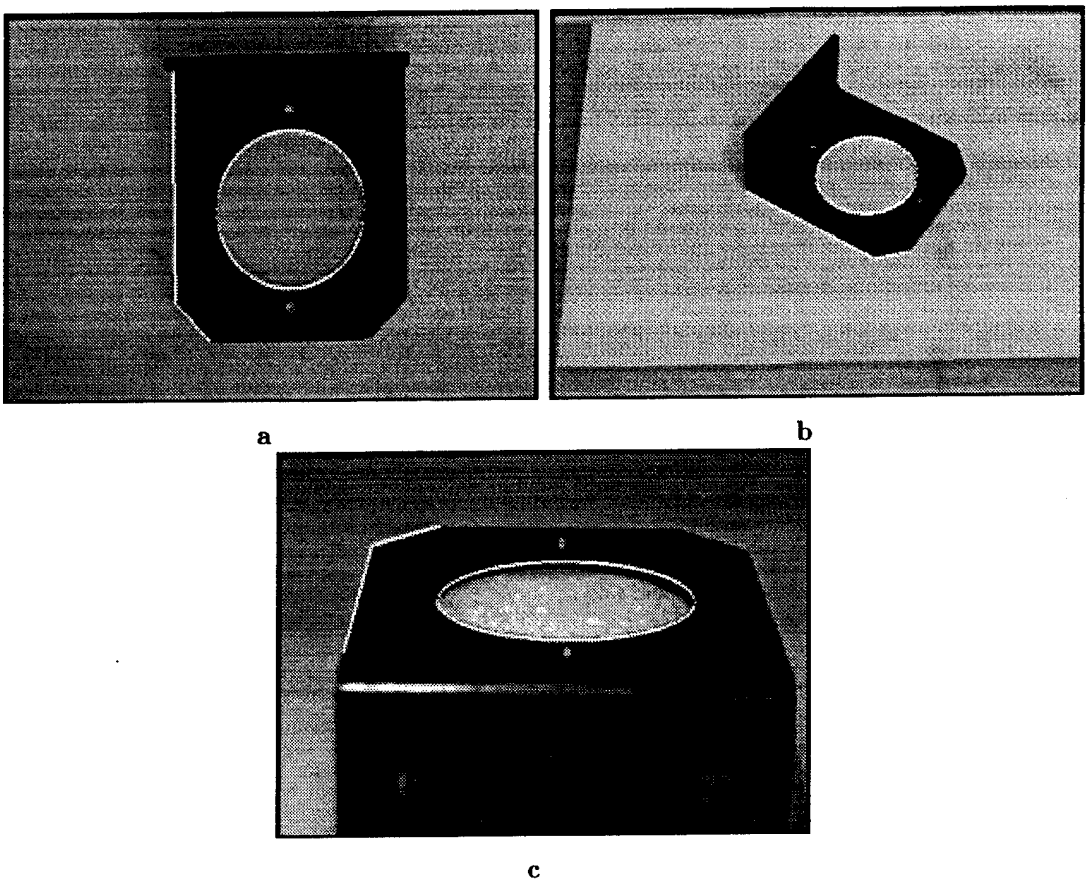


Figure 3: *Similarity, affine and perspective images of a bracket with the conic and two lines used to compute the invariants marked in white. The invariant values are given in table 3.*

We have found in practice that the conic and line pair invariant is not stable enough alone to provide sufficient discrimination for the class of objects used in our experiments. Three independent invariants can be formed from three lines and a conic, using the lines two at a time. The combined index provides better discrimination as explained in section 3.5.1.

2.3 Canonical Frame Invariants

A canonical frame construction can be used to form an invariant *signature* for smooth planar curves. The rest of this section describes the construction of the signature for a non-convex class of plane curves; the work is a projective extension of that of Lamdan, *et al.* [Lamdan88].

First, we illustrate the concept of a canonical frame with a set of of five coplanar points, four used as a projective basis and the fifth to generate invariants. We then show how four *distinguished points* can be defined on a concavity in a plane curve. The rest of the curve can then considered as a set of individual points whose coordinates with respect to the projective basis define the signature.

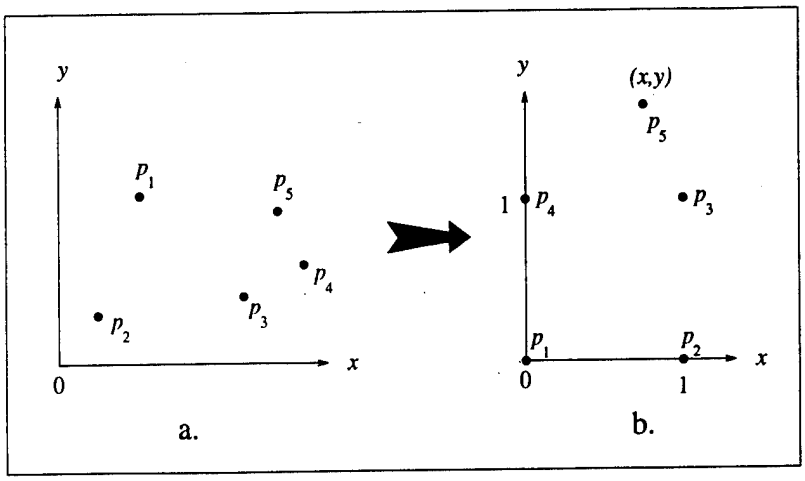


Figure 4: *One way of measuring the invariants of five coplanar points in a image (a) is to compute the projective transformation of four of the points* $\mathbf{p}_i, i \in \{1, \ldots, 4\}$ *to reference points in the canonical frame (b). In this case the projection is to the corners of the unit square. Once this map is known* $\mathbf{p}_5$ *can also be transformed to the new frame and its coordinates* (x, y) *used as invariants.*

2.3.1 Mapping Five Points to the Canonical Frame

As four points define a projective mapping between two frames, the first four points of a set of five can be used to define the map between the image frame and a standard measurement or *canonical frame*. The fifth point can then be mapped to the new frame in which its coordinates are projectively invariant. To ensure that the coordinates really are invariant, the first four points must always be mapped to a standard set of four reference points in the canonical frame. The choice of these points is arbitrary: the corners of the unit square may be used (as in figure 4), or some other frame chosen according to noise performance.

2.3.2 Mapping a Plane Curve to the Canonical Frame

The aim is to find four distinguished points (or lines) on a curve, and use these to define the projectivity $\mathtt{T}$ that can be used to take the whole curve to the canonical frame. The method is shown in figure 5: for the given concavity, the location of the points of bitangency is determined as described in section 2.4.3. These are (A) and (D), and they segment the curve of interest from the rest of the edge chain. This curve segment is known as an $\mathcal{M}$ *curve*. The *cast tangents* are then determined, these are lines tangent to the $\mathcal{M}$ curve that pass through the bitangency points. The points of cast tangency are (B) and (C). The projection of the $\mathcal{M}$ curve into the frame using $\mathtt{T}$ is the *curve signature*; it is a projective representation of the original object curve.

2.3.3 Discrimination

Examples of the canonical frame construction for single views of three different objects are given in figure 6. A single $\mathcal{M}$ curve for each spanner and the pair of scissors is marked in (a), (b) and (c), and these are projected into the same canonical frame in (d). All three canonical curves are different and so the

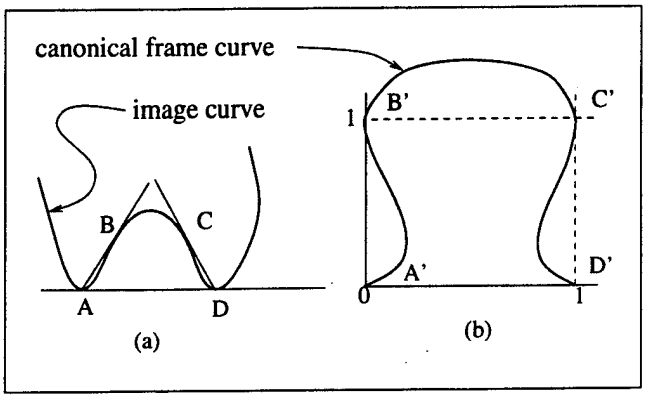


Figure 5: *(a) Construction of the four points necessary to define the canonical frame for a concavity. The first two points, (A) and (D), are points of bitangency that mark the entrance to the concavity. Two further distinguished points, (B) and (C), are obtained from rays cast from the bitangent contact points and tangent to the curve segment within the concavity. These four points are used to map the curve to the canonical frame. (b) the curve in the canonical frame. A projection is constructed that transforms the four points in (a) to the corner of the unit square. The same projection transforms the curve into this frame.*

construction provides discrimination (although the spanner curves extracted from (a) and (b) are reasonably similar, they are sufficiently different for recognition purposes).

2.3.4 Semi-Local Description

Non-global descriptions must be used if objects are to be recognised under occlusion; the canonical frame construction provides a *semi-local* object description. Furthermore, for genuine tolerance to occlusion, there must be a number of different descriptors on each object so that there is not an excessive requirement for any single object region to be visible. This is called *redundancy*. Single objects frequently possess large numbers of bitangents (see figure 13); this provides a high degree of redundancy as each bitangent can be used to derive a canonical frame. However, such a high degree of redundancy is not required for recognition, and only a few bitangents are actually used for shape description. For the spanner in figure 6a, four suitable bitangents exist and bound $\mathcal{M}$ curves. The four $\mathcal{M}$ curves are shown in figure 7.

2.3.5 Stability

The final criteria discussed in this section is stability: if the construction is to be useful, similar frame curves must be obtained from different views of the same object curve. Even if the curves are not identical, they should be sufficiently similar so that discrimination between objects is possible. This is the case for the canonical frame construction. Three very different views of a spanner are given in figure 8 (they vary by a full perspective distortion, and not just an affine one). The same $\mathcal{M}$ curve is marked in each image, and these are mapped to the canonical frame in (d). As can be seen, the construction is stable even over a wide change in viewpoint.

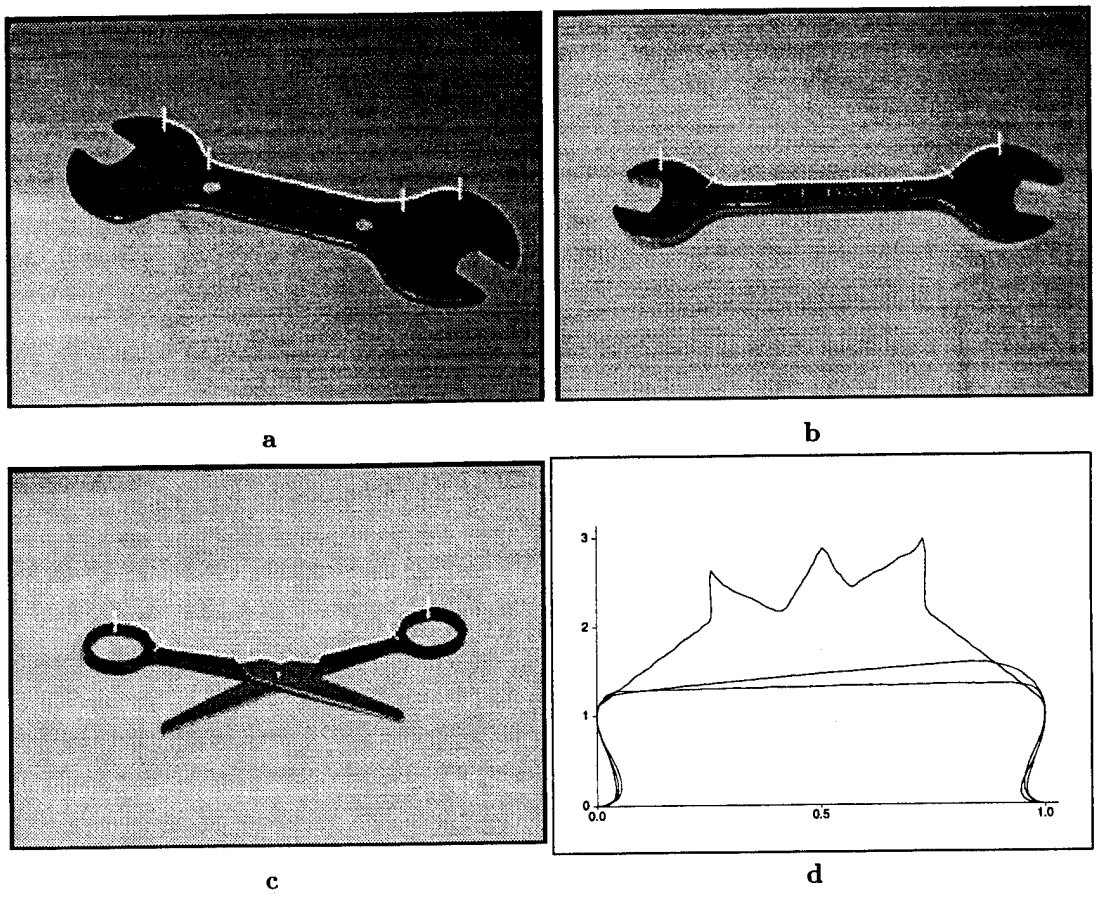


Figure 6: *In (a)-(c) a single $\mathcal{M}$ curve and the four distinguished points are marked on each object. The three curves are projected to the canonical frame and superimposed in (d). The scissor $\mathcal{M}$ curve is obviously very different from each spanner, but in fact the two spanner curves are sufficiently different for recognition purposes.*

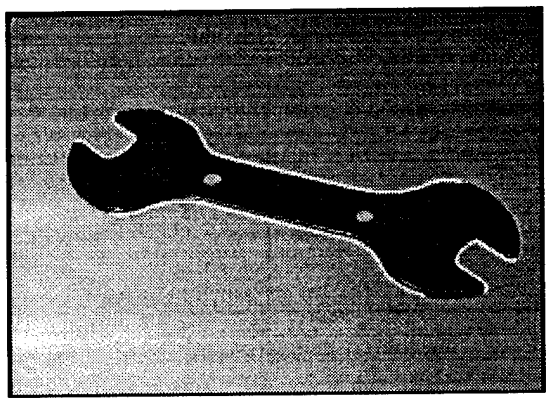


Figure 7: *Even for a simple object such as a spanner there is a sufficient degree of redundancy when the canonical frame construction is used. Here, four useful $\mathcal{M}$ curves are shown that essentially cover the entire perimeter of the object, and yet each one is potentially a sufficient recognition cue on its own.*

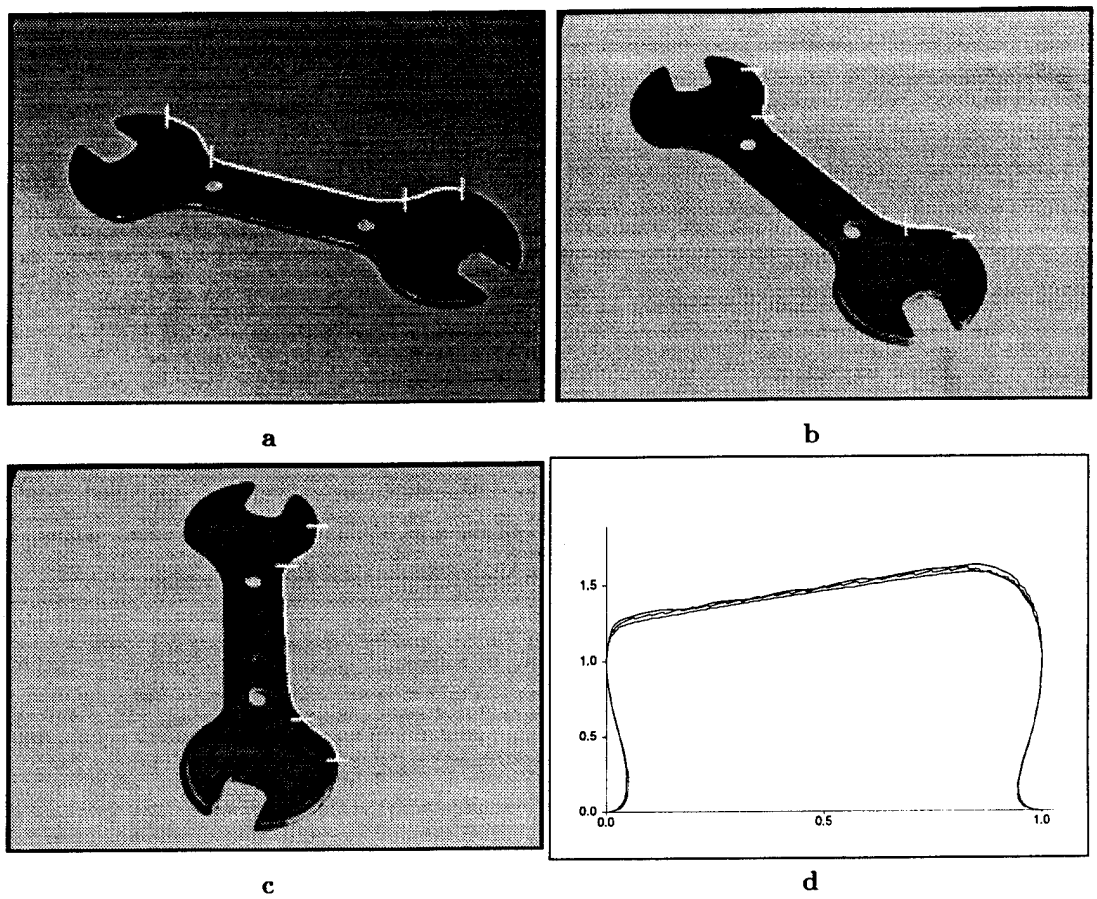


Figure 8: *(a)–(c) three views of a spanner with the extracted $\mathcal{M}$ curves and distinguished points marked. Note the very different appearance due to perspective effects. (d) shows the canonical frame curves for the three different views. The curves are almost identical demonstrating the stability of the method. Of course, the same curve would result from a projective transformation between the object and canonical frame.*

2.3.6 Index Functions and Discrimination

The canonical frame curves are essentially projectively invariant *templates* for the shapes, and so one may attempt $\mathcal{M}$ curve recognition using traditional curve correlation matching techniques with model curves. Such techniques would lead to a linear search of the model library, so instead, an index is constructed from the signature. The goal is to compute a function of the signature that uniquely identifies the $\mathcal{M}$ curve. The current solution is to use a few points along the signature to construct the index. This data is adequate to distinguish the spanners and brackets used in our experiments. The complete signature is retained as part of the model description and used during verification as a more complete representation of the object shape.

The invariant indexes used are constructed using the geometry of figure 9. This construction is similar to the technique of *footprints* [Lamdan88], though points are used rather than areas. The drawback of this method for measuring invariants is the ambiguity occurring when a ray crosses the curve more than once. However, such multiple crossing did not occur for the model base used in our experiments. The vector of invariant line lengths $\mathbf{l}$ is not used directly as an index. Instead, an index vector $\mathbf{M}$ is constructed from $\mathbf{l}$

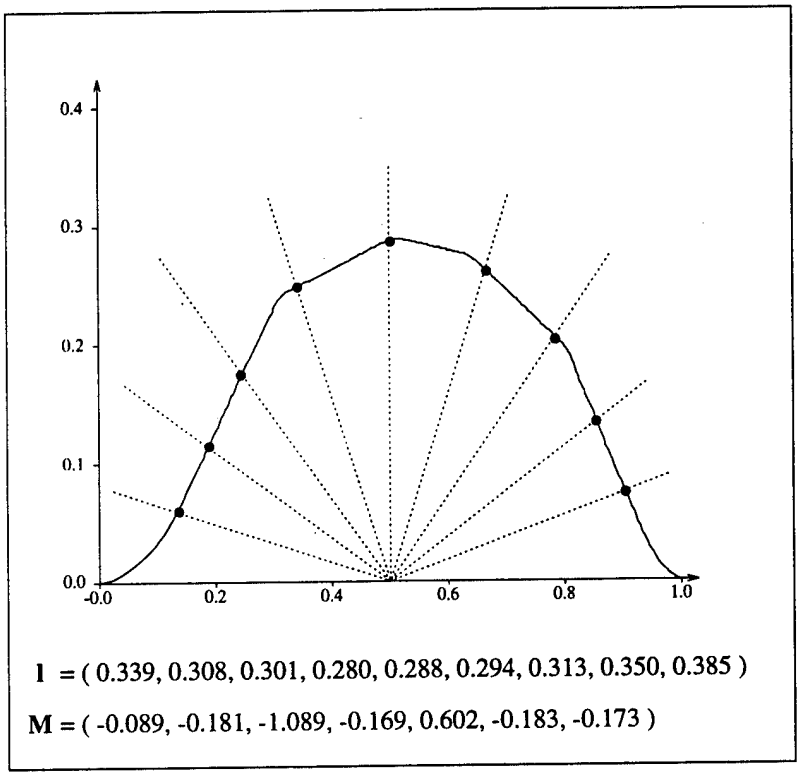


Figure 9: *A set of n equally spaced rays are drawn from the point* $(\frac{1}{2}, 0)$ *so that they intersect the curve signature. The aim is to construct an n-dimensional length vector* $\mathbf{l} = (l_1, \ldots, l_n)^T$, *where* l_i *is the distance from the intersection point of the* i^{th} *ray to the point* $(\frac{1}{2}, 0)$. *This distance is projectively invariant. Here* $n = 9$. *The invariant index* $\mathbf{M}$ *is related to* $\mathbf{l}$ *by* $\mathbf{M} = \mathbf{E}\,\mathbf{l}$, *where* $\mathbf{E}$ *is provided by a linear classifier. See text for details.*

using a statistical classifier over all curves in the model base. There are two advantages of this: first, the index is more discriminating than the "raw" lengths; second, the dimension of the index can be reduced and so the computation of an efficient hashing function is simplified. The Fisher linear discriminant [Duda73], which is an optimal linear classifier, is used for the computation of the index. The discriminant encodes information by minimising the intra-class variance (that is over several examples of the same curve) and maximising the inter-class separation. It does so by transforming the data to a new (orthogonal) basis, $\mathbf{M} = \mathbf{E}\,\mathbf{l}$, such that feature measurement variance is maximised under projection onto some of the basis directions, and minimised onto others.

Each basis coordinate is ranked by how much discrimination it yields. Then, enough of the highest ranked coordinates are chosen to provide the desired separation between the classes. It was found that taking seven elements of the Fisher discriminant basis are sufficient to define and discriminate a projectively invariant description for each curve class. An example of the Fisher basis is shown in figure 10. A benefit of using the classifier for model learning is that an analytic understanding of the statistical characteristics of the invariant measures is not required. Instead, a number of examples of a single class is built up over a number of images, and the classifier adjusts its action to account for the variation within each class.

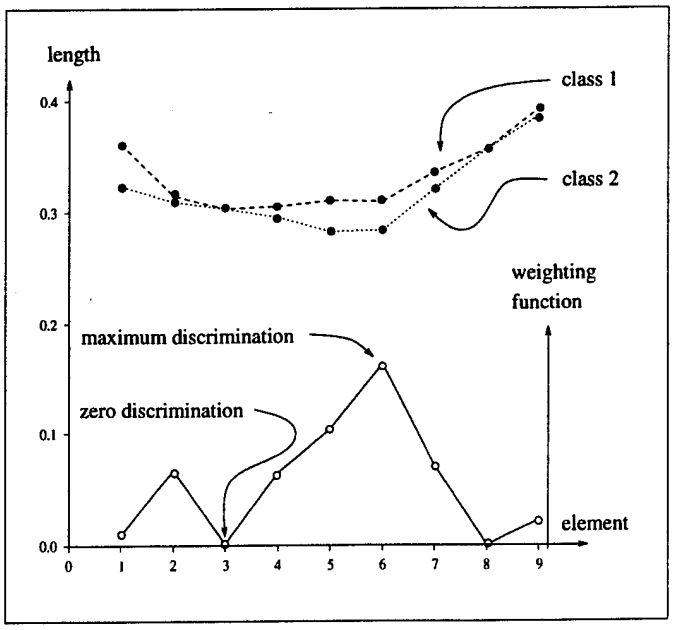


Figure 10: *Example of the Fisher Linear discriminant. The discriminant is trained here only three classes, two of which are shown (black-circles-dashed-line and grey-circles-dotted-line). In each case the curve is represented as a vector of canonical frame ray lengths, each component corresponding to a different angle (see figure 9). For each class a number of vectors, measured for the same curve with varying viewpoint, are included in order to model the intra-class variation. The first eigenvector weighting function produced by the discriminant is shown (white-circles-solid-line). The first invariant is determined as a scalar product of the eigenvector and the (mean) vector for each class. Clearly, this invariant will show good discrimination between the two classes shown.*

2.4 Segmentation and Grouping

LEWIS requires the segmentation of two different categories of features from image data: *lines* and *conics* for the algebraic invariants; and $\mathcal{M}$ *curves* (images curves terminated with common tangents) for the canonical frame construction. The features are grouped depending on the type of invariant that they form.

The first step in the feature detection process is edge detection. We have used an implementation of the Canny edge filter [Canny86]. The next process is segmentation, this can be broken down into three phases:

1. The extraction of discrete edgel chains from the image.
2. The location of breaks between features, and more generally the boundaries between each feature and other data.
3. The accurate representation of image features.

The first step is common for both the algebraic and smooth curve invariants. Single edge curves are extracted from the edge image using a sequential edgel chain linking. The Canny algorithm produces edges with sub-pixel accuracy. This edgel position accuracy yields invariant values with smaller variances (about 10% better) than those computed from integer pixel locations. The reason that using more precise edgel locations does

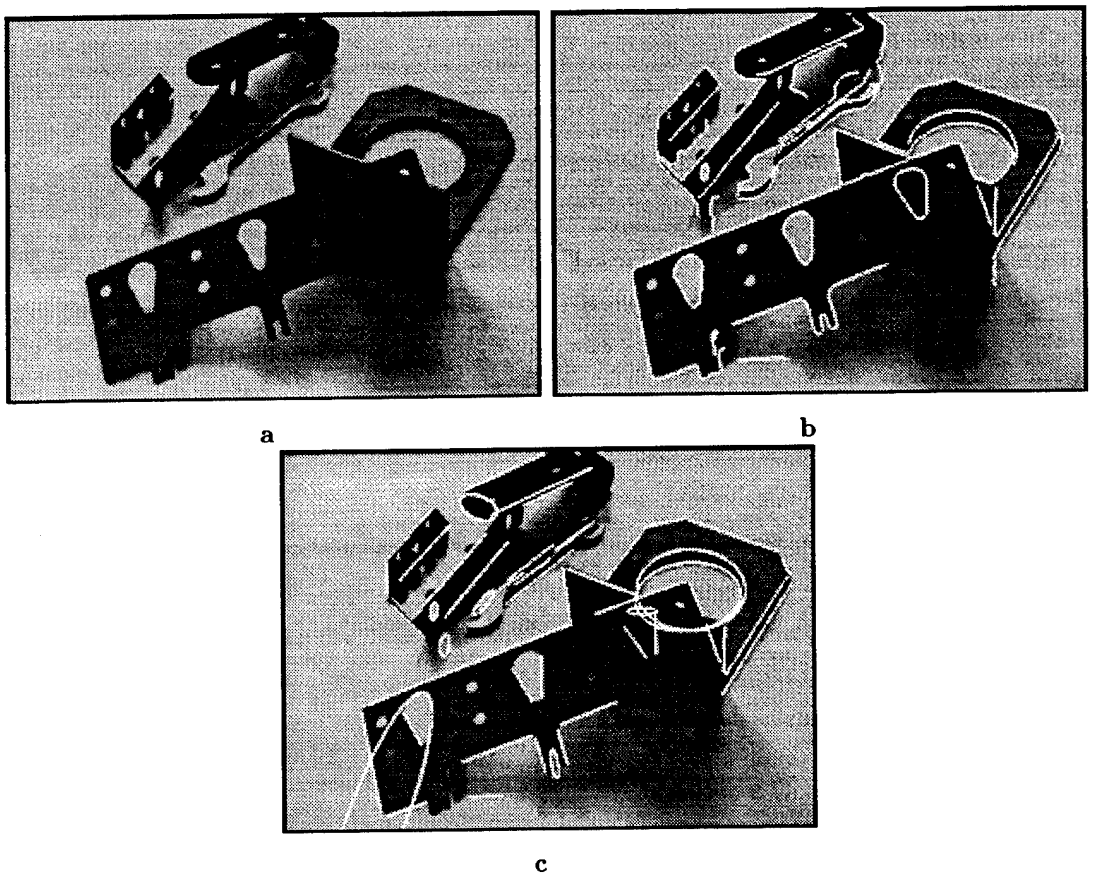


Figure 11: *(b) extracted edge data (Canny) from (a); note that the edge detector fails to locate edges near shadows and that objects (such as the bracket on the right hand side of the image) have a finite thickness and so two edges are reported. The fitted conics and lines are shown in (c) where there is generally accurate location of both tangency and curvature discontinuities.*

not produce such a dramatic improvement in the quality of the measured invariants is that the representation process (principally fitting) is able to smooth out quantisation errors present in the integer edgel locations.

Even with *hysteresis*, single pixel breaks can occur in the edge chains. Such events are accommodated by directional look ahead in a sequential scan of the edge chain. As the quality of the edge data in the images of interest is generally quite good, single pixel look-ahead works well for the objects and illumination conditions used in our experiments. The details of the later stages of the segmentation process depend on the type of invariant that is to be formed, and the different techniques are discussed below.

2.4.1 Algebraic Features

Lines and conics are fitted to extracted edge chains using efficient incremental routines based on orthogonal regression for lines and an improved version of the Bookstein algorithm for conic fitting [Bookstein79]. Full details of the algorithms are given in [Rothwell94]. An example segmentation is shown in figure 11 where it is seen that a reasonably complete description is obtained of the object boundaries.

2.4.2 Grouping

Exploiting structure in the scene for grouping allows invariant indexing to have a low complexity with respect to the number of image features. The approach used makes use of the connectivity provided by the edge chains, this implicitly encodes proximity.

For algebraic invariants, connectivity provides an association and also an ordering on the lines: invariants are formed from sets of consecutive lines within single edge chains at a cost that is linear in the number of lines in the scene, $O(l)$. This type of grouping was also exploited by Huttenlocher who also achieved linear grouping cost [Huttenlocher88].

The use of algebraic curve features rather than isolated points and lines also reduces the combinatorial cost of grouping. In the case of the invariant formed by a conic and three lines the cost of grouping is $O(cl^3)$, where c is the number of conics and l the number of lines. This is for a case in which no image structure is assumed, if connectivity is reliable, the cost reduces to $O(cl)$. The grouping cost for the joint conic invariants is only $O(c^2)$. For the images under consideration l is in the order of a hundred, and c a few tens.

2.4.3 The Canonical Frame

The canonical frame construction requires the accurate location of distinguished points. Stable point constructions are achieved using curve bitangents and points defined by *cast tangents*. In order to form a projective coordinate frame, the canonical frame, four such distinguished points must be found. It is desirable to achieve a canonical projection of the object boundary curve which is minimally distorted and has a roughly uniform variance distribution due to image segmentation effects. In practice, this is achieved by placing the points in the canonical frame in positions that correspond to a fronto-parallel view of the object [Rothwell94] (yielding an equiform distortion of the object).

Bitangent Location

Image bitangents are located using the following four stage algorithm:

- Eliminate points that lie on approximately straight portions of curve. These cannot correspond to actual points of bitangency and so should be ignored.
- Find points on the same edge curve that have approximately common tangents.
- Check that such pairs of points do in fact correspond to bitangents.
- Improve the localisation of the bitangent points using quadratic interpolation.

Straight portions of curve are found by fitting a straight line to short segments of the curve using orthogonal regression and testing the value of the fitting residual. Approximately straight portions will have a low residual. The next step is to map the curve into its tangent dual space and look for self-intersections of the

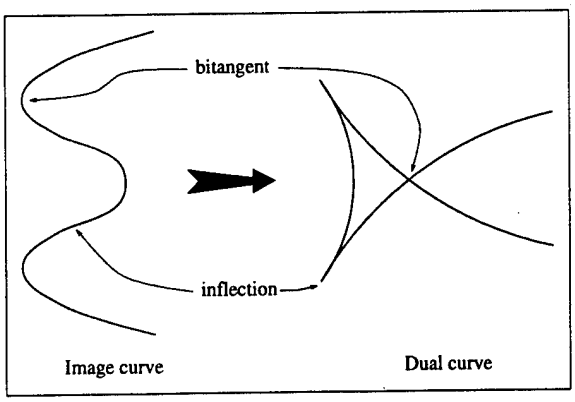


Figure 12: *For continuous curves bitangents in the image correspond to self-intersections in the tangent dual space. Likewise inflections correspond to cusps.*

dual curve. Bitangents, where a line is tangent to the curve at two points, correspond to self intersections in dual space as shown in figure 12. The mapping of a boundary curve into tangent dual space is based on the parameters of a running line fit to the curve. The fitted line is locally tangent at each point along the curve. The representation of the dual space for a curve is essentially the same as a Hough space for lines and is is parameterised by the slope, θ, of the local tangent, and the perpendicular distance of the tangent to the centre of the image.

The dual space is quantised into discrete cells of angle and distance. Since the image curve is discrete, at points of high curvature the difference in tangent direction can vary significantly between adjacent points. This quantisation problem is overcome by linearly interpolating between consecutive points in dual space.

Self intersections, and hence bitangents, are found using a voting scheme in the tangent parameter space. Two image points voting in the same quantised cell represent a self-intersection. Due to small curve fluctuations, joint cell occupancy does not always correspond to actual bitangents. False bitangents are detected by examining regions of the image curve in the proximity of bitangent points. Dual space provides the location of the bitangencies up to discrete pixel coordinates. Significant improvement in accuracy can be obtained by interpolating the bitangent locations between the actual measured edgel locations. A local quadratic fit determines the location of the bitangent points to sub-pixel accuracy. This is done by rotating the image data so that the initial estimate of the bitangent line is horizontal, and fitting (by regression) quadratics of the form $y = ax^2 + bx + c$ to data sets either side of the two bitangent points. The cost used is the error in the y direction. The interpolated bitangent is the line simultaneously tangent to both parabolas. In the implementation the number of points used for each quadratic fit is 13 (6 either side of the hypothesised bitangent point). The data sets are centrally weighted using a Gaussian. The weighting was set empirically by observing how the quality of canonical frame construction changed as the number of points was altered.

The bitangent detection scheme finds many bitangents along single image curves. This is demonstrated in figure 13. Due to excessive redundancy in the shape representation many of the bitangents can be eliminated from consideration, preferably those that are not stable:

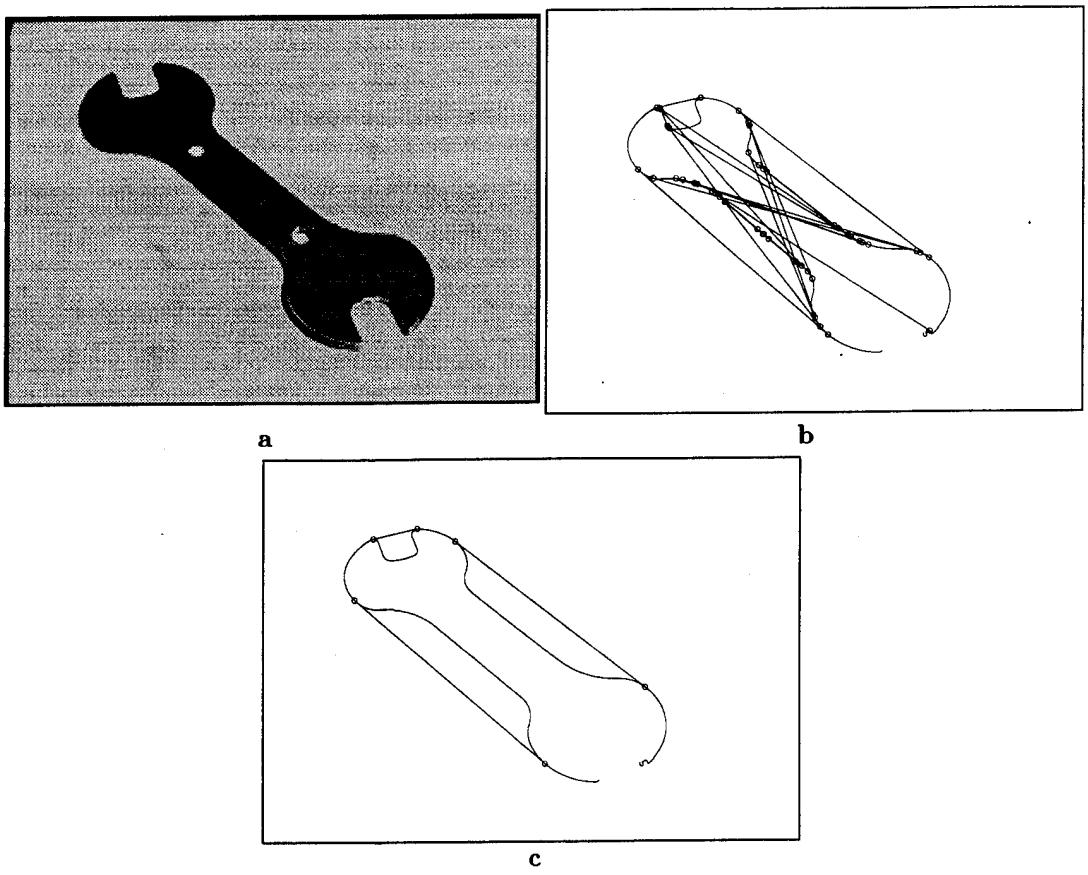


Figure 13: *In (b) it is shown that there are a large number of bitangents that can be found even for a simple object such as the spanner in (a). Each one enables the construction of a canonical frame curve, though only curves that do not cross their own bitangents are used. This reduces the level of redundancy. Bitangents that will not produce a stable construction are also deleted; this leaves the three bitangents shown in (c).*

- Eliminate any bitangents that have their endpoints too close together.
- Remove bitangents whose associated $\mathcal{M}$ curves are not very deep (only a few pixels).
- Do not use tangents that cross the image curves. These tangents will be stable, but eliminating such tangents leads to a simpler canonical frame signature.

Cast Tangents

A cast tangent is a ray from the bitangent point which is tangent to the $\mathcal{M}$ curve. The cast tangent is made unique by selecting the tangent ray making the largest angle with respect to the bitangent line. The construction is projectively invariant and cast tangents are found in a manner similar to that for the bitangent point, again, localisation of the contact point is improved by quadratic fitting.

A sample segmentation for a simple view of a spanner is given in figure 14, in which the bitangent and cast tangent points and lines are superimposed onto the object. The bitangent points bound the $\mathcal{M}$ curves

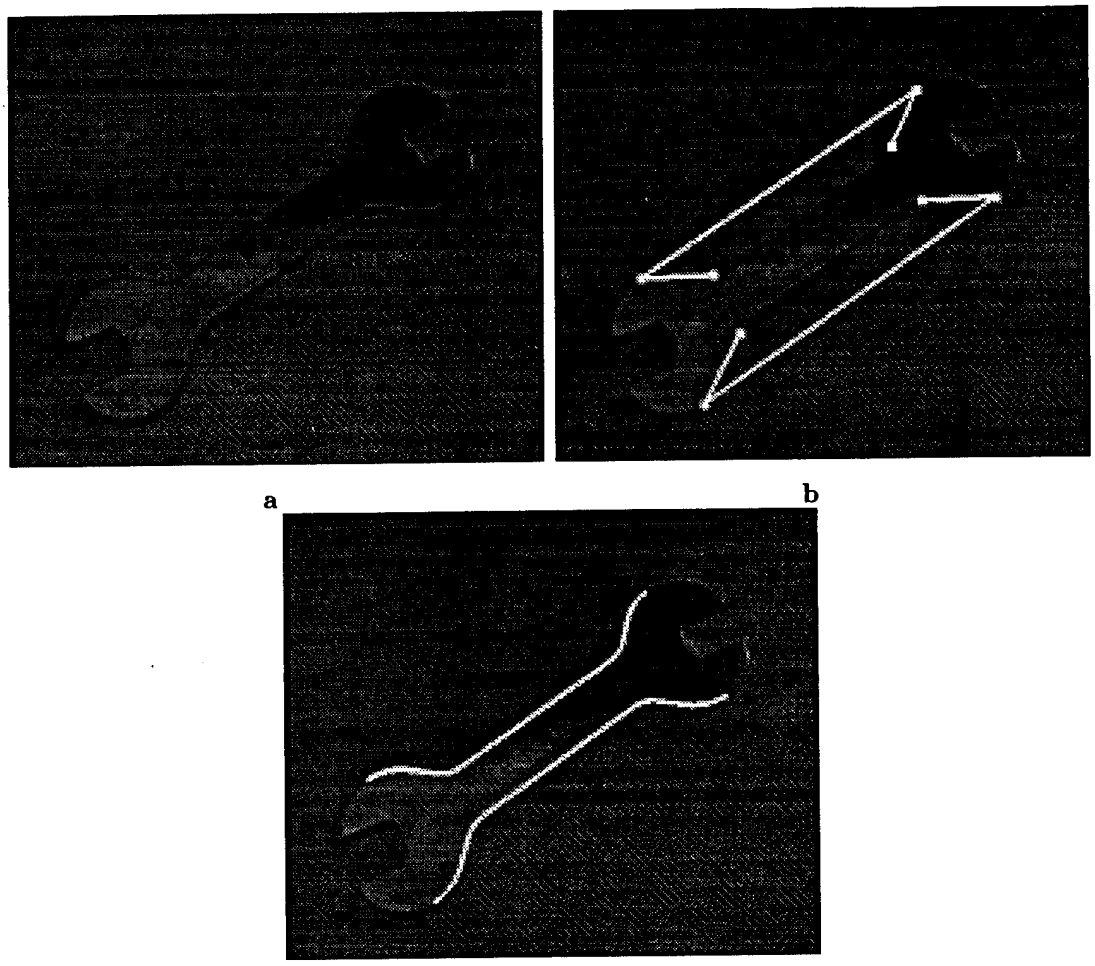


Figure 14: *For a simple object such as the spanner shown in (a) there are two reliable $\mathcal{M}$ curves that can be constructed. The bitangent and cast tangent points and lines are shown superimposed in (b). These are located to sub-pixel accuracy using the quadratic fitting method described in the text. The $\mathcal{M}$ curves bounded by their bitangent points are shown in (c). The $\mathcal{M}$ curves at the ends of the spanner are not used because their canonical frames cannot be determined stably.*

that are shown in (c).

Grouping

The canonical frame construction has a linear grouping cost. This is because all of the features used to form the frame are ordered around single image curves. This result is identical to that of [Huttenlocher88], and means that recognition using the construction is very efficient.

2.5 Errors in the Invariant Measurements

Before an indexing scheme is implemented, the error distribution of the invariant functions must be determined in order to determine whether a measured image index value is within an acceptable experimental

Table 4: *The mean values for the three invariants measured from the image sequence based on the images in figure 15. The standard deviation σ is computed both as an absolute value, and as a percentage of the mean. Note that the 3σ mark is well within 5% of the mean, and so such a bound could be used during recognition. During acquisition, we are more cautious, and use a tighter 3% bound on the allowable errors.*

	I_1	I_2	I_3
mean	(0.707,2.252)	(0.752,1.492)	(0.524,3.043)
σ	(0.0031,0.0170)	(0.0032,0.0086)	(0.0052,0.0433)
σ (%)	(0.44,0.76)	(0.43,0.58)	(0.99,1.42)

error bound of the actual model value. The rest of this section describes a pair of experimental investigations into the expected sizes of the invariant errors (one for algebraic invariants and one for canonical frame invariants). For the algebraic case the empirical investigation is compared to an analytic calculation.

2.5.1 Algebraic Invariant Errors

One can obtain a rough guide to the size of expected errors under *ideal* imaging conditions by differentiating the invariant expressions, and assuming an isotropic noise distribution; such analysis was done in [Forsyth91, Sinclair93], and is also given here. The short-comings of this type of formulation become apparent when real images are observed. The only way to understand the errors that may be encountered within a recognition system is to study real images. All theoretical analysis has to assume some error model in image measurements; frequently this is founded upon a Gaussian error in the locations of individual edge locations due to what is often called *image noise*. The results given below demonstrate that *errors* occur due to behaviour of standard edge detectors used, and cannot be attributed to random *noise*.

Empirical Investigation

The first and twenty-eighth images from the sequence used to do the tests are shown in figure 15. The rest of the sequence of fifty images were constructed by rotating the object at 2° increments on the calibration table beneath the object. The lines fitted to the edge data, with the seven lines used to compute the invariants, are shown in figure 16. The direction of rotation used to form the sequence is also marked. Three different five-line invariants were computed for each image of the object using these lines. Note that the object is specular, and is on a black background that is also somewhat specular. While the images do not represent ideal imaging conditions, edge detection is expected to be quite reliable, since image step edge contrast is large over the entire boundary.

The mean invariant values for the image set are shown in table 4. These results show that the invariants are in fact very stable, with standard deviations less than 1.5% of the mean values. From these results the error measurements that are used during recognition and acquisition are chosen. For the former the aim is to eliminate as many false negatives as possible and so the error bound is high (that is 5%, which is well above the 3σ mark), but during acquisition one should be more cautious so that only stable invariants are used (and so 3% is used, roughly equal to 2σ).

a b

Figure 15: *The first and twenty-eighth image in the fifty image sequence used to test the reliability of the invariants. The rest of the sequence was produced by rotating the calibration table by 2° between images. Three five-line invariants can be computed for this object using the seven longest lines. The twenty-eighth image is when line 3 (labelled on figure 16) becomes vertical and both lower and upper edges of the object are visible. From this viewpoint, the location of the edge boundary becomes ill defined.*

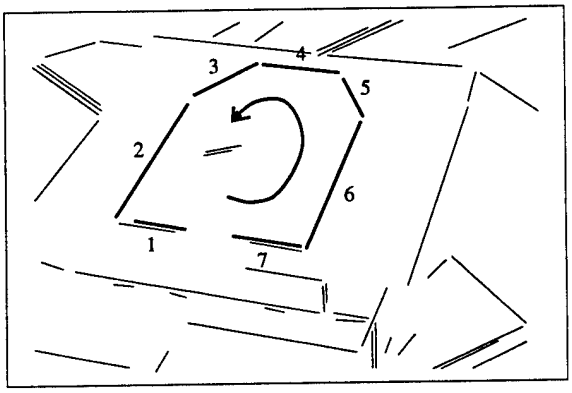


Figure 16: *The lines fitted to Canny edge data from figure 15a. The seven lines used to compute the invariants, and the direction of rotation, are marked.*

The value of I_2, computed on the sequence of lines 2 through 6 is plotted with an enlarged scale in figure 17. The shape of the graph is characteristic of all of the invariant constructions. The graph can be split into three distinct regions:

1. **Region A:** All of the lines are located reasonably well by the Canny edge detector, and so the measured invariants remain constant.

2. **Region B:** When the object has been rotated so that line 2 becomes vertical in the image, the edge on the lower surface parallel to it to becomes visible. The edge detector does not find a pronounced second edge in this orientation, but because of its presence the intensity values no longer form a step edge at the correct feature, but instead a slope. The Canny output locates a position somewhere along the slope and not at the top edge. The fitted line is therefore incorrect and causes the invariant value to be measured erroneously. Notice that as the object is rotated more, the invariant value tends to

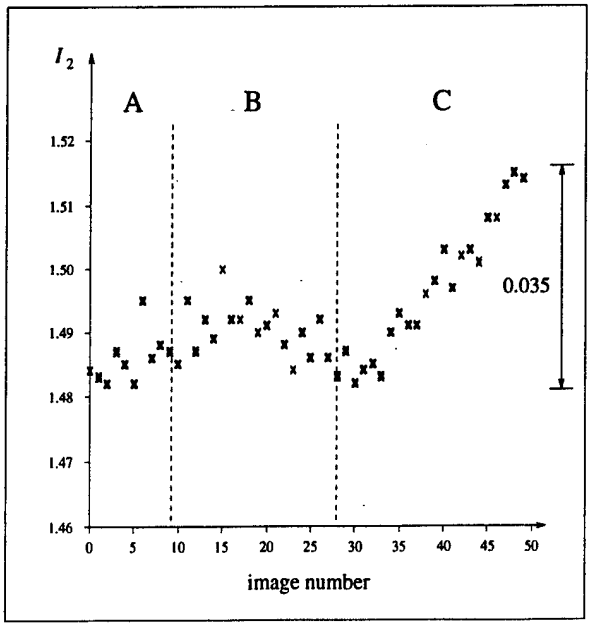


Figure 17: *When the invariants (in this case the second value of the second invariant) are plotted in greater detail, a systematic error becomes apparent in their measurement. This is due to the edge detector becoming distracted towards spurious image features, and is not due to image noise.*

decrease (though noisily), this is because the slope causes the fitted line to move further and further away from the correct edge.

3. **Region C:** In this region the effect is more pronounced as edge 3 moves through the vertical. When the fitted line drifts off the actual geometric edge, there is an obvious systematic error in the invariant measurement.

As can be seen from the graph the systematic errors produced by the edge detector far outweigh any Gaussian or quantisation noise observed in the points. Such noise will still be present, though its effects are small compared to other errors. It could be observed more clearly by removing the effects of the systematic error. Note that other unmodelled image events, such as shadows and close proximity of other objects, will also hinder the extraction of the planar geometric boundary.

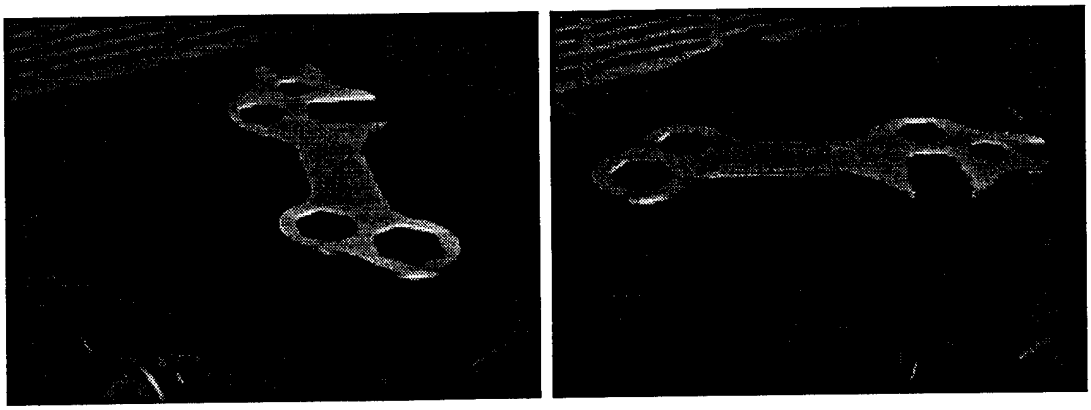
Analytic Investigation

The gross effects of the systematic error can be estimated by perturbing the invariant expressions. Given the expression for the second invariant:

$$I_2 = \frac{|\mathbf{M}_{421}||\mathbf{M}_{532}|}{|\mathbf{M}_{432}||\mathbf{M}_{521}|},$$

the aim is to determine the effect of, say, the third line on I_2. If the lines used to evaluate the expression are of the form $\mathbf{l}_i = (a_i, b_i, 1)^T$, $i \in \{1, \ldots, 5\}$, then:

$$\frac{\partial I_2}{\partial a_3} = I_2 \frac{|\mathbf{M}_{245}|}{|\mathbf{M}_{532}|.|\mathbf{M}_{432}|} (b_2 - b_3),$$



a b

Figure 18: *The first and twentieth views of the spanner in the sequence are shown. The $\mathcal{M}$ curve of interest is the left-most one in (a), and the distant one in (b).*

$$\frac{\partial I_2}{\partial b_3} = I_2 \frac{|\mathbf{M}_{245}|}{|\mathbf{M}_{532}|.|\mathbf{M}_{432}|} (a_3 - a_2). \tag{6}$$

The model for the error observed in the measurement of the fitted lines is a translation parallel to the line normal by δ. If the gradient of the line is $\tan\theta$, by letting $\alpha = \delta/\cos\theta$ the equation of the perturbed line is:

$$\frac{a_3}{1-\alpha b_3}x + \frac{b_3}{1-\alpha b_3} + 1 = 0.$$

This directly yields $(\partial a_3/\partial\delta, \partial b_3/\partial\delta)$, from which δ can be estimated given a known ΔI_2.

From region C of figure 17 the value of ΔI_2 can be estimated as 0.035. This is assumed to be due entirely to the movement of $\mathbf{l}_3$ and that all the other lines are measured correctly. Applying the analysis yields a value of $\delta = 2$ pixels for this ΔI_2; this certainly is of the right order of magnitude for the error in fit observed in the image sequence.

2.5.2 Canonical Frame Invariant Errors

An empirical experiment similar to that for the five line invariant has been done for an object for which canonical frame invariants can be computed. Two images from the sequence used to measure the invariants are shown in figure 18. The value of the first invariant measured for each image is plotted in figure 19 against spanner orientation, which is varied through 180^o in 5^o increments. Note that the value of the invariant is stable, but again a systematic error is apparent when the graph is observed in more detail.

3 Recognition

3.1 Overview

An outline of LEWIS is shown in figure 20.

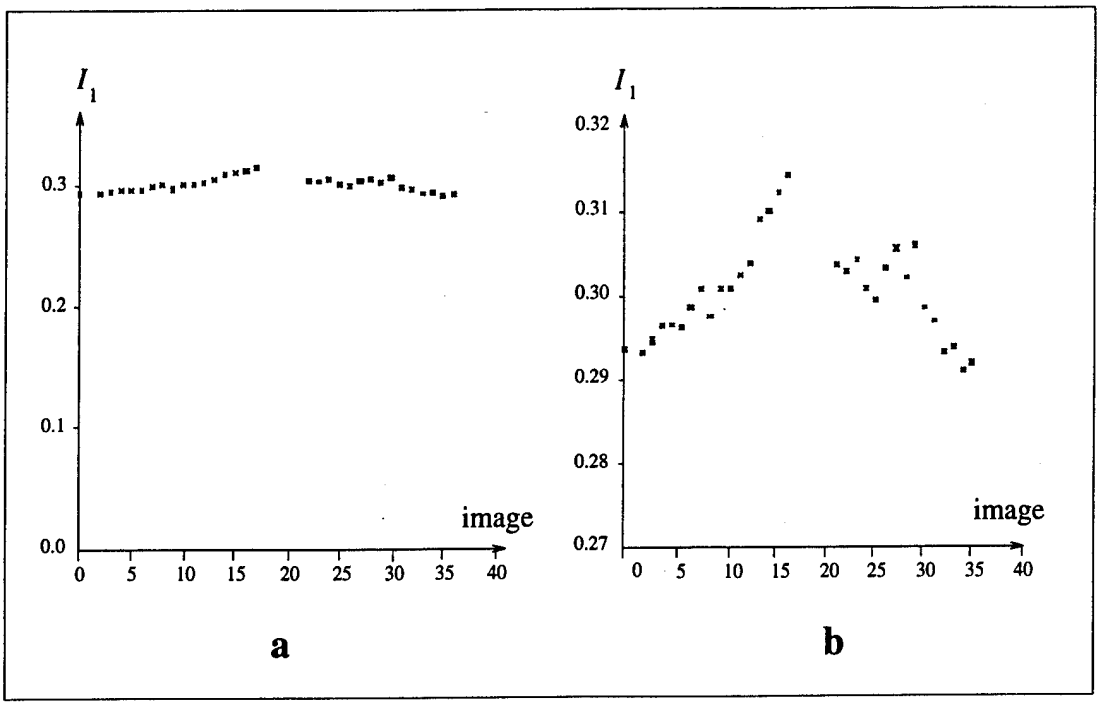


Figure 19: *The first invariant measured for the image sequence in figure 18 is plotted as the spanner is rotated by 5° between images. Note in (b) that the edge detector produces a systematic error due to the it being distracted by the finite thickness of the object.*

3.1.1 Feature extraction and Invariant Formation

The goal of feature extraction is the formation of geometric primitives suitable for constructing invariants. In the algebraic case this involves straight lines and conics, and for non-algebraic curves, $\mathcal{M}$ curves delineated by bitangents. The fitting and grouping processes were described in section 2.4.

Once sets of grouped features, $\mathbf{F}$, have been produced, the invariants listed in sections 2.2 and 2.3 are computed. Each set of grouped features, or $\mathcal{M}$ curve, produces a number of invariants (one or more) which form a vector[2] $\mathbf{M}(\mathbf{F})$. Of course, if the object is occluded to the extent that the number of features visible is insufficient for an invariant, then no index can be formed.

The invariant vector formed by the above process (when quantised), represents a point in the multidimensional invariant space. Each object feature group is represented by a collection of points that define a region in the invariant space, the size of which depends upon the measured variance in the invariant value (see section 2.5).

3.1.2 Indexing

The invariant values computed from the target image are used to index against invariant values in the library. If the value is in the library, a preliminary recognition hypothesis is generated for the corresponding object.

[2] At this stage in the processing, each feature group is used to form a separate $\mathbf{M}$ vector. Interactions between the groups are only considered later.

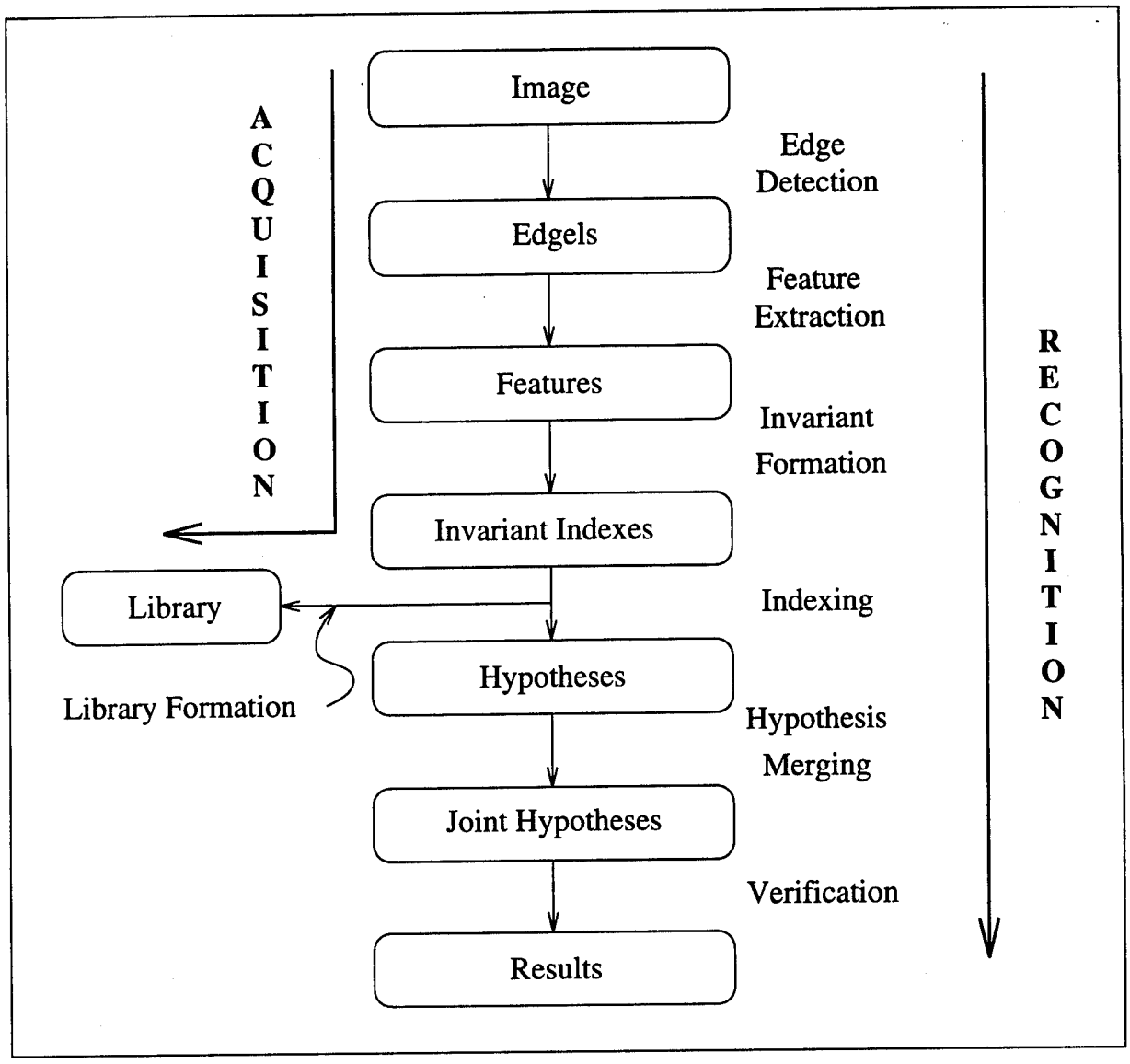


Figure 20: *LEWIS has a single greyscale image as input and the outputs are verified hypotheses with associated confidence values. Many of the processes are shared by the acquisition and the recognition paths. The recognition system is similar to previous systems in all but the indexing and hypothesis formation stages [Grimson90].*

Each type of invariant (for instance that for five lines, or a conic pair) generate separate hypotheses.

This process is made more efficient using a hash table that allows simultaneous indexing on all elements of the measurement vector. In the experiments to date there has not been any significant problem with collisions in the hash table. Hash table collisions[3] should not be confused with the intersection of object invariant measurements in index space. These intersections lead to erroneous hypotheses which cost some effort during the verification stage, but are usually eliminated.

3.1.3 Hypothesis Merging

Because many invariants may actually correspond to the same object, and should therefore be covered by a single recognition hypothesis, *joint hypotheses* are formed prior to recognition by combining 'compatible' hypotheses. There are number of reasons why hypothesis merging is desirable:

1. Backprojection and searching for image support (verification) is computationally expensive and it is more efficient to validate several hypotheses of the same object together.

2. More features facilitate a more accurate least squares calculation of the back projection transformation (there are more matched model and image features), and consequently a reduced error in measuring image support.

3. Many hypotheses indexing the same object in a single part of the scene significantly increase confidence that the match is correct.

During hypotheses merging, an interpretation tree is constructed for each object. The features used in the tree are the groups of invariant features that were successful in indexing. The merging process utilises topology and geometric compatibility. The topological consistency (ordering and connectedness) is illustrated in figure 21. Geometric consistency is implemented efficiently by a second use of invariants; this time joint invariants between the feature groups used to compute each individual hypothesis. This is illustrated in figure 22.

Since topological relations are often unreliable it is possible that two hypotheses could be united into a single joint hypothesis even though they are totally unrelated (for example one may represent a correct match and the other may have been caused by clutter). A list of all the original hypotheses and all possible combinations of compatible hypotheses is therefore maintained. The list is ordered by descending number of simple hypotheses per joint hypothesis. Those with more simple hypotheses are verified first, and if the match is confirmed, other joint hypotheses that represent partial versions of the hypothesis are deleted. If the match is not confirmed only the joint hypothesis under consideration is deleted. The joint hypothesis formation

[3] A hash table collision occurs when a number of models have the same hash index. Such a collision can occur when the number of hash buckets is smaller than the model population or when the hashing function is not uniform and causes many models to hash to the same bucket.

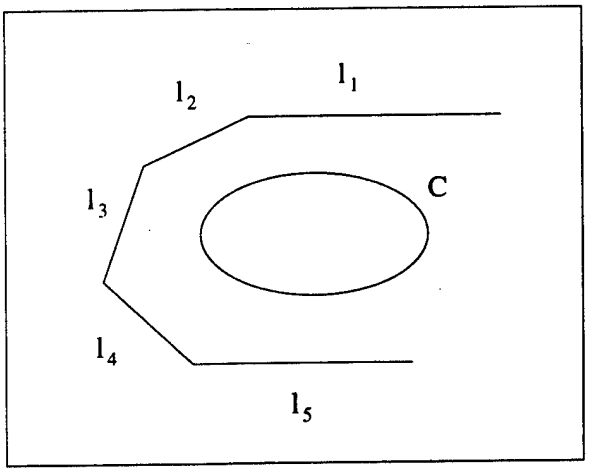


Figure 21: *If the same model is indexed by a five-line invariant (due to lines* $\mathbf{l}_i$, $i \in \{1, \ldots, 5\}$*), and a conic three-line invariant that is* compatible *with it (due to* C *and* $\mathbf{l}_i$, $i \in \{2, \ldots, 4\}$*), then it is wise to verify both hypotheses together. The invariants are compatible if the ordering of the image lines are consistent with those on the model; see text for details.*

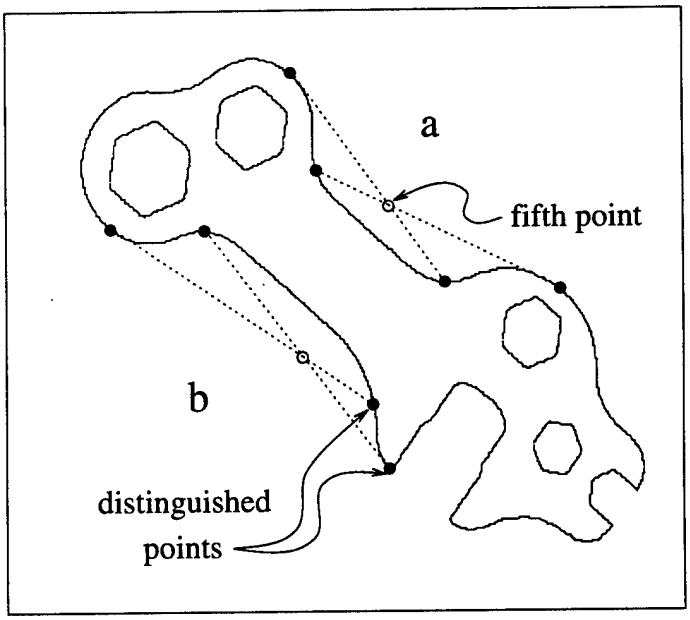


Figure 22: *For a pair of* $\mathcal{M}$ *curves there are 8 distinguished points which could be used to form* $2 \times 8 - 8 = 8$ *different five point invariants. Rather than computing so many, which is unnecessary, invariants are computed between the four distinguished points of each* $\mathcal{M}$ *curve, and the 'central' point of the other. This yields four invariants, and does so using a symmetric construction. These invariants are sufficient to hypothesise compatibility.*

stage can potentially cause an exponential number of hypotheses to be formed. However, in practice, deleting verified hypotheses keeps the verification process under control (as is shown later in table 6).

3.1.4 Verification

There are two steps involved in verification, both of which can reject a (joint) recognition hypothesis. The first is an attempt to compute a common projective transformation between the model features and the putative corresponding features in the target image. The second is to use this transformation to project the entire model onto the target image, and then measure image support.

Incorrect hypotheses arise because grouped image features happen to have an invariant value that coincides (within the error bounds) with one in the library. Also, because the invariants are not complete (completeness is defined in detail in [Rothwell94]), structures with the same invariant may not be projectively equivalent. The features used to produce the matching model and image invariants provide sufficient constraints to compute the projective transformation between the model and image[4]. In general, the projective transformation is over determined as the feature groups tend to provide more than the required eight constraints. Consequently, if a common transformation cannot be computed, the features are not projectively equivalent and the hypothesis is rejected [Rothwell94].

Backprojection and subsequent searching involves the entire model boundary, not just the features used to form the invariant. Projected model edgels must lies close to image edgels with similar orientation (within 5 pixels and 15^0). In the case of algebraic features, two preliminary hypothesis filtering steps can be invoked:

1. The model lines must project to within 15^o of the image lines.
2. The projected model conics must project to ellipses, and they must have similar circumferences and areas to the image conics.

Orientation in the target image is determined from the Canny edgel orientation. The orientation of the projected model feature is determined as follows:

1. For model edgels on straight lines the projected orientation of the line is used for each edgel.
2. For model edgels on conics, the orientation is obtained from the projected conic via their polars[5] which are close approximations to the tangents for edgels close to the conic.
3. For other edgels, the orientation provided by the edge operator is used. This orientation is determined in the target image by projecting the tangent line to the model. Model edgel orientations are less accurate (than a fitted line or conic), so a 30^o threshold is used instead of 15^o.

[4] Unless the invariant exploits an isotropy. In this case, certain parameters of the transformation are unrecoverable because they do not affect the projected geometry, e.g. a circle under rotation about its center.

[5] For a point $\mathbf{x}$ and a conic $\mathbf{C}$, the polar $\mathbf{l}$ of the point with respect to the conic is $\mathbf{l} = \mathbf{Cx}$.

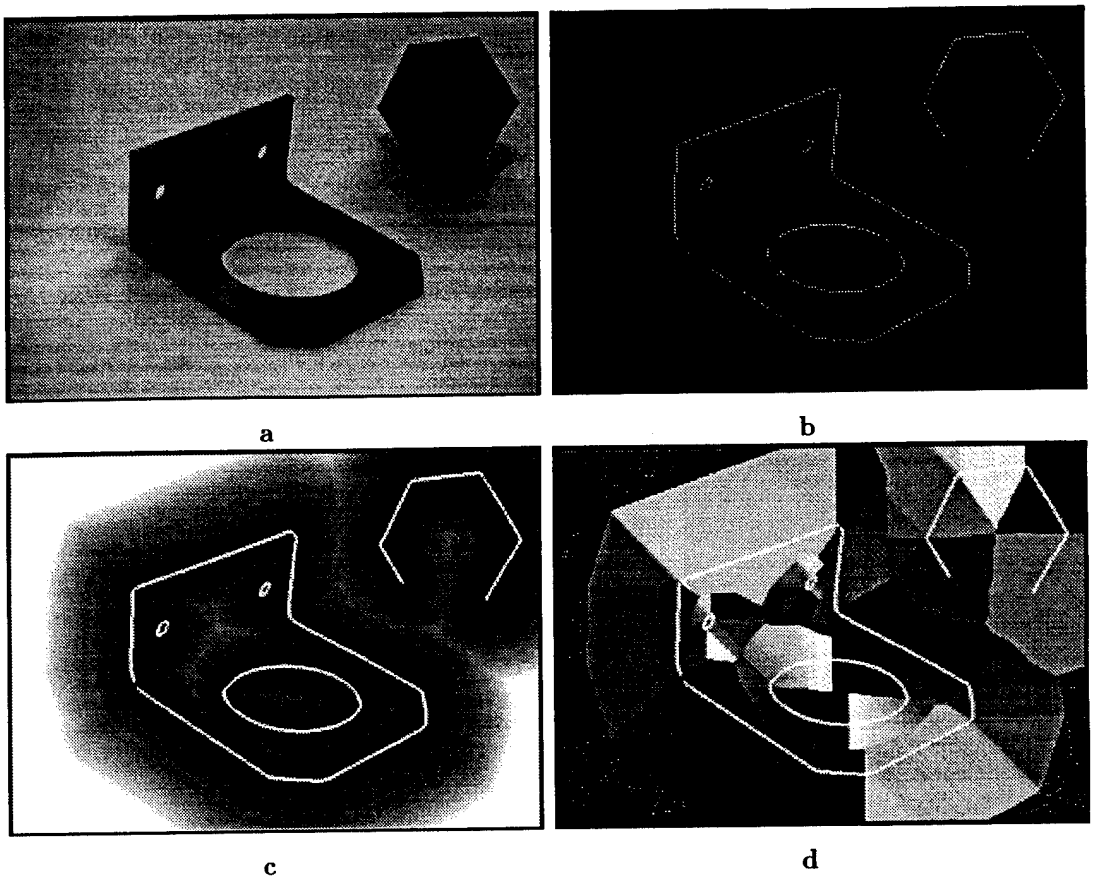


Figure 23: *(a) shows a simple scene with two objects in it. The output of the Canny edge detector is shown in (b). The 3-4 distance transform is computed for all edges over a certain strength and is displayed in (c); distance from the edge (white) is coded by intensity, with zero being black. (d) shows, coded by intensity, the orientation of the the edge nearest to a given image point.*

If more than a certain proportion of the projected model data is supported (the threshold used is 50%), there is sufficient support for the model, and the recognition hypothesis is confirmed. The final part of the process is expensive as $O(10^3)$ edgels need to be mapped onto the image. Efficiency in the distance computation is achieved by approximating the distance using the 3-4 distance transform of Borgefors [Borgefors88]. The distance transform is found by passing chamfer masks over the image, which is carried out within image preprocessing. An example of the 3-4 distance transform output for a simple image is shown in figure 23.

If the projected model is too small in the scene it must have arisen from an object so far away that it would not be observed reliably. An upper bound on the size of the projected model can be computed by bounding the model by a box and projecting that to the image first; if it is too small, then the hypothesised object must be too small and so can be rejected. In practice the bounding box used is the perimeter of the acquisition image.

There is a trade off involved in setting the support threshold. A heavily occluded correct match may have as much support as an incorrect match. Particularly if there is dense edge data (such as wood texture), then it is quite likely that a large number of edges may be close to, and have the same orientation as, the

projected model edges. In a structured scene, a few erroneous straight lines of the right orientation will be sufficient to give over 50% support for a model, and so render a false positive. Obviously, any object which is over 50% occluded will not be found by the recogniser. As the threshold is lowered, an occluded object is more likely to be found, but there will also be more false positives. On the other hand, if more than one invariant forms a hypothesis that passes verification, the level of confidence in the result is high. This is discussed further in section 4.1.

3.2 Model Acquisition and Library Formation

A model consists of the following:

1. A name.
2. A set of edge data from an acquisition view of the object for use in the backprojection stage of verification.
3. The lines, conics and $\mathcal{M}$ curves that represent the edge data.
4. The expected invariant values and which algebraic features or $\mathcal{M}$ curves they correspond to.
5. The bounding box of the model features.
6. Topological connectivity relations between feature groups that will be used in the construction of joint invariants.

The library is segmented into different sub-libraries, one for each type of invariant. Each sub-library has a list of each of the invariant values tagged with an object name, and is structured as a hash table.

One benefit of using only projective representations, rather than Euclidean, is that model acquisition can be done directly from images. No special orientations or calibrations are required. Acquisition is simple and semi-automatic (for instance, curves do not have to be matched by hand). It proceeds as follows:

1. A number of images are taken of the isolated object from a variety of 'standard' viewpoints (for algebraic invariants two images are used, generally for the canonical frame system more are required to compute the Fisher discriminant).
2. The invariants are computed for each view. This involves the same segmentation and invariant computation as used during recognition. For non-algebraic curves *significant* $\mathcal{M}$ curves are extracted.
3. The invariant values are compared between views. The useful invariant shape descriptors will remain reasonably constant under a change in viewpoint. These are recorded in the modelbase. Any measures that are not constant are due to features that do not form correct invariant configurations (for instance lines that are not coplanar), or are caused by unstable features. For matching values (within 3%, see section 2.5), the mean value is entered into the model library.

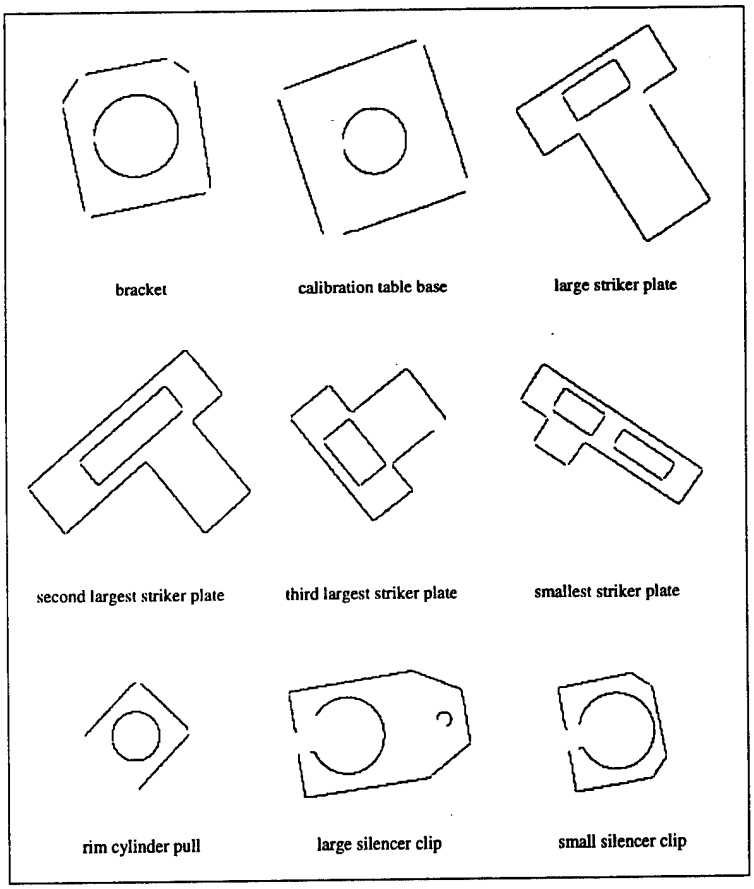


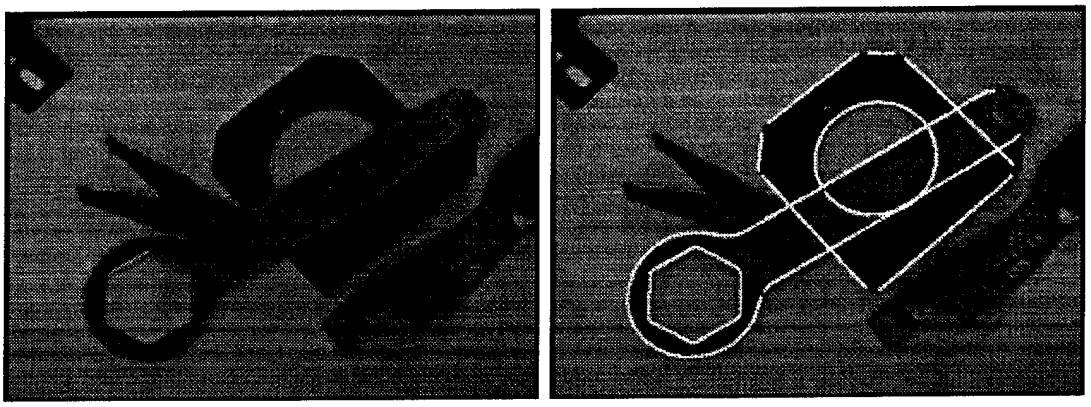
Figure 24: Nine of the models in LEWIS's model base (the edge data of the models is shown).

4. Connectivity between algebraic features or $\mathcal{M}$ curves is utilised to form joint invariants to be used during hypothesis combination.

3.3 Algebraic Invariants Examples

The results reported here have been carried out with a model library containing over thirty objects. Typical algebraic objects in the library are shown in figure 24. Recognition accuracy is excellent if the object boundaries are not severely disrupted by shadows and specularities. On a SPARC IPX, edge detection takes 15 seconds; feature extraction 5 seconds; matching less than a second; and verification normally about 2 seconds.

The first recognition example is a bracket in a scene with occlusion and clutter caused by other objects (figure 25). All possible algebraic invariants are formed from configurations of lines and conics. The measured and the matching values are given in table 5. From a scene such as this, a large number of possible invariants can be derived. It was found that two image five-line invariants matched model invariants of the bracket, with the second (incorrect) one ruled out during verification. Three conic-and-three-line invariants were measured in the scene that matched the invariants of the bracket, and all these constituted correct matches.



a b

Figure 25: *(a) shows the bracket occluded in a scene. Some of the occlusion is due to an object not in the model library. In (b) the edge data from the first calibration scene are shown projected onto the test scene using the model to image transformation hypothesised by the match. The close match between the projected data (shown in white), and the scene edges shows that the recognition hypothesis is valid. Projected edge data from the model of a spanner are also shown projected into the scene as this was also recognised.*

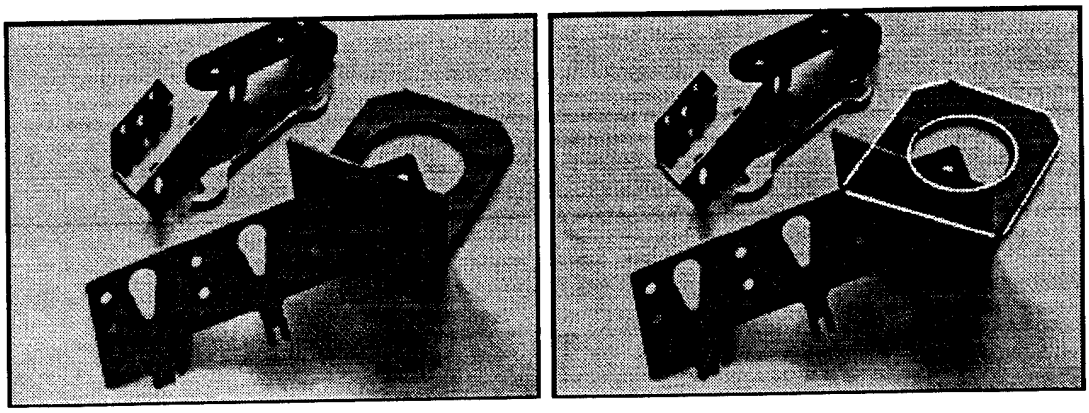
Table 5: *The invariants measured from figure 25 which are formed by features actually corresponding to bracket features. The second column shows the library values and the third column scene values. In the fourth column the deviations from the mean invariant values are given; this shows that the five-line invariant is very stable under real image conditions, and the conic-and-three-line invariant is reasonably stable.*

invariant	library	scene	error %
five-line	(0.8415,1.2340)	(0.842,1.235)	(0.1,0.1)
conic-line	(1.3410,1.3080,2.6285)	(1.372,1.291,2.676)	(2.3,1.3,1.8)
conic-line	(1.3080,1.3025,1.8850)	(1.291,1.287,1.852)	(1.3,1.2,1.8)
conic-line	(1.3025,1.3395,2.5915)	(1.287,1.365,2.604)	(1.2,1.9,0.5)

Incorrectly indexed hypotheses can be ruled out during verification when the hypothesised model is projected into the scene (all such hypotheses were ruled out in this case). For the bracket, 74.5% of the projected edges match to within 5 pixels and 15° of the image data. There is a second object from the model base, a spanner, also in the scene. This is correctly identified using three different invariants. In this case an 84.5% projected edge match is achieved with the model data also shown in white in figure 25.

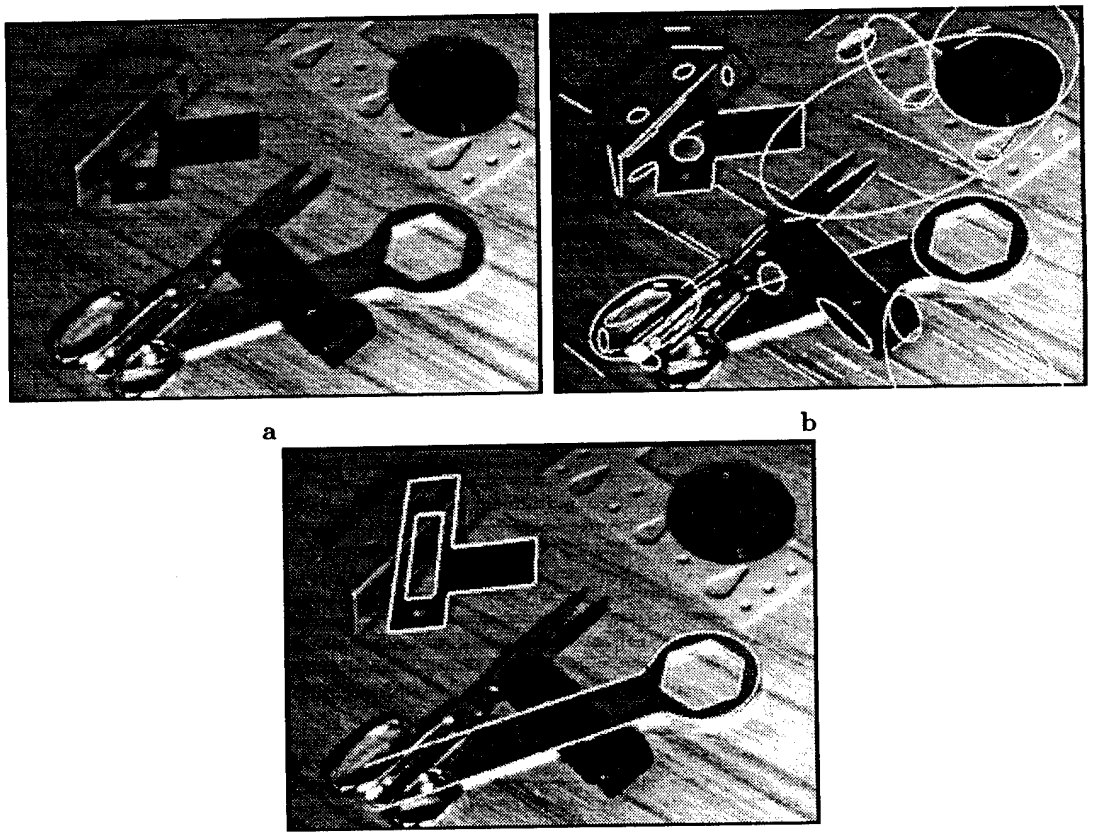
In figure 26 the bracket is recognised despite a significant amount of occlusion (in this case there is only a 59.3% edge match during verification). Figures 27 to 45 show the system operating on a few test scenes with some of the match statistics shown. For figure 27, 1049 invariants were computed which indexed 41 hypotheses. These were converted into 131 joint hypotheses that had to be verified, of which 13 were rejected by first stage verification, based on valid projective transformations, and 78 required the second stage, based on image support. For figure 28, 806 invariants indexed 36 hypotheses, forming 44 joint hypotheses of which 23 needed the second verification stage after 13 were rejected by the first stage.

In table 6 various match statistics are shown that have been taken from a number of scenes (around 100) similar to that of figure 26. In each case, a single object from the model library was in the scene, and it was recognised correctly in all except one instance which was when verification broke down due to



a b

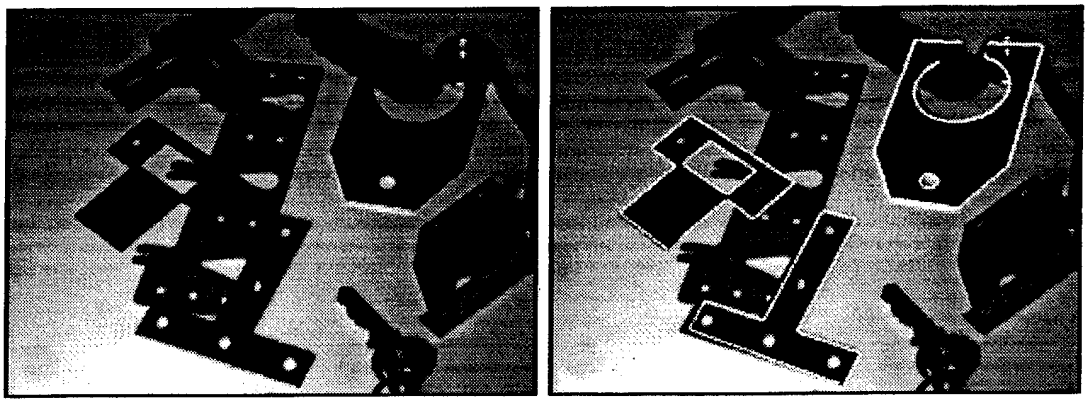
Figure 26: *(a) shows the bracket occluded in a scene by objects not in the library. In (b) the edge data from an acquisition scene are shown projected onto the test scene using the model to image transformation hypothesised by the match. The close match between the projected data (shown in white), and the scene edges shows that the recognition hypothesis is valid.*



a b

c

Figure 27: *(a) shows a scene containing two objects from the model base, with fitted lines (100 of them) and conics (27) superimposed in (b). Note that many lines are caused by texture, and that some of the conics correspond to edge data over only a small section. The lines form 70 different line groups. (c) shows the two objects correctly recognised, the lock striker plate matched with a single invariant and* 50.9% *edge match, and the spanner with three invariants and* 70.7% *edge match.*



a b

Figure 28: *Another typical scene containing three objects from the model base. The recognised objects are outlined with* 74.7% *(2 invariants),* 84.6% *(1 invariant) and* 69.9% *(3 invariants) edge matches for the objects from left to right. 58 lines and 14 conics were found.*

a poor segmentation preventing a sufficient amount of edge support. The total number of indexes formed (an average of 1755.3), depends solely on the number of features in the scene, and the way in which they are grouped. This number roughly equates to the number of hypotheses that would have to be verified per model for a hypothesise and test technique. After indexing, these form only an average of 60.4 hypotheses, which constitutes a nearly thirty-fold reduction. Because of redundancy in the shape representation, multiple hypotheses can occur for a single model instance. Joint hypothesis formation processing yields an average of 72.7 joint hypotheses.

Verification is performed once the joint hypotheses have been constructed. On average, 5.9 hypotheses do not have to be verified as their structures are subsumed by larger joint hypotheses. This means that only 66.8 joint hypotheses actually require verification, which is similar to the original 60.4 individual hypotheses. It is clear that the joint hypothesis formation stage does not lead to an exponential number of hypothesis being formed, and yet it provides improved recognition. Once the projectivity between the hypothesised models and the image features has been computed, a check is made that the projected and image algebraic features are consistent (the preliminary filter). On average this filter removes 41.1 hypotheses, 6.4 due to line correspondences, 4.9 for conic and line configurations, and 29.8 for conic configurations. In the end, only 23.7 hypotheses have to be verified through full back projection, compared with the 1755.3 original indexes formed.

In each case a single object should have been recognised. Essentially, a negligible number of false negatives are observed. One false positive, on the average, is successfully verified in a given image in addition to the correct model hypothesis. The false positive is partly due to the symmetry of an object, where the projected boundary can achieve good support from the set of image features, even though the correspondences and object pose are incorrect (such as in figure 42). False positives also occur due to confusion between projectively similar objects, that is, the projective transformation generates large shape equivalence classes.

Table 6: *The average match statistics for using algebraic invariants within LEWIS taken over a large number of images. See the text for an explanation.*

number of actual model instances	1.0
total number of indexes formed	1755.3
total number of individual hypotheses	60.4
total number of joint hypotheses	72.7
number not requiring any verification	5.9
number rejected by algebraic test	41.1
number rejected through full back projection	23.7
number of correct hypotheses	1.0
number of false positives	1.0
number of false negatives	0.0

3.4 Canonical Frame Invariants Examples

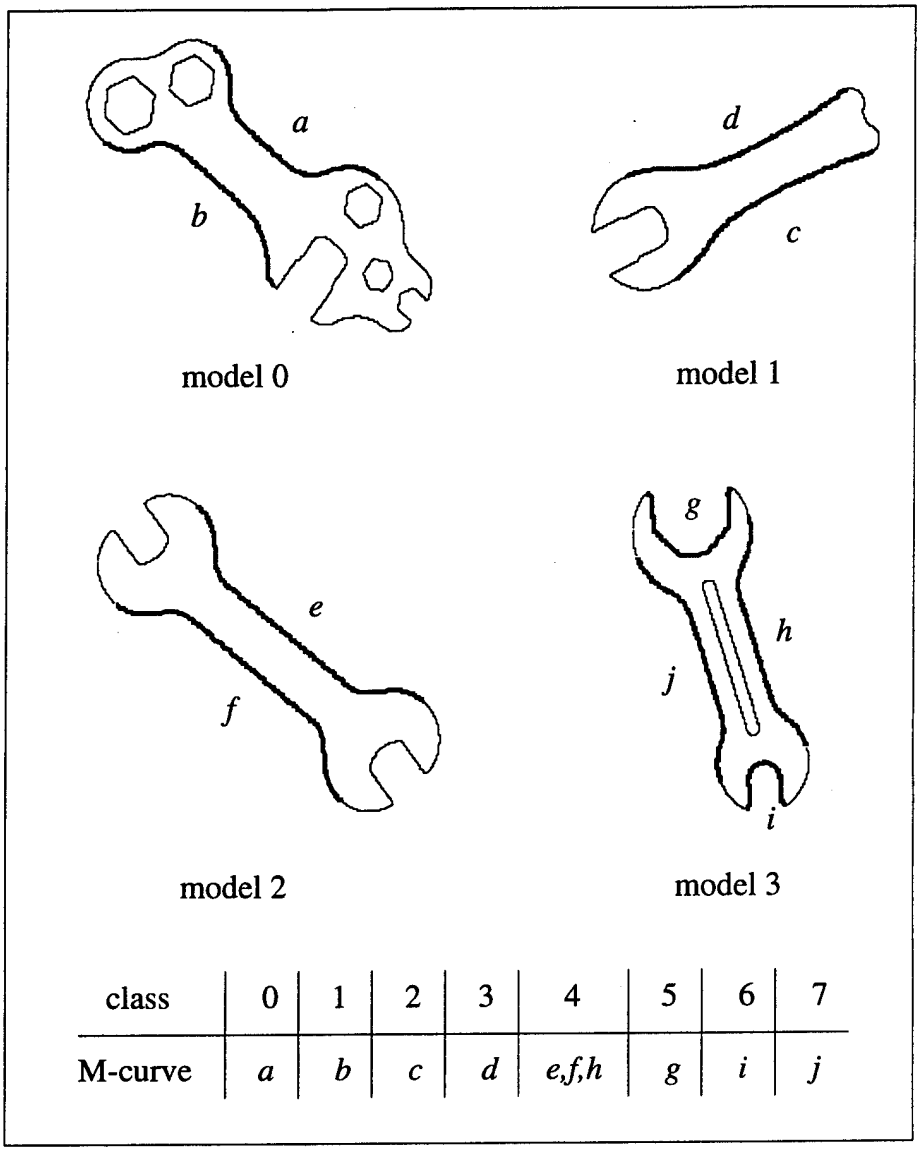
3.4.1 Classes

Typical (non-algebraic) objects in the library are shown in figure 29. The object $\mathcal{M}$ curves are sufficiently similar (in all cases there are only two inflections) to allow a grouping of the library into a number of *classes*, see figure 30. Indexing is then *hierarchical*: first, *sub-parts* (classes) are indexed and verified. For the class verification, rather than backproject the whole model curve, the $\mathcal{M}$ curve alone is projected into the canonical frame. It is verified by measuring the difference in areas between the image class and the model class curves (computed using rectangular quantisation in the canonical frame). If the difference is sufficiently small, the hypothesis is accepted. This covers the non-completeness of the canonical frame invariants. Second, if the class is accepted, hypotheses are generated for each of the models in that class. Joint hypotheses are then formed and verified by back projection to the target image using the entire boundary model curve.

The efficiency of the indexing process can be demonstrated empirically: from a series of typical images an average of 56 $\mathcal{M}$ curves were observed; 27.8% of these produced class hypotheses on indexing; and 23.9% of these were verified as classes (only 6.6% of the original number of $\mathcal{M}$ curves). Note that although a large number of classes were hypothesised in the scene, only 14.0% of the indexed hypotheses were later found to be incorrect. Based on these preliminary results (56 $\mathcal{M}$ curves found in image, 10 model curves, no false positives) it would seem that there is not an excessive tendency towards false positives.

3.4.2 Recognition Examples

The first example shown in figure 31 shows a simple unoccluded view of model 0. This object can be recognised using up to two classes. First, the classification algorithm locates classes 0 and 1 as marked in figure 31b, and uses these to form a single joint hypothesis by the procedure of section 3.1.3. The joint hypothesis is verified through backprojection in which 92.8% of the model outline is matched to image data. A 100% confidence is not found (as would be expected for an unoccluded object) because the Canny edge detector fails to extract and localise all of the object edges correctly. This results mainly from specularity on the object. The same effect can be observed in all of the images in this section because the objects are



class	0	1	2	3	4	5	6	7
M-curve	*a*	*b*	*c*	*d*	*e,f,h*	*g*	*i*	*j*

Figure 29: *For the model base consisting of the four spanners there are ten useful $\mathcal{M}$ curves. These are shown by thick lines and labeled (a) to (i). Due to the projective similarity of (e), (f) and (h), eight classes are sufficient to represent the local shapes of the spanners. The correspondence between the $\mathcal{M}$ curves and the classes is given in the table. The global shape of each spanner is also required for recognition, this includes the geometric constraints between each class (see section 3.1.3 for details), and also the entire set of edge locations and orientation data; this is used for verification.*

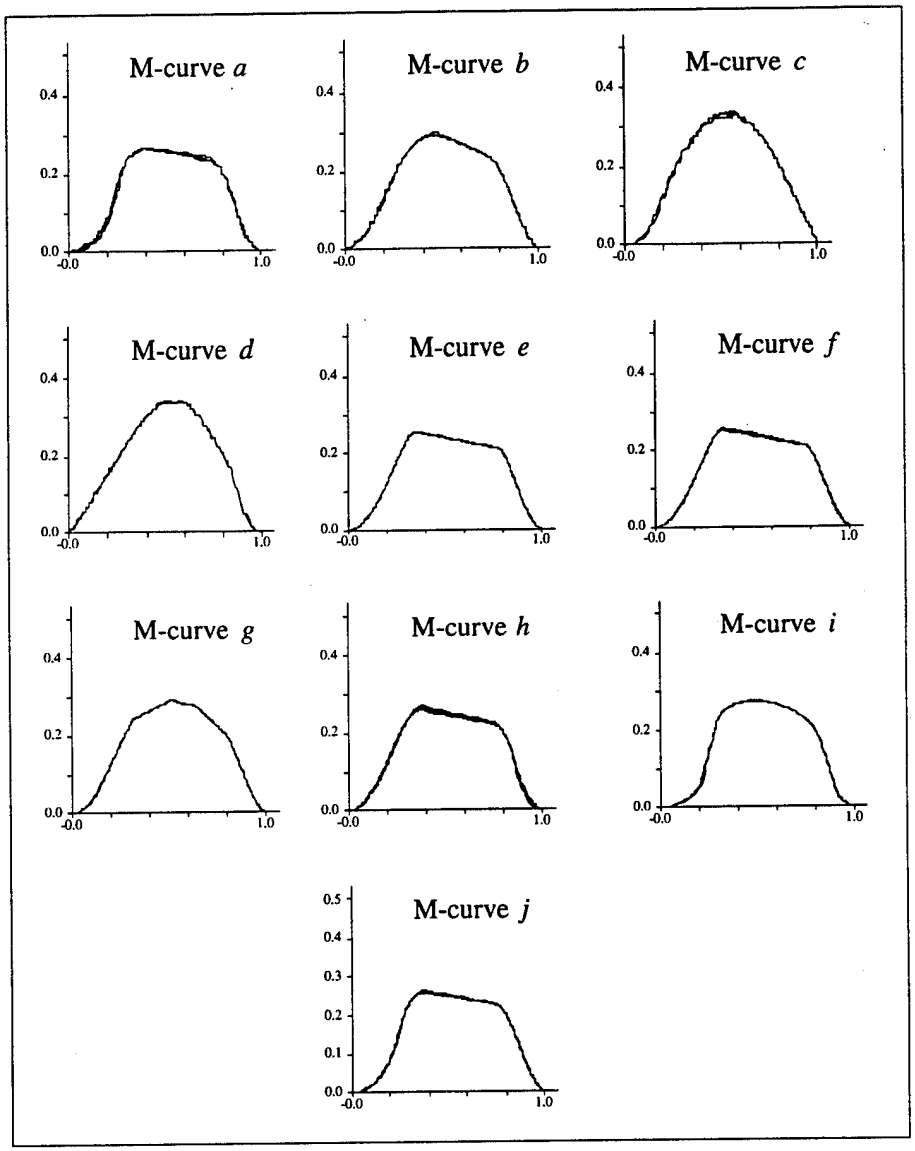


Figure 30: *The canonical curves for the four models shown in figure 29. Three images of each object were used and the curves superimposed; the very close match between each curve highlights the stability of the construction. Note the similarities between signatures (e), (f) and (h); these are essentially the same and are therefore represented by the same class. All the other signatures are in their own class.*

Table 7: *Matching statistics for figures 34 to 37. The number of $\mathcal{M}$ curves extracted from the images and how many class hypotheses result from indexing are shown. The class hypotheses are used to form joint hypotheses that are verified or rejected by the following tests: if a larger subsuming joint hypothesis has already been accepted; if a good model to image projectivity cannot be computed; if backprojection results in an impossible pose.*

Figure	#$\mathcal{M}$ curves	#classes	#jh	#no verification	#poor proj.	#poor pose
34	42	13	18	0	1	5
36	79	18	23	2	0	3
37	99	24	39	4	1	2

metallic. Another cause of edge segmentation failure is due to the finite thickness of the objects, as discussed in section 2.5.1. Frequently, an edge extracted from the image can swap between portions of the outline on the upper and lower surfaces of an object. As the canonical construction is local this does not present a major problem, though its effects are occasionally noticeable.

In figure 32 the recognition system is tested on a more complex scene where there is clutter and occlusion. A single class is found for model 3 (class 5), which is then localised correctly in the image to give 55.5% edge support. Although a total of 16 class hypotheses were formed, yielding 22 joint hypotheses, only the correct hypothesis was given sufficient confidence by backprojection (over 50% projected edge support).

The canonical frame construction works very well under significant perspective distortion. This is demonstrated in figure 33. For this relatively simple scene three classes are found, and only one produces a hypothesis that passes through the object verification procedure. This gives an 83.6% edge match. As may be seen from figure 29, within the range of typical signature variation, model 2 (which is the one identified in figure 33) is projectively 2-cyclic[6]. Thus, the spanner will always be projected into the image in two different poses differing by the equivalent of a 180° rotation, and still match correctly.

Figures 34 to 37 show further recognition examples in which the correct objects are always recognised. No false positives were found in any of the images, though this is not always the case. Although some instances have sufficient edge support the hypothesis is rejected based on size, as described in section 3.1.4. For example, model 0 was identified as subsequently rejected as shown in figure 35. Full details of the recognition performance are given in table 7.

The algebraic invariant and canonical frame representations can be independently applied to an image to recognise objects of both types. Figure 38 shows an example of recognition for both index methods together.

3.5 Complexity

The grouping cost incurred in forming the invariants was discussed in section 2.4. Here we first propose a simple model for recognition complexity, and then verify this experimentally.

[6] An object is projectively 2-cyclic if there is a view of the object for which it can be mapped onto itself with a 180° rotation.

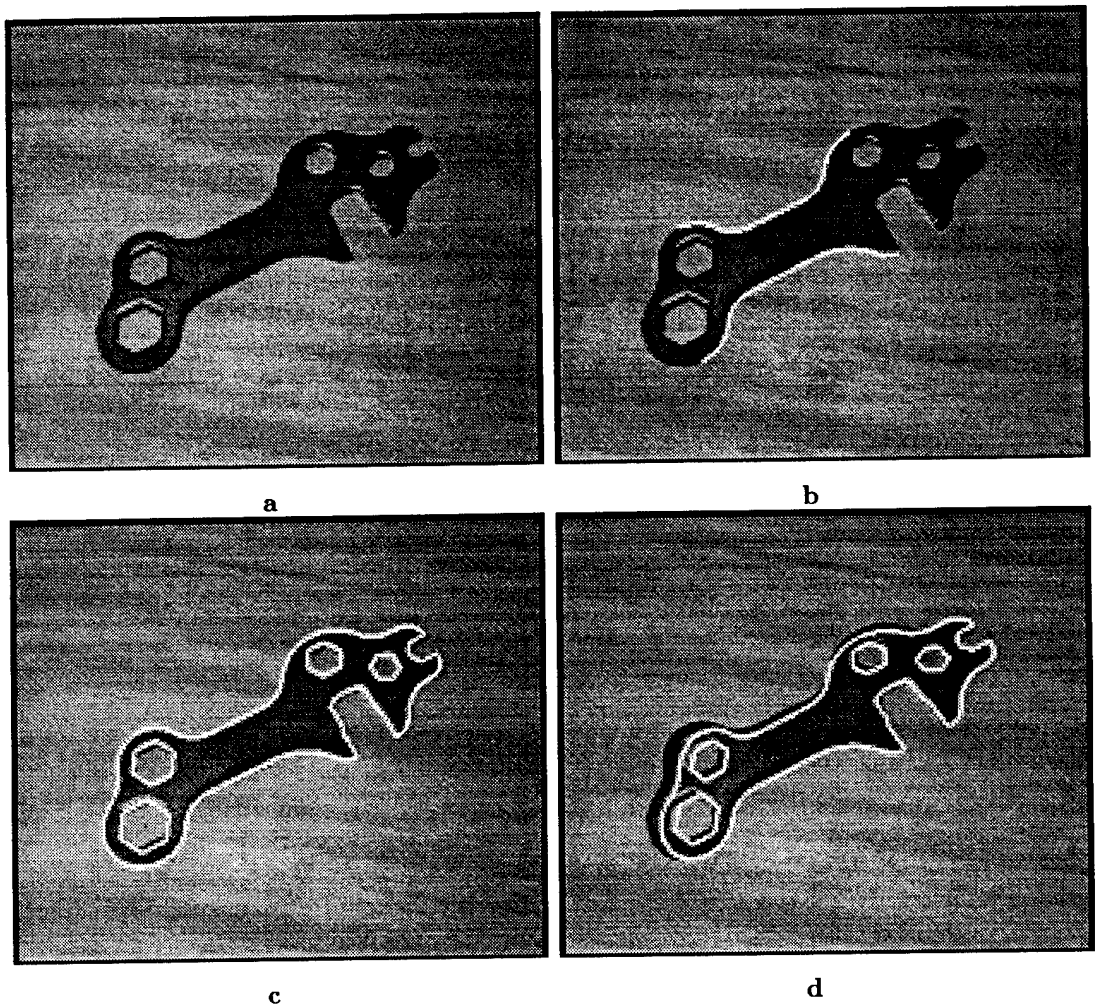


Figure 31: *(a) shows an unoccluded view of model 0. The classifier correctly locates classes 0 and 1 in (b). These are used to form a joint hypothesis for model 0 which is verified using back projection that finds* 92.8% *image support for the model. This is the only model match found that has a reasonable pose (that is, the object is not too small). Note that very good registration of the object is achieved in (c), this is when both* $\mathcal{M}$ *curves are used to compute the model to image transformation. Sometimes, as in (d), if a single* $\mathcal{M}$ *curve is used the registration is good in the region of the curve, but extrapolates poorly over the rest of the object. In this case a single* $\mathcal{M}$ *curve is still sufficient for recognition as a* 68.9% *projected edge match was found.*

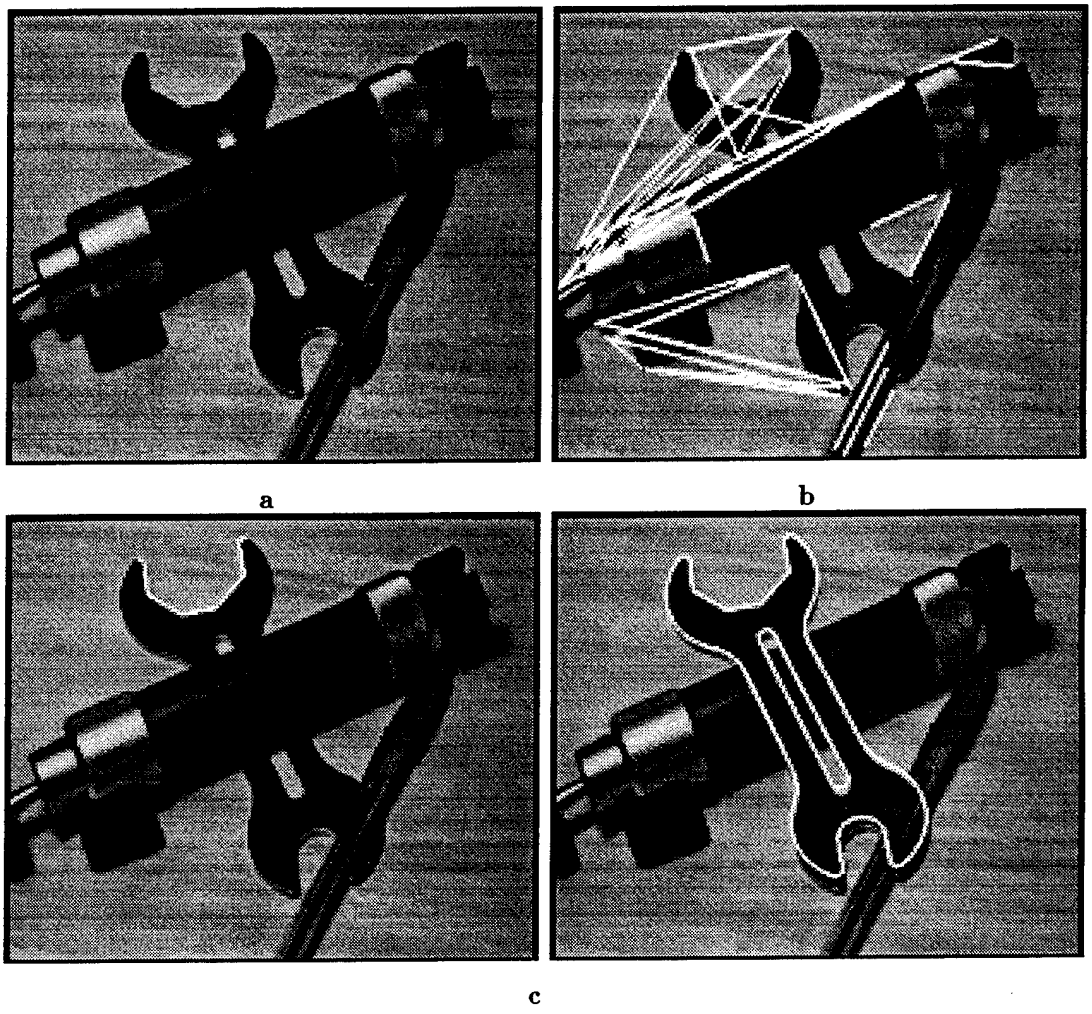


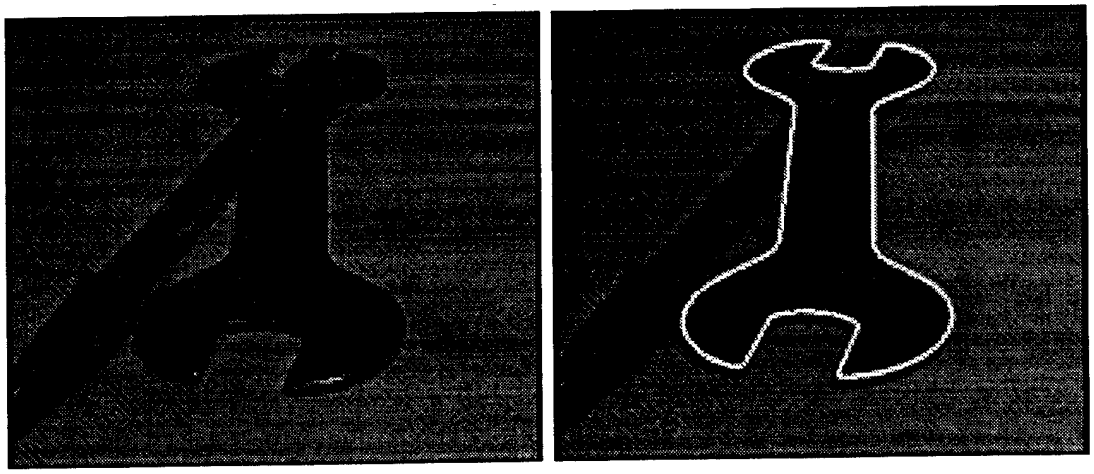
Figure 32: *For the occluded and cluttered view of the spanner (a), there are a large number of bitangents, (b). Note that bitangents are computed only along single continuous edgel chains and not between distinct curves. This further ensures a linear grouping cost. After $\mathcal{M}$ curve formation and indexing, a total of 16 potential class matches were found. However, the one that correctly identifies model 3, marked in (c), was the only one that produced a sufficiently high verification score (55.5%) to be accepted.*

3.5.1 Complexity Model

A major concern with the effectiveness of an indexing function is the probability that an image measurement taken from background clutter actually indexes a model. Often, it is suggested that the number of clashes produced within the hash table is important, but this is not the case. The hash table is simply an implementation of the index space, and should be designed so that only objects with matching image measurements are returned rather than those having only matching hash key values[7].

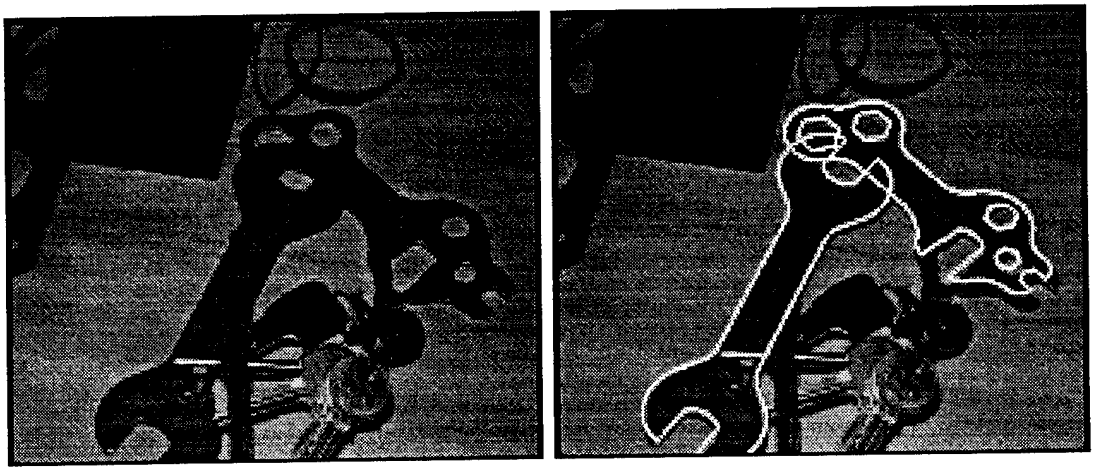
Here an informal argument is given that determines the likelihood that a random measurement will index an actual model; it shows that the indexing paradigm is (non-asymptotically) constant time, or at least can be made so with judicious use of the indexes. Consider a measure for a set of features that forms an n

[7]The function mapping index values onto hash keys is many-to-one.



a b

Figure 33: *Even under severe perspective distortion the recognition system performs well and finds model 2 with* 83.6% *confidence. Note that an affine description, such as the footprints in [Lamdan88], would fail in this case.*



a b

Figure 34: *Single classes are sufficient to recognise the two model instances shown in (b). The redundancy of the canonical frame representation gives much better tolerance to occlusion than global shape methods. The left hand object gained* 67.1% *boundary support, and the right object* 81.6%.

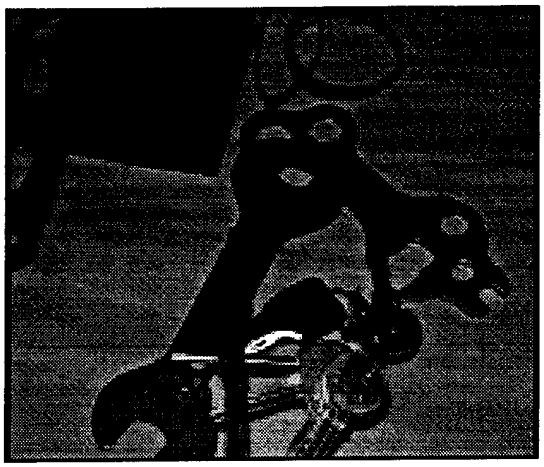
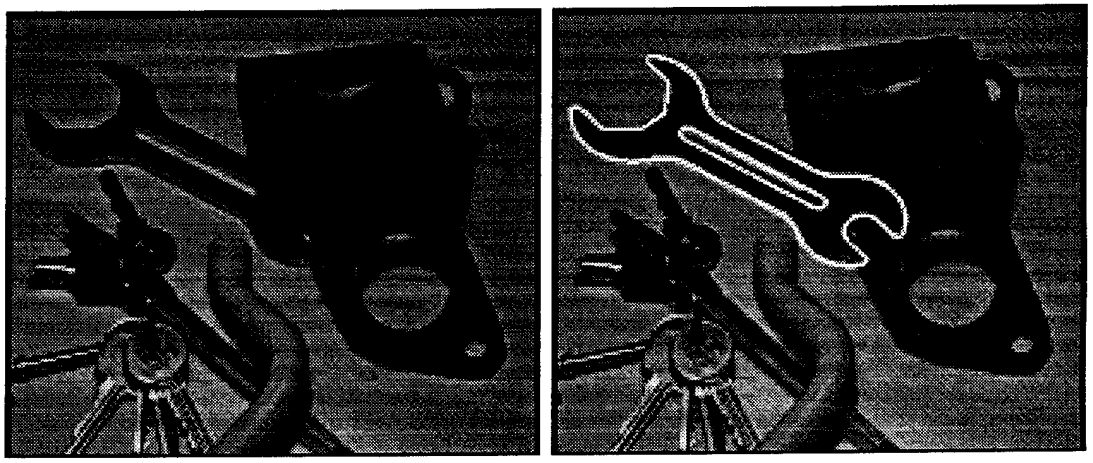


Figure 35: *In figure 34 incorrect objects were identified. Here, model 0 receives* 77.9% *edge match, but has a total projected object width of 24.7 pixels, which is too small to be a reasonable object projection.*



a b

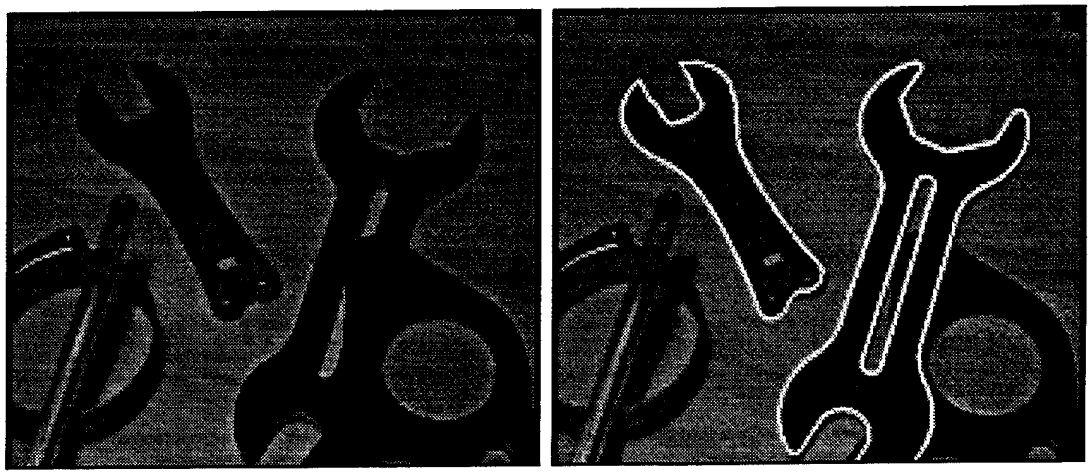
Figure 36: *Two classes are recognised and joined into a single joint hypothesis to recognise model 3 with* 68.0% *edge support.*

dimensional index; assume that each dimension has the same behaviour. Let each index cover a segment on the real line from i_0 to $i_0 + L$ (figure 39), and the quantisation along the line be δ, a constant quantity over the line segment[8]. There are $b = L/\delta$ buckets along the line, and so for n indexes and *assuming* that the measured invariants have a constant PDF over the invariant space[9], the probability of hitting any cell at random is $1/b^n$. If there are λ models in the library, each with α shape descriptors, and each invariant can be measured up to an error of $\pm\delta\epsilon/2$, $\epsilon \in \mathcal{N}$ (the set of natural numbers), there will be $\alpha\epsilon\lambda$ entries in the table[10]. If it is assumed that these entries are spread uniformly over the hash table, the chances of indexing

[8] More exactly a logarithmic scale should be used as the errors in invariant indexes tend to be proportional to the invariant values [Forsyth91].

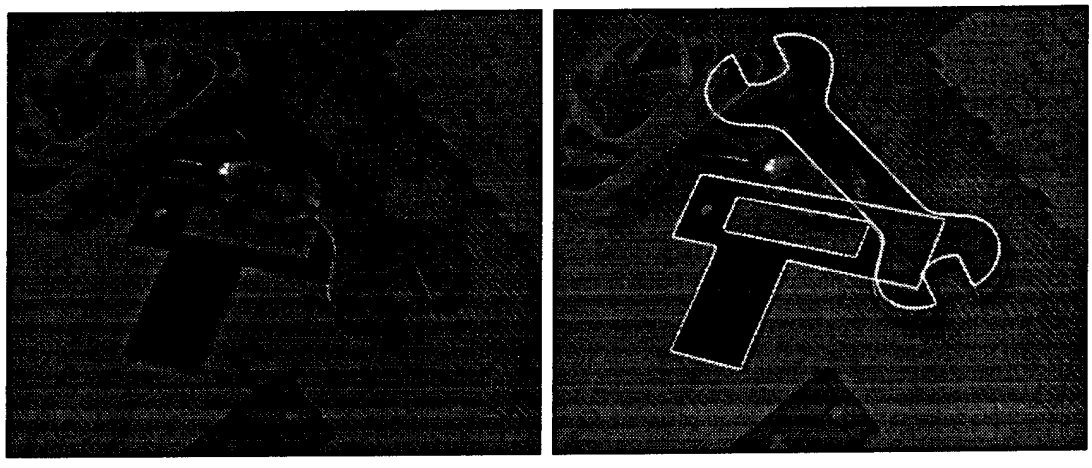
[9] This claim is a current topic of research, and should be compared to the work of Hopcroft, *et al.* [Mundy92] and Maybank [Maybank93].

[10] For efficiency reasons during recognition only a single cell will be read. Models are not stored in single cells, but in as many as defined by the range $\delta\epsilon$ which is the expected measurement error. This contrasts with storing models in single cells and then



a b

Figure 37: *Both models 1 (91.4% support) and 3 (75.7%) are correctly recognised and projected into the image as shown in (b).*



a b

Figure 38: *A demonstration that both types of invariant index can be used to recognise objects in a single image (by applying the invariant constructions independently within LEWIS). The bracket is indexed using algebraic invariants and the spanner is indexed using the canonical frame signature.*

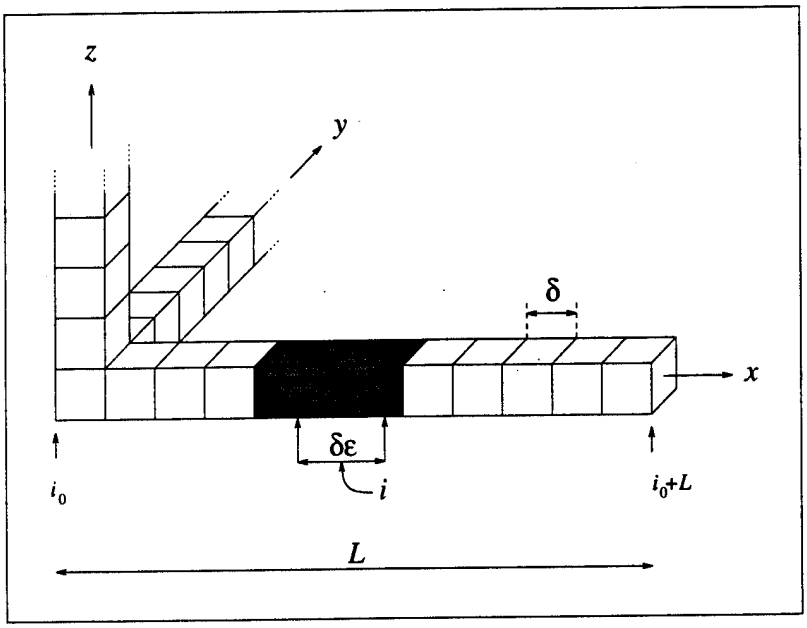


Figure 39: *Although the index space is multidimensional, it can loosely be assumed to be isotropic in all directions. Considering only the x direction: the index space ranges from i_0 to i_0+L, with bucket quantisation of size δ. On measurement of an index i in a scene, all the buckets within a range $\pm\delta\epsilon/2$ must be searched in the index space. These cells are shaded grey.*

a model through noise is $(\alpha\epsilon\lambda)/b^n$.

This analysis means that there is an algorithmic complexity of $O(k_1 + k_2\alpha\epsilon\lambda/b^n)$, where k_1 is the cost of edge detection, feature extraction and grouping (essentially constant), and k_2 another constant dependent on the form of the invariants, etc. It can be seen immediately that by making n large, the term dependent on the number of models λ, becomes arbitrarily small, and so recognition time tends towards k_1, a constant.

There are two problems associated with making n large:

1. For algebraic invariants there is little control over n. If a minimal feature group is used there is no control, but by using larger structures n can be increased. However, the grouping task may then become harder. Alternatively one could index using less discriminatory invariants and then group using results of this first indexing stage before forming higher order invariants and indexing a second time. For the invariants of other structures, such as canonical frame invariants, n can be made large (subject to the noise present in the curve).

2. Making n too large means that the problem of constructing an efficient hashing function must be considered.

During the development of LEWIS [Rothwell94] it was found that an invariant composed of a conic and two lines gave insufficient discrimination between objects. However, as an example of the above argument, when

indexing over a range.

an extra line was used to make $n = 3$ rather than $n = 1$ the invariant increased in utility. Because of the grouping heuristics used in the system there was no loss of efficiency but rather a marked improvement in performance.

3.5.2 Empirical Assessment

The indexing technique computes a number of invariants that is entirely dependent on the number of image features, though only a few of these will be turned into hypotheses on indexing. Indexing dramatically reduces the time taken for the entire recognition process. It was argued above that there should be a small linear growth in the number of hypotheses created as the size of the model base grows.

The linear growth is demonstrated in figure 40. The graph shows data collected over fifty evaluations of the recognition system in which a single model from the model base was placed in a scene and partially occluded by other objects that are not in the model base. Other non-library objects were also placed in the scene as clutter; figure 26 shows a typical scene. The average number of hypotheses computed as more objects were added to the library is plotted. The first model added to the library always corresponded to the actual model in the scene. Although 15.8% of the hypotheses were for the correct model (this is for when a total of 33 objects were present in the library), as predicted by the theory, the shape of the graph is predominately linear. The real benefit of indexing becomes apparent when one considers how many hypotheses would be produced if an alignment technique were used (maintaining the same grouping methods). On average, over 2000 feature groups existed for each image, and so 2000 hypotheses would be produced for each model feature group in the library (generally there are four or five feature groups per object and so the situation would be far worse). This would result in about 7×10^4 hypotheses for the entire model base compared to less than the 60 produced when indexing is used. As these all have to be verified it is clear that indexing produces a dramatic improvement in the system efficiency.

4 Discussion

We have shown how the use of invariants as index functions avoids search at two stages of the recognition process. First, indexes generate hypotheses which give direct access to models, avoiding a search through the model library. Second, at the hypothesis combination stage, invariants of the geometric relationships between feature groups, for instance a pair of $\mathcal{M}$ curves, permit the efficient construction of extended feature groups.

4.1 Verification

The final stage of recognition in most model-based systems [Huttenlocher87, Lowe87] is to verify model-to-image hypotheses. In the system described here, this is a layered process: first determine if there is

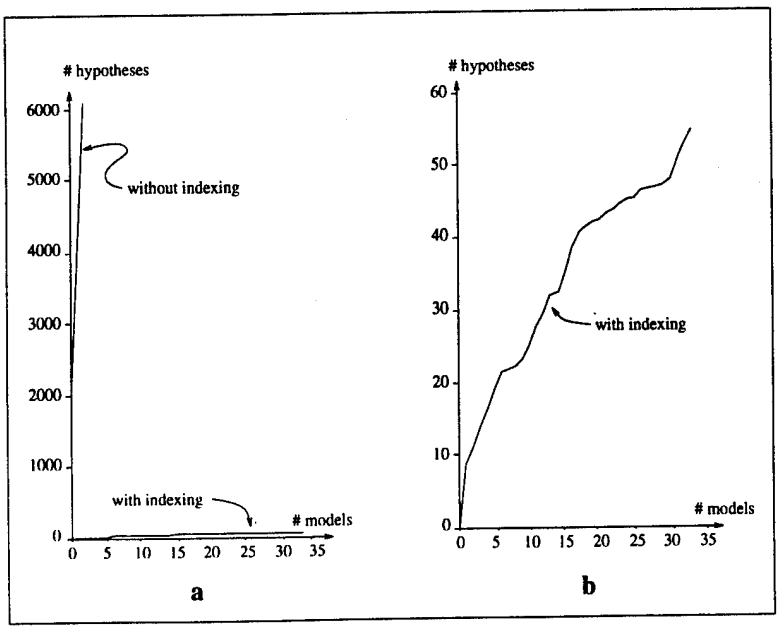


Figure 40: *(a) the number of hypotheses that have to be verified varies with the number of models from the model base. The results show an average over fifty scenes containing only one object in the model base, but with other clutter and occlusion present. Over 2000 indexes are created for the scene, which corresponds to the number of hypotheses that would have to be verified* **per model feature group** *if an alignment paradigm were used. Therefore, there is a rapid linear growth in the number of hypotheses created as the model base is expanded. However, the number of hypotheses created through indexing remains substantially lower; the detail depicted in (b) demonstrates that approximately a low constant of proportionality linear growth is observed. This ties in with the theoretical prediction of section 3.5.1.*

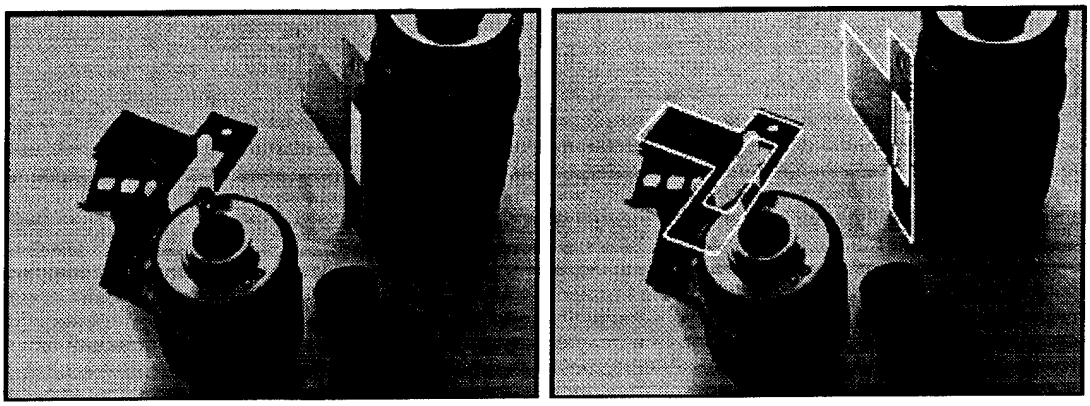
a common projective transformation for all geometric components (lines, conics, $\mathcal{M}$ curves) of the joint hypotheses. Second, back project geometric features and measure image support.

This strategy can fail, generally as a false positive, for two principal reasons. First, only projective geometric structure is used and many object boundary shapes are equivalent up to a projective transformation. In order to discriminate further, it is necessary to assume viewing conditions where an affine transformation is valid, or to use a calibrated camera which enables scaled Euclidean reconstruction. The second type of failure is associated with incomplete image support, which is discussed in more detail in the next section. Examples of both these failure cases are given in figures 42 and 43.

4.1.1 Image Support

Hypothesis validation based on image support is faced with two opposite failure mechanisms: too little support; or too much. When the object boundary exhibits little image contrast, a significant fraction of the achievable perimeter is unrecoverable by edge detection algorithms. On the other hand, when the background is highly textured or cluttered, high support can be achieved for an incorrect hypothesis. Two examples of the latter failure mechanism is shown in figure 45.

Both of these problems are symptomatic of having too sparse a description for the object. Thus far we



a b

Figure 41: *Two objects from the model base are recognised correctly despite strong perspective distortion.*

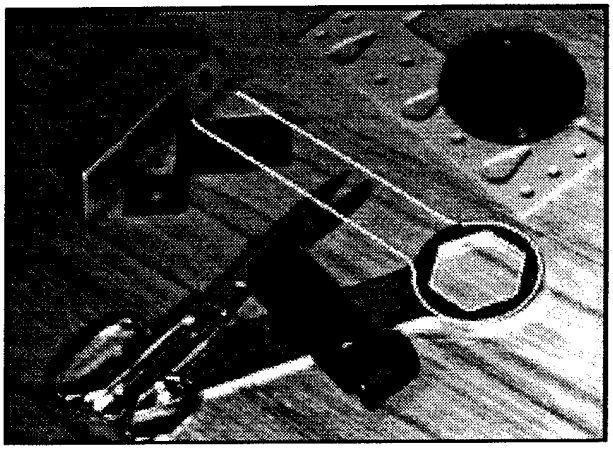
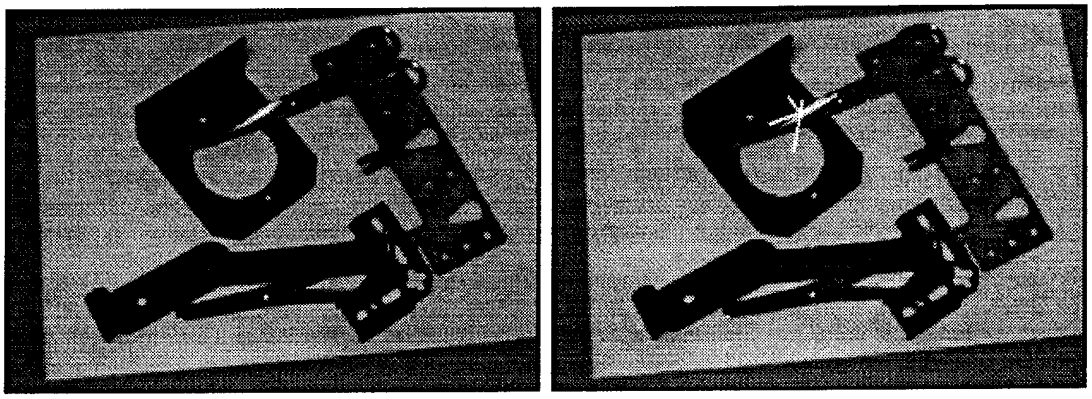
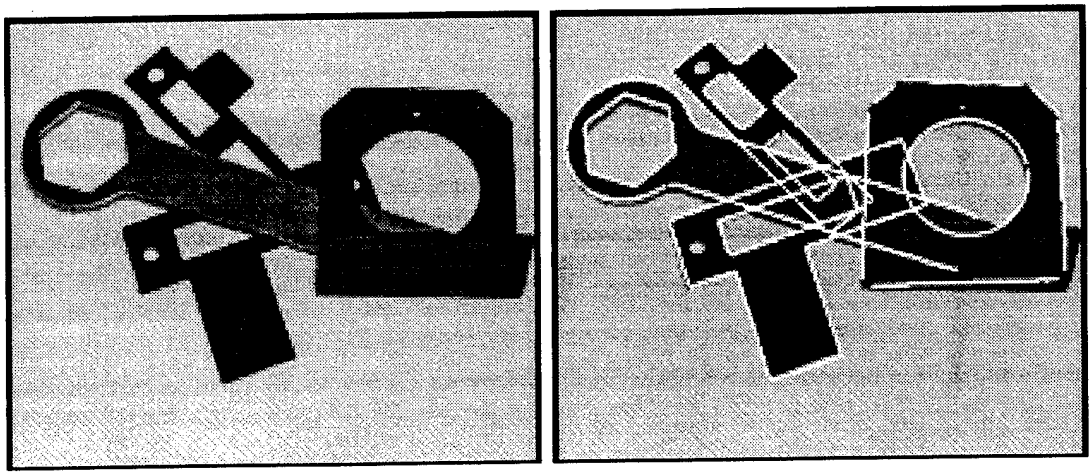


Figure 42: *The spanner from figure 27 is shown and recognised, but with the wrong orientation; due to texture in the image a* 52.1% *edge match is still found.*



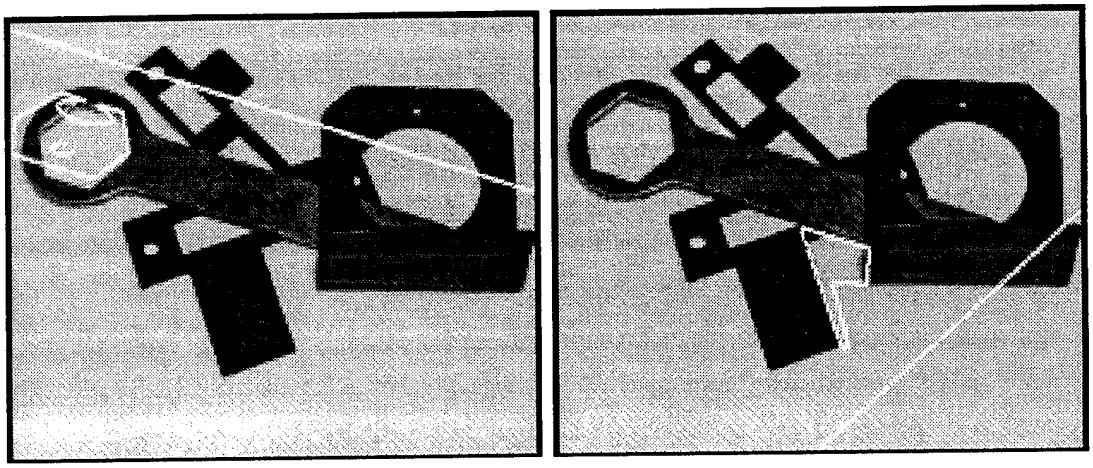
a b

Figure 43: *An object from the model base which is superimposed in (b) can be recognised with over* 50% *edge support from the specularity on the pair of scissors in (a).*



a b

Figure 44: *Six objects were recognised from image (a). The four correct matches are shown in (b), with the two incorrect given in figure 45. The worst of the four correct identifications had two invariants and* 60.3% *match.*



a b

Figure 45: *The two incorrectly recognised objects from the image in figure 44(a). Unlike the matches shown in figure 44(b) these two objects were hypothesised by only a single invariant, and had less than* 52% *image support. In both cases image support is provided by features that have already been used to verify hypotheses in figure 44(b). The straight line across each image is the projection of the line at infinity from the acquisition images. Its closeness to the image center indicates an unlikely object pose (see the discussion in section 4.1.2 for details).*

have relied just on the boundary curve of the object and have ignored any properties of the interior region. It is certainly reasonable to use our knowledge of the model coordinate frame in the image to extract viewpoint independent texture measures. Even very simple measures would have eliminated the false positive shown in figure 42. It is also possible that very simple intensity measures on the internal object surface can be used. For example, the ratio of intensities in the neighbourhood of step discontinuities is a reliable measure of albedo ratio [Nayar93]. Even very weak intensity measures for discrimination can be used to increase confidence in a hypothesis or to break ties between two very similar geometric configurations.

There is also the open question of whether a feature should be used to support a hypothesis when it has already been used in a confirmed hypothesis, as illustrated in figure 45.

When insufficient image support is found it will be necessary to invoke additional understanding of scene of the form "this object is on top of another, and therefore occludes it". An understanding of this kind can be used to guide a search for further object features which support the explanation for missing features in the first object.

4.1.2 Projective Transformation

One limitation of the full projective transformation is that unreasonable perspective projections are allowed. An example is shown in figure 43 where the entire object is backprojected onto a thin specularity. To eliminate such projections, we propose a stratified solution involving progressively more knowledge of affine followed by Euclidean structure and ultimately, full perspective camera calibration.

1. **Real cameras**

   For a physical camera the object must lie in front of the image plane. More precisely the object plane cannot intersect the focal plane (a plane parallel to the image plane, containing the optical center). This is captured by projecting the ideal line, that is the line at infinity, of the model image onto the target image. Any case in which the ideal line passes through the convex hull of the hypothesised image features can be ruled out, since objects are assumed to be finite. More generally, if the model ideal line is observed within the finite bounds of the image plane, the object pose must be sufficiently extreme that the hypothesis can be rejected. In our experiments, about 25% of false positives due to poor poses ruled out by this constraint (see figure 45 and [Rothwell94] for details).

2. **Similarity structure**

   If the acquisition view is taken with the object in a fronto-parallel plane, one can calculate slant and tilt of the plane of the object in the target image. This pose calculation does not require full calibration of internal camera parameters. In the perspective case, the computation does not require focal length and in the affine case only the image pixel aspect ratio is required.

3. **Size**

   Two additional calibration parameters are essential for the computation of size. The first is distance

from the camera. In order to estimate this distance, an approximate knowledge of object area, a Euclidean measure, and focal length are required. The second calibration parameter is the physical size of pixels on the image plane.

4.2 Future Work

1. For non-algebraic curves there are other invariants available which do not require $\mathcal{M}$ curves. For example, Van Gool *et al* [VanGool91] exploit single inflections as distinguished points; Carlsson [Carlsson92], fits conics tangent to the curve at four points. The latter procedure is applicable even to convex curves. There are numerous other covariant constructions (for example tangents between two curve segments) that can be utilised to generate distinguished points and hence invariants. The natural stage for integrating the various categories of invariant is at hypotheses combination. Again joint invariants between features involved in more global invariant groupings can efficiently be used to build larger model hypotheses. This integration strategy is currently under investigation [Rothwell93b].

2. Feature grouping based on sequential connectivity is a somewhat fragile process. It is easy to encounter large gaps in the object boundary due to low image contrast and occlusion. Any recognition algorithm will be adversely affected by these effects, however it is impossible to recover from when the index is constructed based solely on the assumption of boundary connectivity. An immediate way to overcome this problem is to use as many feature groups as possible for a given object to derive a redundant description, however many object shapes do not have sufficient complexity to define more than a few independent feature groups.

   Current work is investigating how grouping can be improved for applications where segmentation provides poor boundary connectivity, for instance in aerial reconnaissance scenes. The primary grouping relations are proximity and collinearity. By constructing a Delaunay triangulation of the set of line segment endpoints, it is computationally feasible to establish line segment sequences which are not actually connected topologically. Similarly, line segments which are reasonably close and collinear can also be grouped efficiently.

3. We observe that a useful goal for image feature segmentation and grouping is to provide feature groups which support invariant computations, for example the algebraic curves and $\mathcal{M}$ curves used in the current system. As a consequence, the evolution and testing of new segmentation and grouping algorithms can be tested by an evaluation of the accuracy and stability of resulting invariants. Additionally, the discovery of new invariant constructions will require the development of associated feature extraction algorithms. Since we know that the robustness of recognition is largely dependent on the success of such group constructions, we can profitably focus research on this stage of the system.

4. We have demonstrated a recognition complexity of low gradient linear growth with the size of the model base, and developed a statistical model of this performance. These results are still preliminary,

firstly because the model base is still relatively small (less that 50 models), and secondly because the objects are fairly similar. It is an open question as to whether this is simply clustering behaviour and if for a large model base (several thousand objects), recognition would remain asymptotically constant time.

5. A number of recent papers have demonstrated that invariants of 3D structures, under 3D projective transformations, can be extracted from image projections of the structure. These can be obtained from multiple views [Demey92, Faugeras92, Hartley92, Koenderink91, Mundy92, Quan91] or from a single view [Forsyth92, Forsyth93, Liu93, Rothwell93a, Rothwell94, Wayner91].

   We propose to employ our experiences with LEWIS by building an improved recognition system called MORSE (Multiple Object Recognition by Scene Entailment) for 3D structures. To recognise such structures an improved architecture is required. For example: first, we require interactions between different types of invariant working simultaneously in a single image; second, fine-grained communication loops are required between the different processing layers than provided within the current grouping-indexing-correspondence architecture. These must be implemented in such a way as to ensure that the implications of each local conclusion are understood by all other layers. Furthermore, multiple representations of objects must be allowed by the model library. For instance, curve g on the spanner in figure 29 could be represented both as a concavity curve, and as a five line sequence. Both representations should be included in the model library.

Acknowledgements

We acknowledge discussions with various members of the Oxford University Robotics Reseach Group, especially Paul Beardsley, Mike Brady, Ian Reid, Mike Taylor and David Sinclair. CAR and JLM were supported by General Electric working under the following contracts: ARPA grant MDA97291C0053 and a grant from the United States Air Force Office of Scientific Research, AFOSR-91-0361. DAF was supported in part by the National Science Foundation under award no. IRI-9209729, and in part by a National Science Foundation Young Investigator Award with matching funds from GE, Eugene Rikel, Rockwell International and Tektronix. AZ acknowledges support of Esprit Basic Research Action 6448 VIVA. The final version of the text was improved through the reviewers' comments.

References

[Ayache86] Ayache, N. and Faugeras, O.D. 1986. HYPER: a new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-8(1), 44-54.

[Ayache87] Ayache, N. and Faugeras, O.D. 1987. Building a consistent 3D representation of a mobile robot environment by combining multiple stereo views. In *Proc.* IJCAI, pp.808-810.

[Binford81] Binford, T.O. 1981. Inferring surfaces from images. *Artificial Intelligence* 17, 205-244.

[Binford93] Binford, T.O. and Levitt, T.S. 1993. Quasi-invariants: theory and explanation. In *Proc.* DARPA IUW, pp.819-829.

[Bolles87] Bolles, R.C. and Horaud, R. 1987. 3DPO: a three-dimensional part orientation system. In *Three Dimensional Vision.* Kanade, T. editor, Kluwer Academic Publishers, 399-450.

[Bookstein79] Bookstein, F. 1979. Fitting conic sections to scattered data. *Computer Vision Graphics and Image Processing* CVGIP-9, 56-71.

[Borgefors88] Borgefors, G. 1988. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-10(6), 849-865.

[Brooks83] Brooks, R.A. 1983. Model-based three-dimensional interpretations of two-dimensional images. *Pattern Analysis and Machine Intelligence* PAMI-5(2).

[Califano92] Califano, A. and Mohan, R. 1992. Multidimensional indexing for recognizing visual shapes. *Visual Form*, 109-118.

[Canny86] Canny J.F. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-8(6), 679-698.

[Carlsson92] Carlsson, S. 1992. Projectively invariant decomposition of planar shapes. In *Geometric Invariance in Computer Vision.* Mundy, J.L. and Zisserman, A.P. editors, MIT Press.

[Cass92] Cass, T.A. 1992. Polynomial-time object recognition in the presence of clutter, occlusion, and uncertainty. In *Proc.* ECCV, pp.834-842.

[Clemens91] Clemens, D.T. and Jacobs, D.W. 1991. Model group indexing for recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-13(10), 1007-1017.

[Cox92] Cox, I.J., Rehg, J.M. and Hingorani, S. 1992. A Bayesian multiple hypothesis approach to contour grouping. In *Proc.* ECCV, pp.72-77.

[Demey92] Demey, S., Zisserman, A. and Beardsley, P. 1992. Affine and projective structure from motion. In *Proc.* BMVC, pp.49-58.

[Duda73] Duda, R.O. and Hart P.E. 1973. *Pattern Classification and Scene Analysis.* Wiley.

[Ettinger88] Ettinger, G.J. 1988. Large hierarchical object recognition using libraries of parameterized model sub-parts. In *Proc.* CVPR, pp.32-41.

[Faugeras92] Faugeras, O. 1992. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc.* ECCV, pp.563-578.

[Forsyth91] Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. 1991. Invariant descriptors for 3-D object recognition and pose. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-13(10), 971-991.

[Forsyth92] Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. 1992. Recognising curved surfaces from their outlines. In *Proc.* ECCV, pp.639-648.

[Forsyth93] Forsyth, D.A. 1993. Recognizing algebraic surfaces from their outlines. In *Proc.* ICCV, pp.476-480, 1993.

[Forsyth94] Forsyth, D.A., Mundy, J.L., Rothwell, C.A. and Zisserman, A.P. 1994. MORSE: Multiple Object Recognition by Scene Entailment. GE Corporate Research and Development, NY, document in preparation.

[Fisher89] Fisher, R.B. 1989. *From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis.* John Wiley and Sons.

[Goad83] Goad, C. 1983. Special purpose automatic programming for 3D model-based vision. In *Proc.* DARPA IUW, pp.371-381.

[Grimson87] Grimson, W.E.L. and Lozano-Pérez, T. 1987. localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-9(4), 469-482.

[Grimson90] Grimson, W.E.L. 1990. *Object Recognition by Computer, The Role of Geometric Constraints.* MIT Press.

[Gueziec93] Gueziec, A. and Ayache, N. 1993. New developments on geometric hashing for curve matching. In *Proc.* CVPR, pp.703-704.

[Hartley92] Hartley, R.I., Gupta, R. and Chang, T. 1992. Stereo from uncalibrated cameras. In *Proc.* CVPR, pp.761-764.

[Huttenlocher87] Huttenlocher, D.P. and Ullman, S. 1987. Object recognition using alignment. In *Proc.* ICCV, pp.102-111.

[Huttenlocher88] Huttenlocher, D.P. 1988. Three-dimensional recognition of solid objects from a two-dimensional image. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT.

[Huttenlocher91] Huttenlocher D.P. 1991. Fast affine point matching: an output-sensitive method. In *Proc.* CVPR, pp.263-268.

[Jacobs92] Jacobs, D.W. 1992. Space efficient 3D model indexing. In *Proc.* CVPR, pp.439-444.

[Kalvin86] Kalvin, A., Schonberg, E., Schwartz, J.T. and Sharir, M. 1986. Two-dimensional, model-based, boundary matching using footprints. *International Journal of Robotics Research* IJRR-5(4), 38-55.

[Koenderink91] Koenderink, J.J. and Van Doorn, A.J. 1991. Affine structure from motion. *J. Opt. Soc. Am. A.* 8(2), 377-385.

[Lamdan88] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. 1988. Object recognition by affine invariant matching. In *Proc.* CVPR, pp.335-344.

[Lowe85] Lowe, D.G. 1985. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers.

[Lowe87] Lowe, D.G. 1987. The viewpoint consistency constraint. *International Journal of Computer Vision* IJCV-1(1), 57-72.

[Liu93] Liu, J., Mundy, J.L., Forsyth, D.A., Zisserman, A. and Rothwell, C.A. 1993. Efficient recognition of rotationally symmetric surfaces and straight homogeneous generalized cylinders. In *Proc.* CVPR, pp.123-128.

[Marr82] Marr, D. 1982. *Vision.* Freeman.

[Maybank93] Maybank, S.J. 1993. Probabilistic analysis of the application of the cross ratio to model based vision. GEC Hirst Research Centre, Technical Report.

[Mundy92] Mundy, J.L. and Zisserman, A.P. 1992. *Geometric Invariance in Computer Vision.* MIT Press.

[Murray87] Murray, D.W. 1987. Model-based recognition using 3D structure from motion. *Image and Vision Computing* IVC-5, 85-90.

[Nayar93] Nayar, S.K. and Bolle, R.M. 1993. Reflectance ratio: a photometric invariant for object recognition. In *Proc.* ICCV, pp.280-285.

[Nielsen88] Nielsen, L. 1988. Automated guidance of vehicles using vision and projective invariant marking. *Automatica* 24, 135-148.

[Pollard89] Pollard, S.B, Pridmore, T.P, Porrill, J., Mayhew, J.E.W. and Frisby, J.P. 1989. Geometrical modeling from multiple stereo views. *International Journal of Robotics Research* IJRR-8(4), 132-138.

[Quan91] Quan, L., Gros, P. and Mohr, R. 1991. Invariants of a pair of conics revisited. In *Proc.* BMVC, pp.71-77.

[Reid91] Reid, I. 1991. Recognising parameterized models from range data. D.Phil. Thesis, Department of Engineering Science, Oxford University, Oxford.

[Rigoutsos91] Rigoutsos, I. and Hummel, R. 1991. Implementation of geometric hashing on the connection machine. In *Proc.* IEEE Workshop on Directions in Automated CAD-Based Vision, pp.76-84.

[Rothwell93a] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L. 1993. Extracting projective information from single views of 3D point sets. In *Proc.* ICCV, pp.573-582.

[Rothwell93b] Rothwell, C.A. 1993. Hierarchical object descriptions using invariants. In *Proc.* 2^{nd} ARPA/NSF-ESPRIT Workshop on Invariance, pp.287-302.

[Rothwell94] Rothwell, C.A. 1994. *Object Recognition through Invariant Indexing.* To appear OUP.

[Schwartz87] Schwartz, J.T. and Sharir, M. 1987. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *International Journal of Robotics Research* IJRR-6(2), 29-44.

[Semple52] Semple, J.G. and Kneebone, G.T. 1952. *Algebraic Projective Geometry.* OUP.

[Shaashua88] Sha'ashua, A. and Ullman, S. 1988. Structural saliency: the detection of globally salient structures using a locally connected network. In *Proc.* ICCV, pp.321-327.

[Sinclair93] Sinclair, D.A., Blake, A., Smith, S. and Rothwell, C.A. 1993. Planar region detection and motion recovery. *Image and Vision Computing* IVC-11(4), 229-234.

[Slama80] Slama, C.C. 1980. *Manual of Photogrammetry.* American Society of Photogrammetry, 4^{th} edition.

[Stein92] Stein, F. and Medioni, G. 1992. Structural indexing: efficient 2-D object recognition. *Pattern Analysis and Machine Intelligence* PAMI-14, 1198-1204.

[Stockman87] Stockman, G. 1987 Object recognition and localization via pose clustering. *Computer Vision Graphics and Image Processing* CVGIP-40, 361-387.

[Taubin91] Taubin, G. and Cooper, D.B. 1991. Object recognition based on moment (or algebraic) invariants. *IBM TR-RC17387* IBM T.J. Watson Research Centre, P.O. Box 704, Yorktown Heights, NY 10598.

[Thompson87] Thompson, D.W. and Mundy, J.L. 1987. Three-dimensional model matching from an unconstrained viewpoint. In *Proc.* ICRA, pp.208-220.

[VanGool91] Van Gool, L. Kempenaers, P. and Oosterlinck, A. 1991. Recognition and semi-differential invariants. In *Proc.* CVPR, pp.454-460.

[Wayner91] Wayner, P.C. 1991. Efficiently using invariant theory for model-based matching. In *Proc.* CVPR, pp.473-478.

[Weiss88] Weiss, I. 1988. Projective invariants of shapes. In *Proc.* DARPA IUW, pp.1125-1134.

[Wolfson92] Wolfson, H.J. 1992. Object recognition by transformation invariant indexing. In *Proc.* Invariance Workshop, ECCV.

Invariants of Three Dimensional Point Sets from Single Images

Charlie Rothwell
INRIA
2004, Route des Lucioles
BP-93, Sophia Antipolis
06902 CEDEX, France

David Forsyth
Computer Science Division
University of California at Berkeley
Berkeley, CA 94702 USA

Andrew Zisserman
Robotics Research Group
Department of Engineering Science
University of Oxford
Parks Road, Oxford, UK

Joe Mundy
GE Corporate Research and Development
1, River Road
Schenectady, 12301
New York, USA

tel: +33.93.65.77.32
fax: +33.93.65.78.45
email: crothwel@sophia.inria.fr

Abstract

A number of recent papers have argued that invariants of three dimensional point sets in general position cannot be measured from a single image [5, 7, 18, 28]. This has often been misinterpreted to mean that invariants cannot be computed for any three dimensional structure. This paper proves by example, that invariants can be measured for *constrained* three dimensional point sets from a single image.

Projective invariants are derived for two classes of object: the first is for points that lie on the vertices of polyhedra, and the second for objects possessing a bilateral symmetry. These invariants can be used both for recognition and to compute projective structure. Examples are given of invariants computed from real scenes imaged with an uncalibrated camera.

Contents

1 Introduction 3
- 1.1 Geometric invariants . . . 4
- 1.2 The limitations of point sets . . . 4
- 1.3 Invariant computation . . . 5

2 Polyhedral cages 5
- 2.1 Reconstruction of polyhedra from single images . . . 5
- 2.2 Forming the constraint equations . . . 6
- 2.3 Interpreting the kernel . . . 8
  - 2.3.1 Position free polyhedra . . . 8
- 2.4 Indexing for polyhedra . . . 10
- 2.5 Examples . . . 10
- 2.6 Verifying the polyhedral assumption . . . 11
  - 2.6.1 Algebraic approach . . . 14
  - 2.6.2 Geometric approach . . . 14

3 Objects with bilateral symmetry 14
- 3.1 Correspondence . . . 16
- 3.2 Planar invariants - geometric method . . . 16
- 3.3 3D invariants - analytic method . . . 19
- 3.4 Verifying the bilateral symmetry assumption . . . 21
- 3.5 Further examples . . . 23

4 Discussion 26
- 4.1 Caged point sets . . . 26
- 4.2 Objects with symmetry . . . 26

A Projective equivalence of all the kernel solutions 27

1 Introduction

Recognizing *planar* objects from a single, uncalibrated, perspective image obtained from an unknown viewing position is a problem that naturally suggests projective invariants; the transformation from object to image is an unknown projective transformation, and the use of invariants eliminates the effect of that transformation. Projective invariants have consequently had a substantial impact on model based object recognition by providing useful examples of *indexing functions*, which facilitate rapid, viewpoint invariant, access to models in model libraries. Indeed, a plane object recognition system using only projective invariants as indexes has been constructed [34, 35, 37]. Recognition has been demonstrated for a model library of over forty objects, with the system tested on hundreds of images, under varying levels of occlusion and clutter.

However, until quite recently there was a notable absence of index functions for *three dimensional* objects. This is primarily because most research on the application of invariance to computer vision (e.g. [1, 12, 20, 27, 41, 43, 45, 47, 46]) has concentrated on planar objects from a single view, and three dimensional objects from two views or range data. Furthermore, a number of recent papers have argued that invariants can not be measured for a 3D set of points in *general position* from a single view [5, 7, 18, 28]. This has often been misinterpreted to mean that *no* invariants can be measured for three dimensional objects from a single image, and consequently that the use of invariants as indexes is confined to planar objects. In contrast, we demonstrate that if points are not in general position, but have *constrained 3D positions*, then invariants are available for use as index functions.

In this paper we describe two examples of constrained (or structured) 3D point sets. In each case the assumed constraint can be verified to a certain extent from image measurements. The two cases are:

1. **Polyhedral cages:** The points lie at the vertices of a virtual polyhedron or are the vertices of a real polyhedron. Allowing virtual polyhedra means that the objects we recognize need not necessarily be polyhedral. The polyhedral class we examine have only trihedral vertices, and the points lie on shapes described by six planes. A minimum of 7 points are required in order to recover invariants. This is described in Section 2. In general each plane of the polyhedron contains only four points. This demonstrates that the invariants extracted are of the 3D object, since five planar points are required for a projective invariant.

2. **Bilateral symmetry:** The point set has a bilateral symmetry. A minimum of eight distinct points (four symmetrically matched pairs) are required. We demonstrate in Section 3 that the full 3D projective structure can be recovered from a single uncalibrated perspective image. This reconstruction method should be compared to the ones presented in [14, 26] which exploit symmetry in a similar manner to determine shape under perspectivity, but do so with a *calibrated* camera.

As in the planar case, index functions based on invariants can be used in a recognition system to provide direct access to a model library, irrespective of the object pose and intrinsic camera parameters.

Even in the case of general point sets, indexing strategies are possible, though they are less efficient. For example, under weak perspective, indexing functions can be defined that return values restricted to a line, rather than a point [7, 19]. The modelbase therefore becomes a space filled with lines representing each model, where the lines have a finite thickness to allow for measurement error. In contrast, in the invariant indexing case each model is represented by points and an associated error ball. Clearly the false-positive rate

of the indexing stage (that an arbitrary value should by chance index a model) is necessarily higher when the space is filled with lines.

In the following we adopt the notation that corresponding entities in two different coordinate frames are distinguished by upper and lower case. In general, lower case is used for image quantities, and upper for 3D quantities. Vectors are written in bold font, for instance $\mathbf{x}$ and $\mathbf{X}$. Matrices are written in typewriter font, for example $\mathtt{c}$ and $\mathtt{C}$. With homogeneous quantities, equality is up to a non-zero scale factor.

1.1 Geometric invariants

Invariants are properties of geometric configurations which remain unchanged under an appropriate class of transformations [30]. Under a linear transformation of coordinates, $\mathbf{X}' = \mathtt{T}\mathbf{X}$, the invariant, $I(\mathbf{P})$, of a configuration $\mathbf{P}$ transforms as

$$I(\mathbf{P}') = |\mathtt{T}|^w I(\mathbf{P}),$$

and is called a *relative* invariant of weight w, where $\mathbf{P}'$ is the transformed configuration. If $w = 0$, the invariant is unchanged under transformations and is called a *scalar* invariant. We will only be interested in scalar invariants in this paper.

Here we will be concerned with *projective* transformations, so $\mathtt{T}$ is a general non-singular square matrix acting on homogeneous coordinates. For planar configurations it is 3×3, and for 3D configurations 4×4. Note, invariants are computed with respect to a *transformation*, which is a mapping between spaces of the same dimension.

We are interested here in measuring invariants of 3D point sets from a perspective *projection* of the configuration. For 3D objects the original and image spaces do not have the same dimension. However, invariants of the 3D configuration can be measured from their image projections for the cases discussed in this paper. We write $\mathtt{P}$ for the projection matrix that covers a 3D Euclidean transformation of the object followed by perspective projection onto the image. $\mathtt{P}$ is a 3×4 matrix mapping three-dimensional homogeneous coordinates $\mathbf{X}$, onto two-dimensional homogeneous coordinates of the image plane $\mathbf{x}$:

$$\mathbf{x} = \mathtt{P}\mathbf{X}. \tag{1}$$

1.2 The limitations of point sets

A set of 3D points is a very impoverished representation of shape. It contains the minimum description possible (position) but no connectivity, and lacks the richness of curves or surfaces. Furthermore, unrestricted point sets are not an important part of the visual experience. Far more significant are polyhedra, faceted surfaces, and smooth surfaces.

It is seldom the case that machine recognition systems must deal with wholly generic objects; more typically, objects are drawn from *classes*, which satisfy geometric constraints. For example, objects might be algebraic surfaces, generalised cylinders, polyhedral or symmetric. Constraining objects to belong to classes very often results in 3D invariants that can be measured from a single image. Some constraints are ineffective; for example, constraining an object to be a smooth surface does not yield global 3D invariants from the outline. However, a number of quite reasonable constraints result in invariants that can be measured in a single view. Known examples include algebraic surfaces [13], rotationally symmetric surfaces [23]; and canal surfaces [32]. This paper adds polyhedral objects and objects with a symmetry to this list.

In addition, the images of constrained three dimensional objects often have natural grouping mechanisms which are not available for general 3D points sets. For example, polyhedral

grouping involves constructing a line drawing, and grouping for images of objects with bilateral symmetry involves a simple line search (see Section 3.1).

1.3 Invariant computation

Recovering geometric invariants of 3D structures from images can proceed in two, quite distinct, ways [15]: these are *explicit reconstruction* and *implicit reconstruction.* These approaches are mathematically equivalent, in that they measure the same numbers, but are conceptually distinct, and may have very different noise properties. To appreciate the difference, consider a stereo pair taken with uncalibrated cameras, depicting a set of matched points. Given the matches, it is possible to reconstruct in space the collection of points, and then to measure its 3D projective invariants from that reconstruction. This is an explicit reconstruction. An alternative approach would involve writing the 3D invariants as functions of the coordinates in each separate image, and evaluating those functions directly; this is called implicit reconstruction. Each approach has its merits and its difficulties in terms of ease of application and sensitivity to measurement error. This paper emphasizes explicit reconstructions of polyhedra, but shows an implicit construction in Section 2.6.2; in the case of bilaterally symmetric objects, an implicit construction appears in Sections 3.2 and an explicit construction appears in Section 3.3.

2 Polyhedral cages

3D object vertices often lie on planes, which can be grouped to form a "virtual polyhedron" — see the examples at the end of this section, for example Fig. 3f. It may be that the object is actually polyhedral, but this is not necessary. The virtual polyhedron "cages" the 3D points. The route we take to forming the projective invariants of the 3D points, is by reconstructing the caging polyhedron in 3-space, and then measuring the projective invariants for the planes of the polyhedron. The following sections outline the extraction of 3D projective invariants for polyhedra from a single perspective image.

2.1 Reconstruction of polyhedra from single images

The reconstruction of polyhedra has roots in the origins of the field with the seminal work of Huffman [17] and independently by Clowes [8] on labeling schemes which allow qualitative interpretation of polyhedral scenes. This work was continued and advanced by Waltz [44] and Mackworth [25], yielding techniques that can determine the type of a polyhedral edge (concave or convex) and can parse junctions and shadow regions.

Sugihara [40] reframed reconstructing polyhedra from single images as an algebraic problem, and showed how to produce a parameterised 3D reconstruction from a single orthographic or (calibrated) perspective image. However, to reduce the ambiguity of the 3D reconstruction to scaled Euclidean, the geometric constraints were augmented with texture or shading information. Our contribution in this section is to demonstrate that the structure recovered from the algebraic formulation for a perspective image is within a 3D projective transformation of the actual Euclidean structure. Consequently, projective invariants of the recovered polyhedron have the same value as those of the actual polyhedron giving rise to the image.

For general polyhedra, viewing issues become important; in particular, self occlusion can lead to situations where the projective structure of the polyhedron can be recovered only in part. This problem bears some relationship to the aspect graph of the polyhedron (clearly,

views that recover different parts of the projective structure lie in different cells of the aspect graph), but has some crucial differences; the most significant is the fact that faces can be inferred by assuming trihedral junctions (an assumption used throughout this paper), so that occluded faces may appear in the solution. For example, all faces of a cube can be correctly recovered, even though at most only three are visible in any one view, as at each vertex that lies on the silhouette, the vertex and the two edges define the occluded plane, and the three occluded planes define the occluded vertex. For more complex polyhedra, complete reconstructions are not always possible; the problem is particularly severe when there are views such that for some faces, no element (edge or vertex) incident on those faces are visible.

2.2 Forming the constraint equations

In the derivation that follows, all polyhedron vertices are assumed to be *trihedral*. Trihedral junctions are *stable* because three planes generically meet in a single point; any one of the planes can be perturbed resulting in a single vertex close to the original vertex. This is not true for higher order junctions. This assumption has little effect on the mathematics presented here, the difference being that if a point lies on n (rather than 3) planes $n-1$ constraints can be formed rather than the 2 for the trihedral case.

A further assumption is that the polyhedra *represent* real solid objects, that is, a polyhedron is not just a polygon or a group of polygons embedded in 3-space. Finally, a genericity assumption will be necessary; the details of this assumption will be developed along with the mathematics below.

The derivation of our method is similar to that of Sugihara except that a perspective imaging model is used, and that the camera is uncalibrated. The input is a set of unknown planes that are bounded by an arbitrary number of edges (≥ 3), and a set of points (projected vertices) whose image locations are known. Using the trihedral assumption the points always lie on three given planes.

Without loss of generality, a coordinate frame is chosen so that the world X and Y axes lie in a plane parallel to the image plane with directions shown in Fig. 1; the origin is at the camera optical centre. The Z axis lies along the principal axis of the camera, and, again without loss of generality, the image plane is the plane $Z = 1$. The differences between the actual camera coordinates and this coordinate system will be taken up in the projective ambiguity below.

The j^{th} plane in the polyhedron has equation:

$$a_j X + b_j Y + c_j Z + 1 = 0. \tag{2}$$

Note that this assumes that no plane in the polyhedron passes through the optical centre, that is, the optical centre is in general position. The plane can then be represented as $\mathbf{v}_j = (a_j, b_j, c_j, 1)^T$ in homogeneous coordinates in projective 3-space, and $j \in \{1, ..., n\}$ where n is the number of planes on the object that are either visible or can be inferred.

The vertices of the polyhedron are (X_i, Y_i, Z_i); under a pinhole projection model, projection onto the plane $Z = 1$ maps the point (X_i, Y_i, Z_i) to (x_i, y_i) where:

$$x_i = \frac{X_i}{Z_i} \quad \text{and} \quad y_i = \frac{Y_i}{Z_i}.$$

Now, in 3D at vertex i, there are three incidence conditions, each of which take the form:

$$a_j X_i + b_j Y_i + c_j Z_i + 1 = 0,$$

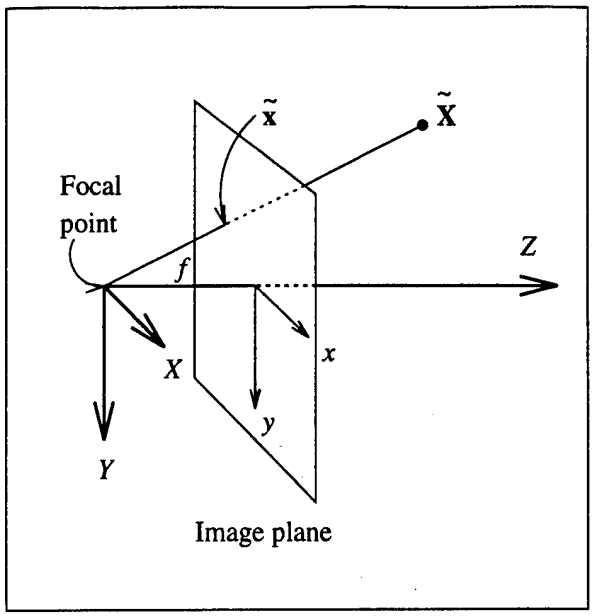


Figure 1: *The camera model used with the world and camera coordinate frames shown. A perspective imaging model is used to project the world point $\tilde{X}$ to the camera plane point $\tilde{x}$. Note that image coordinates only can be measured, rather than the camera coordinates of an image point (these two frames are linked by an affine map).*

and exist because three planes are incident on each vertex. At the i^{th} vertex, dividing the incidence conditions by Z_i setting $t_i = 1/Z_i$ gives:

$$a_j x_i + b_j y_i + c_j + t_i = 0. \tag{3}$$

Note that this equation is linear in the unknowns $\{a_j, b_j, c_j, t_i\}$.

From the trihedral assumption it is known that each image point lies on three planes, say $j \in \{p, q, r\}$. Since the t_i cannot be measured, they must be eliminated between pairs of equations (one equation for each of the three planes at the vertex), yielding two linear equations at each observed image vertex:

$$\begin{aligned} (a_p - a_r)x_i + (b_p - b_r)y_i + (c_p - c_r) &= 0, \\ (a_q - a_r)x_i + (b_q - b_r)y_i + (c_q - c_r) &= 0. \end{aligned} \tag{4}$$

For n observed or inferred object planes and m observed image vertices, there are $2m$ equations (which may or may not be independent) in $3n$ unknowns:

$$\mathtt{A}(\mathbf{x}', \mathbf{y}')\ \mathbf{w}' = 0, \tag{5}$$

where $\mathbf{x} = (x_1, \ldots, x_m)^T$ and $\mathbf{y} = (y_1, \ldots, y_m)^T$ are the observed coordinates of the image vertices, $\mathtt{A}(\mathbf{x}, \mathbf{y})$ is a $2m \times 3n$ array of constraints (2 constraints for each visible point), and $\mathbf{w} = (a_1, b_1, c_1, a_2, \ldots, c_n)^T$. The solution space of this system is the null space or kernel of the matrix $\mathtt{A}$, which represents the set of polyhedra that can be reconstructed from the image. The dimension of the kernel is $d \geq 3n - 2m$. The '$\geq$' arises because the set

of equations $\mathbf{A}$ may contain linear dependencies and so the polyhedra is not *position free*, otherwise we need consider only an '=' sign. The position free property is described in more detail by Sugihara [40] and in Section 2.3. In all cases, the dimension of the kernel immediately gives the number of degrees of freedom of the family of polyhedra that satisfy the image constraints, and d is exactly the number of parameters required to define the solution set.

2.3 Interpreting the kernel

The solution of eqn (5) can be written $\mathbf{w}' = \sum_{j=1}^{d} \mu_j' \mathbf{b}_j'$, where the kernel of $\mathbf{A}$ is spanned by the vectors $\mathbf{b}_i', i \in \{1, \ldots, d\}$. The sub-space spanned by $\mathbf{b}_j$ is a d dimensional subspace of the $3n$ dimensional polyhedral space.

This sub-space includes reconstructions where all vertices are coplanar, and all planes are the same plane. For example, the physical image plane is just such a reconstruction. There is a three-parameter family of planar reconstructions which can be written as:

$$\mathbf{w}' = (u, v, w, u, v, w, u, v, w, ..., u, v, w)^{\mathsf{T}}. \tag{6}$$

The three parameters of the family are u, v and w, each choice corresponding to a single plane as solution. It is necessary to avoid such degenerate reconstruction in order to compute true 3D reconstructions.

Planar reconstructions can be isolated and excluded from consideration by linearly transforming the basis such that $\mathbf{b}_i' \rightarrow \mathbf{b}_i$, $i \in \{1, 2, 3\}$. where

$$\begin{aligned} \mathbf{b}_1 &= (1, 0, 0, 1, \ldots, 1, 0, 0)^T, \\ \mathbf{b}_2 &= (0, 1, 0, 0, \ldots, 0, 1, 0)^T, \\ \mathbf{b}_3 &= (0, 0, 1, 0, \ldots, 0, 0, 1)^T. \end{aligned}$$

The kernel always contains the vectors $\mathbf{b}_i$, $i \in \{1, \ldots, 3\}$ since they span the planar solution in eqn (6). This transformation generates a new basis $\mathbf{b}_i$, $i \in \{1, \ldots, d\}$, and provided only solutions $\mathbf{w} = \sum_{i=3}^{d} \mu_i \mathbf{b}_i$ are considered, the degenerate planar "reconstructions" are excised.

In the sequel we concentrate on polyhedral recovery when $d = 4$; $d > 4$ is a topic for future work, as the relationship between the space of solutions and the geometry of the polyhedron is more complex. If the kernel is three dimensional, then it can contain only the span of the first three three vectors, and so the object must be planar. In all cases, the kernel must have dimension at least three.

2.3.1 Position free polyhedra

Under image noise the dimension of the kernel does not give an indication of whether an object is position free. Algebraically, an object is *position free* if the rank of $\mathbf{A}$ does not change as we move from the exact imaging case to the one in which we have measurement error. The reason for the change in rank is because constraints that are ideally dependent within $\mathbf{A}$ cease to be as coefficients are perturbed.

An example of an object that is not position free is the truncated tetrahedron shown in Fig. 2, where for minor perturbations of the vertices the only possible reconstruction of the polyhedron is as a plane figure. The incidence relations embedded in the tetrahedron cannot normally be observed from properties of measured points or of a measured kernel, as measurement error destroys the relation. As a result, for non-position free objects the

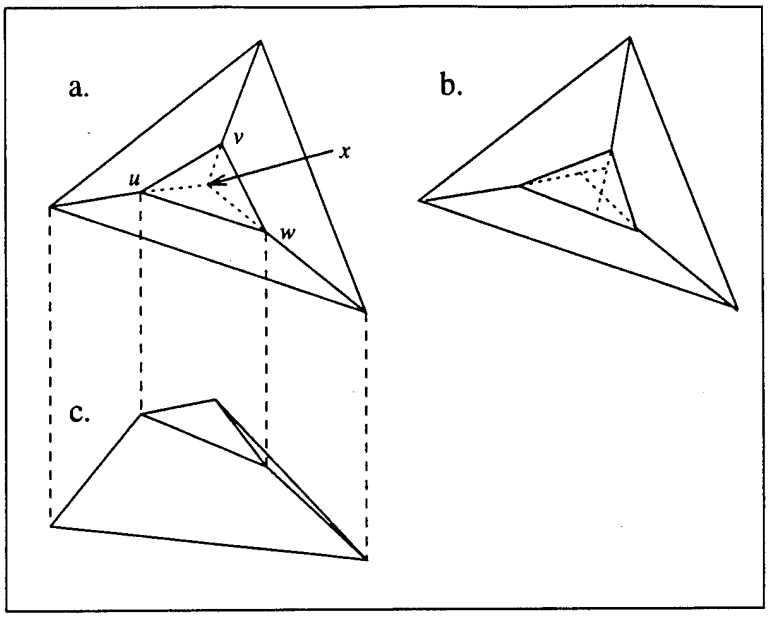


Figure 2: *(a) shows a plan view of a truncated tetrahedron which is not position free but can be realised as a 3D object. A different view of the same figure is shown in (c). If the point v is perturbed, the point x defined by the intersection of three lines in a is no longer defined, and so the polyhedron can be interpreted only as a plane figure (such as the one lying on this page). In real images all vertices are perturbed by noise, and so for a non-position free object such as a truncated tetrahedron one can never recover a full 3D interpretation.*

theory given in this paper does not provide a genuine reconstruction of the object. If useful information is to be extracted from the images of non-position free objects one must be able to spot the degeneracy in some other way (a similar observation was made by Sparr in [38]). The degeneracy in $\mathbf{A}$ is hard to observe through measurements on the matrix, and so one must determine the appropriate incidence constraint from other properties of the outline.[1] Currently techniques based on observing cycles in a planar graph determined from the outline are being investigated [36]. Commonly, but not always, polyhedra that are not position free lead to kernels of dimension three, allowing only plane interpretations.

For a perspective view of a position free polyhedron, where matrix $\mathbf{A}$ has kernel dimension four, distinct generic elements of the kernel represent projectively equivalent configurations of planes. Furthermore, a generic element of the kernel represents a configuration of planes which is projectively equivalent to some subset of the planes of the original polyhedron (or the whole polyhedron). Projective equivalence means that the projective invariants of a generic element of the kernel will be equal to the projective invariants of the actual object. The full proof of this statement is given in Appendix A, here we describe why the solution is invariant to camera calibration. Essentially, re-calibration causes an affine action on the image plane which may be alternatively thought of as an affine action on space prior to imaging. The camera matrix in eqn (1) may be expanded as:

[1] The natural algorithm, noting small eigenvalues in the singular value decomposition of $\mathbf{A}$, is unattractive because it is hard to determine what threshold is appropriate to describe 'small'.

$$\mathtt{P} = \mathtt{A}\ [\mathtt{I}|0]\ \mathtt{G},$$

where $\mathtt{A}$ is the affine calibration matrix, $\mathtt{I}$ the 3×3 identity matrix, and $\mathtt{G}$ a Euclidean action in 3-space. "Commuting" through matrix $\mathtt{A}$ gives

$$\begin{aligned} \mathtt{P} &= [\mathtt{I}|0] \begin{bmatrix} \mathtt{A} & 0 \\ 0 & 1 \end{bmatrix} \mathtt{G}, \\ &= [\mathtt{I}|0]\ \mathtt{A}_4\ \mathtt{G}. \end{aligned}$$

Hence, camera calibration in the image is equivalent to equivalent to an affine action on 3-space. Consequently, provided the measurements in 3-space are affine (or in our case projective), we will have invariance to camera calibration.

2.4 Indexing for polyhedra

Projective invariants are required to index an object model based on these reconstructions; projective invariants for systems of planes are relatively easily formed. For a system of n planes in general position, there are $3n - 15$ independent invariants (using the counting argument that appears in [12]). For example, given 6 planes, a possible set of invariants is:

$$I_1 = \frac{|\mathtt{I}_{3561}|.|\mathtt{I}_{3542}|}{|\mathtt{I}_{3564}|.|\mathtt{I}_{3512}|}, \qquad I_2 = \frac{|\mathtt{I}_{3562}|.|\mathtt{I}_{3142}|}{|\mathtt{I}_{3512}|.|\mathtt{I}_{3642}|}, \qquad I_3 = \frac{|\mathtt{I}_{3564}|.|\mathtt{I}_{5612}|}{|\mathtt{I}_{3561}|.|\mathtt{I}_{5642}|}, \tag{7}$$

where $\mathtt{I}_{abcd} = [\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_d]$, and the vectors $\mathbf{v}_i$ are the homogeneous representations of the planes. Invariants to linear groups typically take the form of ratios of determinants, see [30] for proof of their invariance.

We can therefore compute invariants from a single image of a polyhedron by computing the determinant functions for any of the possible reconstructions (here we are making use of the projective equivalence of all of the reconstruction solutions). Rather than choosing an arbitrary solution (which may be degenerate), we base the solution on the fourth basis vector of the kernel, which is composed of a set of three-vectors $\mathbf{m_i}$, one for each plane:

$$\mathbf{b}_4 = (\mathbf{m}_1^T, ..., \mathbf{m}_n^T)^T.$$

Each vector representing the homogeneous coordinates of a plane is given as $\mathbf{v}_i = (\mathbf{m}_i^{\mathrm{T}}, 1)^{\mathrm{T}}$.

Note that when the four planes $\{a, b, c, d\}$ forming I_{abcd} become concurrent (for example, a cycle of faces on a cuboid), the values of the invariants can become either infinite, undefined or zero because $|I_{abcd}| = 0$. Without *a priori* knowledge of the configuration we would still compute the invariants and index with them, and so such a degenerate configuration could be troublesome. Certainly, under the error-free case we should still recover representative invariants, but with errors in the point locations it becomes impossible to quantify 'near zero' values; this is because under a projectivity any determinant can be made arbitrarily large. An interesting current research issue is whether the determinant values really vary by large amounts for typical viewing conditions, or not.

2.5 Examples

In the following examples of caging, all the virtual polyhedra have at least four identifiably coplanar points. Configurations having at most three points on any plane, and no other constraints, suffer from the nil-invariance result of [5, 7, 18, 28].

	I_1	I_2	I_3
view a	-0.0141	0.9862	-1.0586
view b	-0.0356	0.9657	-0.9770
view c	-0.0189	0.9796	-1.0446
view d	-0.0400	0.9619	-0.9247
view e	-0.0012	0.9988	-1.0043
mean	-0.0219	0.9784	-1.0018
σ	0.0143	0.0135	0.0482
σ (%)	-	1.4	4.8

Table 1: *The invariants computed for the views in Fig. 3, with their mean and standard deviations. In the last row the standard deviation is shown as a percentage of the mean for I_2 and I_3. The value is not computed for I_1 because in this case the mean is ideally zero. In fact, the cage on the punch is projectively equivalent to a cube and so the invariants should be $\{I_1, I_2, I_3\} = \{0, 1, 1\}$. From these values it is also evident that the invariants remain constant over a change in viewpoint (see the graph in Fig. 4).*

In Fig. 3 a series of five images of a punch is shown with the same seven points marked on each image. These points are the images of vertices of the virtual polyhedron for which the invariants are computed. The images are processed using a Canny [6] edge detector, and straight lines fitted to the edge data using an orthogonal regression routine. Vertex positions are found by intersecting pairs of lines (by hand) and the points are then grouped into planes to form a six sided polygon. The invariants for (a), (b) and (d) should be equal, and those for (c) and (e) the same, though, not necessarily equal to those of the three former views. This is because a different set of points were used to compute the invariants for (c) and (e). However, because of the reflectional symmetry of the object the invariants are equal for all five views[2].

The invariants measured for the five views are given in Table 1, and are essentially constant over the change in viewpoint. In the table the standard deviations are given, as well as the value of the standard deviation as a percentage of the mean invariant value (except for the first invariant which is meant to be zero, as four of the planes intersect in a single point, and so the planes are not algebraically independent), and these are below 5%.

In Fig. 5 a second example is shown with two views of a calibration table. The measured invariants are given in Table 2; they remain constant over the change in viewpoint, but are different from the invariants computed for the punch in Fig. 3. This means that the invariants are both stable and give discrimination between different objects. Note that different invariants provide differing amounts of discrimination between objects; for example between the punch and the calibration table I_3 provides the only discrimination.

2.6 Verifying the polyhedral assumption

So far only the projective structure has been computed for the point set represented by the seven visible points in the images. This ignores other image information that can be used to confirm the hypothesis that a caged polyhedron is really being observed. Assuming that the object is a caged polyhedron allows a hypothesis on object identity, which in turn predicts the positions of other points in the image (in this case the eighth point on the caging polyhedron). If these other points are visible, the predicted and actual image positions

[2]The symmetry is equivalent to a projective automorphism on the object and so will not affect the measured invariants.

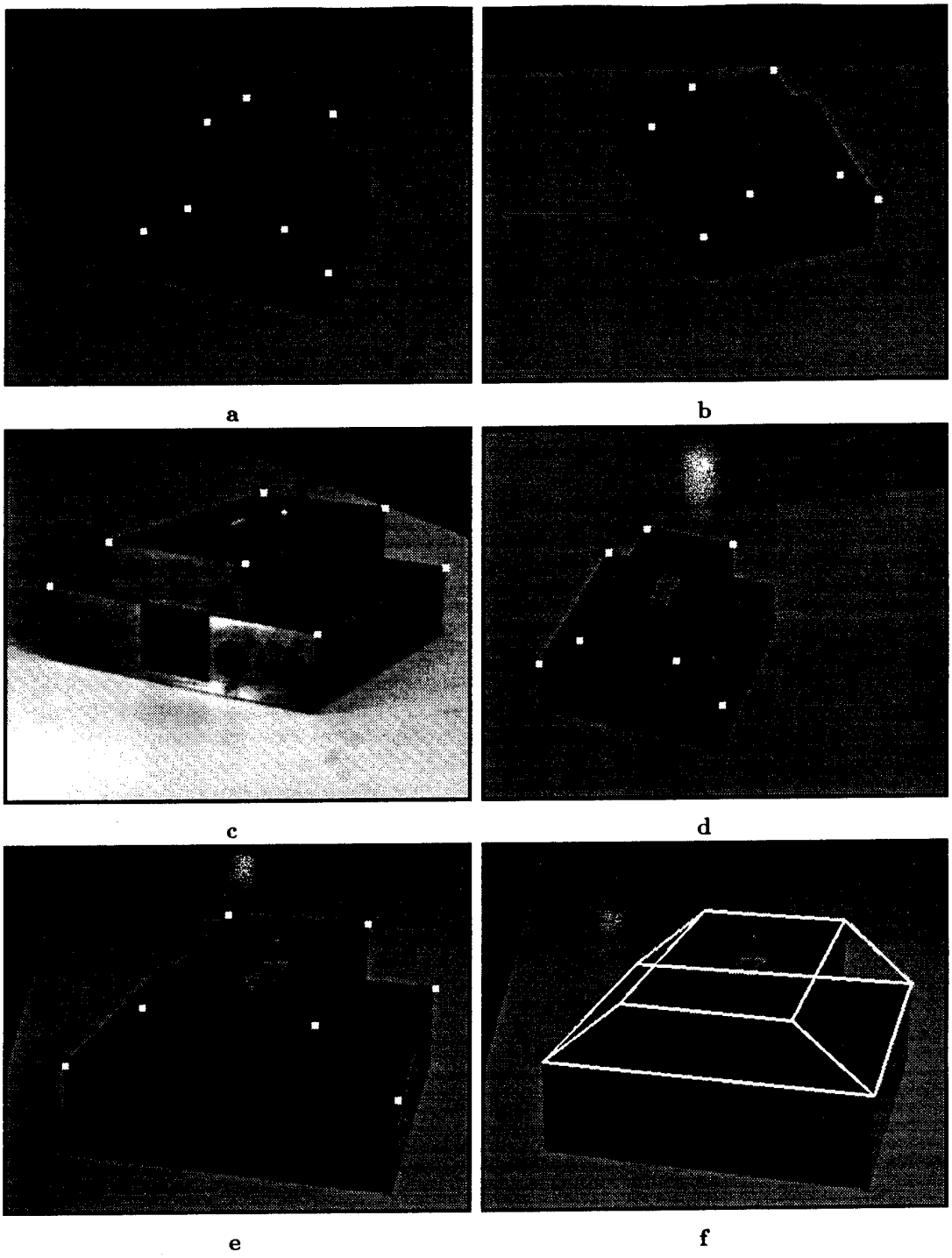


Figure 3: *(a)-(e) shows five views of a punch and the points used to compute the invariants. These points are caged by a polyhedron whose bounding planes are not actual object planes; the caging polyhedron for (e) is shown in (f). The computed invariants are given in Table 1.*

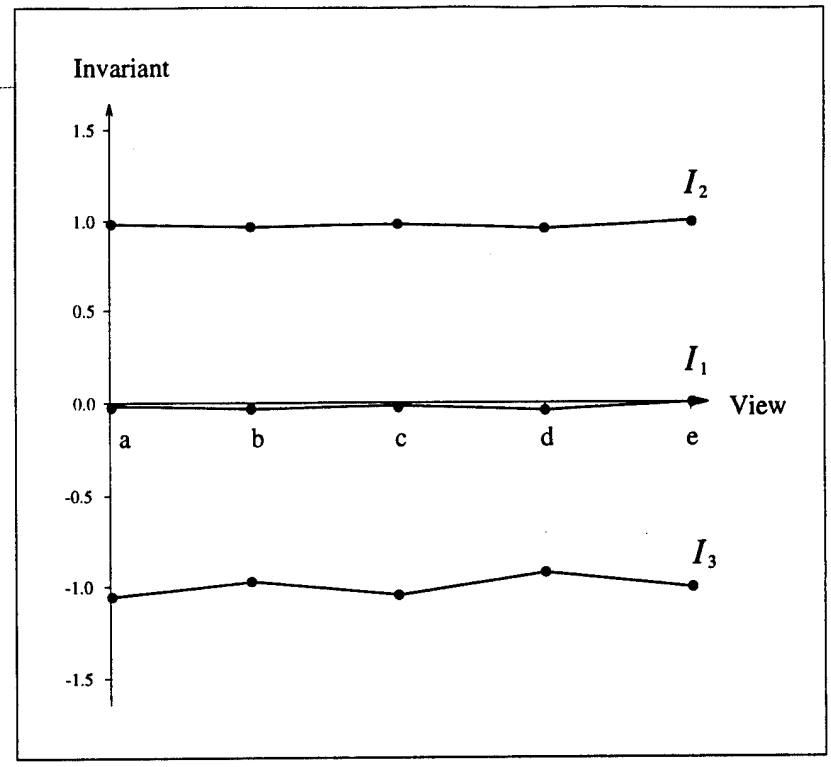
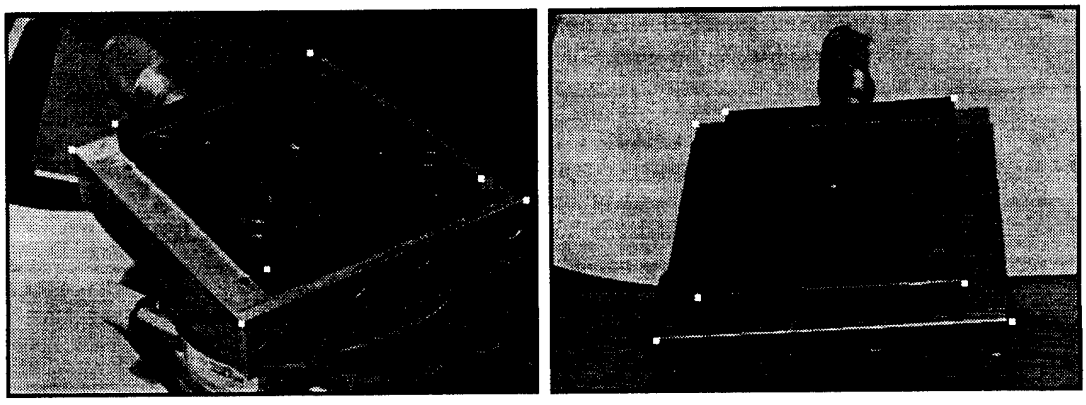


Figure 4: *The measured invariants for Fig. 3 and Table 1 remain unchanged over the change in viewpoint.*



a **b**

Figure 5: *Two views of the calibration table used to test the invariants. The seven points used to compute the invariants are marked in white. In the right image the eighth point is also visible; this could be used to overconstrain the solution, but in this case it is ignored.*

	I_1	I_2	I_3
view 1	-0.00146	0.991	-6.30
view 2	0.00117	1.007	-6.44

Table 2: *The invariants measured for the two views of the calibration table in Fig. 5. The invariants stay fairly constant even under image noise. In this case, I_3 can be used to discriminate between the calibration table and the punch in Fig. 3; again $\{I_1, I_2\} = \{0, 1\}$ for the calibration table.*

provide an independent check on the validity of the caging assumption. The positions of the remaining points can be determined either algebraically or geometrically, as shown for a six sided figure in the following sections.

2.6.1 Algebraic approach

The eighth point is obtained by intersecting the three virtual planes that bound the eighth vertex. It can be shown algebraically that the locus of the eighth point in three dimensions as different solutions are chosen for the reconstruction, is a line in space that passes through the camera centre. As this line projects to a point in the image, we see that the location of the eighth point is again independent of the choice of reconstruction.

The position of the point is shown in Fig. 6 for the examples given in Fig. 3. Instead of just showing the extra point, the complete polyhedron used to cage the data points is outlined. The good agreement between where the eighth point is expected (which is visible in some of the images) and the predicted position highlights the accuracy of the method. Again the position of the eighth point and the caging polyhedron for Fig. 5 are shown in Fig. 7.

2.6.2 Geometric approach

Often geometric manipulations provide a much more intuitive understanding of the imaging process than purely algebraic analysis. In this case, the position of the eighth point can also be predicted geometrically, not by explicitly computing the 3D projective structure of the polyhedra, but by an image-based construction detailed in Fig. 8.

3 Objects with bilateral symmetry

A single camera imaging a bilaterally symmetric object is equivalent to two identical cameras viewing the half structure, where one camera is transformed to the other by a reflection in the object symmetry plane. This is a special case of structure recovery from a single image of repeated structures [24]. Thus, a single uncalibrated perspective image of a bilaterally symmetric object is mathematically identical to an uncalibrated stereo pair of the half structure. Faugeras [10] and Hartley *et al.* [16] have shown that stereo reconstruction from two uncalibrated perspective images, generates a reconstruction differing from the actual 3D Euclidean geometry of the object by a 3D projective transformation. Consequently, 3D projective invariants of this recovered half-structure have the same value as projective invariants measured on the actual Euclidean half-structure.

The equivalence with stereo means that epipolar geometry can be defined within a single image, considerably simplifying the establishment of correspondences between the two half structures. This is described in Section 3.1. As described above, invariants can be

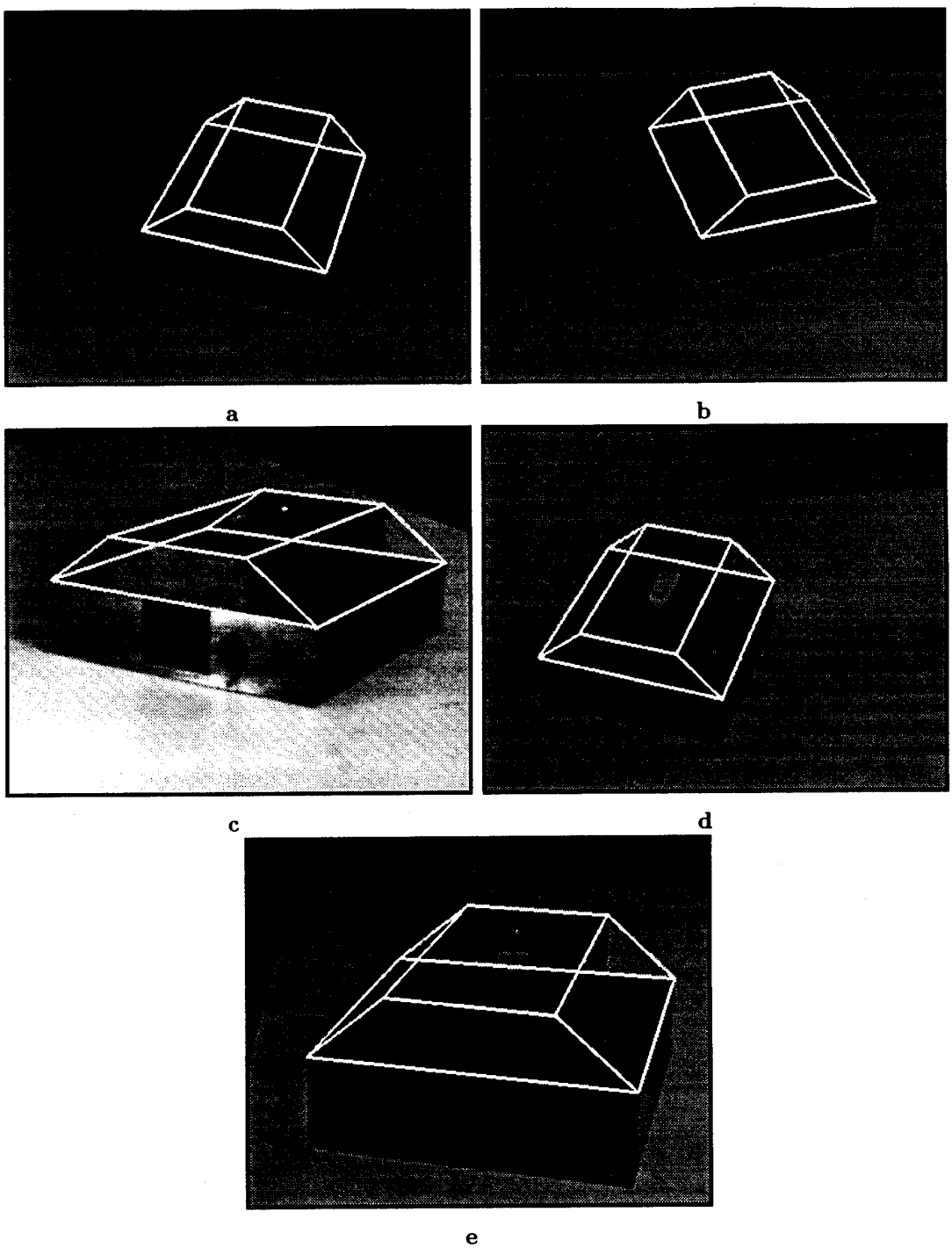
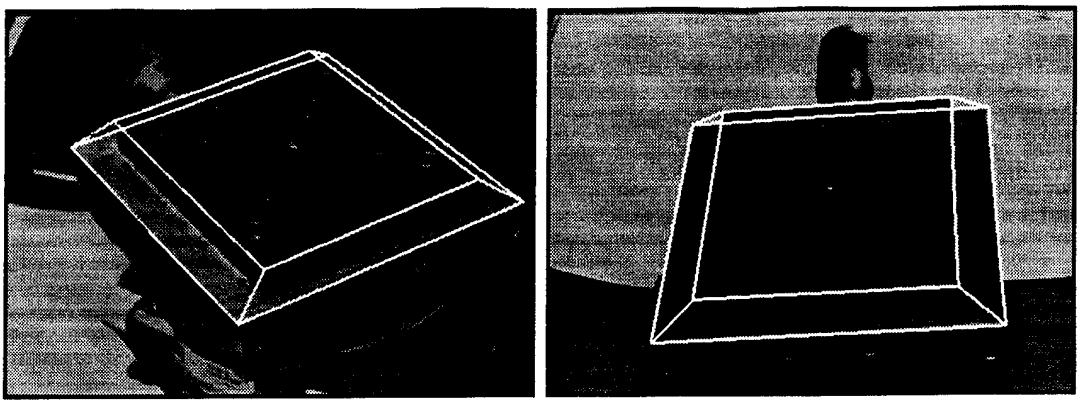


Figure 6: *The eighth point for the caging polyhedron. Note that at no stage has position of the eighth point been measured in the image, but its location has been computed from the other seven points.*



a b

Figure 7: *The caging polyhedron for the examples in Fig. 5.*

computed by recovering projective structure as in classical two view stereo. We describe two alternative methods which exploit the extra constraints available in this case. The first method (Section 3.2) uses a geometric argument to obtain planar projective invariants; each point-pair contributing two extra invariants. The second method (Section 3.3) is analytic, providing an efficient method to recover the full 3D structure. In this case each point-pair contributes three extra projective invariants.

It is worth noting that because only projective geometry is employed, the 3D object need only be projectively equivalent to one with bilateral symmetry.

3.1 Correspondence

Lines joining corresponding points on either side of the symmetry plane are parallel in 3D and are imaged as a set of lines converging to a vanishing point. These imaged correspondence lines and the vanishing point are the analogue of "epipolar lines" and the "epipole" in conventional stereo. We will use these terms from now on. The epipole can be determined using two pairs of corresponding points as illustrated in Figs 9 and 10. These points are matched by hand. Once the epipole has been computed, further correspondences are simplified to a 1D search on the epipolar line.

3.2 Planar invariants - geometric method

Lines joining corresponding 3D points intersect the symmetry plane at the midpoint of the corresponding points. Perspective projection does not preserve mid-points. However, the images of the 3D midpoints can be computed from the image (see below). All 3D midpoints are co-planar (they lie on the symmetry plane). There is therefore a plane projective transformation between the set of imaged mid-points and the 3D points on the plane of symmetry. Thus planar projective invariants can be measured in the image from the computed midpoints. For example five planar points have two plane projective invariants [30]:

$$I_1 = \frac{|\mathbf{N}_{431}||\mathbf{N}_{521}|}{|\mathbf{N}_{421}||\mathbf{N}_{531}|} \qquad \text{and} \qquad I_2 = \frac{|\mathbf{N}_{421}||\mathbf{N}_{532}|}{|\mathbf{N}_{432}||\mathbf{N}_{521}|}, \tag{8}$$

where $\mathbf{N}_{ijk} = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, $|.|$ is the determinant, and $\mathbf{x} = (x, y, 1)^\mathsf{T}$ is the homogeneous representation of a 2D point. Each additional point-pair adds a planar mid-point and

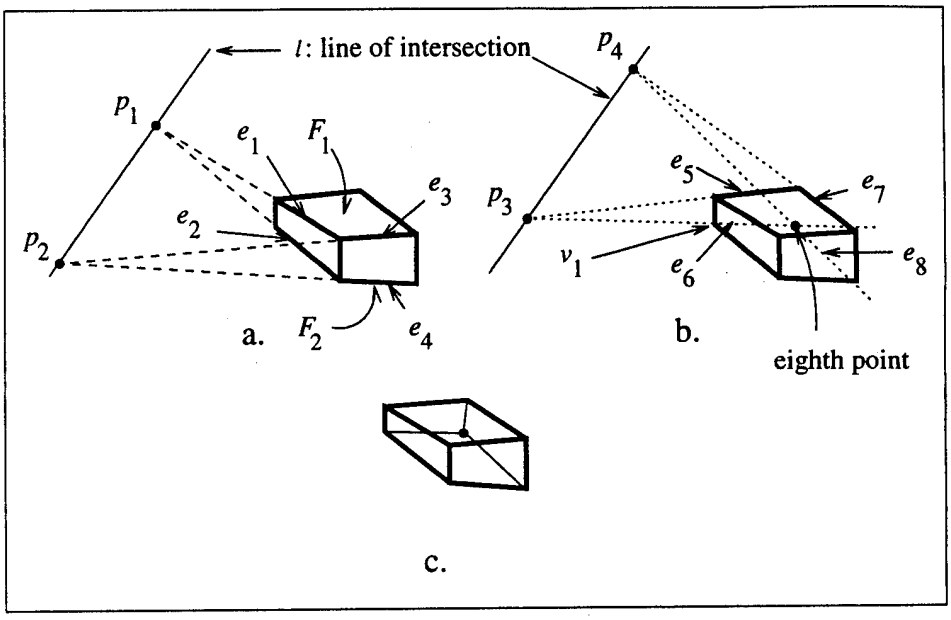


Figure 8: *The geometric construction for finding the eighth hidden point of a six sided polyhedra. Note that all of the properties of interest are projective (hence pairs of planes will always intersect). (a) first find the intersection of planes F_1 (the top plane) and F_2 (the bottom plane): edges e_1 and e_2 both lie on the left hand side plane of the object and so they will intersect at point p_1. As e_1 and e_2 lie respectively on planes F_1 and F_2, then the two planes also intersect at p_1. By a similar argument with edges e_3 and e_4 construct the point p_2 which also lies on both F_1 and F_2. Two planes intersect in projective 3-space in a line, this is given by the points p_1 and p_2 and denoted by l. In (b) reverse the process to find the hidden eighth point. This point must lie on the plane defined by e_5 and e_6. The edge e_6 is not observable, but it can be computed: e_5 and e_6 both lie on the rear plane and so must intersect (at a point p_3). As they lie on F_1 and F_2 respectively, p_3 must lie on l. Therefore p_3 is defined as the point at which e_5 intersects l. Edge e_6 must pass through v_1 as well as p_3 and so e_6 is defined. The eighth point lies on e_6, and so its locus is restricted to a line. The argument can be repeated for e_7 and e_8 to restrict the eighth point to lie on e_8. Thus, the point is defined by the intersection of e_6 and e_8. This is shown in (c) with the reconstruction of the hidden lines.*

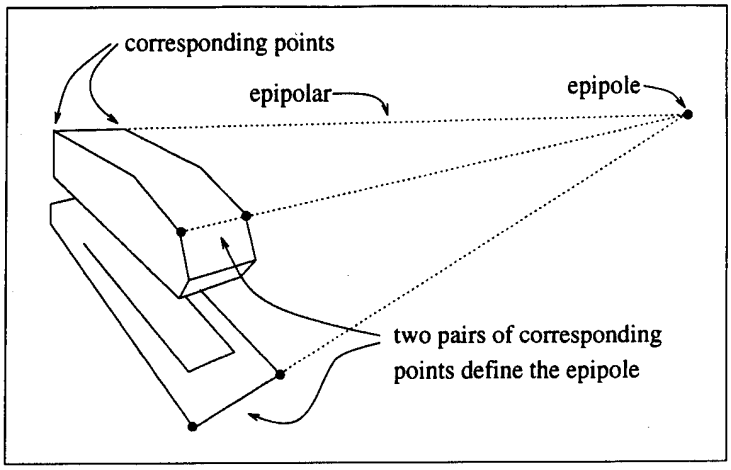


Figure 9: *The epipole can be located using the intersection of lines between two corresponding points on an object possessing a mirror symmetry (the points are marked by filled in circles). Epipolar lines can then be constructed through the epipole to aid correspondence.*

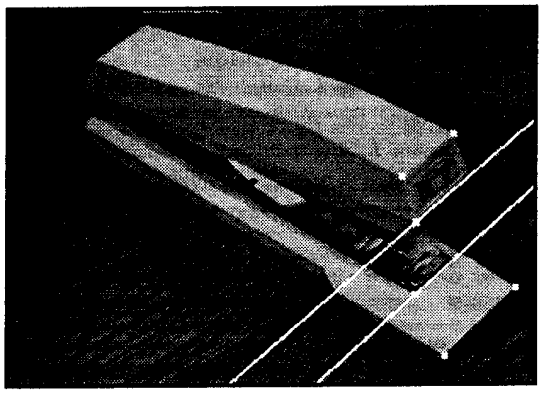


Figure 10: *The epipolar lines for two marked points are shown. Note that the corresponding points (by symmetry) lie on the lines. The four points marked in white are used to determine the epipolar structure.*

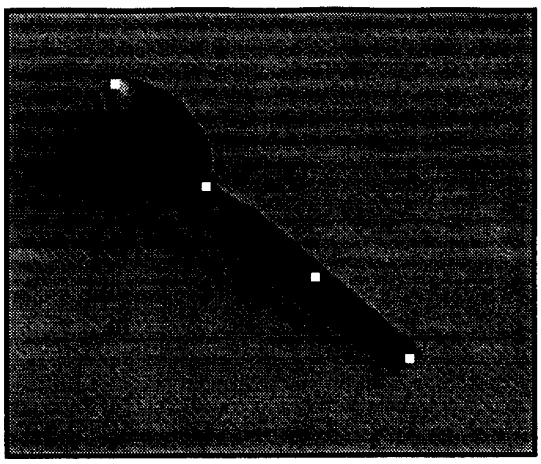


Figure 11: *The midpoint of any pair of points can be computed using the cross ratio once the location of the epipole is known. The midpoints for the four pairs of points of Fig. 15b are marked.*

generates two extra plane projective invariants.

The image of the 3D midpoints can be computed using a property of equally spaced points (see [39]): three collinear points, separated by the same distance, and taken with a point at infinity have a harmonic cross-ratio.[3] The point at infinity on the line joining two corresponding points can be observed since it is imaged as a vanishing point. Thus, the position of the midpoint in the image can be computed from the image coordinates of the corresponding points and from the image coordinates of the vanishing point. This is illustrated in Fig. 11. Furthermore, since computing a point that has a fixed cross-ratio with respect to three other points is linear, there is a unique solution.

An alternative method of obtaining the mid-point is to use the "cross-construction" shown in Fig. 12. This involves *two* corresponding point pairs, and does not use the position of the epipole. In practice the geometric construction is more accurate than the method involving the cross ratio as the dependence on the position of the epipole is removed – problems can occur if the epipole is distant from the image since it tends to become poorly localized; this affects the mid-point computation.

3.3 3D invariants - analytic method

The 3D point positions are reconstructed using a canonical coordinate frame for an object with bilateral symmetry, illustrated in Fig. 13 (cf. the canonical frame construction of [34]). Note, structure is recovered to better than a projective ambiguity because of the orthogonality constraints available. This reconstruction method is an extension to projection of the affine method of Fawcett, et al. [11].

The specification of the canonical frame requires choosing XY coordinates for four basis points, and the Z coordinate for one of the basis points. Once these coordinates are specified the XYZ coordinates of any other point can be determined in the canonical frame.

The four basis points provide a projective coordinate system for the symmetry plane. Since the imaged mid-points and symmetry plane are within a projective transformation,

[3] That is a cross ratio equal to negative unity.

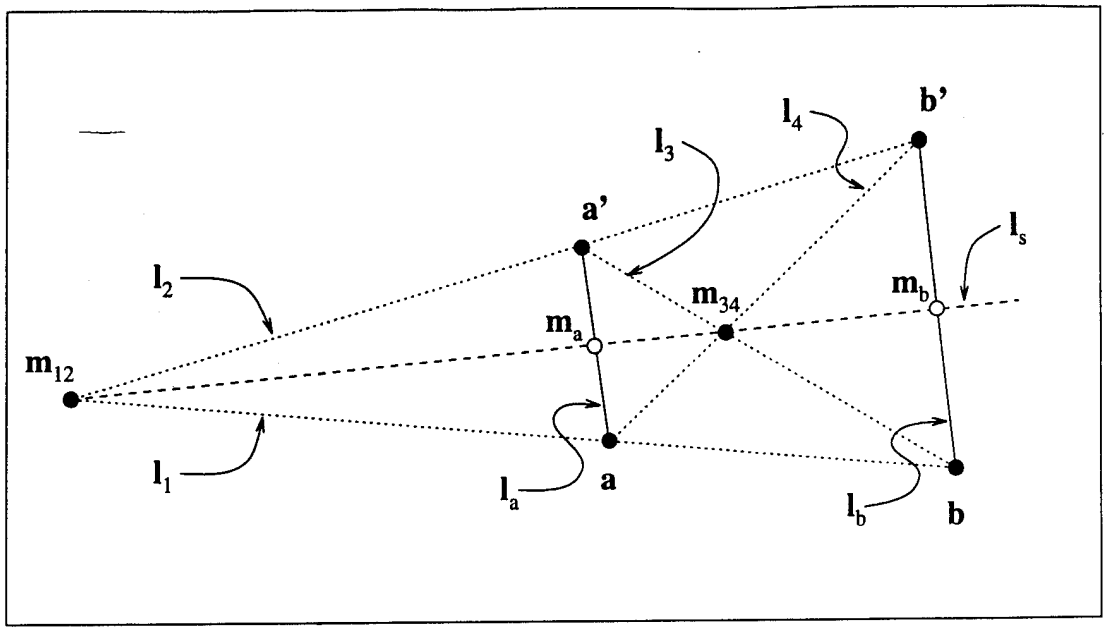


Figure 12: *The line of symmetry* $\mathbf{l}_s$ *of the plane* $\{\mathbf{a}, \mathbf{a}', \mathbf{b}', \mathbf{b}\}$ *can be determined geometrically as follows: compute the lines* $\mathbf{l}_i$, $i \in \{1, \ldots, 4\}$, *and then intersect as shown to give* $\mathbf{m}_{12}$ *and* $\mathbf{m}_{34}$*; these constrain* $\mathbf{l}_s$. *Then, the midpoints of* $\{\mathbf{a}, \mathbf{a}'\}$ *and* $\{\mathbf{b}, \mathbf{b}'\}$ *are defined by the intersections of* $\mathbf{l}_s$, $\mathbf{l}_a$ *and* $\mathbf{l}_b$.

this determines the XY coordinates in the canonical frame. It only remains to determine the Z coordinates. This proceeds in two steps:

1. **Determine the 3×4 matrix P:** This matrix projects from the canonical coordinate frame $(X, Y, Z, 1)^\top$ to the image $(x, y, 1)$.

$$k_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & 1 \end{bmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}. \tag{9}$$

   It is determined from four imaged point-pairs.

   The plane projection transformation for the symmetry plane ($Z = 0$) to the image is given by :

$$\mathtt{T} = \begin{bmatrix} P_{11} & P_{12} & P_{14} \\ P_{21} & P_{22} & P_{24} \\ P_{31} & P_{32} & 1 \end{bmatrix}.$$

   The correspondence between four imaged mid-points and their chosen position (the basis points) in the canonical frame determines T. There remains only three unknowns in P, $\{P_{13}, P_{23}, P_{33}\}$. From eqn (9) the 3D basis point $(X, Y, Z, 1)^\top$ (the one for which Z has been chosen) generates two linear equations. Similarly, the symmetry related point at $(X, Y, -Z, 1)^\top$ generates two equations. There are thus four linear equations for the three unknowns $\{P_{13}, P_{23}, P_{33}\}$, which are solved using least squares.

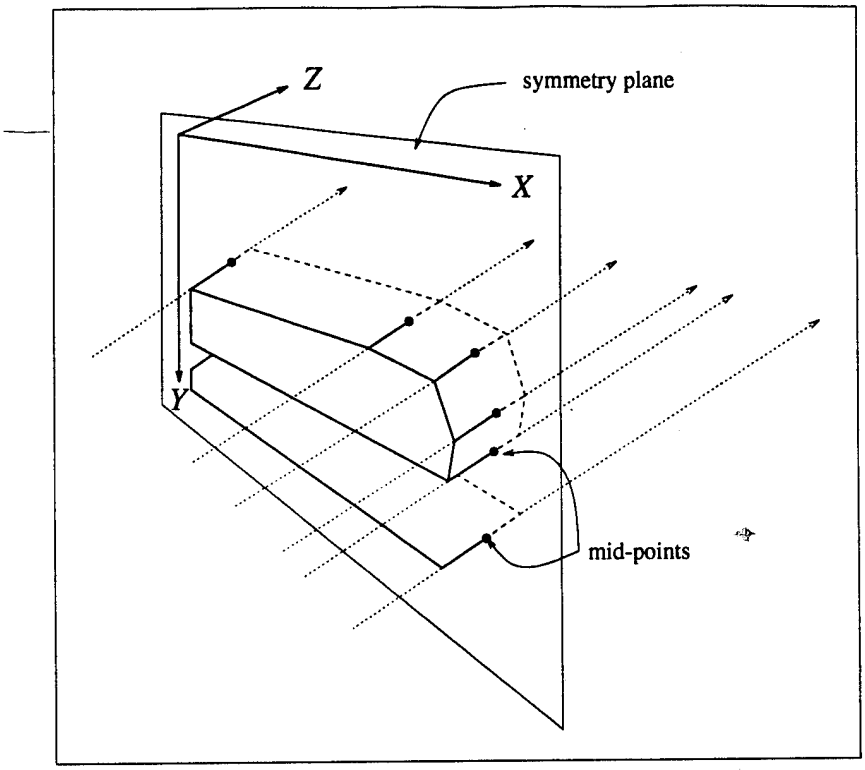


Figure 13: *The 3D reconstruction uses a coordinate frame with the X and Y axes in the plane of symmetry, and the Z direction perpendicular to the plane. Each correspondence line has constant XY coordinates, and the midpoints of any corresponding pair of points has $Z = 0$.*

2. **Determine Z coordinates using P:** XY coordinates for any other point-pairs are determined by applying T to their imaged mid-point. Each individual point then generates two (i.e. over-determined) equations for Z from eqn (9) (one for Z and the other, from the symmetry related point, for $-Z$). In the noise free case the two solutions will be equal; in practice the solutions are averaged to determine Z.

The construction is valid for any projective or affine view of the object provided the camera optical centre does not lie on the object plane of symmetry. Figure 14 illustrates different views of the reconstruction (drawn in a believable Euclidean frame) of a stapler reconstructed from the single perspective image shown in Fig. 10.

3.4 Verifying the bilateral symmetry assumption

The epipole and epipolar geometry, which are determined from the basis point correspondence, can be used as a test of the bilateral symmetry assumption. Points on one side must lie on their corresponding epipolar lines generated by points on the other side.

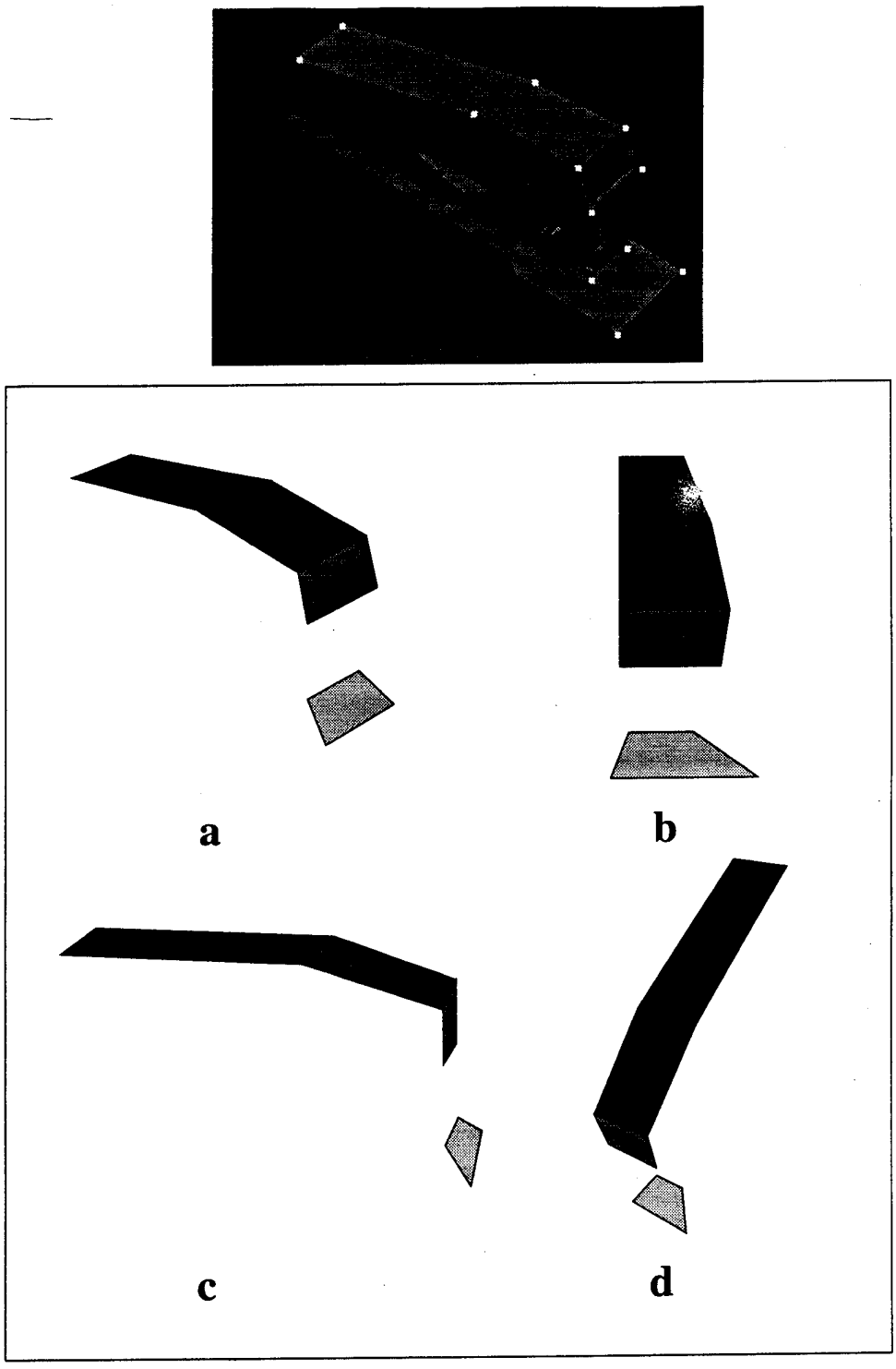
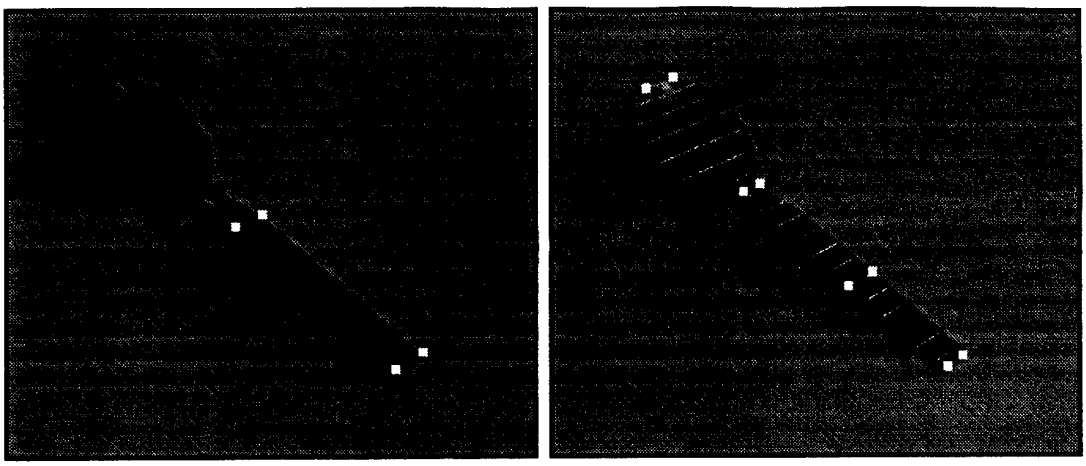


Figure 14: *Three dimensional structure is recovered modulo a projectivity from the points marked on the stapler from a single view. Four typical views are shown: (a) the viewpoint is from a position close that of the original camera view; (b) the observer has moved round to the front of the stapler; (c) and (d) from other general viewpoints.*



a b

Figure 15: *(a) a single view of an object with a plane of bilateral symmetry such as a teaspoon is sufficient to allow a full 3D projective reconstruction. Only two pairs of distinguished points are needed for the approach; these are recovered from surface markings and can be used to determine the epipolar structure of the image (b). Once the epipolar constraints are available one can produce an arbitrary number of correspondences; only eight points are required to specify the reconstruction. All other point-pair correspondences are assigned 3D projective coordinates based on the 3D locations of the first eight points.*

3.5 Further examples

The point set reconstruction method can be adapted to space curves and lines with bilateral symmetry.

First, we discuss space curves. The symmetry related curves are matched point-wise in the image using the epipolar geometry. Only two pairs of corresponding points are required to initiate the construction. For example in Fig. 15, two pairs of points can be extracted from observable object markings on a teaspoon (the part of the spoon of interest is the space curve defined by its boundary). These can be used to determine the position of the epipole and hence define the epipolar structure given in (b). The epipolar lines are initially used to determine the two further points required for the total of four basis points (ideally chosen to span the length of the object to yield better error behaviour[4]). Four different views of the reconstruction for Fig. 15 are shown in Fig. 16 (the 3D projective representation has been constrained to lie in a believable Euclidean frame).

Second, we consider the computation of invariants where the features are lines. The computation method for lines differs from that used for points. The method builds on a construction, proposed in [33] and used by [2, 9], for calculating the intersection of a line, lying out of the plane, with a plane containing four points. The intersection generates an additional point on the plane, and plane projective invariants can then be measured from the five coplanar points. The original construction employed two views of a general configuration. Again, if the configuration has bilateral symmetry (there are now two symmetrically related lines) the intersection can be determined from a single image. The method is illus-

[4]The basis points in Fig. 11 are close to collinear. However, the image locations of the points are known with sufficient (sub-pixel) accuracy that the construction works well. Note that the basis points effectively span the planar region containing the midpoints of the spoon.

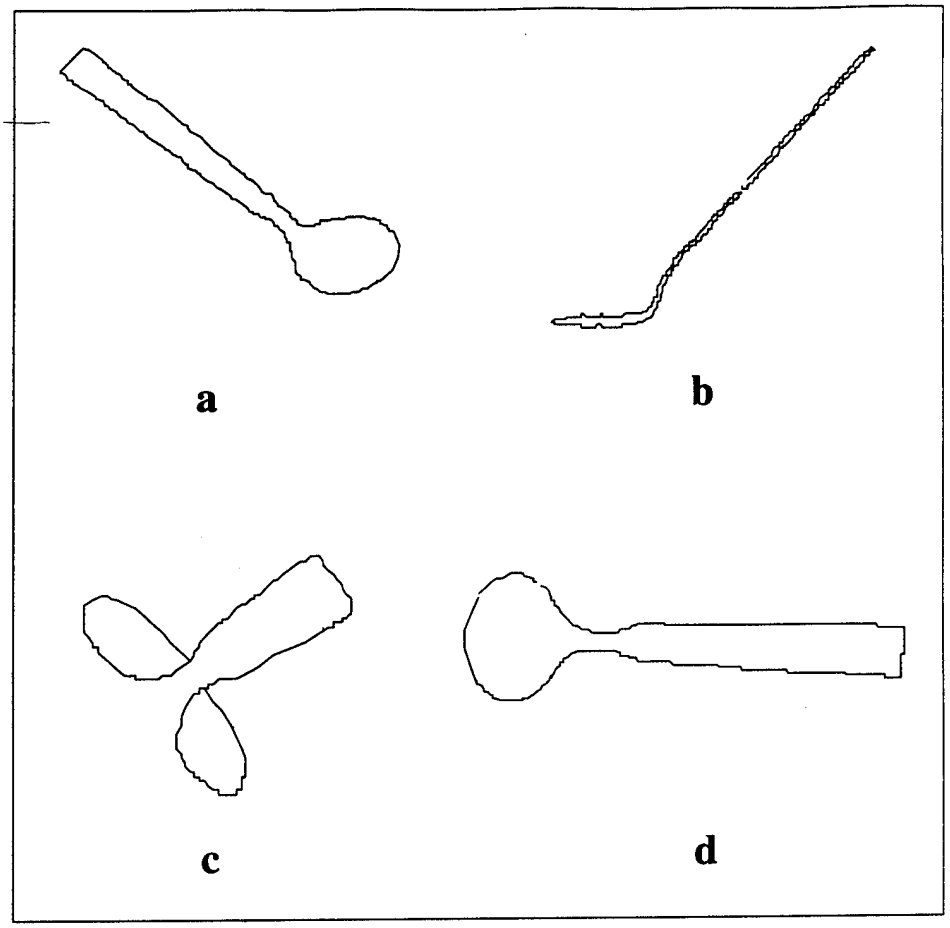


Figure 16: *Four different views of the 3D reconstruction gained from Fig. 15. The construction works very well: note the planarity of the handle recovered in (b), and the full 3D shape in all of the images.*

trated in Fig. 17, and the measured invariants (computed separately from each of two views) from eqn (8) given in Table 3. Clearly, the invariants are stable under change in viewpoint.

	I_1	I_2
view 1	-4.85	-0.211
view 2	-5.01	-0.211

Table 3: *The measured invariants for each of the two views in Fig. 17 are given. The values remain reasonably constant under a change in viewpoint.*

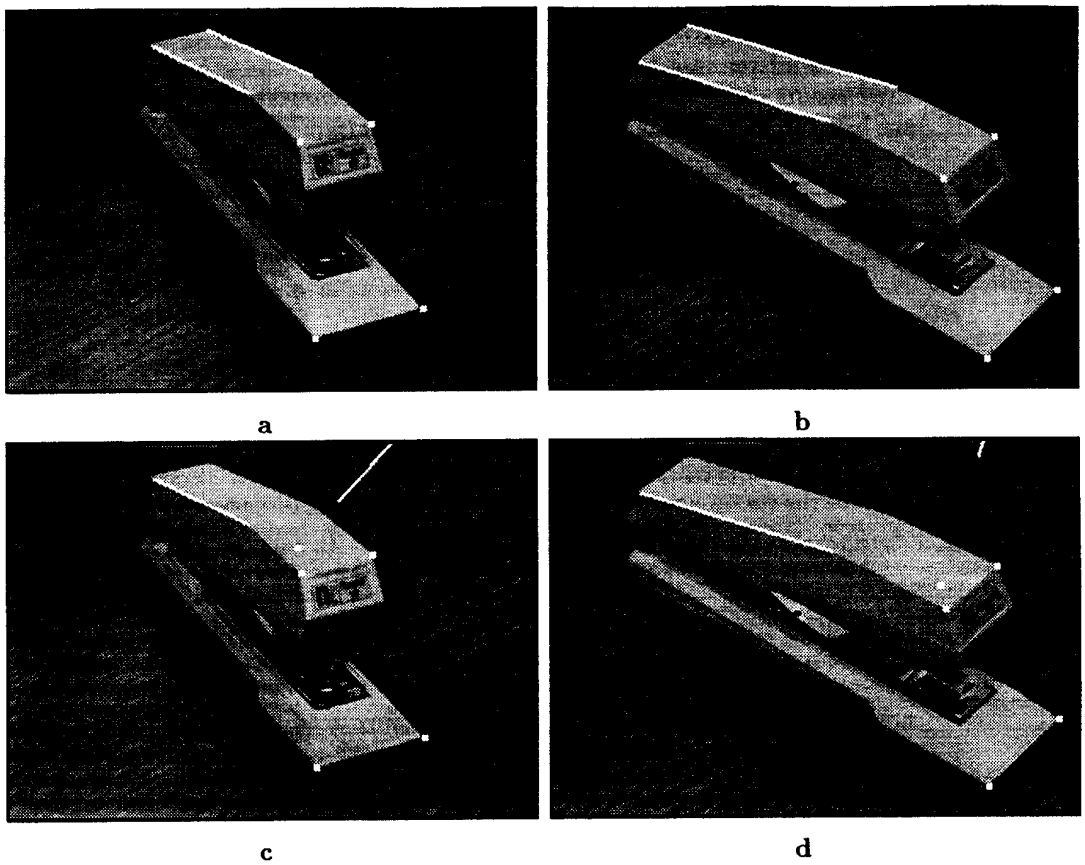


Figure 17: *Two different views of a stapler are shown in (a) and (b). From* each view *a pair of invariants can be measured from the image features highlighted in white: the four coplanar points (coplanar by symmetry) and the symmetrically related lines that lie out of the plane. Invariants are computed from five coplanar points: the four points shown, and a fifth point generated by the intersection of the left white line with the plane containing the four points. We denote the plane by* Π, *and the intersection point by* $\mathbf{X}_I$.
$\mathbf{X}_I$ *is computed in two stages. First, the projective transformation,* $\mathtt{T}$, *is determined which maps each of the four points in* Π *onto its symmetrically related corresponding point. This transformation maps points on* Π *from one side of the symmetry plane to their correspondences on the other. In particular, the correspondence to* $\mathbf{X}_I$ *lies on the extension of the right line. Therefore,* $\mathbf{X}_I$ *will lie on the line that results from mapping the right line by* $\mathtt{T}$. *As* $\mathbf{X}_I$ *also lies on the left line, we know it is given by the intersection of the left line and the result of mapping the right line.* $\mathbf{X}_I$ *is thus determined.*
The process can be repeated for (b) and (d) and the invariants compared. Note that invariants can also be computed by projecting the left line onto the right half image; these invariants will be functionally dependent on the first two values.

4 Discussion

In this paper we have shown how projectively invariant indexing functions can be constructed for ~~3D~~ point sets given some assumption on structure. The types of structure that we have discussed are relevant to polyhedra and objects possessing a bilateral symmetry. Together, these results make a significant contribution to the applications of invariant theory and dispel the belief that invariants can not be measured for (non-general) three-dimensional point sets from a single image. However, there is still much work to be done if these descriptors are to be used within the type of reliable object recognition system that we desire (along the lines of that reported for planar objects in [37]). The following two sections introduce some of the considerations required before such systems can be built.

4.1 Caged point sets

One of the major drawbacks of the polyhedral construction is that it assumes the presence of points grouped onto planes prior to constraint formulation.[5] Additionally, each point must be constrained to lie on multiple planes. This is a much harder problem: perhaps the solution lies in using invariants of lesser groups (such as affine [45] or *quasi-invariants* [4]) to solve the grouping task before proceeding with full projective measures, and to a certain extent forms the basis of Lowe's grouping work [21].

Another problem evolves from the richness of the polyhedral description. To a certain extent, the invariants account for more global shape features than just the local vertex information used for example by Wayner [45]. This develops into a problem with respect to the property of position freeness: generally, large structures are unlikely to be position free. Although a graph matching technique has been suggested to flag when polyhedra are not position free [36], developing a principled way to adjust the algebraic constraints so that they return to being dependent remains very much a topic for future work. An alternative approach would be to measure local descriptions and develop a hypothesis merging strategy as exploited in [37].

The one real benefit of the caging and projective polyhedral approach to recognition is that exact projective information is yielded prior to recognition. This contrasts with the conclusions of Sugihara [40], who through using a Euclidean world model, found that only parameterised families of shapes could be represented. However, another goal of future work should be to extend the 4 *dof.* polyhedral reconstruction presented in this paper to encompass a more general class of polyhedra such as 5 *dof.* figures, or perhaps those of higher degree.

4.2 Objects with symmetry

Two view invariants are attractive because of their simplicity. Very few features are required, though again, grouping (correspondence) must be solved intra-image. Grouping is simplified once the epipolar structure has been computed: with the single view invariants for symmetric objects, solving for epipolar structure is very easy. Once two pairs of matching symmetric points have been found the epipolar structure is defined and many other correspondences are available. When there are a sufficient number of correspondences full 3D projective

[5] Generally the world is not constructed of well formed polyhedra, and so any major application would rely on caging. Even in cases in which actual polyhedra are visible, the extraction of accurate and complete line drawings is currently considered hard. Recent research into the extraction of polyhedral image features using snakes has appeared encouraging [31].

structure can be recovered with further use of the location of the epipole. The construction is also remarkably stable.

As symmetry is prevalent in many environments it is clear that the single view symmetry invariants have a broad scope within computer vision. In fact, the construction can be applied to any projectively related repeated structures. In particular, structures repeated by translation or reflections, and objects projectively equivalent to these, have the simple epipolar constraint provided by four points.

Acknowledgements

This project describes research supported by a number of sources: General Electric; NSF award no. IRI-9209729; a grant from United States Air Force Office of Scientific Research AFOSR-91-0361;a National Science Foundation Young Investigator Award with matching funds from GE, Rockwell International, Tektronix and Eugene Rikel; ESPRIT Project 6448 'VIVA'; and the Universities of Iowa, UC Berkeley and Oxford. We are grateful to Sabine Demey for providing one of the image sequences and its segmentation, and to Tom Binford for helpful discussions.

A Projective equivalence of all the kernel solutions

The derivation in this appendix shows that the measures taken for polyhedra in Section 2 are in fact invariant to three-dimensional projective actions. We prove the following theorem:

Theorem:

For a position free polyhedra with a matrix $\mathbf{A}$ of kernel dimension four, all of the solutions of the kernel represent reconstructions of space are projectively equivalent.

Proof:

The solution space for the polyhedra is represented by $\mathbf{w} = \sum_{i=1}^{4} \mu_i \mathbf{b}_i$. We ensure the basis has the form with the first three vectors as in eqn (7), and then let the fourth basis vector be composed of a set of three-vectors $\mathbf{m_i}$, one for each plane:

$$\mathbf{b}_4 = (\mathbf{m}_1^{\mathrm{T}}, ..., \mathbf{m}_n^{\mathrm{T}})^{\mathrm{T}}.$$

Then, considering solutions for the planes of the form $\mathbf{v} = (\mathbf{u}^{\mathrm{T}}, 1)^{\mathrm{T}}$, we have:

$$\begin{pmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{pmatrix} = \mu_1 \mathbf{b}_1 + \mu_2 \mathbf{b}_2 + \mu_3 \mathbf{b}_3 + \mu_4 \begin{pmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_n \end{pmatrix},$$

Letting $\lambda = 1/\mu_4$, $\mathbf{a} = -\lambda\ (\mu_1, \mu_2, \mu_3)^{\mathrm{T}}$, gives:

$$\mathbf{m}_i = \lambda \mathbf{u}_i + \mathbf{a}.$$

Writing $\mathtt{I}_3$ as the 3×3 identity matrix yields:

$$\begin{pmatrix} \mathbf{m}_i \\ 1 \end{pmatrix} = \begin{bmatrix} \lambda\, \mathtt{I}_3 & \mathbf{a} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{u}_i \\ 1 \end{pmatrix},$$

$$\text{or,} \qquad \mathbf{M}_i = \mathtt{P}\, \mathbf{v}_i.$$

Therefore, each choice of $\mathbf{M}_i$ is a projectivity of the actual world planes $\mathbf{v}_i$.

References

[1] Barrett, E.B., Payton, P.M. and Brill, M.H. "Contributions to the Theory of Projective Invariants for Curves in Two and Three Dimensions," Proceedings 1^{st} DARPA-ESPRIT Workshop on Invariance, p.387-425, March 1991.

[2] Beardsley, P.A, "Applications of Projective Geometry to Robot Vision," D.Phil. Thesis, Department of Engineering Science, Oxford University, 1992.

[3] Beardsley, P.A., Zisserman, A. and Murray, D.W. "Navigation using Affine Structure from Motion," Proceedings ECCV3, 1994.

[4] Binford, T.O. and Levitt, T.S. "Quasi-Invariants: Theory and Explanation," Proceeding Darpa IUW, p.819-829, 1993.

[5] Burns, J.B., Weiss, R.S. and Riseman, E.M. "The Non-Existence of General-Case View-Invariants," in [30], p.120-131, 1992.

[6] Canny J.F. "A Computational Approach to Edge Detection," *IEEE Trans. PAMI,* Vol. 8, No. 6, p.679-698, 1986.

[7] Clemens, D.T. and Jacobs, D.W. "Model Group Indexing for Recognition," Proceedings CVPR91, p.4-9, 1991, and *IEEE Trans. PAMI,* Vol. 13, No. 10, p.1007-1017, October 1991.

[8] Clowes, M.B. "On Seeing Things," *Artificial Intelligence,* Vol. 2, p.79-116, 1971.

[9] Demey, S., Zisserman, A. and Beardsley, P. "Affine and Projective Structure from Motion," Proceedings BMVC92, p.49-58, 1992.

[10] Faugeras, O. "What can be Seen in Three Dimensions with an Uncalibrated Stereo Rig?" Proceedings ECCV2, p.563-578, 1992.

[11] Fawcett, R. Zisserman, A. and Brady, J.M. "Extracting Structure from Single Affine Views of 3D Point Sets with One or Two Bilateral Symmetries," Proceedings BMVC93, 1993.

[12] Forsyth, D.A., Mundy, J.L., Zisserman, A.P., Coelho, C., Heller, A. and Rothwell, C.A. "Invariant Descriptors for 3-D Object Recognition and Pose," *IEEE Trans. PAMI,* Vol. 13, No. 10, p.971-991, October 1991.

[13] Forsyth, D.A. "Recognizing Algebraic Surfaces from their Outlines," Proceedings ICCV4, p.476-480, 1993.

[14] Gordon, G,G. "Shape from Symmetry," Proceedings SPIE Intelligent Robots and Computer Vision VIII, Algorithms and Techniques, Vol. 1192, 1989.

[15] Gros, P. "Outils Geometriques pour la Modelisation et la reconnasissance d'objets polyedriques," Ph.D. thesis, LIFIA-IMAG-INRIA, Grenoble, 1993.

[16] Hartley, R.I., Gupta, R. and Chang, T. "Stereo from Uncalibrated Cameras," Proceedings CVPR92, p.761-764, 1992.

[17] Huffman, D.A. "Impossible Objects as Nonsense Sentences," *Machine Intelligence,* Vol. 6, Meltzer, B. and Michie, D. editors, Edinburgh University Press, 1971.

[18] Huttenlocher, D.P. and Kleinberg, J.M. "On Invariants of Sets of Points or Line Segments Under Projection," *TR-92-1292,* Cornell University, 1992.

[19] Jacobs, D.W. "Space Efficient 3D Model Indexing," Proceedings CVPR92, p.439-444, 1992.

[20] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," Proceedings CVPR88, p.335-344, 1988.

[21] Lowe, D.G. *Perceptual Organization and Visual Recognition,* Kluwer Academic Publishers, 1985.

[22] Lowe, D.G. "The Viewpoint Consistency Constraint," *International Journal of Computer Vision,* Vol. 1, No. 1, p.57-72, 1987.

[23] Liu, J., Mundy, J.L., Forsyth, D.A., Zisserman, A. and Rothwell, C.A. "Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders," Proceedings CVPR93, p.123-128, 1993.

[24] Liu J.S., Mundy J.L. and Walker E.L., "Recognizing Arbitrary Objects from Multiple Projections," *Proc. Asian Conference in Computer Vision,* 1993.

[25] Mackworth, A.K. "Interpreting Pictures of Polyhedral Scenes," *Artificial Intelligence,* Vol. 4, p.99-118, 1973.

[26] Mitsumoto H., Tamura S., Okazaki K., Kajimi N. and Fukui Y., "3D Reconstruction Using Mirror Images Based on a Plane Symmetry Recovery Method", *PAMI,* 14, 9, 941-945, 1992.

[27] Mohr, R. and Morin, L. "Relative Positioning from Geometric Invariants," Proceedings CVPR91, p.139-144, 1991.

[28] Moses, Y. and Ullman, S. "Limitations of Non Model-Based Recognition Systems," Proceedings ECCV2, p.820-828, 1992.

[29] Mundy, J.L. and Heller, A.J. "The Evolution and Testing of a Model-Based Object Recognition System," Proceedings ICCV3, p.268-282, 1990.

[30] Mundy, J.L. and Zisserman, A.P. *Geometric Invariance in Computer Vision,* MIT Press, 1992.

[31] Mundy, J.L., Huang, C., Liu, J., Hoffman, W., Forsyth, D.A., Rothwell, C.A., Zisserman, A., Utcke, S. and Bournez, O. "MORSE: A 3D Object Recognition System Based on Geometric Invariants," to appear ARPA IUW, 1994.

[32] Pillow, N., Utcke, S. and Zisserman, A. ""Viewpoint-Invariant Representation of Generalized Cylinders Using the Symmetry Set", To appear, BMVC94.

[33] Quan, L. and Mohr, R. "Towards Structure from Motion for Linear Features through Reference Points," Proceedings IEEE Workshop on Visual Motion, 1991.

[34] Rothwell, C.A., Zisserman, A., Forsyth, D.A. and Mundy, J.L. "Canonical Frames for Planar Object Recognition," Proceedings ECCV2, p.757-772, 1992.

[35] Rothwell, C.A., Zisserman, A., Mundy, J.L. and Forsyth, D.A. "Efficient Model Library Access by Projectively Invariant Indexing Functions", Proceedings CVPR92, p.109-114, 1992.

[36] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L. "Extracting Projective Information from Single Views of 3D Point Sets," *TR OUEL 1927/92*, Department of Engineering Science, Oxford University, Oxford, 1992.

[37] Rothwell, C.A. "Recognition Using Projective Invariance", D.Phil. Thesis, Department of Engineering Science, University of Oxford, Oxford, 1993, to appear OUP 1994.

[38] Sparr, G. "Depth Computations from Polyhedral Images," Proceedings ECCV2, p.378-386, 1992.

[39] Springer, C.E. *Geometry and Analysis of Projective Spaces,* Freeman, 1964.

[40] Sugihara, K. *Machine interpretation of Line Drawings,* MIT Press, 1986.

[41] Taubin, G. and Cooper, D.B. "Recognition and Positioning of 3D Piecewise Algebraic," Proceeding DARPA Image Understanding Workshop, p.508-514, September 1990.

[42] Thompson, D.W. and Mundy, J.L. "Three-Dimensional Model Matching from an Unconstrained Viewpoint," Proceedings ICRA, p.208-220, 1987.

[43] Van Gool, L. Kempenaers, P. and Oosterlinck, A. "Recognition and Semi-Differential Invariants," Proceedings CVPR91, p.454-460, 1991.

[44] Waltz, D. "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision,* Winston, P.H. editor, M°Graw-Hill, p.19-91, 1975.

[45] Wayner, P.C. "Efficiently Using Invariant Theory for Model-Based Matching," Proceedings CVPR91, p.473-478, 1991.

[46] Weinshall, D. "Model-Based Invariants for 3-D Vision," *IJCV,* Vol. 10, No. 1, p.27.42, 1993.

[47] Weiss, I. "Projective Invariants of Shapes," Proceedings DARPA Image Understanding Workshop, p.1125-1134, April 1988.

[48] Zhang, A., Deriche, R., Faugeras, O. and Luong, Q-T. "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry," *TR* INRIA, Sophia Antipolis, 1994.

[49] Zisserman, A.P., Forsyth, D.A., Mundy, J.L., Rothwell, C.A. and Liu, J. "3D Object Recognition using Invariance", submitted for publication, 1994.

Solving Polynomial Systems Using a Branch and Prune Approach

P. Van Hentenryck
Brown University
Box 1910
Providence, RI 02912
pvh@cs.brown.edu

D. McAllester
MIT AI Lab
Technology Square, 545
Cambridge, USA
dam@ai.mit.edu

D. Kapur
SUNY at Albany
Dep. of Computer Science
Albany, NY-12222
kapur@cs.albany.edu

Abstract

This paper presents **Newton**, a branch & prune algorithm to find all isolated solutions of a system of polynomial constraints. **Newton** can be characterized as a global search method which uses intervals for numerical correctness and for pruning the search space early. The pruning in **Newton** consists in enforcing at each node of the search tree a unique local consistency condition, called box-consistency, which approximates the notion of arc-consistency well-known in artificial intelligence. Box-consistency is parametrized by an interval extension of the constraint and can be instantiated to produce Hansen-Segupta narrowing operator (used in interval methods) as well as new operators which are more effective when the computation is far from a solution. **Newton** has been evaluated on a variety of benchmarks from kinematics, chemistry, combustion, economics, and mechanics. On these benchmarks, it outperforms the interval methods we are aware of and compares well with state-of-the-art continuation methods. Limitations of **Newton** (e.g., a sensitivity to the size of the initial intervals on some problems) are also discussed. Of particular interest is the mathematical and programming simplicity of the method.

AMS subject Classification: 65H10, 65G10
Keywords: System of Equations, Global Methods, Interval and Finite Analysis

1 Introduction

Many applications in science and engineering (e.g., chemistry, robotics, economics, mechanics) require finding all isolated solutions to a system of polynomial constraints over real numbers. This problem is difficult due to its inherent computational complexity (i.e., it is NP-hard) and due to the numerical issues involved to guarantee correctness (i.e., finding all solutions) and to ensure termination. Several interesting methods have been proposed in the past for this task, including two fundamentally different methods: interval methods (e.g., [4, 5, 7, 8, 11, 13, 14, 15, 20, 26, 30]) and continuation methods (e.g., [25, 35]). Continuation methods have been shown to be effective for problems for which the total degree is not too high, since the number of paths explored depends on the estimation of the number of solutions. Interval methods are generally robust but tend to be slow.

The purpose of this paper is to propose and to study a novel algorithm called **Newton**. From a user standpoint, **Newton** receives as input a system of polynomial constraints over, say, variables $x_1, \ldots, x_n$ and a box, i.e., an interval tuple $\langle I_1, \ldots, I_n \rangle$ specifying the initial range of these variables; it returns a set of boxes of specified accuracy containing all solutions.

Operationally, **Newton** is a branch & prune algorithm which was inspired by the traditional branch and bound approach used to solve combinatorial optimization problems. **Newton** uses intervals to address the two fundamental problems listed above. Numerical reliability is obtained by evaluating functions over intervals using outward rounding (as in interval methods). The complexity issue is addressed by using constraints to reduce the intervals early in the search. The pruning in **Newton** is achieved by enforcing a unique local consistency condition, called box-consistency, at each node of the search tree. Box-consistency is an approximation of arc-consistency, a notion well-known in artificial intelligence [17, 19] and used to solve discrete combinatorial problems in several systems (e.g.,, [32, 33]). Box-consistency is parametrized by an interval extension operator for the constraint and can be instantiated to produce various narrowing operators. In particular, box-consistency on the Taylor extension of the constraint produces a generalization of Hansen-Segupta operator [8], well-known in interval methods. In addition, box-consistency on the natural extension produces narrowing operators which are more effective when the algorithm is not near a solution. **Newton** has the following properties:

- **Correctness:** **Newton** finds all isolated solutions to the system in the following sense: if $\langle v_1, \ldots, v_n \rangle$ is a solution, then **Newton** returns at least one box $\langle I'_1, \ldots, I'_n \rangle$ such that $v_i \in I_i$ $(1 \leq i \leq n)$. In addition, **Newton** may guarantee the existence of a unique solution in some or all the boxes in the result. If the solutions are not isolated (e.g., the floating-point system is not precise enough to separate two solutions), then the boxes returned by the algorithm may contain several solutions.

- **Termination:** **Newton** always terminates in finite time.

- **Effectiveness:** **Newton** has been evaluated on a variety of benchmarks from kinematics, chemistry, combustion, economics, and mechanics. It outperforms the interval methods we are aware of and compares well with state-of-the-art continuation methods on many problems. Interestingly, **Newton** solves the Broyden banded function problem [8] and Moré-Cosnard discretization of a nonlinear integral equation [23] for several hundred variables.

- **Simplicity and Uniformity:** **Newton** is based on simple mathematical results and is easy to use and to implement. It is also based on a single concept: box-consistency.

The rest of this paper is organized as follows. Section 2 gives an overview of the approach. Section 3 contains the preliminaries. Section 4 presents an abstract version of the branch & prune algorithm. Section 5 discusses the implementation of the box-consistency. Section 6 describes the experimental results. Section 7 discusses related work and the development of the ideas presented here. Section 8 concludes the paper.

2 Overview of The Approach

As mentioned, **Newton** is a global search algorithm which solves a problem by dividing it into subproblems which are solved recursively. In addition, **Newton** is a branch & prune algorithm which means that it is best viewed as an iteration of two steps

1. pruning the search space;

2. making a nondeterministic choice to generate two subproblems

until one or all solutions to a given problem are found.

The pruning step is responsible to make sure that some local consistency conditions are satisfied. It consists of reducing the intervals associated with the variables so that every constraint appears to be locally consistent. The local consistency condition of **Newton** is called box-consistency, an approximation of arc-consistency, a notion well-known in artificial intelligence [17, 19] and used in many systems (e.g., [33, 34, 31]) to solve discrete combinatorial search problems. Informally speaking, a constraint is arc-consistent if for each value in the range of a variable there exist values in the ranges of the other variables such that the constraint is satisfied. **Newton** approximates arc-consistency which cannot be computed on real numbers in general.

The pruning step either fails, showing the absence of solution in the intervals, or succeeds in enforcing the local consistency condition. Sometimes, local consistency also implies global consistency as in the case of the Broyden banded function, i.e.,

$$f_i(x_1, \ldots, x_n) = x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} x_j(1 + x_j) \quad (1 \leq i \leq n)$$

where $J_i = \{j \mid j \neq i \ \& \ max(1, i-5) \leq j \leq min(n, i+1)\}$. The pruning step of **Newton** solves this problem in essentially linear time for initial intervals of the form $[-10^8, 10^8]$ and always proves the existence of a solution in the final box. However, in general, local consistency does not imply global consistency either because there are multiple solutions or simply because the local consistency condition is too weak. Consider the intersection of a circle and of a parabola:

$$\begin{cases} x_1^2 + x_2^2 = 1 \\ x_1^2 - x_2 = 0 \end{cases}$$

with initial intervals in $[-10^8, 10^8]$ and assume that we want the resulting intervals to be of size 10^{-8} or smaller. The pruning step returns the intervals

$x_1 \in [-1.0000000000012430057, +1.0000000000012430057]$
$x_2 \in [-0.0000000000000000000, +1.0000000000012430057]$

Informally speaking, **Newton** obtains the above pruning as follows. The first constraint is used to reduce the interval of x_1 by searching for the leftmost and rightmost "zeros" of the interval function

$$X_1^2 + [-10^8, 10^8]^2 = 1$$

These zeros are obviously -1 and 1 and hence the new interval for x_1 becomes [-1,1] (modulo the numerical precision of the system). The same reasoning applies to x_2. The second constraint can be used to reduce further the interval of x_2 by searching for the leftmost and rightmost zeros of

$$[-1, 1]^2 - X_2 = 0$$

producing the interval [0,1] for x_2. No more reduction is obtained by **Newton** and branching is needed to make progress. Branching on x_1 produces the intervals

$x_1 \in [-1.0000000000012430057, +0.0000000000000000000]$
$x_2 \in [-0.0000000000000000000, +1.0000000000012430057]$

Further pruning is obtained by taking the Taylor extension of the polynomials in the above equations around $(-0.5, 0.5)$, the center of the above box and by conditioning the system. **Newton** then uses these new constraints to find their leftmost and rightmost zeros as was done previously to obtain.

$$x_1 \in [-1.0000000000000000000, -0.6049804687499998889]$$
$$x_2 \in [+0.3821067810058591529, +0.8433985806150308129].$$

Additional pruning using the original statement of the constraints leads to the first solution

$$x_1 \in [-0.7861513777574234974, -0.7861513777574231642]$$
$$x_2 \in [+0.6180339887498946804, +0.6180339887498950136].$$

Backtracking on the choice for x_1 produces the intervals

$$x_1 \in [-0.0000000000000000000, +1.0000000000012430057]$$
$$x_2 \in [-0.0000000000000000000, +1.0000000000012430057]$$

and to the second solution

$$x_1 \in [+0.7861513777574231642, +0.7861513777574233864]$$
$$x_2 \in [+0.6180339887498946804, +0.6180339887498950136].$$

Note that, in this case, **Newton** makes the smallest number of choices to isolate the solutions.[1] To conclude this motivating section, let us illustrate **Newton** on a larger example which describes the inverse kinematics of an elbow manipulator [11]:

$$\begin{cases} s_2c_5s_6 - s_3c_5s_6 - s_4c_5s_6 + c_2c_6 + c_3c_6 + c_4c_6 = 0.4077 \\ c_1c_2s_5 + c_1c_3s_5 + c_1c_4s_5 + s_1c_5 = 1.9115 \\ s_2s_5 + s_3s_5 + s_4s_5 = 1.9791 \\ c_1c_2 + c_1c_3 + c_1c_4 + c_1c_2 + c_1c_3 + c_1c_2 = 4.0616 \\ s_1c_2 + s_1c_3 + s_1c_4 + s_1c_2 + s_1c_3 + s_1c_2 = 1.7172 \\ s_2 + s_3 + s_4 + s_2 + s_3 + s_2 = 3.9701 \\ s_i^2 + c_i^2 = 1 \qquad (1 \leq i \leq 6). \end{cases}$$

and assumes that the initial intervals are in $[-10^8, 10^8]$ again. The pruning step returns the intervals

$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[+0.3233666666666665800, +1.0000000000000000000]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[-0.0149500000000000189, +1.0000000000000000000]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[-0.0209000000000001407, +1.0000000000000000000]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[+0.6596999999999998420, +1.0000000000000000000]$$
$$[-0.7515290480087772896, +0.7515290480087772896]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$
$$[-1.0000000000000000000, +1.0000000000000000000]$$

[1] This example can also be solved by replacing x_1^2 in the first equation by x_2 to obtain a univariate constraint in x_2 which can be solved independently. However, this cannot always be done and the discussion here is what **Newton** would do, without making such optimizations.

showing already some interesting pruning. After exactly 12 branchings and in less than a second, Newton produces the first box with a proof of existence of a solution in the box.

3 Preliminaries

In this section, we review some basic concepts needed for this paper, including interval arithmetic and the representation of constraints. More information on interval arithmetic can be found in many places (e.g., [1, 8, 7, 20, 21]). Our definitions are slightly non-standard.

3.1 Interval Arithmetic

We consider $\Re^\infty = \Re \cup \{-\infty, \infty\}$ the set of real numbers extended with the two infinity symbols and the natural extension of the relation $<$ to this set. We also consider a finite subset $\mathcal{F}$ of $\Re^\infty$ containing $-\infty, \infty, 0$. In practice, $\mathcal{F}$ corresponds to the floating-point numbers used in the implementation.

Definition 1 [Interval] An interval $[a, b]$ with $a, b \in \mathcal{F}$ is the set of real numbers

$$\{r \in \Re \mid a \leq r \leq b\}.$$

The set of intervals is denoted by $\mathcal{I}$ and is ordered by set inclusion.[2]

Definition 2 [Enclosure and Hull] Let S be a subset of $\Re$. The enclosure of S, denoted by $\overline{S}$ or $box\{S\}$, is the smallest interval I such that $S \subseteq I$. We often write $\overline{r}$ instead of $\overline{\{r\}}$ for $r \in \Re$. The interval hull of I_1 and I_2, denoted by $I_1 \uplus I_2$, is defined as $box\{I_1 \cup I_2\}$.

We denote real numbers by the letters r, v, $\mathcal{F}$-numbers by the letters a, b, l, m, u, intervals by the letter I, real functions by the letters f, g and interval functions by the letters F, G, all possibly subscripted. We use a^+ (resp. a^-) to denote the smallest (resp. largest) $\mathcal{F}$-number strictly greater (resp. smaller) than the $\mathcal{F}$-number a. To capture outward rounding, we use $\lceil r \rceil$ (resp. $\lfloor r \rfloor$) to return the smallest (resp. largest) $\mathcal{F}$-number greater (resp. smaller) or equal to the real number r. We also use $\vec{I}$ to denote a box $\langle I_1, \ldots, I_n \rangle$ and $\vec{r}$ to denote a tuple $\langle r_1, \ldots, r_n \rangle$. $\mathcal{Q}$ is the set of rational numbers and $\mathcal{N}$ is the set of natural numbers. Finally, we use the following notations.

$$\begin{aligned} &left([l, u]) = l \\ &right([l, u]) = u \\ &center([l, u]) = \lfloor (l + u)/2 \rfloor \end{aligned}$$

The fundamental concept of interval arithmetic is the notion of interval extension.

Definition 3 [Interval Extension] $F : \mathcal{I}^n \to \mathcal{I}$ is an interval extension of $f : \Re^n \to \Re$ iff

$$\forall I_1 \ldots I_n \in \mathcal{I} : r_1 \in I_1, \ldots, r_n \in I_n \Rightarrow f(r_1, \ldots, r_n) \in F(I_1, \ldots, I_n).$$

An interval relation $C : \mathcal{I}^n \to Bool$ is an interval extension of a relation $c : \Re^n \to Bool$ iff

$$\forall I_1 \ldots I_n \in \mathcal{I} : r_1 \in I_1, \ldots, r_n \in I_n \Rightarrow [c(r_1, \ldots, r_n) \Rightarrow C(I_1, \ldots, I_n)].$$

[2] Our intervals are usually called floating-point intervals in the literature.

Example 1 The interval function $\oplus$ defined as

$$[a_1, b_1] \oplus [a_2, b_2] = [\lfloor a_1 + a_2 \rfloor, \lceil b_1 + b_2 \rceil]$$

is an interval extension of addition of real numbers. The interval relation $\doteq$ defined as

$$I_1 \doteq I_2 \;\Leftrightarrow\; (I_1 \cap I_2 \neq \emptyset)$$

is an interval extension of the equality relation on real numbers.

It is important to stress that a real function (resp.) can be extended in many ways. For instance, the interval function $\oplus$ is the most precise interval extension of addition (i.e., it returns the smallest possible interval containing all real results) while a function always returning $[-\infty, \infty]$ would be the least accurate.

In the following, we assume fixed interval extensions for the basic real operators $+, -, \times$ and exponentiation (for instance, the interval extension of $+$ is defined by $\oplus$) and the basic real relations $=, \geq$. In addition, we overload the real symbols and use them for their interval extensions. Finally, we denote relations by the letter c possibly subscripted, interval relations by the letter C possibly subscripted. Note that constraints and relations are used as synonyms in this paper.

3.2 Unions of Intervals

Even though many basic real operators can be naturally extended to work on intervals, division creates problems if the interval used for dividing includes 0. A tight extension of division to intervals can be best expressed if its result is allowed to be a union of intervals [10, 6, 12]. Assuming that $c \leq 0 \leq d$ and $c < d$, $[a, b]/[c, d]$ is defined as follows:

$[\lfloor b/c \rfloor, \infty]$	if $b \leq 0$ and $d = 0$
$[-\infty, \lceil b/d \rceil] \cup [\lfloor b/c \rfloor, \infty]$	if $b \leq 0$ and $c < 0 < d$
$[-\infty, \lceil b/d \rceil]$	if $b \leq 0$ and $c = 0$
$[-\infty, \infty]$	if $a < 0 < b$
$[-\infty, \lceil a/c \rceil]$	if $a \geq 0$ and $d = 0$
$[-\infty, \lceil a/c \rceil] \cup [\lfloor b/c \rfloor, \infty]$	if $a \geq 0$ and $c < 0 < d$
$[\lfloor a/d \rfloor, \infty]$	if $a \geq 0$ and $c = 0$.

When $[c, d] = [0, 0]$, $[a, b]/[c, d] = [-\infty, \infty]$. The case where $0 \notin [c, d]$ is easy to define. Note also that other operations such as addition and subtraction can also be extended to work with unions of intervals.

A typical use of unions of intervals in interval arithmetic is the intersection of an interval I with the result of an operation of the form $I_c - I_n/I_d$ to produce a new interval I'. To increase precision, $I_c - I_n/I_d$ is computed with unions of intervals producing a result of the form I_1 or $I_1 \cup I_2$. This result should be intersected with I to produce a single interval as result (i.e., not a union of intervals). A generalized intersection operation $\sqcap$ defined as

$$(I_1 \cup \ldots \cup I_n) \sqcap (I'_1 \cup \ldots \cup I'_m) = (I_1 \cap I'_1) \uplus (I_1 \cap I'_2) \uplus \ldots \uplus (I_n \cap I'_m)$$

is used for this purpose, i.e., we compute

$$I \sqcap (I_c - I_n/I_d).$$

3.3 Constraint Representations

It is well-known that different computer representations of a real function produce different results when evaluated with floating-point numbers on a computer. As a consequence, the way constraints are written may have an impact on the behaviour on the algorithm. For this reason, a constraint or a function in this paper is considered to be an expression written in some formal language by composing real variables, rational numbers, some predefined real numbers (e.g., π), arithmetic operations such as $+, -, \times$ and exponentiation to a natural number, parentheses, and relation symbols such as $\leq, =$.[3] We will abuse notation by denoting functions (resp. constraints) and their representations by the same symbol. Real variables in constraints will be taken from a finite (but arbitrary large) set $\{x_1, \ldots, x_n\}$, the set of all real functions is denoted by *Function*, and the set of all real constraints is denoted by *Constraint*. Similar conventions apply to interval functions and constraints. Interval variables will be taken from a finite (but arbitrary large) set $\{X_1, \ldots, X_n\}$, interval functions will be denoted by the letter F and interval constraints by the letter C. The set of all interval functions is denoted by *FUNCTION* while the set of all interval constraints is denoted by *CONSTRAINT*. For simplicity of exposition, we restrict attention to equations. It is straightforward to generalize our results to inequalities (see Section 5.6).

4 The Branch & Prune Algorithm

This section describes the branch & prune algorithm **Newton**. Section 4.1 defines box-consistency, the key concept behind our algorithm. Section 4.2 shows how box-consistency can be instantiated to produce various pruning operators achieving various tradeoffs between accuracy and efficiency. Section 4.3 defines a conditioning operator used in **Newton** to improve the effectiveness of box-consistency. Section 4.4 specifies the pruning in **Newton**. Section 4.5 describes the algorithm. Recall that we assume that all constraints are defined over variables $x_1, \ldots, x_n$.

4.1 Box Consistency

Box-consistency [2] is an approximation of arc-consistency, a notion well-known in artificial intelligence [17] which states a simple local condition on a constraint c and the set of possible values for each of its variables, say $D_1, \ldots, D_n$. Informally speaking, a constraint c is arc-consistent if none of the D_i can be reduced by using projections of c.

Definition 4 [Projection Constraint] A projection constraint $\langle c, i \rangle$ is the association of a constraint c and of an index i ($1 \leq i \leq n$). Projection constraints are denoted by the letter p, possibly subscripted.

Example 2 Consider the constraint $x_1^2 + x_2^2 = 1$. Both $\langle x_1^2 + x_2^2 = 1\,,\, 1 \rangle$ and $\langle x_1^2 + x_2^2 = 1\,,\, 2 \rangle$ are projection constraints.

Definition 5 [Arc-Consistency] A projection constraint $\langle c, i \rangle$ is arc-consistent wrt $\langle D_1, \ldots, D_n \rangle$ iff $D_i = D_i \cap \{r_i \mid \exists r_1 \in D_1, \ldots, \exists r_{i-1} \in D_{i-1}, \ldots, \exists r_{i+1} \in D_{i+1}, \ldots, \exists r_n \in D_n : c(r_1, \ldots, r_n)\}$. A constraint c is arc-consistent wrt $\langle D_1, \ldots, D_n \rangle$ if each of its projections is arc-consistent wrt $\langle D_1, \ldots, D_n \rangle$. A system of constraints $\mathcal{S}$ is arc-consistent wrt $\langle D_1, \ldots, D_n \rangle$ if each constraint in $\mathcal{S}$ is arc-consistent wrt $\langle D_1, \ldots, D_n \rangle$.

[3] It is easy to extend the language to include functions such as *sin, cos, e, ...*.

Example 3 Let c be the constraint $x_1^2 + x_2^2 = 1$. c is arc-consistent wrt $\langle\ [-1,1]\ ,\ [-1,1]\ \rangle$ but is not arc-consistent wrt $\langle\langle\ [-1,1]\ ,\ [-2,2]\ \rangle$ since, for instance, there is no value r_1 for x_1 in $[-1,1]$ such that $r_1^2 + 2^2 = 1$.

Arc-consistency cannot be computed in general when working with real numbers and polynomial constraints and simple approximations to capture machine precision are very expensive to compute. For instance, a simple approximation of arc-consistency consists in working with intervals and approximating the set computed by arc-consistency to return an interval, i.e.,

$$I_i = box\{I_i \cap \{\ r_i\ \mid\ \exists r_1 \in I_1, \ldots, \exists r_{i-1} \in I_{i-1}, \ldots, \exists r_{i+1} \in I_{i+1}, \ldots, \exists r_n \in I_n : c(r_1, \ldots, r_n)\ \}\}.$$

This condition, used in systems like [27, 3], is easily enforced on simple constraints such as

$$x_1 = x_2 + x_3,\ \ x_1 = x_2 - x_3,\ \ x_1 = x_2 \times x_3$$

but it is also computationally very expensive for complex constraints with multiple occurrences of the same variables. Moreover, decomposing complex constraints into simple constraints entails a substantial loss in pruning, making this approach unpractical on many applications. See [2] for experimental results on this approach and their comparison with the approach presented in this paper.

The notion of box-consistency introduced in [2] is a coarser approximation of arc-consistency which provides a much better trade-off between efficiency and pruning. It consists in replacing the existential quantification in the above condition by the evaluation of an interval extension of the constraint on the intervals of the existential variables. Since there are many interval extensions for a single constraint, we define box-consistency in terms of interval constraints.

Definition 6 [Interval Projection Constraint] An interval projection constraint $\langle C, i\rangle$ is the association of an interval constraint C and of an index i $(1 \leq i \leq n)$. Interval projection constraints are denoted by the letter P, possibly subscripted.

Definition 7 [Box-Consistency] An interval projection constraint $\langle C, i\rangle$ is box-consistent wrt $\vec{I} = \langle I_1, \ldots, I_n\rangle$ iff

$$C(I_1, \ldots, I_{i-1}, [l, l^+], I_{i+1}, \ldots, I_n)\ \wedge\ C(I_1, \ldots, I_{i-1}, [u^-, u], I_{i+1}, \ldots, I_n).$$

where $l = left(I_i)$ and $u = right(I_i)$ An interval constraint is box-consistent wrt $\vec{I}$ if each of its projections is box-consistent wrt $\vec{I}$. A system of interval constraints is box-consistent wrt $\vec{I}$ iff each interval constraint in the system is box-consistent wrt $\vec{I}$.

Intuitively speaking, the above condition states that the ith interval cannot be pruned further using the unary interval constraint obtained by replacing all variables but X_i by their intervals, since the boundaries satisfy the unary constraint. Note also that the above condition is equivalent to

$$I_i = box\{\ r_i\ \in\ I_i \mid C(I_1, ..., I_{i-1}, \overline{r_i}, I_{i+1}, ..., I_n\ \}$$

which shows clearly that box-consistency is an approximation of arc-consistency.[4] The difference between arc-consistency and box-consistency appears essentially when there are multiple occurrences of the same variable.

[4]It is interesting to note that this definition is also related to the theorem of Miranda [26]. In this case, box-consistency can be seen as replacing universally quantified variables by the intervals on which they range.

Example 4 Consider the constraint $x_1 + x_2 - x_1 = 0$. The constraint is not arc-consistent wrt $\langle[-1,1],[-1,1]\rangle$ since there is no value r_1 for x_1 which satisfies $r_1 + 1 - r_1 = 0$. On the other hand, the interval constraint $X_1 + X_2 - X_1 = 0$ is box-consistent, since $([-1,1]+[-1,-1^+]-[-1,1]) \cap [0,0]$ and $([-1,1]+[1^-,1]-[-1,1]) \cap [0,0]$ are non-empty.

4.2 Interval Extensions for Box Consistency

Box-consistency strongly depends on the interval extensions chosen for the constraints and different interval extensions can produce very different (often incomparable) tradeoffs between pruning and computational complexity. In this section, we consider three extensions used in **Newton**: natural interval extension, distributed interval extension, and Taylor interval extension.

4.2.1 Natural Interval Extension

The simplest extension of a function (resp. of a constraint) is its natural interval extension. Informally speaking, it consists in replacing each number by the smallest interval enclosing it, each real variable by an interval variable, each real operation by its fixed interval extension and each real relation by its fixed interval extension.

Example 5 [Natural Interval Extension] The natural interval extension of the function $x_1(x_2+x_3)$ is the interval function $X_1(X_2+X_3)$. The natural interval extension of the constraint $x_1(x_2+x_3) = 0$ is the interval constraint $X_1(X_2 + X_3) = \overline{0}$.

The advantage of this extension is that it preserves the way constraints are written and hence users of the system can choose constraint representations particularly appropriate for the problem at hand. A very nice application where this extension is fundamental is the Moré-Cosnard discretization of a nonlinear integral equation (See Section 6.2). Using the natural extension allows users to minimize the problem of dependency of interval arithmetic and hence to increase precision. In the following, if f (resp. c) is a real function (resp. constraint), we denote by $\widehat{f}$ (resp. $\widehat{c}$) its natural extension.

4.2.2 Distributed Interval Extension

The second interval extension used by **Newton** does not preserve the way constraints are written but uses a distributed form of the constraints. The key advantage of this extension is that it allows the algorithm to enforce box-consistency by applying interval Newton method on univariate real functions. The real functions are derived from univariate interval constraints obtained by replacing all but one variable by their intervals. As a consequence, applying box-consistency will be particularly efficient, although the pruning may be weaker than for the natural extension due to the dependency problem of interval arithmetics.[5] Intuitively, the distributed interval extension should be viewed as a way to speed up the computation of box-consistency on the natural extension. However, it may happen that it gives more precision than the natural extension if users are not careful in stating their constraints.

[5]Note that it is not always necessary to go through the distributed form to obtain the above property but **Newton** adopts it for simplicity.

Definition 8 [Distributed Form] A constraint c in (simplified) sum of products form

$$m_1 + \cdots + m_k = 0,$$

where each monomial m_i is of the form $qx_1^{e_1} \cdots x_n^{e_n}$ with $q \in \mathcal{Q}$ and $e_i \in \mathcal{N}$, is said to be in distributed form.[6]

Definition 9 [Distributed Interval Extension] The distributed interval extension of a function f (resp. constraint c) is the natural extension of its distributed form.

Example 6 [Distributed Interval Extension] The distributed interval extension of the function $x_1(x_2 + x_3)$ is the interval function $X_1X_2 + X_1X_3$. The distributed interval extension of the constraint $x_1(x_2 + x_3) = 0$ is the interval constraint $X_1X_2 + X_1X_3 = \overline{0}$.

In the following, the distributed interval extension of a function f (resp. of a constraint c) is denoted by $\widetilde{f}$ (resp. $\widetilde{c}$).

4.2.3 Taylor Interval Extension

The last interval extension we introduce is based on the Taylor expansion around a point. This extension is an example of centered forms which are interval extensions introduced by Moore [20] and studied by many authors, since they have important properties. The Taylor interval extension of a constraint is parametrized by the intervals for the variables in the constraint. It also assumes that the constraint which it is applied to is of the form $f = 0$ where f denotes a function which has continuous partial derivatives. Given these assumptions, the key idea behind the extension is to apply a Taylor expansion of the function around the center of the box and to bound the rest of the series using the box.

Definition 10 [Taylor Interval Extension] Let c be a constraint $f = 0$, f be a function with continuous partial derivatives, $\vec{I}$ be a box $(I_1, \ldots, I_n)$, and m_i be the center of I_i. The Taylor interval extension of c wrt $\vec{I}$, denoted by $c^{t(\vec{I})}$, is the interval constraint

$$\widehat{f}(\overline{m_1}, \ldots, \overline{m_n}) + \sum_{i=1}^{n} \widehat{\frac{\partial f}{\partial x_i}}(I_1, \ldots, I_n)\ (X_i - \overline{m_i}) \quad = \quad \overline{0}.$$

In the current version of our system, the partial derivatives are computed using automatic differentiation.

4.3 Conditioning

It is interesting to note that box-consistency on the Taylor interval extension is closely related to Hansen-Segupta's operator [8], which is an improvement over Krawczyk's operator [15]. Hansen and Smith [9] also argued that these operators are more effective for a system $\{f_1 = 0, \ldots, f_n = 0\}$ wrt a box $\langle I_1, \ldots, I_n \rangle$ when the interval Jacobian

$$M_{ij} = \widehat{\frac{\partial f_i}{\partial x_j}}(I_1, \ldots, I_n)\ (1 \leq i, j \leq n)$$

[6] The distributed version can easily be turned into a canonical representation for constraints.

is diagonally dominant, i.e.,

$$mig(M_{i,i}) \geq \sum_{j=1, j\neq i}^{n} mag(M_{i,j})$$

where

$$mig([l,u]) = min(|\ l\ |, |\ u\ |) \text{ and } mag([l,u]) = max(|\ l\ |, |\ u\ |).$$

They also suggest a conditioning which consists in multiplying the linear relaxation by a real matrix which is the inverse of the matrix obtained by taking the center of $M_{i,j}$. The resulting system is generally solved through Gauss-Seidel iterations, giving Hansen-Segupta's operator. See also [13, 14] for an extensive coverage of conditioners). **Newton** exploits this idea to improve the effectiveness of box-consistency on the Taylor interval extension. The conditioning of **Newton** is abstracted by the following definition.

Definition 11 [Conditioning] Let $S = \{f_1 = 0, \ldots, f_n = 0\}$. A conditioning of S is a system $S' = \{f'_1 = 0, \ldots, f'_n = 0\}$ where

$$f'_i = \sum_{k=1}^{n} A_{ik} f_k$$

where $A_{ik} \in \mathcal{Q}$.

In its present implementation, **Newton** uses a conditioning $cond(\{f_1 = 0, \ldots, f_n = 0\}, \vec{I})$ which returns a system $\{f'_1 = 0, \ldots, f'_n = 0\}$ such that

$$f'_i = \sum_{k=1}^{n} A_{ik} f_k$$

where

$$\begin{array}{lcl} M_{ij} & = & \widehat{\frac{\partial f_i}{\partial x_j}}(I_1, \ldots, I_n)\ (1 \leq i, j \leq n) \\ B_{ij} & = & center(M_{ij}) \\ A & = & \begin{cases} B^{-1} & \text{if } B \text{ is not singular} \\ I & \text{otherwise.} \end{cases} \end{array}$$

Note that the computation of the inverse of B is obtained by standard floating-point algorithms and hence it is only an approximation of the actual inverse.

4.4 Pruning in Newton

We now describe the pruning of **Newton**. The key idea behind **Newton** is to apply box-consistency at each node of the search tree, i.e., **Newton** reduces the current intervals for the variables in such a way that the constraint system is box-consistent wrt the reduced intervals and no solution is removed. The pruning is performed by using narrowing operators deduced from the definition of box-consistency. These operators are used to reduce the interval of a variable using a projection constraint.

Definition 12 [Box-Narrowing] Let $\langle C, i\rangle$ be an interval projection constraint and $\langle I_1, \ldots, I_n\rangle$ be a box. The narrowing operator **BOX-NARROW** is defined as

BOX-NARROW($\langle C, i\rangle$,$\langle I_1, \ldots, I_n\rangle$) = $\langle I_1, \ldots, I_{i-1}, I, I_{i+1}, \ldots, I_n\rangle$

where I is defined as the largest set included in I_i such that $\langle C, i\rangle$ is box-consistent with respect to $\langle I_1, \ldots, I_{i-1}, I, I_{i+1}, \ldots, I_n\rangle$.

Proposition 1 [Soundness of the Box-Narrowing] Let C be an interval extension of c, $\langle C, i\rangle$ be an interval projection constraint, $\langle I_1, \ldots, I_n\rangle$ be a box, and

$$\langle I_1, \ldots, I_{i-1}, I, I_{i+1}, \ldots, I_n\rangle = \text{BOX-NARROW}(\langle C, i\rangle, \langle I_1, \ldots, I_n\rangle).$$

Then

$$r_1 \in I_1, \ldots, r_n \in I_n \;\&\; c(r_1, \ldots, r_n) \;\Rightarrow\; r_i \in I$$

Proof Assume that $r_1 \in I_1, \ldots, r_n \in I_n$ and that $r_i \notin I$. Then either $r_i < l$ or $r_i > u$, where $I = [l, u]$. If $r_i < l$, we have that

$$C(I_1, \ldots, I_{i-1}, \overline{r_i}, I_{i+1}, \ldots, I_n)$$

by definition of an interval extension and

$$C(I_1, \ldots, I_{i-1}, [u^-, u], I_{i+1}, \ldots, I_n)$$

by hypothesis. Hence, C is box-consistent wrt $\langle I_1, \ldots, I_{i-1}, I \uplus \overline{r_i}, I_{i+1}, \ldots, I_n\rangle$, which contradicts our hypothesis that I is defined as the largest set included in I_i such that $\langle C, i\rangle$ is box-consistent with respect to $\langle I_1, \ldots, I_{i-1}, I, I_{i+1}, \ldots, I_n\rangle$. The case $r_i > u$ is similar. □

We are now in a position to define the pruning algorithm of **Newton** which consists essentially in applying the narrowing operators of each projection until no further reduction occurs. The pruning algorithm is depicted in Figure 1. It first applies box-consistency on the natural and distributed extensions until no further reduction occurs and then applies box-consistency on the Taylor extension. The two steps are iterated until a fixpoint is reached. Termination of the algorithm is guaranteed since the set $\mathcal{F}$ is finite and thus the intervals can only be reduced finitely often.

4.5 The Branch and Prune Algorithm Newton

Figure 2 is a very high level description of the branch and prune algorithm highlighting the control flow. The algorithm applies operation PRUNE on the initial box. If the resulting box is empty (which means that one of its components is empty), then there is no solution by Proposition 1. If the resulting box is small enough (specified by the desired accuracy in solutions), then it is included in the result. The function BRANCH splits the box into two subboxes along one dimension (variable). Variables for splitting are chosen by BRANCH using a round-robin heuristic: if $\{x_1, \ldots, x_n\}$ is the set of variables, then the algorithm splits the variables in the order $x_1, x_2, \ldots, x_n$ and reiterates the process until a solution is found.

```
procedure PRUNE(in S: Set of Constraint; inout Ī : I^n)
begin
    repeat
        Ī_p := Ī;
        BOX-PRUNE({⟨ĉ, i⟩ | c ∈ S & 1 ≤ i ≤ n} ∪ {⟨c̃, i⟩ | c ∈ S & 1 ≤ i ≤ n}, Ī);
        BOX-PRUNE({⟨c^t(Ī), i⟩ | c ∈ cond(S, Ī) & 1 ≤ i ≤ n}, Ī);
    until Ī = Ī_p;
end

procedure BOX-PRUNE(in P: Set of Interval Projection Constraint; inout Ī : I^n)
begin
    repeat
        Ī_p := Ī;
        Ī := ⋂{BOX-NARROW(P, Ī) | P ∈ P & 1 ≤ i ≤ n }
    until Ī = Ī_p;
end
```

Figure 1: Pruning in Newton

```
function BranchAndPrune(S: Set of Constraint; Ī_0 : I^n): Set of I^n;
begin
   Ī := PRUNE(S, Ī_0);
   if ¬ IsEmpty(Ī) then
      if IsSmallEnough(Ī) then
         return {Ī}
      else
         ⟨Ī_1, Ī_2⟩ := BRANCH(Ī);
         return BranchAndPrune(S, Ī_1) ∪ BranchAndPrune(S, Ī_2)
      endif
   else
      return ∅
   endif
end
```

Figure 2: The Branch and Prune Algorithm

```
function LNAR(F, F' : I → I, I : I): I;
begin
   r := right(I);
   if 0 ∉ F(I) then
      return ∅;
   I := N*(F, F', I);
   if 0 ∈ F([left(I), left(I)+]) then
      return I ⊎ r̄
   else if
      ⟨I1, I2⟩ := SPLIT(I);
      if LNAR(F, F', I1) ≠ ∅ then
         return LNAR(F, F', I1) ⊎ r̄
      else
         return LNAR(F, F', I2) ⊎ r̄
   endif
end;
```

Figure 3: The function LNAR

5 Implementation of Box-Consistency

Figure 2 is a precise description of the branch and prune algorithm which leaves open the implementation of procedure BOX-NARROW. The purpose of this section is to describe how this procedure is implemented in Newton. The basic idea of the implementation is to use a different implementation of procedure BOX-NARROW for each interval extension in order to exploit their specific properties. We thus present three procedures in the section: BOX-NARROW-NE, BOX-NARROW-DE, and BOX-NARROW-TE. In addition, it is more convenient to define them in terms of projection constraints (instead of in terms of interval projection constraints).[7] The rest of this section is organized as follows. We start by describing a basic tool used in the implementations, then describe the various narrowing operators, discuss how to prove the existence of solution, and conclude by some implementation issues.

5.1 Extreme Zeros of an Interval Function

Box-consistency can often be reduced to two subproblems which, informally speaking, consist in shrinking the left (resp. the right) of an interval I' to the leftmost (resp. rightmost) zero of a univariate interval function F in I'. The univariate interval function F is an extension of a real function f which is either univariate, in which case F is a traditional interval extension, or multivariate, in which case F is obtained by taking an interval extension of f and substituting all variables but one, say x_i, by their intervals. In addition, we will have at our disposal a function F', the "derivative" of F, which is either an interval extension of the derivative of f (univariate case) or an interval extension of the partial derivative of f wrt x_i in which all variables but x_i have been replaced by their intervals.

[7] There is no difficult in modifying the algorithm of Figure 2 to accommodate this change.

The subproblems can be computed using a variation of the univariate interval Newton method. The method uses the following property for pruning the search space

$$0 \in F(I) \Rightarrow 0 \in N(F, F', I)$$

where

$$N(F, F', I) = I \sqcap [\overline{center(I)} - \frac{F(\overline{center(I)})}{F'(I)}].$$

More precisely, the algorithm uses $N^*(F, F', I) = \bigcap_{i=0}^{\infty} I_i$ where

$$\begin{array}{lcl} I_0 & = & I \\ I_{i+1} & = & N(F, F', I_i) \ (0 \leq i) \end{array}$$

Figure 3 depicts a simple function LNAR to shrink to the leftmost zero (function RNAR is defined similarly). It make uses of a function SPLIT which, given an interval I, returns two intervals I_1 and I_2 such that $I = I_1 \cup I_2$ and $left(I_2) = right(I_1)$. The algorithm first applies the pruning procedure. It terminates if the left zero has been found or the interval cannot contain a zero. Otherwise, the interval is split into two subintervals. The leftmost interval is explored first. If it does not contain a zero, the rightmost interval is explored. It is worthwhile mentioning that the reduction on the right bound must not be used as part of the result, since the function would not meet its specification in this case. Function RNAR is responsible for finding the rightmost zero.

5.2 Box-Consistency on the Natural Interval Extension

We are now in a position to define the narrowing operator for the natural interval extension. The basic idea is very simple. For a constraint $\langle f = 0, i \rangle$, it consists in taking an interval extension of f and replacing all variables but x_i by their interval I_i to obtain a univariate function. The leftmost and rightmost zeros are then computed to produce the result.

Definition 13 [Narrowing Operator for the Natural Interval Extension] The narrowing operator for the natural interval extension is defined as follows:

$$\texttt{BOX-NARROW-NE}(\langle f = 0, i \rangle, \langle I_1, \ldots, I_n \rangle) = \langle I_1, \ldots, I_{i-1}, I, I_{i+1}, \ldots, I_n \rangle$$

where

$$\begin{array}{lcl} I & = & \texttt{LNAR}(F, F', \texttt{RNAR}(F, F', I_i)) \\ F(X) & = & \widehat{f}(I_1, \ldots, I_{i-1}, X, I_{i+1}, \ldots, I_n) \\ F'(X) & = & \widehat{\frac{\partial f}{\partial i}}(I_1, \ldots, I_{i-1}, X, I_{i+1}, \ldots, I_n) \end{array}$$

Example 7 Let c be the constraint $x_1^2 + x_2^2 - 1 = 0$ and $\vec{I}$ be $\langle [-1, 1], [-1, 1] \rangle$. The function F and F' for $i = 1$ in the above definition are defined as follows:

$$\begin{array}{l} F(X) = X^2 + [-1, 1]^2 - 1. \\ F'(X) = 2X. \end{array}$$

It is important to note that box-consistency on the natural extension (and on the distributed extension as well) can be applied even if the function is not differentiable. It suffices to omit the application of operator N^* in the functions LNAR and RNAR.

5.3 Box-Consistency on the Distributed Interval Extension

We now turn to the narrowing operator for the distributed interval extension. Box-consistency on the distributed interval extension can be enforced by using the univariate interval function

$$F(X) = \widetilde{f}(I_1, \ldots, I_{i-1}, X, I_{i+1}, \ldots, I_n)$$

and by searching its extreme zeros. However, since the function is in distributed form, it is possible to do better by using an idea from [11]. The key insight is to sandwitch F exactly between two univariate real functions f_l and f_u defined as follows:

$$\begin{array}{rcl} f_l(x) & = & \mathit{left}(F(\overline{x})) \\ f_u(x) & = & \mathit{right}(F(\overline{x})). \end{array}$$

Note that box-consistency can now be enforced by searching the leftmost and rightmost zeros of some interval extensions, say F_l and F_u, of f_l and f_u using interval extensions, say F_l' and F_u', of their derivatives f_l' and f_u'. Of course, LNAR and RNAR can be used to find these extreme zeros.

The key advantage of the distributed interval extension is that it is easy to define the function f_l and f_u constructively. Let F be of the form

$$F(X) = I_1 X^{n_1} + \ldots + I_p X^{n_p}.$$

Function f_l is defined as

$$f_l(x) = \mathit{low}(I_1, x, n_1) + \ldots + \mathit{low}(I_p, x, n_p)$$

where

$$\mathit{low}(I, x, n) = \begin{cases} \mathit{left}(I)\ x^n & \text{if } x \geq 0 \vee n \text{ is even} \\ \mathit{right}(I)\ x^n & \text{otherwise} \end{cases}$$

Function f_u is defined as

$$f_u(x) = \mathit{high}(I_1, x, n_1) + \ldots + \mathit{high}(I_p, x, n_p)$$

where

$$\mathit{high}(I, x, n) = \begin{cases} \mathit{right}(I)\ x^n & \text{if } x \geq 0 \vee n \text{ is even} \\ \mathit{left}(I)\ x^n & \text{otherwise} \end{cases}$$

It is easy to see that the two definitions of f_l and f_u are similar.

Example 8 Consider the function $x_1(x_1 * x_2) - 4$ and assume that x_1 and x_2 range over $[0, 1]$. The distributed interval extension is

$$X_1^2 + X_1 X_2 - 4.$$

The function F obtained by projecting the distributed interval extension on variable X_1 is

$$F(X) = X^2 - [0, 1]X - 4.$$

The corresponding functions f_l and f_u are

$$f_u(x) = x^2 - 4$$
$$f_l(x) = x^2 - x - 4.$$

Their natural interval extensions are of course

$$F_u(X) = X^2 - 4$$
$$F_l(X) = X^2 - X - 4.$$

The narrowing operator can now be obtained by using interval Newton method on the above two functions. Some care must be applied obviously since f_l and f_u are not differentiable at 0. The method is more efficient than applying the interval Newton method on the interval function, since intervals have been replaced by numbers increasing the precision of the Newton operator N^*. We now present the narrowing operator.

Definition 14 [Narrowing Operator for the Distributed Interval Extension] Let $\langle f = 0, i\rangle$ be a projection constraint, let F, f_l, and f_u be the functions

$$\begin{array}{lcl} F(X) & = & \widetilde{f}(I_1, \ldots, I_{i-1}, X, I_{i+1}, \ldots, I_n) \\ f_l(x) & = & \mathit{left}(F(\overline{x})) \\ f_u(x) & = & \mathit{right}(F(\overline{x})) \\ F_l & = & \widehat{f_l} \\ F_u & = & \widehat{f_u} \\ F_l' & = & \widehat{f_l'} \\ F_u' & = & \widehat{f_u'} \\ l & = & \mathit{left}(I_i) \\ u & = & \mathit{right}(I_i) \end{array}$$

Assuming that $\langle I_1, \ldots, I_n\rangle$ is not empty, the narrowing operator for the distributed interval extension is defined as follows:

$\texttt{BOX-NARROW-DE}(\langle f = 0, i\rangle, \langle I_1, \ldots, I_n\rangle) = \langle I_1, \ldots, I_{i-1}, [l_i, u_i], I_{i+1}, \ldots, I_n\rangle$
where

$$l_i = \begin{cases} l & \texttt{if } 0 \in F([l, l^+]) \\ \mathit{left}(\texttt{LNAR}(F_l, F_l', [l, 0]) \cup \texttt{LNAR}(F_l, F_l', [0, u])) & \texttt{if } F([l, l^+]) > 0 \\ \mathit{left}(\texttt{LNAR}(F_u, F_u', [l, 0]) \cup \texttt{LNAR}(F_u, F_u', [0, u])) & \texttt{otherwise} \end{cases}$$

$$u_i = \begin{cases} u & \texttt{if } 0 \in F([u^-, u]) \\ \mathit{right}(\texttt{RNAR}(F_l, F_l', [l, 0]) \cup \texttt{RNAR}(F_l, F_l', [0, u])) & \texttt{if } F([l, l^+]) > 0 \\ \mathit{right}(\texttt{RNAR}(F_u, F_u', [l, 0]) \cup \texttt{RNAR}(F_u, F_u', [0, u])) & \texttt{otherwise.} \end{cases}$$

5.4 Taylor Interval Extension

We conclude by presenting the narrowing operator for the Taylor interval extension. Box-consistency on the Taylor interval extension can be enforced by using the interval function

$$F(X) = f^{t(\langle I_1, \ldots, I_n\rangle)}(I_1, \ldots, I_{i-1}, X, I_{i+1}, \ldots, I_n)$$

and by applying a simple implementation of LNAR and RNAR where the Newton operator N^* is omitted. However, it is possible to do better by noticing that the constraint is of the form

$$\widehat{f}(\overline{m_1},\ldots,\overline{m_n}) + \sum_{j=1}^{i-1} \frac{\widehat{\partial f}}{\partial j}(\vec{I})(I_j - \overline{m_j}) + \frac{\widehat{\partial f}}{\partial i}(I_1,\ldots,I_n)(X_i - \overline{m_i}) + \sum_{j=i+1}^{n} \frac{\widehat{\partial f}}{\partial j}(\vec{I})(I_j - \overline{m_j}) = \overline{0}$$

where $m_i = center(I_i)$ and contains a single variable which can be isolated to compute box-consistency directly.

Definition 15 [Narrowing Operator for the Taylor Interval Extension] The narrowing operator for the Taylor interval extension is defined as follows:

$$\texttt{BOX-NARROW-TE}(\langle f = 0, i\rangle, \langle I_1,\ldots,I_n\rangle) = \langle I_1,\ldots,I_{i-1},I,I_{i+1},\ldots,I_n\rangle$$

where

$$I = I_i \cap (\overline{m_i} - \frac{1}{\frac{\widehat{\partial f}}{\partial i}(I_1,\ldots,I_n)}[\sum_{j=1,j\neq i}^{n} \frac{\widehat{\partial f}}{\partial j}(I_1,\ldots,I_n)\ (I_j - \overline{m_j}) + \widehat{f}(\overline{m_1},\ldots,\overline{m_n})]).$$

and $m_i = center(I_i)$.

5.5 Existence of Solution

We now briefly describe how Newton proves the existence of solutions. No special effort has been devoted to this topic and the techniques could certainly be improved in various ways. Let $\{f_1 = 0,\ldots,f_n = 0\}$ be a conditioned system of equations over variables $\{x_1,\ldots,x_n\}$, let $\langle I_1,\ldots,I_n\rangle$ be a box and define the intervals I_i' $(1 \leq i \leq n)$ as follows:

$$I_i' = (\overline{m_i} - \frac{1}{\frac{\widehat{\partial f_i}}{\partial i}(I_1,\ldots,I_n)}[\sum_{j=1,j\neq i}^{n} \frac{\widehat{\partial f_i}}{\partial j}(I_1,\ldots,I_n)\ (I_j - \overline{m_j}) + \widehat{f_i}(\overline{m_1},\ldots,\overline{m_n})]).$$

and $m_i = center(I_i)$. If

$$\langle I_1',\ldots,I_n'\rangle \subseteq \langle I_1,\ldots,I_n\rangle$$

then there exists a unique zero in $\langle I_1',\ldots,I_n'\rangle$. A proof of this result can be found in [26] where credit is given to Moore and Nickel. Note also the intervals I_i' have to be computed for box-consistency on the Taylor interval extension.

5.6 Implementation Issues

We now review some implementation issues which arise in programming the algorithm.

Priorities In the pruning algorithm, it is important for efficiency reasons to use a priority queue to ensure that projections over the distributed interval extension be selected before projections over the natural interval extension. Newton also does not enforce box-consistency on the distributed version whenever it is believed to lose too much precision (e.g., an expression raised to some power).

Precision In practice, it is often sufficient to return intervals whose widths[8] are within the desired accuracy instead of returning intervals of the form $[l, l^+]$. It is easy to modify the BRANCH operation to split only intervals whose widths is above the required accuracy. Our system allows users to specify the accuracy.

Improvement Factor Box-consistency can sometimes take much time to remove small parts of the intervals. In these cases, it is probably more cost-effective to branch. Once again, it is easy to modify the algorithm to avoid this problem by making sure that the narrowing operators do not update the intervals unless some significant reduction has taken place. Since the notion of significant reduction may be problem-dependent, our system lets users specify the improvement factor necessary to update an interval in a projection.

Automatic Differentiation As mentioned, our algorithm takes a very simple approach to obtain partial derivatives, i.e., no effort is spent in factoring common expressions to reduce the dependency problem of interval arithmetic. The main reason comes from the fact that we are using automatic differentiation [28] to evaluate the derivatives together with the functions. This choice may be reconsidered in a further version of the system.

Inequalities It is simple to generalize the above algorithms for inequalities. In general, it suffices to test if the inequality is satisfied at the end points of the interval. If it is not, then the problem reduces once again to finding the leftmost and/or rightmost zeros.

6 Experimental Results

This section reports experimental results of Newton on a variety of standard benchmarks. The benchmarks were taken from papers on numerical analysis [23], interval analysis [8, 11, 22], and continuation methods [35, 25, 24, 18]. We also compare Newton with a traditional interval method using Hansen-Segupta's operator, range testing, and branching. This method uses the same implementation technology as Newton and is denoted by HRB in the following.[9] Finally, we compare Newton with a state-of-the-art continuation method [35], denoted by CONT in the following. Note that all results given in this section were obtained by running Newton on a Sun Sparc 10 workstation to obtain all solutions. In addition, the final intervals must have widths smaller than 10^{-8} and Newton always uses an improvement factor of 10%. The results are summarized in Table 1. For each benchmark, we give the number of variables (n), the total degree of the system (d), the initial range for the variables, and the results of each method in seconds. Note that the times for the continuation method are on a DEC 5000/200. A space in a column means that the result is not available for the method. A question mark means that the method does not terminate in a reasonable time ($>$ 1 hour). The rest of the section describes each benchmark and the results in much more detail. For each benchmark, we report the CPU times in seconds, the growth of the CPU time, the number of branch operations *branching*, the number of narrowings on the various

[8] The width of $[l, u]$ is $u - l$.

[9] Some interval methods such as [7] are more sophisticated than HRB but the sophistication aims at speeding up the computation near a solution. Our main contribution is completely orthogonal and aims at speeding up the computation when far from a solution and hence comparing it to HRB is meaningful.

Benchmarks	v	d	range	Newton	HRB	CONT
Broyden	10	3^{10}	[-1,1]	1.65	18.23	
Broyden	20	3^{20}	[-1,1]	4.25	?	
Broyden	320	3^{320}	[-1,1]	113.71	?	
Broyden	320	3^{320}	$[-10^8, 10^8]$	143.40	?	
Moré-Cosnard	20	3^{20}	$[-4, 5]$	24.49	968.25	
Moré-Cosnard	40	3^{40}	$[-4, 5]$	192.81	?	
Moré-Cosnard	80	3^{80}	$[-4, 5]$	1752.64	?	
Moré-Cosnard	80	3^{80}	$[-10^8, 0]$	1735.09	?	
i1	10	3^{10}	[-2,2]	0.06	14.28	
i2	20	3^{20}	[-1,2]	0.30	1821.23	
i3	20	3^{20}	[-2,2]	0.31	5640.80	
i4	10	6^{10}	[-1,1]	73.94	445.28	
i5	10	11^{10}	[-1,1]	0.08	33.58	
kin1	12	4608	$[-10^8, 10^8]$	14.24	1630.08	
kin2	8	256	$[-10^8, 10^8]$	353.06	4730.34	35.61
eco	4	18	$[-10^8, 10^8]$	0.60	2.44	1.13
eco	5	54	$[-10^8, 10^8]$	3.35	29.88	5.87
eco	6	162	$[-10^8, 10^8]$	22.53	?	50.18
eco	7	486	$[-10^8, 10^8]$	127.65	?	991.45
eco	8	1458	$[-10^8, 10^8]$	915.24	?	
eco	9	4374	$[-10^8, 10^8]$	8600.28	?	
combustion	10	96	$[-10^8, 10^8]$	9.94	?	57.40
chemistry	5	108	$[0, 10^8]$	6.32	?	56.55
neuro	6	1024	$[-10, 10]$	0.91	28.84	5.02
neuro	6	1024	$[-1000, 1000]$	172.71	?	5.02

Table 1: Summary of the Experimental Results

extensions *na-ne, na-ee, na-te*, the total number of narrowings *na-tot*, the number of function evaluations (including evaluation of derivatives which are counted as normal function evaluations) for each of the extensions *fe-ne, fe-ee, fe-te* and the total number of function evaluations *fe-tot*. We also indicate the number of preconditionings by *pr-con* and whether the algorithm can prove the existence of the solutions in the resulting intervals by *proof*.

6.1 Broyden Banded Functions

This is a traditional benchmark of interval techniques and was used for instance in [7]. It consists in finding the zeros of the functions

$$f_i(x_1, \ldots, x_n) = x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} x_j(1 + x_j) \quad (1 \leq i \leq n)$$

where $J_i = \{j \mid j \neq i \ \& \ max(1, i-5) \leq j \leq min(n, i+1)\}$. One of the interesting features of this benchmark is that it is easy to scale up to an arbitrary dimension and hence provides a good basis to compare various methods. Table 2 reports the results of our algorithm for various sizes assuming initial intervals $[-1, 1]$.

The results indicate that `Newton` solves the problem using only constraint propagation: no branching is needed. In addition, the growth of the computation times is very low and indicates that `Newton` is essentially linear and can thus solve very large instances of this problem. Finally,

	5	10	20	40	80	160	320
CPU time	0.20	1.65	4.25	9.79	22.13	48.30	113.71
growth		8.25	2.57	2.30	2.26	2.18	2.35
branching	0	0	0	0	0	0	0
na-ne	57	260	661	1607	4351	8096	17126
na-ee	1226	8334	21236	48540	102797	206926	414798
na-te	35	110	260	560	1200	2400	4480
na-tot	1318	8704	22157	50707	108348	217422	436404
fe-ne	81	1828	2943	5103	11993	20431	42125
fe-ee	3462	21518	53722	121984	257398	517640	1036210
fe-te	95	320	920	2720	8800	30400	111360
fe-tot	3638	23666	57585	129807	278191	568471	1189695
pr-con	0	0	0	0	0	0	0
proof	yes	yes	yes	yes	yes	yes	yes

Table 2: **Newton** on the Broyden Banded functions with initial intervals $[-1, 1]$

	5	10	20	40	80	160	320
CPU time	0.31	2.15	5.09	12.49	27.69	61.60	143.40
growth		6.93	2.36	2.45	2.21	2.22	2.32
branching	0	0	0	0	0	0	0
pr-con	0	0	0	0	0	0	0
proof	yes	yes	yes	yes	yes	yes	yes

Table 3: **Newton** on the Broyden Banded functions with initial intervals $[-10^8, 10^8]$

Newton proves the existence of a solution in the final intervals. To our knowledge, no other algorithm has all these functionalities. Table 3 shows the same results when the initial intervals are $[-10^8, 10^8]$. They indicate that the CPU time increases only slightly in this problem when the initial intervals become substantially larger. It is interesting to note that substantial pruning is obtained by box-consistency on the natural and distributed extensions alone. For $n = 10$, maximal box-consistency on these two extensions produces the intervals

$$[-0.4283028737061274627, -0.4283028534683728794]$$
$$[-0.4765964317901201786, -0.4765964169224605195]$$
$$[-0.5196524683730758821, -0.5196524589206473754]$$
$$[-0.5580993358758108425, -0.5580993137885511545]$$
$$[-0.5925061654931400579, -0.5925061481657747375]$$
$$[-0.6245036923913307448, -0.6245036720076052594]$$
$$[-0.6232394806883442274, -0.6232394621928379896]$$
$$[-0.6213938520278742273, -0.6213938315652728361]$$
$$[-0.6204536054436834425, -0.6204535878744913413]$$
$$[-0.5864692773020701023, -0.5864692641387999616]$$

which have widths lower than 10^{-6}. Note that the Hansen-Segupta's operator alone does not produce any pruning initially and returns the initial intervals whether they be of the form $[-10^8, +10^8]$ or $[-1, 1]$. This indicates that box-consistency on the natural and distributed interval extensions are particularly effective when far from a solution while box-consistency on the Taylor extension

	5	10	20	40	80
CPU time	0.75	4.07	24.49	192.81	1752.64
growth		5.42	6.02	7.87	9.08
branching	0	0	0	0	0
na-ne	3663	12616	46555	213949	1236532
na-te	104	255	505	1005	2807
na-tot	3767	12871	47060	214954	1239339
fe-ne	8775	31837	107977	466595	2586907
fe-te	884	3111	11211	42411	166415
fe-tot	9659	34948	119188	509006	2753322
fe-grow		3.62	3.41	4.27	5.40
pr-con	1	1	1	1	1
proof	yes	no	no	no	no

Table 4: **Newton** on the Moré-Cosnard nonlinear integral Equation with initial intervals in $[-4, 5]$

	5	10	20	40	80
CPU time	0.70	3.82	20.81	189.94	1735.09
growth		5.45	5.44	9.12	9.13
branching	0	0	0	0	0
pr-con	0	0	0	0	0
proof	yes	no	no	no	no

Table 5: **Newton** on the Moré-Cosnard nonlinear integral Equation with initial intervals in $[-10^8, 0]$

(and the Hansen-Segupta's operator) is effective when near a solution.

It is also interesting to stress the importance of box-consistency on the natural extension in this example to reduce the growth factor. Without it, the algorithm takes about 48 and 440 seconds instead of 27 and 61 for **Newton** for $n = 80$ and $n = 160$, since the distributed interval extension loses precision due to the dependency problem.

Finally, it is interesting to compare **Newton** with traditional interval methods. HRB takes 0.34 seconds on $n = 5$ with 18 branchings, about 18 seconds for $n = 10$ with about 300 branchings, and does not return after more than an hour on $n = 20$.

6.2 Discretization of a Nonlinear Integral Equation

This example comes from [23] and is also a standard benchmark for nonlinear equation solving . It consists in finding the root of the functions $f_k(x_1, \ldots, x_m)$ $(1 \leq k \leq m)$ defined as

$$x_k + \frac{1}{2(m+1)}[(1-t_k)\sum_{j=1}^{k} t_j(x_j+t_j+1)^3 + t_k \sum_{j=k+1}^{m} (1-t_j)(x_j+t_j+1)^3]$$

where $t_j = jh$ and $h = 1/(m+1)$. These functions come from the discretization of a nonlinear integral equation, giving a constraint system denser than the sparse constraint system for the Broyden banded functions. The variables x_i were given initial domains $[-4, 5]$ as in [29] and the computation results are given in Table 4.

Once again, it is interesting to note that **Newton** is completely deterministic on this problem, i.e., it does not do any branching. **Newton** is probably cubic in the number of variables for this

	5	10	20	40	80
CPU time	0.66	7.76	968.25	?	?
branching	5	24	508	?	?
fe-tot	3709	20194	1285764	?	?
pr-con	7	32	667	?	?
proof	yes	no	no	?	?

Table 6: The HRB algorithm on the Moré-Cosnard nonlinear integral Equation with initial intervals in $[-4, 5]$

problem. It is important to point out the critical role of box-consistency on the natural extension to solve this problem efficiently. **Newton** without the natural extension would not be deterministic and would slow down exponentially, since box-consistency on the distributed extension loses too much precision due to the dependency problem (multiple occurrences of the same variable) and box-consistency on the Taylor interval extension is not helpful initially. Once again, we observe that box-consistency over the natural extension is helpful when far from a solution while box-consistency on the Taylor extension is useful to terminate the search quickly. Table 5 gives the result for the initial intervals of size $[-10^8, 0]$, which shows that the algorithm continues to perform well in this case. Finally, Table 6 gives the results for the HRB algorithm on this problem. Once again, **Newton** outperforms the HRB method substantially.

6.3 Interval Arithmetic Benchmarks

This section considers standard benchmarks from interval arithmetic papers [21, 11]. Benchmark `i1` is the following set of equations

$$\begin{cases} 0 = x_1 - 0.25428722 - 0.18324757\ x_4\ x_3\ x_9 \\ 0 = x_2 - 0.37842197 - 0.16275449\ x_1\ x_{10}\ x_6 \\ 0 = x_3 - 0.27162577 - 0.16955071\ x_1\ x_2\ x_{10} \\ 0 = x_4 - 0.19807914 - 0.15585316\ x_7\ x_1\ x_6 \\ 0 = x_5 - 0.44166728 - 0.19950920\ x_7\ x_6\ x_3 \\ 0 = x_6 - 0.14654113 - 0.18922793\ x_8\ x_5\ x_{10} \\ 0 = x_7 - 0.42937161 - 0.21180486\ x_2\ x_5\ x_8 \\ 0 = x_8 - 0.07056438 - 0.17081208\ x_1\ x_7\ x_6 \\ 0 = x_9 - 0.34504906 - 0.19612740\ x_{10}\ x_6\ x_8 \\ 0 = x_{10} - 0.42651102 - 0.21466544\ x_4\ x_8\ x_1 \end{cases}$$

with initial intervals $[-2, 2]$. Benchmark `i2` is the set of equations

$$\begin{cases}
0 = x_1 - 0.24863995 - 0.19594124\, x_7\, x_{10}\, x_{16} \\
0 = x_2 - 0.87528587 - 0.05612619\, x_{18}\, x_8\, x_{11} \\
0 = x_3 - 0.23939835 - 0.20177810\, x_{10}\, x_7\, x_{11} \\
0 = x_4 - 0.47620128 - 0.16497518\, x_{12}\, x_{15}\, x_1 \\
0 = x_5 - 0.24711044 - 0.20198178\, x_8\, x_9\, x_{16} \\
0 = x_6 - 0.33565227 - 0.15724045\, x_{16}\, x_{18}\, x_{11} \\
0 = x_7 - 0.13128974 - 0.12384342\, x_{12}\, x_{13}\, x_{15} \\
0 = x_8 - 0.45937304 - 0.18180253\, x_{19}\, x_{15}\, x_{18} \\
0 = x_9 - 0.46896600 - 0.21241045\, x_{13}\, x_2\, x_{17} \\
0 = x_{10} - 0.57596835 - 0.16522613\, x_{12}\, x_9\, x_{13} \\
0 = x_{11} - 0.56896263 - 0.17221383\, x_{16}\, x_{17}\, x_8 \\
0 = x_{12} - 0.70561396 - 0.23556251\, x_{14}\, x_{11}\, x_4 \\
0 = x_{13} - 0.59642512 - 0.24475135\, x_7\, x_{16}\, x_{20} \\
0 = x_{14} - 0.46588640 - 0.21790395\, x_{13}\, x_3\, x_{10} \\
0 = x_{15} - 0.10607114 - 0.20920602\, x_1\, x_9\, x_{10} \\
0 = x_{16} - 0.26516898 - 0.21037773\, x_4\, x_{19}\, x_9 \\
0 = x_{17} - 0.20436664 - 0.19838792\, x_{20}\, x_{10}\, x_{13} \\
0 = x_{18} - 0.56003141 - 0.18114505\, x_6\, x_{13}\, x_8 \\
0 = x_{19} - 0.92894617 - 0.04417537\, x_7\, x_{13}\, x_{16} \\
0 = x_{20} - 0.57001682 - 0.17949149\, x_1\, x_3\, x_{11}
\end{cases}$$

with initial intervals $[-1, 2]$. Benchmark `i3` has the same set of equations as `i2` but has initial intervals $[-2, 2]$. Benchmark `i4` has the set of equations

$$\begin{cases}
0 = x_1^2 - 0.25428722 - 0.18324757\, x_4^2\, x_3^2\, x_9^2 \\
0 = x_2^2 - 0.37842197 - 0.16275449\, x_1^2\, x_{10}^2\, x_6^2 \\
0 = x_3^2 - 0.27162577 - 0.16955071\, x_1^2\, x_2^2\, x_{10}^2 \\
0 = x_4^2 - 0.19807914 - 0.15585316\, x_7^2\, x_1^2\, x_6^2 \\
0 = x_5^2 - 0.44166728 - 0.19950920\, x_7^2\, x_6^2\, x_3^2 \\
0 = x_6^2 - 0.14654113 - 0.18922793\, x_8^2\, x_5^2\, x_{10}^2 \\
0 = x_7^2 - 0.42937161 - 0.21180486\, x_2^2\, x_5^2\, x_8^2 \\
0 = x_8^2 - 0.07056438 - 0.17081208\, x_1^2\, x_7^2\, x_6^2 \\
0 = x_9^2 - 0.34504906 - 0.19612740\, x_{10}^2\, x_6^2\, x_8^2 \\
0 = x_{10}^2 - 0.42651102 - 0.21466544\, x_4^2\, x_8^2\, x_1^2
\end{cases}$$

and initial intervals $[-1, 1]$. The number of solutions must be a multiple of 1024. Benchmark `i5` has the following set of equations

$$\begin{cases}
0 = x_1 - 0.25428722 - 0.18324757\, x_4^3\, x_3^3\, x_9^3 + x_3^4\, x_9^7 \\
0 = x_2 - 0.37842197 - 0.16275449\, x_1^3\, x_{10}^3\, x_6^3 + x_{10}^4\, x_6^7 \\
0 = x_3 - 0.27162577 - 0.16955071\, x_1^3\, x_2^3\, x_{10}^3 + x_2^4\, x_{10}^7 \\
0 = x_4 - 0.19807914 - 0.15585316\, x_7^3\, x_1^3\, x_6^3 + x_1^4\, x_6^7 \\
0 = x_5 - 0.44166728 - 0.19950920\, x_7^3\, x_6^3\, x_3^3 + x_6^4\, x_3^7 \\
0 = x_6 - 0.14654113 - 0.18922793\, x_8^3\, x_5^3\, x_{10}^3 + x_5^4\, x_{10}^7 \\
0 = x_7 - 0.42937161 - 0.21180486\, x_2^3\, x_5^3\, x_8^3 + x_5^4\, x_8^7 \\
0 = x_8 - 0.07056438 - 0.17081208\, x_1^3\, x_7^3\, x_6^3 + x_7^4\, x_6^7 \\
0 = x_9 - 0.34504906 - 0.19612740\, x_{10}^3\, x_6^3\, x_8^3 + x_6^4\, x_8^7 \\
0 = x_{10} - 0.42651102 - 0.21466544\, x_4^3\, x_8^3\, x_1^3 + x_8^4\, x_1^7
\end{cases}$$

	i1	i2	i3	i4	i5
CPU time	0.06	0.30	0.31	73.94	0.08
branching	0	0	0	1023	0
na-ee	625	2107	1949	290776	381
na-te	0	80	80	37930	21
na-tot	625	2187	2029	328706	401
fe-ee	1760	5698	5318	752220	992
fe-te	0	560	560	269760	140
fe-tot	1760	6258	5878	1021980	1132
pr-con	0	1	1	1939	1
proof	yes	yes	yes	yes	yes

Table 7: `Newton` on the Traditional Interval Arithmetic Benchmarks

	i1	i2	i3	i4	i5
CPU time	14.28	1821.23	5640.80	445.28	33.58
branching	498	9031	36933	11263	1173
fe-tot	77380	6441640	19979025	2554066	154948
pr-con	586	12817	42193	14335	1211
proof	yes	yes	yes	yes	yes

Table 8: `HRB` on the Traditional Interval Arithmetic Benchmarks

and initial intervals $[-1, 1]$.

`Newton` solves all the problems with one solution without branching and solves the problem having 1024 solutions with 1023 branchings. Note also that box-consistency on the distributed extension solves benchmark `i1` alone. The results once again confirm our observation on when the various extensions are useful. Closely related results were observed in [11] on these benchmarks (see the related work section for a more detailed comparison) but our algorithm is in general about 4 times faster (assuming similar machines) and does not do any branching on `i5`. Table 8 also describes the results for the traditional interval arithmetic method. The importance of box-consistency on the distributed extension can easily be seen from these results. Note also that `Newton` (and interval methods) can prove the existence of a solution in the final intervals for all these problems.

It is also interesting to note that problem `i4` can be solved dramatically more efficiently simply by introducing intermediary variables $y_i = x_i^2$. The excution times then dropped to less than 0.5 seconds.

6.4 Kinematics Applications

We now describe the performance of `Newton` on two kinematics examples. Application `kin1` comes from robotics and describes the inverse kinematics of an elbow manipulator [11]. It consists of a

- 0.249150680	+ 0.125016350	- 0.635550070	+ 1.48947730
+ 1.609135400	- 0.686607360	- 0.115719920	+ 0.23062341
+ 0.279423430	- 0.119228120	- 0.666404480	+ 1.32810730
+ 1.434801600	- 0.719940470	+ 0.110362110	- 0.25864503
+ 0.000000000	- 0.432419270	+ 0.290702030	+ 1.16517200
+ 0.400263840	+ 0.000000000	+ 1.258776700	- 0.26908494
- 0.800527680	+ 0.000000000	- 0.629388360	+ 0.53816987
+ 0.000000000	- 0.864838550	+ 0.581404060	+ 0.58258598
+ 0.074052388	- 0.037157270	+ 0.195946620	- 0.20816985
- 0.083050031	+ 0.035436896	- 1.228034200	+ 2.68683200
- 0.386159610	+ 0.085383482	+ 0.000000000	- 0.69910317
- 0.755266030	+ 0.000000000	- 0.079034221	+ 0.35744413
+ 0.504201680	- 0.039251967	+ 0.026387877	+ 1.24991170
- 1.091628700	+ 0.000000000	- 0.057131430	+ 1.46773600
+ 0.000000000	- 0.432419270	- 1.162808100	+ 1.16517200
+ 0.049207290	+ 0.000000000	+ 1.258776700	+ 1.07633970
+ 0.049207290	+ 0.013873010	+ 2.162575000	- 0.69686809

Table 9: Coefficients for the Inverse Kinematics Example.

sparse system with 12 variables and the set of equations is as follows:

$$\begin{cases} s_2c_5s_6 - s_3c_5s_6 - s_4c_5s_6 + c_2c_6 + c_3c_6 + c_4c_6 = 0.4077 \\ c_1c_2s_5 + c_1c_3s_5 + c_1c_4s_5 + s_1c_5 = 1.9115 \\ s_2s_5 + s_3s_5 + s_4s_5 = 1.9791 \\ c_1c_2 + c_1c_3 + c_1c_4 + c_1c_2 + c_1c_3 + c_1c_2 = 4.0616 \\ s_1c_2 + s_1c_3 + s_1c_4 + s_1c_2 + s_1c_3 + s_1c_2 = 1.7172 \\ s_2 + s_3 + s_4 + s_2 + s_3 + s_2 = 3.9701 \\ s_i^2 + c_i^2 = 1 \quad (1 \le i \le 6). \end{cases}$$

The second benchmark, denoted by **kin2**, is from [24] and describes the inverse position problem for a six-revolute-joint problem in mechanics. The equations which describe a denser constraint system are as follows:

$$\begin{cases} x_i^2 + x_{i+1}^2 - 1 = 0 \quad (1 \le i \le 4) \\ a_{1i}x_1x_3 + a_{2i}x_1x_4 + a_{3i}x_2x_3 + a_{4i}x_2x_4 + a_{5i}x_5x_7 + a_{6i}x_5x_8 + a_{7i}x_6x_7 + a_{8i}x_6x_8 \\ a_{9i}x_1 + a_{10i}x_2 + a_{11i}x_3 + a_{12i}x_4 + a_{13i}x_5a_{14i}x_6 + a_{15i}x_7 + a_{16i}x_8 + a_{17i} = 0 \ \ (1 \le i \le 4) \end{cases}$$

where the coefficients a_{ki} are given in table 9. In both examples, the initial intervals were given as $[-10^8, 10^8]$.

The results of **Newton** on these two benchmarks are given in Table 10. **Newton** is fast on the first benchmark and does not branch much to obtain all solutions. The algorithm in [11] branches more (the reported figure is 257 branches but it is not really comparable due to the nature of the algorithm) and is about 16 times slower on comparable machines. We are not aware of the results of continuation methods on this problem. **Newton** is slower on the second application and takes about 6 minutes. The continuation method described in [35] requires about 30 seconds on a DEC 5000/200. This method exploits the fact that the Newton polytopes for the last 4 equations are the same. Note that HRB requires about 1630 and 4730 seconds on these examples. Note also that

	kin1	kin2
CPU time	14.240	353.06
branching	89	5693
na-ee	17090	784687
na-te	10176	123032
na-tot	27266	907719
fe-ee	45656	1714779
fe-te	62080	854384
fe-tot	107736	2569163
pr-con	163	9505
proof	yes	yes

Table 10: **Newton** on the Kinematics Benchmarks

	4	5	6	7	8	9
CPU time	0.21	1.22	8.20	46.59	352.80	3311.42
growth		5.80	6.72	5.68	7.57	9.38
branching	24	119	517	2231	12248	82579
na-ee	834	4701	42481	214430	1622417	14031838
na-te	480	1976	6325	29238	157402	1219960
na-tot	1314	6677	48806	243668	1779819	15251798
fe-ee	2220	116628	99994	489894	3626847	30782836
fe-te	1293	6304	29825	160284	1062789	9118448
fe-tot	3513	17932	129819	650178	4689636	39901284
pr-con	37	147	687	2828	15265	104352
proof	yes	yes	yes	yes	yes	yes

Table 11: **Newton** on the economics modelling problem with initial intervals in $[-100, 100]$

Newton can prove the existence of solutions in the final intervals for these problems and that our computer representation uses intermediate variables

$$\begin{cases} x_{13} = x_4 + x_6 + x_8 \\ x_{14} = x_3 + x_5 + x_7 \\ x_{15} = x_4 + x_6 + x_8 \\ x_{16} = 3 * x_4 + 2 * x_6 + x_8 \end{cases}$$

to improve efficiently slightly in the first problem.

6.5 An Economics Modelling Application

The following example is taken from [25]. It is a difficult economic modelling problem that can be scaled up to arbitrary dimensions. For a given dimension n, the problem can be stated as the system

$$\begin{cases} (x_k + \sum_{i=1}^{n-k-1} x_i x_{i+k}) x_n - c_k = 0 \quad (1 \leq k \leq n-1) \\ \sum_{l=1}^{n-1} x_l + 1 = 0 \end{cases}$$

and the constants can be chosen at random.

Table 11 reports the results for various values of n with an initial interval of $[-100, 100]$. It is interesting to compare those results with the continuation methods presented in [35]. [35] reports

	4	5	6	7	8	9
CPU time	0.60	3.35	22.53	127.65	915.24	8600.28
growth		5.58	6.72	5.66	7.16	9.39
branching	102	500	1778	7527	38638	244263
na-ee	2689	15227	122662	606805	4413150	36325819
na-te	978	3296	14055	64632	366436	2647016
na-tot	3667	18523	136717	671437	4779586	38972835
fe-ee	7140	38160	291206	1400216	9913850	80264897
fe-te	3288	15024	80110	429396	2825368	22672208
fe-tot	10428	53184	371716	1829612	12739218	102937105
pr-con	148	527	2080	8337	42704	271534
proof	yes	yes	yes	yes	yes	yes

Table 12: **Newton** on the economics modelling problem with initial intervals in $[-10^8, 10^8]$

times (on a DEC-5000/200) of about 1 second for $n = 4$, 6 seconds for $n = 5$, 50 seconds for $n = 6$ and 990 seconds for $n = 7$. **Newton** is substantially faster on this problem than this continuation method, since it takes about 47 seconds for $n = 7$. More importantly, the growth factor seems much lower in **Newton**. The continuation method has growths of about 8 and 20 when going from 5 to 6 and 6 to 7, while **Newton** has growths of about 6.72 and 5.68. Table 12 gives the same results for initial intervals in $[-10^8, 10^8]$. It is interesting to note that the computation times increase by less than a factor 3 and that the growth factor is independent of the initial intervals. Note also that Newton can establish the existence of solutions for these problems. Finally, it is worthwhile stating that the results were obtained for a computer representation where x_n has been eliminated in a problem of dimension n.

6.6 Combustion Application

This problem is also from Morgan's book [25] and represents a combustion problem for a temparature of 3000°. The problem is described by the following sparse systems of equations

$$\begin{cases} x_2 + 2\ x_6 + x_9 + 2\ x_{10} = 10^{-5} \\ x_3 + x_8 = 3\ 10^{-5} \\ x_1 + x_3 + 2\ x_5 + 2\ x_8 + x_9 + x_{10} = 5\ 10^{-5} \\ x_4 + 2\ x_7 = 10^{-5} \\ 0.5140437\ 10^{-7}\ x_5 = x_1^2 \\ 0.1006932\ 10^{-6}\ x_6 = 2\ x_2^2 \\ 0.7816278\ 10^{-15}\ x_7 = x_4^2 \\ 0.1496236\ 10^{-6}\ x_8 = x_1\ x_3 \\ 0.6194411\ 10^{-7}\ x_9 = x_1\ x_2 \\ 0.2089296\ 10^{-14}\ x_{10} = x_1\ x_2^2 \end{cases}$$

which is typical of chemical equilibrium systems. Table 13 describes the results of **Newton** on for the initial intervals $[-1, 1]$ and $[-10^8, 10^8]$. **Newton** behaves well on this example, since the continuation method of [35] takes about 57 seconds. Note once again that a substantial increase in the size of the initial intervals only induces a slowdown of about 2.5 for **Newton**. Note also that **Newton** can prove the existence of the solutions and that we use a formulation where variable x_7 and x_3 have been eliminated.

	[-1,1]	$[-10^8,10^8]$
CPU time	4.06	9.94
branching	183	523
na-ee	10676	28473
na-te	5248	7952
na-tot	15924	36425
fe-ee	28928	77508
fe-te	22464	49632
fe-tot	51392	127140
pr-con	187	527
proof	yes	yes

Table 13: **Newton** on the Combustion Problem.

CPU time	*branching*	*na-ee*	*na-te*	*na-tot*	*fe-ee*	*fe-te*	*fe-tot*	*pr-con*	*proof*
6.32	256	13725	3400	17125	34811	17425	52236	425	yes

Table 14: **Newton** on the Chemestry Problem with Initial Intervals $[0, 10^8]$

6.7 Chemical Equilibrium Application

This problem originates from [18] and describes a chemical equilibrium system. The set of equations is as follows:

$$
\begin{cases}
R = 10 \\
R_5 = 0.193 \\
R_6 = 0.002597/\sqrt{40} \\
R_7 = 0.003448/\sqrt{40} \\
R_8 = 0.00001799/40 \\
R_9 = 0.0002155/\sqrt{40} \\
R_{10} = 0.00003846/40 \\
x_1\ x_2 + x_1 - 3\ x_5 = 0 \\
2\ x_1\ x_2 + x_1 + x_2\ x_3^2 + R_8\ x_2 - R\ x_5 + 2\ R_{10}\ x_2^2 + R_7\ x_2\ x_3 + R_9\ x_2\ x_4 = 0 \\
2\ x_2\ x_3^2 + 2\ R_5\ x_3^2 - 8\ x_5 + R_6\ x_3 + R_7\ x_2\ x_3 = 0 \\
R_9\ x_2\ x_4 + 2\ x_4^2 - 4\ R\ x_5 = 0 \\
x_1\ x_2 + x_1 + R_{10}\ x_2^2 + x_2\ x_3^2 + R_8\ x_2 + R_5\ x_3^2 + x_4^2 - 1 + R_6\ x_3 + R_7\ x_2\ x_3 + R_9\ x_2\ x_4 = 0
\end{cases}
$$

and all x_i's must be positive. The results are depicted in Table 14 for an initial interval $[0, 10^8]$. They indicate that **Newton** is particularly effective on this problem, since it takes about 6 seconds and proves the existence of a solution in the final intervals. Note that the continuation method of [35] takes about 56 seconds on this problem.

	$[-10,10]$	$[-10^2,10^2]$	$[-10^3,10^3]$	$[-10^4,10^4]$
CPU time	0.91	11.69	172.71	2007.51
growth		12.84	14.77	11.62
branching	52	663	9632	115377
na-ee	4290	57224	645951	6541038
na-te	810	6708	96888	1173456
na-tot	5100	63932	742839	7714494
fe-ee	11012	144843	1620538	16647442
fe-te	4104	48804	769056	9376632
fe-tot	15116	193647	2389594	26024074
pr-con	69	983	15980	195270
proof	yes	yes	yes	yes

Table 15: `Newton` on the Neurophysiology Problem.

6.8 A Neurophysiology Application

We conclude the experimental section by showing an example illustrating the limitations of `Newton`. The application is from neurophysiology [35] and consists of the following system of equations:

$$\begin{cases} x_1^2 + x_3^2 = 1 \\ x_2^2 + x_4^2 = 1 \\ x_5\ x_3^3 + x_6\ x_4^3 = c_1 \\ x_5\ x_1^3 + x_6\ x_2^3 = c_2 \\ x_5\ x_1\ x_3^2 + x_6\ x_4^2\ x_2 = c_3 \\ x_5\ x_1^2\ x_3 + x_6\ x_2^2\ x_4 = c_4 \end{cases}$$

No initial intervals for the variables were given and the constants c_i can be chosen at random. The continuation method of [35] solves this problem in about 6 seconds. The results of `Newton` are depicted in Table 15 for various initial intervals. `Newton` is fast when the initial intervals are small (i.e., $[-10,10]$). Unfortunately, the running time of the algorithm increases linearly with the size of the initial intervals, showing a limitation of the method on this example.

7 Related Work and Discussion

The research described in this paper originated in an attempt to improve the efficiency of constraint logic programming languages based on intervals such as `BNR-Prolog` [27] and `CLP(BNR)` [3]. These Constraint Logic Programming languages use constraint solving as basic operation and they were based on the simple generalization of arc-consistency described previously, i.e.,

$$I_i = box\{I_i \cap \{\ r_i\ \mid\ \exists r_1 \in I_1, \ldots, \exists r_{i-1} \in I_{i-1}, \ldots, \exists r_{i+1} \in I_{i+1}, \ldots, \exists r_n \in I_n : c(r_1, \ldots, r_n)\ \}\}.$$

This approximation was enforced on simple constraints such as

$$x_1 = x_2 + x_3,\ \ x_1 = x_2 - x_3,\ \ x_1 = x_2 \times x_3$$

and complex constraints were decomposed in terms of these simple constraints.

As mentioned earlier, this approach is not very effective [2] and our main goal was to design new approximations of arc-consistency that could make use of existing interval methods. The main problem was the difficulty in characterizing the pruning of the Newton operator N^* in a declarative way (in order to introduce it nicely in the above programming languages) and box-consistency emerged as an attempt to generalize the operator to make sure that the bounds of the interval were locally consistent. Subsequent research made us realize that box-consistency is independent of the Newton operator and can be enforced even if the functions are not continuous or differentiable. In addition, the value of applying box-consistency on several extensions became clear. On the one hand, box-consistency on the Taylor extension generalizes interval methods based on Gauss-Seidel iterations and enables us to capture nicely Hansen-Segupta's operator. On the other hand, box-consistency on the natural and distributed extensions is really orthogonal to the pruning obtained from the Taylor expansion, producing a particularly effective algorithm. It is also worth pointing out that `Newton` spends most time in the natural and distributed extensions. However, for many applications, the use of the Taylor interval extension is critical to terminate the search quickly and to avoid generating many small intervals around the solutions. As a general observation, box-consistency on the natural and distributed extensions seem effective when far from a solution while box-consistency on the Taylor expansion seems effective when near a solution. It is worth mentioning that that the interval community has spent much effort to design additional techniques to speed up further the computation when near a solution but have not considered techniques to improve pruning when far from a solution.

It is interesting to note that the idea of using approximations of arc-consistency was also used independently by Hong and Stahl [11], who were also exposed to research on Constraint Logic Programming. Their use of projections is however quite different from ours. The key idea is to work with a set of boxes and to use projections to split a box into several subboxes by isolating all zeros of a projection. This gives an algorithm of a very different nature which cannot easily be characterized as a branch & prune algorithm since constraints are used to branch. Our approach seems to be more effective in practice, since their use of projections may generate many subboxes that may all need to be pruned away later on, implying much redundant work. Our approach postpones the branching until no pruning takes place and generates only subboxes when they are strictly necessary to progress. It is also very interesting to report that, on all benchmarks that we tested, the projection never contained more than two zeros. This seems to indicate that searching for all zeros may not be worthwhile in most cases and that box-consistency may be the right trade-off here. Finally, note that their approach seems to use implicitly an distributed extension[10] but they do not make use of the natural extension which is very important for some applications.

Note also that our current implementation does not use some of the novel techniques of the interval community such as the more advanced conditioners and splitting techniques of [13]. It is of course possible to include them easily, since the overall recursive structure of the implementations is essentially similar. Integrating these results would obviously be of benefit, since these techniques are complementary to ours.

The research described here also provides a uniform framework to integrate these techniques in Constraint Logic Programming, to understand the importance of the various pruning operators and their relationships and to suggest further research directions. For instance, higher notions of consistency such as path-consistency [19] may be worth investigating for some applications.

[10] The idea of sandwitching the interval function in between two real functions is described there.

8 Conclusion

In this paper, we presented a branch & prune algorithm to find all isolated solutions to a system of polynomial constraints over the reals. If the solution are not isolated the algorithm will return boxes that contain several solutions. The algorithm is based on a single concept, box-consistency, which is an approximation of arc-consistency, a notion well-known in artificial intelligence. Box-consistency can be instantiated to produce Hansen-Segupta operator as well as other narrowing operators which are more effective when the computation is far from a solution. The algorithm and its mathematical foundations are simple. Moreover, the algorithm is shown to behave well on a variety of benchmarks from kinematics, mechanics, chemistry, combustion, and economics. It outperforms the interval methods we know of and compares well with continuation methods on their benchmarks. In addition, problems such as the Broyden banded function and the Moré-Cosnard discretization of a nonlinear integral equation can be solved for several hundred variables. Limitations of the method (e.g., a sensitivity to the size of the initial intervals on some problems) have also been identified.

Acknowledgments

We would like to thank the referees for their help in making the paper accessible to a wider audience, for pointing out several related works, and for their careful comments on the manuscript. We would like to thank F. Benhamou, A. Colmerauer, and B. Le Charlier for many interesting discussions on this topic. Pascal Van Hentenryck was supported in part by the Office of Naval Research under grant ONR Grant N00014-94-1-1153, the National Science Foundation ▪▪▪▪ grant numbers CCR-9357704, a NSF National Young Investigator Award with matching funds of Hewlett-Packard. Deepak Kapur was supported in part by a grant from United State Air Force Office of Scientific Research AFOSR-91-0361.

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, NY, 1983.

[2] F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) Revisited. In *Proceedings of the International Symposium on Logic Programming (ILPS-94)*, pages 124–138, Ithaca, NY, November 1994.

[3] F. Benhamou and W. Older. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming*, 1995. To appear.

[4] R. Hammer, M. Hocks, M. Kulisch, and D. Ratz. *Numerical Toolbox for Verified Computing I – Basic Numerical Problems, Theory, Algorithms, and PASCAL-XSC Programs.* Springer-Verlag, Heidelberg, 1993.

[5] E. Hansen. *Global Optimization Using Interval Analysis.* Marcel Dekker, New York, 1992.

[6] E.R. Hansen. Global Optimization Using Interval Analysis: the Multi-Dimensional Case. *Numer. Math*, 34:247–270, 1980.

[7] E.R. Hansen and R.I. Greenberg. An Interval Newton Method. *Appl. Math. Comput.*, 12:89–98, 1983.

[8] E.R. Hansen and S. Sengupta. Bounding Solutions of Systems of Equations Using Interval Analysis. *BIT*, 21:203–211, 1981.

[9] E.R. Hansen and R.R. Smith. Interval Arithmetic in Matrix Computation: Part II. *SIAM Journal on Numerical Analysis*, 4:1–9, 1967.

[10] R.J. Hanson. Interval Arithmetic as a Closed Arithmetic System on a Computer. Jet Propulsion Laboratory Report 197, 1968.

[11] H. Hong and V. Stahl. Safe Starting Regions by Fixed Points and Tightening. *Computing*, 53(3-4):323–335, 1994.

[12] W.M. Kahan. A More Complete Interval Arithmetic. Lecture Notes for a Summer Course at the University of Michigan, 1968.

[13] R.B. Kearfott. Preconditioners for the Interval Gauss-Seidel Method. *SIAM Journal of Numerical Analysis*, 27, 1990.

[14] R.B. Kearfott. A Review of Preconditioners for the Interval Gauss-Seidel Method. *Interval Computations 1*, 1:59–85, 1991.

[15] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.

[16] R. Krawczyk. A Class of interval Newton Operators . *Computing*, 37:179–183, 1986.

[17] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[18] K. Meintjes and A.P. Morgan. Chemical Equilibrium Systems as Numerical test Problems. *ACM Transactions on Mathematical Software*, 16:143–151, 1990.

[19] U. Montanari. Networks of Constraints : Fundamental Properties and Applications to Picture Processing. *Information Science*, 7(2):95–132, 1974.

[20] R.E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[21] R.E. Moore. *Methods and Applications of Interval Analysis.* SIAM Publ., 1979.

[22] R.E. Moore and S.T. Jones. Safe Starting Regions for Iterative Methods. *SIAM Journal on Numerical Analysis*, 14:1051–1065, 1977.

[23] J.J. More and M.Y. Cosnard. Numerical Solution of Nonlinear Equations. *ACM Transactions on Mathematical Software*, 5:64–85, 1979.

[24] A.P. Morgan. Computing All Solutions To Polynomial Systems Using Homotopy Continuation. *Appl. Math. Comput.*, 24:115–138, 1987.

[25] A.P. Morgan. *Solving Polynomial Systems Using Continuation for Scientific and Engineering Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[26] A. Neumaier. *Interval Methods for Systems of Equations.* PHI Series in Computer Science. Cambridge University Press, Cambridge, 1990.

[27] W. Older and A. Vellino. Extending Prolog with Constraint Arithmetics on Real Intervals. In *Canadian Conference on Computer & Electrical Engineering*, Ottawa, 1990.

[28] L.B. Rall. *Automatic Differentiation: Techniques and Applications.* Springer Lectures Notes in Computer Science, Springer Verlag, New York, 1981.

[29] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization.* Ellis Horwood Limited, Chichester, 1988.

[30] S.M. Rump. Verification Methods for Dense and Sparse Systems of Equations. In J. (Ed.) Herzberger, editor, *Topics in Validated Computations*, pages 217–231. Elsevier, 1988.

[31] J. Siskind and D. McAllester. Nondeterministic Lisp as a Substrate for Constraint Logic Programming. In *AAAI-93*, pages 133–138, 1993.

[32] P. Van Hentenryck. A Logic Language for Combinatorial Optimization. *Annals of Operations Research*, 21:247–274, 1989.

[33] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming.* Logic Programming Series, The MIT Press, Cambridge, MA, 1989.

[34] P. Van Hentenryck, V. Saraswat, and Y. Deville. The Design, Implementation, and Evaluation of the Constraint Language `cc(FD)`. In *Constraint Programming: Basics and Trends.* Springer Verlag, 1995.

[35] J Verschelde, P. Verlinden, and R. Cools. Homotopies Exploiting Newton Polytopes For Solving Sparse Polynomial Systems. *SIAM Journal on Numerical Analysis*, 31(3):915–930, 1994.

3D Object Recognition using Invariance

by

Andrew Zisserman, David Forsyth, Joe Mundy,
Charlie Rothwell, Jane Liu and Nic Pillow

Robotics Research Group

Report No. OUEL 2027/94

November 1994

University of Oxford
Department of Engineering Science
19 Parks Road
Oxford OX1 3PJ
U.K.

Tel: +44 865 273926
Fax: +44 865 273908
Email: az@robots.oxford.ac.uk

Contents

1 Introduction 1
- 1.1 Related approaches to object recognition . . . 2
- 1.2 Geometric invariants in modelling and recognition . . . 3
  - 1.2.1 Indexing . . . 4
  - 1.2.2 Invariance and representation . . . 4
  - 1.2.3 Model acquisition from images . . . 5

2 The planar recognition system 5
- 2.1 Projective invariants used . . . 6
- 2.2 Architecture . . . 6
  - 2.2.1 Feature extraction and invariant formation . . . 7
  - 2.2.2 Indexing to generate recognition hypotheses . . . 8
  - 2.2.3 Hypothesis merging . . . 8
  - 2.2.4 Verification . . . 8
- 2.3 Model acquisition and library formation . . . 9
- 2.4 Recognition examples . . . 9
- 2.5 Summary of performance . . . 10
- 2.6 Appraisal . . . 12

3 Extending invariant descriptions to 3D structures 12
- 3.1 Object classes . . . 12
- 3.2 Definitions — 3D projective invariants . . . 13
- 3.3 Constrained Point Sets . . . 14
- 3.4 Repeated Structure . . . 15
  - 3.4.1 Bilateral Symmetry . . . 16
  - 3.4.2 Translational Repetition . . . 19
  - 3.4.3 Other Repeated Structures . . . 20
- 3.5 Rotational Symmetry . . . 21
- 3.6 Canal Surfaces . . . 23
- 3.7 Polyhedra . . . 26
- 3.8 Extruded Surfaces . . . 27
- 3.9 Algebraic Surfaces . . . 28

4 An architecture for a 3D recognition system 28
- 4.1 Fundamental Principles . . . 28
  - 4.1.1 Class . . . 29
  - 4.1.2 Consistency . . . 29
- 4.2 The Architecture . . . 30
- 4.3 Model acquisition . . . 33
- 4.4 MORSE . . . 33

5 Discussion 35
- 5.1 Critique of the invariance approach . . . 35
- 5.2 Avenues of future research . . . 36

Abstract

The systems and concepts described in this paper document the evolution of the geometric invariance approach to object recognition over the last five years. Invariance overcomes one of the fundamental difficulties in recognising objects from images: that the appearance of an object depends on viewpoint. This problem is entirely avoided if the geometric description is unaffected by the imaging transformation. Such invariant descriptions can be measured from images without any prior knowledge of the position, orientation and calibration of the camera. These invariant measurements can be used to index a library of object models for recognition and provide a principled basis for the other stages of the recognition process such as feature grouping and hypothesis verification. Object models can be acquired directly from images, allowing efficient construction of model libraries without manual intervention.

A significant part of the paper is a summary of recent results on the construction of invariants for 3D objects from a single perspective view. A proposed recognition architecture is described which enables the integration of multiple general object classes and provides a means for enforcing global scene consistency.

Various criticisms of the invariant approach are articulated and addressed.

1 Introduction

The computer recognition of objects has attracted considerable research effort over the last 25 years. It is now widely accepted that object recognition, in the setting of real world scenes and based on a single perspective view, is a difficult problem and cannot be achieved without the use of object models to guide the processing of image data and to confirm object hypotheses. It is also accepted that the most reliable information which is available in a scene is derived from a geometric description of the object based on its projection in the form of 2D geometric image features, as opposed to, for example, its intensity shading. Thus, object recognition systems draw on a library of geometric models, which usually contain information about the shape and appearance of a set of known objects, to determine which, if any, of those objects appear in a given image or image sequence. Recognition is considered successful if the geometric configuration in an image can be explained as a perspective projection of a geometric model of the object.

At present, 3D recognition systems generally have small modelbases containing relatively simple objects. Progress is needed on three fronts:

- **Larger modelbases**: Systems should be able to deal with modelbases containing hundreds to thousands of models. The methods of pose consistency (reviewed in section 1.1), which are commonly used for modelbases with only a few objects, are infeasible for large modelbases because of the computational expense. Coping with such sizes clearly requires some partitioning of the modelbase.

- **More general shape models**: Typically polyhedra are used, which are a poor model for curved objects. A direct representation for non-trivial curved objects is required.

- **Automatic segmentation and grouping**: This is the process, also called figure-ground separation, of extracting image feature groups which correspond to individual object outlines without including the background and other occluding objects. The lack of such grouping is a significant barrier to successful recognition in current systems. In addition to representing the shape of 3D objects, models will have to provide *mechanisms* for their feature segmentation and grouping.

This paper establishes a framework for the next generation of 3D model based vision recognition systems which will have large modelbases, with objects partitioned into a number of different 3D object *classes*. Recognition is from single perspective images of scenes, where the camera is uncalibrated, the objects could be partially occluded, and the scene might contain objects not in the model library. The object classes are defined geometrically in terms of symmetry or other 3D geometric constraints. The *constraint* enables *invariants* of a 3D object in the class to be extracted from a single image of the object outline; and also generates invariant relations on the image outline that enable *grouping*.

Although the paper concentrates on perspective images, the methods are, of course, applicable in weak-perspective (or "affine") imaging situations. Weak-perspective, a linear approximation to perspective, is appropriate as a camera model when object relief is small compared to distance from the camera. A consequence is that parallel world lines are imaged as parallel lines. Invariants computed for perspective imaging are also valid for weak-perspective.

A major constraint underlying the work presented here is that recognition is based on one uncalibrated view of a scene. Our motivation is that this restriction applies in many of the current and future applications for object recognition, such as aerial surveillance, image database query processing, and image-hypertext editing. Even if more images are available, for example in the case of video processing, camera calibration will not generally be known initially. Any grouping, recognition hypothesis, or object recovered up to some ambiguity from a single image, can be propagated to advantage to subsequent views.

A central question explored in this paper is the nature of the shape representation necessary for recognition. Euclidean (metric) representations are routinely used in many existing recognition systems. However, under the most general imaging conditions, structure is recovered up to a projective transformation (i.e., a more general transformation than Euclidean). We demonstrate that projective representations are adequate for recognition. A stratification of representations is provided by the hierarchy of transformation groups: projective, affine, similarity (scaled Euclidean), and Euclidean. This representation hierarchy is progressively more restrictive; for example, two objects that are projectively equivalent need not be affine or similarity equivalent. We will be primarily concerned with the projective stratum, since this covers the "worst case" ambiguity. The other strata will be used to advantage at particular stages of the recognition process.

A related area is the use of *quasi-invariants* [4]. A quasi-invariant is an object property or relation that is not invariant to projective transformations, but is stable over a useful range of views. Invariants of other transformation groups in the hierarchy given above are sometimes quasi-invariants [5]. Quasi-invariants can be very effective in grouping and partial indexing even though they vary under perspective projection. Examples of quasi-invariants are given in the paper.

Our geometric notion of class differs from the more usual functional one. For example, in our definitions, a vase is considered as a surface of revolution as opposed to a container for flowers and water. A geometric class is not specific to a particular object but instead describes a family of objects which are unified by their common 3D constraint relations. A number of examples of these 3D object classes are given in section 3.

We have defined a recognition architecture which integrates these ideas. Class influences each level of the architecture, from image grouping through to organisation of the model base and 3D scene constraints. Recognition is class based, proceeding first by a classification based on image curves, and subsequently the identification of a particular model within the class using values of geometric attributes. This contrasts with many existing recognition systems where a particular object is directly identified. The architecture, combined with the success of existing implementations, demonstrates that a large-scale system implementation based on an invariant framework is now warranted. This effort will culminate in an object recognition system that can recognise a broad class of 3D structures with thousands of individual object instances in the model library.

1.1 Related approaches to object recognition

Recognition is the establishment of a correspondence between image and model features. Most recent approaches to recognition have been implemented in three stages (similar to those defined in [28]): grouping, indexing, and verification.

The aim of grouping (also called *perceptual organisation* [37], *selection*, or *figure-ground discrimination*) is to provide an association of features that are likely to have come from a single object in a scene. Features are typically grouped together using cues such as proximity, parallelism [3, 37] collinearity, and approximate continuity in curvature [12, 62]. The indexing stage hypothesises an association between the grouped image features, and features on a model in the library. The final stage, verification, determines the consistency of this hypothesis with the image data. The image-model match is used to project the model onto the image, and to test the validity of the model hypothesis and model-to-image feature

correspondences determined by measuring image support.

There are three distinct categories of algorithm that have been used to compute correspondence:

1. **Interpretation trees** frame the model-to-image correspondence task as a search tree to allow all possible model and image feature associations, and then control and prune this search process. Although inefficient, this has proved reliable for planar object recognition for a small modelbase when single images are used [27], and has been extended by Ettinger to include useful notions about how hierarchical object descriptions can be realised [14]. However, interpretation trees are not generally able to work with single images of three dimensional objects (though effective when 3D data is provided as direct input to the system [2, 27, 48, 49, 52]). Interpretation trees are not restricted to rigid objects; the *sup-inf* framework for geometric reasoning used in ACRONYM [8] allows the interpretation tree to account for tolerance interval constraints on parameterised objects. Brooks' work has been extended by both Fisher [19] and Reid [54] for different types of sensor and constraint framework. Other ways to treat parameterisations have been suggested by Grimson [28].

2. **Hypothesise and test**, also called *alignment*, first aligns a model-to-image features [31] to yield an initial estimate of pose. This hypothesised alignment is tested by searching for other model-to-image correspondence predicted by the model pose (verification). This algorithm has been implemented for a variety of data formats and feature types [1, 6, 15, 24, 38, 69]. In fact, extensions to 3D curved surfaces have even been created [13, 33].

3. **Pose clustering** is implemented by computing the object pose from a group of features corresponding to a particular model, and storing the estimate in an accumulator in pose space; if enough local groups have the same pose, a hypothesis for the model is formed. This approach (frequently called *generalised Hough*) has the disadvantage that the pose space is high dimensional (six *dof* for 3D Euclidean space), so searching for consistent pose is expensive. Two ways round this are to use a decomposition of the pose space into separable parameters [44, 68], or to use an adaptive Hough transform [66]. Another approach eliminates the requirement to quantise the pose space into rectangular cells by constructing a quantisation that depends both on the estimates of pose, and on the expected error bounds of the pose measurements [10].

For a small number of models, for example two or three, it is reasonable simply to try to find image feature support for each model. This approach is typical of many existing systems [1, 2, 28, 31, 38, 48, 52]. As the size of the model library increases, this approach becomes computationally too expensive. It is then more effective to choose potential models from the library based on the observed image features. That is, image feature measurements are used to *index* into the model base. In constructing such *index functions*, invariance plays a major role, since a model should be identified irrespective of object pose.

1.2 Geometric invariants in modelling and recognition

Invariants are properties of geometric configurations which remain unchanged under an appropriate class of transformations. Within the context of vision we are interested in determining the invariants of an object under perspective projection onto an image. For example, for a planar object the perspective projection between object and image planes is a *projective* transformation. Properties such as intersection, collinearity, and tangency are unaffected by a projective transformation; however, invariant *values* can also be computed. Examples are given in section 2.1.

More formally, under a linear transformation of coordinates, $\mathbf{X}' = \mathtt{T}\mathbf{X}$, the invariant, $I(\mathbf{P})$, of a configuration $\mathbf{P}$ transforms as

$$I(\mathbf{P}') = |\mathtt{T}|^w I(\mathbf{P})$$

and is called a *relative* invariant of weight w, where $\mathbf{P}'$ is the transformed configuration. If $w = 0$, the invariant is unchanged under transformations and is called a *scalar* invariant. We will only be interested in scalar invariants in this paper.

In general we seek invariance to *projective* transformations, so $\mathtt{T}$ is a general non-singular square matrix acting on homogeneous coordinates. For planar configurations it is 3×3, and for 3D configurations 4×4. Note that invariants are computed with respect to a *transformation*, which is a mapping between

spaces of the same dimension. The goal is to measure the invariants from a perspective *projection* of the configuration, where the image may have a lower dimension than the object. We write $\mathtt{P}$ for the projection matrix that covers a 3D Euclidean transformation of the object followed by perspective projection onto the image. For *planar* objects the original and image spaces are the same dimension and $\mathtt{P}$ is simply a projective transformation represented by a 3×3 matrix. This is discussed in detail in section 2. For *three-dimensional* objects, the original and image spaces are no longer of the same dimension and $\mathtt{P}$ is a 3×4 matrix mapping 3D homogeneous coordinates onto the image plane. This is described in detail in section 3.2.

1.2.1 Indexing

One of the most important uses of invariants in vision is as indexing functions. In traditional model-based recognition systems (section 1.1), recognition proceeds by hypothesising a correspondence between image and object features, and then evaluating the hypothesis based on the consistency of the best projection of the model onto the image features. This constitutes simultaneously finding pose and performing recognition, and is generally of a complexity *linear* in the number of models in the library, since each model must be evaluated.

An index function provides direct access to a certain model in the model base without using specific information about the model, or model pose in advance. Ideally, the index function should uniquely retrieve a model from the library (thus facilitating *constant* time, as opposed to linear, access to the library), but in practice it is likely that a small number of models are retrieved with the same index. Even so, the search cost is considerably reduced below that of testing the full library. The index is typically a vector of independent invariant measurements.

More formally: the index is considered to be a vector, $\mathbf{M}$, which selects a particular model from the library. The index is a function $\mathbf{M}(\mathbf{f})$ of a set of *projected* object features only, where $\mathbf{f} = \mathtt{P}\mathbf{F}$, with $\mathbf{F}$ object features, and $\mathbf{f}$ the corresponding image features. Assuming that $\mathbf{M}$ can be computed from any image projection of the object features, then library values for $\mathbf{M}$ can be constructed simply by acquiring one or a few images of the object in isolation.

For planar objects, $\mathtt{P}$ is a planar projective transformation, $\mathtt{T}$, from the object in an arbitrary pose onto the image plane, and

$$\mathbf{M}(\mathtt{T}(\mathbf{F})) = \mathbf{M}(\mathbf{F})$$

i.e., the index has the same value computed on the original object and after the transformation (a scalar invariant). Each element of the index vector $\mathbf{M}$ is an invariant measure computed from a group of image features such as conics, lines, points and plane curve segments. A typical example is shown in figure 1. For 3D objects the same function cannot be applied to object and image, since they differ in dimension. However, again $\mathbf{M}$ is defined so that each element is a projective invariant of the 3D structure that is measured from the perspective image. Examples are given in section 3.2.

1.2.2 Invariance and representation

The term "invariance" does not simply refer to the viewpoint-invariant measurement vector described above. The term also includes the idea of an invariant *relation*, which is distinct from an invariant *value*. For example, the cross-ratio is an invariant value of four collinear points. The collinearity of the points is a projectively invariant relation between the points which is independent of the cross-ratio value. In the definition of generic geometric classes, the identification of invariant relations is often a more important issue for representation than the computation of specific invariant indexing values.

Another general aspect of the invariant approach is the symbiotic application of geometric and algebraic analysis. It is often the case that geometric insights provide the first clue to the nature of invariants for a particular object class. Then subsequent algebraic analysis can generalise and simplify invariant computation, and in turn provide additional insight.

1.2.3 Model acquisition from images

A model consists of the set of significant geometric features of the object boundary known up to a projective, or more restrictive, transformation (for example affine). Projective models can be constructed from images without requiring knowledge of the intrinsic camera parameters or known 3D ground control points. In the case of 2D objects the model can be acquired from a single image, for 3D objects more images are generally required. Model acquisition is discussed further in sections 2.3 and 4.3.

Before proceeding to the case of more general 3D object recognition, we review a mature system for 2D object recognition. This review will illustrate many of the issues in object recognition by invariants and provide a context for our more general discussion of recognition architectures at the end of the paper.

Notation

We adopt the notation that corresponding entities in two different coordinate frames are distinguished by upper and lower case. In general lower case is used for image quantities, and upper for 3D quantities. Vectors are written in bold font, e.g., $\mathbf{x}$ and $\mathbf{X}$. Matrices are written in typewriter font, e.g., $\mathtt{c}$ and $\mathtt{C}$. With homogeneous quantities, equality is up to a non-zero scale factor.

For smooth surfaces the *profile* (also called the *apparent contour*) is the outline of the surface in the image. It is the image projection of a surface curve, the *contour generator*, where rays from the optical centre are contained in the surface tangent plane.

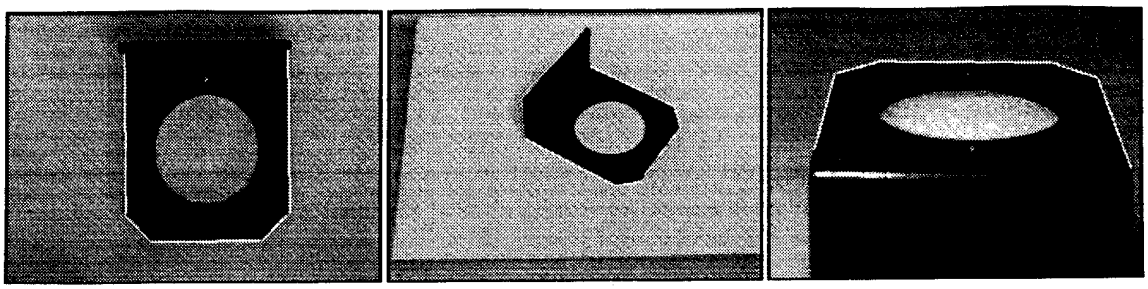
2 The planar recognition system

The use of *planar projective* invariants for planar object recognition is particularly appropriate and straightforward because a projective transformation between object and image planes covers all the major imaging transformations: the plane to plane projectivity models the composed effects of 3D rigid rotation and translation of the world plane (camera extrinsic parameters), perspective projection to the image plane, and an affine transformation of the final image which covers the effects of camera intrinsic parameters. Consequently, projective invariants, which are unaffected with respect to all of these parameters, have a high currency for this domain [40, 50, 55, 57, 58, 60, 70, 71].

Here we summarise the main features of a planar object recognition system that has been developed during the past four years. The projective representation of shape used in the system has the key advantages of simple model acquisition (direct from images), no need for camera calibration or object pose computation, and the use of index functions. Recognition proceeds by measuring invariants in the target image. The invariants are used to construct index vectors to select models from the library. If the index value coincides with that associated with a model, a recognition hypothesis is generated. Recognition hypotheses corresponding to the same object are merged to form *joint hypotheses*, provided they are geometrically compatible. The (joint) hypotheses are then verified. The system[1] has been tested on a large set of images and under varying levels of occlusion and clutter. A detailed description of this system appears in [60].

The projective nature of the representation is utilised at a number of stages in the recognition process, for example in both model acquisition and verification. In acquisition, any image provides a projective model of the object outline because the image and object planes are related by a projective transformation. This is because the object is mapped by a perspective transformation onto the image, and perspective is a restricted form of a projective transformation. In verification, the target image outline is projectively related to the model image outline. This follows because the target image outline is a projective transformation of the object outline, which is a projective transformation of the model image outline. Plane projective transformations are a group, and a sequence of projective transformations is equivalent to a single projective transformation (group closure).

[1] The system is called LEWIS. The motivation for this name is explained in section 4.4.



a b c

Figure 1: The lines used to compute the five line planar projective invariant for the above images are highlighted in white. The values are given in table 1.

Five line invariants		
Measured on	I_1	I_2
Object	0.840	1.236
Figure 1(a)	0.842	1.234
Figure 1(b)	0.840	1.232
Figure 1(c)	0.843	1.234

Table 1: Values of plane projective invariants measured on the object, and from images with varying perspective effects. The values vary (due to measurement noise) by less than 0.4%.

2.1 Projective invariants used

There are three different algebraic invariant constructions used in the system: five lines; a conic and two lines; and a conic pair. For example the two invariants of five lines are given by

$$I_1 = \frac{|\mathbf{N}_{431}||\mathbf{N}_{521}|}{|\mathbf{N}_{421}||\mathbf{N}_{531}|} \qquad \text{and} \qquad I_2 = \frac{|\mathbf{N}_{421}||\mathbf{N}_{532}|}{|\mathbf{N}_{432}||\mathbf{N}_{521}|}, \tag{1}$$

where $\mathbf{N}_{ijk} = (\mathbf{l}_i, \mathbf{l}_j, \mathbf{l}_k)$, $|\mathbf{N}_{ijk}|$ is the determinant, and $\mathbf{l} = (l_1, l_2, l_3)$ is the homogeneous representation of a line: $l_1 x + l_2 y + l_3 = 0$. (See [45] for the other invariants.) Table 1 gives examples of these invariants computed from the images shown in figure 1, which have varying degrees of perspective distortion. These are applicable to image curves that are "algebraic" (lines, conics). For non-convex smooth curve segments canonical frame invariants [45, 57] are used. These are constructed from projective coordinates of a concavity delineated by a bitangent.

In all cases there is tolerance to partial occlusion, i.e., the invariants can still be formed if part of the outline is occluded. This is a result of using *semi-local* invariant descriptions, i.e., not global, like moments of the entire boundary, and *redundancy*: there are a number of different descriptors for each object so that there is not an excessive requirement for any single object region to be visible. In the algebraic case lines and conics can still be extracted if part of the curve is occluded.

2.2 Architecture

The stages of recognition are shown in figure 2. In the following sections we describe these stages in sufficient detail to expose the important issues for consideration in extending these ideas to 3D object recognition.

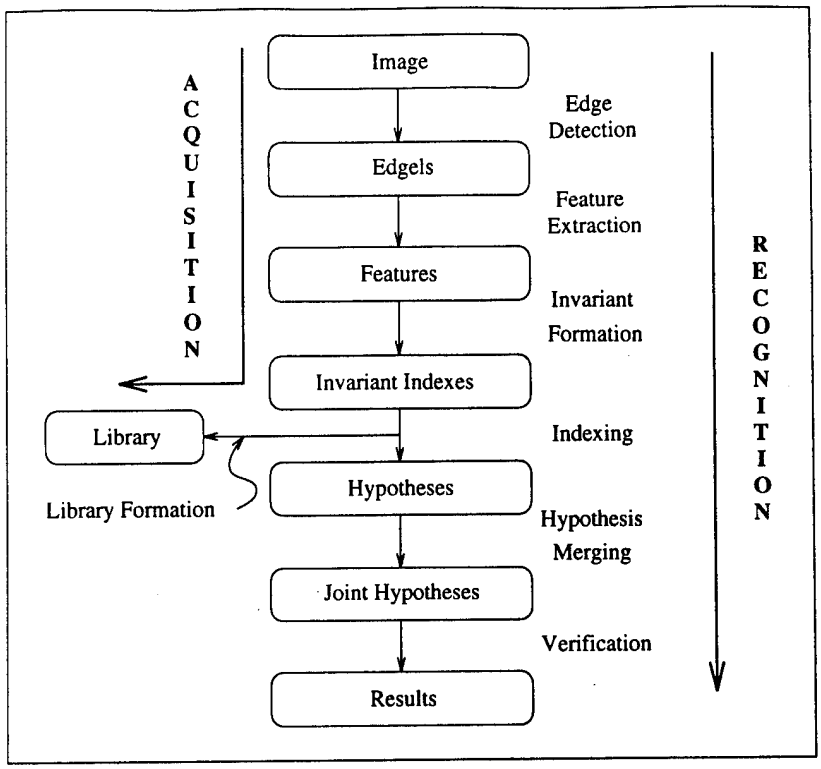


Figure 2: The recognition system has a single grey scale image as input and the outputs are verified hypotheses with associated confidence values. Many of the processes are shared by the acquisition and the recognition paths. The recognition system is similar to previous systems [28] in all but the indexing and hypothesis merging stages.

2.2.1 Feature extraction and invariant formation

The goal of the segmentation is the extraction of geometric primitives suitable for constructing invariants. In the algebraic case this involves straight lines and conics, and for non-algebraic curves, concavities delineated by bitangents. An example of algebraic segmentation is shown in figure 4.

A local implementation of Canny's edge detector is used to find edgels to sub-pixel accuracy. These edgels are linked into chains, extrapolating over any small gaps. Considerable advantage is made of local image feature topology. In many recognition systems, the local connectivity of edgel chains and fitted features is ignored; but we have found that feature grouping, based on the connectivity provided by edgel chains and proximity, allows index formation to have a low complexity with respect to the number of image features.

For algebraic invariants, connectivity enables efficient linking and ordering of line segments. For example, five line invariants are formed from sets of consecutive lines within single edgel chains at a cost that is linear in the number of lines in the scene (i.e., $O(l)$, compared to $O(l^5)$ if all groupings are attempted). For concavities, the curve again provides an ordering for the feature points used (bitangent and cast tangent points [73]) and only the two cases of global curve reversal have to be considered.

Once sets of grouped features, $\mathbf{f}$, have been produced, the algebraic and canonical invariants are computed. Each set of grouped features, or concavity curve, generally produces a number of invariant values which are collected into a vector $\mathbf{M}(\mathbf{f})$. The invariant vector formed by the above process represents a point in the multidimensional invariant space. The space is quantised to enable hashing. Each object feature group is represented by a collection of points that define a region in the invariant space, the size of which depends upon the measured variance in the invariant value[2].

[2] See section 2.3.

2.2.2 Indexing to generate recognition hypotheses

The invariant values computed from the target image are used to index against invariant values in the library. If the value is in the library a preliminary recognition hypothesis is generated for the corresponding object. Each type of invariant (e.g., five lines, conic pair) separately generate hypotheses.

This process is made more efficient using a hash table that allows simultaneous indexing on all elements of the measurement vector. In the experiments to date there has not been any significant problem with collisions in the hash table. Hash table collisions[3] should not be confused with the intersection of object invariant measurements in index space. These intersections lead to erroneous hypotheses which cost some effort during the verification stage, but are usually eliminated.

2.2.3 Hypothesis merging

Many collections of primitives may come from the same model instance: for example, an object consisting of a square plate with a circular hole in it admits four collections, each consisting of a conic and two connected lines. Each collection has an invariant which may generate a recognition hypothesis. Such a set of recognition hypotheses is *compatible* if a single model instance could explain all of them simultaneously. Prior to verification, compatible hypotheses are combined into *joint hypotheses.* There are number of reasons why hypothesis merging is desirable:

1. Backprojection and searching for image support is computationally expensive and it is more efficient to validate several hypotheses of the same object together.

2. More features facilitates more accurate least squares calculation of the back projection transformation (there are more matched model and image features), and consequently a reduced error in measuring image support.

3. Many hypotheses indexing the same object in a single part of the scene significantly increase confidence that the match is correct.

The hypothesis merging process is equivalent to forming an interpretation tree for the indexed object based on the features which index a particular model. The merging is controlled by topological and geometric compatibility. The topological consistency (ordering and connectedness) is illustrated in figure 3a. Geometric consistency is implemented efficiently by a second use of invariants — this time joint invariants between the feature groups used to compute each individual hypothesis. This is illustrated in figure 3b.

2.2.4 Verification

There are two steps involved in verification, both of which can reject a (joint) recognition hypothesis. The first is to attempt to compute a common projective transformation between the model features and the putative corresponding features in the target image. The second is to use this transformation to project the entire model onto the target image, and then *measure* image support.

Incorrect hypotheses arise because grouped image features happen to have an invariant value that coincides (within the error bounds) with one in the library. The features used to produce the matching model and image invariants provide sufficient constraints to compute the projective transformation between the model and image. In general this will be over constrained — many more constraints than the eight unknowns of the projective transformation are available. Consequently, if a common transformation cannot be computed the features are not projectively equivalent and the hypothesis is rejected.

Backprojection and subsequent searching involves the entire model boundary, not just the features used to form the invariant. Projected model edgels must lies close to image edgels with similar orientation (within 5 pixels and 15^0). If more than a certain proportion of the projected model data is supported (the threshold used is 50%), there is sufficient support for the model, and the recognition hypothesis is confirmed. The final part of the process is expensive as $O(10^3)$ edgels need to be mapped onto the image. Efficiency is improved by approximating the distances using the 3-4 distance transform of Borgefors [7].

[3] A hash table collision occurs when a number of models have the same hash index. Such a collision can occur when the number of hash buckets is smaller than the model population or when the hashing function is not uniform and causes many models to hash to the same bucket.

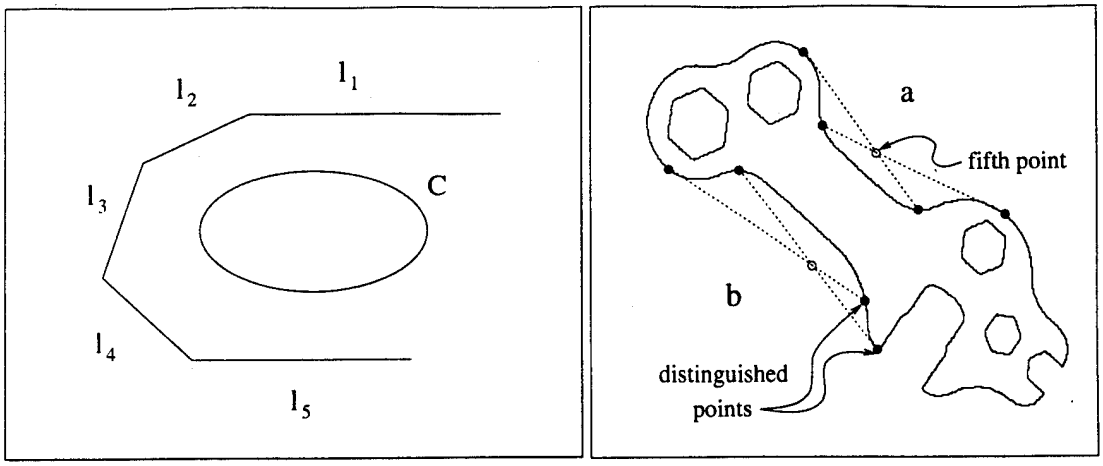


Figure 3: Hypothesis compatibility: (a) If the same model is indexed by a five-line invariant (due to lines $\mathbf{l}_i$, $i \in \{1,\ldots,5\}$), and a conic three-line invariant that is *compatible* with it (due to C and $\mathbf{l}_i$, $i \in \{2,\ldots,4\}$), then it is wise to verify both hypotheses together. The invariants are compatible if the ordering of the image lines are consistent with those on the model. (b) For a pair of concavity curves there are 8 distinguished points which could be used to form $2 \times 8 - 8 = 8$ different five point invariants. Rather than computing so many, which is unnecessary, invariants are computed between the four distinguished points of each concavity, and the 'central' point of the other. This yields four invariants, and does so using a symmetric construction. These invariants are sufficient to hypothesise compatibility.

2.3 Model acquisition and library formation

One benefit of using only projective representations, rather than Euclidean ones, is that a model can be acquired directly from an image. No special orientations or calibrations are required. Acquisition is simple and semi-automatic (for instance, curves do not have to be matched entirely by hand between images), using the same software for segmentation and invariant computation as used during recognition.

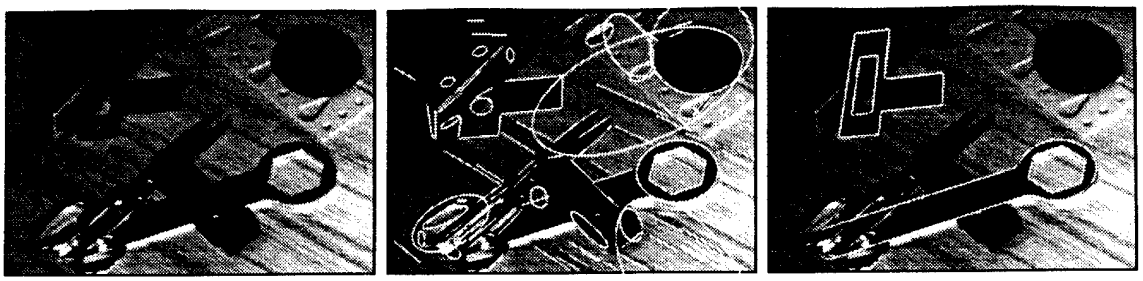
A model consists of the following: a name; a set of edges from an acquisition view of the object (used in the backprojection stage of verification); the lines, conics and concavities fitted to the edges; the expected invariant values and to which algebraic features and curve portions they correspond. (The mean and variance of the invariant values are computed from a variety of 'standard' viewpoints of the object.); and, finally, topological connectivity and geometric relations between feature groups used in the construction of joint invariants.

The library is partitioned into different sub-libraries, one for each type of invariant (e.g., one for the five-line invariant, another for the conic pair). Each sub-library then has a list of each of the invariant values tagged with an object name, and is structured as a hash table.

2.4 Recognition examples

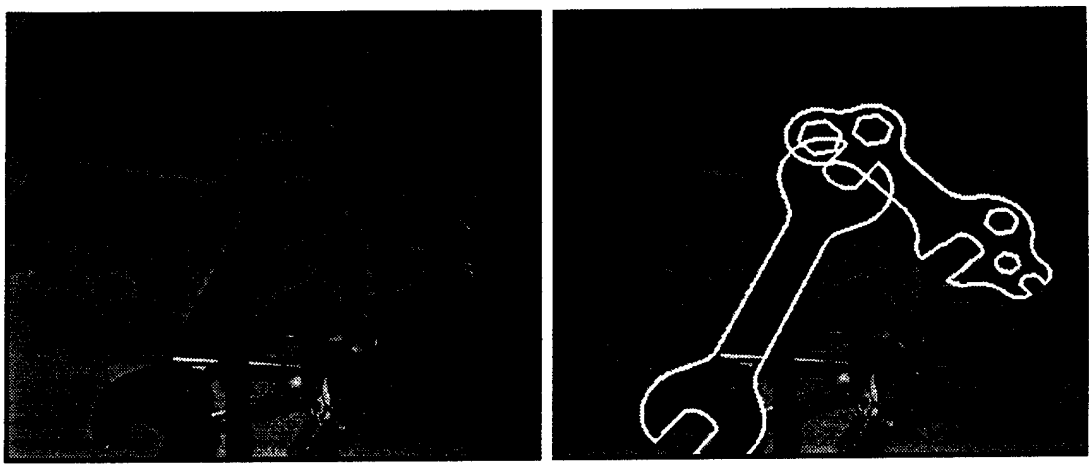
Only a small number of examples are included since others appear elsewhere [45, 58, 60]. In each case successful recognition is demonstrated by projecting the model outline onto the image. Segmentation for algebraic features is shown in figure 4. The two objects in the scene which are contained in the library are successfully recognised using algebraic invariants computed from these features despite substantial occlusion and clutter. 1049 invariants are computed which index 41 hypotheses. These are converted into 131 joint hypotheses[4] that have to be verified, of which 13 are rejected by first stage verification, based on valid projective transformations, and 78 require the second stage, based on image support. Figure 5 shows recognition based on canonical frame invariants. The algebraic and canonical frame invariants can be independently applied to an image to recognise objects of both types. Figure 6 shows an example of

[4] The joint hypothesis list consists of combinations of compatible hypotheses, together with all the original hypotheses.



a b c

Figure 4: (a) A scene containing two objects from the model base, with fitted lines (100 of them) and conics (27) superimposed in (b). These numbers are typical for images of this type. Note that many lines are caused by texture, and that some of the conics correspond to edge data over only a small section. The lines form 70 different line groups. (c) shows the two objects correctly recognised, the lock striker plate matched with a single invariant and 50.9% edge match, and the spanner with three invariants and 70.7% edge match.



a b

Figure 5: Single concavities are sufficient to recognise the two model instances shown in (b). The redundancy of the canonical frame representation gives much better tolerance to occlusion than global shape methods. The left hand object gained 67.1% boundary support, and the right object 81.6%.

recognition for both index methods together.

2.5 Summary of performance

Figure 7 shows data collected over fifty evaluations of the recognition system in which a single object from the model base was placed in a scene and partially occluded by other objects that are not in the model base (clutter). The average number of hypotheses computed as more models were added to the library is plotted. The first model added to the library always corresponded to the actual object in the scene. With 33 models in the library, on average 15.8% of the hypotheses were for the correct model. Although predominately linear, the graph has a very low gradient.

The real benefit of indexing becomes apparent when one considers how many hypotheses would be produced if an alignment technique is used, maintaining the same grouping methods. On average, over 2000 feature groups are produced for each image, and so 2000 hypotheses would be generated for each model feature group in the library (generally there are four or five feature groups per object and so the situation would be far worse). This would result in about 7×10^4 hypotheses for the entire model base compared to fewer than 60 produced when indexing is used. As these all have to be verified it is clear that indexing produces a dramatic improvement in the system efficiency.

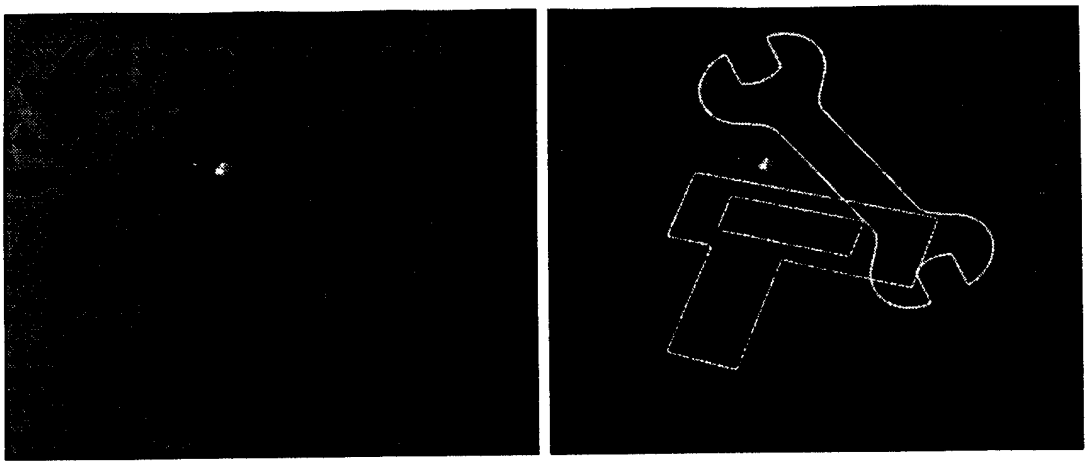


Figure 6: A demonstration that both types of invariant index can be used to recognise objects in a single image. The bracket is indexed using algebraic invariants and the spanner is indexed using the canonical frame signature.

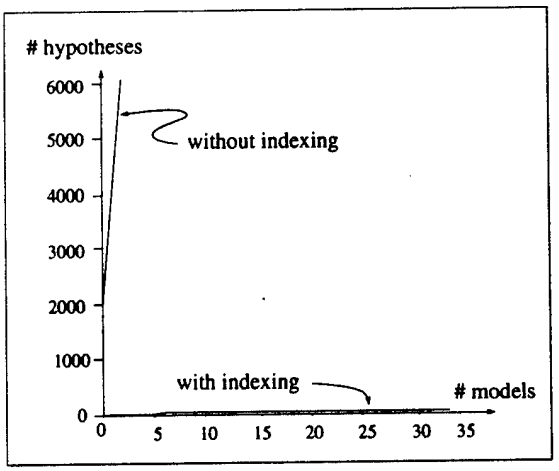


Figure 7: The number of hypotheses that have to be verified as the number of models in the library is varied. The results show an average over fifty scenes containing only one object in the library, but with other clutter and occlusion present. Over 2000 indexes are created for the scene, which corresponds to the number of hypotheses that would have to be verified **per model feature group** if alignment is used. Therefore, there is a rapid linear growth in the number of hypotheses created as the model base is expanded. However, the number of hypotheses created through indexing remains substantially lower — there is a linear growth, but with a very low constant of proportionality.

2.6 Appraisal

This system is an effective and reliable recognition system, and demonstrates a number of features that are likely to be important in building the next-generation system:

- **Hypothesis combination:** simply verifying each indexed model is prohibitive, particularly for complex objects with many features. Hypothesis combination is an effective way of combining semi-local information from different parts of the scene to obtain a single recognition hypothesis.
- **Untrustworthy and expensive verification:** verification is neither cheap nor reliable, as it involves back-projecting a large number of features, and testing for distance between those features and possibly unrelated image events. Verification scores can be incorrectly high, due to background clutter and texture which leads to false positives. The next generation system must have more extensive verification mechanisms using region properties as well as edge geometry. Also much more careful analysis of edge and junction intensity events must be carried out with respect to constraints imposed by the model. For example, specialised corner detection can be supervised by the model hypothesis.
- **A need for global scene analysis:** in many cases, ambiguities arise which must be settled globally by a scene analysis approach: for example, does a given image line come from object A or object B? Are the recognition hypotheses consistent? The lack of local support for a model hypothesis can be augmented by global relationships, e.g., A is on top of and partially occluding B. In this case we can predict the features which are potentially available to support hypotheses for B, once A is recognised.

Next, we take up the problem of 3D object recognition. First, the central question of the existence of invariants for the perspective projection of general 3D structures is discussed.

3 Extending invariant descriptions to 3D structures

Much recent debate has focused around a theorem, proven by a number of authors [9, 11, 42], which states that invariants can not be measured for a 3D set of points in general position from a single view. The theorem has frequently been misinterpreted to mean that *no* invariants can be formed for three dimensional objects from a single image. For the theorem to hold, however, the points must be completely unconstrained (like a cloud of gnats). If a 3D structure is *constrained*, then invariants are available. For example, six points constrained to lie on two planes in a "butterfly" configuration, as in figure 8, have a cross-ratio that can be measured in the image. This is a projective invariant of the entire 3D structure, and not simply a disguised planar invariant, since each plane contains only four points (five coplanar points are required to form a plane projective invariant from points alone).

In fact, a set of points in general position in space is a poor model of what we see: the world is full of curves, polyhedra, and surfaces; sets of isolated points are an irregular occurrence. An analogue of the above "no-invariants" theorem, in the case of surfaces, would be to ask whether a *generic* surface has invariants measurable in a single image from its profile. Other than qualitative descriptions such as non-convexity (from the sign of the profile curvature [32]) and the Euler characteristic (from the profile of a *transparent* surface) no projective invariant can be obtained. A similar result holds for space curves. However, if the surface satisfies constraints, much can be recovered from a single image, as the following section demonstrates.

3.1 Object classes

The form of the constraint on the object defines an object *class*. The class determines both the process by which the 3D invariants are measured in images, and the particular segmentation and grouping strategies that are applied during "early vision". For example, surfaces of revolution define a class, with a specific vase or wine glass being particular instances of the class. Projective invariants of the 3D surface can be recovered from the image profile, and further the two matching "sides" of the profile are projectively equivalent (section 3.5). That is, one side can be mapped onto the other by a projective transformation.

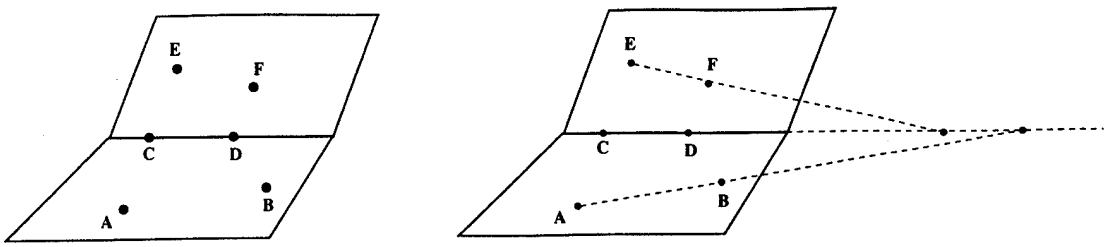


Figure 8: A "butterfly" configuration of six 3D points with a projective invariant measurable from a single perspective image. Points $ABCD$ and $CDEF$ lie on two planes intersecting in the line CD. The lines AB and EF intersect the line CD generating four collinear points. This construction can be carried out in 3D and the image to generate corresponding points. The cross-ratio of these points is the projective invariant. Note that the planes can articulate about the line CD without altering the value of the cross-ratio. Many analogous structures exist e.g., if the points AB are replaced by a line.

The segmentation and grouping for this class is guided by the association of these projectively related image contours.

It is important to distinguish this notion of geometric *class* from the idea of a generic *type*. For example, the class of rotationally symmetric objects is not the same as the generic type category of *wine glass*. There can be many different shapes of wine glass but the class of rotationally symmetric objects is still larger and does not capture the functional notion of a wine drinking container. A related discussion of class is given by Moses and Ullman [42] who contrast the notions of generic and specific classes with regard to recognition functions.

Another significant aspect of the class definition is its imposition of constraints which can be measured and verified in the image. This consideration is a significant departure from the hypothesise and test paradigm of conventional model-based recognition systems operating on specific objects. Here, the class assumption can be immediately confirmed without committing to the full chain of recognition processing. For example, for a rotationally symmetric object, the two "sides" of the image outline are related by a planar projective transformation (section 3.5). This relation can be immediately tested when a pair of image profile curves are hypothesised as belonging to an object of class *rotationally symmetric*.

In the following sections we catalogue a number of object classes where each is defined by an associated constraint. In each case the recovery of invariants is illustrated and other geometric consequences, such as invariant relations, described.

3.2 Definitions — 3D projective invariants

In what follows, we assume a perspective camera with unknown internal parameters, and measure only projective properties in the image. In turn, this means in general that only projective properties of the 3D objects can be recovered. Algebraically, the camera is modelled as $\mathbf{x} = \mathbf{PX}$:

$$k \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

where $(x, y)^T$ are image coordinates, and $(X, Y, Z)^T$ world coordinates, and k is a scaling: in this case, $k = (p_{31}X + p_{32}Y + p_{33}Z + p_{34})^{-1}$.

We now introduce 3D projective invariants because they are the basis for image invariants that we can hope to recover from a single view of a constrained structure. These are invariants under projective transformations of $\mathcal{P}^3$. A projective transformation of $\mathcal{P}^3$ can be written as:

$$k \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where k is again the appropriate scaling to ensure the fourth coordinate is one. Fifteen parameters are required to define the 3D projective transformation matrix up to an arbitrary scale factor. Thus five 3D points are sufficient to construct a projective coordinate system. A sixth point will then have invariant 3D coordinates in the projective basis defined by the other five. These 3D point invariants can also be interpreted as the cross-ratio of tetrahedral volumes computed by taking determinants of point coordinates, four at a time.

For example, an invariant for six 3D points is given by

$$I_{6pts}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5, \mathbf{X}_6) = \frac{|\mathbf{X}_1\ \mathbf{X}_2\ \mathbf{X}_3\ \mathbf{X}_4|\,|\mathbf{X}_1\ \mathbf{X}_2\ \mathbf{X}_5\ \mathbf{X}_6|}{|\mathbf{X}_1\ \mathbf{X}_2\ \mathbf{X}_3\ \mathbf{X}_5|\,|\mathbf{X}_1\ \mathbf{X}_2\ \mathbf{X}_4\ \mathbf{X}_6|}$$

where $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)^t$. This invariant has the familiar property of invariants that

$$I_{6pts}(\mathbf{X}'_1, \mathbf{X}'_2, \mathbf{X}'_3, \mathbf{X}'_4, \mathbf{X}'_5, \mathbf{X}'_6) = I_{6pts}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5, \mathbf{X}_6),$$

i.e., both the value and the form of the expression are unaffected by the transformation.

By assuming that a set of constraints hold among the 3D projective invariants of a point set, it becomes possible to measure 3D projective invariants in a single view. The following section illustrates the nature of these constraints and provides a geometric interpretation for the measurable invariants.

3.3 Constrained Point Sets

It is possible in general to predict whether invariants of a three dimensional structure can be measured from images, by counting the number of image measurements available. While such counting arguments cannot cover every degeneracy, and therefore never offer a proof that an invariant is or is not possible, they offer a useful guide to what is likely to be true. A complication in counting the degrees of freedom of a geometric configuration, and the number of parameters of a transformation, is the existence of isotropies. An isotropy is an action of a transformation which does not alter the geometry of a configuration. For example, translation along the tangent direction of a line or rotation about the centre of a circle does not affect the structure. Therefore an isotropy reduces the effective number of transform parameters, and generally increases the number of invariants.

Consider a 3D configuration M. Plane projective invariants are denoted I_2, and projective invariants of 3D denoted I_3. Then,

> For m perspective images of M, if there is no isotropy group acting in $\mathcal{P}^3$, then to recover n_{I_3} functionally independent invariants of the three-dimensional structure M from image information alone, the following inequality must be satisfied:
>
> $$m \times n_{I_2} \geq n_{I_3} + 3m$$
>
> where n_{I_2} is the number of functionally independent plane projective invariants of the image of M. If there is an isotropy group of dimension ($\dim I_S$) acting, then (provided $\dim I_S \leq 3$) the following inequality must be satisfied:
>
> $$m \times n_{I_2} \geq n_{I_3} + m(3 - \dim I_S)$$

We sketch the reasoning when there is no isotropy group acting for the case of a single image. The image projective invariants are functions only of the projective invariants of the configuration consisting of M *taken together* with the optical centre, $\mathbf{O}$. To see this, consider a projective transformation of $\mathcal{P}^3$. This projectively distorts the $\{M, \mathbf{O}\}$ configuration and the image plane. However, the image plane geometry is transformed by only a plane projective transformation. This means that the projective invariants of both the image configuration and the 3D configuration are unaffected. The image projective invariants can depend only on the rays linking $\mathbf{O}$ to points of M, and depend, therefore, on the optical centre $\mathbf{O}$ as well as on M. Since the image projective invariants are unaffected by the position of the image plane, the relationship between the 2D projective image invariants and the 3D projective object invariants is a function only of the three unknown coordinates of the centre of projection. Provided there are three or

	A	B	C
DOF	16	14	12
$\dim I_S$	0	2	4
n_{I_3}	1	1	1
n_{I_2}	4	2	1
dof of O that matter	3	1	0
Counting Relation	$4 = 1 + 3$	$2 = 1 + 1$	$1 = 1 + 0$

Table 2: Examples of the counting argument for various butterfly like structures. $A = 6$ point butterfly; $B =$ butterfly with two points of wing replaced by line (4 points one line); $C =$ butterfly with lines on both wings (2 points 2 lines).

more such image invariants, it is possible (in principle) to eliminate the (unknown) contribution of the optical centre.

The counting argument then simply relates the number of unknowns and the number of measurements: in m views there are $3m$ unknowns for the optical centres, and n_{I_3} unknown 3D projective invariants for the configuration M; the number of measurements is n_{I_2} in each of the m images. Note that, like most such arguments, the condition is necessary but may not be sufficient; this means that there could be cases where the counting argument indicates that invariants can be measured from the image, when in fact they cannot. The significance of the argument is that it indicates where a further analysis may be useful.

As an example, consider the case of six points in space which have three 3D projective invariants, as discussed above. If we specify, or assume, the values for two of the invariants, then we can compute the value of the third from a single image. The so-called butterfly configuration in figure 8 is an example where we assume that two of the 3D invariants are zero, which corresponds to the coplanarity of two sets of four points in the six point configuration. The counting argument goes as follows: the number of degrees of freedom for the image points is 12 (2 for each point) less 8 for the plane projective group gives $n_{I_2} = 4$. For 6 points in space on two planes there are 16 degrees of freedom (3 for each point, less two for the planarity constraints) less 15 for the 3D projective group, gives $n_{I_3} = 1$. There are also three unknown coordinates of the centre of projection. Thus the counting argument shows that the unknown 3D invariant can be measured in a single view, i.e.,

$$1 \times 4 \geq 1 + 1 \times 3,$$

Similar counts for a number of butterfly analogues in the case of isotropies are given in table 2. Sparr [63] has constructed many other examples of butterfly-like configurations, and provides a method for generating such invariants algebraically.

The counting argument is used in this manner to focus attention on configurations where invariants may be available. As a further example, consider recognising algebraic surfaces from their profiles. In this case, the surface has degree d, and has $[(1/6)(d+3)(d+2)(d+1)-1]-15$ functionally independent projective invariants. The profile has degree $d(d-1)$, and has $[(1/2)(1-d+d^2)(2-d+d^2)-1]-8$ functionally independent projective invariants. For $d > 2$, the number of invariants of the profile substantially exceeds the number of invariants of the surface, and so it is reasonable to expect to recover invariants from the profile of an algebraic surface. In fact, such invariants can be recovered, though the procedure is complicated; details are given in [22].

3.4 Repeated Structure

Structures that repeat in a single image of a scene are equivalent to multiple views of a single instance of the structure. Thus, for example, a view of two similar cars in a car park where the cars are parked within translations of one another, is equivalent to a stereo pair of images of one such car, with the cameras related by a pure translation. The 3D shape of the car can be recovered by the familiar techniques of stereopsis. More formally,

A repeated structure is defined by a geometric structure $\mathcal{S}$, and a 3D transformation $\mathcal{T}$, which generates a transformed *copy* of $\mathcal{S}$, i.e., $\mathcal{S}' = \mathcal{T}(\mathcal{S})$. Both $\mathcal{S}$ and $\mathcal{S}'$ are viewed in the same perspective image.

In many cases the internal calibration parameters of the camera will be unknown. In this case a single image of a repeated structure is mathematically identical to an uncalibrated stereo pair where the two cameras are related by the transformation between $\mathcal{S}$ and $\mathcal{S}'$. It has been shown by Faugeras [16] and Hartley *et al.* [29] that if one carries out stereo reconstruction from two uncalibrated perspective images, the reconstruction can differ from the actual 3D Euclidean geometry of the object by a 3D projective transformation. Thus, 3D projective invariants of this recovered structure have the same value as projective invariants measured on the actual Euclidean structure.

The equivalence with stereo means that epipolar structure can be defined within a single image and represents the geometric relationship between corresponding features on the object copies. As a simple example, consider the case where $\mathcal{T}$ is a 3D translation, i.e., $\mathcal{S}$ and $\mathcal{S}'$ are related by a simple 3D translation. In this case, it can be shown [41] that *affine*, rather than projective, 3D structure can be recovered. Lines joining corresponding points on $\mathcal{S}$ and $\mathcal{S}'$ are parallel in 3D and are imaged as a set of lines converging to a vanishing point. These imaged correspondence lines and vanishing point are the analogue of "epipolar lines" and "epipole", and these terms will be used from now on. For translation only, there is a single epipole and corresponding points in $\mathcal{S}$ and $\mathcal{S}'$ lie on the same epipolar line. We call this convenient correspondence relation *auto-epipolar correspondence*. This correspondence relation is an example of the more general idea that repeated 3D geometric structure imposes 2D constraints on corresponding image features which can be used to advantage in grouping and verification.

We reserve the epipolar terminology for the case where the centres of projection of the two cameras are displaced. For some repeated structures, the transformation between $\mathcal{S}$ and $\mathcal{S}'$ does not alter the camera centre and thus does not yield an epipolar structure. However, it is still possible to construct a correspondence structure in the image which is not an epipolar geometry but has many similar advantages. An example of this is given in section 3.5 for surfaces of revolution.

Thus we have two recurring issues that arise in the context of repeated structures and in most cases where object class produces invariants that can be measured in a single image view:

1. The computation of image-measureable 3D invariants.
2. The correspondence relationship between the *imaged* features of the 3D structure, and associated grouping strategies.

In the next few sections we review some mature examples of repeated structure [36, 47] where the discussion is organised around these two issues.

3.4.1 Bilateral Symmetry

In the case of a single bilateral symmetry, the repeated structure is the half object on one side of the symmetry plane. A single camera imaging a bilaterally symmetric object is equivalent to two identical cameras, viewing the half structure, where one camera is transformed to the other by a reflection in the object symmetry plane. A similar observation was made by [26, 39], though in the context of a calibrated camera. Below we give examples of 3D point sets and space curves with a single bilateral symmetry.

3D geometry Lines joining corresponding points (on either side of the symmetry plane) are parallel and orthogonal to the plane of symmetry. There is a natural coordinate system provided by these correspondence directions and the symmetry plane (figure 9). The correspondence lines intersect the symmetry plane at the midpoint of the corresponding points.

Measuring Invariants Perspective projection does not preserve mid-points. However, the images of the 3D midpoints can be computed (see below). All 3D mid-points are co-planar (they lie on the symmetry plane). There is a projective transformation between the set of imaged mid-points and the 3D points on the plane of symmetry. Thus planar projective invariants can be measured in the image from the computed midpoints.

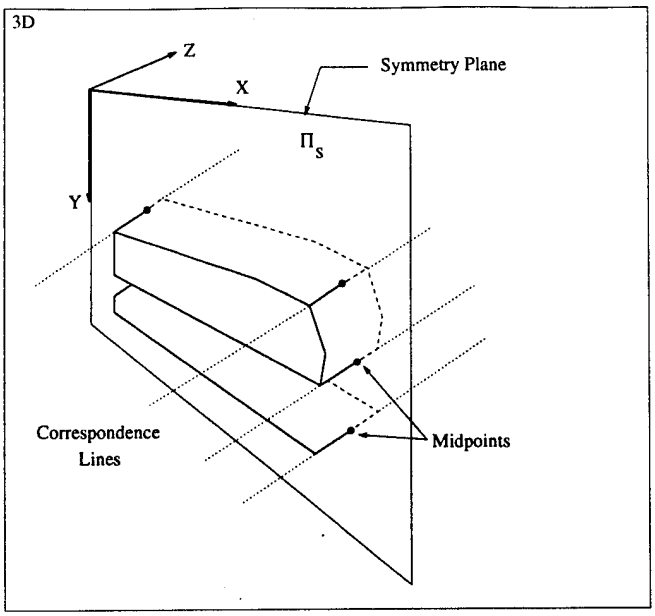


Figure 9: The natural coordinate frame for an object with bilateral symmetry. The XY plane is the plane of reflection, and the Z axis is parallel to lines joining corresponding points. Note, all mid-points are coplanar.

The image of the 3D midpoints can be computed using a property of equally spaced points (see [64]): three collinear points, separated by the same distance, and taken with a point at infinity have a harmonic cross-ratio. Since the point at infinity on the line joining two corresponding points is imaged as a vanishing point (see below, and figure 10), it can be observed. Thus, the position of the midpoint in the image can be computed from the image coordinates of the corresponding points and from the image coordinates of the vanishing point. Furthermore, since computing a point that has a fixed cross-ratio with respect to three other points is linear, there is a unique solution. Other geometric methods for computing the imaged mid-point are available, based on point pairs or triplets.

The 3D structure can be reconstructed up to a projective ambiguity, based on the equivalence with uncalibrated stereo [16, 29]. 3D projective invariants can be measured from this recovered structure. In the case of bilateral symmetry, structure is recovered to better than a projective ambiguity because of the orthogonality constraints available in the "natural" coordinate frame [18, 60].

Correspondence and Grouping The epipolar structure in this case arises from the parallel lines joining corresponding points (on each side of the object). Under perspective projection, these correspondence lines (the epipolars) image to a family of lines converging to a single vanishing point (the epipole). The epipole can be determined using two pairs of corresponding points (figure 10). Once the epipole has been computed, further correspondences are found by a 1D search on the epipolar line. This is a recurring theme — the constraints that define the object class not only show how invariants may be recovered, but also facilitate and direct the image grouping.

We demonstrate two examples of bilateral symmetry, a polyhedral point set and a space curve.

1. **3D polyhedron:** figure 11 shows different views of the 3D reconstruction of a stapler obtained from a single view. The reconstruction is placed in a Euclidean frame to give a normal presentation of the object shape, but any projective frame could be used.

2. **A space curve:** corresponding points on the two imaged space curves are determined using the epipolar geometry. Four different views of the reconstruction for the outline of a spoon are shown in figure 12 (the 3D projective representation has again been constrained to lie in a believable Euclidean frame).

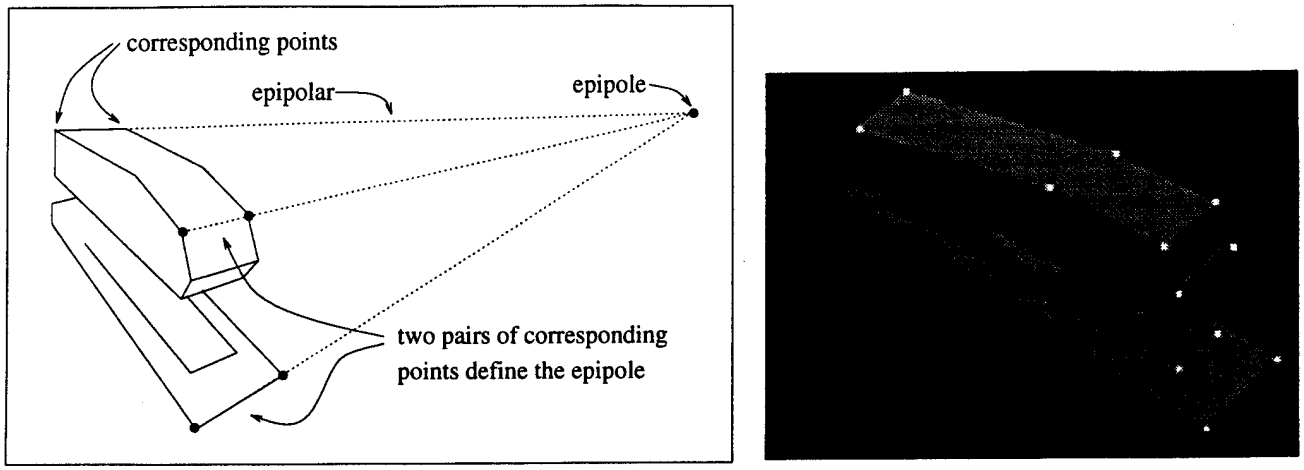


Figure 10: (a) The epipole can be located using the intersection of lines between two corresponding points on a bilaterally symmetric object (the points are marked by solid circles). Epipolars can then be constructed through the epipole to aid correspondence. (b) Typical corresponding points determined in this manner.

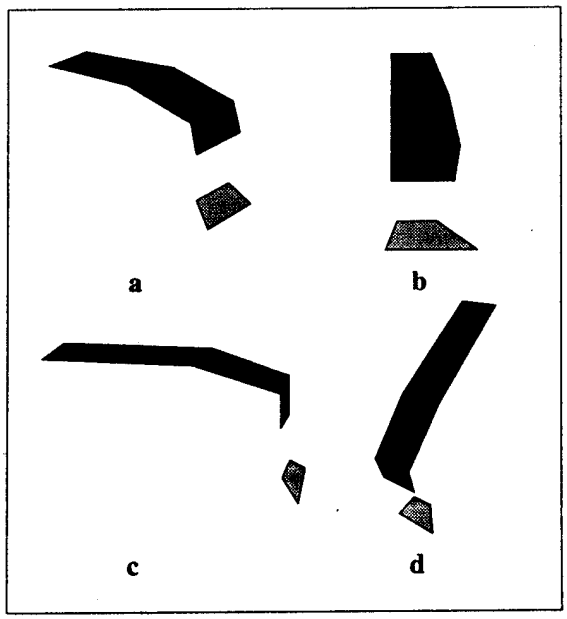


Figure 11: Three dimensional structure is recovered, modulo a projectivity, from the single view of the points marked on the stapler in figure 10b. Four typical views are shown with only (a) at a viewpoint close to that of the original image. Note the collinearity of the line segments in (b), this demonstrates the accuracy of the recovered structure.

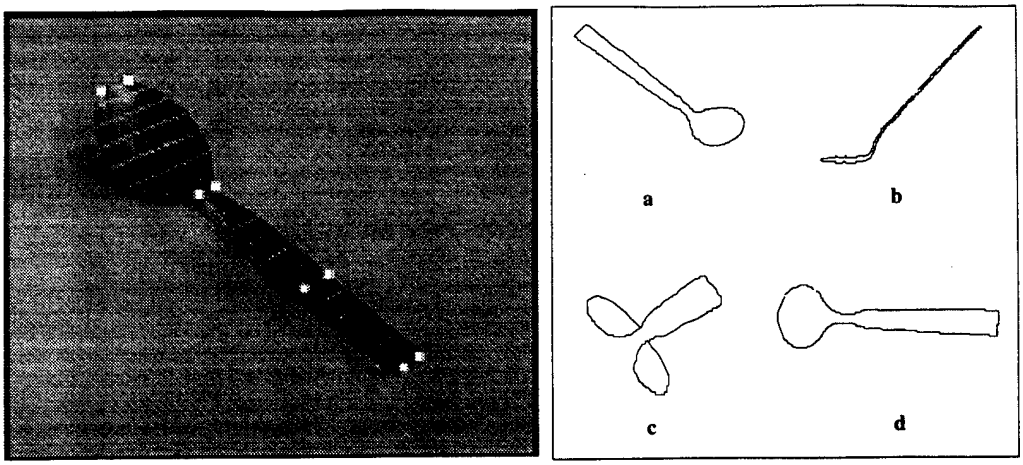


Figure 12: A single view of an object with a plane of bilateral symmetry, such as a teaspoon, is sufficient to allow a full 3D projective reconstruction. Only two pairs of distinguished points are needed for the approach, these are recovered from surface markings and can be used to determine the epipolar structure of the image. Using this epipolar structure an arbitrary number of correspondences can be produced. Four different views are shown of the 3D reconstruction computed from the image. The construction works very well: note the planarity of the handle recovered in (b), and the full 3D shape in all of the images.

The reasoning outlined above can be applied to objects with more than one bilateral symmetry [18] and to objects projectively equivalent to ones with bilateral symmetry.

3.4.2 Translational Repetition

In this case the structures, $\mathcal{S}$ and $\mathcal{S}'$, are related by a 3D translation. As described above, the structure of $\mathcal{S}$ can be recovered up to an affine ambiguity, from a single image of the duplicate structure.

Measuring Invariants Affine invariants are computed from the perspective image in three stages. First, structure is recovered up to a projective ambiguity using uncalibrated stereo. Second, the plane at infinity [64] is determined in this projective coordinate frame as follows: a line on $\mathcal{S}$ is parallel to its counterpart on $\mathcal{S}'$, so the intersection of corresponding lines is a point $\mathbf{P}$ on the plane at infinity. The image, $\mathbf{p}$, of $\mathbf{P}$ is computed by intersecting the imaged corresponding lines. Since $\mathbf{p}$ lies on both lines its 3D position, $\mathbf{P}$, can be determined by stereo. Three such points determine the plane at infinity. Third, the structure is projectively transformed such that the plane at infinity has the standard form $X_4 = 0$. The structure is then known up to an affine ambiguity, and affine invariants measured from this structure have the same value as invariants measured on the 3D Euclidean structure $\mathcal{S}$.

Correspondence and Grouping As in the bilateral symmetry case, lines joining corresponding 3D points are parallel. The image correspondence is again auto-epipolar (all corresponding lines intersect in a single epipole).

As an example, invariants are calculated from the images of two translated polyhedral structures shown in figure 13. The translation between the duplicated structure differs in each case demonstrating that the invariants are associated with the structure itself, i.e., $\mathcal{S}$. Affine invariants are computed for the 3D vertex positions which are computed using the epipolar geometry of the translated copies. The values of the invariants are given in table 3. The differences between the invariants computed from each image are small, even though the translation vector between the speakers and the viewpoint varies significantly. In many cases of such repeated structures, the object copies are rigidly connected, but this example illustrates that the affine invariants are independent of the translation vector as well.

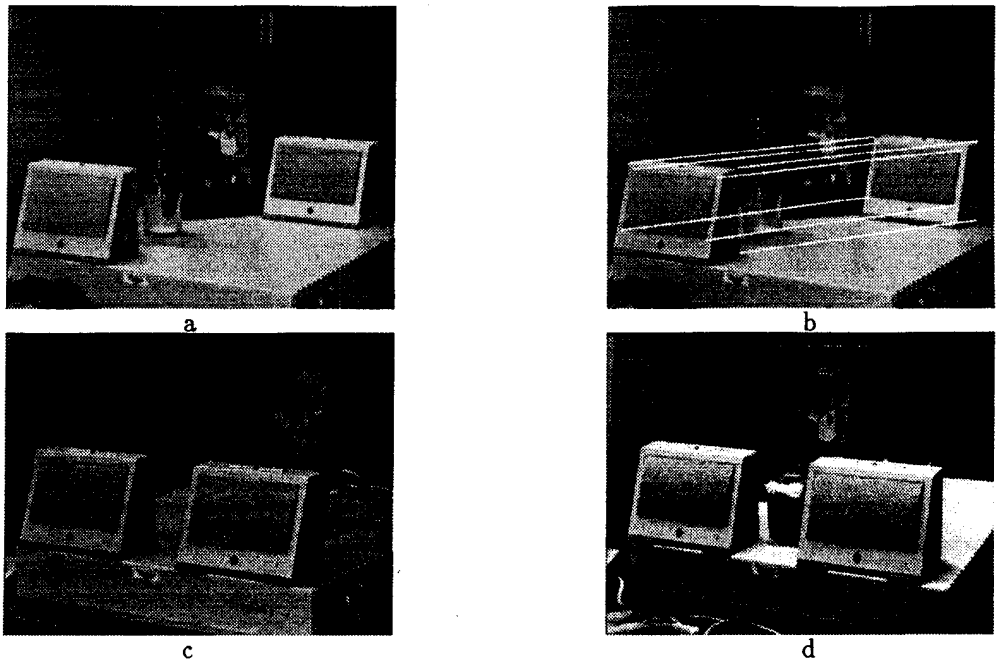


Figure 13: One object (a speaker) repeated under translation. The epipolar correspondence lines for image (a) are shown in (b). The translation vector is different for images (a) and (c) and the same between (c) and (d). Affine invariants computed from these images are compared in table 3.

image a	image c	image d
-0.2249	-0.2324	-0.2317
-0.0642	-0.0685	-0.0626
1.2833	1.2979	1.2849

Table 3: Comparison of 3D affine invariants computed for the speaker from figure 13. The invariant is the 3D position of one corner of the speaker in an affine frame defined by four other points on the speaker. The values are fairly stable, even though the images have different translation vectors and viewpoints.

3.4.3 Other Repeated Structures

The notion of structure repetition under a transformation is extensible to more general situations. It is not necessary that the copies be Euclidean equivalent and repeated under translation. The copy transformation can be a full 3D projective transformation whilst still preserving an epipolar correspondence within the image. The 3D reconstruction of the object geometry is then known only up to a 3D projective transformation of space.

In the case that there are three or more Euclidean equivalent structures, the geometry can be recovered up to a 3D similarity. This follows from the equivalence of this case to three views of a single object taken with an identical camera, where it has been demonstrated that structure can be recovered up to a 3D similarity [17].

It is also interesting to speculate about *approximately-repeated* structures. Suppose that the structure is not repeated according to a rigid 3D transformation but is only an approximation to such a transformation. This approximate repetition occurs in natural objects such as animals and vegetation. It seems that the invariants which one can compute from an idealised form of the approximate repetition will not be very far from an invariant description of the actual structure. For example in a bunch of grapes, it can be assumed that each grape is copy of the other under an affine transformation or perhaps even a scaled Euclidean transformation. Another example is texture which can be thought of as a statistical repeated

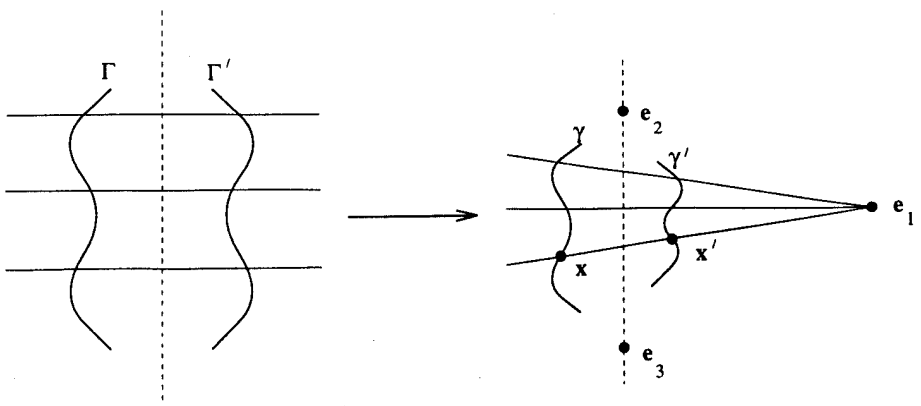


Figure 14: The profile of a surface of revolution is projectively equivalent to two curves with bilateral symmetry. Under a projective transformation parallel correspondences (left) converge to a vanishing point (right). Corresponding points $\mathbf{x} \leftrightarrow \mathbf{x}'$ are related in this case by a particular plane projective transformation, $\mathtt{T}$, called a planar harmonic homology. The transformation has a line of fixed points, the image of the axis of symmetry, which result from two of the eigenvalues of $\mathtt{T}$ being equal. There is also a fixed point, $\mathbf{e}_1$, not on the line, called the centre of the homology which defines correspondences between symmetrical points on each side of the contour. That is, corresponding point pairs and the centre of the homology are collinear. The cross-ratio of $\mathbf{e}_1$, the corresponding points $\mathbf{x}, \mathbf{x}'$, and the intersection of their join with the axis, is harmonic. (The line of fixed points is $\mathbf{e}_2 \times \mathbf{e}_3$, where $\mathbf{e}_2$ and $\mathbf{e}_3$ are the eigenvectors with equal eigenvalues. The third eigenvector, $\mathbf{e}_1$, is distinct and non-zero, and is the centre for a pencil of fixed lines.)

structure.

3.5 Rotational Symmetry

Surfaces of revolution have had considerable attention, though generally with calibrated cameras [13], or as a special case of a generalised cylinder [53, 72].

Profile Geometry The image curves forming the two "sides" of the profile are related by a plane projective transformation, $\mathtt{T}$, with the property that $\mathtt{T}^2 = \mathtt{I}$. Such a projective transformation is called a *planar harmonic homology* [64]. It arises in this case because the image transformation is a conjugate reflection (whose conjugating element is a projective transformation). To see this, construct the plane containing the axis of the surface and the optical centre. The surface then has a mirror symmetry in this plane, as does the cone of rays through the optical centre and tangent to the surface. This cone yields the profile when it is intersected with the image plane. Clearly, the contour generators are, in general, space curves, related by a mirror symmetry in space. If the image plane is perpendicular to the plane of symmetry, then the profile has a mirror symmetry; but the profile for any other image plane is within a projective transformation of the perpendicular plane.

In this case, there is no epipolar geometry defined, since reflection in the symmetry plane does not move the optical centre. However, a correspondence relation still exists and is generated by the planar homology between the opposing sides. A planar *harmonic* homology (see figure 14) is a special case of a planar homology (see figure 20b) for which the characteristic invariant of the homology is harmonic [64]. For planar homologies there is a fixed point which is the centre of a pencil of fixed lines which define correspondence pairs. That is, corresponding points lie on the same line of the pencil, in the same manner as the epipolar geometry of translated cameras.

Measuring Invariants The intersections of "corresponding" profile bitangents lie on the projection of the object's axis. The image intersection points are projections of the intersection points between planes bitangent to the surface and the 3D object axis. This point is viewpoint independent. This is shown schematically in figure 15. Four such points are sufficient to measure a cross-ratio (the points are

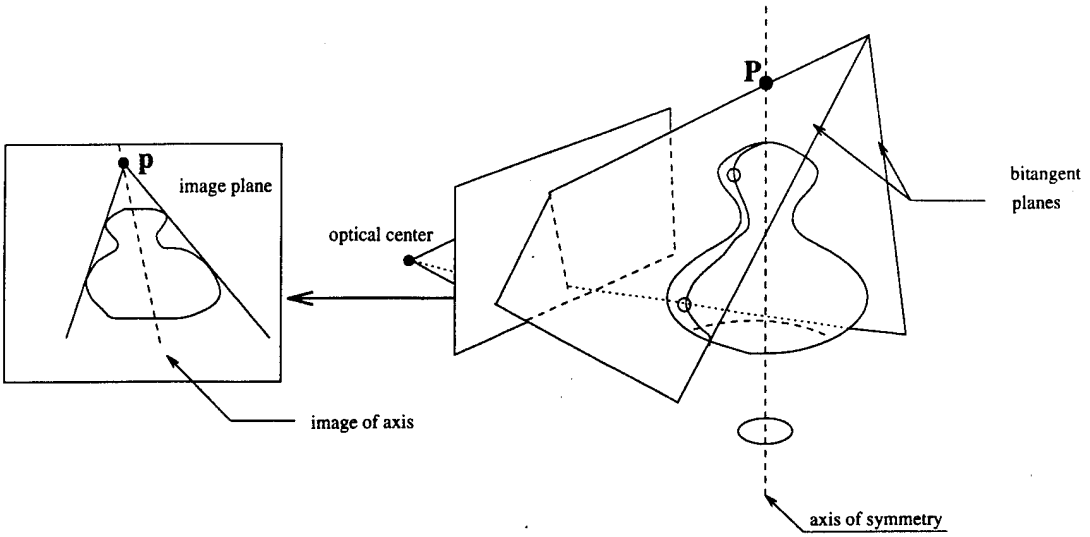


Figure 15: A rotationally symmetric object, and the planes bitangent to the object and passing through the optical centre, are shown. It is clear from the figure that the intersection of these planes is a line, also passing through the optical centre. Each plane appears as a line in the image: the intersection of the planes appears as a point, $\mathbf{p}$, which is the image of the point, $\mathbf{P}$, at which the bitangent planes intersect the axis of symmetry.

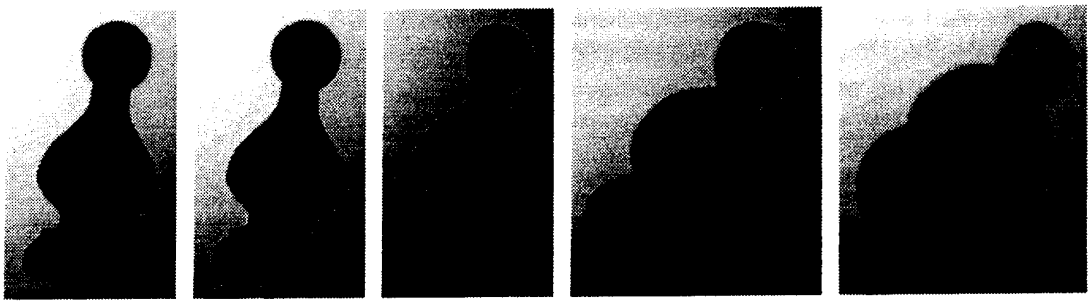


Figure 16: Five perspective images of a surface of revolution at different inclinations. The invariant values are given in table 4.

collinear in space, all lying on the axis of rotational symmetry). In this manner a projective invariant (the cross-ratio) is associated with the surface [20, 35, 45].

The construction extends to straight homogeneous generalised cylinders (SHGCs) [3, 35]. Again, the intersections of corresponding profile bitangents correspond to a viewpoint-invariant 3D point, and are collinear, so cross-ratios can be formed. Figure 16 shows images of a surface of revolution with the calculated invariants given in table 4.

Correspondence and Grouping As described above, the profile of a rotationally symmetric surface can be separated into two "sides", which are related by a planar harmonic homology, $\mathtt{T}$. There are a number of consequences of this result:

1. The two sides of the profile can be grouped by associating curves which are projectively equivalent. For example, by matching projectively equivalent concavity curves. This correspondence can be achieved automatically by the planar recognition system described in section 2.

2. If the projective transformation between two projectively related curves is not a harmonic homology, then the grouped curves can be ruled out as arising from a surface of revolution. This is simply tested by checking if $\mathtt{T}^2 = \mathtt{I}$.

angle	cross-ratio	length ratio
45.0	0.486187	1.40862
40.0	0.490561	1.98153
35.0	0.486796	2.14017
25.0	0.486640	2.38409
15.0	0.486260	2.70539
0.0	0.494849	4.13687

Table 4: Stability of invariants for a surface of revolution. The invariants are computed from measured points. The angle is the inclination of the axis of the lamp-base to the camera plane (figure 16). Typical affine (length ratio) and projective (cross-ratio) invariants are shown. Note that the value of the affine invariant changes at extreme angles, whereas to two significant figures the projective invariant remains stable to the second decimal place.

3. Under real imaging conditions the transformation T relating the two sides of a profile will be close to affine. This quasi-invariant condition can be used in two ways: first, lines joining corresponding points on the two sides of the profile will be almost parallel. Second, relative (not scalar) affine invariants can be used to match concavity curves [43].

4. T provides point to point correspondence between the sides of the profile, this can be used to disambiguate bitangent matches. This correspondence can be used to *repair* missing profile portions, filling in gaps by transforming over points from the other side of the profile.

5. The projected axis can be determined directly from the projectivity as a line of fixed points of the homology [64].

To illustrate the power of this grouping constraint, figure 17 shows an image with many surfaces of revolution of various types and sizes. The matched concavities are partitioned into sets, and the profile curves corresponding to each set are grouped. The entire process is automatic and relies only on the properties of the homology between symmetrical portions of the profile.

3.6 Canal Surfaces

A canal surface is the parallel surface of a space curve. It is the locus of points which are a fixed perpendicular distance from the curve. Equivalently it can be generated as the envelope of a sphere swept with the centre on the curve. Common examples are pipes or tubes such as occur in plumbing. In the following we consider canal surfaces for which the generating curve, α, is *planar*. For such surfaces we have:

> Under general viewing conditions, an inflection in the generating curve gives rise to two inflections in the profile, one on either "side". The tangents on the contour generator at the pre-images of the profile inflections, and the tangent at the generating curve inflection, are parallel.

The consequence of this is that tangents at the paired profile inflections intersect in the vanishing point of the generating curve inflection tangent. This vanishing point lies on the vanishing line of the plane of the canal surface generating curve. This is illustrated in figure 18a. Note that a straight line is simply a degenerate inflection, so invariants can be obtained from a piecewise linear generating curve.

Computing Invariants The canal surface is the envelope of spheres, and the canal profile the envelope of sphere profiles [61]. Under affine imaging conditions, provided the image has the correct aspect ratio (scaled orthographic projection), the sphere profile is a circle, and the sphere centre projects

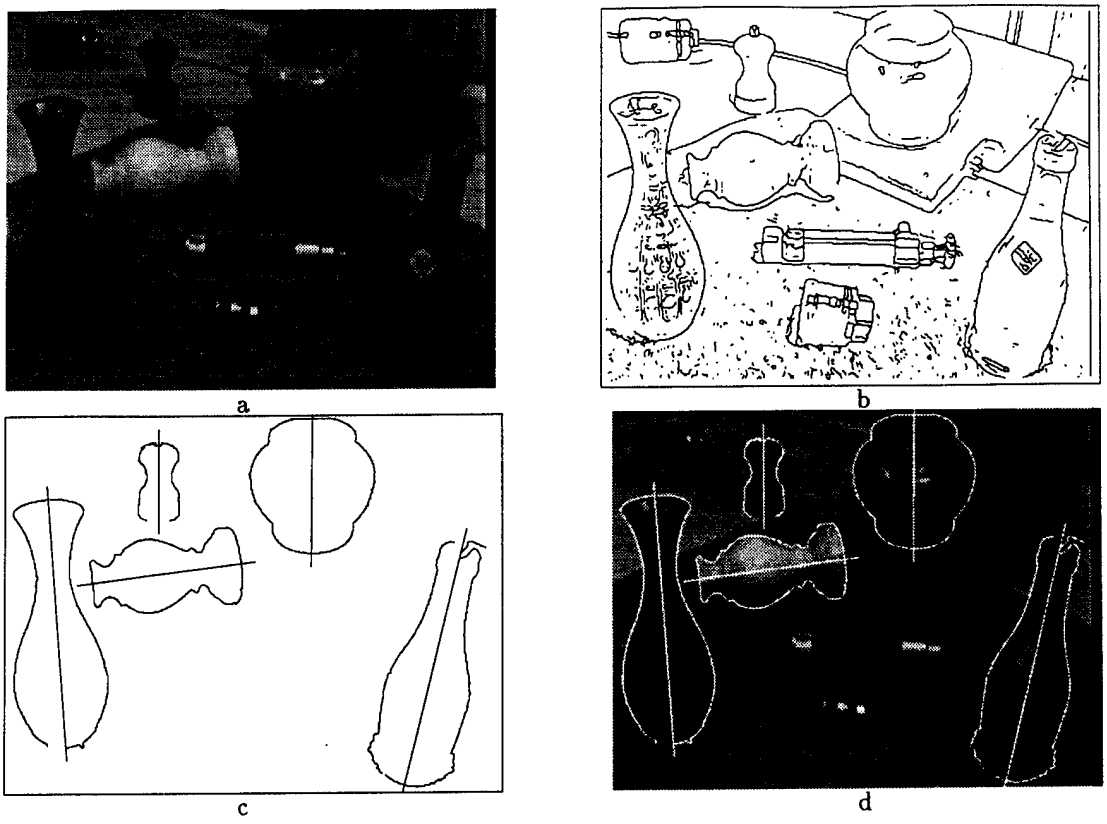


Figure 17: (a) Original image containing several surfaces of revolution. (b) The linked edges computed from (a). (c) Extracted surface of revolution profiles with axes computed automatically using grouping constraints based on a harmonic homology. (d) Extracted surface of revolution profiles and axes superimposed on the original image.

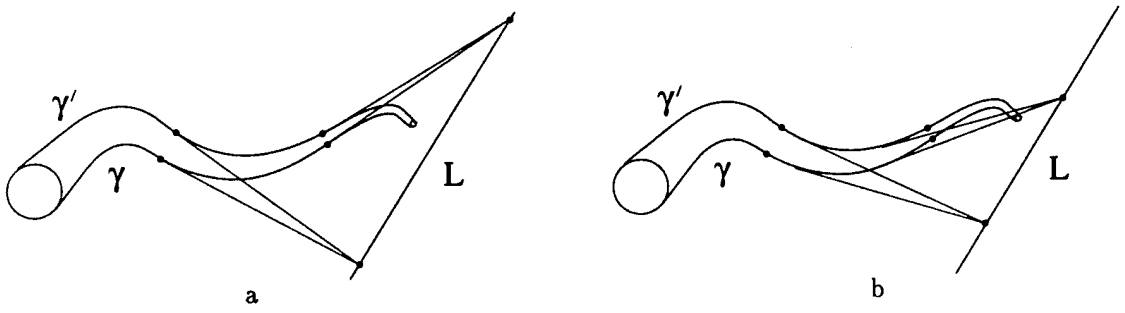


Figure 18: For a canal surface with a planar axis, (a) inflections in the profile occur in pairs for each inflection of the axis. The intersection of a pair of inflection tangents determines the vanishing point of the tangent line at the axis inflection. Two such vanishing points determine the vanishing line, $\mathbf{l}_\infty$, of the plane of the axis; (b) corresponding profile tangents (profile points arising from the same surface circular cross-section) also intersect on $\mathbf{l}_\infty$. Their intersection point is the vanishing point of the corresponding axis tangent line.

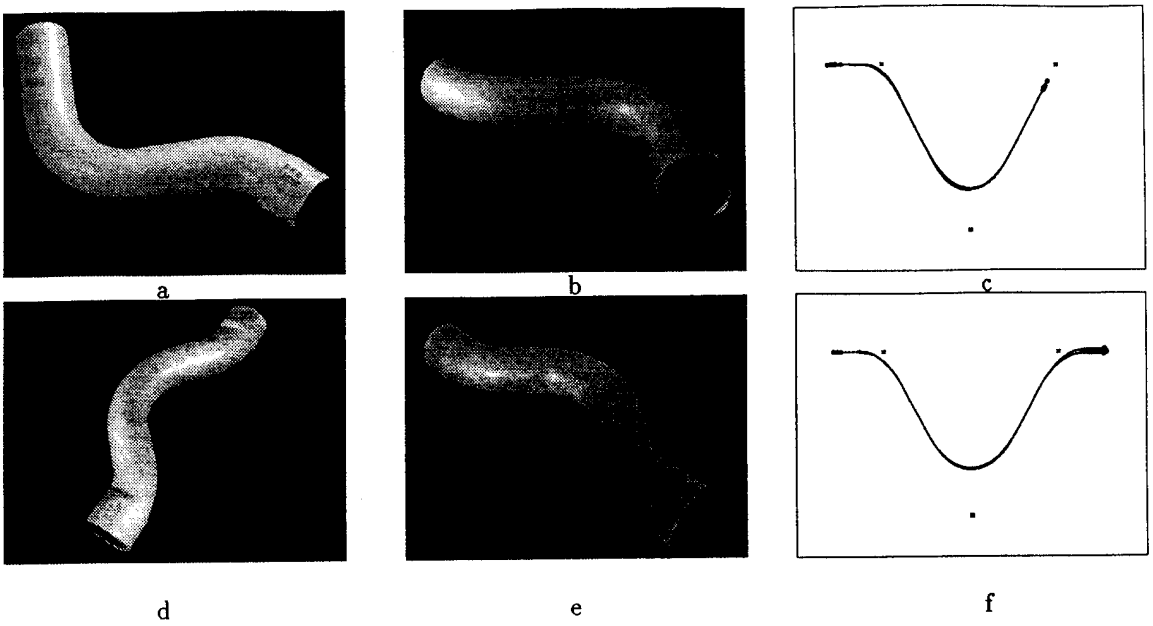


Figure 19: Affine normalisation of canal surface symmetry sets. Each row shows two images of the same pipe, and the symmetry set from these views and others transformed to an affine canonical frame. The canonical frames (c) and (f) contain symmetry sets generated from thirteen and five images respectively. At least half of each set show significant perspective distortions. Note the variation in pipe width in the middle column due to perspective. Affine (as opposed to projective) normalisation can be achieved because the vanishing line of the plane of the generating curve is known (It is computed using the construction of figure 18a). The canonical frame curves are clearly very stable against variation in viewing position. Moreover, different pipes can be distinguished based solely on this affine representation [51]. The slight instability present towards the ends of the canonical frame curves are due to errors in the extracted symmetry set which occur where the pipe radius changes.

to the circle centre. The circle centre can be recovered from the *symmetry set*[5] of the canal profile, which is thus projectively related to the generating curve, α. This relation is exact under affine imaging conditions, and is an extremely good approximation under perspective with a realistic field of view — another example of a quasi-invariant. Consequently, invariants computed for the symmetry set are invariants of the generating curve. For example, invariants can be computed from measurements on the symmetry set curve in a canonical frame in a similar manner to the 'footprints' of Lamdan *et al.* [34]. Figure 19 shows examples of such curves.

Correspondence and Grouping As in the case of bilateral symmetry, the constraint of a canal surface with a planar generating curve establishes a planar projective constraint in the image. In this case two vanishing points determine the vanishing line, $\mathbf{l}_\infty$, of the plane containing the generating curve. Subsequently, inflections on the profile can be paired by the intersections of their tangents on this vanishing line. Furthermore, it can be shown that this intersection constraint holds for *all* corresponding profile points, i.e.,

> Corresponding profile tangents (i.e., points whose pre-image is on the same circular cross-section) intersect on $\mathbf{l}_\infty$.

See figure 18b.

[5]The symmetry set is the locus of centres of circles bitangent to a plane curve. It is studied in detail by Giblin & Brassett [23].

Under affine imaging conditions the two sides of the profile are parallel curves of the symmetry set (the projection of the generating curve). This follows directly from the profile curves being the envelope of constant-radius circles swept along the symmetry set.

3.7 Polyhedra

Recovering the structure of polyhedral objects from a single view has been widely explored, with the most detailed study appearing in [67]. In this work, Sugihara shows that the incidence equations between polyhedral vertices and faces, observed in the image, lead to a linear system of equations in the coefficients of the polyhedron's faces and image observations.

The equations in this system are incidence equations for vertices of the polyhedron incident on plane faces. In particular, given vertex $\mathbf{V}_i = (X_i, Y_i, Z_i)$ lying on face $\mathbf{F}_j = (A_j, B_j, C_j, 1)$, it must be the case that

$$A_j X_i + B_j Y_i + C_j Z_i + 1 = 0.$$

Assume that the camera image plane is the plane $Z = 1$ and the focal point is at $(0, 0, 0)$; these assumptions can be accounted for by the geometric ambiguity in the reconstruction. Then $\mathbf{V}_i$ projects to image point $(u_i, v_i) = (X_i/Z_i, Y_i/Z_i)$. If vertex $\mathbf{V}_i$ also lies on face $\mathbf{F}_k$, we can divide the incidence equations by Z_i and subtract to eliminate $1/Z_i$, obtaining:

$$(A_j - A_k)u_i + (B_j - B_k)v_i + (C_j - C_k) = 0$$

where u_i and v_i are known, and the coefficients of the planes are unknowns. This system of equations always has at least a three dimensional family of solutions, corresponding to a polyhedron where all faces are the same plane (since a plane figure has three degrees of freedom in 3D space). If the family of solutions is four dimensional, then a generic element of the family is a system of planes that is projectively equivalent to the faces of the original polyhedron. This case holds when the reconstruction of the polyhedron can not be made impossible by a small shift of the vertices, that is, is "position free" in the terminology of Sugihara, and many or most of the visible faces have at least four vertices per face. Although this is by no means a generic polyhedron, it is a useful case because many human artifacts satisfy these constraints. Given the added assumption that vertices are trihedral, it is possible to reconstruct faces for which only two edges are visible; thus, on viewing a cube, all six faces can be recovered. This leads to a novel formulation of the aspect graph idea, where substantively fewer aspects are necessary for effective representation. The case where the polyhedron consists only of triangular faces is equivalent to an unconstrained set of points. That is, a set of points can always be triangulated to form a polyhedron. As in the case of a general point set (section 3), vertex positions are unconstrained by the image view and no invariants can be constructed.

Computing Invariants Assuming an uncalibrated camera, it can be shown [59, 60] that for polyhedra that lead to a system of equations having a four dimensional solution space (such as cubes) any solution of this system is projectively equivalent to the original (Euclidean) polyhedron. Consequently, projective invariants of the solution are the same as those measured on the original polyhedron.

Correspondence and Grouping Approaches to grouping and correspondence for this class are well established from the decade or so of blocks world vision research. The main basis for grouping is topological, where one seeks to construct a complete polyhedral structure with consistent incident relations between vertices, edges and faces.

For particular sub-classes of polyhedra, and for particular aspects, further constraints are available. For example, a cube has three major directions which define a triple of vanishing points in the image. All edges aligned with a major direction must pass through the same vanishing point. Similar incidence constraints apply to any polyhedra projectively equivalent to a cube. Constraints of this type can be used to extract a polyhedral wireframe from a polyhedral silhouette, inferring internal boundaries.

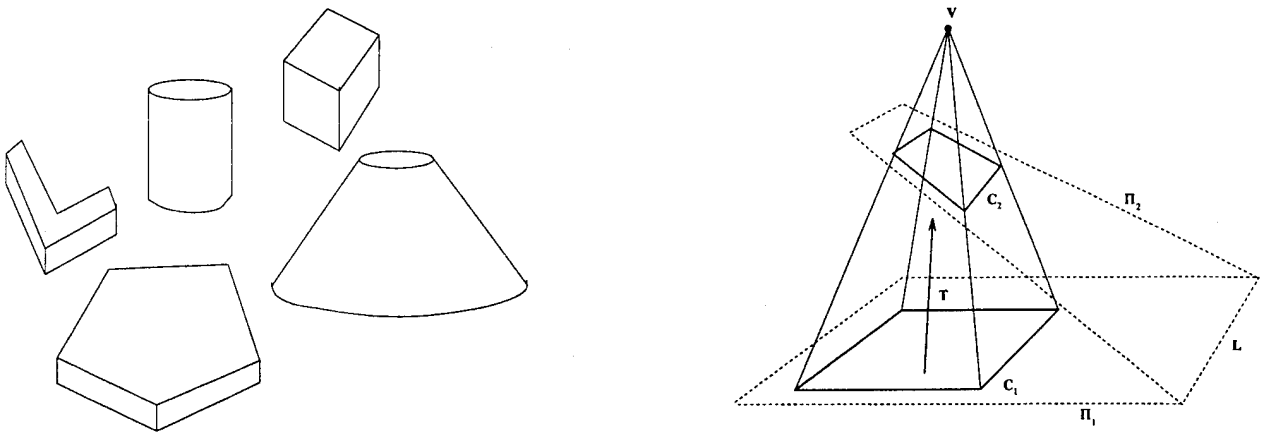


Figure 20: An extruded surface is a section cut from a general cone by two planes. (a) A range of examples of extruded surfaces; note that for most examples, the vertex is at infinity. (b) The top and base image curves, C_1 and C_2, of an extruded surface are related by a particular projective transformation T, called a planar homology. Corresponding points lie on lines through **V**, which is the fixed point of the transformation (the centre or vertex). The line **L**, which is the image of the intersection of the two planes that cut the "cone", is a line of fixed points of the transformation (the axis).

3.8 Extruded Surfaces

An extruded surface is a special case of a generalised cylinder, formed by a section cut from a general cone by two planes (see figure 20a) in such a way that the section of surface does not include the vertex of the cone[21]. This is the projective generalisation of a surface formed by a system of parallel lines, with plane ends (such a surface can be extruded from a nozzle). Extruded surfaces, and surfaces made up from extruded components, are extremely common — examples include most tin cans, boxes, books, and many plastic bottles.

Outline Geometry The base and top curve are perspectively related in 3D, and thus related in the image by a projective transformation, T. This transformation is a *planar homology* [64]. It has five degrees of freedom: the vertex (2 dof), axis (2 dof) and the cross ratio defined by the vertex, a pair of corresponding points, and the intersection of the line joining these points with the axis (1 dof). The cross-ratio is the same for all points related by the homology[6]. As in the case of a planar harmonic homology (figure 14) a planar homology has a line of fixed points and a fixed point not on this line:

1. The homology vertex is the projection of the 3D cone vertex. It is the fixed point of T.
2. The homology axis is the projection of the line of intersection of the top and base planes. It is the line of fixed points of T.

The profile curves of an extruded surface are a pair of lines which intersect at the image vertex.

Computing Invariants The projective geometry of an extruded surface is completely defined by three elements: a plane cross-section; a cone vertex, not on the plane; and, a line in the plane. The plane cross-section and vertex together define the cone. The line is the axis of the pencil of planes which intersect the cone to generate the top and base curves. These elements can be recovered from an image of the surface since the cross-section of the cone is determined up to a projective transformation from the imaged base curve or top curve, and the line is the line of fixed points of the projective transformation relating top and base image curves.

[6] In the case of a harmonic homology, the cross-ratio is harmonic, i.e., known, so there are only four remaining degrees of freedom. The sides of the profile of a surface of revolution are related by a harmonic homology, as described in section 3.5.

In essence, the invariants of an extruded surface are those of the plane cross-section plus an extra line in the plane, obtained from intersection of the base and top planes. Thus extra invariants are available over the plane cross-section alone. For example, in figure 20 a five line invariant can be computed from the image, although the top curve only contains four lines. In the case that the top and base planes are parallel, affine invariants of the curve can be measured from a perspective image.

Correspondence and Grouping As described above, for an extruded surface the top and base image curves are related by a planar homology, T. Grouping proceeds by finding curves which are projectively related. The class assumption can then be tested immediately since the projective transformation must be a homology if the curves are from an extruded surface (for example, two of the eigenvalues will be equal). The homology then defines the vertex, axis, and correspondence for the surface, which is used for further grouping. Additionally, since the surface is ruled, and all rulings pass through the vertex, the intersection of line segments in the profile determines the imaged vertex. Similarly, all corresponding point pairs on C and C' (figure 20b) define a pencil of lines which pass through the vertex, and corresponding tangents intersect on the line of fixed points, $\mathbf{l}$.

3.9 Algebraic Surfaces

Algebraic surfaces are surfaces for which a single polynomial vanishes: examples include spheres ($x^2 + y^2 + z^2 - 1 = 0$) and ellipsoids which are both degree two surfaces (quadrics), and a wide range of popular surfaces in modelling such as rational bicubic patches. Smooth quadrics are all projectively equivalent (just as all conics are projectively equivalent) so that there are no projective invariants of the surface to recover from images. Although a single quadric does not have any projective invariants, two or more quadrics do. Similarly, if the surface has degree 3 or greater, there are projective invariants to recover from images. In theory these invariants can be recovered from the surface profile alone [22], though this has not been implemented in practice.

4 An architecture for a 3D recognition system

We have demonstrated that a large vocabulary of 3D invariants can be derived from the geometric constraints associated with object class definitions, e.g., that of a surface of revolution. In general, these curve, surface or volume class constraints enable the construction of invariants, and permit at least partial reconstruction of the 3D structure from a single perspective view. Class constraints also provide image feature grouping mechanisms and associated indexing machinery.

The work to date, however, has focused on the derivation of invariants, structure recovery, and grouping for single object classes. Experimental validation has been restricted to isolated objects of a given class against an uncluttered background. An important next step is to integrate the approaches which have been developed into a unified 3D object recognition system. It is only in the context of a full system that the effectiveness of a class-based invariant representation for recognition can be convincingly demonstrated.

4.1 Fundamental Principles

Object recognition should be based on 3D geometric descriptions, both of objects and of the relationships between objects. To date, systems have largely ignored these relationships; as we show below, requiring *consistency* in inter-object relationships yields substantial information. In the architecture we describe, this information is encapsulated in an internal database, known as the *scene*. The scene provides a working reconstruction against which hypotheses can be checked to provide immediate detection of a false recognition hypothesis. For example, if two objects are hypothesised in such a way that one must be wholly occluded by the other, then at least one of the object hypotheses must be wrong.

Central to the architecture is efficient management of *control* of each level. Even for relatively small images, vast numbers of hypotheses for feature correspondences and model interpretations can be constructed. It is impossible to explore all avenues of interpretation, so some basis must be established for

scheduling feature combination, hypothesis generation and verification of hypotheses. The priority of scheduling should be based on a tradeoff between the cost and the benefit of a computation.

Finally, *class* pervades the architecture, influencing segmentation, grouping, indexing, and hypothesis confirmation.

4.1.1 Class

The idea that objects should be organised in a taxonomy and classified before proceeding to recognition is a natural and well-accepted principle. The problem with this philosophy is that many ontological distinctions are not manifested in observable properties, for example, the difference between a hollow container and a solid block. Our geometric approach to object classification is based directly on visible features; its main strength is that it is not vested in abstract, philosophical differences, so much as in image observable distinctions. Object class has its most important effects in considering feature grouping and the structure of the modelbase.

Class drives grouping, as opposed to the usual "heuristics" that are used to associate image features. Each object class defines a grouping mechanism based on its image invariant relation. For this reason, object class is typically settled at an early stage in the grouping process, and identity emerges only after modelbase access. For example, there is no point in grouping lines into faces, as required for polyhedral class grouping, to recognise a rotationally symmetric object. A rotational symmetry hypothesis requires image curves related by a planar harmonic homology. The projective matching of these curves can be carried out by computing and matching projective invariants of the curves. This is an application of planar object recognition techniques within a single image.

Class determines the access functions and partitioning of the modelbase. The modelbase itself is a collection of facts about objects and their properties. These facts must be organised in such a way as to allow easy retrieval; a hashing mechanism is appropriate. By the time the modelbase is accessed, the object group will contain a strong implicit hypothesis about object class — for example, a pair of concavity-curves cannot be passed to the polyhedral hashing mechanism. In fact, the modelbase can be viewed as a rather conventional database, organised to answer certain queries very efficiently.

4.1.2 Consistency

Consistency tests arise from computing and representing relationships between objects. To date, there have been few "hard" geometric consistency tests for inter-object relations. In fact, strong geometric tests emerge from the observation that objects share the same Euclidean frame and the same camera. These tests make it possible to recover the Euclidean identity of objects even if the calibration of the camera is initially unknown.

Suppose that models are Euclidean (i.e., the relation between the model and object is an Euclidean transformation, as opposed to the projective transformation of the 2D recognition system), and recognition hypotheses have been formed for a number of objects. Even though the camera is uncalibrated, the Euclidean consistency of the recognition hypotheses can be tested by a comparison of the set of ray cones from the optical centre to each object.

The cones are determined from $\mathtt{P}$, the rank three, 3×4 projection matrix of equation (2). Given a hypothesised object, $\mathtt{P}$ is determined from the known 3D Euclidean geometry and the image features by standard resectioning (as in, for example [56]). Partitioning $\mathtt{P}$ as $\mathtt{P} = [\mathtt{M}|-\mathtt{Mt}]$ [29], then $\mathtt{t}$ is the optical centre which is the null space of $\mathtt{P}$. A cone of rays from the optical centre to other Euclidean objects in the scene can then be constructed.

If the hypotheses for each object are correct, the ray cones for each object should be Euclidean equivalent. That is, there will be a rotation about the optical centre which superimposes the cones for each object from each hypothesis. Thus, inconsistent hypotheses can be detected by the failure of this test and the goal is to build up the largest pairwise consistent set of object hypotheses. An example of hypothesis labelling and reconstruction is shown in figure 21.

Another consistency test involves decomposing the matrix $\mathtt{M}$ (above) as $\mathtt{M} = \mathtt{KR}$ by QR decomposition [25], where $\mathtt{R}$ is a rotation matrix, and $\mathtt{K}$ an upper triangular matrix containing the intrinsic parameters of the camera. Each hypothesis must agree on the camera intrinsic parameters and inconsistent hypotheses can be detected from differing decompositions for $\mathtt{K}$.

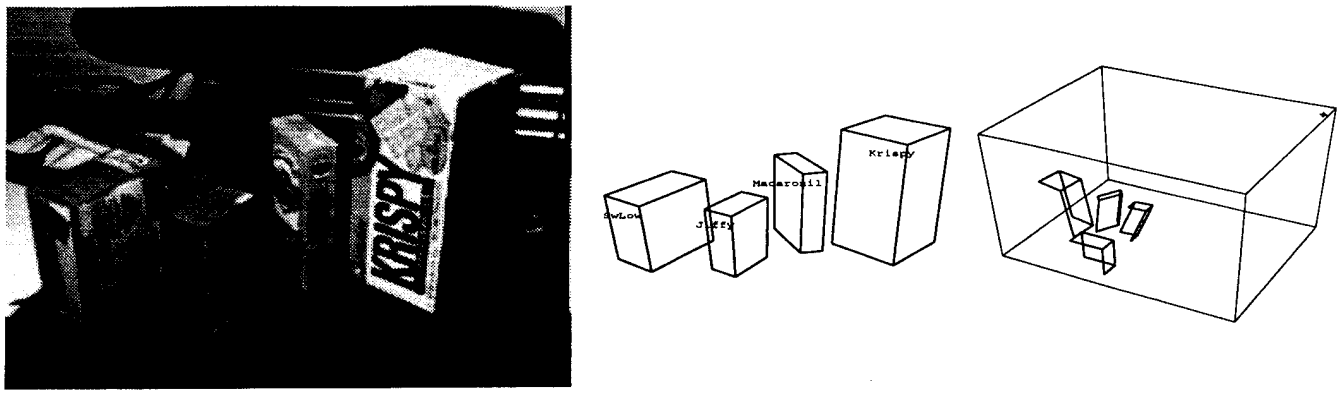


Figure 21: (a) A scene containing polyhedra all of which are projectively equivalent, but Euclidean inequivalent. (b) A labelling of the scene. (c) A Euclidean reconstruction of the polyhedral world shown in (a). The optical centre is the marked point in the top right-hand corner of the figure.

Given an image containing a large number of known objects, once the first few have been recognised and used to construct a consistent world frame, this frame can be accepted and used to prune additional hypotheses in a depth-first search for consistency. At this point, rather than searching for consistent groups of object hypotheses, individual hypotheses can be tested against the established frame with little risk of error. Furthermore, if this frame is accepted, then it can be used to condition grouping and indexing activities. The Euclidean reconstruction of the world forms the scene database.

4.2 The Architecture

Representation is organised into a number of layers as illustrated in figure 22. These stages of representation are not very different from other recognition architectures, however the three principles of *class*, *global consistency* and *control* provide a unifying theme.

Segmentation and grouping The key to successful recognition is efficient and robust feature segmentation and grouping. There are four levels of image feature representation and grouping:

Level I: **Pixel-level features** are defined with respect to an image coordinate system and reflect the quantised nature of pixel coordinates. Typically, features will be produced using an edge operator with sub-pixel accuracy, and the resulting edgels linked into a network reflecting the topology of the image boundaries.

Level II: **Geometric features** curves from level I are described in terms of geometric primitives, where appropriate. For example, algebraic curves such as line segments and conics, smooth curves, and concavities defined by bitangents.

Level III: **Generic Grouped features** This level of grouping is applied to all features produced at level II. The output is a number of groupings and databases which are used by the class-based groupers described below. Generic grouping includes: near incidence (jumping small gaps, completing corners and junctions); collinearity; marking bitangent and other distinguished points; finding sets of parallel line segments; affine or projective equivalence of curve segments (e.g., concavities). These relations can be viewed as queries to a spatially organised data base. For example, typical queries might be: "what other lines are parallel to a given line and above a certain length in the region of interest?", or "what other lines are collinear with the given line over the entire image?". In the current design there is no attempt at enforcing "backwards compatibility". For example, if a grouper at level III hypothesises that two curves should be joined there is no attempt to correct the level II representation. Ultimately, it may be important to ensure such consistency between levels.

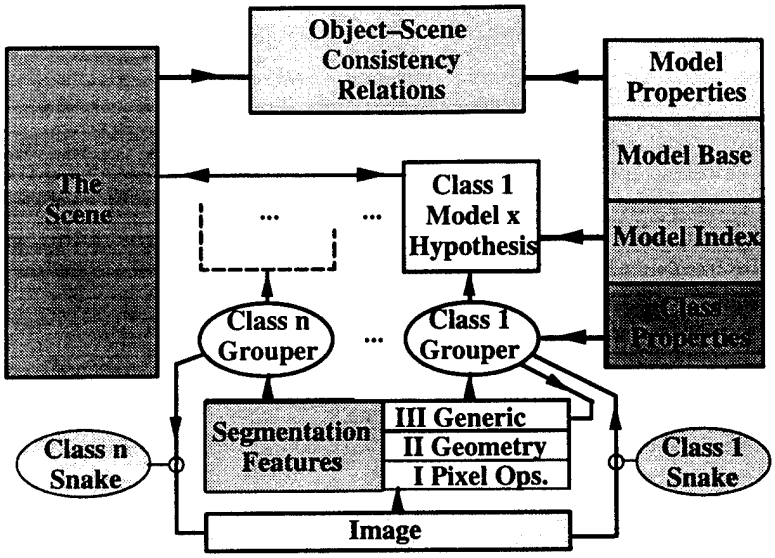


Figure 22: The proposed architecture for object recognition organised around geometric *classes* with associated grouping and indexing methods.

Level IV: **Class-based grouping** Each class has an associated "class-based grouper" that interrogates the level III groupings and databases, and attempts to form groups appropriate for its class. The grouping mechanism is based on the image invariant-relation as described in section 3 for each class. A good example is given by the rotationally symmetric class which defines a grouping constraint in terms of the harmonic homology between the corresponding sides of the profile (section 3.5).

In addition to grouping, such constraints can be used to *repair* missing portions of the outline due to occlusion or poor contrast. For example, for a surface of revolution, a "snake" or deformable template can be defined by one side and applied to the other under the transformation of the homology. The transformation between both sides can be iterated to improve the geometric correspondence of both sides. Such class-based snakes can also augment the initial edgel extraction process. Figure 23 shows an example of repair and augmentation, where a polyhedral class snake recovers poorly defined interior edges from the exterior polyhedra outline, again based on the class constraints.

Indexing and Hypothesis Combination The groups defined by each class also define the indexing function used to retrieve specific objects from the modelbase. For example, for a canal surface the indexes are computed from the symmetry set of the profile, for a surface of revolution from distinguished points on the axis.

Indexing is handled by a series of hash-tables, one per class, that take the invariants of a set of grouped features and associate with them models in the modelbase. For complex objects, there may be many feature groups that index to the object, leading to a situation where a single instance could generate many recognition hypotheses. In the planar recognition system, this problem is handled by merging consistent object hypotheses into joint hypotheses.

Forming joint hypotheses (cliques), is fairly successful for small numbers of feature groups, but for more complex objects, there are potentially quite substantial combinatorial problems. However, the

principle that feature groups belonging to the same object should accrete into a more complex feature grouping, is a good one. This accretion can be implemented in a more general fashion as follows: if a feature group results in a successful indexing attempt to a relatively small number of models, it leaves a record of that attempt in an image-scene relational data structure. When another feature group indexes to the same object or list of objects, and is within some grouping horizon of the first group, the two feature groups can be associated in a larger feature grouping, based on their correspondence to the same object structure. To make this record, the system forms a collection of keys out of the image feature position and each possible object model in turn, and stores a unique identifier for the image feature group in an image-scene database using these keys. The storage mechanism is such that, if the database is queried using a model identity and feature position, it will return any image features that indexed that model and are "near" (for some horizon) the original feature group. Note that other forms of image-scene information could be used in addition to Euclidean distance in the image; for example, an indexing hypothesis might be associated with a pose or frame hypothesis.

Verification In the planar system, two stages of verification were used: plausibility of the projective transformation taking the object from the model to image frame, and image support measured by the proportion of the back-projected model perimeter that lines up with image features. Such verification, based only on object outline, can fail through accidental correspondences with texture (for example, oriented markings such as wood grain).

To avoid this problem, verification is augmented in a number of ways. First, surface markings and surface texture will be stored for each object in the model library. During verification, the internal surface properties of an object can be compared with the properties actually observed in the interior of a model hypothesis. Second, the reliability of verification will be improved by scene consistency analysis. For example, if one object is deemed to be behind another with respect to a given camera viewpoint, then it would be inconsistent to declare a large portion of confirmed boundary for the occluded object. More generically, the "score" for a hypothesis is improved if, when portions of the perimeter cannot be matched, there is independent evidence of an occlusion occurring. For example, one piece of evidence for occlusion is that aligned "T" junctions occur at each end of the occlusion.

The modelbase The modelbase will be organised around object class. For each class there will be appropriate hash-tables for indexing, and a data base of models. For example, canal surfaces and surfaces of revolution will have separate indexing tables, containing respectively affine and projective invariants, and separate model libraries.

In the 2D recognition system (section 2.3) the modelbase typically acted as a passive repository that contained geometric models, and was indexed to identify an object. The modelbase can be more powerful than this. It can also store aggregated statistics derived from all the models in each class library. These can be used to improve efficiency. For example, suppose the maximum number of undulations of any surface of revolution in the library is stored. Then if a putative profile is returned by the grouper which has more undulations than this, there is no point computing invariants or indexing. Similarly, if there are only trihedral vertices for any polyhedra, then there is no need for the polyhedral grouper to attempt to group or index with four concurrent lines. Exploiting the modelbase in this manner can greatly strengthen the performance of the system.

Objects which do not correspond to a single volumetric primitive, i.e., *composite* [72] objects, will have multiple representations: each representation covering a possible image segmentation and grouping. For example, a mug might be represented, in the composite 3D structure class, as a surface of revolution together with a canal surface (the handle). Equally, the handle could be represented as a digital plane curve, and the mug body as a canal surface or extruded surface. All such representations will be included in the modelbase. It is only through recognition that the common concept "mug" is achieved.

The scene An additional source of constraints and parameters is the 3D scene, which can also be viewed as a database which reflects the current configuration of the world and cameras. It provides a representation of all the information currently available about the common Euclidean frame in which objects reside. This 3D spatial layout can be used at a number of stages, for example, to determine occlusion relations amongst model hypotheses, and for camera viewpoint consistency.

4.3 Model acquisition

Typically, models will be acquired from multiple views of objects. The fact that such models can serve as sufficient representations for recognition is a major advantage of the invariance approach to recognition. Our goal is to provide a model acquisition tool which permits additions to the model library to be as simple as providing four or five unoccluded views of the object. This goal was achieved in the planar object recognition system, where only one or two unoccluded views were required to construct a model.

Of course, the problem is more difficult for 3D objects, but the partitioning into classes, based on geometry, will permit the efficient grouping and correspondence construction required for model description. Since the object classes consist of 3D volumetric primitives, we expect that only a small number of views will be required for most objects, and that these views will be defined by the extraction of a sufficient set of stable features over a wide range of viewpoints. This contrasts with the construction of aspect graphs based on topology [65], which define a large number of aspects, or distinct views, which cannot be reliably distinguished from image feature groups. That is, fine topological properties of an image feature group are unlikely to be reliably recovered from image segmentation and grouping. The integrity of the object boundary topology is a secondary result achieved after an initial recognition hypothesis has occurred.

Euclidean information in object models is required for scene consistency techniques to work. One approach is to derive the Euclidean properties from self-calibrated camera views. Three or more general views with a single camera are sufficient to derive internal camera parameters [17, 30], and a 3D scaled Euclidean reconstruction for point sets.

For manufactured objects, Computer Aided Design (CAD) models can be used to provide a Euclidean description. However, it should be noted that it is often the case that CAD models used for part design do not necessarily correspond exactly to the manufactured version of the part. The description obtained from imagery is more "realistic" and incorporates many details which are not practical to include in a CAD model, such as filets and attachment hardware. Conversely, certain CAD features may be irrelevant in practice since they are not manifested visually in any image.

On the other hand, it is important to develop the idea of Platonic generalisation of a model description. For example, if an image curve is sufficiently straight, it can be interpreted as an instance of an "ideal" line even though its manifestation as an image feature is never perfectly straight. Similarly, a pair of profile curves may match closely enough to be considered the outline of a rotationally symmetric object, even though they are not perfect instances of such a projection. The benefit of constructing a Platonic ideal description is that over a large set of views and feature reconstructions, the ideal description represents the natural *mean* over the set of reconstructions. Also, the Platonic description is in accordance with the formal mathematical constraints used to drive the grouping and indexing process.

4.4 MORSE

These ideas are being incorporated into a system, called MORSE, whose implementation is currently underway. MORSE is named after the detective character originated by Colin Dexter, who is able to ferret out truth given apparently unpromising evidence. Our earlier system for 2D object recognition is called LEWIS, the name of Morse's less capable assistant. MORSE will provide an environment for research on object representation for recognition, by providing a context in which issues such as the distinctiveness of representations, the usefulness of feature groups, and the significance of consistency, can be addressed. The system is being implemented in C++ using a class hierarchy based on the Image Understanding Environment (IUE)[7].

The current state of progress of MORSE is illustrated in figure 23. Class-based groupers have been implemented for surfaces of revolution (section 3.5), canal surfaces (section 3.6), and polyhedra (section 3.7). In each case the grouping is based solely on the constraints on the structure of the profiles for each class. Profiles for each class are extracted completely automatically. As is demonstrated in the figure, recognition proceeds by first recognising an object as belonging to one of the classes (for example a surface of revolution). Subsequently the object will be identified (for example as a particular vase).

[7] The IUE is an ARPA funded project to produce an object-oriented programming environment for vision research. A central object hierarchy in the IUE is the *spatial-object* which incorporates many of the descriptive requirements described in the previous sections. The IUE also has an extensive set of classes for object and image transformations which are a central issue in MORSE.

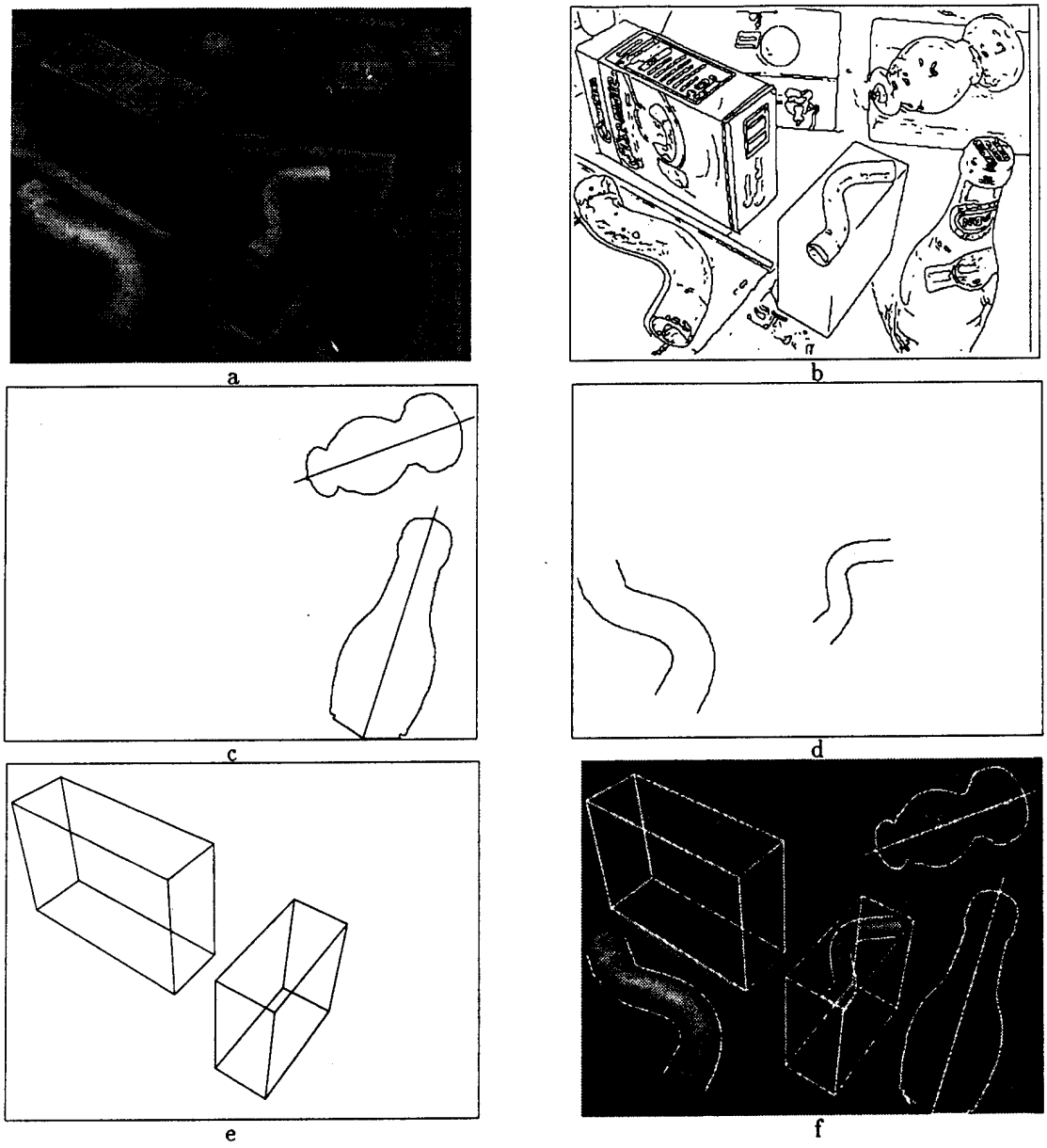


Figure 23: (a) Original image containing two surfaces of revolution, two canal surfaces, and two polyhedra. (b) The linked edges computed from (a). Profiles are extracted and grouped automatically from these linked edges by the class-based groupers. (c) Extracted surface of revolution profiles with axes. Note that gaps in the edgel chains have been repaired in the recovered profile. (d) Extracted canal profiles. (e) Extracted polyhedra outlines. (f) Extracted profiles superimposed on original image. All the correct instances of a class have been grouped, and no false instances grouped.

5 Discussion

5.1 Critique of the invariance approach

To conclude, it is useful to clarify many of the points just presented by responding to a number of major criticisms which can be made of the invariance approach to recognition. It will be instructive to employ these critical points as a benchmark of the progress in recognition which can be attributed to the invariant framework.

1. **The extreme nature of projective ambiguity:** Invariance concentrates on projective representations. In practice, perspective distortions in images are small and so can be ignored. Furthermore, the projective equivalence class is too large — a sphere and an ellipsoid are in the same class, as are a cube and a truncated pyramid. Thus, the recognition system can not distinguish between them.

   Response: First, we have demonstrated with the planar recognition system that a projective representation *is* sufficient for many practical examples. Second, although it is almost always the case that only projective structure can be recovered in a single uncalibrated image of an object, this does not mean that the recognition system is bound to projective ambiguities. For example, for certain classes, affine or similarity invariants can be measured in a perspective image, e.g., a structure repeated by translation (section 3.4). Euclidean consistency, section 4.1.2, can be used to reduce ambiguity from projective to similarity for an image of multiple objects.

2. **The exclusiveness of geometry:** Invariance at present concentrates on geometry to the exclusion of other important object properties that should be used in a recognition system such as: colour, texture (e.g., wood grain), surface markings (e.g., pictures or lettering on a can), and surface properties (e.g., metal vs dielectric).

   Response: Geometry very largely dominates object descriptions in the system sketched above. There is some way to go before colour, texture, surface markings or surface properties can stand on an equal footing with geometric information. These are, at present, measured in images relatively unreliably compared to geometry. Nevertheless, such cues fit into the proposed architecture. For example surface markings and texture can be used as additional invariant indexes (see section 5.2), and could certainly be used as additional measures during verification.

3. **The lack of abstract classification:** Invariance does not address the problem of classification, only identification. In a typical model-based system, the class to which an object belongs can be determined only by recognising it as a specific object. For example, an unknown object might be identified as a "1991 Red Mazda 323 Hatchback", which is a member of the class "car". This class membership is determined by subsequent reasoning: it cannot be directly identified as a car, as distinct from a fish, despite the differences between the two classes.

   Response: Abstract classification in its broadest sense presents severe conceptual and philosophical problems, which we have carefully avoided addressing. Until it is possible to address these problems concretely, by, for example, stating exactly what a program that distinguished between a general fish and a general bicycle would do, they will be difficult to solve. However, the architecture proposed contains a first step in this direction, by distinguishing between classes of object on the basis of the techniques required to construct representations from images. In particular, if a group of edge segments is classified as, for example, the profile of a rotationally symmetric object, techniques exist for confirming that classification (in this case, by determining that there is a projective equivalence, $\mathtt{T}$, on the profile, such that $\mathtt{T}^2 = \mathtt{I}$).

4. **The rigidity of exact geometry:** Geometry is not the appropriate language to represent objects such as clothes, plants and animals, which can articulate and deform. A deformable template, or even non-geometric descriptions, such as a set of colour histograms, may be much more effective in representing such objects.

Response: It is not yet clear what representations one would want to extract for objects that have no clearly defined geometry (for example, what aspects distinguish one shirt from another?). As a result, exact geometry is probably going to dominate recognition for some time to come. However, there is clearly a need for a hybrid of geometric invariance and statistics for certain classes of deformation; invariance would allow for change of viewpoint, while statistics would cover the deformation.

5. **Complex objects:** Invariants might be suitable for representing and recognising "simple" parts, such as surfaces of revolution or quadrics ("geons") but do not yet cover assembling these shapes into complex objects, such as telephones or aeroplanes.

   Response: Some of the 3D classes, for example, surfaces of revolution or canal surfaces, are "simple" — essentially, little more than plane curves. This does not affect their usefulness in representing a large number of real objects. Other classes of objects are more genuinely 3D objects — for example, repeated structures or polyhedra.

   Objects consisting of, for example, a hierarchy of parts, are not explicitly addressed in this approach. However, the seeds of a solution are present in the use of geometric relations between feature groups (intra-object invariants), such as those shown in figure 3, in forming joint-recognition hypotheses. One could advance the system architecture above to do the same thing, recognising parts individually and then using projective, or Euclidean information about object pose that results from the consistency checking, to determine whether components lie in such a way as to make up a composite object.

   Concerning segmentation, it may be the case that the grouping relations defined for each class provide a natural means of segmenting a complex outline into primitive volumetric parts. For example, when the harmonic homology on the profile ceases to apply this would indicate that the surface of revolution part had finished.

   Of course, there will be objects, e.g., potatoes, that can not be represented by a combination of the classes described here. However, such generic shapes are currently difficult to distinguish with any representation under the distortions of perspective imaging. Our view is that it is better to proceed with a set of classes which can support reliable recognition and establish a benchmark of performance for future systems to build on.

5.2 Avenues of future research

Indexing allows fast recognition of objects drawn from a diverse collection of classes: a range of specific techniques for recovering the projective invariants necessary for indexing various object classes has been displayed and demonstrated. These ideas have been integrated to produce a recognition system architecture that should be capable of handling large, diverse modelbases, and that addresses many of the concerns recently raised about indexing in recognition systems.

However, object recognition is not yet "solved." There are a range of avenues of research that promise exciting developments; we indicate a few topics most interesting to us:

- **The role of quasi-invariants:** Using quasi-invariants for indexing is a problem, because of the cost incurred if the "wrong" object is indexed by a quasi-invariant applied outside its domain of stability. Avoiding this requires complex hypothesis combination to ensure the "right" object is indexed. The benefit of quasi-invariants is the use of simpler feature groups at the start of the recognition process. However, simple feature groups are often not very discriminating. Instead, we propose that quasi-invariants should be used to *schedule* grouping. The quasi-invariants identified in this paper (e.g., the relation between sides of a surface of revolution profile), can be used to schedule promising groups for further growth.

- **Learning invariants:** invariant indexes are a good goal for a learning algorithm; an ideal algorithm would, given a large model base, determine by some offline process of generating views, the functions and image features most useful in indexing models effectively. Alternatively, invariants could be

extracted from a large number of real images of an object taken from varying viewpoints. The advantage of the latter is that the invariant descriptors would only involve features that could be reliably measured in images.

- **The use of texture and surface markings:** Clearly, texture and surface markings have a part to play in verification. However, surface markings, together with the profile of certain surface classes, can be used to generate further projective invariants. For example, by facilitating the backprojection of the markings onto the surface for that class. These marking invariants can augment indexes based on the object profile alone. For example, without surface markings, quadrics are projectively equivalent, but four points (markings) on a quadric surface have two projective invariants in space, which can be recovered from a single image.

- **Extensions to Grouping Computation:** In recent experiments with control for grouping features for rotationally symmetric objects and repeated structures, the idea of *synchrony* in edgel curve and line segment linking has emerged. For example in exploring the topological links along the occluding boundary of a rotationally symmetric object, it should be possible to use the constraints of the planar homology to control the linking sequence. In a complex scene with many possible edgel chain connections, these constraints will considerably reduce the number of feasible paths generated for symmetrical association. Once a single concavity is determined, the rest of the boundary can be recovered by a synchronised edge following algorithm. As new parts of the boundary are confirmed, the homology transform parameters can be iteratively refined.

  The same type of strategy can be followed for any geometric class based on symmetry or structural repetition. The constraints inherent in these classes can be extended right down to the edgel linking stage. Such an approach is currently being implemented.

Acknowledgements

We are grateful for discussions with Santanu Chaudhury, Peter Giblin, Richard Hartley, Jitendra Malik, Yael Moses, Sven Utcke and Luc Van Gool. Ellen Walker of RPI provided support and guidance for Jane Liu. Martin Armstrong and Paul Beardsley computed the affine invariants for the translational repeated structure. Financial support was provided by several agencies: ESPRIT Project 6448 'VIVA'; a NSF Young Investigator Award with matching funds from GE, Tektronix, Rockwell International and Eugene Rikel; NSF grant no. IRI-9209729; US Air Force Office of Scientific Research grant no. AFOSR-91-0361; General Electric; and The Newton Institute, Cambridge, under SERC grant GRG59981.

References

[1] Ayache, N. and Faugeras, O.D. "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. PAMI,* Vol. 8, No. 1, p.44-54, January 1986.

[2] Ayache, N. and Faugeras, O.D. "Building a Consistent 3D Representation of a Mobile Robot Environment by Combining Multiple Stereo Views," *Proc. IJCAI,* p.808-810, 1987.

[3] Binford, T.O. "Inferring Surfaces from Images," *Artificial Intelligence,* Vol. 17, p.205-244, 1981.

[4] Binford, T.O. and Levitt, T.S. "Quasi-Invariants: Theory and Explanation," *Proc. DARPA IU Workshop*, p.819-829, 1993.

[5] Binford, T.O., Kapur D. and Mundy J.L., "The Relation between invariants and quasi-invariants", *Proc. Asian Conference in Computer Vision*, 1993.

[6] Bolles, R.C. and Horaud, R. "3DPO: A Three-Dimensional Part Orientation System," *Three Dimensional Vision,* Kanade, T. editor, Kluwer Academic Publishers, p.399-450, 1987.

[7] Borgefors, G. "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm," *IEEE Trans. PAMI,* Vol. 10, No. 6, p.849-865, November 1988.

[8] Brooks, R.A. "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," *PAMI-5*, No. 2, March 1983.

[9] Burns, J.B., Weiss, R.S. and Riseman, E.M. "The Non-existence of General-case View-Invariants," in [45].

[10] Cass, T.A. "Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty," *Proc. ECCV*, LNCS 588, Springer-Verlag, p.834-842, 1992.

[11] Clemens, D.T. and Jacobs, D.W. "Model Group Indexing for Recognition," *IEEE Trans. PAMI*, Vol. 13, No. 10, p.1007-1017, October 1991.

[12] Cox, I.J., Rehg, J.M. and Hingorani, S. "A Bayesian Multiple Hypothesis Approach to Contour Grouping," *Proc. ECCV*, LNCS 588, Springer-Verlag, p.72-77, 1992.

[13] Dhome, M., Lapreste, J.T., Rives, G. and Richetin, M. "Spatial Localization of Modelled Objects of Revolution in Monocular Perspective Vision," *Proc. ECCV*, LNCS 427, Springer-Verlag, p.475-485, 1990.

[14] Ettinger, G.J. "Large Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-Parts," *Proc. CVPR88*, p.32-41, 1988.

[15] Faugeras, O.D. and Hebert, M. "The Representation, Recognition, and Locating of 3-D Objects," *International Journal of Robotics Research*, Vol. 5, No. 3, p.27-52, 1986.

[16] Faugeras, O.D., "What can be Seen in Three Dimensions with an Uncalibrated Stereo Rig?" *Proc. ECCV*, LNCS 588, Springer-Verlag, p.563-578, 1992.

[17] Faugeras, O.D., Luong, Q.T. and Maybank, S.J. "Camera Self-Calibration: Theory and Experiments", *Proc. ECCV*, LNCS 588, Springer-Verlag, 1992.

[18] Fawcett R., Zisserman A. and Brady J.M., "Extracting Structure from an Affine View of a 3D Point Set with One or Two Bilateral Symmetries", *Image and Vision Computing*, 12, 9, 615-622, 1994.

[19] Fisher, R.B. *From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis*, John Wiley and Sons, 1989.

[20] Forsyth, D.A., Mundy, J.L., Zisserman, A.P. and Rothwell, C.A. "Recognising Curved Surfaces from their Outlines," *Proc. ECCV*, LNCS 588, Springer-Verlag, p.639-648, 1992.

[21] Forsyth, D.A. and Rothwell, C.A. "Representations of 3D objects that incorporate surface markings", in [46].

[22] Forsyth, D.A., "Recognizing Algebraic Surfaces from their Outlines," To appear *International J. of Computer Vision*, 1995.

[23] Giblin P.J. and Brassett S.A., "Local Symmetry of Plane Curves", *American Mathematical Monthly*, vol. 92, no. 10, pp. 689–707, 1985.

[24] Goad, C. "Special Purpose Automatic Programming for 3D Model-Based Vision," *Proc. DARPA IU Workshop*, p.371-381, 1983.

[25] Golub, G.H. and Van Loan, C.F., *Matrix Computations*, John Hopkins University Press, 1983.

[26] Gordon, G.,"Shape from Symmetry", *Proc. SPIE Conf. Intelligent Robots and Computer Vision VIII*, Philadelphia, 1989.

[27] Grimson, W.E.L. and Lozano-Pérez, T. "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Trans. PAMI*, Vol. 9, No. 4, p.469-482, July 1987.

[28] Grimson, W.E.L. *Object Recognition by Computer, The Role of Geometric Constraints*, MIT Press, 1990.

[29] Hartley, R.I., Gupta, R. and Chang, T. "Stereo from Uncalibrated Cameras," *Proc. CVPR92*, p.761-764, 1992.

[30] Hartley, R.I., "Euclidean Reconstruction from Uncalibrated Views", in [46].

[31] Huttenlocher, D.P. and Ullman, S. "Object Recognition Using Alignment," *Proc. ICCV1*, p.102-111, 1987.

[32] Koenderink, J.J., "What does the Occluding Contour tell us about Solid Shape?" *Perception*, Vol. 13, 1984.

[33] Kriegman, D.J. and Ponce, J. "On Recognizing and Positioning Curved 3-D Objects from Image Contours," *IEEE Trans. PAMI*, Vol. 12, No. 12, p.1127-1137, December 1990.

[34] Lamdan, Y., Schwartz, J.T. and Wolfson, H.J. "Object Recognition by Affine Invariant Matching," *Proc. CVPR88*, p.335-344, 1988.

[35] Liu J.S., Mundy J.L., Forsyth D.A., Zisserman A. and Rothwell C.A., "Efficient Recognition of Rotationally Symmetric Surfaces and Straight Homogeneous Generalized Cylinders", *CVPR*, 1993.

[36] Liu J.S., Mundy J.L. and Walker E.L., "Recognizing Arbitrary Objects from Multiple Projections," *Proc. Asian Conference in Computer Vision*, 1993.

[37] Lowe, D.G. *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[38] Lowe, D.G. "The Viewpoint Consistency Constraint," *International Journal of Computer Vision*, Vol. 1, No. 1, p.57-72, 1987.

[39] Mitsumoto H., Tamura S., Okazaki K., Kajimi N. and Fukui Y., "3D Reconstruction Using Mirror Images Based on a Plane Symmetry Recovery Method", *PAMI*, 14, 9, 941-945, 1992.

[40] Mohr, R., Morin, L. and Grosso, E., "Relative Positioning with Uncalibrated Cameras,", in [45].

[41] Moons, T., Van Gool, L., Van Diest, M. and Pauwels, E. "Affine structure from perspective images pairs", in [46].

[42] Moses, Y. and Ullman, S. "Limitations of Non Model-Based Recognition Systems," *Proc. ECCV*, LNCS 588, Springer-Verlag, p.820-828, 1992.

[43] Mukherjee D.P., Zisserman A. and Brady J.M., 'Shape from symmetry—detecting and exploiting symmetry in affine images'. To appear, *Proc. Royal Soc.*, 1995.

[44] Mundy, J.L. and Heller, A.J. "The Evolution and Testing of a Model-Based Object Recognition System," *Proc. ICCV3*, p.268-282, 1990.

[45] Mundy, J.L. and Zisserman, A. *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[46] Mundy J.L., Zisserman A., and Forsyth D.A., editors, *Applications of Invariance in Computer Vision*, LNCS 825, Springer-Verlag, 1994.

[47] Mundy, J.L. and Zisserman, A. "Repeated structures: image correspondence constraints and 3D structure recovery," in [46].

[48] Murray, D.W. "Model-Based Recognition using 3D Structure from Motion," *Image and Vision Computing*, Vol. 5, p.85-90, 1987.

[49] Murray, D.W. and Cook, D.B. "Using the Orientation of Fragmentary 3D Edge Segments for Polyhedral Object Recognition," *IJCV*, No. 2, p.153-169, 1988.

[50] Nielsen, L. "Automated Guidance of Vehicles using Vision and Projective Invariant Marking," *Automatica*, Vol. 24, p.135-148, 1988.

[51] Pillow N., Utcke S. and Zisserman A. "Viewpoint-Invariant Representation of Generalized Cylinders Using the Symmetry Set", To appear *Image and Vision Computing*, 1995.

[52] Pollard, S.B, Pridmore, T.P, Porrill, J., Mayhew, J.E.W. and Frisby, J.P. "Geometrical Modeling from Multiple Stereo Views," *International Journal of Robotics Research*, Vol. 8, No. 4, p.132-138, 1989.

[53] Ponce, J. "Invariant properties of straight homogeneous generalised cylinders," *IEEE PAMI*, **11**, 9, 951-965, 1989.

[54] Reid, I. "Recognising Parameterized Models from Range Data," D.Phil. Thesis, Department of Engineering Science, University of Oxford, Oxford, 1991.

[55] Reiss, T.H., *Recognizing Planar Objects Using Invariant Image Features*, LNCS 676, Springer Verlag, 1993.

[56] Roberts, L.G. "Machine Perception of Three-Dimensional Solids," In *Optical and Electro-optical Information Processing*, editor Tippett, J. *et al.*, MIT Press, 1965,

[57] Rothwell, C.A., Zisserman, A., Forsyth, D.A. and Mundy, J.L., "Canonical Frames for Planar Object Recognition" *Proc. ECCV*, LNCS 588, Springer-Verlag, p.757-772, 1992.

[58] Rothwell, C.A., Zisserman, A., Mundy, J.L. and Forsyth, D.A. "Efficient Model Library Access by Projectively Invariant Indexing Functions", *Proc. CVPR92*, p.109-114, 1992.

[59] Rothwell, C.A., Forsyth, D.A., Zisserman, A. and Mundy, J.L. "Extracting Projective Structure from Single Perspective Views of 3D Point Sets", *ICCV*, 573-582, 1993.

[60] Rothwell, C.A. *Recognition Using Projective Invariance*, D.Phil. Thesis, Department of Engineering Science, University of Oxford, 1993. To appear OUP.

[61] Rycroft J.E. *A Geometrical Investigation into the Projections of Surfaces and Space Curves*, Ph. D. Thesis, University of Liverpool, 1992.

[62] Sha'ashua, A. and Ullman, S. "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network," *Proc. ICCV2*, p.321-327, 1988.

[63] Sparr, G., "Notes on geometric invariants in vision", *Lund Research Report*, 1993.

[64] Springer, C.E., *Geometry and Analysis of Projective Spaces*, Freeman, 1964.

[65] Stewman, J. and Bowyer, K.W. "Creating the Perspective Projection Aspect Graph of Polyhedral Objects," *Proc. ICCV2*, 1988.

[66] Stockman, G. "Object Recognition and Localization via Pose Clustering," *Computer Vision Graphics and Image Processing*, Vol. 40, p.361-387, 1987.

[67] Sugihara, K. *Machine interpretation of Line Drawings*, MIT Press, 1986.

[68] Thompson, D.W. and Mundy, J.L. "Three-Dimensional Model Matching from an Unconstrained Viewpoint," *Proc. ICRA*, p.208-220, 1987.

[69] Ullman, S. and Basri, R., "Recognition by Linear Combination of Models", *PAMI*-13, No. 10, p.992-1006, October, 1991.

[70] Van Gool, L. Kempenaers, P. and Oosterlinck, A. "Recognition and Semi-Differential Invariants," *Proc. CVPR91*, p.454-460, 1991.

[71] Weiss, I. "Geometric Invariants and Object Recognition," *IJCV*, 10, 3, 1993.

[72] Zerroug M. and Nevatia R., "Using invariance and quasi-invariance for the segmentation and recovery of curved objects", in [46].

[73] Zisserman A., Forsyth D.A., Mundy J.L. and Rothwell C. "Recognizing General Curved Objects Efficiently", in [45].

Is Epipolar Geometry Necessary to Recover Invariants from Multiple Views?

Andrew Zisserman, Richard I. Hartley, Joe L. Mundy and Paul Beardsley

Abstract

We examine and contrast the projective properties of two simple 3D configurations. The first consists of six points, four of which are coplanar. We prove that epipolar geometry and the essential matrix can be recovered uniquely for this structure and give a constructive algorithm for this. The second configuration has four coplanar points and a single non-coplanar line. In this case it is not possible to determine the epipolar geometry. However, both structures have two projective invariants, and these are recoverable from two (uncalibrated) perspective images. We include examples of the invariants for real objects.

1 Introduction

A number of recent papers have demonstrated that vision tasks such as recognition and structure recovery can be accomplished using only projective properties [4, 7, 14]. This contrasts with "conventional" approaches where full Euclidean reconstruction of 3-space is sought. One of the immediate advantages of the projective approach is that no camera calibration is required. The intrinsic parameters need not be known since only projective properties of the rays (not angles) are used.

Given two perspective images of particular 3D configurations and assuming only image feature correspondences, we consider the following questions:

1. Can the epipolar geometry of the two cameras be uniquely recovered?
2. Can projective invariants of the 3D structure be computed?
   (So called "multiple view invariants").

The invariants sought will be to transformation by the projective group (i.e. multiplication of the homogeneous representation of 3-space by an arbitrary non-singular 4×4 matrix).

If the epipolar geometry is known, then 3D structure can be recovered up to 3D collineation i.e. up to an arbitrary projective transformation [4, 7]. Consequently, invariants to this transformation can be computed from the recovered structure (since they are unaffected by the projective transformation relating the recovered and "true" Euclidean configurations). Here we examine cases where multiple view invariants can be obtained in the absence of epipolar geometry. In particular we contrast two structures:

1. Four coplanar points, and two non-coplanar points. The non-coplanar points must be in "general position". This is made more precise below.
2. Four coplanar points, and a non-coplanar line.

The essential benefit of the four coplanar points is that they define a projective basis for the plane which can be used to *transfer* [1, 12] coordinates between the world plane and images. Any other

Structure (S)	$\dim S$	$\dim G_{\mathbf{x}}$	#invar
6 points general position	18	0	3
7 points general position (*)	21	0	6
5 points, 4 coplanar	14	1	0
6 points, 4 coplanar (*)	17	0	2
line and 4 coplanar points	15	2	2
line and 5 points, 4 coplanar	18	0?	3?
2 lines and 4 coplanar points	19	0?	4?

Table 1: The number of functionally independent scalar invariants for 3D configurations under the action of the projective group. In all cases general position is assumed e.g. the line is not coplanar with any two points. (*) indicates that the epipolar geometry can be determined from two views of the structure (though not uniquely in the case of 7 points).

planar configuration which uniquely defines a projective basis for the plane could equally well be used. For example, four coplanar lines.

That the epipolar geometry can be recovered for the six point structure has been established by [2, 11]. The derivation is repeated here, see figure 1. We extend this analysis to the determination of the essential matrix[1], Q. It is shown that Q is uniquely determined by the set of six point matches. Further, a method will be given for computing Q. The method is linear and non-iterative. This result is remarkable, since previously known methods have required 8 points for a linear solution [9] or 7 points for a solution involving finding the roots of a cubic equation [6]. In addition, the solution using 7 points leads to three possible solutions, corresponding to the three roots of the cubic. Since Q has 7 degrees of freedom [6] it is not possible to compute Q from less than 7 arbitrary points. Therefore it is somewhat surprising that the condition that four of the points are co-planar should mean that a solution from six points is possible and unique.

The second structure (four coplanar points and a line) is interesting because the simple replacement of two points by a line generates two significant changes: First, it is not possible to recover the epipolar geometry from this alone; second, the structure has an isotropy under the projective group. However, both structures have two projective invariants which can be recovered from two views.

1.1 Number of Invariants and Isotropies

As described in [13] the number of (functionally independent scalar) invariants to the action of a group G is given by:

$$\#\text{invar} = \dim S - \dim G + \dim G_{\mathbf{x}}$$

where $\dim S$ is the "dimension" of the structure, $\dim G$ the dimension of G, in this case 15, and $\dim G_{\mathbf{x}}$ the dimension of the isotropy sub-group (if any) which leaves the structure unaffected under the action of G. Examples are given in table 1.

The key point about an isotropy is that a structure with fewer degrees of freedom than the group dimension can still have invariants. In section 3.3 we discuss the isotropy of the line and four coplanar

[1]This matrix was introduced by Longuet-Higgins [9] assuming the two cameras were calibrated, and has since been extensively investigated e.g. [10]. Most of the results also apply to uncalibrated cameras of the type considered in this paper [6].

point configuration.

2 Six points, four coplanar

Consider a set of matched points $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ for $i = 1,\ldots,6$ and suppose that the points $\mathbf{X}_1,\ldots,\mathbf{X}_4$[2] corresponding to the first four matched points lie in a plane in space. Let this plane be denoted by Π. Suppose also that no three of the points $\mathbf{X}_1,\ldots,\mathbf{X}_4$ are collinear. Suppose further that the points $\mathbf{X}_5$ and $\mathbf{X}_6$ do **not** lie in that plane. Various other assumptions will be necessary in order to rule out degenerate cases. These will be noted as they occur.

In the following sections we first explain how the epipolar geometry is determined. We then prove that this configuration is sufficient to uniquely define the $\mathbf{Q}$ matrix and hence five of the points may be used as a projective basis for $\mathcal{P}^3$.

2.1 Epipolar Geometry

First it will be shown that the problem may be reduced to the case in which $\mathbf{x}'_i = \mathbf{x}_i$ for $i = 1,\ldots,4$. From the assumption that points $\mathbf{X}_1,\ldots,\mathbf{X}_4$ lie in a plane and that no three of them are collinear, it may be deduced that no three of the points $\mathbf{x}_1,\ldots,\mathbf{x}_4$ are collinear in the first image and that no three of $\mathbf{x}'_1,\ldots,\mathbf{x}'_4$ are collinear in the second image. Given this, it is possible in a straight-forward manner to find a 3×3 projective transformation matrix $\mathbf{T}$, such that $\mathbf{x}'_i = \mathbf{T}\mathbf{x}_i, i \in \{1,..,4\}$. Denoting $\mathbf{T}\mathbf{x}_i$ by the new symbol $\mathbf{x}''_i$, we see that $\mathbf{x}'_i = \mathbf{x}''_i$ for $i = 1,\ldots,4$.

Therefore, we will assume for now that $\mathbf{x}'_i = \mathbf{x}_i$ for $i = 1,\ldots,4$. This being so, it is possible to characterize the points that lie in the plane Π defined by $\mathbf{X}_1,\ldots\mathbf{X}_4$. A point $\mathbf{Y}$ lies in the plane Π if and only if it is mapped to the same point in both images.

Now consider any point $\mathbf{Y}$ in space, not on Π, and consider the epipolar plane defined by $\mathbf{Y}$ and the two camera centres (see figure 1). This plane will meet the plane Π in a straight line $L(\mathbf{Y}) \subset \Pi$. The line $L(\mathbf{Y})$ must pass through the point $\mathbf{P}$ in which the line of the camera centres meets the plane Π. This means that for all points $\mathbf{Y}$ the lines $L(\mathbf{Y})$ are concurrent, and meet at the point $\mathbf{P}$. Now we consider the images of the line $L(\mathbf{Y})$ and the point $\mathbf{P}$ as seen from the two cameras. Since the line $L(\mathbf{Y})$ lies in the plane Π it must be the same as seen from both the cameras. Let the image of $L(\mathbf{Y})$ as seen in either image be $\ell(\mathbf{Y})$. If $\mathbf{y}$ and $\mathbf{y}'$ are the image points at which $\mathbf{Y}$ is seen from the two cameras, then both points $\mathbf{y}$ and $\mathbf{y}'$ must lie on the line $\ell(\mathbf{Y})$. Since the point $\mathbf{P}$ lies in the plane Π, it must map to the same point in both images, so $\mathbf{p} = \mathbf{p}'$ and this point lies on the line $\ell(\mathbf{Y})$. Therefore, $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{p}$ are collinear. The point $\mathbf{p}$ can be identified as the epipole in the first image, since points $\mathbf{p}$ and the two camera centres are collinear. Similarly, $\mathbf{p}'$ is the epipole in the second image. Thus one point not on Π is sufficient to determine a line in each image on which the epipole must lie[3].

This discussion may now be applied to the points $\mathbf{X}_5$ and $\mathbf{X}_6$. Since $\mathbf{X}_5$ and $\mathbf{X}_6$ do not lie in the plane Π it follows that $\mathbf{x}'_5 \neq \mathbf{x}_5$ and $\mathbf{x}'_6 \neq \mathbf{x}_6$. Then the point $\mathbf{p}$ may easily be found as the point of intersection of the lines $< \mathbf{x}'_5, \mathbf{x}_5 >$ and $< \mathbf{x}'_6, \mathbf{x}_6 >$. As an aside, the point of intersection of the lines $< \mathbf{x}_5, \mathbf{x}_6 >$ and $< \mathbf{x}'_5, \mathbf{x}'_6 >$ is of interest as being the image of the point where the line through $< \mathbf{X}_5, \mathbf{X}_6 >$ meets the plane Π, see section 2.3.

[2] We adopt the notation that corresponding points in the world and image are distinguished by large and small letters. Vectors are written in bold font, e.g. $\mathbf{x}$ and $\mathbf{X}$. Homogeneous representations are used e.g. $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)^t$. $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points in two views.

[3] Another way to see this is that $\mathbf{Y}$ and $\mathbf{Y}_1$ (a virtual point), see figure 1, are collinear in the first image. This is the condition for motion parallax. As described in [8], their positions in the second image ($\mathbf{y}'$ and $\mathbf{T}\mathbf{y}$) are coincident with the focus of expansion (the epipole). We are grateful to Andrew Blake for this observation.

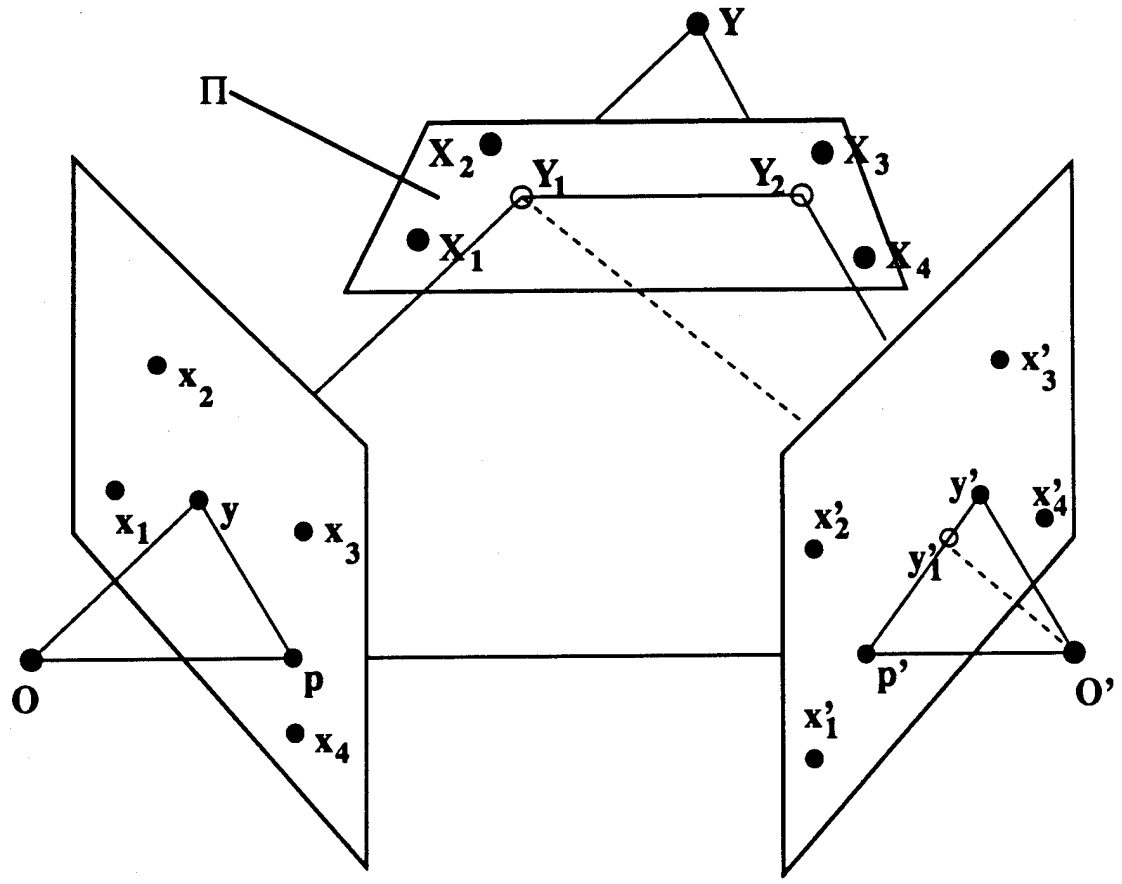


Figure 1: Epipolar geometry. The points $\mathbf{X}_1, \ldots, \mathbf{X}_4$ are coplanar, with images $\mathbf{x}_i$ and $\mathbf{x}'_i$ in the first and second images respectively. The epipolar plane defined by the point $\mathbf{Y}$ and optical centers $\mathbf{O}$ and $\mathbf{O}'$ intersects the plane Π in the line $L(\mathbf{Y}) = < \mathbf{Y}_1, \mathbf{Y}_2 >$, where $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are the intersections of Π with the lines $< \mathbf{Y}, \mathbf{O} >$ and $< \mathbf{Y}, \mathbf{O}' >$ respectively.
The epipolar line may be constructed in the second image as follows: Determine the plane projective transformation such that $\mathbf{x}'_i = \mathtt{T}\mathbf{x}, i \in \{1, .., 4\}$. Use this transformation to transfer the point $\mathbf{y}$ to $\mathbf{y}'_1 = \mathtt{T}\mathbf{y}$. This determines two points in the second image, $\mathbf{y}'$ and $\mathtt{T}\mathbf{y}$, which are projections of points ($\mathbf{Y}$ and $\mathbf{Y}_1$) on the line $< \mathbf{O}, \mathbf{Y} >$. This defines the epipolar line of $\mathbf{Y}$ in the second image. A second point, not on Π, will define its corresponding epipolar lines. The epipole lies on both lines, so is determined by their intersection. A similar construction gives epipolar lines and hence the epipole in the first image.

The previous discussion indicates how the epipole may be found. This construction will succeed unless the two lines $< \mathbf{x}'_5, \mathbf{x}_5 >$ and $< \mathbf{x}'_6, \mathbf{x}_6 >$ are the same. The two lines will be distinct unless the two points $\mathbf{X}_5$ and $\mathbf{X}_6$ lie in a common plane with the two camera centres.

To summarise:

1. Calculate the plane projective transformation matrix $\mathbf{T}$, such that $\mathbf{x}'_i = \mathbf{T}\mathbf{x}_i, i \in \{1, .., 4\}$.
2. Determine the epipole, $\mathbf{p}'$, in the second image as the intersection of the lines $< \mathbf{T}\mathbf{x}_5, \mathbf{x}'_5 >$ and $< \mathbf{T}\mathbf{x}_6, \mathbf{x}'_6 >$. Note, these lines are given by $\mathbf{T}\mathbf{x}_i \times \mathbf{x}'_i$, $i \in \{5, 6\}$ [16]. Similarly, the epipole in the first image is the intersection of the lines $\mathbf{T}^{-1}\mathbf{x}'_i \times \mathbf{x}_i$, $i \in \{5, 6\}$.
3. The epipolar line in the second image corresponding to a point $\mathbf{x}$ in the first is given by $\mathbf{T}\mathbf{x} \times \mathbf{p}'$.

2.2 Computation of Essential Matrix

The essential matrix, Q, satisfies the condition

$$\mathbf{x}'^{\top}_i Q \mathbf{x}_i = 0 \tag{1}$$

for all i. As in the previous section the problem of determining the matrix Q is reduced to the case in which $\mathbf{x}'_i = \mathbf{x}_i$ for $i = 1, \ldots, 4$. If $\mathbf{x}''_i = \mathbf{T}\mathbf{x}'_i$ for $i = 1, \ldots, 4$, then

$$0 = \mathbf{x}'^{\top}_i Q \mathbf{x}_i = \mathbf{x}'^{\top}_i Q \mathbf{T}^{-1} \mathbf{x}''_i \ . \tag{2}$$

So, denoting $Q_1 = Q\mathbf{T}^{-1}$, the task now becomes that of determining Q_1 such that

$$\mathbf{x}'^{\top}_i Q_1 \mathbf{x}''_i = 0 \tag{3}$$

for all i. In addition, $\mathbf{x}'_i = \mathbf{x}''_i$ for $i = 1, \ldots, 4$. Once Q_1 has been determined, the original matrix Q may be retrieved using the relationship

$$Q = Q_1 \mathbf{T} \ . \tag{4}$$

Now, if Q is the essential matrix corresponding to the set of matched points, then since $\mathbf{p}$ is the epipole in the first image, we have an equation

$$Q\mathbf{p} = 0$$

and since $\mathbf{p}' = \mathbf{p}$ is the epipole in the second image, it follows also that

$$\mathbf{p}^{\top} Q = 0$$

Furthermore, for $i = 1, \ldots, 4$, we have $\mathbf{x}_i = \mathbf{x}'_i$, and so, $\mathbf{x}_i^{\top} Q \mathbf{x}_i = 0$. For $i = 5, 6$, we have $\mathbf{x}'_i = \mathbf{x}_i + \alpha_i \mathbf{p}$. Therefore, $0 = \mathbf{x}'^{\top}_i Q \mathbf{x}_i = (\mathbf{x}_i + \alpha_i \mathbf{p})^{\top} Q \mathbf{x}_i = \mathbf{x}_i^{\top} Q \mathbf{x}_i$. So for all $i = 1, \ldots, 6$,

$$\mathbf{x}_i^{\top} Q \mathbf{x}_i = 0 \ .$$

This should give more than enough equations in general to solve for Q, however, the existence and uniqueness of the solution need to be proven

Now, a new piece of notation will be introduced. For any vector $\mathbf{t} = (t_x, t_y, t_y)^\top$ we define a skew-symmetric matrix, $S(\mathbf{t})$ according to

$$S(\mathbf{t}) = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} . \tag{5}$$

Any 3×3 skew-symmetric matrix can be represented in this way for some vector $\mathbf{t}$. Matrix $S(\mathbf{t})$ is a singular matrix of rank 2, unless $\mathbf{t} = 0$. Furthermore, the null-space of $S(\mathbf{t})$ is generated by the vector $\mathbf{t}$. This means that $\mathbf{t}^\top S(\mathbf{t}) = S(\mathbf{t})\,\mathbf{t} = 0$ and that any other vector annihilated by $S(\mathbf{t})$ is a scalar multiple of $\mathbf{t}$.

We now prove the existence and uniqueness of the solution for the essential matrix.

Lemma 2.1 *Let* $\mathbf{p}$ *be a point in projective 2-space and let* $\{\mathbf{x}_i\}$ *be a further set of points. If there are at least three distinct lines among the lines* $< \mathbf{p}, \mathbf{x}_i >$ *then there exists a unique matrix* Q *such that*

$$\mathbf{p}^\top Q = Q\mathbf{p} = 0$$

and for all i

$$\mathbf{x}_i' Q \mathbf{x}_i = 0$$

Furthermore, Q *is skew-symmetric, and hence* $Q \approx S(\mathbf{p})$.

Proof : Let us assume without loss of generality that the lines $< \mathbf{p}, \mathbf{x}_i >$ for $i = 1, \ldots, 3$ are distinct.

Let T_2 be a non-singular matrix such that

$$\begin{aligned} \mathrm{T}_2\mathbf{p} &= (0,0,1)^\top \\ \mathrm{T}_2\mathbf{x}_1 &= (1,0,0)^\top \\ \mathrm{T}_2\mathbf{x}_2 &= (0,1,0)^\top \end{aligned}$$

Suppose that $\mathrm{T}_2\mathbf{x}_3 = (r,s,t)^\top$. Since the lines $< \mathbf{p}, \mathbf{x}_i >$ are distinct, so must be the lines $< \mathrm{T}_2\mathbf{p}, \mathrm{T}_2\mathbf{x}_i >$. From this it follows that both r and s are non-zero, for otherwise, the line $< \mathrm{T}_2\mathbf{p}, \mathrm{T}_2\mathbf{x}_3 >$ must be the same as $< \mathrm{T}_2\mathbf{p}, \mathrm{T}_2\mathbf{x}_i >$ for $i = 1$ or 2. Now, define the matrix $Q_2 = \mathrm{T}_2{}^\top Q \mathrm{T}_2$. Then

$$\mathrm{T}_2{}^\top Q_2 (0,0,1)^\top = \mathrm{T}_2{}^\top Q_2 \mathrm{T}_2 \mathbf{p} = Q\mathbf{p} = 0$$

and so

$$Q_2(0,0,1)^\top = 0 \tag{6}$$

Similarly,

$$(0,0,1)Q_2 = 0 \tag{7}$$

Next,

$$(1,0,0)Q_2(1,0,0)^\top = \mathbf{x}_1{}^\top \mathrm{T}_2{}^\top Q_2 \mathrm{T}_2 \mathbf{x}_1 = \mathbf{x}_1{}^\top Q \mathbf{x}_1 = 0 \tag{8}$$

and similarly,

$$(0,1,0)Q_2(0,1,0)^\top = 0 \ . \tag{9}$$

and

$$(r,s,t)Q_2(r,s,t)^\top = 0 \ . \tag{10}$$

Now, writing

$$Q_2 = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix}$$

equation (6) implies $c = f = j = 0$. Equation (7) implies $g = h = j = 0$. Equation (8) implies $a = 0$ and equation (9) implies $e = 0$. Finally, equation (10) implies $rs(b + d) = 0$ and since $rs \neq 0$ this yields $b + d = 0$. So,

$$Q_2 = \begin{pmatrix} 0 & b & 0 \\ -b & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

which is skew-symmetric. Therefore, $Q = \mathrm{T}_2^{-1\mathsf{T}} Q_2 \mathrm{T}_2^{-1}$ is also skew-symmetric.

The first part of the lemma has been proven. Now, since Q is skew-symmetric and $Q\mathbf{p} = 0$, it follows that $Q = S(\mathbf{p})$, as required. This shows uniqueness of the essential matrix Q. To show the existence of a matrix Q satisfying all the conditions of the lemma, it is sufficient to observe that a skew-symmetric matrix Q has the property that $\mathbf{x}_i^\mathsf{T} Q \mathbf{x}_i = 0$ for any vector $\mathbf{x}_i$. □

This lemma allows us to give an explicit form for the matrix Q expressed in terms of the original matched points.

Theorem 2.2 *Let $\{\mathbf{x}_i'\} \leftrightarrow \{\mathbf{x}_i\}$ be a set of 6 image correspondences derived from 6 points $\mathbf{X}_i$ in space, and suppose it is known that the points $\mathbf{X}_1, \ldots, \mathbf{X}_4$ lie in a plane. Let T be a 3×3 matrix such that $\mathbf{x}_i' = \mathrm{T}\mathbf{x}_i$ for $i = 1, \ldots, 4$. Suppose that the lines $< \mathbf{x}_5', \mathrm{T}\mathbf{x}_5 >$ and $< \mathbf{x}_6', \mathrm{T}\mathbf{x}_6 >$ are distinct and let $\mathbf{p}$ be their intersection. Suppose further that among the lines $< \mathbf{x}_i', \mathbf{p} >$ there are at least three distinct lines. Then there exists a unique essential matrix Q satisfying the point correspondences and the condition of coplanarity of the points $\mathbf{X}_1, \ldots, \mathbf{X}_4$ and Q is given by the formula*

$$Q = S(\mathbf{p})\mathrm{T}$$

The conditions under which a unique solution exists may be expressed in geometrical terms. Namely :

1. Points $\mathbf{X}_1, \ldots, \mathbf{X}_4$ lie in a plane Π, but no three of them are collinear.
2. Points $\mathbf{X}_5$ and $\mathbf{X}_6$ do not lie in the plane Π, and do not lie in a common plane passing through the two camera centres.
3. The points $\mathbf{X}_1, \ldots, \mathbf{X}_6$ do not all lie in two planes passing through the camera centres.

Under the above conditions, the essential matrix Q is determined uniquely by the set of image correspondences. Note that according to [4, 7], this in turn determines the locations of the points themselves and the cameras up to a projective transformation of 3-space.

2.3 Projective invariants

The meaning of the 3D projective invariants can most readily be appreciated from figure 2. The line, Λ, formed from the two non-coplanar points intersects the plane Π in a unique point $\mathbf{X}_I$. This construction is unaffected by projective transformations of $\mathcal{P}^3$. There are then 5 coplanar points and consequently two plane projective invariants - which are also invariants of the 3D transformation.

As described in section 2.1, the image of $\mathbf{X}_I$ can be determined from two views (see [15] for an alternative derivation). We then have the image of five coplanar points, for which plane projective invariants may be calculated. These invariants have the same value calculated on Π or from any projection of Π. This construction does not require epipolar calibration to be known.

To summarise:

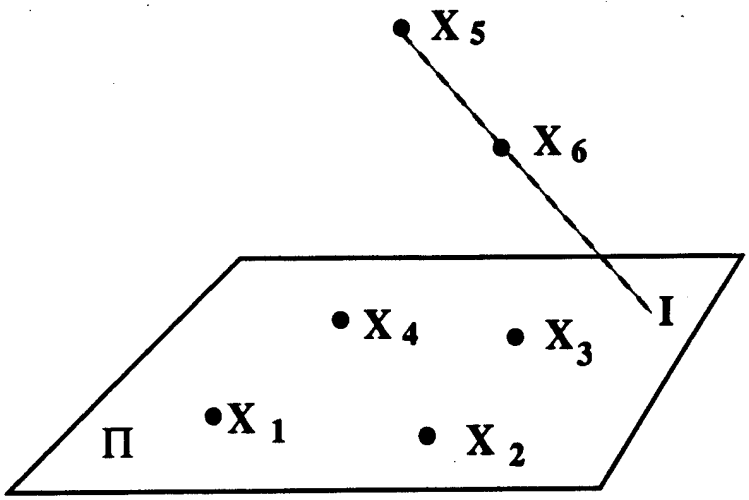


Figure 2: The projective invariant of 6 points, 4 coplanar (points 1-4), can be computed by intersecting the line, Λ, through the non-planar points (5 and 6) with the common plane. There are then 5 coplanar points, for which two invariants to the plane projective group can be calculated.

1. Determine the imaged intersection $\mathbf{x}'_I$ of the plane Π and the line Λ in the second image as the intersection of the lines $\mathbf{T}\mathbf{x}_5 \times \mathbf{T}\mathbf{x}_6$ and $\mathbf{x}'_5 \times \mathbf{x}'_6$.

2. Calculate the two plane projective invariants of five points, $\mathbf{x}'_i$, $i \in \{1,..,4\}$, and $\mathbf{x}'_5 = \mathbf{x}'_I$. These are given by

$$I_1 = \frac{|m_{125}||m_{134}|}{|m_{124}||m_{135}|} \qquad I_2 = \frac{|m_{124}||m_{235}|}{|m_{234}||m_{125}|} \tag{11}$$

where m_{jkl} is the matrix $[\mathbf{x}'_j \mathbf{x}'_k \mathbf{x}'_l]$ and $||$ is a determinant.

2.3.1 Relation between 2D invariants and algebraic invariants of 3D points

Six 3D points in $\mathcal{P}^3$ in general position have three projective invariants. The coplanarity reduces by one the number of invariants (one of the invariants will be zero). We may arbitrarily choose coordinates for 5 of the points of the six point configuration (any other coordinates of the five points can be transformed to these by a collineation of $\mathcal{P}^3$):

$$\begin{aligned} \mathbf{X}_1 &= (1,0,0,0)^\mathsf{T} \\ \mathbf{X}_2 &= (0,1,0,0)^\mathsf{T} \\ \mathbf{X}_3 &= (0,0,1,0)^\mathsf{T} \\ \mathbf{X}_5 &= (0,0,0,1)^\mathsf{T} \\ \mathbf{X}_6 &= (1,1,1,1)^\mathsf{T} \end{aligned}$$

The fourth coplanar point then has coordinates:

$$\mathbf{X}_4 = (\alpha,\beta,\gamma,0)^\mathsf{T}$$

The coordinates of this point give rise to the two independent projective invariants of the six points:

$$I_1^3 = \alpha/\gamma \qquad I_2^3 = \beta/\gamma \tag{12}$$

For the five point planar invariants we use the coordinates of the first four points restricted to the plane (the subordinate geometry):

$$\begin{aligned}\bar{\mathbf{X}}_1 &= (1,0,0)^\mathsf{T}\\ \bar{\mathbf{X}}_2 &= (0,1,0)^\mathsf{T}\\ \bar{\mathbf{X}}_3 &= (0,0,1)^\mathsf{T}\\ \bar{\mathbf{X}}_4 &= (\alpha,\beta,\gamma)^\mathsf{T}\end{aligned}$$

The intersection of the line $< \mathbf{X}_5, \mathbf{X}_6 >$ with Π is given by

$$\bar{\mathbf{X}}_I = (1,1,1)^\mathsf{T}$$

Then from (11) the five point planar invariants are:

$$I_1 = \beta/\gamma \qquad I_2 = \gamma/\alpha \tag{13}$$

i.e. simply functions of the two 3D invariants in (12) above as expected.

3 A line and four coplanar points

Here the two non-coplanar points of the previous section are "replaced" by a line, Λ. As with the previous configuration this structure has two projective invariants (which are determined from the two five point invariants of the four coplanar points and the intersection of the line with the plane Π) which can be determined from two views. However, it is no longer possible to recover the epipolar geometry, it is not even possible to restrict the epipole to a line in each image.

3.1 Constraints on epipolar geometry

Surprisingly the image of Λ in each view *adds no constraints at all* towards solving for the epipolar geometry or essential matrix. To see this geometrically consider the back projection of a point imaged in two views. Each point back projects to a ray. In general two lines are skew in $\mathcal{P}^3$, so the condition that they intersect (since they arise from a common point) constrains the imaging geometry. In contrast the back projection of a line is a plane, and in general two planes intersect in a line in $\mathcal{P}^3$. Consequently, no constraint is given. Adding a third view does constrain the geometry since three planes intersect in a point in general, not a line, in $\mathcal{P}^3$.

3.2 Projective invariants

As in the six point case the invariant under projective transformations of $\mathcal{P}^3$ can be obtained from the five point planar invariants of the four coplanar points and the intersection of Λ with Π. Again this can be calculated from two views, where here the plane projective transformation is used to transfer a line, the image of Λ. This construction does not require epipolar calibration to be known.

As described in table 1, the line and four coplanar point configuration has only 15 degrees of freedom. That two invariants can be constructed indicates the presence of an isotropy sub-group. The action of this group is described below.

To summarise:

Given a set of matched points $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ for $i = 1,\ldots,4$ which are the images of coplanar points, together with the images $\mathbf{l}$ and $\mathbf{l}'$ of a line in $\mathcal{P}^3$ not on Π[4]

[4] If Λ does lie on Π, the transferred line will be coincident with $\mathbf{l}'$, i.e. $\mathbf{l}' = \mathbf{T}^{-T}\mathbf{l}$.

1. Determine the imaged intersection $\mathbf{x}'_I$ of the plane Π and the line Λ in the second image as the intersection of the lines $\mathbf{l}'$ and $\mathtt{T}^{-T}\mathbf{l}$. This is given by $\mathbf{x}'_I = \mathbf{l}' \times \mathtt{T}^{-T}\mathbf{l}$ [16].

2. Calculate the two plane projective invariants of five points, $\mathbf{x}'_i$, $i \in \{1,..,4\}$, and $\mathbf{x}'_5 = \mathbf{x}'_I$ using equation (11).

3.3 Existence of an Isotropy

As explained in section 1.1 in order for the line and four coplanar points to have two invariants under collineation of $\mathcal{P}^3$there must by an isotropy sup-group acting. In this section we give a simple derivation of this sub-group which leaves the structure unchanged under the action of the projective group, and determine its action on $\mathcal{P}^3$.

The construction of the sub-group is in two stages:

1. Construct the sub-group for which Π is a plane of fixed points. This is necessary since four points remain fixed under the action of the sub-group, and consequently every point on the plane is unchanged (as four points define a basis for the plane).

2. Construct the sub-group of (1) for which the line Λ is a fixed line. Note this does not have to be a line of fixed points since only one point on the line (its intersection with Π must be unchanged).

We adopt the notation of section 2.3.1 for the six points. The line Λ is given in its homogeneous parametric representation by

$$\Lambda = \zeta(1,1,1,1)^\mathsf{T} + \eta(0,0,0,1)^\mathsf{T} \tag{14}$$

First, in order for Π to be a plane of fixed points it is necessary and sufficient that the 4×4 transformation matrix T satisfies

$$\mathbf{X}_i = \mathtt{T}\mathbf{X}_i, \qquad i \in \{1,..,4\}$$

It is a simple matter to show that T must have the form

$$\mathtt{T} = \begin{pmatrix} \mu_1 & 0 & 0 & \mu_2 \\ 0 & \mu_1 & 0 & \mu_3 \\ 0 & 0 & \mu_1 & \mu_4 \\ 0 & 0 & 0 & \mu_5 \end{pmatrix} \tag{15}$$

where $\mu_i, i \in \{1,..,5\}$ parameterise the sub-group which has four dof (only their ratio is significant).

Second, we determine the sub-group which leaves Λ fixed. This can be carried out using Pluckerian line coordinates [16], but here we use the parametric representation (14) above. Under the action of the isotropy group the points on the line need not be fixed, but the transformed points must still lie on Λ. The transformation of two points is sufficient to determine the transformed lines (three are required to determine the transformation of all the points on the line). By inspection $\mathtt{T}\mathbf{X}_5$ and $\mathtt{T}\mathbf{X}_5$ satisfy (14) iff $\mu_2 = \mu_3 = \mu_4$. Hence we arrive at

$$\mathtt{T} = \begin{pmatrix} \mu_1 & 0 & 0 & \mu_2 \\ 0 & \mu_1 & 0 & \mu_2 \\ 0 & 0 & \mu_1 & \mu_2 \\ 0 & 0 & 0 & \mu_5 \end{pmatrix} \tag{16}$$

which is a two dimensional sub-group of the collineations of $\mathcal{P}^3$.

Images	I_1	I_2
D,A	0.440	-0.968
D,B	0.378	-1.117
B,A	0.371	-1.170
C,E	0.370	-1.150
F,A	0.333	-1.314
D,A,B	0.372	-1.151
C,E,D	0.369	-1.148
F,A,C	0.370	-1.196
C,A,B,D,E	0.375	-1.140
F,A,B,C,D,E	0.369	-1.170

Table 2: This table shows the line and four coplanar point invariants extracted from several combinations of views using points 2,4,14,17 and the line between points 6 and 13.

It is interesting to examine the transformation of $\mathcal{P}^3$under the action of T. The clearest way to see this is to determine the eigen-vectors of T. These are the fixed points of the collineation. We find

$$\begin{aligned}\mathbf{E}_1 &= (1,0,0,0)^\mathsf{T}\\ \mathbf{E}_2 &= (0,1,0,0)^\mathsf{T}\\ \mathbf{E}_3 &= (0,0,1,0)^\mathsf{T}\\ \mathbf{E}_4 &= (\mu_2,\mu_2,\mu_2,\mu_5-\mu_1)^\mathsf{T}\end{aligned}$$

The first three are degenerate with eigen-value μ_1, the fourth has eigen-value μ_5. As expected any point on the plane $\mathbf{X} = \nu_1\mathbf{X}_1 + \nu_2\mathbf{X}_2 + \nu_3\mathbf{X}_3$ is unchanged by T (since after the transformation all the basis vectors are multiplied by μ_1). The fourth eigen-vector is a fixed point on Λ. To see the effect of the isotropy group on points not on Π, consider any line L containing $\mathbf{E}_4$. This will intersect Π at some point, $\mathbf{X}_\Pi$ say, and any point on the line $\mathbf{X}$ is given by $\mathbf{X} = \zeta\mathbf{E}_4 + \eta\mathbf{X}_\Pi$. After the transformation the point is $\mathtt{T}\mathbf{X} = \mu_5\zeta\mathbf{E}_4 + \mu_1\eta\mathbf{X}_\Pi$ which still lies on L i.e. any line through $\mathbf{E}_4$ is a fixed line under the isotropy. Consequently, since every point in $\mathcal{P}^3$ lies on a line through $\mathbf{E}_4$, the action of T on $\mathcal{P}^3$is to move points towards (or away from) $\mathbf{E}_4$, with only $\mathbf{E}_4$ and points on Π remaining unchanged.

4 Experimental Results

The images used for acquisition and assessment are shown in figure 3.

A local implementation of Canny's edge detector [3] is used to find edges to sub-pixel accuracy. These edge chains are linked, extrapolating over any small gaps. A piecewise linear graph is obtained by incremental straight line fitting. Edgels in the vicinity of tangent discontinuities ("corners") are excised before fitting as the edge operator localisation degrades with curvature. Vertices are obtained by extrapolating and intersecting the fitted lines. Figure 4 shows a typical line drawing.

Although invariants obtained from two views are fairly stable, improvements in stability are achieved by augmenting with measurements from other views. At present this is carried out in a primitive fashion by determining the intersection point in a least squares manner. See table 2.

5 Conclusions

We have demonstrated that multiple view invariants can indeed be recovered without epipolar calibration being necessary. The discussion applies as well to analogues of this configuration, for example: four coplanar lines and two non-coplanar points, and five lines (four coplanar).

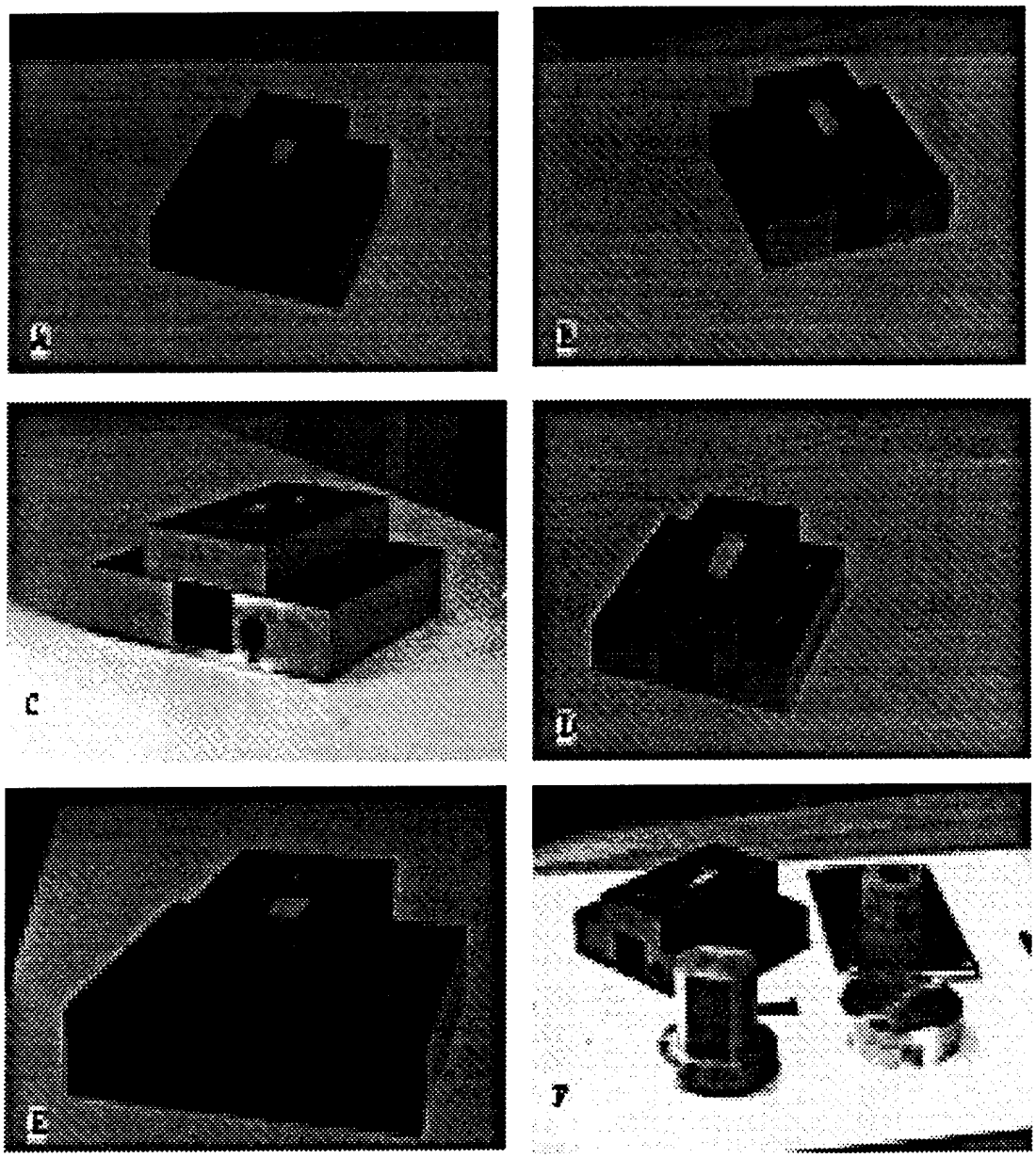


Figure 3: Images of a hole punch captured with different lenses and viewpoints. These are used for structure acquisition and transfer evaluation.

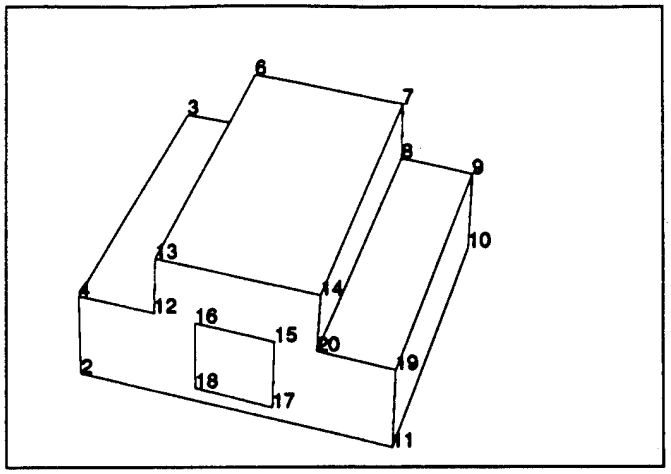


Figure 4: Line drawing of the hole punch extracted from image A in figure 3. Points 1 and 5 are occluded in this view.

We have also shown that for the structure with an isotropy it is not possible to determine the epipolar geometry. We conjecture that this is always the case i.e. if the 3D structure has an isotropy under the projective group then it is not possible to determine the epipolar geometry (it can only be constrained up to a family of solutions). Of course the converse is not true - four coplanar points and n non-coplanar lines is not sufficient to determine the epipolar geometry for any n.

Appendix: Why are six points (four coplanar) sufficient ?

With 8 points or more it is possible to solve for the matrix Q by solving a set of linear equations. If there are fewer than 8 points, the set of linear equations will be under-determined, and hence there will be a family of solutions. It is instructive to consider how the extra condition that four of the points should be coplanar cuts this family down to a single solution. Let us consider a particular example.

Consider a set of 6 matched points $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ as follows :

$$\begin{array}{l} (1,0,0)^\top \leftrightarrow (1,0,0)^\top \\ (0,1,0)^\top \leftrightarrow (0,1,0)^\top \\ (0,0,1)^\top \leftrightarrow (0,0,1)^\top \\ (1,1,1)^\top \leftrightarrow (1,1,1)^\top \\ (1,0,0)^\top \leftrightarrow (-1,1,1)^\top \\ (0,1,0)^\top \leftrightarrow (-1,1,1)^\top \end{array} \tag{17}$$

Assume that the first 4 points lie in a plane. From the previous discussion, it is obvious that the epipole is the point $(-1,1,1)^\top$, and hence that

$$Q = S((-1,1,1)^\top = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} .$$

However, we will compute Q directly. Each of the six point correspondences gives rise to an equation $\mathbf{x}'^\top_i Q \mathbf{x}_i = 0$ which is linear in the entries of Q. Since there are six equations in nine unknowns, there

will be a 3-parameter family of solutions. It is easily verified, therefore, that the general solution is given by

$$Q = \begin{pmatrix} 0 & A & -A \\ B & 0 & B \\ C & -C-2B & 0 \end{pmatrix} . \tag{18}$$

Now, the condition $\det(Q) = 0$ yields an equation $2AB(C+B) = 0$, and hence, either $C = -B$ or $A = 0$ or $B = 0$. Thus, Q has one of the forms

$$Q = \begin{pmatrix} 0 & A & -A \\ B & 0 & B \\ -B & -B & 0 \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 0 & 0 \\ B & 0 & B \\ C & -C-2B & 0 \end{pmatrix} \text{ or } \begin{pmatrix} 0 & A & -A \\ 0 & 0 & 0 \\ C & -C & 0 \end{pmatrix} . \tag{19}$$

We consider the first one of these solutions Since Q is determined only up to scale, we may choose $B = 1$, and so

$$Q = \begin{pmatrix} 0 & A & -A \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} . \tag{20}$$

Next, we investigate the condition that the first four matched points lie in a plane. To do this, it is necessary to find a pair of camera matrices that realize (see citeHartley92) the matrix Q. It does not matter which realization of Q is picked, since any other choice will be equivalent to a projective transformation of object space (see [7]), which will take planes to planes. Accordingly, since Q factors as

$$Q = \begin{pmatrix} -A & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

a realization of Q is given by the two camera matrices

$$M = (I \mid 0) \quad \text{and} \quad M' = \left(\begin{array}{ccc|c} 1 & & & 1 \\ & -A & & A \\ & & -A & A \end{array}\right)$$

Then it is easily verified that the points

$$\mathbf{X}_1 = (1,0,0,0)^\top \ , \ \mathbf{X}_2 = (0,1,0,0)^\top \ , \ \mathbf{X}_3 = (0,0,1,0)^\top \ , \ \mathbf{X}_4 = (1,1,1,k)^\top \ ,$$

where k is defined by $1+k = -A+kA$, are mapped by the two cameras to the required image points as specified by (17). However, the requirement that these four points lie in a plane means that $k = 0$ and hence that $A = -1$. Substituting this value in (20) yields the expected matrix $Q = S((-1,1,1)^\top)$. It may be verified that the two other choices for Q given in (19) do not lead to any further solution.

The role of the coplanarity condition now becomes clear. Without this condition, there are a family of solutions for the essential matrix Q. Only one of the family of solutions is consistent with the condition that the four points lie in a plane.

Acknowledgements

We are very grateful for helpful discussions with David Forsyth. Sabine Demey carried out the numerical experiments. This research was supported by the UK Science and Engineering Research Council and by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

References

[1] E. B. Barrett, Michael H. Brill, Nils N. Haag and Paul M. Payton, *"Invariant Linear Methods in Photogrammetry and Model Matching"* In [14], pp. 319-336, (1992).

[2] Beardsley, P., Sinclair, D., Zisserman, A., Ego-motion from Six Points, Insight meeting, Catholic University Leuven, Feb. 1992.

[3] Canny J.F. "A Computational Approach to Edge Detection," *PAMI*-6, No. 6. p.679-698, 1986.

[4] Faugeras, O., "What can be seen in three dimensions with an uncalibrated stereo rig?", Proc. of ECCV-92, G. Sandini Ed., LNCS-Series Vol. 588, Springer- Verlag, 1992, pp. 563 – 578.

[5] R. Hartley, "An investigation of the essential matrix", GE internal report, available upon request, preparing for publication.

[6] R. Hartley, "Invariants of Points Seen in Multiple Images", Submitted for publication, 1992.

[7] R. Hartley, R. Gupta and Tom Chang, "Stereo from Uncalibrated Cameras" *Proceedings of CVPR92*, 1992.

[8] H.C. Longuet-Higgins and K. Pradzny, "The interpretation of a moving retinal image" Proc. R. Soc. Lond., B208, 385–397, 1980.

[9] H.C. Longuet-Higgins, *"A computer algorithm for reconstructing a scene from two projections,"* Nature, Vol. 293, 10 Sept. 1981.

[10] Maybank, S.J. Properties of essential matrices, *Int. J. Imaging Systems and Technology*, 2, 380–384, 1990.

[11] Mohr, R., Projective geometry and computer vision, To appear in *Handbook of Pattern Recognition and Computer Vision*, Chen, Pau and Wang editors, 1992.

[12] Mohr, R., Morin, L. and Grosso E. "Relative Positioning with Uncalibrated Cameras" In [14], pp. 440-460.

[13] J. L. Mundy and A. Zisserman. "Introduction – towards a new framework for vision" In [14], pp. 1–49.

[14] J. L. Mundy and A. Zisserman (editors), "Geometric Invariance in Computer Vision,", MIT Press, Cambridge Ma, 1992.

[15] Quan, L. and Mohr, R., Towards Structure from Motion for Linear Features through Reference Points, *Proc. IEEE Workshop on Visual Motion*, 1991.

[16] J.G. Semple and G. T. Kneebone *"Algebraic Projective Geometry"* Oxford University Press, (1952), ISBN 0 19 8531729.