CAR-TR-804
CS-TR-3578

N00014-95-1-0521
December 1995

# Knowledge-Based Integration of IU Algorithms

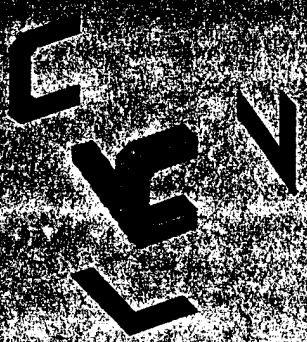
Chandra Shekhar  Shyam Kuttikkad  Rama Chellappa

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

Monique Thonnat

INRIA Sophia-Antipolis
2004 Route des Lucioles
BP 93, 06902 Sophia-Antipolis Cedex
France

COMPUTER VISION LABORATORY

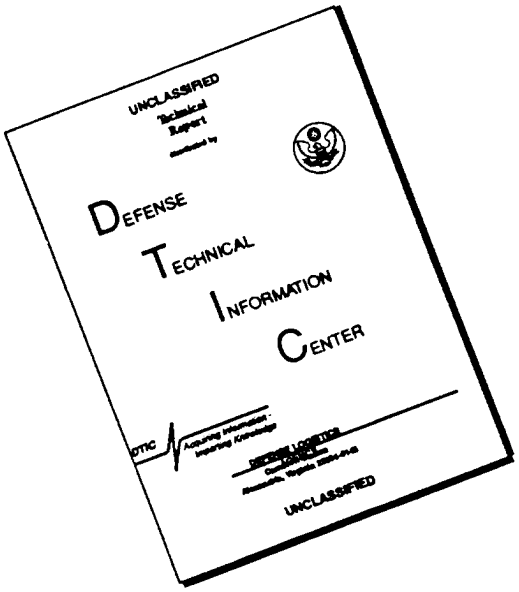




CENTER FOR AUTOMATION RESEARCH

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742-3275

# DISCLAIMER NOTICE




THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

CAR-TR-804 N00014-95-1-0521
CS-TR-3578 December 1995

# Knowledge-Based Integration of IU Algorithms

Chandra Shekhar Shyam Kuttikkad Rama Chellappa

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

Monique Thonnat

INRIA Sophia-Antipolis
2004 Route des Lucioles
BP 93, 06902 Sophia-Antipolis Cedex
France

## Abstract

This paper deals with the integration of image understanding (IU) programs using a knowledge-based approach. The basic concepts of program integration are discussed, and a simple problem-solving model for program integration is outlined. Two types of reasoning, planning and execution control, are identified. A system developed using this model, called OCAPI (Optimizing, Controlling and Automating the Processing of Images), is introduced. OCAPI is in an AI environment in which the reasoning used by the IU specialist is formally represented using frames and production rules. An example of an application developed using OCAPI is presented, and the advantages and shortcomings of this approach are discussed.



The support of the Office of Naval Research under Grant N00014-95-1-0521, and the Advanced Research Projects Agency (ARPA Order No. C635) is gratefully acknowledged, as is the help of Sandy German in preparing this paper.

## 1 Introduction

In any rapidly-evolving field of research such as image understanding (IU), the development of new methods is often far ahead of their actual use in real applications. New methods continue to proliferate, each with its own advantages, disadvantages, constraints, and areas of applicability. These new techniques, if used correctly, can often provide solutions that are robust, efficient, and adaptive. Unfortunately, they also tend to be difficult to understand and utilize because of their complexity. Consequently, the real users of IU methods, who are often non-specialists, may not be able to use them effectively. There is thus a wide gulf between specialists and end-users of IU algorithms.

To provide users with greater flexibility and power in solving their problems, program integration systems have been developed. These systems range from graphical script generators, to IU toolkits, to object-oriented protocols for data and program interchange. The emphasis is on abstracting the types of objects and computational geometries used in image understanding into useful programming constructs [4]. One of the objectives is to enable the rapid design of algorithms using a tool-box of pre-existing constructs, and prototyping of longer processing chains by linking simpler elements together. The user is often provided with a visual programming environment (VPE), which enables him to mix and match the available methods. For the most part, these systems offer only *syntactic integration* of IU programs. They provide a means of integrating code and usage syntaxes, but they do not incorporate knowledge about the programs.

The task of solving an IU problem using a computer may involve complex decision making at various levels, for which IU expertise may be necessary. Very often a number of stages of processing may have to be linked together to achieve the desired IU objective. At each of these stages, there may be a number of possible methods, from which the most appropriate one has to be selected. A method may have one or more parameters, which have to be initialized before execution. Often, due to uncertainty in the data and in the problem model, it may be necessary to start with a rough guess of the parameters, execute the algorithm, examine the results, and if necessary, modify the parameter values, and repeat the procedure until results of the desired quality are obtained.

It is clear from the above remarks that purely syntactic integration is inadequate for effective problem-solving in IU. In this paper, we focus on *semantic integration* of IU procedures. The goal of such systems is not merely to provide a few basic IU tools and a graphical interface for generating scripts, but to enable the formal and precise representation of the problem-solving expertise of the IU specialist, and to emulate the strategy used by an expert in employing available IU programs. The objective of automatic supervision of programs is to maximally automate an existing processing activity, so that the intervention of the specialist during the use of the programs is minimized or even completely eliminated. The user is thus freed from the necessity of going through the same kind of reasoning as the specialist does, in order to make the IU programs work in his application. Semantic integration can be viewed as a means of achieving automatic supervision. In the rest of this paper, these two terms are used interchangeably.

It is obvious from the preceding discussion that a formal and structured software architecture is necessary to integrate and automate IU algorithms. OCAPI (Optimizing, Controlling and Automating the Processing of Images) is one such architecture. In this paper, we describe the basic principles of algorithm integration, the OCAPI architecture [5], and the integration of an IU problem (in SAR ATR) using OCAPI.

## 2 Previous Work

The use of a knowledge-based approach to the integration of IU methods is a relatively recent phenomenon. In Japan, considerable effort has been devoted to this problem, both in research laboratories and in industry [9, 14, 15].

In Europe, this problem was addressed in the context of the VIDIMUS project [1, 3], with the aim of developing an intelligent IU environment for industrial inspection. A knowledge-based system (VDSE) was built within this environment, which can automatically configure a vision system for a given inspection problem. More recent work in which a multi-specialist architecture is used for aerial image interpretation is reported in [17]. In this system, a scene specialist carries out various strategies for scene interpretation, employing a number of semantic object specialists to detect specific types of objects such as rivers and roads.

These semantic object specialists, in turn, use a library of low-level specialists which perform specific image processing tasks such as edge detection. A global conflict specialist resolves the various conflicts that may arise between the results obtained by the semantic object specialists. The whole scheme is embedded in a blackboard architecture.

In the US, early work on this problem is reported in [2], and also in [7] in the context of a specific application (astronomical data analysis).

More recent work in the US has focussed on context-based vision, where the basic aim is to use contextual information to select methods and parameters in an IU application. Strat [11] provides a taxonomy of contextual information commonly available in an IU application. He identifies three types of context: physical, photogrammetric and computational. Physical context refers to information about the visual world that is independent of any particular set of image acquisition conditions, such as weather conditions, type of scene, etc. Photogrammetric context is information pertaining to image acquisition, such as camera location, image resolution, etc. Computational context is the internal state of the processing. These three types of context can be used to compute parameters, to guide search, to cue recognition processes, etc. Further, [11] stresses the need for explicitly encoding semantic knowledge about IU algorithms such as assumptions about their use and their inherent limitations. These ideas are implemented in the fully autonomous CONDOR system [12], as well as in the Radius HUB architecture for semi-automated image analysis [10].

## 3 A problem-solving model for algorithm integration

In this section we examine the strategies used by an IU specialist in solving a given IU problem. We are not interested in the *design* of IU algorithms—we assume that the necessary programs or libraries of IU methods are already available. We roughly classify the specialist's tasks as follows.

1. *Selection* of basic elements (programs or methods),

2. *Assembly* of basic elements (scheduling of processes),

3. *Generation* of commands or code,

4. *Execution* of programs,

5. *Monitoring* of results,

6. *Optimization* of processing.

The reasoning used by the expert in performing the above tasks can be separated into reasoning for *planning* and reasoning for *execution control.* The problem-solving model is shown echematically in Figure 1. This model gives us a way of comparing systems for semantic integration of programs.

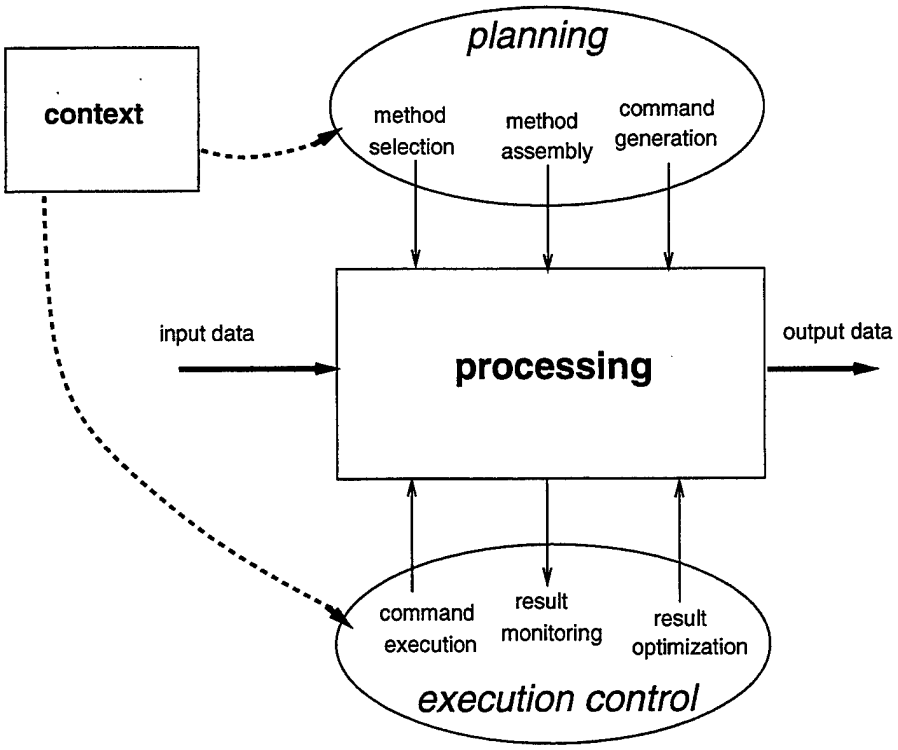



Figure 1: The basic problem-solving model for program integration

## 4 The OCAPI architecture

OCAPI is an AI environment based on the model shown in Figure 1. It uses frames and production rules as knowledge representation schemes. The main components of this archi-

tecture, and the relationships between them, are shown in Figure 2. The functions of each component are explained in this section.

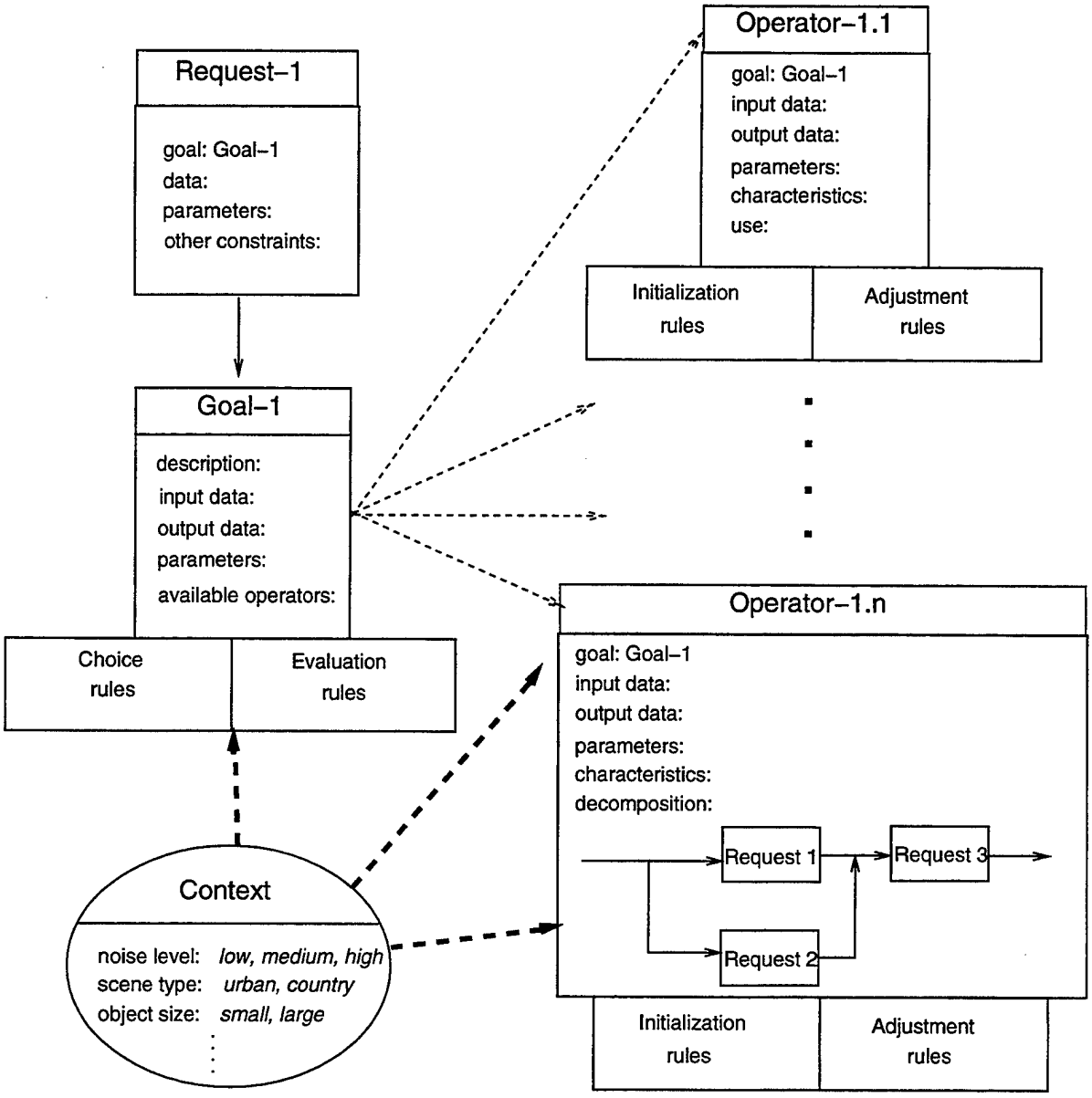



Figure 2: Relationships between the various entities in OCAPI

## Problem description

Knowledge about IU methods is expressed at two levels of abstraction. A *goal* is the abstract form of an IU functionality, which is realized in a more concrete form by one or more *operators* corresponding to it. An operator may be either a simple one, corresponding to an executable

program, or a composite one, with a *decomposition* into sub-parts. A *request* consists of the data and the type of processing to be done on them, and other constraints. All available contextual information about the problem, such as user specifications, types of sensors used, etc, is stored in the *context*. Goals, operators, requests and context are represented as frames. These frames and their inter-relationships are shown in Figure 2.

### Planning

The basic planning mechanism provided is *hierarchical script-based planning* [6]. A plan is a set of steps to attain the desired IU objective. It is similar to the block schematics often used in the design of IU algorithms, but is different in two ways:
(a) Plan hierarchy: each "box" in the plan can be a complex IU task, with its own plan,
(b) Plan abstraction: the components of the plan represent abstract IU tasks (goals), and not specific algorithms or programs. During execution, *choice rules* are used for selection of the operator best-suited for the task at hand. They are of the form
**if**

the data have property $x$

AND context field f has value $y$

AND the request has a constraint $z$

**then**

choose an operator with characteristic $b$

AND do not choose operator $w$

(The above example shows only some of the possible types of premises and actions in a choice rule. Many other types of premises and actions are possible, for choice rules as well as for the three other kinds of rules described later.)

Skeletal plans are physically stored in composite operators in the form of decomposition diagrams, consisting of one or more sub-tasks connected in parallel or in series. These sub-tasks are treated as requests (and not goals or operators) since the data flow between the sub-tasks of a composite operator is known.

### Execution control

Execution control, in this architecture, can be described as the task of executing a plan, making it work on the given data and in the given context. *Initialization rules* are used to set the initial values of the parameters. They are of the form

**if**

the data have property $x$

AND context field f has value $y$

**then**

set parameter $p$ to value $f(y)$

The performance of an IU request after execution is analyzed using *evaluation rules*, which are of the form

**if**

the output does not satisfy criterion $c$

**then**

declare that the quality $q$ of the results is unsatisfactory

declare failure of execution

The re-processing strategy consists of *adjustment rules* for operators, of the form

**if**

quality $q$ of the results was declared unsatisfactory

**then**

adjust parameter $p$ using method $m$

Planning and execution control are interleaved in the sense that choice rules may be re-applied if operator adaptation does not produce the desired results. The different kinds of rules and their relationships to the other objects in the architecture are shown in Figure 2.

### Building a knowledge base

In order to solve a given problem or set of problems the specialist's knowledge about the problem and the methods used to solve it are expressed using the mechanisms discussed in the preceding subsections.

The first step is the building of *task blocks*, which are combinations of goals and the associated operators. Corresponding to each problem type, a goal instance is created. The specialist may have one or more different ways of fulfilling a chosen goal, and for each one an operator instance is created and linked to the goal. For a simple operator, the necessary information about the program which corresponds to it is entered into its "use" field. If the operator is a composite one, with a decomposition into one or more parts, task blocks have to be created for each sub-problem type, unless they have already been created. The procedure of defining task blocks is carried out for all problems, all sub-problems, ... until each simple operator is linked to a program, and task blocks have been defined for each sub-problem of every composite operator.

The ensemble of the task blocks of a problem constitute the static part of the knowledge base. The next step is the creation of the rules which express the strategy of the specialist in executing a plan. This is a more difficult phase than the previous one, but is crucial to the success of the entire approach to problem-solving. It is difficult because, as in many other engineering disciplines, the specialist's reasoning may not often be easily expressible in the form of clear-cut rules. It should also be borne in mind that multiple rules will often be required to provide the required depth of reasoning. These inherent difficulties notwithstanding, experience has shown that it is possible to express the specialist's reasoning in a concrete fashion, provided the representation mechanisms have been correctly chosen.

The OCAPI architecture enables the systematic expression of the "rules of thumb" and the "approximate reasoning" which are crucial to successful problem-solving. This is done by means of the four types of rules discussed earlier. A mixture of numeric and symbolic reasoning may be needed for all four types of rules. Choice rules, being the easiest, are defined first. They use the values of context fields, data definitions, constraints in the request, etc. to select an operator from among the choices available. Initialization rules are defined next.

These may be somewhat more difficult, in that they have to formalize the "rough initial guesses" that the specialist makes before starting an IU task. Adjustment rules are then defined for operators which have adjustable parameters. Step sizes for parameters have to be carefully chosen so that the change in the behavior of the algorithm is neither too sudden nor too gradual. Finally, evaluation rules are defined. This is a rather difficult task since the question "Are the results good enough?" is often subjective, and appropriate quality measures may not be readily available. However, a close examination of the specialist's strategy often reveals hidden reasoning capable of being expressed in concrete terms as evaluation rules.

### Algorithm integration using OCAPI

The previous subsection outlines how to build a knowledge base for an application. Here we describe how an application problem is solved in OCAPI using this knowledge base and the available methods.

The solving of a problem starts with the creation of a request, which states the goal required, the data on which this goal is to be achieved, and the context in which the problem is being solved. (The goal should have already been defined while creating the knowledge base.) Using choice rules, the available operators for the goal are rank-ordered. The best operator is selected, and executed on the given data after the operator's initialization rules have been applied. If the operator is a simple one, this corresponds to the execution of the corresponding program. If it is a composite one, requests for its sub-tasks are created, and a tree of requests is created. Requests are chosen from this tree and executed depending on their sequencing. If the mode of execution control calls for it, the results of executing a request are judged using evaluation rules, and in the event of a failure, either the same operator is re-executed after its parameters have been adjusted using the relevant rules, or the next best operator is applied. In the event of successful execution, the next request in the tree is selected for execution. This continues, and the execution terminates when the tree of requests is empty.

## 5 Example: SAR image analysis

This application is based on the work of Kuttikkad and Chellappa [8]. Its goal is to analyze a SAR image to identify semantic objects such as targets, buildings, roads, and trees, and to segment the remaining parts of the image into various categories such as grass, water and bare ground. The first step is the detection of regions of high backscatter using a Constant False Alarm Rate (CFAR) technique [8]. In the next step non-target pixels in the image are classified as grass, tree, bare ground, road or shadow using a Maximum Likelihood (ML) approach. Training data obtained from other images of similar scenes is employed. This is a preliminary classification, using no high-level information whatsoever. A large percentage of pixels are likely to be misclassified.

Shadow regions are detected and then eroded and grown using morphological operations. The same is done for pixels classified as road. Very small regions of either class are eliminated, and the rest are grouped into homogeneous regions. Shadow regions that are adjacent to a bright streak (in the CFAR output) extending toward the sensor are classified as building shadows. Roads are verified using a shape/size criterion. ML segmentation is then repeated for pixels previously misclassified as road and shadow, this time classifying them as grass, bare ground or trees. Tree regions are grown using morphological operations, and verified using a size argument, as well as by the presence of adjoining shadow regions extending away from the sensor. In the CFAR output, streaks corresponding to buildings are eliminated, and the remaining target pixels are grouped into clusters. In the final step, ML segmentation is repeated, and based on the previous steps, pixels misclassified as shadow, road or tree are re-classified into grass or bare ground.

Examples of results obtained by the approach are shown in Figure 3.

### Knowledge base

The hierarchy of goals for the SAR application is shown in Figure 4. The knowledge base consists of the following major components: 18 goals, 19 operators (13 simple and 6 composite), and 24 production rules (2 choice rules, 2 initialization rules, 7 evaluation rules and 11 adjustment rules). This knowledge base is currently under development, and more objects

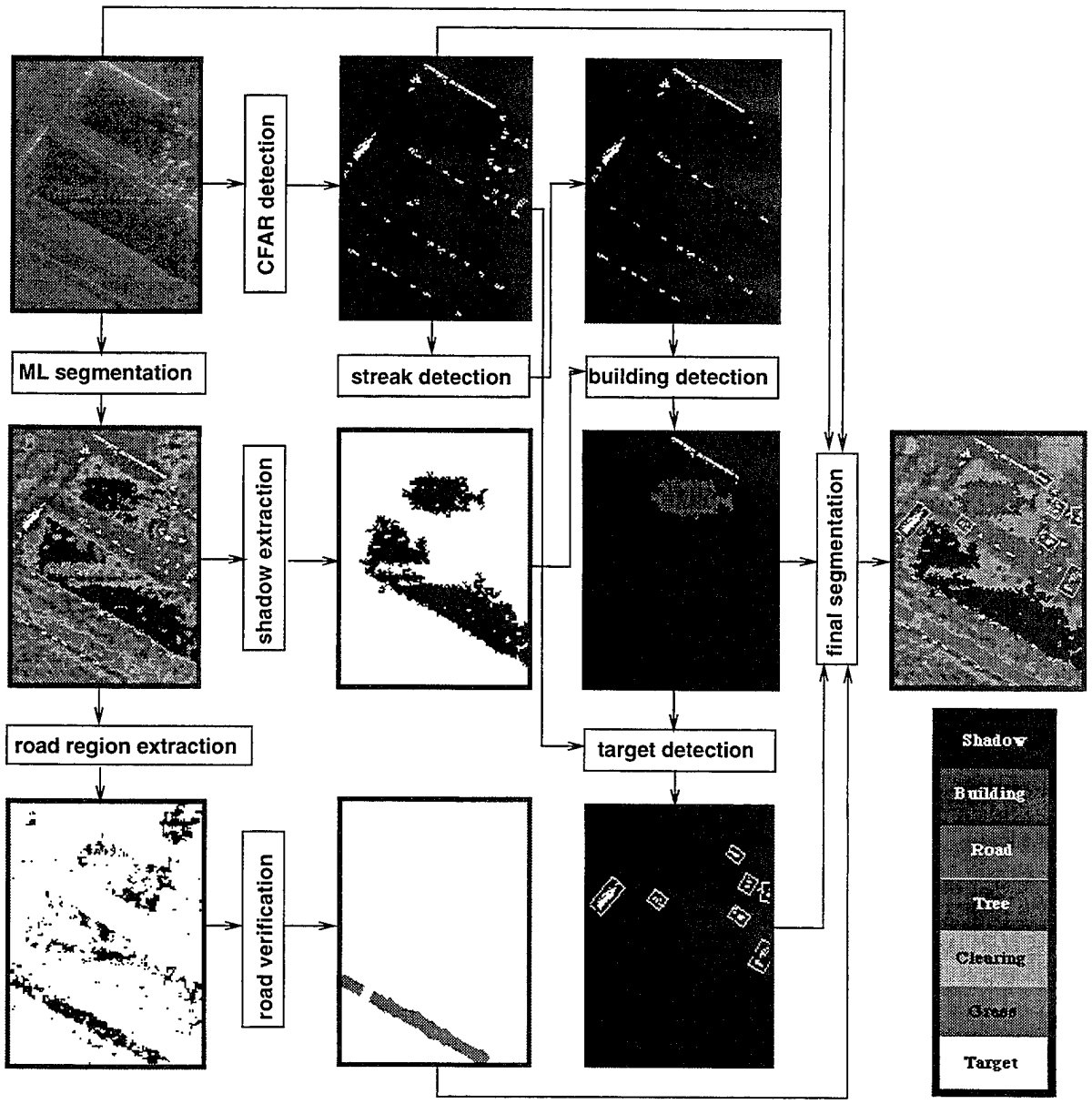



Figure 3: Examples of results of SAR image analysis.

are likely to be added to it. The complete knowledge base will not be described in detail in this paper. Instead, some simple examples from the KB are presented to give the reader a feel for the kinds of objects and reasoning involved in a real application.

The context frame for this application is shown in Figure 5.

An example of a goal is shown in Figure 6. The goal is road verification, which verifies road hypotheses obtained by pixel classification and region growing. An operator corresponding to this goal is shown in Figure 7. This is a simple operator, which has characteristics

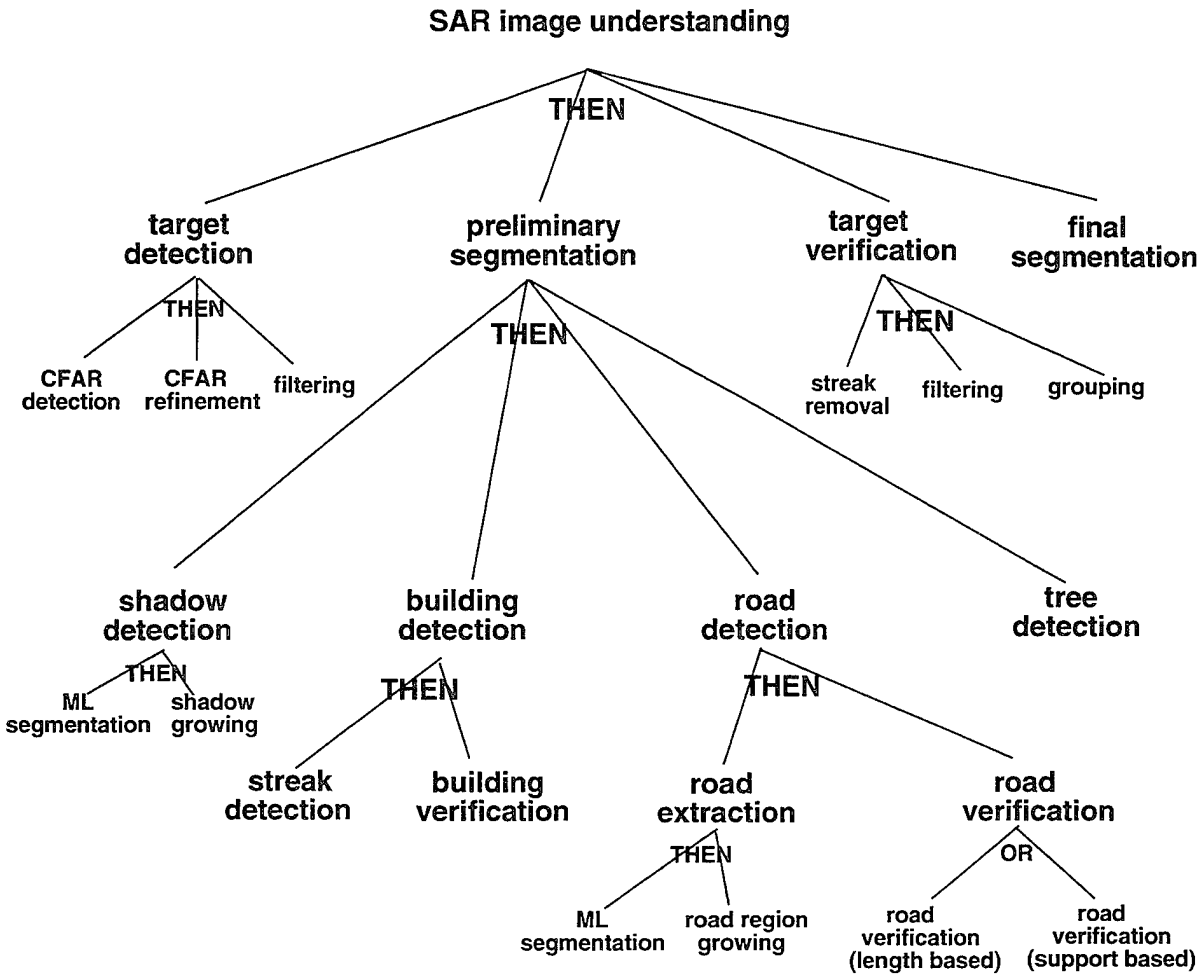



Figure 4: Goal hierarchy for SAR image analysis.

"length-based" and "fast".

There are two operators corresponding to the goal in Figure 6, and two choice rules are provided to help choose the best operator, one of which is shown in Figure 8. The reasoning is as follows: in rural areas, roads are likely to have longer unbroken stretches, whereas roads in urban areas have many intersections. Hence an operator for verifying road hypotheses in rural areas should use length as a criterion.

An example of an initialization rule for the parameter PFA (probability of false alarm) for the operator "o-cfar" is shown in Figure 9. This operator corresponds to the goal target-detection, whose function is to detect targets in the SAR image. The parameter PFA is a threshold which determines the number of bright pixels that are classified as target pixels. The higher the PFA, the more likely it is that a given pixel is classified as a target

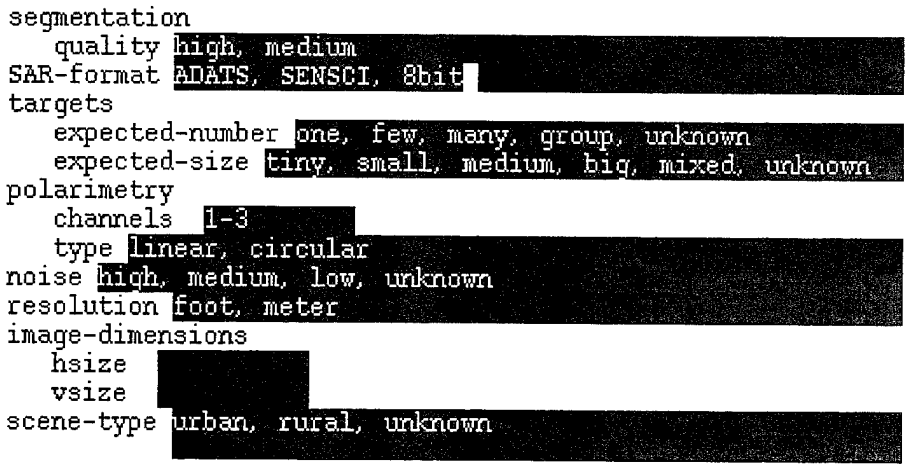



Figure 5: Context structure for SAR image analysis

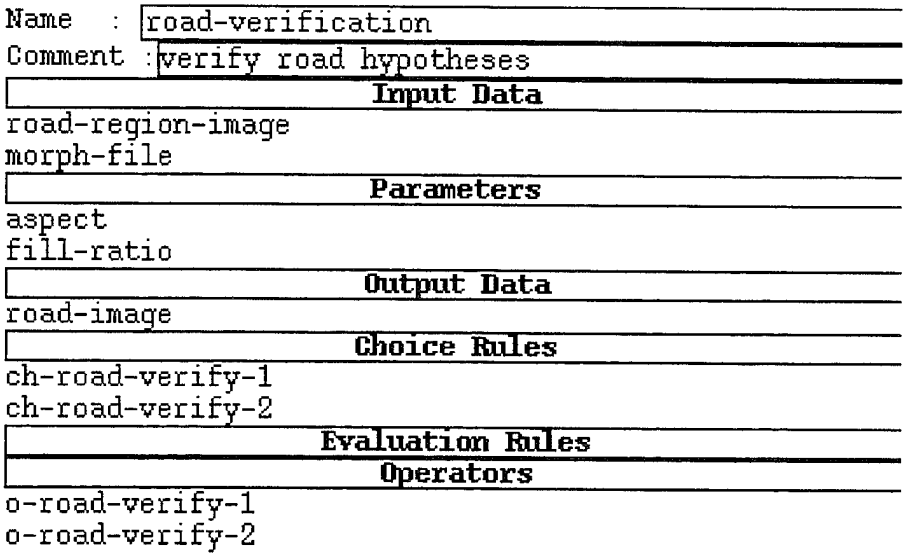



Figure 6: Example of an OCAPI goal

pixel. An example of an evaluation rule for this goal is shown in Figure 10. The user is asked to judge if the result of target detection are satisfactory. If not, appropriate action is taken via an adjustment rule, such as the one shown in Figure 11. An example of this chain of reasoning is shown in Figure 12.

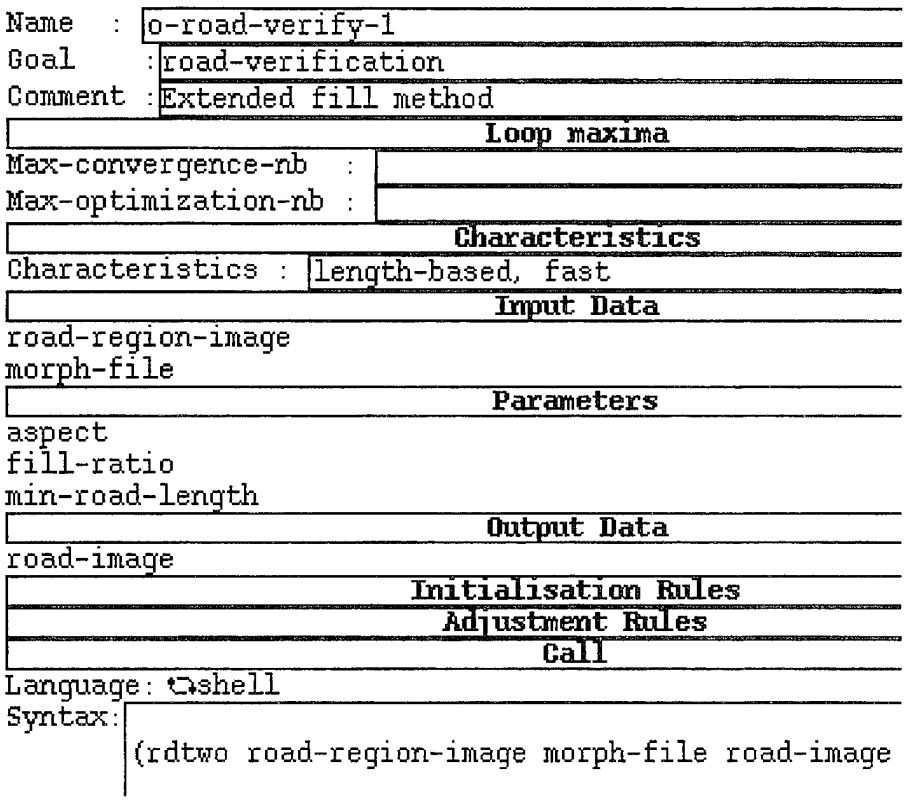



Figure 7: Example of an operator

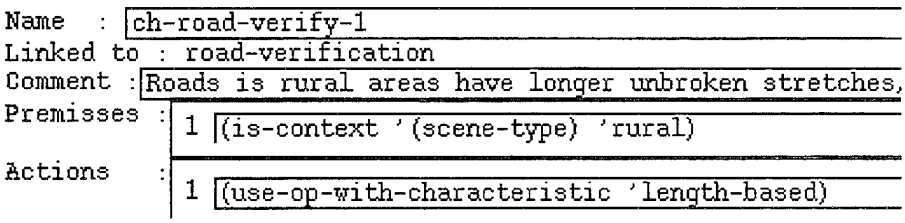



Figure 8: Example of a choice rule

## 6 Discussion and Conclusion

This paper highlights the need for knowledge-based systems for semantic integration of IU procedures, the goal of such systems being the partially or fully automatic supervision of these procedures in the context of real-life applications. One such system, OCAPI, was described along with an example of an application built using it. Other existing knowledge-based approaches were also reviewed.

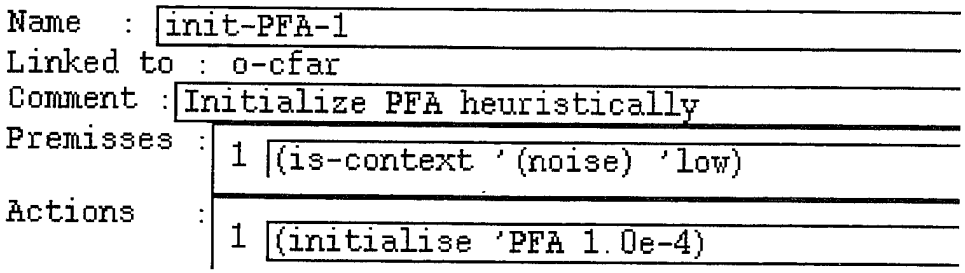



Figure 9: Example of an initialization rule

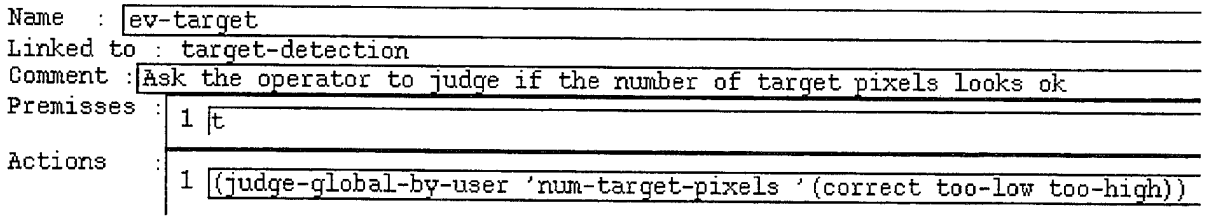



Figure 10: Example of an evaluation rule

Although these systems are a significant step forward, the problem of semantic integration is far from being solved. One major difficulty is in expressing the expertise of the specialist in the form of concrete rules and other knowledge structures. It is helpful to have an overall structure for the integration system that closely matches the nature of the problem domain. This enables the representation of expertise in a natural fashion.

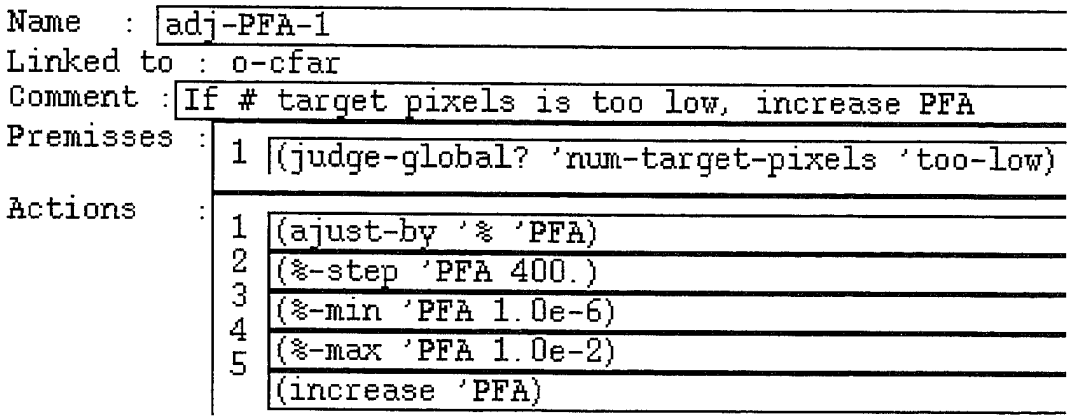



Figure 11: Example of an adjustment rule

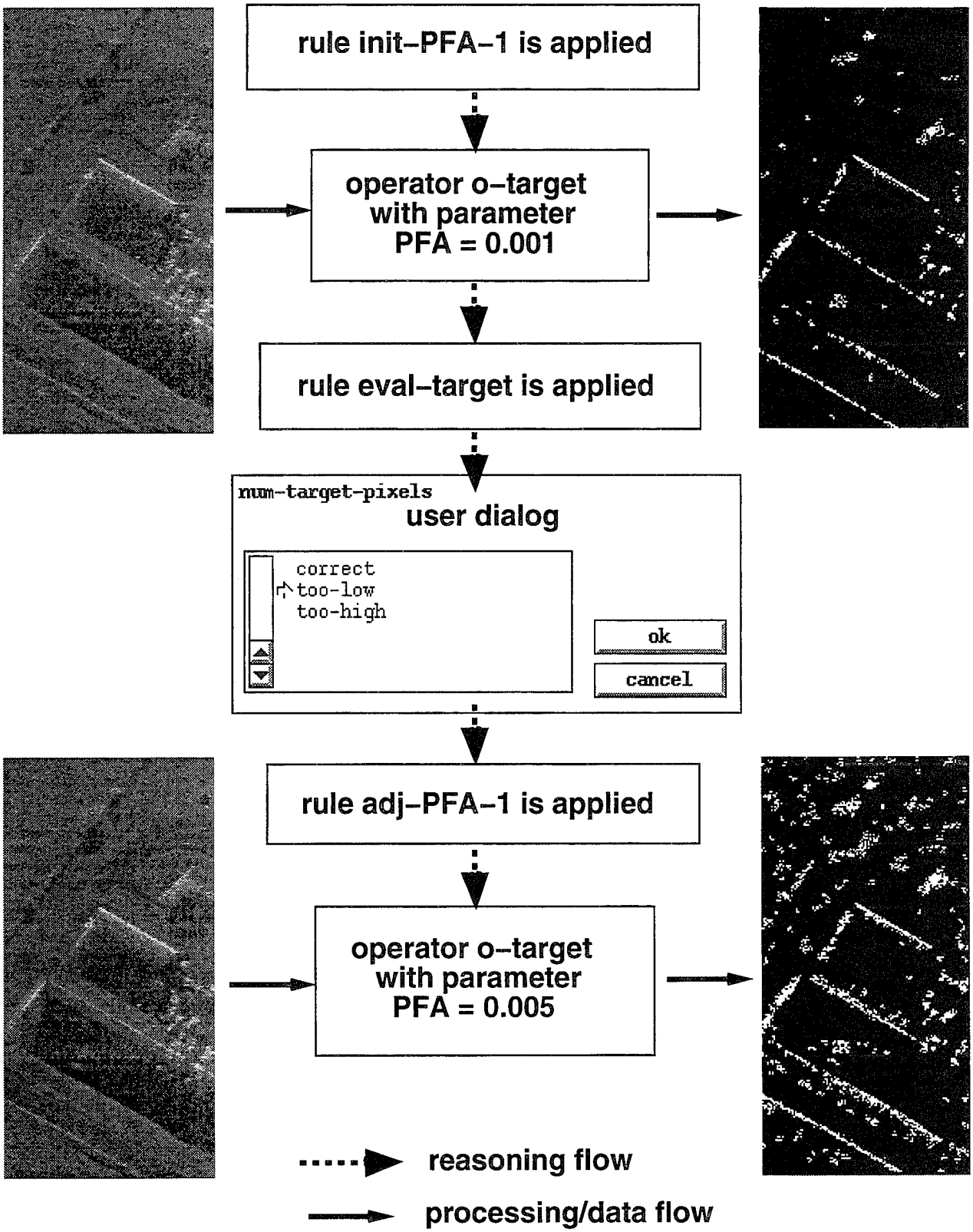



Figure 12: Example of the reasoning used for the target-detection goal

The example presented in this paper highlights various interesting aspects of the OCAPI approach. One important issue is that of reusability of the various components of a knowledge base, which is a special case of the general problem of software reuse [16]. The hierarchical structure of an OCAPI knowledge base encourages the development of modular components that can be later re-used in a different application. For instance, the **target-detection** goal and its associated operators and rules could be used independently in a completely different situation, since all the knowledge needed to use this goal is encoded within it.

As mentioned in the introduction, the ultimate goal of semantic integration is the completely automatic supervision of programs, but this goal may never be truly attained since certain stages in the processing may require visual evaluation of results by the specialist. Further, in complex domains such as IU, the results of a sequence of processing steps and the accompanying stages of automatic reasoning can never be guaranteed to satisfy the user. In such situations, a certain amount of interactive problem-solving is inevitable, but in systems such as the applications built with OCAPI an effort is made to keep this interaction at the level of the (non-specialist) user rather than at the level of the IU specialist, as illustrated by the example in Section 5.

At the heart of automatic supervision is a trial-and-error problem solving strategy, where a certain processing sequence is employed, based on all available prior knowledge about the problem and its context, and is then fine-tuned "on the fly" to produce optimal results. The performance of this trial-and-error strategy can be greatly enhanced if the system learns from its past experience. Preliminary work on incorporating a learning module in OCAPI is reported in [18].

OCAPI was developed as a general architecture for program supervision, and as such it does not provide many mechanisms for reasoning about the content of images. There are no tools for representing and reasoning about common semantic objects such as lines and regions. OCAPI, being a general-purpose system, reasons more in terms of programs and parameters than in terms of image and scene objects. This latter kind of reasoning would be useful for image analysis where typically one starts with a raw image and progressively builds a symbolic representation of its contents. A hybrid system could be imagined consisting of two knowledge-based systems, one for program supervision, and the other specifically for

high-level interpretation. This would have the advantage of separating the two types of reasoning. An example of such a system is described in [13]. This is in contrast to the approach in [17] where both types of reasoning are used in the same framework. It remains to be seen which of the two approaches is more suitable for general IU problems.

## Acknowledgments

The work discussed in this paper resulted from the efforts of many researchers. We gratefully acknowledge the contributions and assistance of Sabine Moisan, Regis Vincent, John van den Elst, and many others. We would also like to thank Hany Tolba for his helpful comments about this paper, and Dr. Les Novak of MIT Lincoln Laboratories for the SAR image used in the examples.

## References

[1] "VIDIMUS Esprit Project Annual Report," Technical Report, British Aerospace, Sowerby Research Centre, Bristol, England, 1991.

[2] D.G. Bailey, "Research on computer-assisted generation of image processing algorithms," in *Proceedings, IAPR Workshop on Computer Vision—Special Hardware and Industrial Applications* (Tokyo, Japan), Oct. 1988.

[3] R. Bodington, "A software environment for the automatic configuration of inspection systems," in *First International Workshop on Knowledge-Based Systems for the (re)Use of Program Libraries* (Sophia Antipolis France), Nov. 1995.

[4] C.C. McConnell and D.T. Lawton, "IU software environments," in *Proceedings of the DARPA Image Understanding Workshop*, pp. 666–676, April 1988.

[5] V. Clément and M. Thonnat, "A knowledge-based approach to the integration of image processing procedures," *CVGIP: Image Understanding*, Vol. 57, pp. 166–184, 1993.

[6] J. Hendler, A. Tate, and M. Drummond, "AI planning: Systems and techniques," *AI Magazine*, Vol. 11, No. 2, pp. 61–77, 1990.

[7] M. Johnston, "An expert system approach to astronomical data analysis," in *Proceedings, Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, pp. 1–17, 1987.

[8] S. Kuttikkad and R. Chellappa, "Building wide area 2D site models from high resolution polarimetric synthetic aperture radar images," Technical Report CAR-TR-776, Computer Vision Laboratory, University of Maryland, College Park, MD, June 1995.

[9] H. Sato, Y. Kitamura, and H. Tamura, "A knowledge-based approach to vision algorithm design for industrial parts feeder," in *Proceedings, IAPR Workshop on Computer Vision, Special Hardware and Industrial Applications* (Tokyo, Japan), pp. 413–416, Oct. 1988.

[10] T. Strat, "Integrating IU algorithms into the RADIUS HUB." Software documentation.

[11] T. Strat, "Employing contextual information in computer vision," in *Proceedings of the DARPA Image Understanding Workshop* (Washington, DC), April 1993.

[12] T. Strat and M.A. Fischler, "Context-based vision: Recognizing objects using both 2D and 3D imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 1050–1065, 1991.

[13] M. Thonnat, V. Clément, and J.C. Ossola, "Automatic galaxy description," *Astrophysical Letters and Communication*, Vol. 31, 1995.

[14] T. Matsuyama, "Expert systems for image processing: Knowledged-based composition of image analysis processes," *Computer Vision, Graphics, Image Processing*, Vol. 48, pp. 22–49, 1989.

[15] T. Toriu, H. Iwase, and M. Yoshida, "An expert system for image processing," *Fujitsu Sci. Tech. Journal*, Vol. 23.2, pp. 111–118, 1987.

[16] J. van den Elst, F. van Harmelen, and M. Thonnat, "Modeling software components for reuse," in *Seventh International Conference on Software Engineering and Knowledge Engineering* (Rockville, MD), pp. 350–357, June 1995.

[17] V. Clément, G. Giraudon, S. Houzelle, and F. Sandakly, "Interpretation of remotely sensed images in the context of multi-sensor fusion using a multispecialist architecture," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 31, pp. 779–791, July 1993.

[18] R. Vincent, S. Moisan, and M. Thonnat, "Learning as a means to refine a knowledge-based system," in *Proceedings of the Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop* (Hatoyama, Japan), pp. 17–31, Nov. 1994.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | December 1995 | Technical Report |

**4. TITLE AND SUBTITLE**

Knowledge-Based Integration of IU Algorithms

**5. FUNDING NUMBERS**

N00014-95-1-0521

**6. AUTHOR(S)**

Chandra Shekhar, Shyam Kuttikkad, Rama Chellappa, and Monique Thonnat

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

**8. PERFORMING ORGANIZATION REPORT NUMBER**

CAR-TR-804
CS-TR-3578

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release.
Distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

This paper deals with the integration of image understanding (IU) programs using a knowledge-based approach. The basic concepts of program integration are discussed, and a simple problem-solving model for program integration is outlined. Two types of reasoning, planning and execution control, are identified. A system developed using this model, called OCAPI (Optimizing, Controlling and Automating the Processing of Images), is introduced. OCAPI is in an AI environment in which the reasoning used by the IU specialist is formally represented using frames and production rules. An example of an application developed using OCAPI is presented, and the advantages and shortcomings of this approach are discussed.

**14. SUBJECT TERMS**

Image understanding, program integration, planning, execution control, OCAPI, SAR imagery

**15. NUMBER OF PAGES**

24

**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 |