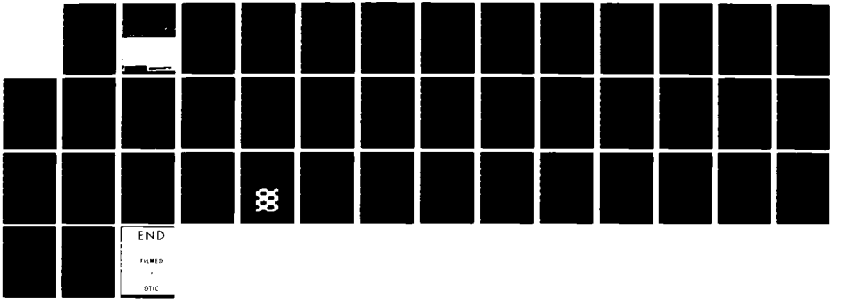
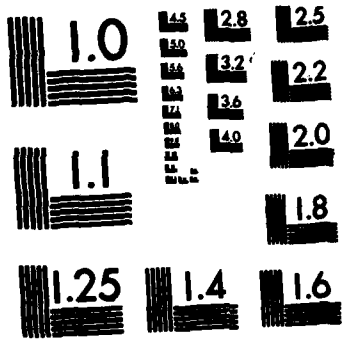
AD-A124 054 MULTILEVEL RELAXATION IN LOW LEVEL COMPUTER VISION(U) 1/1
MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND
INFORMATION SCIENCE F GLAZER 1982 COINS-TR-82-30
UNCLASSIFIED N00014-82-K-0464 F/G 9/2 NL





(12)

Multilevel Relaxation in Low Level Computer Vision[1]

Frank Glazer[2]

COINS Technical Report 82-30

Computer and Information Science Departmen
University of Massachusetts
Amherst, Massachusetts 01003

Computer and Information Science

Multilevel Relaxation in Low Level Computer Vision[1]

Frank Glazer[2]

COINS Technical Report 82-30

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

Variational (cost minimization) and local constraint approaches are generally applicable to problems in low-level vision (e.g., computation of intrinsic images). Iterative relaxation algorithms are "natural" choices for implementation because they can be executed on highly parallel and locally connected processors. They may, however, require a very large number of iterations to attain convergence. Multi-level relaxation techniques converge much faster and are well suited to processing in cones or pyramids. These techniques are applied to the problem of computing optic flow from dynamic images.

---

1. This paper will appear in Multiresolution Image Processing and Analysis, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

2. This research was supported in part by the National Science Foundation under Grant MCS79-18209 and by DARPA under Grant N00014-82-K-0464.

TABLE OF CONTENTS

LIST OF FIGURES

0.0 INTRODUCTION

Much work in low level computer vision has involved the dense interpolation or approximation of sparsely-known or noisy data. A few examples are image smoothing [NOR82], surface interpolation [GR81,I80,IH81], and optic flow computation [HS81]. A recent approach to these problems has formulated them in terms of optimization or constrained minimization. In general these techniques are equivalent to solving elliptic partial differential equations (PDE's) with boundary conditions and constraints.

In either formulation, these problems can be solved by a class of algorithms well suited to computer vision. It includes the Gauss-Seidel iterative method [HY81] and assorted variants. These methods are local, parallel, and distributed, attributes which make them ideal for implementation on locally connected parallel processors. They have one attribute, however, that currently limits their applicability. The number of iterations required for convergence is often very high - on the order of O(d**n), where 'd' is the distance (in nodes) that information has to travel and 'n' is the order of the PDE's being solved [B77b,p.281].

In the problem domain of elliptic PDE's this slowness has been overcome by using multi-level relaxation algorithms [B77a,B77b]. In this approach, a standard iterative algorithm is applied on grids of different resolution all of which cover the data in registration. At each level the problem is solved in a different spatial bandwidth. Thus, the various processing levels

cooperate to compute the final result, which is represented at the highest resolution level. The number of iterations required is of order O(d) .

Given the value of multi-level relaxation to solving elliptic PDE's with boundary conditions, we are interested in the extension of these techniques to problems of optimization, constrained minimization, and PDE's with obstruction (points through which the solution must pass). Our work in areas such as computing optic flow, interpolating sparse displacement data (for motion analysis and geometric correction), and object surface reconstruction has led us to study how multi-level methods can be applied to these more difficult problems. We will present a discussion of this along with preliminary results in the specific problem domain of computing optic flow.

Regular hierarchical structures - such as multi-level grids of varying levels of resolution - are established computational architectures in computer vision [T78,TK80]. These have included processing architectures [HR80], data structures [TP75,KD76], and algorithms [K71,WH78,MP79,BURT82]. They are founded on cellular array architectures which provide high-speed efficient processing for image-oriented operations. Beyond this, they add multiple levels of spatial resolution [HR80,TP75,WH78,MP79,BURT82] and selective attention mechanisms [K71,KD76,TP75].

Hanson and Riseman's processing cone [HR80] is a parallel array computer hierarchically organized into layers of decreasing spatial resolution. It is designed to provide parallel processing power both locally (fine resolution) and globally to

varying degrees (coarse resolutions). Tanimoto and Pavlidis' pyramids [TP75] are sequences of digitizations of the same image at increasingly higher degrees of spatial resolution. Klinger and Dyer's regular decompositions [KD76] (later, quad trees) divide a picture area into successively smaller quadrants. In these last two cases, the data structures were used to provide a coarse to fine focus of attention which significantly speeded up recognition algorithms such as edge finding and object detection. This idea had been used earlier in Kelly's edge detection algorithm [K71], which used edges in a coarsened image as a "plan" in locating edges in higher resolution versions of the same image. Wong and Hall [WH78] did scene matching by correlation in a hierarchy of reduced resolution images. Matches found in the coarse resolution images constrained the search for matches in finer resolutions. This algorithm matched scenes which were difficult for standard correlation techniques and did so with far fewer computations. Marr and Poggio [MP79] presented another hierarchical matching algorithm as a theory of human stereo vision based on the matching of edges between a pair of images. Edges detected in coarse resolution channels are matched, and the matches are used to control eye vergence movements, thus allowing finer resolution channels to match unambiguous edge pairs. Finally, Burt [BURT82] has developed pyramid-based algorithms for multi-resolution low-pass and band-pass filtering which are low in both cost and complexity. He has shown how these techniques can be applied to feature extraction, image compression, image merging, scene matching and motion detection.

1.0 A COMMON GROUND

In this section we describe a number of problems, approaches, and algorithms, covering a range of recent work in low level computer vision. These methods are not entirely analogous in their development; hence a single, general formulation is not attempted. Instead, we try to provide a unifying framework within which various methods can be viewed. To motivate this framework we first present some specific examples of recent work.

1.1 Four examples

Narayanan et.al. [NOR82] approached the problem of smoothing noisy images in the following way. Given a noisy image F(x,y) find an approximation G(x,y) which minimizes the total error measure

$$\underset{x,y}{\text{SUM}} \; [\; R^2 \; + \; a * (F - G)^2 \;] \qquad (1.1)$$

where R(x,y) is a roughness measure of G, computed for a neighborhood of each point and 'a' is a relative weighting factor. The first term penalizes roughness of G, while the second term penalizes deviation of the approximation G from the data F. The roughness measures used were discrete approximations to 1) the Laplacian, 2) the gradient magnitude, and 3) a measure of strong corners (viz. $G_{xx} * G_{yy} - G_{xy}^2$). A steepest descent

---

1. See L73 for a general discussion of optimization methods.

In his theory of visual surface interpolation, Grimson formulated the problem as one of finding surfaces S(x,y) having minimum total curvature and passing through (interpolation) or near (approximation) points of known depth [GR81]. Approximation was formulated as minimization of the functional

$$\int\int (S_{xx}^2 + 2S_{xy}^2 + S_{yy}^2)dxdy \quad + \quad B*\underset{P}{SUM}\ [S(x,y) - C(x,y)]^2 \qquad (1.2)$$

where P is the set of points with known depth C(x,y). The discrete version of this minimization problem was solved using the conjugate gradient algorithm [L73]. Interpolation was formulated as constrained minimization of the functional

$$\int\int (S_{xx}^2 + 2S_{xy}^2 + S_{yy}^2)\ dxdy \qquad (1.3)$$

constrained to S(x,y) = C(x,y) in P. The discrete version of this constrained minimization problem was solved using the gradient projection algorithm [L73].

Ikeuchi approached the problem of shape from shading as follows [I80,IH81]. Given a brightness image I(x,y), a reflectance map R(f,g), and an occluding contour dA, find the surface orientation (F(x,y),G(x,y)) in A which minimizes the functional

$$\underset{x,y}{SUM}\ [I - R(F,G)]^2 + a*[(\nabla F)^2 + (\nabla G)^2] \qquad (1.4)$$

where ∇ is a discrete gradient. The first term of the functional penalizes deviation from the image irradiance/reflectance equation I(x,y) = R(f,g). The second term penalizes roughness of F and G. This minimization problem was solved by setting the gradient of the functional to 0

and using Gauss-Seidel iteration to solve the resulting system of linear equations.

Finally we consider Horn and Schunck's method of determining optic flow given a dynamic image E(x,y,t) [HS81]. Their approach is to find the optic flow vector field (U(x,y),V(x,y)) which minimizes the functional

$$\int\int (E_x U + E_y V + E_t)^2 + a * [|\nabla U|^2 + |\nabla V|^2] \; dxdy \quad (1.5)$$

The first term in the functional is the square of the rate of change of image brightness (measured in 3D space-time of the dymnamic image space). When $E_x U + E_y V + E_t = 0$, spatial image brightness changes are due purely to motion. The second term is a roughness measure of U and V, where ∇ is the gradient operator and 'a' is a relative weighting factor. The Euler equations were derived for this problem and a finite difference approximation produced a system of linear equations that was solved using Gauss-Seidel iteration. This particular example is discussed in depth in section 3.

1.2 The General Framework

We will now consider a framework of that makes plain the similarities and differences among the methods described above. Figure 1 diagrams the major relationships. Each box is labeled by a general technique or method. Within each box a specific example is shown. The examples refer to the same problem in each case, the solution of Laplace's equation or its variational equivalent.

The Continuous Case.

Variational calculus and elliptic partial differential equations (EPDE's) represent the problem in a continuous domain. In general a variational problem can be reduced to a partial differential equation given by Euler's equation [CH53,p.183] (Figure 1A -> Figure 1B). These PDE's are elliptic in all of the cases we consider. Moreover, for any EPDE a corresponding variational problem can be constructed. Thus we are free to start our analysis on either side in the top row of Figure 1 . Both Grimson and Horn & Schunck began their analyses with variational problem statements.

Discrete Approximations.

The introduction of finite difference approximations in place of continuous derivatives transforms both variational problems and EPDE's into discrete problems as shown in the middle row of Figure 1. The integral in a variational problem becomes a summation over a discrete set of grid points (Figure 1A -> Figure 1C). In Figure 1C this gives a summation, over the 2-D grid indices i & j, of first finite difference approximations of the horizontal and vertical partial derivatives.

In Figure 1D we see that the EPDE generates a separate equation for each grid point relating that point to its immediate neighbors. Δ_{ij} refers to a discrete Laplacian on the i,j grid. The bottom line in that box expresses this set of equations as a 2-D convolution with the 5-point Laplacian mask. This system of equations can also be generated from the discrete functional by

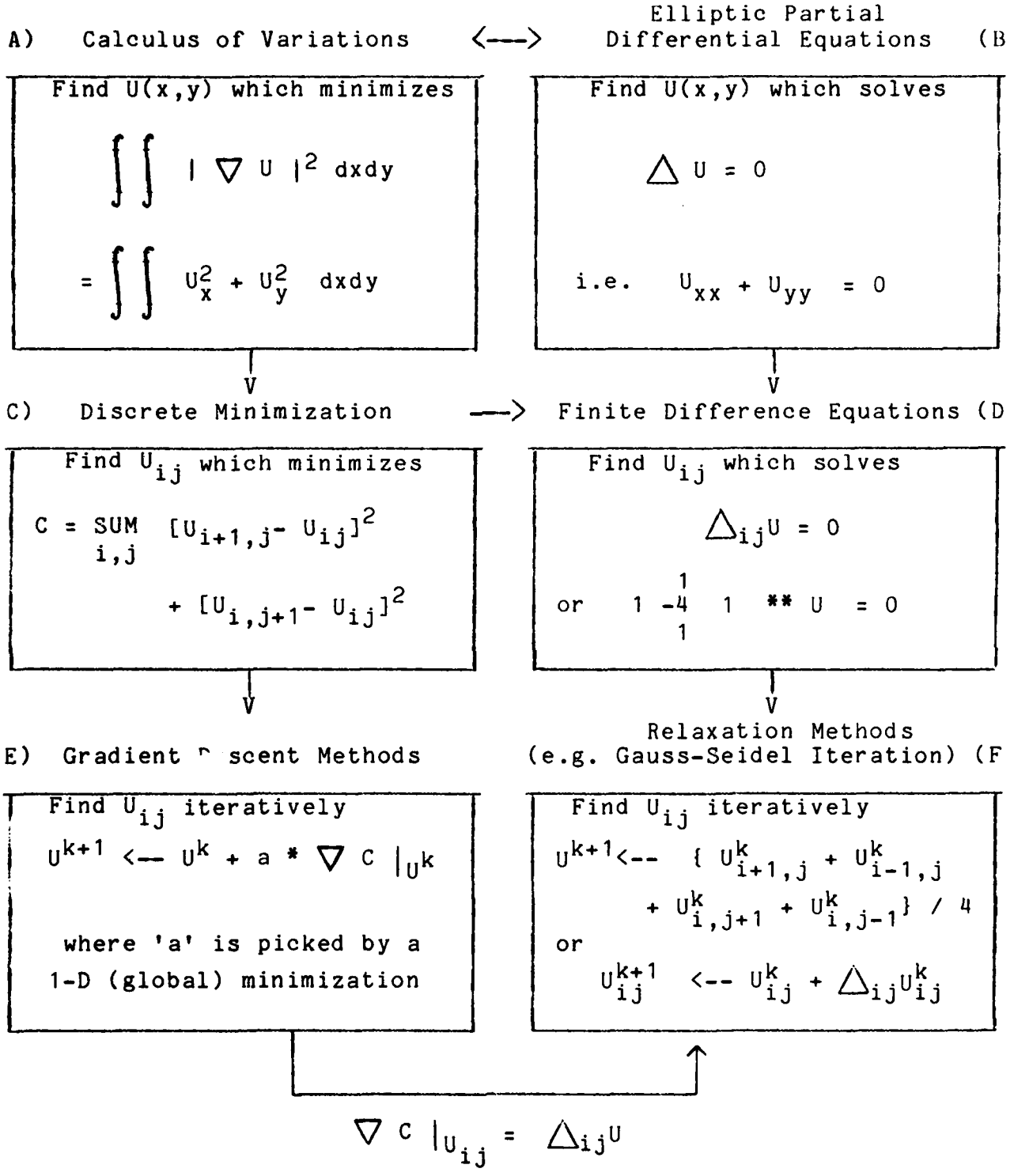


Figure 1: General framework of approaches and algorithms.

A,B) Continuous problem representations
C,D) Discrete approximations to continuous problems
E,F) Iterative algorithms for solving discrete problems

computing its gradient (considered as a function of all the grid point variables) and setting it equal to the 0-vector - this being a necessary condition for the existence of a minimum. This operation takes us from Figure 1C to Figure 1D. It is analogous to using the first variation of the continuous functional to derive the corresponding PDE. Ikeuchi takes this path.

Algorithms.

Finally, we come to specific algorithms for solving these discrete problems. Consider first the system of finite difference equations in Figure 1D. This system is a large matrix equation Ax = b, where the vector 'x' contains all of the grid point variables, 'A' is a sparse banded matrix containing the finite difference coefficients, and 'b' is a vector of right-hand sides which can include forcing terms, boundary conditions, or obstructions. Although many methods exist for solving such a system of equations, the requirements of low level vision problems quickly narrow our choice. In particular, iterative relaxation algorithms, being local and parallel, are natural candidates given the computational constraints of a vision system. Both Ikeuchi and Horn & Schunck use Gauss-Seidel iteration to solve their equations.

Now consider minimization of the discrete functional in Figure 1C. This is a problem that can be solved using the techniques of mathematical programming (also known as non-linear programming and cost minimization) [L73]. These methods use an iterative scheme involving a two step cycle; 1) pick a "direction" to search for the minimum, and 2) perform a one

dimensional search along that direction. Two common methods for choosing the search direction are the steepest descent (gradient) method and the conjugate gradient method. The former method is used by Narayanan et.al. while Grimson uses the latter. Steepest descent is shown in the box of Figure 1E where the current iteration U^k is updated by an optimal multiple of the gradient of C evaluated at U^k.

An interesting parallel can now be seen between gradient descent and relaxation methods. As the arrow joining the two lower boxes in Figure 1 indicates, computing the gradient of the cost functional is essentially equivalent to two dimensional convolution with the discrete finite difference operator (up to a scale factor). The only difference we see in the two methods is the application of a one dimensional minimization procedure along the direction of the gradient in the gradient descent method. When the functional is quadratic this minimization can be computed directly [GR81,NOR82]. Unfortunately this computation is a global one in that it uses the full current solution estimate and gradient images. In contrast, relaxation methods remain strictly local. The use of a global factor does provide faster convergence.

The Finite Element Method.

An alternative path from variational problems to relaxation on linear systems of equations is given by the finite element method. A good example of this can be found in Terzopolous' extension of Grimson's work [T82]. While not equivalent to finite difference methods, FEMs do produce similar systems of

equations which can be solved with relaxation methods, thus providing an alternate path from boxes (A) to (D) in Figure 1 .

1.3 An Aside on Constraints

We must distinguish between three types of constraints that determine a problem - boundary conditions, obstructions, and inherent constraints.[2]

Inherent constraints are those that are represented in the form of the variational problem or PDE. All of the problems we have discussed involve an inherent constraint of smoothness of the solution. The order of this smoothness can be defined as the order of the partial derivatives in the functional. First order smoothness typically uses the gradient magnitude as in [NOR82, IH81, and HS81] and leads to second order EPDEs (e.g. Laplace's Eq.). Second order smoothness constraints are seen in Grimson's functionals where the corresponding EPDE is the biharmonic equation

$$\Delta^2 U = U_{xxxx} + 2U_{xxyy} + U_{yyyy} = 0$$

Smoothness conditions constrain the solution in local neighborhoods. The other case of inherent constraint we see is a pointwise one derived from some relationship that must hold (as well as possible) between the solution surface and the data. Narayanan's formulation requires a solution near the initial data (a relationship of equality). Ikeuchi's requires surface orientation and image intensity to relate according to the

---

2. A fourth type - the isoperimetric condition [CH53] - is not listed here since it does not typically occur in the class of problems we are considering.

reflectance map. Horn & Schunck require optic flow vectors to lie near the velocity constraint line in velocity space.

The second type of constraint is the boundary condition. These conditions constrain the solution along the boundary of the domain within which a solution is to be found. Typical conditions include the vanishing of the function and/or its derivative normal to the boundary (the Dirichlet and Neumann conditions). Boundary conditions are determined both by explicit specification (essential boundary conditions) and as an implicit consequence of the particular variational problem or PDE (natural boundary conditions).

It may seem that boundary conditions are an unfortunate complication in low-level vision problems since processing should be independent of the location of the retinal boundary. We would argue that this is too limiting an idea of where boundaries occur. They should be viewed as demarcating regions within which the inherent constraints are to hold. This can be seen in Ikeuchi's work where boundaries were placed at the occluding contours in an image. What if the occluding contours are not known in advance? We are then faced with a key problem that remains to be addressed in our framework. Consider the smoothness constraints used by Grimson and Horn & Schunck. Dependent as they are on the continuity of objects (which holds almost everywhere in the image), these conditions break down across occluding boundaries. Enforcing smoothness across these boundaries corrupts the solution well into the adjacent regions. It remains to be seen whether these conditions can be detected

and whether a general approach to this problem exists.

The final type of constraint is the obstruction or obstacle. We see a specific case of this in Grimson's interpolation where the smooth surface is constrained to pass through points of known depth. Generally, such a constraint is given by a) a subdomain A_1 of the full domain A, b) a function C(x,y) over the subdomain A_1, and c) an inequality between the solution S(x,y) and C(x,y) that must hold within A_1. Another example of this is Yachida's use of exact optic flow estimates derived from prominent feature points in his variant of the Horn & Schunck method [Y81]. These known vectors were used as "fixed points" or seeds that did not change value during the course of iteration.

2.0 MULTILEVEL RELAXATION

We have seen how various problems in low level computer vision can be formulated as continuous or discrete variational problems and equivalently as partial differential or finite difference equations. Iterative relaxation algorithms are then "natural" choices for solving these problems since they are local, parallel, and distributed. Unfortunately the number of iterations necessary for convergence can be very high - on the order of O(d**n), where d is the distance (in nodes) that information has to travel and n is the order of the PDE's being solved. Grimson's algorithm, requiring second order smoothness, takes thousands of iterations to approach final solutions.[1] This slowness is due to the fact that solutions which must satisfy a global condition (the variational problem) are arrived at by the local propagation of information.

Multi-level relaxation is an algorithmic extension of iterative relaxation designed to overcome asymptotically slow convergence. By representing the spatial domain at multiple levels of resolution (in registration) these algorithms apply the basic local iterative update to a range of neighborhood sizes. Local updates on coarser grids introduce a more global propagation of information.

In the basic multi-level method the domain of definition A of the problem is represented discretely by a set of uniform square grids $G^0,\ldots,G^k,\ldots,G^M$ with grid sizes $h_0,\ldots,h_k,\ldots,h_M$.

---

1. This number is based on the author's experiments. See also T82.

Each grid covers the full domain and we assume that $h_i:h_{i+1} = 2:1$. Suppose we have a PDE we wish to solve over A, say

$$\triangle U(x,y) = 0 \quad \text{or more generally} \quad LU(\bar{x}) = F(\bar{x}) \qquad (2.1)$$

where L is a general linear differential operator and U and F are vector valued functions over R^n.

On each grid G^k we can form the finite difference approximation to these PDEs;

$$\triangle_k U_{ij} = 0 \quad \text{or} \quad L^k U^k(\bar{x}) = F(\bar{x}) \quad k = 0,\ldots,M \qquad (2.2)$$

The discrete approximation on a coarse grid G^k approximates the same problem on the finest grid G^M. This fact was used long ago by engineers in the block relaxation method. In this method the solution on a coarse grid is used as the initial estimate for the finest resolution problem. Such an approach can be applied at multiple levels of resolution.

Brandt [B77a,B77b] significantly extended the multi-level approach by showing how the coarser grid can be used in the process of improving the first approximation on G^M. He combined these ideas and developed iterative schemes that move up and down a cone of multiple grids.

If we look a little more closely at the nature of convergence in an iterative relaxation algorithm, we see that sharp (high frequency) changes or errors are quickly smoothed. This is due to the local nature of the smoothing. It is the slow (low frequency) error components that resist elimination. These ideas can be formalized in a local mode analysis of the particular update equation [B77a,B77b] whereby the error

reduction is considered as a function of spatial frequency. The theory of multi-level relaxation is based on these ideas and on the fact that one grid's coarse resolution is another grid's fine. After a few iterations on a given grid, high frequency error is considerably reduced while low frequency error remains. At this point we can approximate the remaining problem on a coarser grid. The remaining error is a higher frequency on the coarse grid, hence further relaxation at this level can reduce it effectively. When convergence is attained at the coarse level, that solution can be interpolated back to the fine level. This interpolation introduces some high frequency error which is easily reduced by a few more iterations. These processes easily generalize to multi-level cyclic algorithms running on a cone of grids. Approximate solutions are sent down the cone to finer levels after they have converged. When convergence slows at finer levels they are sent up the cone for coarser processing. The role of relaxation in such a system is not to reduce the error, but to smooth it out; i.e. to reduce high frequency components of the error. Lower frequencies are reduced on coarser grids. What is essentially happening in such a system is that different grid levels solve the problem in different spatial frequency bands.[2]

Following Brandt's development [B77a],[3] let u^M be an

---

2. This particular idea breaks down on non-linear problems. However, the multilevel approach does generalize. All of the problems we are considering are linear.

3. We will only show how the PDE approximation is handled. The equations (& algorithm) for the boundary conditions are handled in a similar manner.

approximate solution of the G^M problem and let

$$L^M u^M = F^M - f^M \tag{2.3}$$

where the discrepancy f^M is called the residual. Assuming L is a linear operator, the exact discrete solution is $U^M = u^M + V^M$, where the correction V^M satisfies the residual equation $L^M V^M = f^M$. If u^M is computed by some relaxation iterations on G^M then f^M has little high frequency content (relative to the grid size h_M). This allows the residual equation to be accurately approximated on a coarser grid. The optimal time to perform this switch to a coarser grid occurs when the residual f^M is smoothed out and convergence has slowed down. Relaxation on the coarse grid produces an approximation v^k of the correction V^M. An improved level M solution is then obtained by interpolating v^k to level M and adding this interpolated correction to u^M.

A simple multilevel relaxation algorithm based on these ideas called Cycle C [B77a] is shown in Figure 2. In this notation, I^k_{k-1} interpolates level k-1 data down to level k (coarse to fine), while I^k_{k+1} interpolates level k+1 data up to level k (fine to coarse). The basic rule in Cycle C is that each v^k (the function defined on the grid G^k; $k = 0,\ldots,M-1$) is designed to serve as a correction for the approximate v^{k+1} previously obtained on the next finer grid G^{k+1}. The equation to be (approximately) satisfied by v^k is

$$L^k v^k = f^k \tag{2.4}$$

where f^k approximates the residual left by v^{k+1}, that is

$$f^k = I^k_{k+1}(f^{k+1} - L^{k+1}v^{k+1}) \tag{2.5}$$

The equation on G^k is thus defined in terms of the approximate

```
k <-- M                                  ; start at finest level
f^k <-- F^M                              ; initial RHS
v^k <-- u^M                              ; initial solution estimate
Until v^M has converged Do
  Begin
    v^k <-- Relax [ L^k . = f^k] v^k     ; a relaxation "sweep"
    If v^k has converged Then
      If k < M Then
        k <-- k + 1                      ; go down one level
        v^k <-- v^k + I^k_{k-1} v^{k-1}  ;add interpolated correction
    Else if convergence is slow Then
      If k > 0 Then
        k <-- k - 1                      ; go up one level
        v^k <-- 0                        ; initial estimate
        f^k <-- I^k_{k+1}(f^{k+1} - L^{k+1} v^{k+1})   ; new RHS
    End if
  End
```

Figure 2: Cycle C, multilevel relaxation.

---

solution on G^{k+1}. On the finest grid, the equation is the original one;

$$f^M = F^M \tag{2.6}$$

Convergence is measured using the Euclidean (L_2) norm of the residual function which we will call the residual norm. The rate of convergence is measured as the ratio of consecutive residual norms from one iteration to the next. Convergence is "slow" when this ratio rises above a threshold parameter (0.6 in our experiments and in B77a). Convergence at the finest level is defined by a user supplied tolerance (threshold) below which the residual norm must fall. Convergence at an intermediate level is

defined by a dynamic threshold. This coarse level threshold is set, when we pass up the cone a level, to a fraction of the current residual norm at the fine level. This fraction is another parameter in the algorithm (0.3 in our experiments and in B77a). Brandt claims robustness of such algorithms to the extent that variations of these two parameter settings produce little qualitative change in performance.

The two parameterized decisions that are used in controlling the cycles are based on global computations, i.e. they are computed from the current solution estimate over the entire grid. It will be seen later - as Brandt has pointed out - that for some problems we can forego the computation of the residual norm and thus attain a purely local and parallel computation.

In the basic Cycle C algorithm an approximate solution only exists on the fine level grid. All coarser levels deal only in correction surfaces which approximate solutions for changing residual equations. Brandt also developed FAS (Full Approximation Storage) algorithms in which each grid level stores the full current approximation. This approximation u^k is the sum of the correction v^k and its base approximation u^{k+1};

$$u^k = I^k_{k+1} u^{k+1} + v^k \qquad (k = 0,1,\ldots M-1) \qquad (2.7)$$

Using these full-approximation functions, the correction equations can now be rewritten as

$$L^k U^k = \bar{F}^k \qquad (2.8)$$

where

$$\bar{F}^k = L^k(I^k_{k+1} u^{k+1}) + I^k_{k+1}(\bar{F}^{k+1} - L^{k+1} u^{k+1}) \quad k = 0,..,M-1 \qquad (2.9)$$

and

$$\bar{F}^M = F^M \qquad (2.10)$$

For linear problems equations (2.4)-(2.6) are equivalent to (2.8)-(2.10). One advantage of the FAS method is that equations (2.8)-(2.10) apply equally well to nonlinear problems [B77a,p.347].

A key aspect of the FAS method is that the function stored on a coarse grid G^k approximates the fine grid solution in that $u^k = I_M^k u^M$. These functions - at varying levels of resolution - provide a hierarchy of descriptions of the solution. This makes the FAS type methods particularly appealing to problems in computer vision where structures of interest in the image can occur at many sizes. For this reason, we have chosen to work with FAS methods. The FAS generalization of the Cycle C algorithm is shown in Figure 3.

2.1 An Example: Laplace's Equation

Consider the standard example problem of solving Laplace's equation with a fixed boundary condition; find U(x,y) satisfying

$$\triangle U(x,y) = 0 \qquad \text{on the domain A} \qquad (2.11)$$

and

$$U(x,y) = C(x,y) \qquad \text{for x,y in dA} \qquad (2.12)$$

where dA is the boundary of A. This specific problem and its various equivalents appear in the boxes in Figure 1. Figure 4 shows the results of performing Gauss-Seidel relaxation on a finite difference approximation to equation (2.11) (as formulated in the lower right hand box of Figure 1). Intermediate stages in the iteration are shown in Figure 4a with the initial solution

```
k  <--  M                                  ; start at finest level
F^k <--  F^M                               ; initial RHS
v^k <--  u^M                               ; initial solution estimate
Until  u^M has converged  Do
  Begin
    u^k <--  Relax [ L^k . = F^k] u^k       ; a relaxation "sweep"
    If  u^k has converged  Then

      If  k < M  Then

        k  <--  k + 1                      ; go down one level
        u^k <--  u^k + I^k_{k-1}(u^{k-1} - I^{k-1}_k u^k)   ; add correction
      If  k > 0  Then

        k  <--  k - 1                      ; go up one level
        u^k <--  I^k_{k+1} u^{k+1}         ; initial estimate

        F^k <--  I^k_{k+1}(F^{k+1} - L^{k+1} u^{k+1}) + L^k u^k    ; new RHS
      End if
    End if
  End
```

Figure 3: Cycle C/Full Approximation Storage, multilevel relaxation.

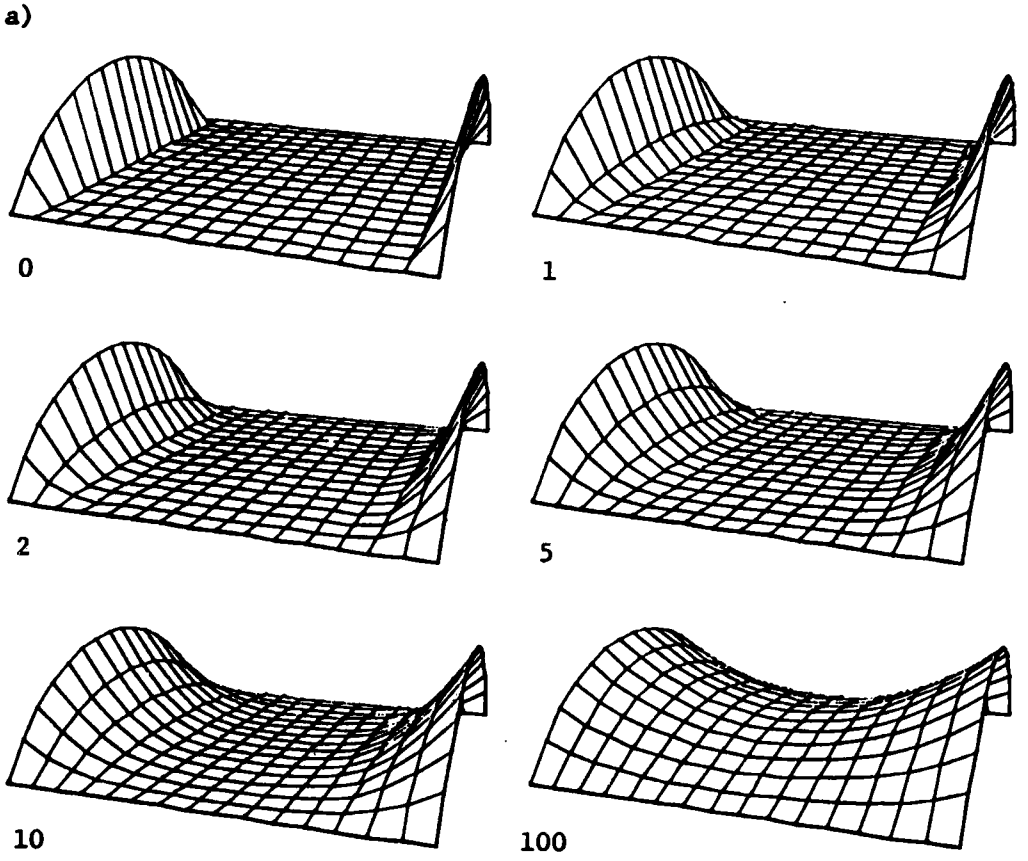
---

estimate (all zeroes in the interior of A ; A - dA) labeled as iteration 0. The residual norm is plotted against iteration number in Figure 4b. In both types of display we see an initial large reduction in error followed by a prolonged slow convergence. In the surface plot, the quick smoothing of sharp changes in the first few iterations is apparent.

The same problem was run under the Cycle C/FAS multilevel relaxation algorithm. Results are shown in Figure 5 and Figure 6 The surface plots show various stages of the computation at different levels. The progress of the algorithm is measured in work units. At the finest level one work unit is defined as one

Figure 4: Gauss-Seidel iteration on Laplace's equation, fixed boundary.

a) Initial data (iteration 0) and iterations 1,2,5,10, and 100; b) Semi-log graph of residual norm vs. iteration number



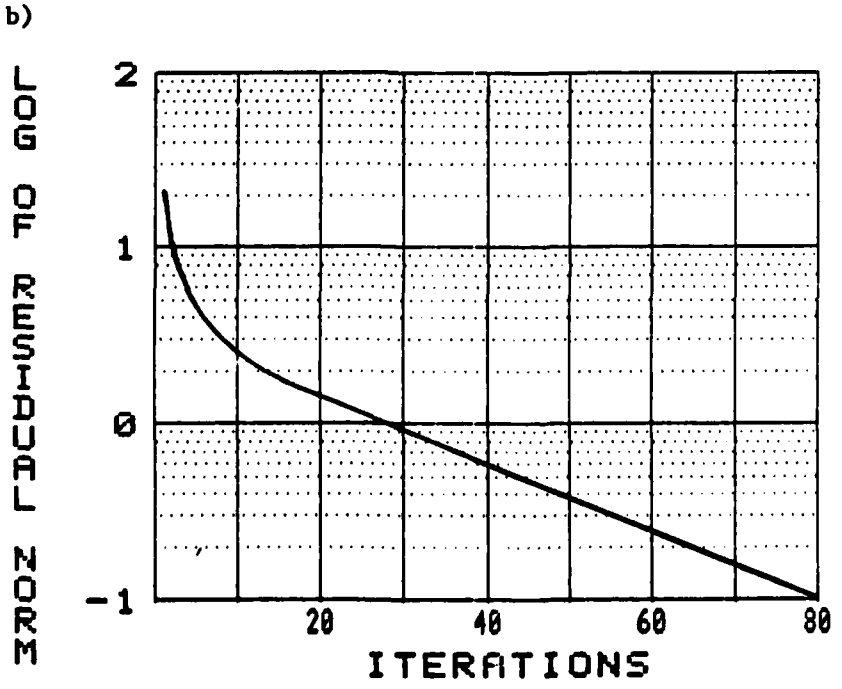
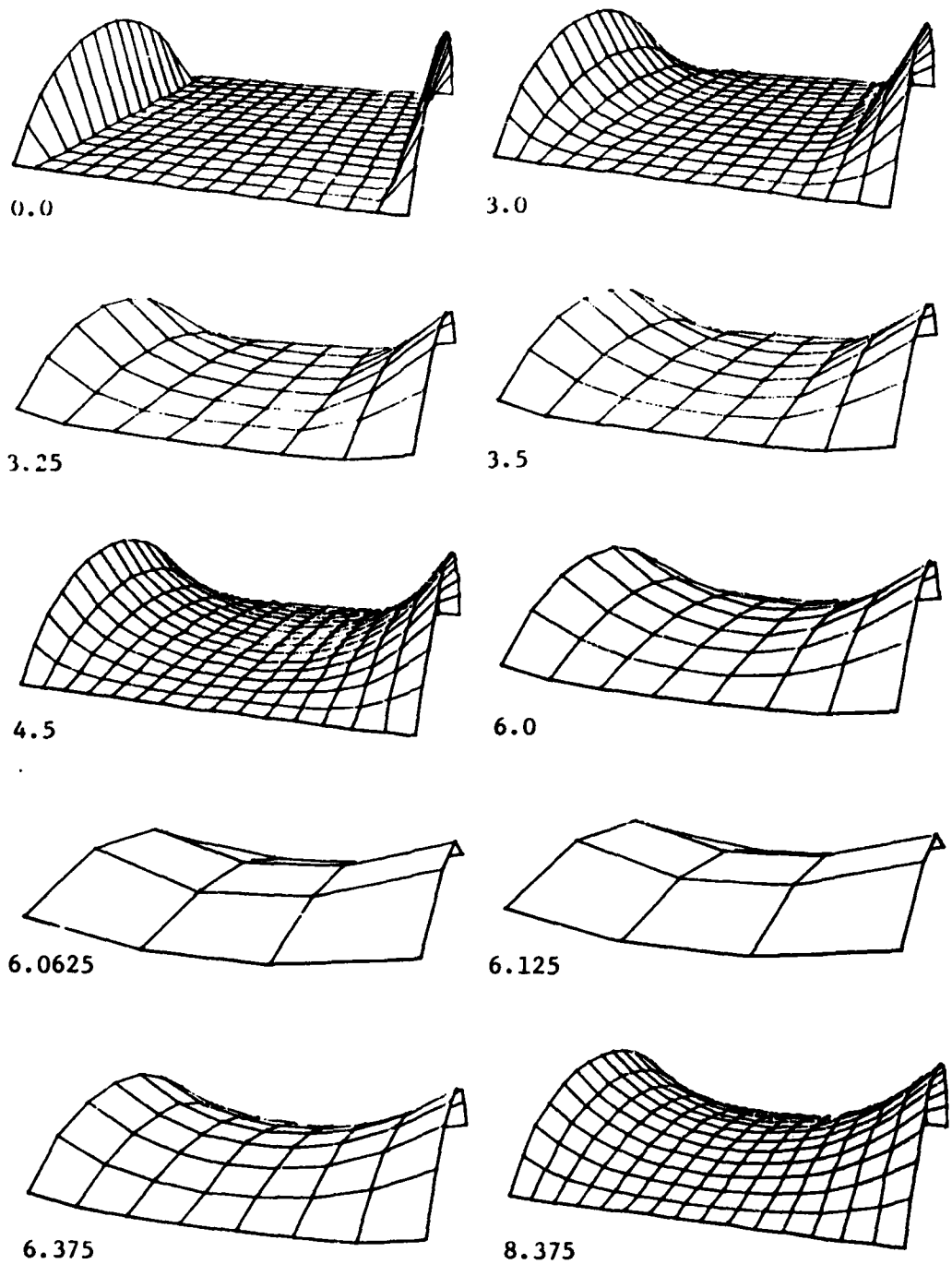


Figure 5: Multilevel relaxation on Laplace's equation, fixed boundary.

Selected intermediate iterations labeled by work number



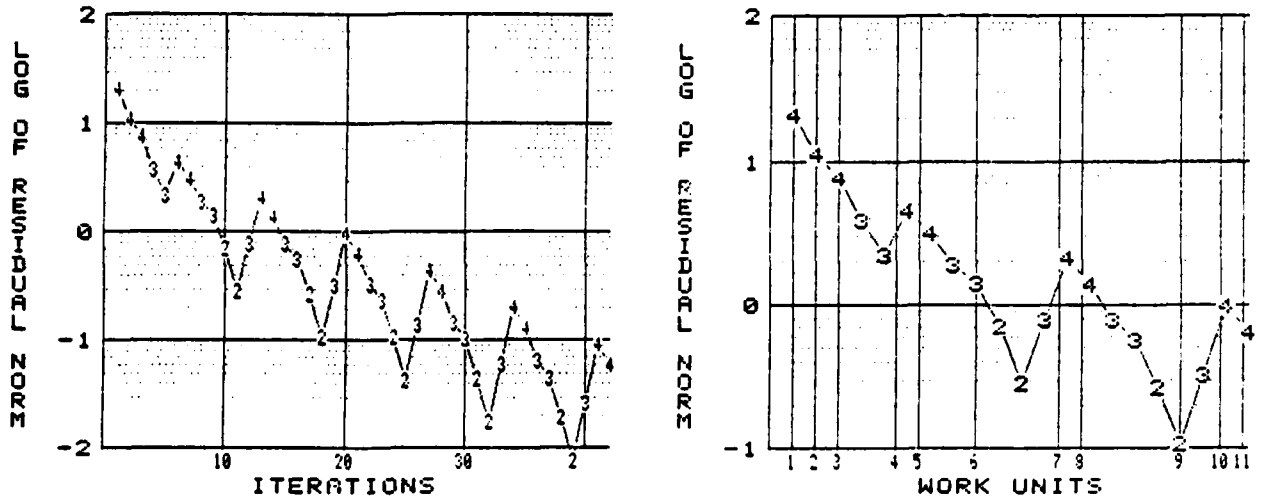


Figure 6: Multilevel relaxation on Laplace's equation.

a) Residual norm vs. iteration number; b) Residual norm vs. iteration with iteration axis labeled with the corresponding work numbers

---

iteration. One level up it takes four work units to equal one iteration. This reflects the fact that only one fourth as much processing is done at that level since processing is proportional to the number of nodes in the grid. In general an iteration at level k costs $(1/4)^{M-k}$ work units. The residual norm is plotted against iterations in Figure 6a. Each iteration is labeled with the level at which relaxation took place. Note that interpolation down the cone (increasing level number) introduces a temporary increase in error. This high frequency interpolation error is quickly smoothed by a few relaxation iterations. Figure 6b also plots residual norm versus iterations but the iteration axis is labeled by work units. This gives a better indication of the work involved in reaching a given residual error.

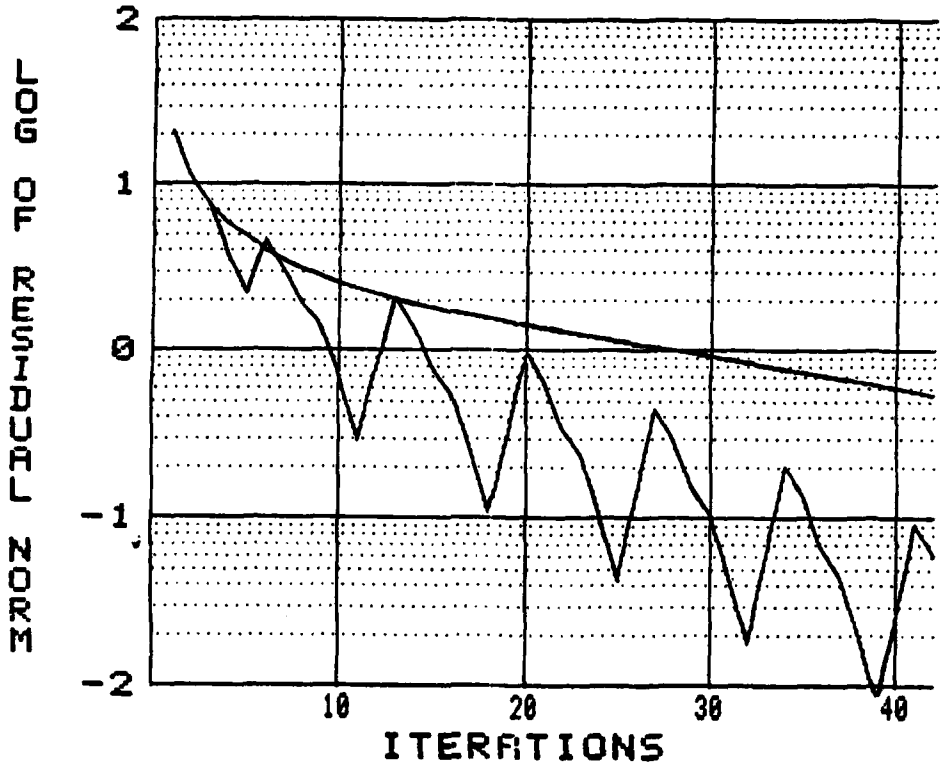


Figure 7: Multilevel vs. single-level Gauss-Seidel relaxation.

This graph compares the results of the experiments shown in Figure 4, Figure 5 and Figure 6.

A comparison of single and multi-level relaxation is given in Figure 7 . Residual norm for both experiments is plotted against iteration number. The multilevel algorithm clearly converges faster. The situation is even better than the graph indicates in that the comparison is based on iterations and not work units. In the single grid algorithm one iteration equals one work unit while in the multigrid experiment work units accumulate much more slowly (see Figure 6b).

3.0 MULTILEVEL OPTIC FLOW COMPUTATION

The optic flow field is a vector field defined over the image space. It specifies the instantaneous velocity of the corresponding image component. Originally it was defined as the projection of the velocities of the environmental objects being imaged [reference Gibson].

3.1 An Optic Flow PDE

Let us represent the optic flow field as $(U,V) = (U(x,y),V(x,y))$ where U is the x-component of velocity and V is the y-component. Let the dynamic image be given as $E(x,y,t)$. Consider the total derivative of E;

$$\frac{dE}{dt} = E_x U + E_y V + E_t \qquad (3.1)$$

This equation relates the gradient of the dynamic image E to the optic flow field (U,V). It also tells us the rate at which E is changing along the direction of motion in the (dynamic) image. Under simple viewing conditions (e.g. orthographic projection and single distant light source) we expect the projection of an environmental point to remain at a constant intensity, i.e. $dE/dt = 0$ and so

$$E_x U + E_y V + E_t = 0 \qquad (3.2)$$

This equation specifies a line in U,V velocity space called the velocity constraint line. This line is perpendicular to the spatial gradient (E_x,E_y). The vector perpendicular to this line with length equal to the distance between the line and the origin is

$$\left(\frac{-E_x E_t}{(E_x^2 + E_y^2)^{1/2}} , \frac{-E_y E_t}{(E_x^2 + E_y^2)^{1/2}} \right) \qquad (3.3)$$

For any (u,v) on the velocity constraint line, this vector is its component parallel to the spatial gradient (perpendicular to an edge).

Although velocity constraint lines can be computed at all points in the image that have non-zero gradient, they do not determine an optic flow field. Other information must be brought to bear to constrain further our choice of a flow field. Horn & Schunck built a variational principle to accomplish this using a first order smoothness constraint on U and V. If in addition to this, we required equation (3.2) to be satisfied, this would lead to a problem in constrained minimization. Such a constraint is likely to prove too strong due to the presence of sensor noise and discretization (truncation) error. This consideration led Horn & Schunck to include the inherent constraint of small dE/dt in their variational principle for optic flow. The resulting variational problem is to find the optic flow field satisfying equation (1.5). The equivalent PDE system (Euler's equations) is

$$a^2 \triangle U - E_x^2 U - E_x E_y V = E_x E_t \qquad (3.4a)$$

$$a^2 \triangle V - E_x E_y U - E_y^2 V = E_y E_t \qquad (3.4b)$$

This elliptic system of PDEs generalizes Laplace's equation in that 1) it is a vector field equation and the component equations are coupled, 2) 0th order terms appear, and most significantly 3) the coefficients are non-constant (they depend on x & y).

3.2 Iterative Relaxation

A Gauss-Seidel method of iterative solution of a finite difference approximation was used;

$$u^{k+1} := \bar{u}^k - E_x * [E_x \bar{u}^k + E_y \bar{v}^k + E_t] / (a^2 + E_x^2 + E_y^2) \quad (3.5a)$$

$$v^{k+1} := \bar{v}^k - E_y * [E_x \bar{u}^k + E_y \bar{v}^k + E_t] / (a^2 + E_x^2 + E_y^2) \quad (3.5b)$$

where $\bar{u}$ and $\bar{v}$ are local averages.

3.3 Experiments

The first frame of the test data for the experiments is shown in Figure 8. In the first experiment the motion is translational : 1/2 pixel to the right and 1 pixel up. One

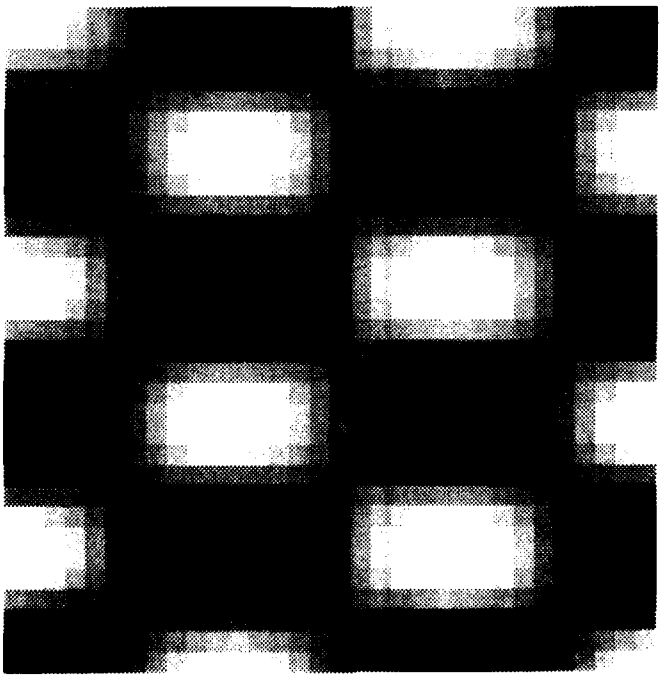


Figure 8: Optic flow test data - frame 1.

Cross product of sinusoids with 1% uniform noise added

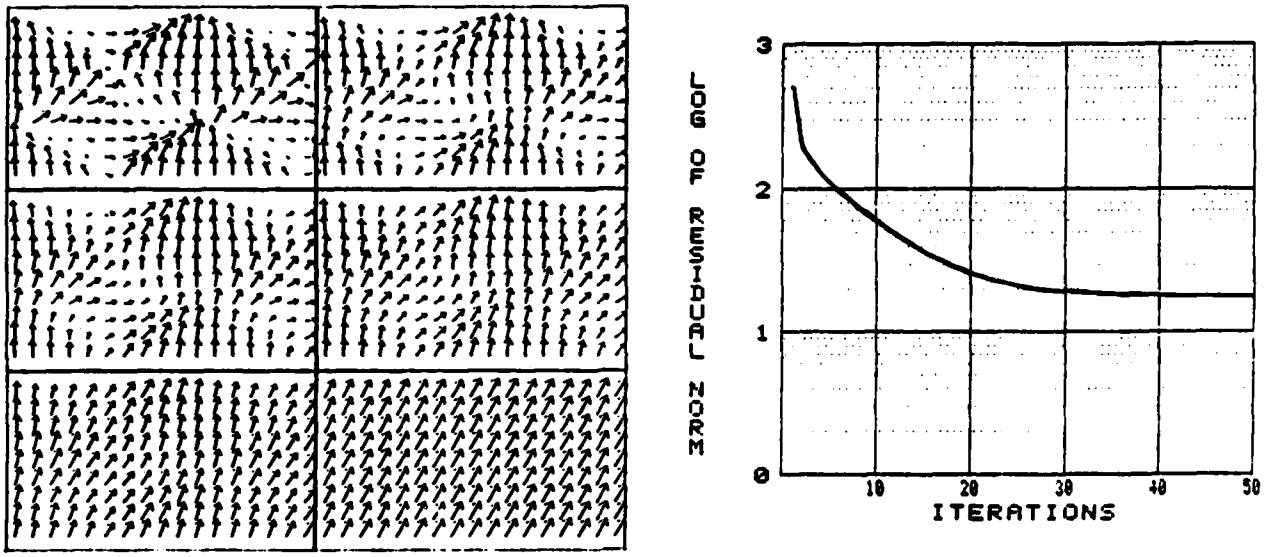


Figure 9: Iterative optic flow computation, Horn & Schunck algorithm.

a) Iterations 0,1,2,5,10 and 50; b) semi-log graph of residual norm vs. iteration number.

---

percent uniform noise has been added to all test data and is uncorrelated between frames. The single grid Horn & Schunck algorithm is shown in Figure 9. Figure 9a shows a portion of the image plane at various stages in the iteration. The initial estimate (iteration 0) for the optic flow field is computed from equation (3.3). An error graph is plotted in Figure 9b.

The results of the multilevel algorithm are shown in the next two figures. In Figure 10 the first 14 iterations are shown for a portion of the full image (the upper left corner). Consecutive iterations at a given level are juxtaposed in the vector plots. In this figure and succeeding vector plots, coarse resolution vectors have been plotted with lengths scaled to the distance between pixel centers. Figure 11a plots residual norm

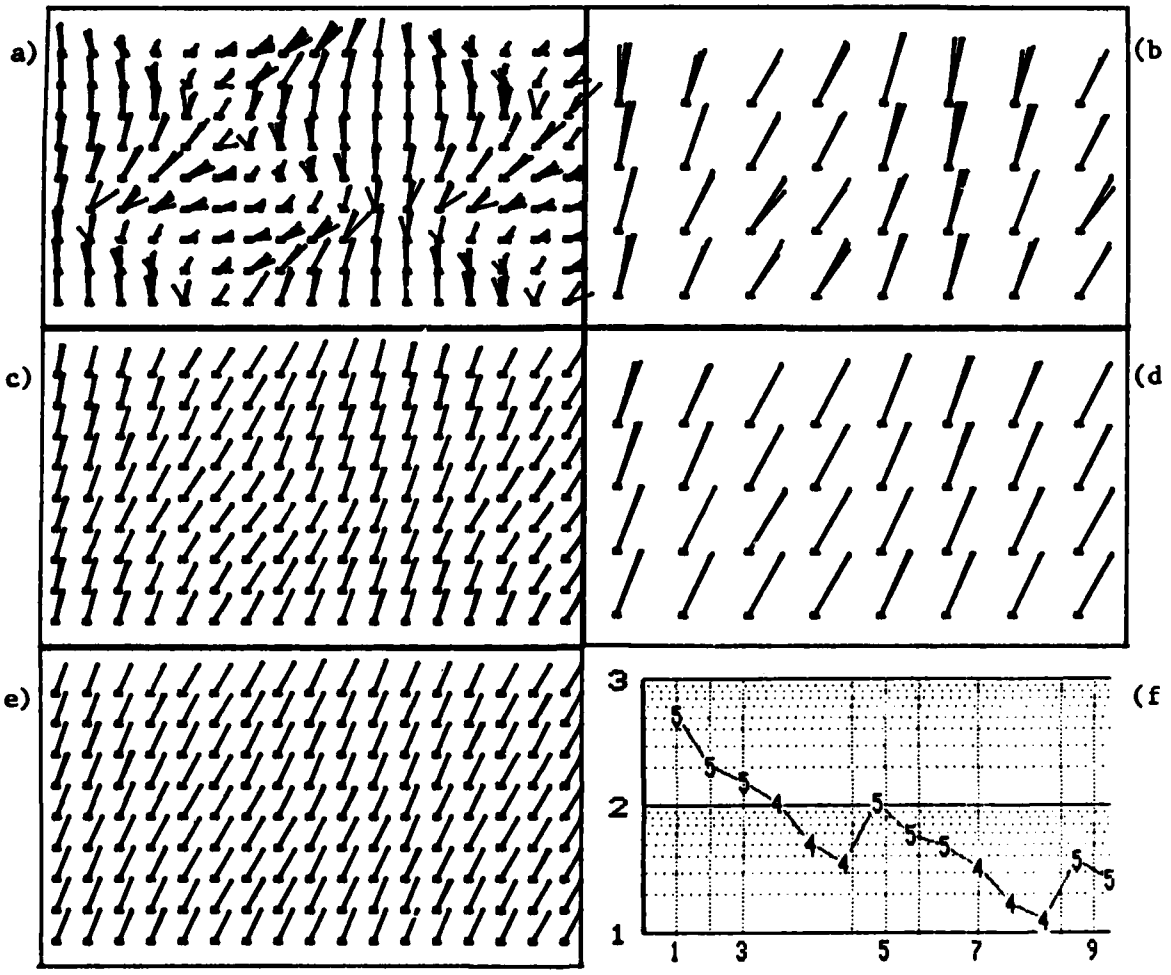


Figure 10: Multi-level optic flow computation.

a) iterations 0,2,3 b) iterations 4,5,6
c) iterations 7,8,9 d) iterations 10,11,12
e) iterations 13,14 f) error graph, iterations 0-14

vs. iteration. Convergence is reached at the 21st iteration. In Figure 11b the iteration axis also runs from 0-26 but it is labeled in work units. Convergence occurs at 13.5 work units.

Finally we present some experiments based on other types of motion in Figure 12 and Figure 13. In both cases the first frame is the same as the first frame in Figure 8. In Figure 12a the

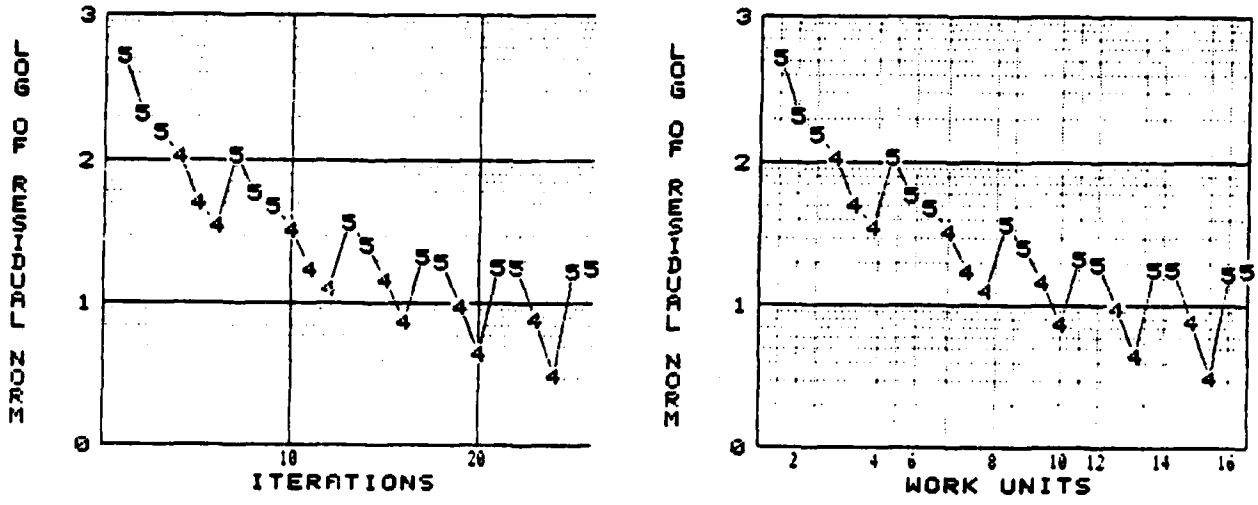


Figure 11: Multi-level optic flow computation: error graphs.

a) Residual norm vs. iteration number;
b) Residual norm vs. iterations, labeled by work number

---

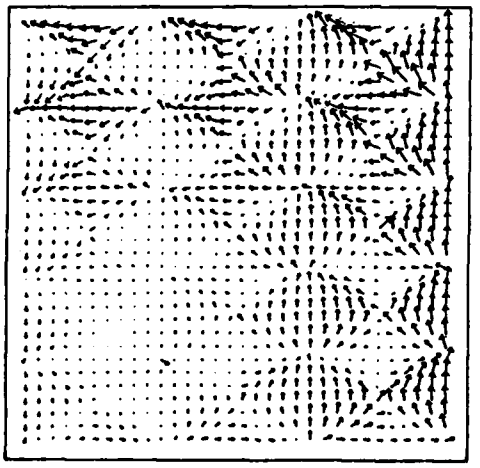
initial estimate for a rotational motion is shown. Figure 12b shows later stages in the multilevel relaxation. The iterations shown as vector plots correspond to the rightmost occurence in the error graph of the given level. Figure 13 shows similar results for depth motion, i.e. translation perpendicular to the image plane.

In both of the above experiments the basic Cycle C/FAS algorithm fails to work. The successful runs shown in the figures were accomplished by preventing the algorithm from going higher (coarser) than the coarsest shown level (level 3) and by specifying how many iterations to perform at each level per cycle. Relaxation done at level 2 causes the estimate to diverge from the correct solution only to be corrected when back at level 3. This appears to be due to a failure to obtain an adequate finite difference approximation of the problem at level 2.

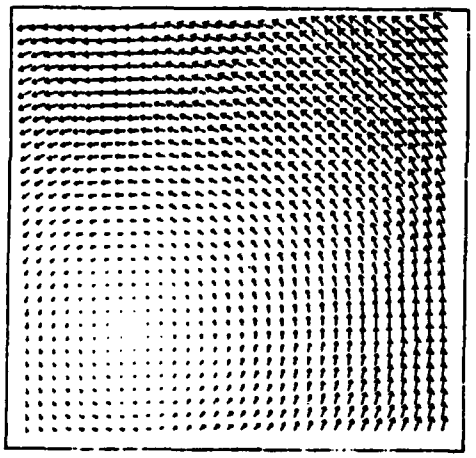
Figure 12: Multilevel optic flow computation, rotational motion.

a) iteration zero
b) iteration 23 at level 5
c) iteration 21 at level 4
d) iteration 19 at level 3
e) error graph.

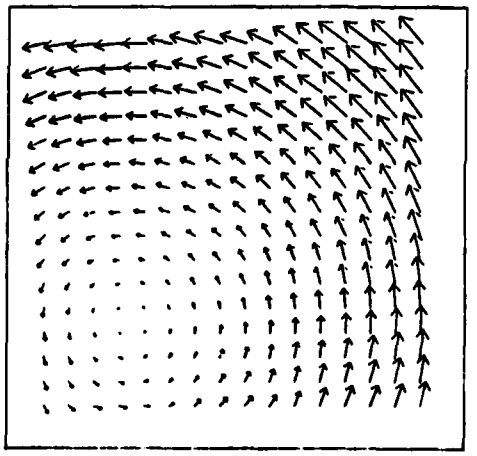
a)



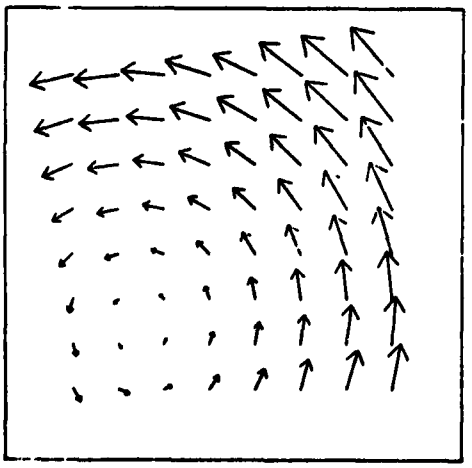
b)



c)



d)



e)

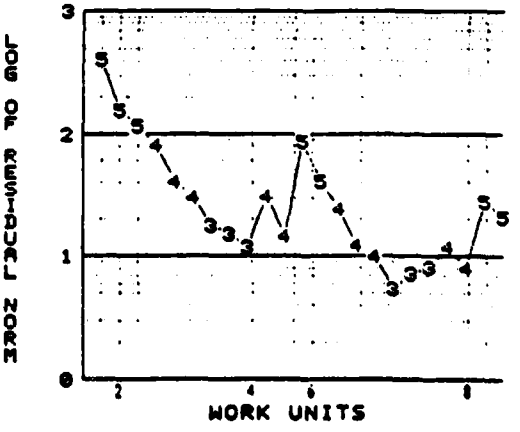
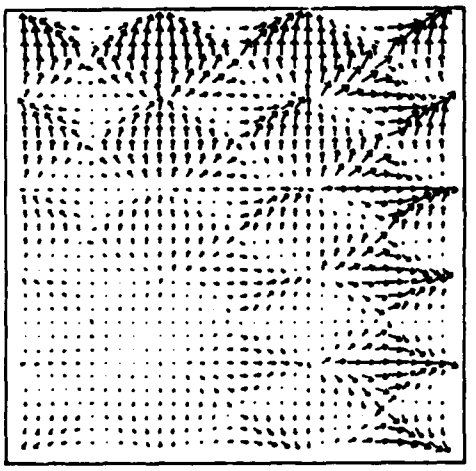


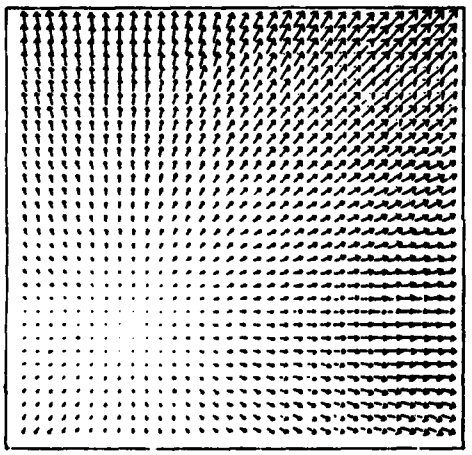
Figure 13: Multilevel optic flow computation, motion in depth.

a) iteration 0
b) iteration 23 at level 5
c) iteration 21 at level 4
d) iteration 19 at level 3
e) error graph.

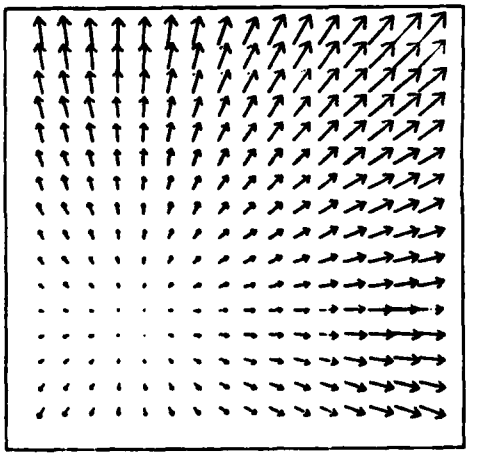
a)



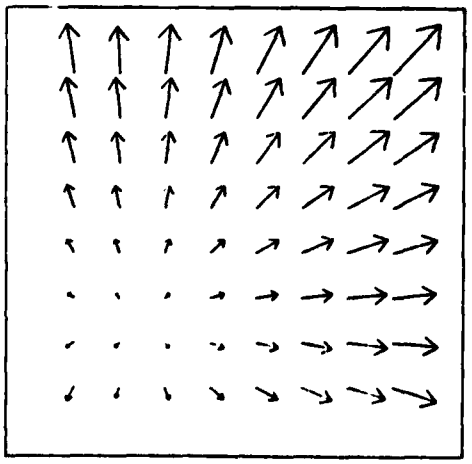
b)



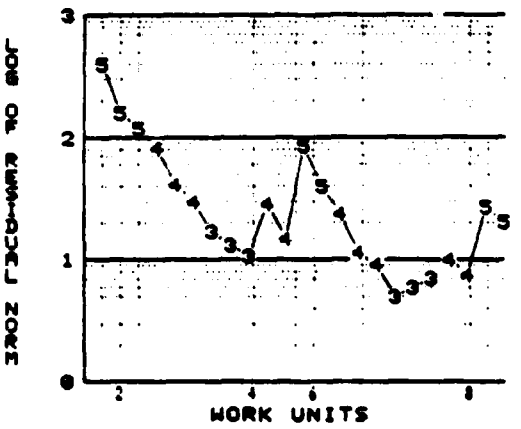
c)



d)



e)



Recall that the coefficients of the EPDE we are solving here (see equation 3.2) are non-constant. The level 2 grid is too coarse (low frequency) to represent the data E_x, E_y, and E_t. Merely restricting the algorithm to levels finer than level 2 is not an adequate solution, since convergence at level 3 is not soon attained. Instituting a fixed pattern of travel up and down the cone solves this problem. More importantly, it eliminates the need to measure the residual norm as iteration progresses, thus reestablishing the locality of the algorithm.

Brandt has also suggested fixed cycling patterns to reduce computation expense, on the basis of experiments in which cycling was observed to occur in regular patterns. For some problems he has shown how to calculate optimal cycling patterns.

4.0 SUMMARY

Variational (cost minimization) and local constraint approaches are generally applicable to problems in low-level vision (eg. computation of intrinsic images). They provide a sound mathematical basis for ideas such as smoothness and "best possible" constraint satisfaction. Moreover they admit computational implementations well suited to the domains of human and machine vision. These are the iterative relaxation algorithms which are "natural" choices for implementation because they can be executed on highly parallel and locally connected processors. Four examples were sketched to show both the range of problems that can be addressed and the variety of approaches that can be taken. These approaches and the corresponding algorithms were related in a general framework embodied in Figure 1.

Multilevel relaxation techniques were introduced as an extension of the basic relaxation algorithms. They provide an efficient way of performing computations which at a single level may require a very large number of iterations to attain convergence. As hierarchical techniques, these algorithms are also suitable for implementation by hierarchical computational architectures. We mention specifically the cones and pyramids that have been studied extensively in computer vision [see various papers in TK80]. The solving of Laplace's equation was shown as an example of the operation of multilevel relaxation and it was compared to standard (single level) relaxation.

A multilevel relaxation was applied to the problem of computing optic flow from dynamic images. Following the development of Horn & Schunck [HS81], a variational problem is established, Euler's equations are derived, and a Gauss-Seidel iterative relaxation algorithm is formulated. This algorithm was extended to a multilevel relaxation algorithm in the style of Brandt [B77a,B77b]. Experiments exhibit the operation of this algorithm and attest to its quick convergence.

References

B77a Brandt,A.,
Multi-Level Adaptive Solutions to Boundary-Value Problems,
Mathematics of Computation, 31(138):333-390, 1977.

B77b Brandt,A.,
Multi-Level Adaptive Techniques (MLAT) for Partial Differential Equations: Ideas and Software,
in Mathematical Software III, Ed:J.R.Rice, Academic Press, 1977.

BURT82 Burt,P.J., Pyramid-Based Extraction of Local Image Features with Applications to Motion and Texture Analysis,
SPIE Conf. on Robotics and Industrial Inspection, San Diego, 1982.

CH53 Courant,R. and Hilbert,D.,
Methods of Mathematical Physics, Volume I,
Interscience Publishers, Inc., New York, 1953.

GR81 Grimson,W.E.L.,
A Computational Theory of Visual Surface Interpolation,
A.I. Memo-613, MIT AI Lab, Cambridge,MA, June,1981.

HR80 Hanson,A. and Riseman,E.M.,
Processing Cones: A Computational Structure for Image Analysis,
In: Structured Computer Vision, Tanimoto,S. and Klinger,A. (eds.), Academic Press, New York, 1980.

HS81 B.K.P. Horn and B.G. Schunk,
Determining Optical Flow,
Artificial Intelligence, 17(1-3):185-204, 1981.
Also in Proc. Image Understanding Workshop (DARPA), April,1981.
Also A.I. Memo-572, MIT AI Lab, Cambridge,MA, April,1980.

HY81 Hageman,L.A. and Young,D.M.,
Applied Iterative Methods,
Academic Press, New York, 1981.

I80 Ikeuchi,K.,
Numerical Shape from Shading and Occluding Contour in a Single View,
A.I. Memo-566, MIT AI Lab, Cambridge,MA, Feb.,1980.

IH81 Ikeuchi,K. and Horn,B.K.P.,
Numerical Shape from Shading and Occluding Boundaries,
Artificial Intelligence, 17(1-3):141-184, 1981.

K71 Kelly,M.D.,
Edge Detection in Pictures by Computer Using Planning,
In: Machine Intelligence Vol. 6, Meltzer,B. and Mitchie,D. (eds.), Edinburgh University Press, New York, 1971.

KD76 Klinger,A. and Dyer,R.D.,
Experiments on Picture Representation Using Regular Decomposition,
Computer Graphics and Image Processing, 5(1):68-105, 1976.

L73 Luenberger,D.G.,
Introduction to Linear and Nonlinear Programming,
Addison-Wesley, Reading,MA, 1973.

MP79 Marr,D. and Poggio,T.,
A Computational Theory of Human Stereo Vision,
Proc. R. Soc. Lond. B, 204:301-328, 1979.

NOR82 Narayanan,K.A., D.P. O'Leary and A. Rosenfeld,
Image Smoothing and Segmentation by Cost Minimization,
IEEE-SMC, 12(1):91-96, 1982.

T78 Tanimoto,S.L.,
Regular Hierarchical Image and Processing Structures in Machine Vision,
In: Computer Vision Systems, Hanson,A. and Riseman,E.M. (eds.), Academic Press, New York, 1978.

TK80 Tanimoto,S., and Klinger,A. (Editors),
Structured Computer Vision: Machine Perception through Hierarchical Computation Structures,
Academic Press, New York, 1980.

TP75 Tanimoto,S., and Pavlidis,T.,
A Hierarchical Data Structure for Picture Processing,
Computer Graphics and Image Processing, 4(2):104-119, 1975.

T82 Terzopolous,D.,
Multi-Level Reconstruction of Visual Surfaces: Variational Principles and Finite Element Representation,
AI Memo-671, MIT AI Lab, Cambridge,MA, 1982.
Also to appear in this volume.

WH78 Wong,R.Y. and Hall,E.L.,
Sequential Hierarchical Scene Matching,
IEEE Trans. Computers, 27(4):359-366, 1978.

Y81 Yachida,M.,
Determining Velocity Map by 3-D Iterative Estimates,
Proc. 7th IJCAI, pp.716-718, Vancouver, B.C., Canada,
1981.