Hierarchical Motion Detection

Frank C. Glazer

COINS Technical Report 87-02

Hierarchical Motion Detection

Frank C. Glazer

Hierarchical Motion Detection

A Dissertation Presented

by

Frank C. Glazer

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1987

Department of Computer and Information Science

Research supported in part by:
National Science Foundation grants MCS75-16098 A01 and MCS79-18209
DARPA grant N00014-82-K-0464
University of Massachusetts under a University Fellowship

Hierarchical Motion Detection

A Dissertation Presented

by

Frank C. Glazer

Approved as to style and content by:

Dr. Allen R. Hanson, Chairperson of Committee

Dr. Edward M. Riseman, Member

Dr. F.S. Hill, Jr., Member

Dr. George D. Reynolds, Member

Dr. Steven J. Zeil, Graduate Program Director
Department of Computer and Information Science

Abstract

Hierarchical Motion Detection

February 1987

Frank C. Glazer

B.A., Yale University
M.A., Northeastern University
Ph.D., University of Massachusetts

Directed by: Professor Allen Hanson

This thesis presents algorithms for the efficient computation of image motions using hierarchical multiresolution methods operating on image data pyramids in the processing cone architecture. Three topics are addressed: (1) fast construction of image pyramids; (2) hierarchical motion detection algorithms: correlation-based and gradient-based; and (3) multilevel relaxation algorithms.

Pyramid building is the first step in hierarchical motion detection. A family of *discrete Gaussian* low pass filters for building low pass pyramids is presented that provide good anti-aliasing characteristics, efficient computation, and a good hierarchical Gaussian approximation. Frequency space analysis, using Fourier Transform theory, is used to compare alternative filters.

Hierarchical correlation overcomes two disadvantages of correlation matching: large search areas which require expensive searches, and repeating features which cause incorrect matches. Coarse-to-fine control provides speed and efficiency when search spaces are large, and accuracy when repeating details can be confused.

Hierarchical gradient-based algorithms extend single level gradient-based algorithms to the computation of large disparities. They use a coarse-to-fine method in which approximate disparities are refined at each level by computing relatively small updates. This allows the gradient-based method's assumption of local linearity to apply in spite of the large total disparities. Experiments show the failure of single level methods (for large disparities) and the success of hierarchical methods.

The two hierarchical algorithms are shown to have comparable accuracy. Comparison of the computational costs, both arithmetic and data transfer, show that the gradient-based algorithm are, in general, less costly.

Multilevel relaxation algorithms for the computation of optic flow are developed and experiments show the expected increased convergence rate over single level relaxation, although some experiments present a problem of divergence at coarse levels. A *local mode analysis* of the relaxation equations shows that convergence is at least as fast as simple smoothing, and that, with strong gradients, convergence is accelerated towards the constraint line. The local mode analysis does not account for coarse level divergence. Divergence is then shown to be due to spatial variation in the image data. Fixed up/down cycling schemes are used to overcome the divergence problem.

Table of Contents

List of Tables

List of Figures

CHAPTER I

Introduction

1 Overview

This thesis presents algorithms for the efficient computation of image motions using multiresolution methods in a hierarchical processing architecture. Image motion is the motion of components of images in a dynamic image sequence. The components of interest may range from single pixels to large regions. We are interested in obtaining a dense set of motion estimates, comprising a field of vectors over the image space. This field of *optic flow* or *displacement* vectors is the primary input to further stages of motion analysis in visual processing systems.

Multiresolution methods operate on a hierarchy of image grids of varying resolution covering the image space in registration. The hierarchical processing architecture includes a general architecture, the *processing cone*, which is regular, parallel and hierarchical, and a multiresolution image representation, the *image pyramid*. The processing cone forms the bridge between the algorithms developed in this thesis and proposed multiresolution processing hardware. Image pyramids are the basic image data upon which multiresolution motion detection algorithms operate.

This thesis contains contributions in three areas: (1) fast construction of the multiresolution image pyramids; (2) hierarchical coarse-to-fine motion detection algorithms in which motion estimates are progressively refined; and (3) multiresolution relaxation algorithms for optic flow computation.

Pyramid building is the first step in hierarchical motion detection, involving low-pass filtering to reduce resolution and interpolation to increase it. A family of *discrete Gaussians* is presented which provides good anti-aliasing characteristics, efficient computation, and a good hierarchical Gaussian approximation. Frequency space analysis, using Fourier Transform theory, is used to compare alternative filters.

Two hierarchical motion detection methods are presented: hierarchical correlation and hierarchical gradient-based methods. Hierarchical correlation overcomes two disadvantages of correlation matching: large search areas which require expensive searches; and repeating features which cause incorrect matches. Coarse-to-fine control provides speed and efficiency when search spaces are large, and accuracy when repeating details can be confused.

Hierarchical gradient-based algorithms extend single-level algorithms to the computation of large disparities by formulating a hierarchical generalization of the single-level method that operates on low-pass image pyramids. A complete formal analysis of the hierarchical method is presented, including (1) the basic equations for computing refined disparity vector, (2) the discrete representations and computations for solving these equations, and (3) a geometric interpretation of the resultant relaxation algorithm. Implementation in a hierarchical processing algorithm is detailed. Experiments show both the failure of single-level methods (for large disparities) and the success of hierarchical methods.

The two hierarchical algorithms are shown by experiment to have comparable accuracy. The computational costs—arithmetic and transfer—are determined and compared. The hierarchical gradient-based algorithm is seen to be a less costly algorithm.

Multilevel relaxation provides efficient computation of otherwise slow relaxation processes. A formal application of multilevel methods to the computation of optic flow is presented and experiments show the expected increased convergence rate over single-level relaxation. However, some experiments present a problem of divergence at coarse levels. A *local mode analysis* of the relaxation equations shows that convergence is at least as fast as simple smoothing, and that, when the gradient

is strong, convergence is accelerated towards the constraint line. The local mode analysis does not account for coarse-level divergence. Divergence is then shown to be due to spatial variation in the image data. Fixed up/down cycling schemes are used to overcome the divergence problem.

2 Motion Analysis

2.1 Definitions

Motion analysis in vision systems uses interframe computations on multiple images of a scene. Motion information can be important at any level of a vision system and is likely to be a component of the ultimate output — we want to see where things are going.

Image motion is the motion of components of the image sequence. We will be concerned with two types of such motion: image flow and image disparity. **Image disparity** is the measure of the frame to frame displacement of components. **Image flow** is the measure of velocity of components. Flow is the limiting case of disparity (divided by time) as the inter-frame interval goes to zero. While the "components" that move can be defined many ways — as single pixels, small neighborhoods, edges, corners, homogeneous regions, etc. — we consider the low end of this spectrum. Single pixels will be assigned motion vectors computed by analyzing small neighborhoods of those pixels. Image motions are represented as velocity fields (flow fields) and as displacement fields (disparity fields) defined over the image plane.

2.2 Motivation

We are interested in the **early detection of motion**. By this we mean motion detection that takes place soon after the capture (sensing) of the dynamic image data, with minimal intervening "preprocessing". This implies that we do *low-level motion detection*, i.e. we use image data that is spatially organized according to the

geometry of the image plane (the "retina") using image processing operators applied uniformly, locally, and in parallel over the image space. This type of processing is often called **iconic**. In contrast, **symbolic** processing is that for which data is represented in more general data structures, such as lists and graphs, upon which more general operations are performed.

Motion detection can also be done at later stages of visual processing. For example, it is conceivable that motion can be detected by running segmentation algorithms on individual frames of an image sequence and comparing the resulting segmentations. However, there are strong reasons to begin detecting motion at the lowest levels of a vision system. The primary one is that motion information can provide valuable early contributions to the overall vision task. A good example of this is the idea of *motion segmentation*, segmenting the image into areas of uniform or slowly varying motion. Low-level motion is also studied in human vision since there are various phenomena that suggest motion detection is done at a low level in natural (human and animal) vision.

In Figure 1 we show the motion analysis topics covered in this thesis and how they relate to each other. The classification of two-dimensional motion analysis shown in the Figure will be described in Chapter III.

2.3 Applications and Related Areas

2.3.1 2D Vision

In a 2D vision system image motion corresponds directly to the motion of viewed objects. Thus identification of moving objects and the measurement of their motion can be computed in a straightforward way from the motion field. *Object tracking* is an application in which the motion of selected objects is detected and the camera moved so as to keep the image of the object centered in the image plane.

Image registration is a 2D vision problem in which the difference between images with similar but relatively displaced components is determined and used to align those images. This is necessary for comparing two images of the same scene

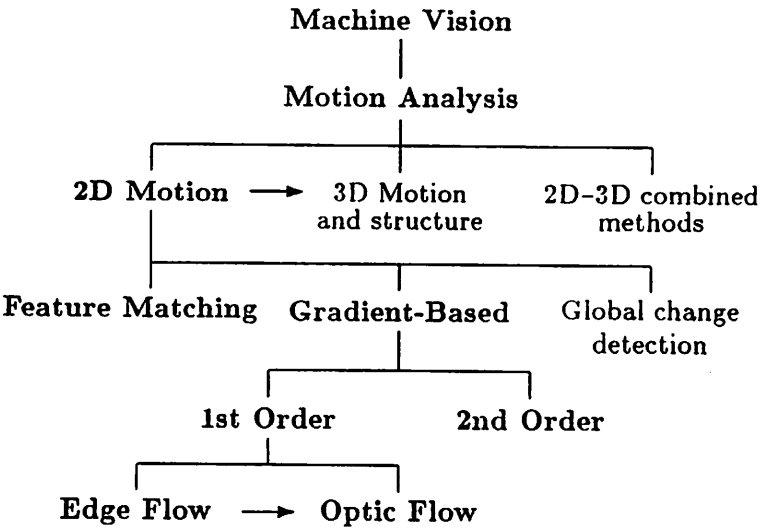


Figure 1: Motion Analysis Topics

Items shown in bold are considered in this thesis.

which have been acquired with differing camera parameters. If the camera model is known then this problem reduces to estimating the two sets of camera parameters (or their relative difference) from the two images. However, if it is not known then individual displacements may still be computed from the image by finding matching regions. The displacement of components from one image to another can be described as a "motion field", each vector of which represents the displacement at a given location. 2D motion detection algorithms can be applied to compute this displacement field. An example of this is reviewed later in this chapter.

2.3.2 3D Vision

The general three-dimensional problem is difficult because 3D scenes are projected onto two-dimensional image planes. This is a many-to-one mapping — multiple combinations of factors in the image forming process can result in identical images or sub-images. The problem then is to compute a reasonable solution to the inverse

one-to-many problem. We do this by imposing constraints on the possible outputs, which are "reasonable" in that these constraints are generic, i.e., they hold for most images. Optical illusions can be explained as images which fool vision systems by violating these constraints.

We define **environmental motion** as the motion of objects in the scene. Image motion results from environmental motion, though the relationship may be complicated depending on what is allowed to vary in the image forming process. Many techniques for computing environmental motion have been proposed and most often they involve or assume the prior computation of image motion. In this case, motion analysis in the general 3D case can be separated into two steps: (1) computation of image motion, and (2) computation of environmental motion from image motion. There are techniques in which this separation cannot be made because 2D processing is constrained by knowledge of the 3D solution space, e.g., see [Lawton 84].

2.3.3 Compression

Compression of moving images is another area in which low-level motion detection is useful. In general, compression techniques exploit redundancy in images to recode them in fewer bits. Static image compression uses spatial redundancy — neighboring pixels have similar (correlated) values. Moving images have both spatial and temporal redundancy, the latter because viewed objects tend to stay fixed or move with constant velocity or acceleration. Erratic motion is the exception. When motion fields are detected in dynamic image sequences, the position of image components in future frames can be closely predicted. Then only the difference (error) between predicted and actual values need be coded. This can be done with a reduced number of bits since the error values lie in a smaller range. This is a form of *predictive coding*.

It is interesting to note that the gradient-based techniques of motion detection presented in Chapter III and extended to hierarchical algorithms in Chapter V were first studied for their use in image compression.

2.4 Assumptions and Non-Assumptions

We assume that motion fields are *smoothly varying* across the image or equivalently that they are *locally translational* when measured in a neighborhood. By this we mean that neighboring points in the image will have similar motion vectors. In a small local neighborhood, the motion vectors will all point in approximately the same direction with the same magnitude thus approximating a purely translational (in the image plane) field. We thus exclude motion fields in which the pixel to pixel variations are large. On the other hand, we do include motion fields more complicated than uniform translation. Moreover, the global variation is not constrained in any way, such as to a pure rotation field.

We also assume that *objects and surfaces are opaque*. This assures us that only one motion can be correctly assigned to any given point in the image. While this avoids full generality, the great majority of actual scenes satisfy this constraint.

Objects are not assumed to be rigid in their motion. Besides narrowing the applicability of a given motion detection algorithm, such a constraint does not provide information that can be used by purely local processing since rigidity is a property defined over a relatively long distance in the image plane. For a 2D vision system, rigid motion is either translational or rotational.

2.5 Processing Constraints

For reasons mentioned above, the approaches presented here are constrained to finding low-level solutions. This has three aspects. First, processing is *coordinate-oriented* (retino-topic). Input data, intermediate data, and output data are represented as two-dimensional arrays in register with the original sensor array. Secondly, *minimal preprocessing* should be required. Computations should begin as soon after the sensed data arrives as possible. Finally, *minimal a priori knowledge* is available. Processing is primarily bottom-up.

The other major constraint we have is to find implementations that fit easily into the processing cone architecture. This is discussed in the next section.

2.6 Extensions

An important problem in motion analysis is dealing with motion boundaries: occlusion boundaries at which the image motion is different on either side. Local motion detection techniques operate on image data from neighborhoods of small but not insignificant extent. When a motion boundary traverses such a neighborhood, motion detection may fail because the various constraints and assumptions used by different detection algorithms fail to hold. In particular, the assumption that motion is locally translational, i.e. approximately constant in small neighborhoods, is used both by correlation methods which match small image neighborhoods and by gradient-based methods which measure image gradients using very small neighborhoods and then combine the information in adjacent neighborhoods. While not addressed in this thesis, our intent is that most current work on this problem is directly applicable to the algorithms presented here.

3 Hierarchical Processing Architecture

When motion detection is done early in the visual process, the low-level processing is well suited to parallel, uniform, locally-connected cellular array architectures. However, there are aspects of the potential solution algorithms which are relatively non-local and hence inefficient for cellular processing. Hierarchical processing architectures and data structures allow us to efficiently extend the single-level algorithms. The hierarchy provides both (1) *computational speedup*: fast filtering (Chapter II), fast "search" (Chapters IV and V), fast interpolation (Chapter I), and fast relaxation (Chapter I); and (2) the straightforward *extension* of existing computational techniques *to cases of extended motion*. The latter includes the use of feature matching when multiple matching features exist (Chapter IV) and the extension of gradient-based methods for computing flow to the computation of disparity (Chapter V).

3.1 Cellular Processing

Low-level vision processes must deal with large two-dimensional arrays of data within which information is distributed homogeneously. This suggests that efficient low-level processing should adopt the following characteristics:

uniform The same operation is performed at all locations in the image space. Uniform machines remain relatively simple in design, independent of the size they attain.

local Direct communication of information can only be made with nearby sites. This avoids the tangle and combinatorial explosion of more extensive connections.

parallel Processing occurs simultaneously throughout the spatial array of sites. This allows for the full use of computational power.

Cellular array architectures are two-dimensional arrays of simple processing elements. All processors run the same algorithm or even the same instruction, executing in parallel (simultaneously). This is a "classic" example of a SIMD architecture — single instruction multiple data stream. Neighboring processors can communicate directly only with their nearest neighbors. Cellular array processing architectures possess the above characteristics of uniformity, locality, and parallelism and thus may provide highly efficient solutions to low-level vision problems. These characteristics can be violated by an algorithm at the cost of reduced efficiency of computation. For example, algorithms that analyze global histograms must perform this analysis in one place (one processing element, one memory). While this occurs the remaining processing elements would be idle.

Cellular array processors are machines that implement cellular array architectures. Many such machines have been built in research projects (see [Fountain 83] for a brief survey and [Preston & Uhr 82] for a large set of individual reports). Current commercial offerings include ICL's DAP, Stonefield's implementation of CLIP, and NCR's GAPP.

3.2 Pyramidal Processing

Cellular arrays are ultimately constrained by the very short range in the image space over which data can be transferred quickly. There are many instances of algorithms which, while naturally defined over an image space, require longer range data transfers to operate efficiently. For example, local parallel image segmentation algorithms, are more effective when information regarding the global distribution of image data values is available. (Specific algorithms will be cited in Chapter II.) The computation of image motion involves image filtering, search for matches, interpolation, and relaxation. All of these can be performed efficiently with local parallel algorithms if hierarchical processing techniques are used to implement the global transfer of information. Image pyramids and processing cones have been proposed to provide data structures, algorithms, and processing architectures that accommodate these tasks.

The motion we wish to detect in an image can have a wide range of magnitudes relative to the pixel spacing in a single-level grid. Thus, motion detection algorithms must examine the image at a correspondingly wide range of locations. Such *single-level processing is resolution dependent* — different algorithms or different parameters are used at different resolutions. Multiresolution image representations with subsampling at coarser resolutions can provide scale invariance. Hierarchical algorithms can thus be applied uniformly at all levels. This is clearly seen in all of the algorithms we will present in that resolution independent image operators are always used.

The detection of large motions must involve the comparison of image data separated by distances comparable to the displacement produced by the largest detectable motion. In a cellular array with communication links only between neighboring processing elements, long range data transfers take many cycles. Thus, *single-level processing is communication bound.* Hierarchical pyramid interconnections allow for global (long range) data transfer while retaining locality, uniformity, and parallelism. We use this general feature for fast computation of low-pass data, fast smoothing by relaxation, and fast interpolation.

Motion detection algorithms may confuse multiple occurrences of small features or image components which look similar. We call this the *repeating feature problem*. If a coarse estimate of motion is available, then such ambiguity can be overcome. Coarse estimates are obtained by examining larger more global image structure. This is a case in which global information is needed to constrain local processing. *Single-level processing does not provide efficient interaction of global and local information*. Hierarchical pyramid interconnection networks do provide this. Specifically, we will use coarse-to-fine control structures, easily defined in the processing cone, to perform scene matching and motion detection.

3.3 The Processing Cone

The previous section generally presents the merits of hierarchical multiresolution algorithms. We wish to further specify the architecture for such processing and thus make a connection with actual or forthcoming machines. However, we do not go so far as to identify a particular pyramid machine implementation with specific processing elements, interconnect network, instruction set, etc. Instead, we adopt the general architecture provided by the processing cone [Hanson & Riseman 74, Hanson & Riseman 80]. It will be described in detail in Chapter II. This architecture determines our notion what is fast to compute and what is slow at an intermediate level of algorithm design — a level above that of specifying instruction sequences.

The distinction is comparable to that between high level languages and machine languages. While we expect our algorithms to run efficiently on an actual pyramid machine, the exact machine code is not specified, nor is it important to do so. We thus retain a degree of machine independence. On the other hand, the operations performed by our algorithms will help determine what specific functionality should be built into a real machine.

4 Related Work

Various hierarchical algorithms have been proposed and used in vision research for motion analysis and related areas such as stereo, registration, and object matching. In this section we review those contributions. Chapter II contains a more complete review of hierarchical processing algorithms and architectures, while Chapter III contains a more complete review of motion analysis techniques. Some of the following work is more closely related to our goals than the rest. In particular, the hierarchical correlation techniques of Wong and Hall, Moravec, Burt *et al.*, and Quam will be reviewed in detail in the next section.

The work reviewed below includes some stereo vision studies. An important distinction between stereo and more general motion analysis is that the former involves matching along a line or thin band in the stereo images. This simplifies the problem since the search area is reduced and the disparity vector is essentially one-dimensional. Even so, the stereo algorithms we review below to some extent utilize data structures and processing methods which may be extended to more general motion detection. There are many other stereo algorithms which offer no such extension. Thus, the discussion below is by no means an overview of stereo vision work, hierarchical or otherwise.

We have divided the discussion into categories based on the motion or matching technique that is used. The first two techniques — correlation and feature matching — are defined and reviewed in Chapter III, independent of their application in hierarchical algorithms. The third category is a miscellaneous one.

4.1 Correlation, Template Matching

Rosenfeld and Vanderbrug studied coarse-fine *template matching* as a way of minimizing computational cost [Rosenfeld & Vanderbrug 77]. The templates were rectangular arrays of pixel intensities. They proposed that templates coarsened by block-averaging be applied to similarly coarsened images. Wherever the degree of mismatch is below a given threshold, the full template is applied to the full image.

They present a theoretical study of this method, showing what degree of resolution reductions minimize expected computational cost under various parameter settings. This limited study was a first step into coarse-to-fine multiresolution matching techniques. Simple pictures, consisting mostly of background, with one to five bits of grayscale were assumed. There were only two levels in the hierarchy. No actual matching experiments were reported.

Wong and Hall used a coarse-to-fine algorithm to efficiently compute the *registration* between two images of the same scene which differ by some translation [Wong & Hall 78]. They used direct correlation as the match measure. In their coarse-fine search strategy, they project all the potential matches where the match measure is above a given threshold. Barnea and Silverman's SSDA (sequential similarity detection algorithm) [Barnea & Silverman 72] is used to speed computation of the match measure. In the SSDA, the accumulation of cross correlation sums for any given match candidate is compared to a running threshold. If that threshold is exceeded, consideration of that match candidate can be ended before the entire correlation sum is computed, thus cutting computation costs. The use of SSDA is not relevant to us since it is not suitable for paralleling processing in cellular arrays.

Wong and Hall built image pyramids with a 2-1 interlevel ratio of pixel spacings. They present a qualitative analysis of the anti-aliasing performance of the discrete Gaussian masks. While their discussion suggests that the larger masks with the most high-pass attenuation should be used, they in fact use simple 2×2 averaging in their experiments.

The sample windows they used were fixed in size relative to the corresponding area in the original (high resolution) image. Hence, in terms of size measured in pixel count, the windows were smaller at coarse levels and larger at fine levels. In their experiments the windows were 8×8 at level 5 (32×32) up to 64×64 at level 8 (256×256). Large sample windows increase the likelihood of obtaining a correct match. Such large windows could be used because of the assumption that the "motion" field was purely translational.

At intermediate stages of the process, for any given window in one of the

images, multiple match possibilities were accepted in the other image and a list of coarse disparity estimates were passed down.

This work was a very good extension of Barnea and Silverman's technique with estimated speedups of 1000 to 3000 over the single-level algorithm. However, since this is a sequential algorithm it cannot be efficiently implemented on a parallel array architecture. In any case, the coarse-to-fine search strategy contributes to computational efficiency independent of the use of SSDA.

Moravec designed a *stereo vision* program as the component that provided visual guidance in an autonomous robot vehicle [Moravec 81]. At each robot position images were obtained from nine camera positions. In the central image at the first robot position, regions to match were selected by an interest operator. A hierarchical correlation matching program was used to find those points in the other eight images. Information from the multiple matches for a given point were pooled to obtain a range measure for the corresponding point out in the world. Matching points in a pair of images were found by a coarse-to-fine search strategy over a multiresolution representations. However, a full multiresolution image data structure was not computed. Rather, 6×6 and 16×16 rectangular blocks about the points to be compared were computed and passed to the matcher. This is in line with Moravec's goal of building a real system on non-specialized computing hardware. The program was designed to run on a standard sequential computer. Even so, there are aspects of the Moravec's basic hierarchical matching method which might lend themselves to processing cone implementations.

The correlation matcher was used for "slider stereo" to match features in multiple images taken at one location by moving the camera along a linear track and for "motion stereo" to match features in images taken at different times. The match measure was a specially defined normalized correlation comparable to variance normalized correlation but simpler to compute.

Moravec uses a coarse-to-fine search strategy, where he matches a 6×6 window around an interesting point in one coarse resolution image with the same size windows at all points on the other image. The window that matches best is pro-

jected down to the next finer level (a 12×12 region at this level), and is used as the search region in the second frame. He then picks a 6×6 window around the interesting point in the first frame at this level and matches it with similar sized windows that are completely contained within the search region. The search area is thus 7×7. This process is iterated until the finest level is reached.

Moravec picked out "interesting" points in the first image as points about which matching would be performed [Moravec 81, p.13]. Besides choosing points that could be easily found in other images, he also wanted a relatively uniform scattering of points and a maximal probability that a few features would be picked on every visible object. It is the criterion of "matchability" that affects the likelihood of correct matching.

Burt proposed the use of pyramid operators to efficiently analyze motion. He suggested both a correlation and a gradient-based technique [Burt 82a]. The correlation method was developed further into the "flow-through" motion analysis algorithm reported by **Burt, Yen, and Xu** [Burt *et al.* 83]. This method assumes that velocity estimates for rapidly moving objects need not be as accurate as those for more slowly moving ones. This allows motion detection to be done independently at different resolutions. The image data is represented as a multiresolution pyramid upon which a discrete Laplacian operator was applied at each level to eliminate low frequencies and thus facilitate correlation matching. At each resolution, the magnitude of detectable motions and the accuracy of the measure of motion is proportional to the resolution. At low resolutions, only large motions are detectable and only at a low accuracy. At high resolutions, only small motions are detectable but at a high accuracy. The match measure is correlation and it is performed on bandpass filtered and subsampled versions of the original image.

Burt *et al.* share our goal of obtaining a local, uniform, parallel algorithm which is computationally efficient, uses simple arithmetic, and can be implemented in special purpose digital hardware at "real time" rates. Their algorithm is similar in its use of Burt's fast pyramid building operators [Burt 81] and the use of small 3×3 search areas. (Although they did use 5×5 neighborhoods in their experiments.)

We retain the requirement of obtaining a single final motion field to be represented at the highest level of resolution. There it can be used by known motion field analysis processes.

Burt *et al.* did not specify how the motion field at various levels might be combined into one field. It is conceivable that further processing could take place at all levels, but this remains unexamined.

Quam applied coarse-to-fine hierarchical correlation matching to a stereo vision problem [Quam 84]. He used Burt's Gaussian (low-pass) pyramid as a multiresolution data structure. Coarse-to-fine hierarchical matching was done using one-dimensional search along the epipolar lines. Candidate matches were considered in a (one-dimensional) neighborhood of -2 to $+2$ pixels. The match measure was Gaussian weighted normalized cross-correlation in a 13×13 window. A confidence measure was applied to accept or reject the match. Matches were accepted if they were not at either end of the -2 to $+2$ neighborhood and if the best and next-best matches were next to each other.

A disparity surface interpolation was used to fill holes wherever the matching operator failed. This is a hierarchical, bottom-up then top-down method. First data is reduced up until the holes are small. Then the holes are filled in (interpolated) by solving a linear system of equations defined over a 7×7 neighborhood centered at the hole. The number of equations equals the number of holes in that neighborhood.

After coarse disparity estimates were projected to the next finer level, geometric warping of the target image using the projected disparity estimates was performed to improve performance of the cross-correlation matching operator. Details of the warping algorithm are not supplied.

Quam presents a fairly complete system for performing stereo matching. The relative importance of the component parts and the parameters selected for them is not clear. Alternative choices exist for the building of the data pyramids, the warping algorithm, the match measure (type and size), the match confidence measure (the sharpness measure), the estimation of sub-pixel disparities, and the surface interpolation algorithm. In any case, successful experiments are shown on real data.

4.2 Feature Matching

Marr and Poggio's theory of *human stereo vision* [Marr & Poggio 79] was implemented as a computer program by Grimson [Grimson 81a] to test that theory. This theory hypothesizes a coarse-to-fine *feature matching* process. The features are vertical or near vertical edges which have been detected as zero-crossings of bandpass filtered versions of the original image. The bandpass filters are discrete $\nabla^2 G$ operators, computed by taking the Laplacian of a Gaussian "blurred" version of the image, and implemented as on-center off-surround convolution kernels with center widths of 4, 9, 17, and 35 pixels. The outputs of these operators comprise a multiresolution representation of the image. The matching process begins at the coarsest level. For each edge in one image, a "symbolic matching" algorithm is used to find a match in the second image.

Mutch and Thompson report on hierarchical *feature matching* for *motion detection* using local extrema in $\nabla^2 G$ bandpassed images as the features [Mutch & Thompson 83]. Matching was done using a relaxation labeling process [Barnard & Thompson 80] with coarse matches constraining the search area for finer matches. Since the features occur relatively sparsely in the image, an interpolation was performed to fill in displacement vectors. They refer to interpolation by "blurring" without a detailed explanation. It is not clear what the cost of such a computation is, nor the range over which adequate interpolation can occur. It is likely that low computation cost and a large interpolation range are mutually exclusive.

Both Grimson, and Mutch and Thompson do not use subsampling to get reduced size for the coarser images. This makes building the multiresolution image structure expensive. (Burt's HDC — hierarchical discrete correlation [Burt 81] — can provide fast computation in this case, but this method was not apparently used.) Using single sized image grids also complicates matching processes at coarser resolution levels, where features will be far apart relative to the grid size. On a cellular array of processors, this would require long communication paths to perform the "local" feature comparisons. Note also that many processors would be idle in

such a system. Of course Grimson, and Mutch and Thompson were not specifically considering any particular hardware implementation within which their algorithms might ultimately run.

Crowley and Parker present a multiresolution representation for shape to be used in *object detection*, by matching descriptions of forms in the image and object models [Crowley & Parker 84]. They use features similar to Mutch and Thompson's, in this case peaks (local extrema) and ridges (local directional extrema) in bandpass image. These features become nodes in a graph that describes the form of an object or a component of an image. The nodes are linked in paths of ridges that connect peaks at single levels of resolution and paths which connect peaks and ridges between adjacent levels. The focus of this work is on the representation itself. Various aspects of the use of the representation in matching object models to images are discussed, but specific matching algorithms are not described. In any case, the general nature of their matching strategies is that of graph matching, in which similar node-arc structures in two graphs are discovered. This is quite a bit removed from our goal of low-level motion computation. Also, the implementation of such an algorithm on a regular grid of processors or in a processing cone is hardly straightforward. Motion information could be derived after matching has taken place, but it would be sparse.

4.3 Other Work

Davis and Roussopolous developed hierarchical coarse-to-fine fast matching algorithms to perform *pattern matching* in a pattern database system [Davis & Roussopoulos 80]. The database contains binary pictures, each of which has been normalized for size, rotation, and position. A "sum quad-tree" representation is used for each picture. This is a cross between a pyramid and a quad-tree built bottom-up from the binary image. Each node stores the sum of values of its son's nodes. A compact quad-tree like structure is obtained if nodes over regions of all 0's or all 1's in the base image are made terminal (i.e. their sons are not explicitly stored). A hierarchical matching algorithm they presented determines if

two binary pictures differ at fewer than some given number of positions. The match measure is *total absolute difference*.

Davis and Roussopolous's algorithm only determines the existence of an overall global match. Motion or displacement of patterns is not computed. In fact it is assumed that the pictures have been "normalized" to remove any positioning differences. The extension to grayscale data is not presented nor is it clear what it would be.

Lucas and Kanade have developed a new general method of image matching based on differences in image values between frames and measures of derivatives of the images [Lucas & Kanade 81]. As such their method is closely related to the gradient-based methods discussed in Chapter III. They have also noted the need for a coarse-to-fine approach using a multiresolution image representation. While it can be applied to the problem of motion analysis, they have not developed such an application. It is likely that their approach can be used to build a hierarchical motion detection system operating within the processing cone. Without further details it is difficult to measure its success. In Chapter V, we present a hierarchical gradient-based algorithm for motion detection. A further discussion of Lucas and Kanade's method in that chapter will show how their technique might be implemented in the processing cone.

Grossky and Jain have presented plans for a *region matching* algorithm for *dynamic scene analysis* [Grossky & Jain 83]. They fit elliptical parabaloids to regions using a pyramid linking strategy. (Pyramid linking, a hierarchical image segmentation technique, is discussed in Chapter II in a review of current research on hierarchical algorithms.) where it is assumed that moving surfaces have already been segmented out of the image. Regions are matched based on the closeness of the values of the parameters of the fit. The hierarchical pyramid structure is only used for simple segmentation and fast computation of the fit parameters. It is not used in the matching process. This proposal is quite far from our goals in that motion detection is not early because of the reliance on a prior segmentation, nor is a relatively dense field of motion vectors obtained since only large regions

are matched. In some sense such a method tries to jump past the problem we have set and achieve an analysis of motion at the level of objects or surfaces in the images. While this is one of the ultimate goals of motion analysis, this proposal would accomplish it at the cost of an over-constrained image model (a stationary background of gray level 0 is also assumed).

Thomas and Martin have presented some preliminary ideas on applying focus-of-attention mechanisms to control motion processing [Thomas & Martin 84]. In their scheme, processes would "attach" to levels in a hierarchy of resolution: noticing objects at coarse levels, going to finer levels to confirm object identities, and staying at fine levels to track objects. This is far removed from our requirement of local, uniform, parallel processing. It would only be applicable to single-processor or MIMD multiprocessor machines, with no extension to cellular array machines.

Neveau, Dyer, and Chin have applied hierarchical coarse-to-fine methods to do *object matching* using generalized Hough transforms [Neveau *et al.* 85]. Because the generalized Hough transforms are first applied at a coarse level, the 3D array indexed by position and orientation can be kept small, thus minimizing the task of filling the array and finding peaks (clusters). The clusters that are found represent possible match candidates. Their positions and orientations constrain the search for matches at the next finer resolution. Thus, these can also be performed in relatively small parameter arrays.

5 Outline

The problem we have posed is: the computation of motion early, densely, and efficiently with low-level hierarchical algorithms operating on pyramid data structures in the processing cone architecture.

Chapters II and III are introductory in nature. Processing cones and image pyramids are defined in Chapter II. Other efforts are reviewed, their usefulness is discussed, and the various filtering methods used in building image pyramids are presented there. Chapter III presents the task of low-level motion analysis and

computation. The two major methods of feature matching and gradient analysis are discussed in detail.

Hierarchical feature matching is presented in Chapter IV. All of the advantages of hierarchical methods are seen to come into play. In Chapter V, hierarchical gradient methods are developed, extending these methods to the computation of disparities. A computational cost comparison of the two hierarchical motion detection algorithms, correlation and gradient-based, is presented in Chapter VI. Chapter I presents a hierarchical algorithm for motion computation that uses multilevel relaxation to compute optic flow from edge flow. This adds to the method in Chapter V while presenting another class of hierarchical methods distinct from those seen in Chapters IV and V.

CHAPTER II

Processing Cones and Image Pyramids

This chapter includes (1) a review of the use of regular hierarchical structures in machine vision, (2) definitions of the processing cone architecture and image pyramids, (3) discussion of the basic image operators used in building image pyramids and passing data up and down the processing cone.

1 Hierarchical Structures in Machine Vision

Processing cones and image pyramids are regular hierarchical retinotopic processing structures used to process image data at varying levels of resolution. "Regular" refers to the homogeneous geometric structure of the grids at each level and to the level-to-level registration of these grids. The "hierarchy" is the ordered sequence of image grids of increasing resolution. "Retinotopic" describes the geometric 1-to-1 correspondence between processing elements and locations in the image plane. These hierarchies should be distinguished from conceptual hierarchies also found in machine vision which span the low to high level range of abstraction (e.g., pixels, edges, shapes, objects) that will be found in a complete vision system. Rather, processing cones are a generalization of cellular array architectures which provide high-speed efficient processing for the image-oriented operations used in low-level machine vision. Pyramids and cones extend the usefulness of cellular arrays, but do remain within the realm of low-level vision. Such extensions include: (*i*) focus of attention and planning, (*ii*) scale invariance, (*iii*) computational speedup, and

(*iv*) a robust way of easing the locality constraint of cellular architectures.

Regular hierarchical structures are established computational architectures in computer vision (see the surveys and collected papers [Tanimoto 78, Tanimoto & Klinger 80, Rosenfeld 83, Samet 84]). In this section we review (1) the original work in hierarchical structures, (2) current major areas of research, and (3) applications in motion analysis.

1.1 Early Work in Hierarchical Machine Vision

Hierarchical processing was first introduced to solve specific problems in computer vision. Kelly used a multiresolution image representation in a program that detected the major boundary curves (outlines) in pictures of human faces [Kelly 71]. He used edges in a coarsened image as a "plan" for locating edges in higher resolution versions of the same image. This is the first use of coarse-to-fine focus of attention in a multiresolution representation.

Rosenfeld and Thurston applied multiresolution edge, spot, and curve detectors in a coarse-to-fine manner to detect edges and curves of varying resolutions. A multiresolution image representation was not used in this work; rather, all sizes of detector were applied to the full image. For each possible edge location the best size edge to posit was determined by a "global" decision among the levels of resolution. This is the first use of interlevel processing to combine global and local information in a multiresolution representation.

Hanson and Riseman, realizing the broad applicability of hierarchical solutions to vision problems, proposed a general architecture for hierarchical algorithms. Their **processing cone** *is a parallel array architecture, hierarchically organized into layers of decreasing spatial resolution* [Hanson & Riseman 74, Hanson & Riseman 80]. It is designed to provide parallel processing power both locally (fine resolution) and globally to varying degrees (coarse resolutions). Besides handling multiresolution image data, this architecture is generally applicable to all low-level vision data, including, for example, representations of edges, local 2D shape, and local 2D motion. This is the architecture we will use. It is described

further in Section 2.2.

Tanimoto and Pavlidis also proposed a general formulation of hierarchical processing [Tanimoto & Pavlidis 75]. However, their emphasis was not on processing architecture, but rather on the data structure used to represent multiresolution data. Tanimoto and Pavlidis' pyramids are *sequences of digitizations of the same image at increasingly higher degrees of spatial resolution.* This idea is can also be generalized to allow pyramids to contain other than just image intensity data.

We will use pyramids to represent high-pass filtered data, vector fields, and binary (image) masks. This pyramid data will be manipulated by hierarchical algorithms operating in the processing cone. In terms of both image pyramids and the processing cone architecture, there is a correspondence between the data and the image coordinate space. We define a **pyramid** as *a sequence of arrays of decreasing grid size, which represent image data at multiple levels of resolution.* It is interesting (and perhaps clarifying) to note that the standard graphical depiction of pyramids and cones as a stack of grids of varying size and constant grid spacing (see Figure 3) does not well represent the fact that all of the grids are meant to "cover" the original image. Thus, in the image space geometry, the grid size (i.e. extent, NOT point count) remains constant while the grid spacing increases as we go up the hierarchy.

Uhr's recognition cone is a parallel-serial computer composed of a hierarchical sequence of layers of processing elements, the layers decreasing in size up the hierarchy [Uhr 72, Uhr 78]. Each layer is composed of processing elements each of which is assigned a single transform to compute. Each transform looks at the output of previous stages and combines them with a weighted sum followed by thresholding. All transforms in one layer of the cone are performed in parallel. Lower (early) layers are strongly retinotopic in that the data they contain and operate on is location specific. At higher levels, the correspondence between location in the layer and location in the input image is more approximate and certainly of less importance. At these layers, more symbolic object or scene data is represented. This is the one hierarchical architecture we review which includes an abstraction hierarchy as well

as the resolution hierarchies we are interested in.

There is a second major type of regular hierarchical data structure: regular decompositions and quadtrees. These are hierarchies of square regions built by recursive subdivision. Klinger and Dyer's **regular decompositions** [Klinger & Dyer 76] recursively divide a picture area into quadrants, discarding quadrants which are "non-informative", storing those which are, and further subdividing those which are ambiguous. Thus, the density of subdivision varies across the image depending on the amount of detail present in each part of the image. Their intent was to find an efficient data representation to facilitate two-dimensional search using a representation in which large non-informative areas were "deleted". As a measure of discrimination they used (1) thresholding the gray level, and (2) edge detection. The former is a form of the quadtree.

Horowitz and Pavlidis used an image data structure similar to regular decompositions to segment images into homogeneous regions, where *homogeneous* means that the range of values within the region is less than some threshold [Horowitz & Pavlidis 76]. They combined two basic segmentation techniques of (a) merging small primitive regions, and (b) recursively splitting the entire image, into one which gave significant speed improvements. Their algorithm was neither bottom-up nor top-down but rather "middle-out". An initial segmentation was built as a $2^k \times 2^k$ grid of squares of intermediate size covering the $2^N \times 2^N$ image space. Three processing stages were then applied: splitting, merging, and grouping. In the splitting stage, all quadrants which are not relatively homogeneous are *split* into their four sub-quadrants. In the merging stage, four homogeneous quadrants with a common "father" are merged if the father remains homogeneous. Finally in the grouping stage, equivalence classes of square region are formed in which the union of the regions in a class is a connected and homogeneous set of pixels. The splitting and merging stages used simple hierarchical segmentation strategies (top-down splitting and bottom-up merging) applied to a hierarchical image data structure essentially the same as Klinger and Dyer's regular decompositions. This work was a precursor to the more elaborate pyramid-linking hierarchical strategies

referred to in the next section.

Quadtrees are quaternary (four way branching) trees whose depth varies locally with the amount of detail in the image. In general they are used to represent binary images again using a recursive subdivision into quadrants. Leaf nodes represent (square) areas of the image which are all white or all black. Branch nodes represent areas with detail and have pointers to the four sub-quadrants. For an extensive review of quadtree research see Samet's survey [Samet 84].

While pyramids represent data at multiple levels of resolution, regular decompositions and quadtrees are multilevel representations of data. Or, as Samet says, "A pyramid is a multiresolution representation, whereas the quadtree is a variable resolution representation" ([Samet 84, p.224]). One level of a pyramid is a meaningful entity in itself. It represents the image data at a given spatial resolution. In general, one "level" of a quadtree is not, since image areas represented by branch nodes of the tree have an unspecified content. That content can only be known by looking at lower levels of the tree.

1.2 Current Areas of Research

Regular hierarchical structures are now finding applications in a variety of processing tasks. Algorithms have been proposed for fast sorting and rank filtering in a pyramid processor [Tanimoto 83a, Stout 83, Tanimoto 84]. These include the computation of local maxima and minima (sorting) and median filtering (rank filtering). Multiresolution shape representations have been proposed deriving from image pyramids [Crowley & Parker 84, Neveau *et al.* 85]. Image pyramid representations have been applied to achieve compression and progressive transmission of image data [Sloan & Tanimoto 79, Burt & Adelson 82, Hill *et al.* 83]. Alternative pyramid structures have been proposed such as hexagonal pyramids [Burt 80, Hartman & Tanimoto 84, Lucas & Gibson 84] in which individual levels are composed of hexagonal grids. Image segmentation, a fundamental problem in machine vision, has been approached with hierarchical algorithms very successfully. These are briefly mentioned below.

1.2.1 Image Segmentation

Image segmentation is the spatial grouping of image pixels into homogeneous regions. These regions are uniform in some attribute of the pixels or neighborhoods they contain. A central problem in image segmentation is that the attributes that give a region its identity are best determined globally. This is why histogramming techniques have been used so extensively for image segmentation, they give a global picture of image statistics. However, histogram techniques involve global processing over the entire image space. Recursive segmentation techniques [Nagin *et al.* 82] can reduce the extent of global processing but remain essentially non-local. Hierarchical processing can also introduce a global sharing of region attribute information, but in a way that cleanly bridges the gap between pixel-level and region level image analysis processes. Such techniques include split-and-merge algorithms [Horowitz & Pavlidis 76, Chen & Pavlidis 79, Pietikainen *et al.* 82] and pyramid-linking [Burt *et al.* 81, Pietikainen & Rosenfeld 81, Hong *et al.* 82, Antonisse 82, Cibulskis & Dyer 84].

1.2.2 Linear Filtering

Multiresolution representations of images can be computed by using linear filters to perform low-pass and band-pass filtering. For coarser levels, low cutoff frequencies and pass bands are used. The spatial filters that perform such filtering are relatively large since they must average together data in large neighborhoods. The coarser the resolution, the bigger the spatial filter.

Hierarchical filtering schemes can be used to efficiently compute all levels of resolution. The original data is low-pass filtered and sub-sampled to compute a reduced resolution version. The same filtering and sub-sampling is then applied to the reduced resolution version to yield a yet coarser version. This operation is iterated going up the cone. The computational efficiency derives from the fact that each interlevel operation involves the same relatively small filter.

Both Burt and Crowley have proposed such schemes [Burt 81, Crowley 81].

They will be discussed further in Sections 4 where we develop our own pyramid building filters.

1.3 Hierarchical Matching and Motion Analysis

In Chapter I, we reviewed work in hierarchical matching and motion analysis. Some of those methods are readily implemented in the processing cone. Specifically, those of Burt *et al.* and Quam [Burt *et al.* 83, Quam 84].

1.4 Multigrid Relaxation

The solution of partial differential equations (PDE's) is a major topic in numerical analysis because of the wide use of PDE's in many types of mathematical models of physical phenomena. These models are most often defined over some spatial and/or temporal domain which is discretized for the purpose of computer simulation. Computation costs can be very high when accuracy requirements demand fine resolutions and hence many data points in the discretization. **Multigrid relaxation** is an efficient new technique for solving PDE equations [Brandt 77a, McCormick & Trottenberg 83]. It involves a multiresolution representation of the problem domain over which relaxation is performed at all resolution levels.

A recent development in the study of low-level computational vision is the formulation of global cost minimization approaches which are translated into local neighborhood constraints and solved by iterative relaxation. Examples include noise elimination [Narayanan *et al.* 82], surface interpolation [Grimson 81b], shape from shading [Ikeuchi 80, Ikeuchi & Horn 81], and optic flow computation [Horn & Schunck 81]. Iterative relaxation algorithms are good choices for implementation of such algorithms 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. The new multigrid relaxation techniques converge much faster and can, in general, be applied to these vision problems [Glazer 82, Terzopolous 82]. Moreover, they are well suited to programming in processing

cones. In Chapter I, these techniques are applied to the problem of computing optic flow from dynamic images.

2 The Processing Cone Structure

2.1 Image Grids and Pyramid Multigrids

A *continuous image* is a function $I(x,y)$ of two variables over some continuous domain A in $\Re^2$. We typically consider A to be a square. A *discrete image* is a function $I(i,j)$ of two variables over some discrete domain $G_h = \{(h \cdot i, h \cdot j) | (i,j) \in \dot{G} \subseteq Z^2\}$. Note that $G_h \subset \Re^2$ and so we can think of the discrete image as being sampled from a continuous image. We typically consider the index set $\dot{G}$ to be "square" such as the set $\{(i,j) | i,j = 0, \ldots N-1\}$. In this case h is the **grid spacing**; the smallest distance between sampled points as measured in the corresponding continuous domain. We will call such a discrete domain an **image grid**.

In single level representations discrete images are defined over image grids. In hierarchical representations image pyramids are defined over multigrids. A **multigrid representation** is a set of image grids $G_0, \ldots, G_M$ of varying resolutions $h_0, \ldots, h_M$, all relating to a single continuous domain in $\Re^2$. This relationship to a particular continuous domain implies an interlevel registration (or alignment) of the grids. Multigrid representations can be specified in many ways. The basic parameters of choice are (a) the type of grid (rectangular 1:1 aspect ratio, hexagonal, or other), (b) the relative grid spacings (scale change between levels) and (c) the *registration* (relative placement) of grids at adjacent levels. We will be using (a) square (rectangular 1:1 aspect ratio) grids with (b) a 2:1 ratio of grid spacings between levels (i.e. $h_i : h_{i+1} = 2 : 1$). Two alternatives then remain for (c) the registration of adjacent grids G_k and G_{k+1}. These choices, shown in Figure 2, are referred to as even and odd multigrids.

The distinction between even and odd pyramids is easily seen if we relate them

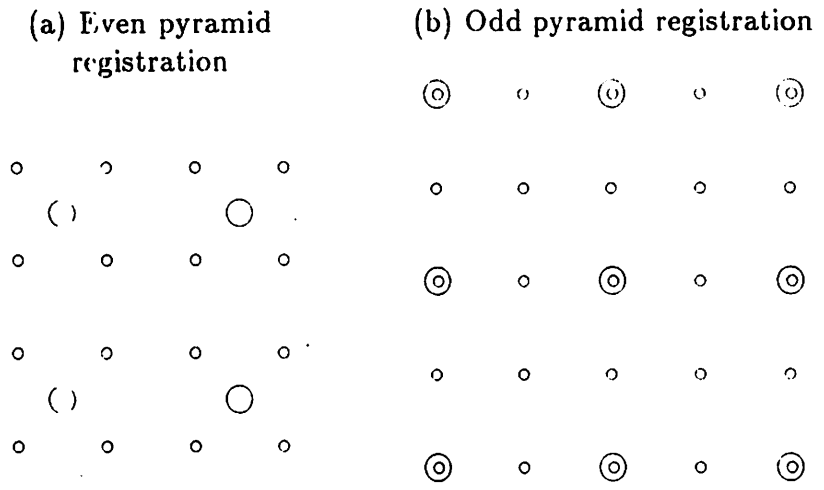


Figure 2: Even and Odd Grid Registrations

The grid points at (a) levels 1 (2×2) and 2 (4×4) of an even pyramid and (b) levels 1 (3×3) and 2 (5×5) of an odd pyramid are shown. Fine resolution grid points are indicated by '$\circ$', and coarse resolution grid points by '$\bigcirc$'.

to the geometry of recursive subdivision into square quadrants. The grid points of an even pyramid are the centers of the squares. The grid points of an odd pyramid are chosen at the (shared) corners.

The coarsest level of an even pyramid is a grid consisting of a single point representing the entire square image domain. One level down, a 2×2 grid represents the four quadrants. At level k the image grid is a $2^k \times 2^k$ square. In even pyramids, a coarse-level square (the **father**) is broken into quadrants by four of the corresponding fine-level squares (the **sons**). The grid points at two adjacent levels of an even pyramid are nowhere coincident. The even hierarchical relationship is also used for quadtrees.

The coarsest level of an odd pyramid is a grid consisting of the four corners of the entire square image domain. One level down a 3×3 grid represents the corners of the four quadrants. A finer level contains all grid points of the next coarser level plus the points between each pair of these points (the central point being redundantly specified). At level k the image grid is a $2^k+1 \times 2^k+1$ square.

on the particular hardware implementation. Pyramid data structures are stored by distributing the multilevel pixel data among the processors. The processing elements do not have individual instruction memories. Rather, instructions are broadcast to all processors. Thus, the cone belongs to the class of SIMD machines: single instruction stream multiple data stream.

Hierarchical algorithms can be broken down into a sequence of operations performed at a single level or between two levels. A simple example of this is the building of low-pass pyramids described in the next section. Operations which produce coarse images from finer ones are called **reductions**, while those that produce finer resolution images from coarse ones are called **projections**. **Single level** operations have their input and output at the same level.

2.3 Flow of Control in the Processing Cone

The particular order in which the basic single level, reduction, and projection operations are performed in a hierarchical processing algorithm determines a *flow of control* within the processing cone. In this section we mention the major types.

Fine-to-coarse algorithms have a flow of control which, as the name implies, starts at the bottom of the cone and proceeds to the top. At any stage in such an algorithm, processing is either single level or a reduction. The building of low-pass pyramids is a good example.

Coarse-to-fine algorithms have a flow of control which starts at the top of the cone and proceeds to the bottom, involving only single level processing and projections. A simple example of a coarse-to-fine cone algorithm would be the interpolation of coarse data to a finer level. The hierarchical motion analysis techniques of Chapters IV and V are coarse-to-fine.

All other algorithms fall into the class of **hybrid** algorithms, involving single level, reduction and projection operators. The multigrid relaxation algorithms of Chapter I fall into this class.

The above modes of flow of control are defined assuming that only one single level, reduction or projection operator is performed at a time. We call such an

algorithm **pyramid sequential**. If our implementation of the processing cone is done on either a single pixel processor (sequential) or a single level processor (cellular array), then no further compute time parallelism could be achieved. However, it may be the case that at some stage the operations to be performed at different levels of the pyramid are independent. We call such processing **pyramid parallel**. One example of pyramid parallel processing is the computation of a band-pass pyramid from a low-pass pyramid by the subtraction of adjacent levels.

3 Image Pyramids

3.1 Multiresolution Representation

Image pyramids are multiresolution representations of images that provide a range of coarse to fine views of the data. We want the coarse representation to be represented in a sub-sampled grid at coarse levels of the processing cone. Also, the pyramid should be easily computed in the processing cone.

What is coarse structure in an image ? We could define it as being comprised of the relatively larger objects. However, this would require exact size and location measurements, and small (coarse) grids could NOT be used capture such information. The alternative is to consider a coarsened image as one that has been blurred. Only relatively large objects can be seen in a blurred version AND their location and size is known with reduced accuracy. Such data can be captured in coarse grids.

Since blurring can be accomplished by local averaging, it can be easily computed in the processing cone. In fact, a bottom-up sequence of local-neighborhood weighted-averaging operators can compute a complete pyramid with one interlevel operation per resolution level. Other image operators which compute representative image values in a local neighborhood, such as the median filter, could be used. However, weighted averaging is a member of a larger family of image filters for which Fourier Transform methods can be used to perform spectral analysis of these operators.

In odd pyramids every other grid point at a fine level corresponds to a point in the adjacent coarse level. Thus, these points have a unique nearest grid point in the coarse level. This is not true of the other points which may have two or four nearest pixels in the coarse plane.

The choice between even and odd pyramids is largely an implementation issue. Regular decompositions [Klinger & Dyer 76, Horowitz & Pavlidis 76] and quadtrees are even because of the isomorphism between even pyramids and recursive subdivision into quadrants. Even pyramids are used in [Hanson & Riseman 74, Tanimoto & Pavlidis 75] and in all of the work on pyramid linking. Odd pyramids are used by [Burt *et al.* 83, Terzopolous 82].

2.2 The Processing Cone

A natural architecture for image pyramids is the **processing cone**, a multilayer, multiresolution organization of image planes within which inter- and intra-layer image operators are applied (see Figure 3) [Hanson & Riseman 74, Hanson & Riseman 80]. The processing cone is a highly parallel architecture composed of many simple processing elements with nearest neighbor intra-level communication links and father-son interlevel links. It is composed of levels 0 to L, each level being 2^L PE's on a side, with even registration between levels.

The exact number and bit-width of the communication links depends on the particular hardware implementation. A typical machine might include four intra-level nearest neighbor links, one father link, and four son links. Alternatively only one son link could be used. In this case non-linked sons of a father node would send data to the father through the linked son. While this takes extra cycles, the cost is still small compared to any kind of non-local single level processing. The important point is that in the processing cone architecture communication among near neighbors in the pyramid takes few cycles.

The processing elements are identical in function and each contains a small amount of local memory. The exact instruction set and size of memory depends

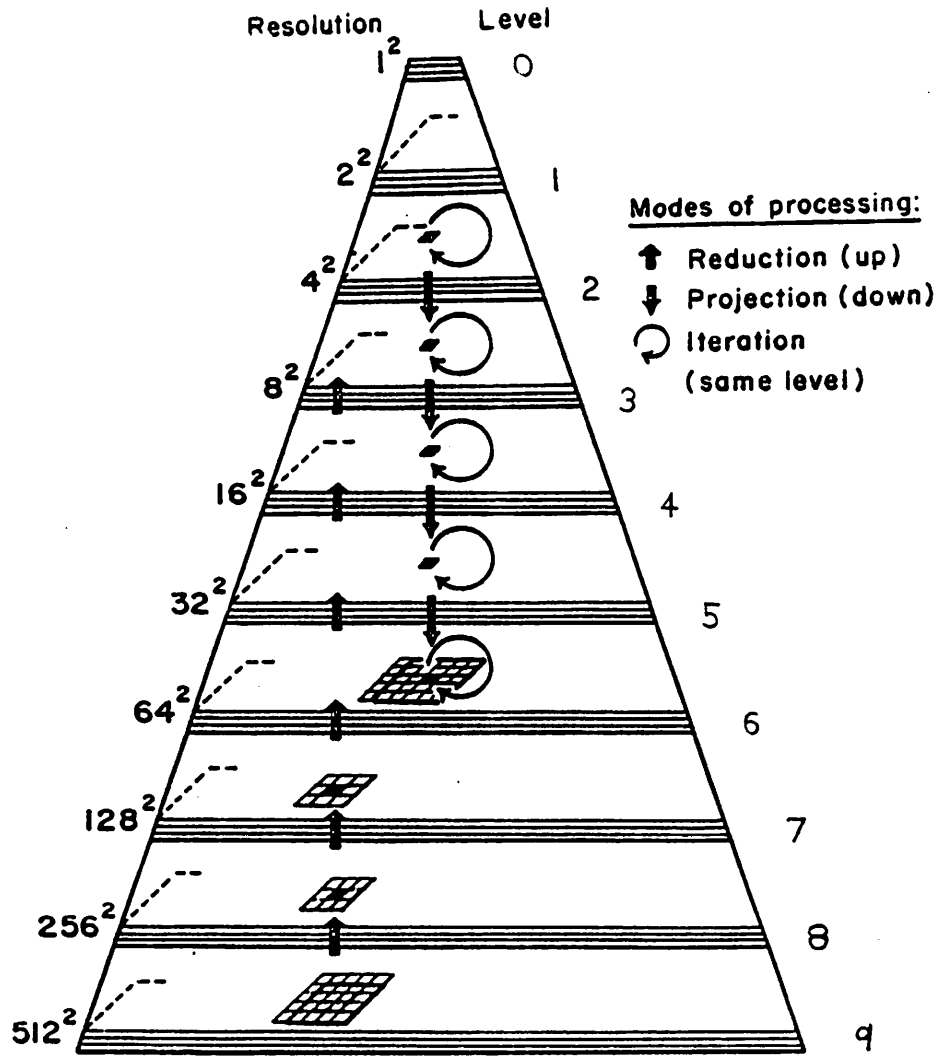


Figure 3: The processing cone

This parallel array computer is hierarchically organized into layers of decreasing spatial resolution. The particular multigrid representation used is a *square* (rectangular 1:1 aspect ratio), *two-to-one* linear ratio, *even* pyramid. Information within the cone is transformed by means of functions operating on local windows of data. Cone algorithms are composed of these parallel functions applied in one of three processing modes: reduction (up the cone), projection (down the cone), and iteration (at the same level).

3.2 Image Filters

Many of the basic operators we use in the pyramid are parallel (cellular) neighborhood computations in which all pixels get a value which is the weighted sum of the values at neighboring pixels. Examples include smoothing, edge detection, and interpolation. These image operators are essentially *linear shift invariant* operators or "filters". They are not fully shift invariant because of edge effects — border pixels have neighborhoods which fall off the image and hence require special handling. These operators can be represented as *convolutions* (or alternatively *correlations*) in the image space and also as *frequency filters* in the spatial frequency domain. Convolutions are a formal way of defining the weighted sum of neighboring values which we use in our algorithms.

We represent convolution masks as arrays in square brackets such as $m = [\, w_{-i,-j}]$ (the "negative" coordinate system is explained below) so that the convolution of an image f with the mask m is defined by

$$m * f(k,l) = \sum_{i,j} w_{ij} f(k-i, l-j) \quad (1)$$

$$= \sum_{i,j} w_{-i,-j} f(k+i, l+j) \quad (2)$$

To simplify the writing of convolution masks such as the following "edge detector"

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

we adopt the following conventions:

1. Only mask values in the smallest rectangle containing the non-zero support of the mask are shown. The rectangle of points is $\{-1,0,1\} \times \{-1,0,1\}$ for the above mask.

2. Scale factors may be shown outside the braces. The scale factor is 1/4 for the above mask.

3. The central weight at the origin (0,0) is shown in **bold** unless the mask is odd by odd and the central weight is the center of the displayed values. (The central 0 need not be bolded in the above mask.)

4. Weights are shown in the $(-m, -n)$ coordinate system.

5. $[\cdot]^T$ denotes the transpose of a mask. For example $[1\ 2\ 3]^T \triangleq \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$

Convention 4 makes it easier to picture the action of the mask by giving it the appearance of a correlation operator (see Equation 2). An output pixel is computed by using the mask as a "stencil", that is placing the mask "over" the input pixel with off-center entries in register with the corresponding pixels in the neighborhood of the input pixel. The values at those pixels are multiplied by the respective mask weights and the results are summed to give the value of the output pixel. More formally, we can describe the convolution of a mask and an image at a particular (output) pixel—as formulated in equation 1—as involving (1) "flipping" or reflecting the mask about the origin, i.e. $(x, y) \leftarrow (-x, -y)$, (2) placing the origin of the mask over the given pixel, (3) pointwise multiplying the corresponding mask weight and image pixel values, and (4) summing all these terms. Convention 4 allows us to write the mask as it looks after it has been "flipped". Steps (2) through (4) define a correlation operator.

Convention 5 allows us to write tall thin masks "inline". Note the difference in appearance between this transpose and the standard matrix transpose. This is due to the different indexing "directions", namely, rows down and columns right-to-left for matrices ($\searrow$); $-m$ left-to-right and $-n$ down for convolution masks ($\swarrow$). The arrows thus become the axes of reflection for transposition.

3.3 Pyramid Filters

Pyramid filters are interlevel image operators. Given a discrete image at level L, **reductions** produce a coarse images at level $L-1$ and **projections** produce finer

images at level $L + 1$ (see Figure 3).

3.3.1 Reductions

The simplest reduction—sometimes called *injection*—is a subsampling of the finer level, in which the value a father pixel receives comes from one of its sons. More complicated reductions compute a coarse pixel from a number of pixels in the corresponding neighborhood in the finer image. These operators are equivalent to applying some image filter to the fine-level image followed by a subsampling. Of course pixels not "chosen" in the subsampling need not have the image filter performed there. We will represent these reductions by the masks of the image filter that is applied at the fine level noting that the spacing between mask points corresponds to the fine-level spacing. Filtering is easily restricted to only those pixels that will be sampled by noting the 1-1 correspondence between coarse output pixels and filtering locations.

For injection in odd pyramids, the choice of which son to sample is straightforward since only one corresponds spatially to the father. For even pyramids there is no "natural" choice of sampling of the four equally distant sons. Thus, the simplest reduction we most often see used in even pyramids uses a 2×2 weighting mask (see Section 4.3). Note that this is a form of interpolation, in that we can consider the result as interpolating the value at a point midway between the four sons.

3.3.2 Projections

Projections always involve interpolation in that a finer grid must be "filled in" with data points. The interpolated values at points in the (fine) input image are computed as a weighted sum of values of pixels in neighborhoods of the (coarse) input image. If we consider the placement of coarse pixels relative to given fine pixels, we see that different finer pixels see different relative configurations of coarse pixels above it. For even pyramids there are four possible configurations, since the coarse pixel nearest a given fine pixel (the father) can be located in one of four directions (see Figure 2). These configurations are identical up to rotation by 90

degrees about the central (fine) pixel.

We represent projection masks as convolution masks applied at the coarse level. The registration of the mask and the "output" pixel in the finer grid is shown by bolding the weight(s) applied to the nearest father pixels, unless, as in Convention 3 on page 36, the mask is symmetrically placed about the central pixel. Figure 4 demonstrates this for both even and odd grid registrations.

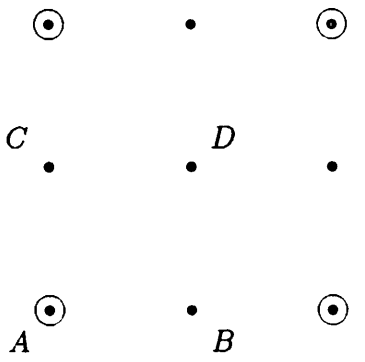
4 Building Image Pyramids

In this section, we introduce the basic single level image operators and interlevel pyramid operators which we will use to build multiresolution image representations. Transfer functions (spatial frequency representations) of these operators are presented as a tool in analyzing and comparing them. Specifically, we address the use of low-pass filtering to reduce aliasing at coarse levels, and the use of interlevel differencing to obtain band-pass data.

4.1 Hierarchical Frequency Spaces

An image sampled at the discrete set of points $\{(ih, jh) | -\infty < i, j < +\infty\}$ has a Fourier Transform which is periodic and completely specified by the values in the **baseband** $B_h \triangleq [-\frac{1}{2h}, \frac{1}{2h}]^2 = \{(s,t) | -\frac{1}{2h} \leq s, t \leq \frac{1}{2h}\}$ (see Appendix A, Section 7). In the frequency domain, B_h is a square centered at the origin with length $\frac{1}{h}$ on a side. In the processing cone, we let the sample spacing at the lowest (highest resolution) level be $h_L = 1$ by convention. Proceeding up the cone, the sample spacings are $h_{L-1} = 2$, $h_{L-2} = 4$, etc. The corresponding basebands $B_1, B_2, B_4, \ldots$ are concentricly placed, decreasing in (linear) size by a factor of 1-to-2. In Figure 5, we show the basebands for the three bottom levels of a pyramid.

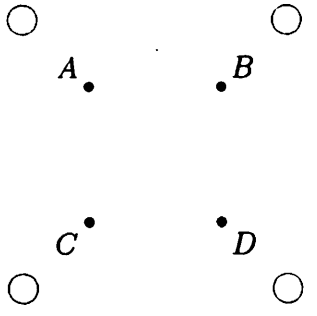
Figure 4: Representation of projection masks



$$A \leftarrow [1] \qquad B \leftarrow \tfrac{1}{2}[1\ 1]$$

$$C \leftarrow \tfrac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad D \leftarrow \tfrac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(a) Bilinear interpolation masks for projection in an even pyramid.
There are four possible masks. Each case is shown with the relative placement of the fine level pixel (on the left) and the projection masks (on the right).



$$A \leftarrow \tfrac{1}{16}\begin{bmatrix} \mathbf{9} & 3 \\ 3 & 1 \end{bmatrix} \qquad B \leftarrow \tfrac{1}{16}\begin{bmatrix} 3 & \mathbf{9} \\ 1 & 3 \end{bmatrix}$$

$$C \leftarrow \tfrac{1}{16}\begin{bmatrix} 3 & 1 \\ \mathbf{9} & 3 \end{bmatrix} \qquad D \leftarrow \tfrac{1}{16}\begin{bmatrix} 1 & 3 \\ 3 & \mathbf{9} \end{bmatrix}$$

(b) Bilinear interpolation masks for projection in an odd pyramid.
There are four possible masks. The bolded **9**'s show that those are the weights corresponding to the nearest father pixel.

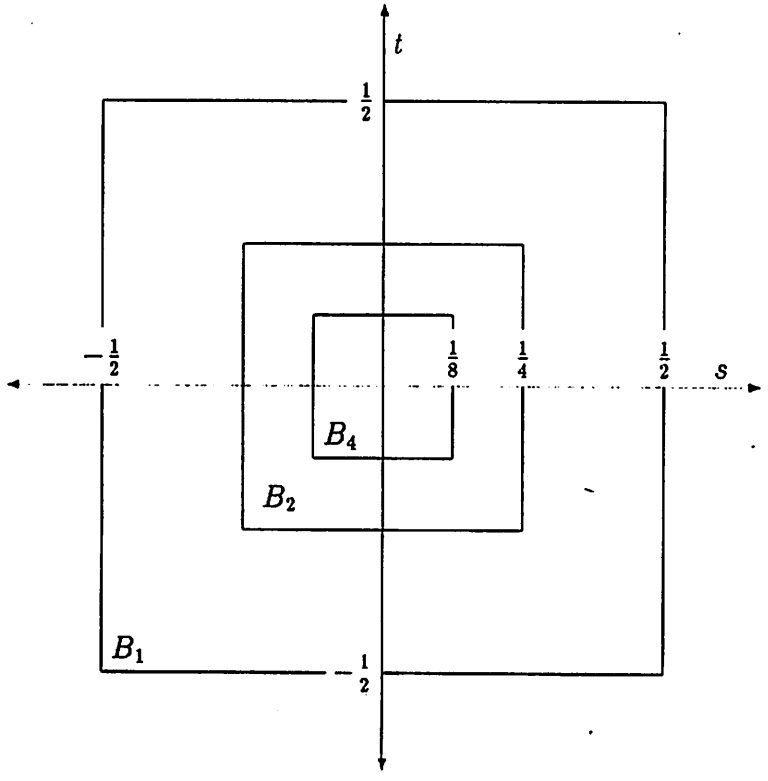


Figure 5: Hierarchical Frequency Space

$B_1 = [-\frac{1}{2}, \frac{1}{2}]^2$ is the baseband for the finest level sampling grid with spacing $h_L = 1$. $B_2 = [-\frac{1}{4}, \frac{1}{4}]^2$ is the baseband at level $L-1$ where $h_L = 2$. $B_4 = [-\frac{1}{8}, \frac{1}{8}]^2$ is the baseband at level $L-2$ where $h_L = 4$.

4.2 Sub-sampling and Aliasing

In Section 3.3, injection was defined as the creation of a coarsened image by sub-sampling. It is likely that injection will not be a satisfactory operator for building image pyramids. The use of data values from only 1 in 4 pixels from the finer level is susceptible to problems such as the over emphasis of noise spikes, or the dropping of small features. These problems may get worse as further sub-sampling is used to compute coarser levels. One way to quantify such problems is to consider the effect of injection on the frequency content of the image data. We can do this because sub-sampling in the spatial domain has a counterpart in the frequency domain, namely, the aliasing of high frequencies which show up as low frequencies (Appendix A, Section 8).

When we sub-sample data in the finest level of processing cone, the coarser grid has a baseband B_2 one half the size (linearly) of the baseband B_1 of the finer grid (see Figure 5). We call the boundary of B_2 the **Nyquist boundary** within B_1 for the 1-of-2 sub-sampling. High spatial frequencies in the finer level outside of this Nyquist boundary (i.e. $(s,t) \in B_1 - B_2$) show up in the coarse subsampled image as low frequencies inside the Nyquist boundary. (The actual relationship between these frequencies is developed in Appendix A, Section 8.2.) This added low frequency content is unrelated to the actual content of the unsampled data. This aliasing introduces false structure into an image, such as Moire patterns. Low-level motion detection algorithms can be misled by this false structure.

4.3 Low-Pass Filtering

Low-pass filters pass low frequencies and attenuate high frequencies. A **low-pass filter** has a gain near unity at low frequencies, gain near zero at high frequencies, and a decreasing gain in the intermediate frequencies. The **transition zone** is the set of intermediate frequencies within which the gain drops from near unity to near zero. For a 1D filter, the **cutoff frequency** is that frequency (within the transition zone) at which the gain is 1/2. For a 2D filter, the **cutoff boundary** is the locus

of frequencies at which the gain is 1/2.

Aliasing can be reduced if image data is low-pass filtered prior to sub-sampling. "Ideal" low-pass filtering would pass (gain=1) all spatial frequencies within the coarse baseband and reject (gain=0) all those outside. That is, the cutoff boundary is at the Nyquist boundary and the transition zone vanishes. The convolution mask that performs such filtering is a sampled product of 1D sinc functions, namely, $\frac{1}{4}\mathrm{sinc}_2(i/2, j/2) \triangleq \frac{1}{4}\mathrm{sinc}(i/2)\mathrm{sinc}(j/2)$ (see Appendix A, Section 8.3). This function has global support: it is defined over all of the spatial domain. This is definitely unsuited to a processing cone implementation in which only local transfers can be performed quickly. In general, to get a low-pass filter with a small transition zone (a "sharp" cutoff) at the Nyquist boundary, a relatively large spatial filter is needed.

Low-pass filtering can be done by averaging the values in a local neighborhood. This is equivalent to filtering with a mask having a neighborhood of support containing n pixels, all with the value $1/n$. The simplest such operator we can use in an even pyramid is the low-pass filter obtained by taking the average of the four sons of a coarse pixel (as in [Tanimoto & Pavlidis 75] and [Wong & Hall 78]):

$$\frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

The transfer function of this filter is shown in Figure 6. Only the amplitude is shown, since the phase shift is always zero. Figure 6a shows the transfer function over the entire baseband. Figure 6b is a cutaway view through the s-axis. The profile of the cut is the 1D transfer function of the 1D filter $\frac{1}{2}[1\ 1]$. High frequencies are not very strongly attenuated. The average gain outside of the Nyquist boundary is .270 (see Table 1).

The transfer function of a 4×4 averaging filter is shown in Figure 7. Again only the amplitude is shown, both over the full baseband and in cutaway. The values go negative in some places because we use the convention of representing phase shifts of π radians as negative amplitudes. (This convention is described in Appendix A, Section 6.) Thus, we need not show the phase data since it is incorporated into the amplitude plot. This mask does have more high frequency

attenuation with an average gain of .0806 outside the Nyquist boundary, but it also has a much increased low frequency attenuation with an average gain of .360 inside the Nyquist boundary. In the next section we show how better filter characteristics can be obtained while remaining in the confines of a 4×4 filter. In Table 1 on page 51 qualitative comparisons of the transfer characteristics are made between these and other low-pass filters.

4.4 Discrete Gaussians

Our requirement for computational simplicity favors convolution masks with small neighborhoods of support. This can be referred to as *localization in space*. The elimination of high frequencies implies a concentration of passed frequencies near zero. This can be referred to as *localization in frequency*. These are mutually conflicting requirements. The sinc filter, with its perfect low-pass characteristics passing only those frequencies below the cutoff, is completely localized in frequency. However, since it has infinite support it is not at all localized in space. On the other hand, the 2×2 averaging filter is very localized in space but not in frequency.

The tradeoff between localization in space vs. frequency is quantitatively given by the *uncertainty relation* [Bracewell 78, p.160]: for the function $f(x)$ and its Fourier Transform $F(s)$, if we let $(\triangle x)^2$ and $(\triangle s)^2$ be the variances of $|f(x)|^2$ and $|F(s)|^2$ respectively, then $\triangle x \triangle s \geq 1/4\pi$. The variances of the squared moduli are width measures equal to the mean-square departure from the centroids of $|f(x)|^2$ and $|F(s)|^2$. If $f(x)$ is a Gaussian function, then $F(s)$ is also, and $\triangle x \triangle s = 1/4\pi$. This led Marr and Hildreth to propose the Gaussian function for the purpose of obtaining optimal localization in space and frequency in a low-pass image filter [Marr & Hildreth 80].

Two-dimensional Gaussians are both separable and rotationally symmetric. The latter property is important for low-level image processing in which there are no preferred orientations for the image structure. The former property allows for reduced computations, when the Gaussian is applied as two 1D operators.

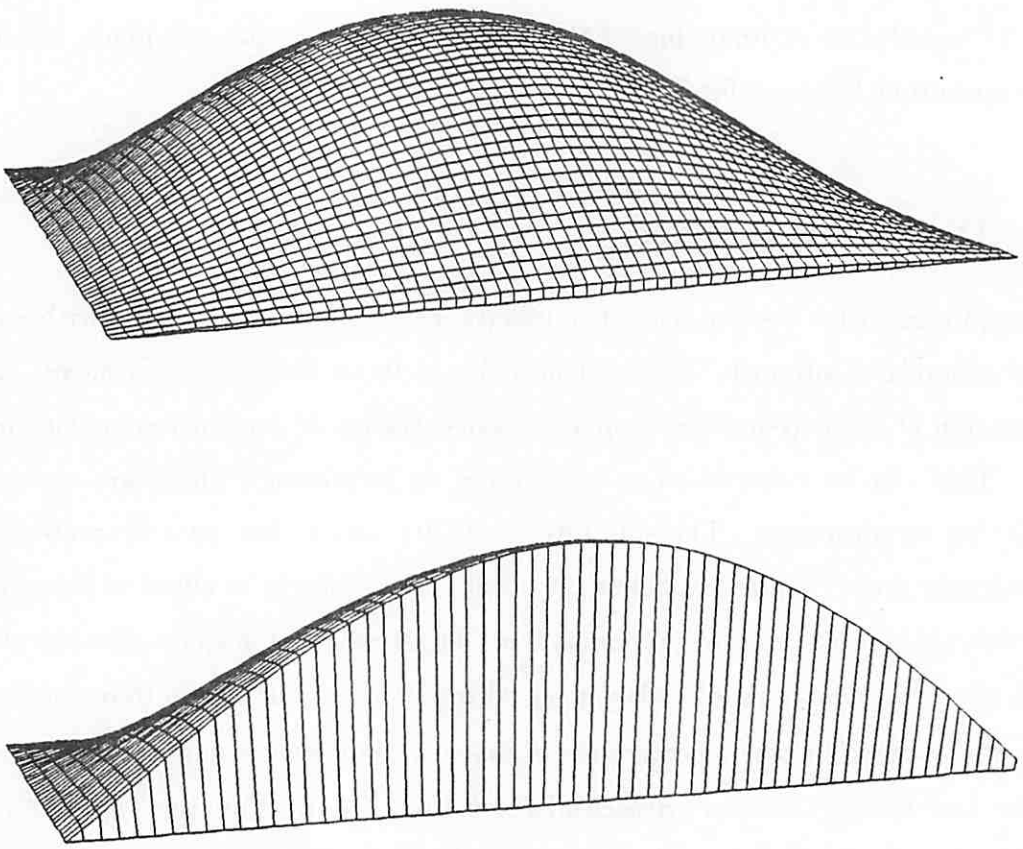


Figure 6: Transfer function of 2×2 averaging filter

Frequency response of G_1. Attenuation is complete at the boundary of the base-band, i.e. at the high frequencies $(\pm 1/2, \cdot)$ and $(\cdot, \pm 1/2)$. The maximum gain is 1 for DC.

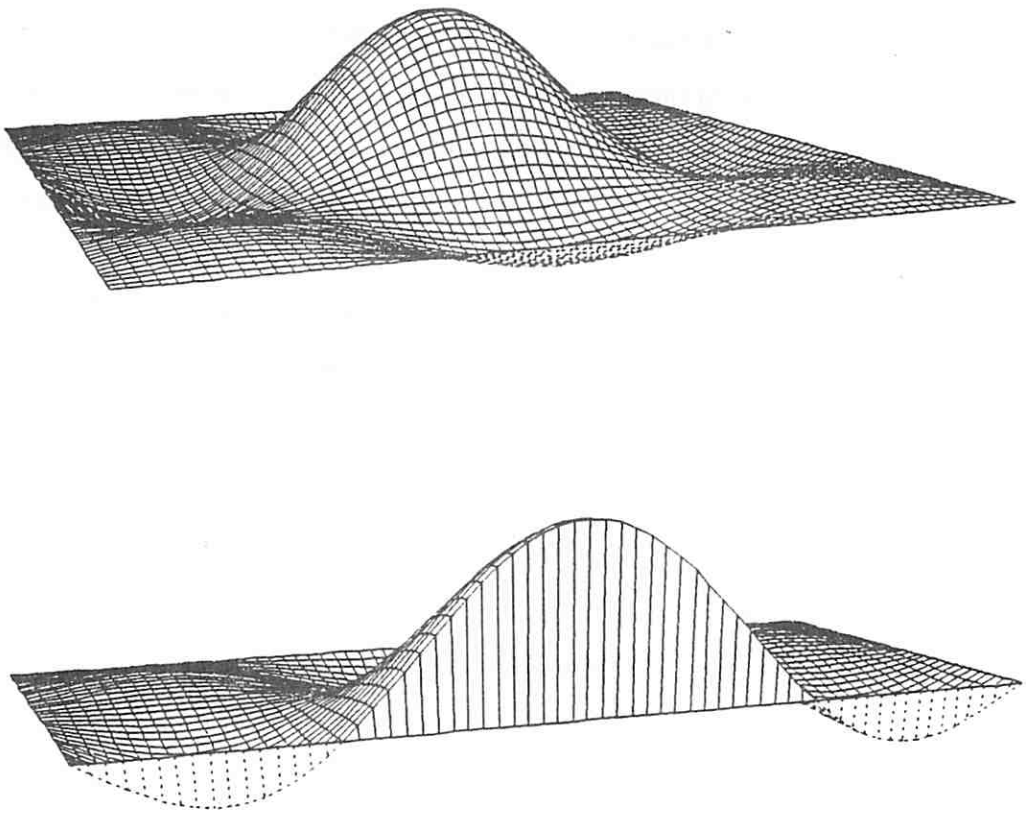


Figure 7: Transfer function of 4×4 averaging filter

Gain is zero along the Nyquist boundary, but increases outside that boundary to a maximum of .272 (with a phase shift of $-\pi$).

A discrete Gaussian with finite support could be obtained by sampling a continuous Gaussian function, and setting to zero those values outside some "window" about the origin. We call this the **sampling-and-truncation** method. Truncation is equivalent to multiplying the Gaussian function by a weighting function which is zero-valued outside the truncation window. Simple truncation is given by a weighting function which takes the value 1.0 everywhere inside the window. More complicated weighting functions with weights that taper off to zero as the distance from the origin increases can be used to get better high frequency attenuation (see for example [Oppenheim & Schafer 75, Section 5.5] and [Dudgeon & Mersereau 84, Section 3.3]).

Crowley uses simple truncation [Crowley 81] in a window of radius 4 about the origin. We show one quadrant of his 49-point filter, the full mask being given by horizontal and vertical symmetry.

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \mathbf{.081248} & .063276 & .029890 & .008564 & .001488 \\ \cdots & .063276 & .049280 & .023278 & .006669 & .0 \\ \cdots & .029890 & .023478 & .010996 & .003150 & .0 \\ \cdots & .008564 & .006669 & .003150 & .0 & .0 \\ \cdots & .001488 & .0 & .0 & .0 & .0 \end{bmatrix}$$

The transfer function of this filter is shown in Figure 9. Crowley chose his filter to have very high attenuation outside the Nyquist boundary. The Nyquist boundary for his sub-sampling scheme surrounds the "diamond"-shaped baseband $B_{\sqrt{2}h} \triangleq \{(s,t)|\ |s|+|t| \leq 1/2h\}$. Applying his attenuation criterion to the smaller baseband B_2 would likely call for a yet larger convolution mask.

A major disadvantage of designing Gaussian low-pass filters by sampling and truncation is the use of real arithmetic. Rational approximations can be used, but the integers will in general still be very large. The next method defined produces Gaussian filters with relatively small weights.

First, we define a family of discrete one-dimensional Gaussians formed by

iterated convolution with the basic averaging mask $\frac{1}{2}[1\ 1]$:

$$\begin{aligned} g_0 &\triangleq [1] \\ g_k &\triangleq \underbrace{\frac{1}{2}[1\ 1] * \cdots * \frac{1}{2}[1\ 1]}_{k} \\ &= \frac{1}{2^k}\underbrace{[1\ 1] * \cdots * [1\ 1]}_{k} \end{aligned}$$

The first seven masks are:

$$\begin{aligned} g_0 &\triangleq [1] \\ g_1 &\triangleq \frac{1}{2}[1\ 1] \\ g_2 &\triangleq \frac{1}{4}[1\ 2\ 1] \\ g_3 &\triangleq \frac{1}{8}[1\ 3\ 3\ 1] \\ g_4 &\triangleq \frac{1}{16}[1\ 4\ 6\ 4\ 1] \\ g_5 &\triangleq \frac{1}{32}[1\ 5\ 10\ 10\ 5\ 1] \\ g_6 &\triangleq \frac{1}{64}[1\ 6\ 15\ 20\ 15\ 6\ 1] \\ g_7 &\triangleq \frac{1}{128}[1\ 7\ 21\ 35\ 35\ 21\ 7\ 1] \end{aligned}$$

Note that the weights in these masks are equal to the rows of Pascal's triangle (up to a power-of-two multiplicative factor). They are easily computable because the weights can be represented as fractions of relatively small numerators and a common power-of-two denominator. The division by 2^k can be performed by a simple shift in the final step. The convolutions can also be computed as repeated pairwise addition of neighboring values (convolutions with [1 1]) followed by shifting to divide. *This involves* NO *multiplication!*

Two-dimensional discrete Gaussians can now be defined as the convolution of two one-dimensional Gaussians, one for each spatial direction. Thus, we get functions $G_k \triangleq g_k * g_k^T$ which are separable and easy to compute. These convolution masks provide a family of low-pass filters of increasing spatial size and decreasing cutoff frequency.

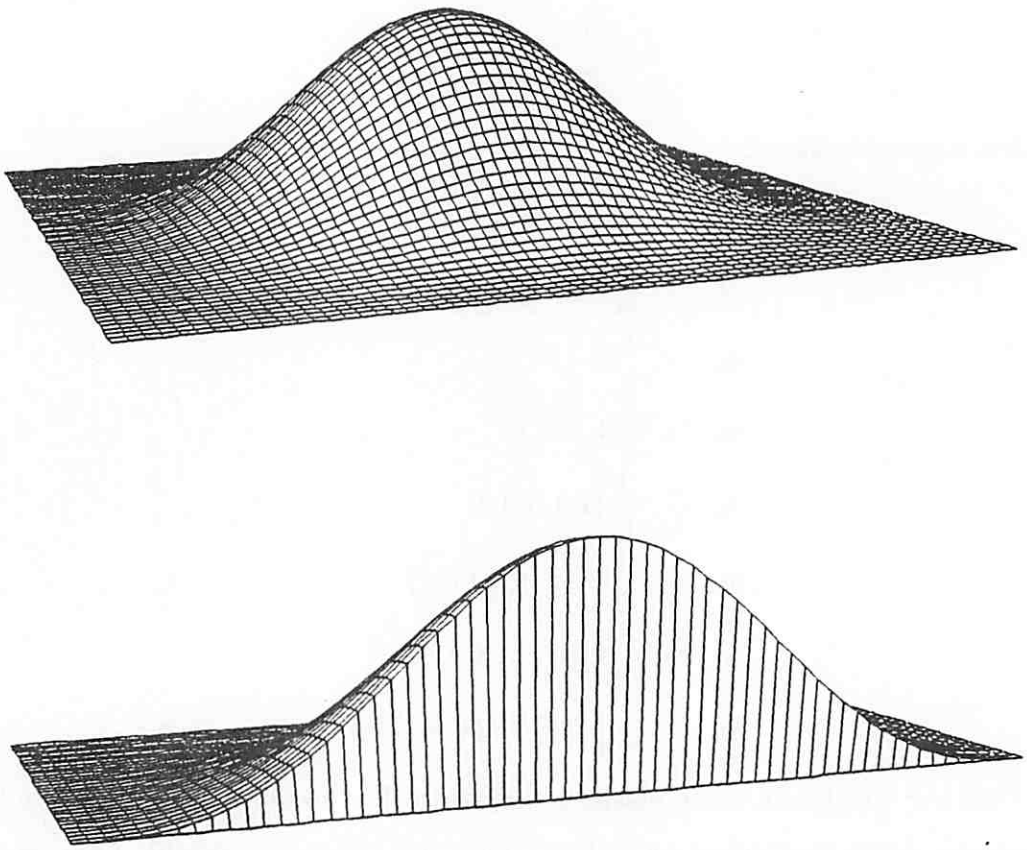


Figure 8: Transfer function of G_3 low-pass filter

G_3 is a 4×4 weighted average equal to $\frac{1}{8}[1\ 3\ 3\ 1] * \frac{1}{8}[1\ 3\ 3\ 1]^T$. A graph of the cutoff boundary of G_3 appears in Figure 10.

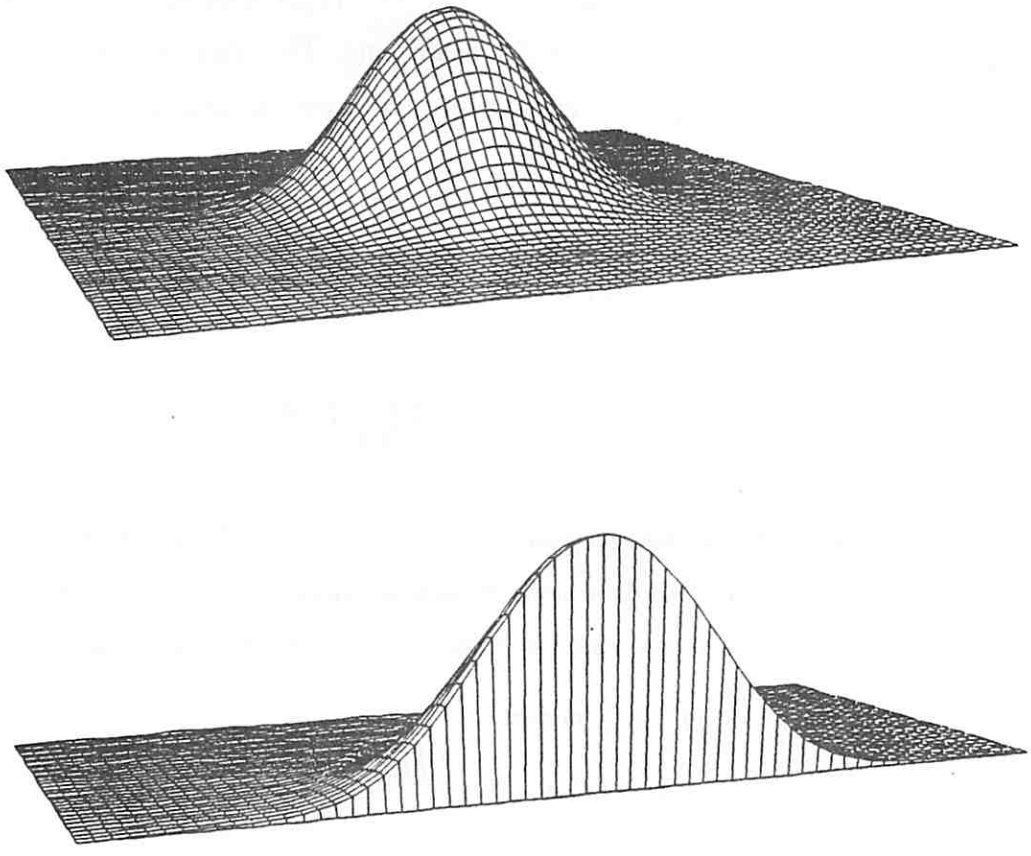


Figure 9: Transfer function of Crowley's low-pass filter

Crowley's low-pass filter is a 49 point filter and is shown on page 46. A graph of the cutoff boundary appears in Figure 10.

In Table 1 the filtering characteristics of some low-pass filters are compared. The average gain over low and high frequency ranges are shown for one and two-dimensional filters. In all but one row of the table, the two-dimensional filters are separable, being outer products of $1 \times n$ and $n \times 1$ versions of the one-dimensional filter in that row. For one-dimensional filters, the low frequencies are those below $|s| = 1/4$, the Nyquist frequency for 1-of-2 subsampling. The highs are from $|s| = 1/4$ to $|s| = 1/2$. The average gains over lows and highs are defined as:

$$P_L^{1D} \triangleq \frac{\int_{-1/4}^{+1/4} F(s)\,ds}{\int_{-1/4}^{+1/4} 1\,ds} = 4\int_0^{1/4} F(s)\,ds$$

$$P_H^{1D} \triangleq \frac{\left(\int_{-1/2}^{-1/4} + \int_{1/4}^{1/2}\right) F(s)\,ds}{\left(\int_{-1/2}^{-1/4} + \int_{1/4}^{1/2}\right) 1\,ds} = 4\int_{1/4}^{1/2} F(s)\,ds$$

For two-dimensional filters, the low frequencies are those within the Nyquist boundary for 1-of-2 subsampling in both directions, that is within $B_2 = [-\frac{1}{4}, \frac{1}{4}]^2$. The highs are those in $B_1 - B_2$. The average gains over lows and highs are defined as:

$$P_L^{2D} \triangleq \frac{\iint_{B_2} F(s,t)\,dsdt}{\iint_{B_2} 1\,dsdt} = 4\iint_{B_2} F(s,t)\,dsdt$$

$$P_H^{2D} \triangleq \frac{\iint_{B_1-B_2} F(s,t)\,dsdt}{\iint_{B_1-B_2} 1\,dsdt} = \frac{4}{3}\iint_{B_1-B_2} F(s,t)\,dsdt$$

If the low-pass filter is separable, then its transform is also, and it is easy to show that:

$$P_L^{2D} = P_L^{1D} P_L^{1D}$$

$$P_H^{2D} = P_H^{1D}\left[\frac{2}{3}P_L^{1D} + \frac{1}{3}P_H^{1D}\right]$$

For the one-dimensional masks, the cutoff frequencies s_c are also shown in Table 1. The cutoff boundaries of G_1, G_3, G_5, and Crowley's filter are shown in

1D Filters					2D Filters		
Filter	s_c	P_L^{1D}	P_H^{1D}	M_H^{1D}	Filter	P_L^{2D}	P_H^{2D}
Ideal	.25	1.0	0.0	0.0	Ideal	1.0	0.0
g_1	.333	.900	.373	.707	G_1	.810	.270
$\frac{1}{4}[1\ 1\ 1\ 1]$	.154	.600	.176	.272	4×4 Avg.	.360	.0806
g_3	.208	.750	.098	.353	G_3	.563	.0525
g_5	.164	.645	.034	.177	G_5	.416	.0149
	—	—	—	—	Crowley's filter	.308	.0077
g_7	.139	.569	.013	.088	G_7	.324	.0050

Table 1: Comparison of Low-Pass Filters

For the one-dimensional filters: s_c is the cutoff frequency; P_L^{1D} is the average gain below Nyquist; P_H^{1D} is the average gain above Nyquist; M_H^{1D} is the maximum gain above Nyquist.

For the two-dimensional filters: P_L^{2D} is the average gain within the Nyquist boundary; P_H^{2D} is the average gain outside the Nyquist boundary.

Figure 10. For the separable two-dimensional masks, the cutoff boundaries pass through the frequencies $(\pm s_c, 0), (0, \pm s_c)$ where s_c is the cutoff frequency of the corresponding one-dimensional filter.

The table shows the superiority of the 4×4 filter G_3 over the simple 4×4 averaging filter for anti-aliasing. G_3 has much less attenuation of the low frequencies within the Nyquist boundary, and significantly greater attenuation outside that boundary. Also compare Crowley's filter a 9×9 filter with 49 non-zero coefficients to G_7 a 7×7 filter. We see slightly better performance on each side of the Nyquist boundary: less attenuation inside, more outside. G_7 is also much more easily calculated because it is separable, with the 1D coefficients being small integers and the normalizing divisor a power of 2. The cutoff boundary for G_7 is not shown in Figure 10 since it is very close to that of Crowley's filter.

Given the choice of low-pass filters that the family of discrete Gaussians gives us, which should be used in building a pyramid? To the extent that aliasing must be reduced, larger filters are needed to provide maximum attenuation of high fre-

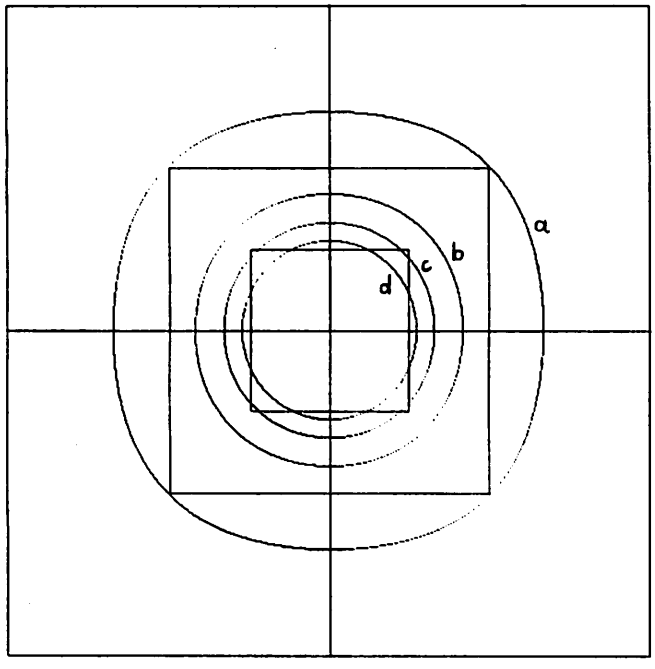


Figure 10: Cutoff boundaries of low-pass filters

The cutoff boundaries four filters are shown: (a) G_1; (b) G_3, (c) G_5, and (d) Crowley's low-pass filter. The outer rectangle is the boundary of the baseband. The inner rectangle is the Nyquist boundary for 1-of-2 horizontal and 1-of-2 vertical subsampling.

quencies. In general, images have less high frequency content than low frequency. On the other hand, larger filters have higher attenuation of the mid-range frequency components which comprise the high frequencies in the baseband of the coarse sampling grid. Too large a filter will introduce excessive smoothing in the coarse image. The other disadvantage of large filters is the added computational cost they bring. These factors all relate to the low-pass filtering and subsampling used to go up one level in the cone. When we consider the overall pyramid building process, a final factor arises which influences our choice of filter: the *equivalent net low-pass filter* which results from the bottom-up building of pyramids with a given low-pass filter. This equivalent filter is defined in the next section and used to further constrain our choice of pyramid building filter.

4.5 Bottom-Up Low-Pass Filtering

Each level of a low-pass pyramid could be computed by sub-sampling the suitably low-pass filtered "base" image. For example, to go up two levels 1-of-4 subsampling (in both the row and column directions) is done. This method is computationally costly since low-pass filters with cutoff boundaries close to the Nyquist boundaries at coarser levels are very large. For example, in the one-dimensional case, to obtain cutoff frequencies at approximately the frequencies $s_c = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$, the filters $g_2, g_9, g_{36}, g_{144}$ must be used, having the respective widths 3, 10, 37, and 135.

A much more efficient way is to build the second finest level (one level up) this way and then build the next higher from that, etc. Going up two levels then involves two stages of filtering followed by 1-of-2 subsampling, first at the high resolution then at the lower.

4.5.1 Spread Masks

Suppose G_1 (2×2 averaging) is used as the low-pass filter. In the first stage G_1 is applied at level L. In the second stage, G_1 is applied at level $L-1$ where the grid spacing is $h_{L-1} = 2$. This is equivalent to a using a mask at level L in which the

weights of G_1 have been "spread" out so that they are then 2 grid spaces apart (in both directions) and filling in the intervening points with 0.0 weights. This mask would be [1 0 1]*[1 0 1]^T. The weights of 0.0 correspond to the grid points in the fine grid that are not included in the coarse grid.

More formally we define the **spread** of a filter mask $m(i,j)$ as:

$$S_k[m](i,j) \triangleq \begin{cases} m(\hat{i},\hat{j}) & \text{if } 2^k\hat{i} = i \text{ , } 2^k\hat{j} = j \text{ , and } m(\hat{i},\hat{j}) \text{ is defined} \\ 0.0 & \text{otherwise} \end{cases}$$

In this definition the parameter k allows for generalizing to an arbitrary number of levels in the cone. A mask m applied at level $L-k$ corresponds to a spread mask $S_k[m]$ applied at level L. For example $S_2[G_1] \triangleq$ [1 0 0 0 1]*[1 0 0 0 1]^T.

4.5.2 Net Convolution Masks

Applying a mask consecutively at two levels going up the cone with intermediate 1-of-2 subsampling is equivalent to applying both the mask m and the spread mask $S_1[m]$ with a final 1-of-4 subsampling. More generally, if filtering and subsampling is applied k times, this is equivalent to applying m, $S_1[m]$,..., $S_{k-1}[m]$ and finally subsampling by 1-of-2^k. Since convolution is associative, the application of multiple image filters by repeated convolutions is equivalent to convolution by the one mask that is the net convolution of the individual masks. Also, since convolution is commutative, the order in which we convolve the individual masks is irrelevant. So we have:

$$S_{k-1}[m] * (S_{k-2}[m] * \cdots * (S_1[m] * (m * I))\ldots) = (S_{k-1}[m] * \cdots * S_1[m] * m) * I$$

We define the **net convolution mask** as the convolution of the individual masks.

For example, consider two-by-two averaging reduction when applied at two consecutive levels. The net convolution of G_1 and $S_1[G_1]$ is shown.

$$\frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \frac{1}{4}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

It is equivalent to a four-by-four averaging reduction up two levels in the cone. Continued application of two-by-two averaging up the cone always results in "flat" net convolution masks (i.e., unweighted averaging).

Burt has approached the interlevel low-pass filter design by considering the shape of the net convolution that derives from a given mask and its corresponding spread masks [Burt 81]. He was interested in finding low-pass filters for which the net convolution approximates a Gaussian. He studied one-dimensional filters of length four and five. By starting with some simple constraints such as symmetry and weights summing to one, both cases were reduced one-parameter families of filters.

In the length four case, the family of masks is given by $[b\ a\ a\ b]$ where $b = \frac{1}{2} - a$ and $\frac{1}{4} \leq a \leq \frac{1}{2}$. In Burt's analysis of these filters, the best mean square Gaussian approximation to the net convolutions is obtained when the parameter a is approximately .37. The mask $g_3 \triangleq \frac{1}{8}[1\ 3\ 3\ 1]$ corresponds to the case $a = .375$, $b = .125$. This mask is very close to best (if not identical). Since Burt's analysis is empirical, calculating the fit for various choices of a, we do not precisely know the exact best value. In any case, any slight variation of a away from .375 will require using larger weights. For example, when $a = .37$ the mask is $\frac{1}{100}[13\ 37\ 37\ 13]$.

If we consider Burt's analysis of filters of width 4, it can be seen that the discrete Gaussian G_3 is close to the optimal choice of a filter which provides a Gaussian shaped net convolution mask. The same cannot be said of G_1 or of G_2. We saw above that the net convolution for G_1 is an unweighted averaging mask. The net convolutions for g_2 are always "triangular" in shape, e.g. [1 2 1]*[1 0 2 0 1]= [1 2 3 4 3 2 1].

4.5.3 Transfer Functions

In this section we examine the transfer functions of spread masks and of some net convolution masks. The Stretching Theorem (Appendix A, Section 4.4) tells us that if the mask $m(i,j)$ has the transform $M(s,t)$, then the spread mask $S_k[m](i,j)$ has the transform $M(2^k s, 2^k t)$. Thus, while the mask "spreads out" by a factor of 2^k, its

transform squeezes in by the inverse factor $1/2^k$. In Figure 11 the transfer function of the one level spread of G_3 is shown. In Figure 12 we show the transfer function of the net convolution of G_3 and its one level spread $S_1[G_3]$. By the Convolution Theorem, this transfer function is equal to the (pointwise) multiplication of the transfer functions of G_3 and $S_1[G_3]$ (Figures 8 and 11 respectively). The cutoff boundary of this net convolution is shown in Figure 13 along with that of G_3.

The transfer functions of spread masks have high gains at the highest frequency. However, the transfer function of the basic (unspread) mask has low gain at these frequencies. These two gains multiply together in the transfer function of the net convolution. The exact interaction of these gains determines just how much attenuation is achieved at high frequencies. Certainly if the basic mask has a very low gain this insures that the net convolution does also. However, it does appear that for some masks a neat compromise is drawn in that the basic mask has very low gain where the spread mask goes high, and has a moderate gain where the spread mask itself has low gain. Thus, the basic mask need not be very low everywhere outside of the Nyquist boundary.

4.6 Building Low-Pass Pyramids

Given the analysis of low-pass filtering in the preceding sections we have used the following kernel in most of our experiments

$$G_3 \triangleq \frac{1}{8}[1\ 3\ 3\ 1] * \frac{1}{8}[1\ 3\ 3\ 1]^T = \frac{1}{64}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

The frequency response of G_3 is shown in Figure 8. The filter G_3 provides a compromise between the retention of low frequencies and the attenuation of high frequencies as shown in Table 1. It is very easy to compute since (a) it is small; (b) it is separable, thus two 1D weighted sums can be performed; and (c) it uses small weights and a power of 2 divisor. Moreover, it can be computed using addition only since it

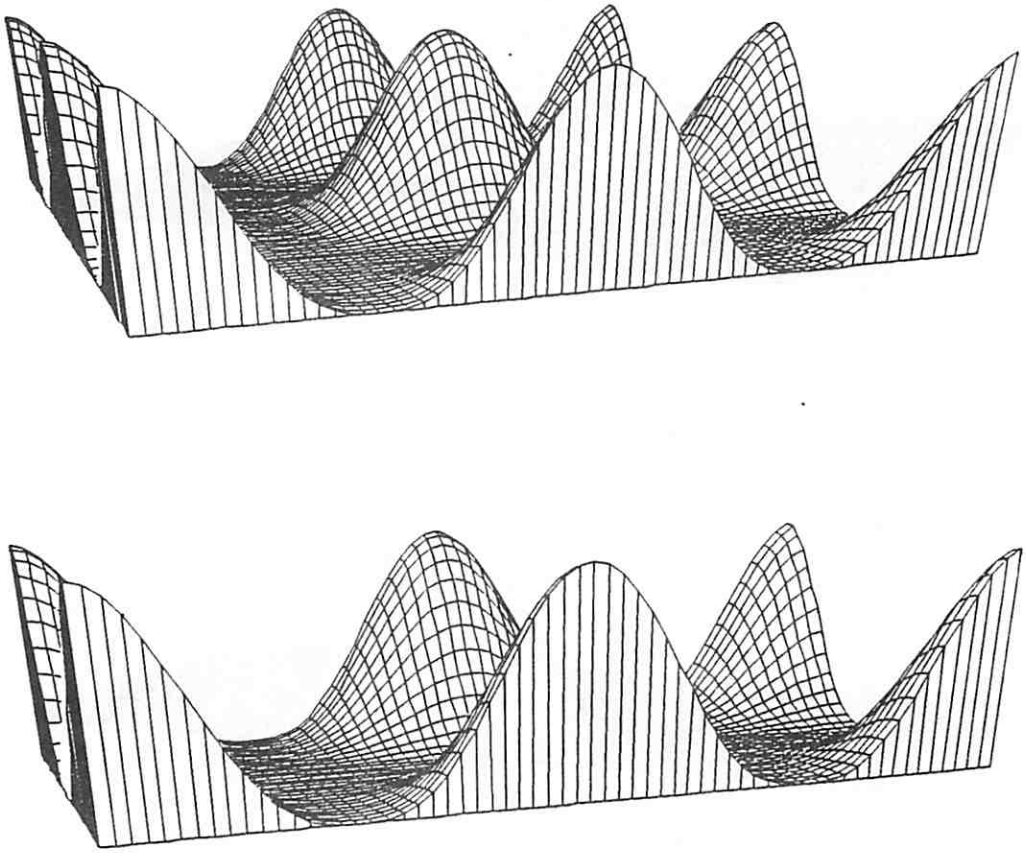


Figure 11: Transfer function of $S_1[G_3]$ spread low-pass filter

$S_1[G_3]$ is the one level spread of G_3 and is equal to $\frac{1}{8}[1\,0\,3\,0\,3\,0\,1]*\frac{1}{8}[1\,0\,3\,0\,3\,0\,1]^T$.

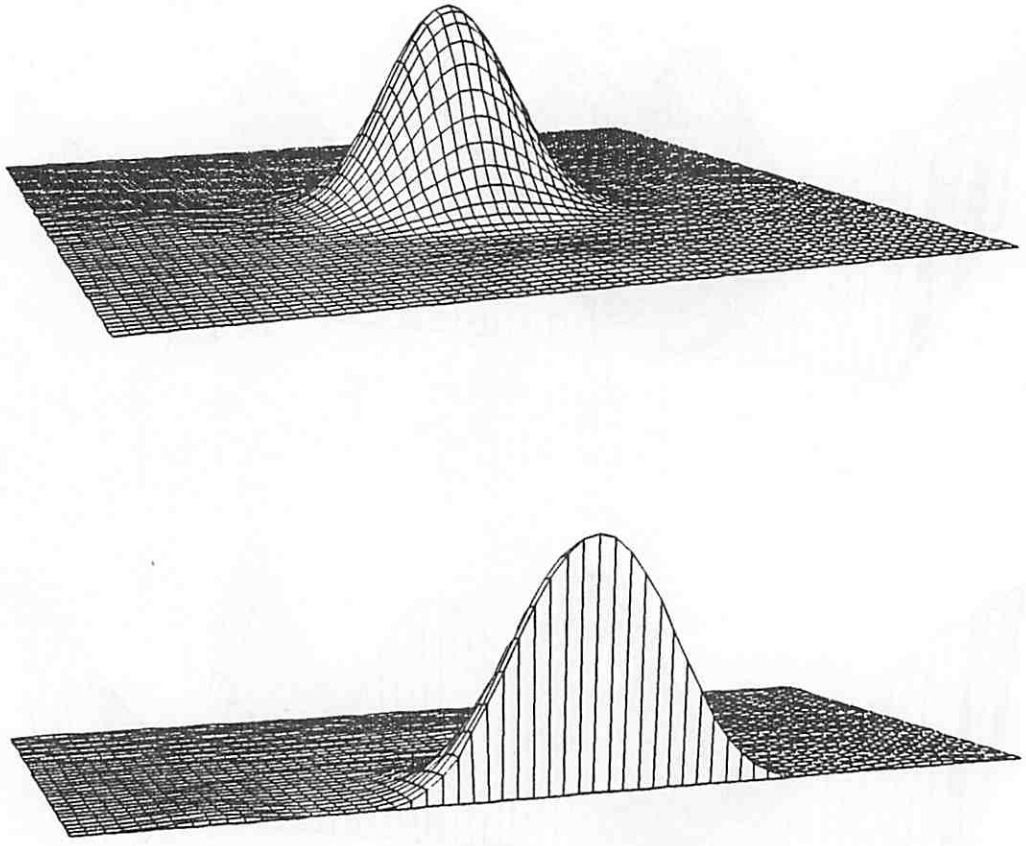


Figure 12: Transfer function of $S_1[G_3] * G_3$ net low-pass filter

This transfer function is equal to the (pointwise) multiplication of the transfer functions of G_3 and $S_1[G_3]$ (Figures 8 and 11 respectively).

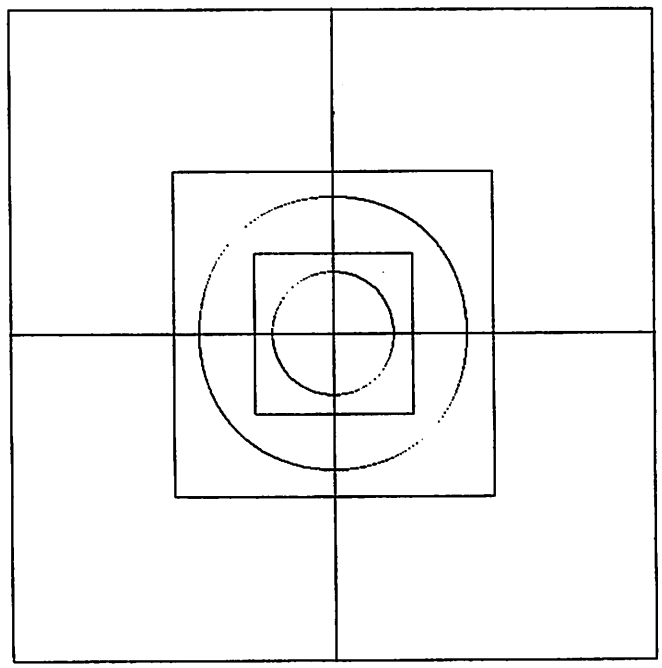


Figure 13: Cutoff boundaries of G_3 and $S_1[G_3] * G_3$

The cutoff boundary of G_3 is just within the boundary of the baseband B_2. This is the Nyquist boundary for 1-of-2 subsampling. The cutoff boundary of $S_1[G_3] * G_3$ is just within the boundary of the baseband B_4. This is the Nyquist boundary for 1-of-4 subsampling.

is equal to $g_1 * g_1 * g_1 * g_1^T * g_1^T * g_1^T$ where $g_1 = \frac{1}{2}[1\ 1]$. Finally in the last section we saw that using G_3 to build image pyramids is equivalent to using net convolution mask which are approximately Gaussian. The result of using the 4×4 operator $G_3 = g_3 * g_3^T$ to build a low-pass pyramid appears in Figure 14.

4.7 Band-Pass Pyramids

In Chapter IV we will see that correlation matching is better performed when low spatial frequencies (relative to the grid size) have been filtered out. Such high-pass filtering performed at each level of a low-pass pyramid effectively produces a band-passed image at each level.

High-pass filters can be readily constructed from given low-pass filters. Let φ_L be some low-pass filter and define the new filter $\varphi_H \triangleq Id - \varphi_L$ where Id is the identity filter, that is $Id(f) = f$. Id is a linear shift invariant filter with convolution mask (impulse response) $\delta(i,j) \triangleq [1]$ and transfer function $\mathcal{F}(\delta) \equiv 1$. The transfer function of φ_H is $\mathcal{F}(\varphi_H) = 1 - \mathcal{F}(\varphi_L)$. Since φ_L is a low-pass function, it has gain near 1 at low frequencies and attenuation near 0 at high frequencies. Thus, φ_H will have attenuation near 0 at low frequencies and gain near 1 at high frequencies making it a high-pass filter. One example is

$$\delta - G_2 \triangleq [1] - \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

We define **discrete Laplacians** as finite difference approximations to the Laplacian operator $\nabla^2 \triangleq \partial^2/\partial x^2 + \partial^2/\partial y^2$. Two good choices for discrete Laplacians are:

$$Lap_1 \triangleq \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \quad \text{and} \quad Lap_2 \triangleq \frac{1}{h^2}\begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & -3 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

where h is the grid spacing. Note that Lap_2 is equal to $\delta - G_2$ up to a scale factor. Lap_1 can similarly be represented as δ minus a local averaging (low-pass) filter.

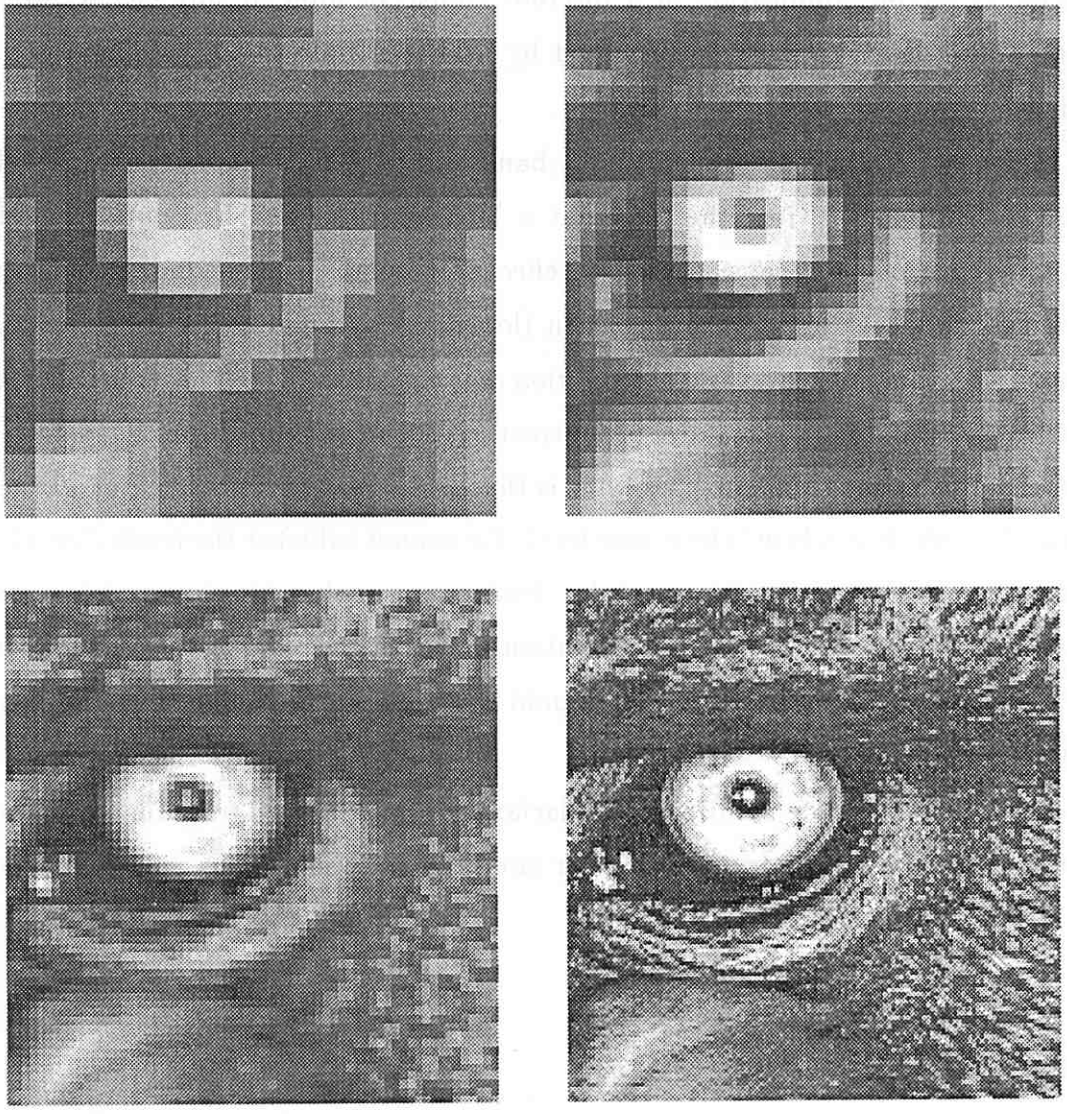


Figure 14: Low-pass pyramid

Levels 4 through 7 of the low-pass pyramid obtained from the mandrill eye image by applying the 4×4 reduction operator $\frac{1}{8}[1\ 3\ 3\ 1] * \frac{1}{8}[1\ 3\ 3\ 1]^T$.

Since our low-pass pyramid has been generated by filters that are approximately Gaussian in shape, application of a discrete Laplacian filter at any coarse level is comparable to $\nabla^2 G$ filtering proposed by [Marr & Hildreth 80] for band-pass filtering images prior to edge extraction.

A second method for generating a band-pass is Burt's Laplacian pyramid [Burt 82a, Burt 82b]. Here the fact that a $\nabla^2 G$ operator can be approximated by a difference of Gaussians is used to effectively compute band-pass filters by differencing adjacent levels of a Gaussian (low-pass) pyramid. The difference is taken between the finer level and a projection of the coarser level. This technique is equivalent to that mentioned above of subtracting a low-pass filter from the identity operator . In this case the low-pass filter is that which was performed prior to the subsampling which produced the coarse level. We cannot subtract the levels directly but must project down the coarser data. Projection involves the interpolation of data to fill in values on the higher resolution fine grid given values on the coarse grid. Bilinear projection in the even pyramid is performed using the masks shown in Figure 4b.

Figure 15 shows the band-pass (Laplacian) pyramid derived from the low-pass (Gaussian) pyramid in Figure 14. Bilinear interpolation was used to project coarse data down.

5 Summary

In this chapter we have presented the hierarchical multiresolution processing architecture upon which the motion detection algorithms of Chapters IV, V, and I are based. The *processing cone* was presented as a general architecture for multiresolution image processing. All of the algorithms presented in later chapters will be seen to fit readily into this architecture. Moreover, all experiments have been performed in a processing cone simulator. *Image pyramids* were presented as general multiresolution image representations. The hierarchical motion detection algorithms in Chapters IV and V operate on these pyramids.

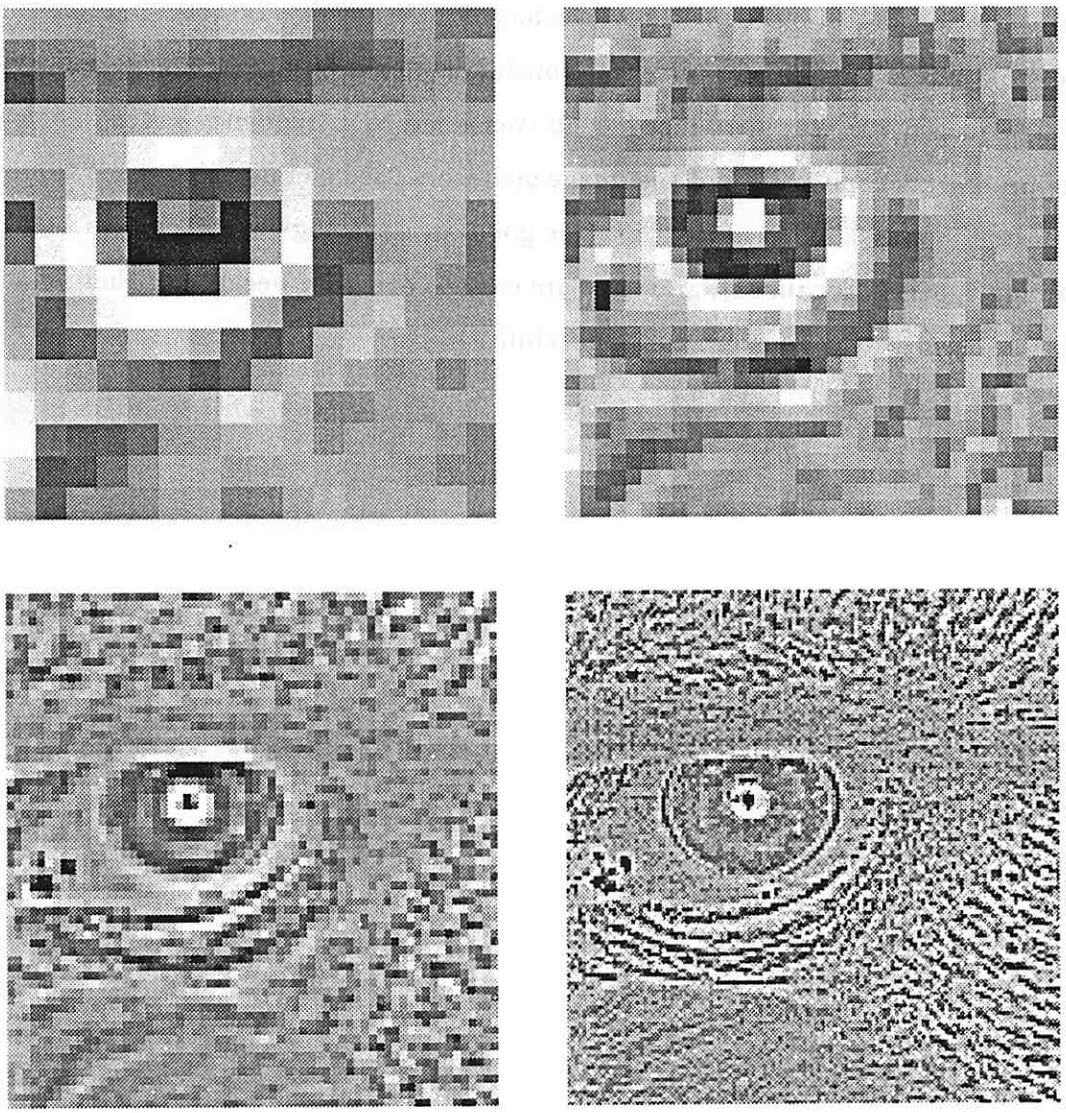


Figure 15: Band-pass pyramid

Levels 4 through 7 of the band-pass pyramid obtained from the low-pass pyramid in Figure 14. Coarse data is projected down using bilinear interpolation filter shown in Figure 4b.

Low-pass and band-pass pyramids provide the image data used by the hierarchical motion detection algorithms in Chapters IV and V. In this chapter, the process of building image pyramids was considered and a family of "discrete Gaussian" low-pass filters was introduced. This was aided by a frequency space analysis of the processing cone grids and the image operators used to build image pyramids. The discrete Gaussians presented, provide good anti-aliasing characteristics (while remaining spatially localized), and they are easy to compute because of their small weights, power of two divisors, and separability.

CHAPTER III

Motion Detection

1 Introduction

The determination of the change in position of objects from one image to another is an important step in many image processing tasks. These include two-dimensional problems such as image registration and three-dimensional ones such as stereo vision and motion analysis. In either case *image motion detection* — the detection of the disparity between two (or more) images — contributes to the solution of the problem. In Chapter I, we defined *image motion* as the motion of components of an image sequence. Two types of motion were defined: image flow and image disparity. They will be further described in Section 2.1.

In this chapter, we review the various techniques used for low-level motion analysis. The two major methods of feature matching and gradient analysis are emphasized in preparation for the hierarchical extensions presented in later chapters. In Section 3, correlation matching, a specific type of feature matching, is reviewed. In Section 4, first order gradient-based methods for the computation of flow are reviewed.

2 Overview of Motion Analysis Techniques

In this section we discuss the major factors that distinguish between the various types of image sequence analysis techniques.

2.1 Disparity and Flow

Motion is represented and measured in two ways: (1) displacement — net change in position; or (2) velocity — instantaneous rate of change in position. With regard to image motion we will refer respectively to *disparity* and *flow*.

Disparity is a spatial displacement vector that points from a feature in one image to its corresponding location in another image. It is measured in units of spatial length in the image coordinate system. **Flow** is a spatial vector that points in the direction of instantaneous motion of a given feature in one "frame" of a dynamic image sequence. It is measured in units of velocity in the image coordinate system, i.e. length/time. Flow is the limiting case of disparity divided by intra-frame time as we let the time difference between images go to zero. Flow vectors can only be defined when the dynamic image sequence is continuous, since they represent an instantaneous rate of change. Though all dynamic image sequences we deal with are discretely sequenced, we can consider them as sampled from a continuous sequence, and, if the intra-frame times are small (relative to the maximum velocities), we can speak of (and approximately compute) flow.

Depending on the task at hand, one or the other of disparity and flow will be the desired representation. For image registration and stereo problems, disparity is used. For motion analysis both disparity and (optic) flow are used.

Optic flow

In a dynamic imaging situation, the motion of objects, viewers, or light sources induce the corresponding motions of their projections on the imaging plane. **Optic flow** is the apparent motion of image features in the imaging plane and can be represented as a vector field in that plane. This field specifies the instantaneous velocity of the corresponding image component at points in the image plane. It has also been defined as the projection of the velocities of the environmental objects being imaged [Lawton 84]. We do not use this definition since the optic flow in a dynamic image can be defined independent of any three-dimensional imaging

situation that produced it.

2.2 Motion Extraction Methods

We divide the various motion detection techniques into the three broad categories of feature matching, gradient-based methods, and global change detection. We concentrate on the first two methods because they provide local detection of image motion at early stages in the visual process.

Feature matching involves (1) the "preprocessing" of images to extract meaningful features, such as edges, corners, blobs, etc., followed by (2) the determination of the correspondence between the features in the images being matched [Prager 79, Barnard & Thompson 80, Lawton 84]. *Correlation matching* is a special case of feature matching in which step 1 is reduced to simply selecting local neighborhoods from the images, and step 2 is performed by a pixel-by-pixel comparison of neighborhoods. This method is described more fully in Section 3.

Gradient-based techniques extract motion information from image sequences using the structural information contained in spatial and temporal derivatives of the image "function". They can be thought of (or derived) in two ways. First, we can consider the dynamic image sequence as a function of not just two spatial coordinates but of three coordinates, the standard two spatial coordinates and the third being time (or more generally, sequence number). Motion in the two spatial coordinates then appears as "structure" in the three-dimensional space. It can then be detected by (a) looking for "edges" in the three-dimensional space [Glazer 81, Haralick & Lee 83] or (b) computing gradients of the image in the three-dimensional space [Fennema & Thompson 79, Marr & Ullman 79, Horn & Schunck 81]. The second way gradient-based techniques are formulated is as a variational problem in which the disparity field is solved for by minimizing some measure of dissimilarity [Nagel 83a]. This is equivalent to the first way in that the variational problem can be reduced to a corresponding gradient-based formulation. We will use the former type of derivation. Gradient-based methods are more fully described in Section 4.

Global change detection methods look for large scale changes between images. One example of this is *differencing techniques* which involve the analysis of difference images formed by subtracting one frame from another. This analysis is similar to standard image segmentation techniques in that regions of change and non-change are found. The detected structure in this case is a sign of both image content and object motion. Such techniques can extract only very coarse and simple image motions. *Region matching* (between frames) is another global change detection method. Preprocessing by some type of region extraction (segmentation) algorithm is followed by the matching of corresponding regions between frames. This technique is also coarse and it is very dependent on the capabilities of the region extractor.

Gradient-based, feature matching, and region matching methods lie along a spectrum of possibilities defined by the amount of preprocessing of individual frames that takes place before inter-frame processing begins. Gradient-based techniques start with inter-frame (as well as intra-frame) processing. Feature matching techniques spend some time extracting interesting feature points from individual frames. Region matching requires extensive segmentation processing in the individual frames to extract regions that can be matchable reliably between frames. Our interests lie strongly with the former techniques since we are interested in motion detection algorithms which involve local processing that must occur in the early stages of visual processing.

2.3 Local Detection of Image Motion

Both correlation matching and gradient methods use the information in local neighborhoods to detect motion. The extent to which this is possible depends on the image structure in a given local neighborhood and the particular motion analysis algorithm. We will distinguish between three basic cases: (1) motion cannot determined; (2) only one component (of two) of the motion vector can be determined; and (3) the complete motion vector can be determined.

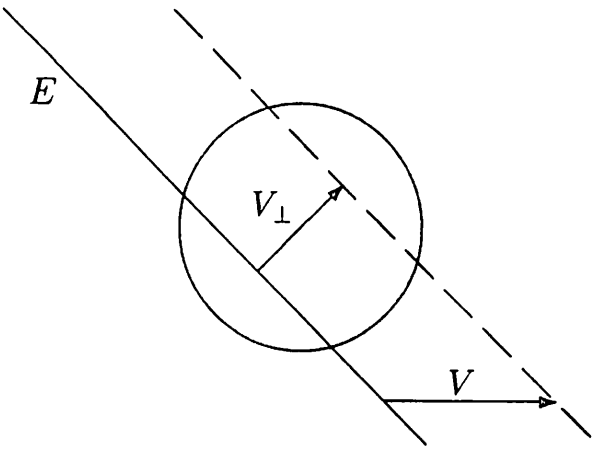


Figure 16: The local ambiguity of a moving edge

The edge E is shown before (solid) and after (dashed) motion. V is the disparity vector, the actual motion. In the local neighborhood shown by the circle, the detectable motion is $V_\perp$ — the component of V in the direction perpendicular to the edge.

When we try to determine the motion of a linear edge in a dynamic image sequence, in general, only one component of the motion (which is represented as a two-dimensional vector) can be detected. We call this the **local ambiguity** of the image motion of that feature. Figure 16 shows a basic example. The local ambiguity is due to the fact that the motion of a linear feature along the direction of no change (i.e. perpendicular to the gradient) is undetectable, while motion in a direction of change (parallel to the gradient) is detectable. If the motion is not in either of these two directions, then only that component of the motion perpendicular to the linear feature is detectable.

As we will in Section 4, first order gradient-based methods suffer from the same local ambiguity, for similar reasons. Namely, the (first order) gradient is essentially a *linear approximation* to the local (dynamic) image structure. This linear approximation only allows for the component of motion in the direction of the gradient to be directly computed

Methods of motion analysis which detect the motion of linear features in an

image or which are based on first order approximations must employ a second stage of processing to arrive at the true motion in the image given the detected "perpendicular" flow. These methods involve the combination of information over local neighborhoods under various constraints upon the computed motion vectors. Various two stage methods are reviewed in Section 4.3. In the first stage the component of optic flow perpendicular to an edge (or parallel to the image gradient) is measured locally. These vectors will be called **edge flow** because they represent the local motion of edges (and image isocontours). In the second stage the mutual constraints of neighboring edge flow are used to compute the optic flow.

2.4 Dense Flow Fields

Many algorithms have been proposed for the analysis of 3D motion based on 2D image motion. In general these algorithms are sensitive to errors in the measurement of the 2D image motion. Such errors can lead to bad 3D motion estimates. To avoid these errors we need either very accurate flow vectors or very many flow vectors with a low overall error. The first case is satisfied by feature matching techniques which use highly distinguished features such as oriented high contrast edges or gray level "corners". Since only a relatively small set of such points can be found in an image these algorithms produce a sparse set of flow vectors. The advantage to such techniques is that incorrect matches are much less likely and the accuracy of matches is higher.

The second case includes those techniques designed to compute a dense set of flow vectors. This includes both correlation matching and gradient-based algorithms. While the occurrence of bad matches is higher and the flow vectors may be less accurate, the large number of image motion estimates allows further motion analysis to extract good 3D motion estimates.

While both sparse and dense motion detection methods are useful in further motion analysis, the computation of dense motion is better suited to cellular arrays and the processing cone. This is because such processing is more local and uniform than that for sparse methods. Sparse methods put much effort into feature

detection, after which matching need only be done at a relatively small subset of points. There are two disadvantages here: (1) the matching process must operate over larger distances in the processing grid, and (2) many nodes in the grid will be idle. On the contrary, the dense methods we investigate operate over smaller distances in the grid and use all nodes for processing.

3 Correlation Matching

In correlation matching a *sample window* about a point in one image is compared to trial windows in the second image. The point in the second image whose trial window gives the optimal correlation value is chosen as the match point. This can be considered a special case of the more general method of feature matching. An example of alternative features are the edges that were used in Marr and Poggio's matching algorithm [Marr & Poggio 79]. With correlation matching the "feature" is the ordered set of pixel values of the sample window. There have been a number of studies of correlation techniques for matching images [Aggarwal *et al.* 81, Rosenfeld & Kak 82]. In this section we review the key issues.

Strictly speaking, with correlation we are matching not points in the image space but regions of the image. This is most apparent when matching two images which differ only by a translation, using large sample windows. We are, however, concerned with the more general case of a disparity field that varies across the image. We then wish to compute a disparity vector for each of a dense set of points in the image. For each such point a small (local) neighborhood must be chosen as sample window.

3.1 Matching Geometry

The matching geometry specifies which neighborhoods in the two images will be compared. This includes: (1) a set of points in the first frame for which matches must be found, (2) a neighborhood of points about each such point (the sample window), (3) a set of candidate match points in the second frame for each point

to be matched in frame one. The first set is chosen to be all points in the image because of our requirement of dense flow fields plus the fact that we assume a cellular array of processing nodes, one per pixel. The size of the neighborhood is considered in Section 3.3. The candidate matches are determined by the maximum possible displacements and, in the hierarchical method, by estimates of the actual displacement.

3.2 Correlation Measures

There are more than a few variations of the basic correlation measure, e.g., see [Burt *et al.* 82]. The *direct correlation* from which the family of measures derives its name is the sum of the pairwise product of corresponding pixels in two windows. Common variations include (1) mean normalized correlation, in which the mean of the values in each window is subtracted from each value; (2) variance normalized correlation, in which the correlation sum is divided by the variance of the two windows; (3) sum of squared differences, and (4) sum of the magnitude of differences.

For two images f_1 and f_2, the correlation between a neighborhood about the (two-dimensional) points $\mathbf{x}$ in f_1 and $\mathbf{x}+\delta$ in f_2 is given by the following definitions. The index set N includes the relative location of all points n defined to be in a local neighborhood.

Direct correlation:

$$C_D(\mathbf{x},\delta) = \sum_{n\in N} f_1(\mathbf{x}+n) f_2(\mathbf{x}+\delta+n)$$

Mean normalized correlation:

$$C_M(\mathbf{x},\delta) = \sum_{n\in N} |f_1(\mathbf{x}+n) - \hat{f}_1(\mathbf{x})||f_2(\mathbf{x}+\delta+n) - \hat{f}_2(\mathbf{x}+\delta)|$$

where

$$\hat{f}(\mathbf{x}) = \frac{1}{|N|}\sum_{n\in N} f(\mathbf{x}+n)$$

$|N|$ is the size of the neighborhood, i.e. the number of elements in N. With mean normalized correlation any offsets — constant values that have been added to all of the values in either neighborhood — are ignored.

Variance normalized correlation:

$$C_V(\mathbf{x},\delta) = \frac{\sum_{n\in N}[f_1(\mathbf{x}+n)-\hat{f}_1(\mathbf{x})][f_2(\mathbf{x}+\delta+n)-\hat{f}_2(\mathbf{x}+\delta)]}{\sqrt{\sum_{n\in N}[f_1(\mathbf{x}+n)-\hat{f}_1(\mathbf{x})]^2}\sqrt{\sum_{n\in N}[f_2(\mathbf{x}+\delta+n)-\hat{f}_2(\mathbf{x}+\delta)]^2}}$$

With variance normalized correlation any contrast changes — scale factors that all of the values in either neighborhood are multiplied by — are ignored.

If we think of the set $\{f(\mathbf{x}+n)\,|\,n\in N\}$ of values as a vector of length $|N|$, then these correlation measures have a geometric significance in the vector space of all such possible sample windows. *Direct correlation is an inner product since it is the sum of the pair-wise multiplication of the elements of the vectors.* With $\hat{f}(\mathbf{x})$ defined as above, we call $\{f(\mathbf{x}+n)-\hat{f}(\mathbf{x})\,|\,n\in N\}$ the *mean-normalized vector.* The sum of the components of the mean-normalized vector is zero, so all such vectors lie on the hyperplane $\{\mathbf{x}\,|\,\sum_{i=1}^{|N|}x_i = 0\}$ in $\Re^{|N|}$. Mean-normalizing is equivalent to projection onto this hyperplane. *Mean normalized correlation is the inner product of the mean-normalized vectors.* Now consider the definition of variance normalized correlation. The two terms in the denominator are the lengths of the mean-normalized vectors. Thus, the *variance normalized correlation is the cosine of the angle between the mean-normalized vectors.*

Two other common "correlation" measures are distance measures in the $|N|$-dimensional vector space. The sum of squared differences is the (square of) Euclidean distance between the windows. The sum of the magnitude of differences is the "city block" distance between the windows.

Sum of squared differences:

$$C_SD(\mathbf{x},\delta) = \sum_{n\in N}[f_1(\mathbf{x}+n)-f_2(\mathbf{x}+\delta+n)]^2$$

Sum of the magnitude of differences:

$$C_MD(\mathbf{x},\delta) = \sum_{n\in N}|f_1(\mathbf{x}+n)-f_2(\mathbf{x}+\delta+n)|$$

3.3 Sample Window Size

The size of the sample window is constrained by two basic considerations. First, the sample window must be large enough for the pattern of pixel values in the window (the image "structure") to be distinctive. Second, the sample window must be small enough to resolve variations in the disparity field. Two further constraints should be considered. If there is significant noise in the images, then the sample window must be large enough to minimize the effect of bad pixel values. The computational cost of correlation matching is proportional to the size (area) of the sample window. For this reason it should be kept as small as possible.

The use of correlation measures is predicated on the assumption that the neighborhood of a point to be matched (as defined by the correlation window about it) maps via a translation to a congruent neighborhood about the match point in the other frame. For a given size correlation window, the degree to which the disparity mapping is translational is a major factor in the efficacy of the matching process. Conversely, for a given mapping, the size of the correlation window is a major factor. It must be chosen small enough so that the mapping is approximately translational within neighborhoods that are compared.

In general, we are concerned with disparity fields that vary across the image, while remaining only locally translational. Smaller correlation windows, within which the field is approximately a uniform translation, must be used and correlation matching must be performed relatively densely.

4 Gradient-Based Methods

In this section we cover the basic ideas behind (first order) gradient-based methods for computing image motion.

4.1 2D Motion from Edges in 3D XYT space

Discontinuities (edges) in a static image result from occluding boundaries, shadowing (occlusion of lighting), surface markings, and internal edges of objects. In a dynamic imaging situation we have an image $I(x, y, t)$ with objects that can move, spin, shrink, expand, and deform (translate, rotate, scale, etc.) relative to the camera. Discontinuities in $I(x, y, t)$ can be viewed as two-dimensional surfaces swept out by the edges in successive time-slices of $I(x, y, t)$. We refer to these two-dimensional discontinuities in a three-dimensional space as **2D-edges**, while we call edges in a two-dimensional image **1D-edges** when the distinction must be made. 2D-edges contain information about both the location of 1D-edges in a single frame and the velocity of these edges in the xy image plane.

If V is the velocity of a 1D-edge, $U_\perp$ a unit vector perpendicular to that edge, and $P = (P^x, P^y, P^t)$ a vector perpendicular to the 2D-edge P swept out by the moving 1D-edge, then we get (see Appendix B and [Glazer 81])

$$V \cdot U_\perp = \frac{-P^t}{\sqrt{(P^x)^2 + (P^y)^2}} \tag{3}$$

This is the magnitude of the component of the velocity of the 1D-edge perpendicular to that edge. The component of velocity parallel to the 2D edge cannot be computed from the 3D edge. This is due to the local ambiguity in the detection of optic flow. Thus, Equation 3 only constrains the velocity V to lie on a line in 2D velocity space. This line is called the velocity constraint line and Equation 3 is called the velocity constraint equation.

4.2 First Order Image Gradient Analysis: Flow

In the previous section, the velocity constraint equation was derived for the case of discontinuities in 3D dynamic images. A similar equation has been derived for the case of continuous 3D dynamic images. This equation relates gradients in a dynamic image to image motion. In this section we present such a derivation. The equation has been derived before in various ways, notably [Horn & Schunck 81]

and [Fennema & Thompson 79]. Our derivation in terms of Taylor series will be generalized to image disparities (with coarse estimates) in Chapter V.

Consider the path of a moving point in the dynamic image sequence: $(x(t), y(t), t)$. The velocity or flow of this point is given by $(u, v, 1) \triangleq (\dot{x}, \dot{y}, \dot{t})$, where the dot notation $(\cdot)$ refers to differentiation with respect to time (d/dt). Let $F(x, y, t)$ be the dynamic image. Then the change in image value along the path for a given time difference $\triangle t$ is given by:

$$\triangle F = F(x(t + \triangle t), y(t + \triangle t), t + \triangle t) - F(x(t), y(t), t) \tag{4}$$

A Taylor series expansion gives us:

$$\begin{aligned} F(x + \triangle x, y + \triangle y, t + \triangle t) &= F(x, y, t) + (\triangle x, \triangle y, \triangle t) \cdot \nabla F(x, y, t) \\ &\quad + \text{higher order terms} \end{aligned} \tag{5}$$

Noting that $x + \triangle x = x(t + \triangle t)$ and $y + \triangle y = y(t + \triangle t)$, we can combine Equations 4 and 5 to get:

$$\triangle F = (\triangle x, \triangle y, \triangle t) \cdot \nabla F(x(t), y(t), t) + \text{higher order terms} \tag{6}$$

Now divide Equation 6 by $\triangle t$ and let $\triangle t \to 0$. The higher order terms vanish since they contain powers of $\triangle t$. Also $\frac{\triangle F}{\triangle t} \to \frac{dF}{dt}$, $\frac{\triangle x}{\triangle t} \to u \triangleq \frac{dx}{dt}$ and $\frac{\triangle y}{\triangle t} \to v \triangleq \frac{dy}{dt}$, leaving us with

$$\frac{dF}{dt} = (u(t), v(t), 1) \cdot \nabla F(x(t), y(t), t) = F_x u + F_y v + F_t \tag{7}$$

Note that the assumption that x and y depend on t allows the above equation for the total derivative to be written directly. This derivation is intended to stress the fact that the relationship between dF/dt, (u, v), and ∇F depends on a particular path $(x(t), y(t), t)$.

Equation 7 relates the total derivative of the dynamic image dF/dt (relative to the path $(x(t), y(t), t)$) to the gradient of the dynamic image ∇F and the optic flow field (u, v). It tells us the rate at which F 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

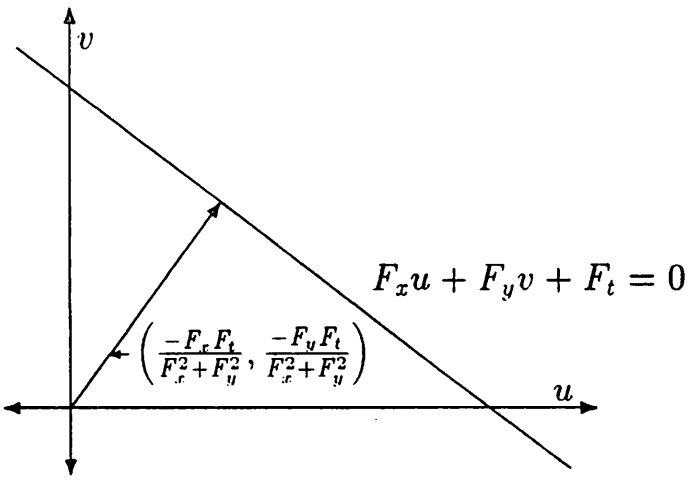


Figure 17: The velocity constraint line

The velocity constraint line is shown in uv "velocity space". It is the locus of velocities (u, v) satisfying the equation $F_x u + F_y v + F_t = 0$. The corresponding edge flow vector is also shown along with the equations for its components.

environmental point to remain at a constant intensity, i.e., $dF/dt = 0$ and so

$$F_x u + F_y v + F_t = 0 \tag{8}$$

Equation 8 specifies a line in u, v velocity space called the **velocity constraint line** (see Figure 17). This line is perpendicular to the spatial gradient (F_x, F_y). This line can be represented by the point on it which lies closest to the origin (in velocity space), that point is

$$\left(\frac{-F_x F_t}{F_x^2 + F_y^2}, \frac{-F_y F_t}{F_x^2 + F_y^2}\right) \tag{9}$$

This point defines the **edge flow vector**. It is perpendicular to the constraint line and has a length equal to the distance between the line and the origin. Figure 17 shows a velocity constraint line and the corresponding edge flow vector. For any (u, v) on the velocity constraint line, this edge flow is its component parallel to the spatial gradient (perpendicular to an edge). The magnitude of this edge flow is

$$\frac{F_t}{\sqrt{F_x^2 + F_y^2}} \tag{10}$$

essentially the same result as given in equation 3. In that equation edge flow is expressed in terms of the components of a vector normal to a 3D edge, i.e. parallel to the direction of maximum change, the gradient.

Velocity constraint lines — or the equivalent edge flow vectors — can be computed at all points in the image that have non-zero gradient. However, due to the local ambiguity of image motion, they do not determine an optic flow field. Other information must be brought to bear to constrain further our choice of a flow field.

The assumption that the image brightness of a particular point does not change as it moves ($\triangle F = 0$ or $dF/dt = 0$) depends on the imaging parameters and may or may not be satisfied. In general for it to be satisfied in a dynamic image sequence of a three-dimensional scene we would need (1) diffuse and/or point-at-infinity lighting, (2) orthographic projection, (3) no mutual illumination of neighboring surfaces, and (4) opaque objects of constant reflectance. While we cannot expect these conditions to all hold, the assumption of constant brightness is still viable since violations will most often be minor. Moreover, most methods we will use will require not that object brightness remains constant, but only that local variations are small relative to overall image brightness changes.

4.3 Optic Flow from Edge Flow

4.3.1 Velocity space and the constraint line

As we saw in the previous section, first order gradient-based methods can compute edge flow — the component of optic flow in the direction of the gradient. The optic flow vector can lie anywhere along the velocity constraint line determined by the edge flow. If (e, f) is the edge flow vector, then the velocity constraint line is the set of all points (u, v) in velocity space lying on the line $(u, v) \cdot (e, f) = (e, f) \cdot (e, f)$ or $eu + fv - e^2 - f^2 = 0$. The edge flow vector runs from the origin (in velocity space) to the velocity constraint line and is perpendicular to it. First order gradient-based methods can compute edge flow vectors at all points having a non-zero gradient.

If the gradient is zero at a point, then no information about the optic flow can be extracted. The optic flow will be unconstrained at that point, i.e., it can lie anywhere in velocity space. We now review various methods for computing optic flow from edge flow.

All of the methods we will discuss depend on there being sufficient variation in the gradient directions at different points in the image. This requirement may be necessary globally or locally depending on the scope of the particular algorithm. In the worst case where the gradients (at all points) share a common direction, no constraints would be placed on the optic flow perpendicular to that direction. Of course such an image is essentially one-dimensional and it is pointless to speak of motion perpendicular to the direction of variation.

4.3.2 The global histogram method

Fennema and Thompson [Fennema & Thompson 79] use a modified Hough transform clustering approach. They assume that there are only a few values for optic flow in the image. For each optic flow vector the related flow vectors lie on a cosine curve in the (ρ, θ) Hough transform space corresponding to u, v velocity space. Thus, for each such cosine curve "cluster" that can be found in the Hough transform space, there corresponds a single optic flow which is the optic flow value for all points that contributed to that cluster. The major disadvantage of this technique is the reliance on there being relatively few values of optic flow throughout the image, a situation that only holds for scenes with objects with translational motion parallel to the image plane. Note that this is essentially a 2D motion problem. If the originating scene is actually 3D, depth cannot be extracted from the given motion since no motion in depth information would be available.

4.3.3 The pseudo-intersection method

If we assume that the optic flow vector is locally constant (slowly varying), then we can combine the constraints given by the velocity constraint lines computed for each of the points in a small local neighborhood of the image [Glazer 81]. Ideally the

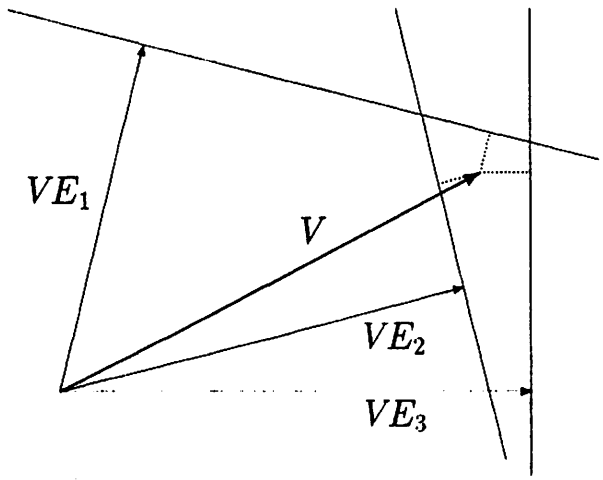


Figure 18: The pseudo-intersection of velocity constraint lines

The pseudo-intersection of velocity constraint lines provides a best fit (in the mean-square sense) flow velocity V for a set of non-parallel edge flow vectors VE_i.

velocity constraint lines intersect at the point in velocity space which is the optic flow. Noisy image data and discretization error cause deviations in the locations of these lines preventing a single intersection. The pseudo-intersection method finds that point which best "fits" the lines in the least-squares sense. It computes the point which minimizes the total of the squared distances between the point and each of the lines. The example in Figure 18 shows the pseudo-intersection of three velocity constraint lines. The computation of the pseudo-intersection is described in detail in Appendix C.

Successful computation of optic flow as the pseudo-intersection of velocity lines depends on those lines being derived from a single velocity. If optic flow cues from other velocities are introduced to the computation, the pseudo-intersection is moved towards those velocities and the mean square error increases. This condition may be detected by its high error but the offending cues cannot be isolated by the methods discussed so far.

4.3.4 Optimization methods

Horn and Schunck built a variational principle to accomplish the computation of optic flow from edge flow using a first order smoothness constraint on U and V requiring that the total (integrated) magnitude of the gradient of U and V be kept small. If in addition to this, it is required that equation 8 be satisfied ($F_x u + F_y v + F_t = 0$), 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 and Schunck to include the constraint of small dF/dt in their variational principle for optic flow. Thus, small values are encouraged but a value of 0 is not forced. The resulting variational problem is to find the optic flow field minimizing the following functional

$$\iint (F_x u + F_y v + F_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2)\, dxdy \tag{11}$$

The first term in the functional is the square of the rate of change of image brightness (measured in 3D space-time of the dynamic image space) as expressed in Equation 7. When $F_x u + F_y v + F_t = 0$, 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 α a relative weighting factor between the two constraints. The equivalent PDE system (Euler's equations, [Courant & Hilbert 53, Section IV.3.4]) is

$$\alpha^2 \Delta u - F_x^2 u - F_x F_y v = F_x F_t \tag{12a}$$

$$\alpha^2 \Delta v - F_x F_y u - F_y^2 v = F_y F_t \tag{12b}$$

The optic flow equations for the continuous case involve partial derivatives of the dynamic image which can only be approximately computed in the discrete case by finite differences. This ultimately limits the range of applicability of such methods. This constraint comes into play first along the time axis since the rate of temporal sampling is limited by sensor characteristics and by processing rates. If the frames are too far apart in time, the assumptions inherent in this method break down. In Chapter V, a first order gradient-based analysis is performed for the case

of disparity computation and its limitations are discussed. A hierarchical method will be presented which overcomes these limitations.

5 Summary

In this chapter an overview of motion computation methods was presented to provide a background for the development of hierarchical algorithms in Chapters IV, V, and I. After a general discussion of motion analysis techniques, we focused on two classes which best satisfy our goal of obtaining local, uniform, parallel processing: correlation matching and gradient-based algorithms. In both cases, the general concepts behind these methods were presented including the choices to be made between variations in the algorithms and the setting of parameters.

In Chapter IV, two disadvantages of correlation matching are considered: high compute costs and incorrect matches due to repeating features. A hierarchical correlation algorithm will be developed that reduces the costs while improving the accuracy. In Chapter V, the inherent limitation of gradient-based algorithms to the computation of small disparities is considered. A hierarchical algorithm will be developed that extends gradient-based techniques to the case of extended disparities. The hierarchical algorithm of Chapter V and the single-level algorithm which it generalizes both use iterative relaxation to compute the disparity vectors. In Chapter I, we consider the high cost involved when many iterations must be applied. A hierarchical algorithm will be developed that significantly reduces this cost.

CHAPTER IV

Hierarchical Correlation Matching

1 Introduction

This chapter describes a hierarchical approach to correlation matching. The technique described here eases the computational burden while overcoming the problem of false matches. Basically the technique consists of matching multilevel representations of the images, starting at the coarsest resolution and proceeding to the finest. Band-pass pyramids are used as the multilevel representation. They are built with filters which approximate convolution with $\nabla^2 G$ operators of different sizes. The size of the Gaussian increases as the resolution becomes coarser, in such a way as to limit the frequency content in the image to avoid aliasing due to the sampling rate at each level of resolution.

The coarse-to-fine matching strategy greatly reduces computational costs. It also eliminates false matches due to repetitive details when the search area is chosen correctly. The matching is done initially based on the larger structures in the images (since they become prominent at low frequencies), providing ball-park estimates for matching higher frequency information at levels below. The elimination of low-frequencies at each level of the pyramid by the band-pass filtering helps overcome any problems due to illumination and scaling differences.

Standard single-level correlation matching techniques were described in Chapter III. In the remainder of this section, the inadequacies of such methods are noted and the use of coarse-to-fine matching techniques are proposed. In Section 2 the

major design choices in building a hierarchical matching algorithm are listed and our particular approach is overviewed. In Section 3 the major details of our hierarchical correlation matching algorithms are developed. Computational costs are considered in Section 4. In Section 5 we describe the details of the algorithms. The use of interest operators and confidence measures to deal with bad matches is discussed in Section 7. Experiments are presented in Sections 6 and 8, including results on noisy real world images. The development here of hierarchical correlation algorithms follows that presented by Glazer, Reynolds, and Anandan [Glazer *et al.* 83]. (One significant difference is that in this presentation, all image pyramids were built using the methods of Chapter II.) Extensions to include detection of incorrect matches, specifically those due to motion boundaries, can be found in [Anandan 84, Anandan 87].

In general, we are concerned with disparity fields that vary across the image, while remaining only locally translational. Smaller correlation windows, within which the field is approximately a uniform translation, must be used and correlation matching must be performed relatively densely. The computation of a dense set of correlation matches using small correlation windows gives rise to two major problems for single-level correlation: (1) added computational cost, and (2) false matches (made with high confidence).

High computational costs are incurred in proportion to the density of matching, the size of the correlation windows, and the size of the search areas. We want a dense set of matches and the choice of size of the correlation windows will fall under other constraints. Thus, we look to reducing the search area. The **search area** is defined as containing those points in the second frame which will be tested for match with a given point in the first frame. It can be reduced if we know that the maximum disparity is small. However, this is too restrictive. A more general and useful constraint is given if we have a coarse estimate of disparity. The inaccuracy of the estimate then determines the size of the search area. As we show later, a coarse-to-fine sequence of matches significantly reduces computational costs.

Matching with small correlation windows is more susceptible to incorrect

matches for two reasons. High frequency texture in the image may provide enough repetitive pattern that a number of different matches may be considered equally valid by the correlation technique. On the other hand the lowest frequencies may be due to illumination differences and may bias the correlation measure away from the correct match. Different kinds of normalizations can help overcome this latter problem, but only at greater computational cost [Burt *et al.* 82]. The former problem can only be overcome if we have an estimated displacement which, while not accurate, allows us to search for a match in an area containing only one copy (hopefully, the right one) of the repeating feature. This also suggests coarse-to-fine strategies.

2 Key Issues

When designing a hierarchical matching algorithm, the following design choices must be made: (1) the multiresolution representation of the image data to be used; (2) the correlation or matching measure; (3) the size of correlation window; (4) the size of the search area; (5) the density of points to be matched; (6) handling of false matches. Our choices are overviewed in the next few paragraphs. Detailed descriptions of our choices occur in Section 3.

A coarse-to-fine matching algorithm begins by matching *coarse details* in an image. How do we define and choose coarse details ? We could use the larger objects in the image. A similar approach would be to choose large regions in some segmentation of the image. These choices of coarse detail require relatively sophisticated preprocessing to identify such objects or regions. This conflicts with our goal of performing motion computation at an early stage in the visual process. The other alternative is to blur the image. In this case the coarse details are not actually identified. Rather the fine details are eliminated, leaving only coarse slowly varying image structure. Blurring is best represented as low-pass filtering of the image. The degree to which fine detail is eliminated is determined by the size (and shape) of the low-pass filter or equivalently by its frequency response. A low-pass filtered

image can be represented by a smaller (coarser) sampling grid. Thus, we are led to (low-pass) image pyramids as a multiresolution (coarse-to-fine) representation of the image data.

The choice of a *matching measure* is almost independent of the other design factors we can consider. The one interaction lies in the fact that the benefits of mean-normalized correlation can be achieved by using direct correlation on high-pass filtered data. Recall (from Chapter III) that mean-normalized correlation should be chosen over direct correlation when the mean values of otherwise matching windows might be different. High-pass filtering attenuates the DC component of an image along with other low frequencies. In such a filtered image the mean values of neighborhoods as large as the sample windows we will consider are effectively set to zero or some relatively small value. Thus, the result of using mean-normalized correlation in a low-pass pyramid is comparable to using direct correlation in a band-pass pyramid. This idea is further described in Section 3.2.

In Chapter III, Section 3.3, we noted the factors involved in choosing a *sample window size*. Windows as small as 5×5 can be used when noise is minimal, but more generally 7×7 up to 10×10 should be used.

For any given point in the first image to be matched, the *search area* is that set of pixels in the second image which will be considered as match candidates. The size (area) of the search area is a major factor in the computational cost of the overall matching algorithm. One of the main benefits of coarse-to-fine search strategies is the reduction in size of search areas.

The *search area* we use is a 3×3 block of pixels at all levels of the processing cone. Given some upper limit on the possible displacements, matching can be started at a level of resolution corresponding to a displacement of less than 1 pixel at that level. This restricts the search area to 3×3. A match within this area provides an estimate of the displacement within $\pm 1/2$ pixel accuracy at this level. The projection of this estimate to the next finer level provides an estimated displacement of ± 1 pixel accuracy at that level. Matching can now again be done in a 3×3 area around the estimated displacement at the finer level. This process is repeated

down to the finest level to finally obtain an estimated displacement of $\pm 1/2$ pixel accuracy.

The motion fields we wish to detect, while constrained to be locally translational, can otherwise show significant variation across the image. We must compute matches at a dense set of points to capture them. On sequential computers, computational cost is proportional to the number of points to be matched. This is not the case in the processing cone where matching at one level of resolution can be performed at all points in parallel. We attempt to obtain a *maximum density* of matches. Matching is performed at every point in the pyramid.

The use of both interest operators and sharpness measures are considered as techniques of *handling false matches*. Interest operators are used before matching to select distinctive or "interesting" points for which matching is more easily performed. Sharpness measures are used after matching to compute the degree of confidence for given matching points.

3 Hierarchical Correlation

3.1 Image Pyramids

In our hierarchical algorithm images are represented at varying levels of spatial resolution. Coarse resolution images are obtained by low pass filtering and subsampling. The filtering is done by convolution smoothing with Gaussian-like kernels. This low-pass filtering allows us to sub-sample these images and store them in coarse grids.

In the next section, the benefits of using band pass pyramids with direct correlation are described. In Chapter II, two methods of computing band pass pyramids were described: (1) applying a high-pass filter at each level of a low-pass pyramid, and (2) differencing adjacent levels of a low-pass pyramid.

3.2 Correlation and Match Measures

In Chapter III the major similarity measures used in correlation matching were discussed. We have used both direct and variance normalized correlation with band pass pyramids. Direct correlation performed on the band pass filtered images in the band pass pyramids we use is comparable to mean normalized correlation performed on the corresponding low pass filtered images in low pass pyramids.

Recall that mean normalized correlation was defined in Chapter III as

$$C_M(\mathbf{x},\delta) = \sum_{n\in N} [f_1(\mathbf{x}+n) - \hat{f}_1(\mathbf{x})][f_2(\mathbf{x}+\delta+n) - \hat{f}_2(\mathbf{x}+\delta)]$$

where

$$\hat{f}(\mathbf{x}) = \frac{1}{|N|}\sum_{n\in N} f(\mathbf{x}+n)$$

$|N|$ is the size of the neighborhood, i.e. the number of elements in N. Mean normalized correlation can be approximately performed by direct correlation after a specific pre-filtering of the two images as follows:

$$C_{M2}(\mathbf{x},\delta) = \sum_{n\in N} [f_1(\mathbf{x}+n) - \hat{f}_1(\mathbf{x}+n)][f_2(\mathbf{x}+\delta+n) - \hat{f}_2(\mathbf{x}+\delta+n)]$$

The only difference here is that we have "mean-normalized" by subtracting the values of $\hat{f}_1$ and $\hat{f}_2$ not at the center of the window but at the individual points. If we define $g_1(\mathbf{x}) \triangleq f_1(\mathbf{x}) - \hat{f}_1(\mathbf{x})$ and similarly g_2 then by simple substitution

$$C_{M2}(\mathbf{x},\delta) = \sum_{n\in N} [g_1(\mathbf{x}+n)][g_2(\mathbf{x}+\delta+n)]$$

This is just direct correlation. The computation of g_1 and g_2 from f_1 and f_2 is a linear shift invariant operator. The filter (or convolution mask) used in this case is the difference of a "flat" averaging mask and the identity mask (a discrete impulse). As such it can be considered as a high-pass filter and is thus related to other high pass filters such as the discrete Laplacian. (In fact, conversely, approximate discrete Laplacians can be generated up to a scale factor by subtracting the identity mask from any small averaging mask.)

Replacing mean-normalized correlation by direct correlation on band pass filtered images has two advantages. The first is that computation is lessened by performing all of the "normalization" before the matching process. Thus, areas on the second image which are tested as match candidates for multiple points in the first image are only normalized once.

The second advantage is that other high pass and band pass filters with better filtering characteristics can be used for normalization. The frequency response of the subtract-local-mean filter is not very flat at high frequencies. For example, in Figure 19 the transfer function of a 7×7 subtract-local-mean filter is shown. Better high frequency characteristics can be obtained with high pass filters given by discrete Laplacians, high pass filters related to discrete Gaussians, or by band pass pyramid operators. Figure 20 shows the transfer function of a high pass filter given by subtraction of a discrete Gaussian (a *weighted* local mean). This filter provides a cutoff frequency comparable to that of the 7×7 subtract-local-mean filter with flat high frequency response.

Variance normalization is a scaling normalization. It is a non-linear operation and so cannot be performed by linear convolution filtering. Although it remains as an option to be used in our correlation measure, it may not be worth the extra computational burden. If the sample windows are much larger than the search area (e.g., 8×8 vs. 3×3 in our experiments), then the normalization values in the search area should be very similar. Thus, they would have little effect on the choice of match. We will mention though that Burt has shown how a certain class of normalization values can be computed efficiently in the processing cone [Burt 82b].

3.3 Sample Windows

In our experiments we have used 8×8 sample windows. This choice is intended to provide a tradeoff between small windows which are more immune to occlusion and distortion problems and large windows which capture a large amount of matchable structure.

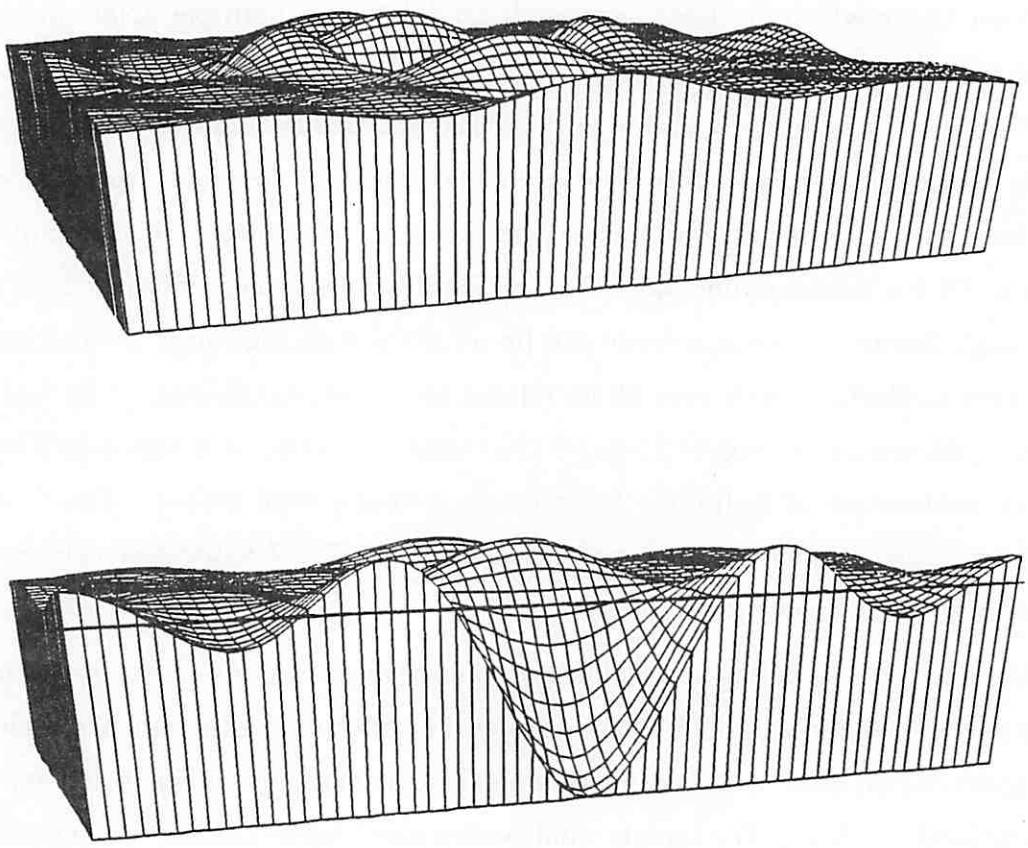


Figure 19: Transfer function of the 7×7 subtract-local-mean filter

The 7×7 subtract-local-mean filter is given by subtracting from each pixel the mean value of pixels in the surrounding 7×7 neighborhood. More formally it is given by $[1] - \frac{1}{7}[1\ 1\ 1\ 1\ 1\ 1\ 1] * \frac{1}{7}[1\ 1\ 1\ 1\ 1\ 1\ 1]^T$

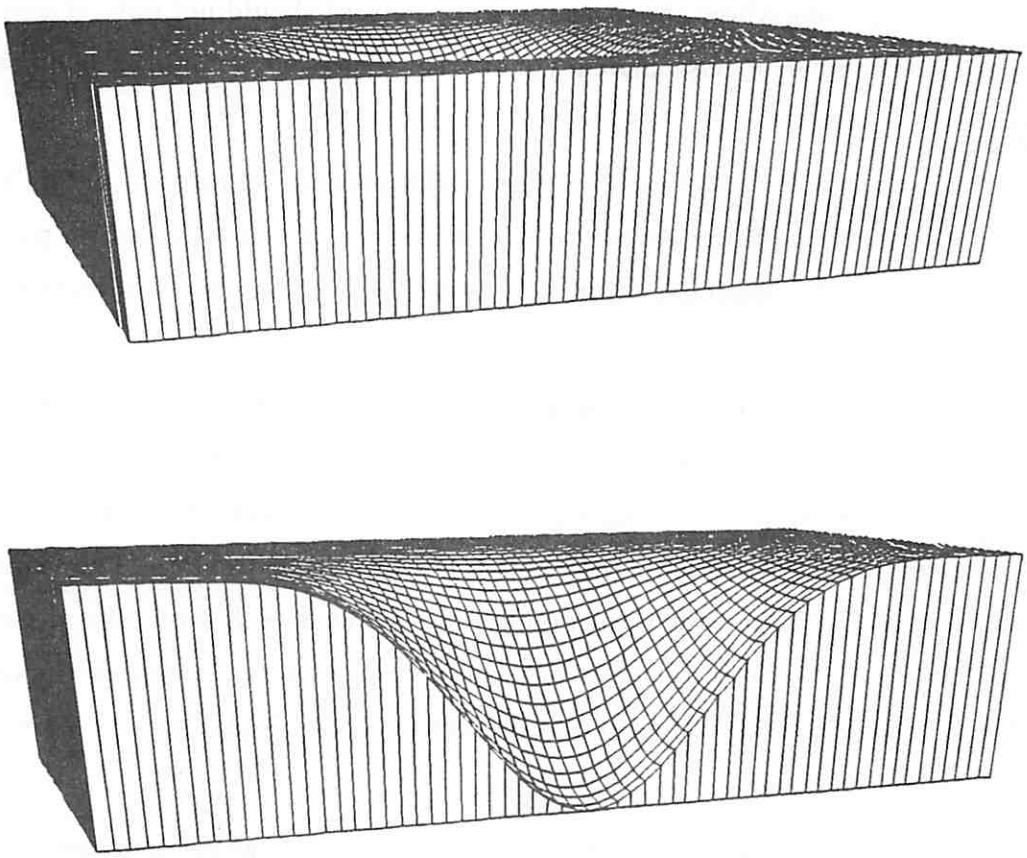


Figure 20: Transfer function of $Id - G_6$

$Id - G_6$ is a 7×7 filter which subtracts a *weighted* local mean from the central pixel value. The weighted local mean G_6 is a two-dimensional discrete Gaussian equal to $g_2 * g_2^T$ where $g_6 \stackrel{\Delta}{=} \frac{1}{64}[1\ 6\ 15\ 20\ 15\ 6\ 1]$.

3.4 Search Strategy

One way of looking at matching is as a process of searching for the point that optimizes some measure of similarity. In our case the measure is the local correlation measure between the two images. The strategy adopted for searching for the point of match (i.e., the point where the measure is maximized) should not only attempt to decrease the number of false matches, but also reduce the computational cost involved in searching.

We define the following terms. The **search area** is defined as containing those points in the second frame which will be tested for match with a given point in the first frame. The **search region** is the union of all sample windows in the second frame centered about points in the search area.

The search strategy adopted in our process is confined to a 3×3 set of potential matching pixels for a given pixel to be matched. This is true at all levels of the processing cone. It begins at a coarse level where the maximum displacement is within one pixel in both directions (see Figure 21). Let this be level L_d. The search is conducted in a 3×3 area around the point of interest at level L_d in the band-pass images at that level. The resulting displacements at this level (along each axis) are either -1, 0, or 1. The value obtained here is within $\frac{1}{2}$ pixel of the correct displacement at this level.

At the level below, L_{d+1}, the displacement values for a given point are projected down from its father pixel in level L_d. Since the unit of length we use is the interpixel spacing, and this decreases by a factor of $\frac{1}{2}$ when we go down a level, lengths must be multiplied by a conversion factor of two. Accuracy of $\pm\frac{1}{2}$ at level L_d is equivalent to that of ± 1 at level L_{d+1}. Searching in a 3×3 area at this level refines the displacement to within $\frac{1}{2}$ pixel accuracy at this level. The process is repeated down to and including the finest level of resolution of the image.

The coarse level d at which matching begins depends on the maximum row or column displacement D measured at the finest level L. It is chosen to be the finest level in which the maximum displacement corresponds to a distance less than

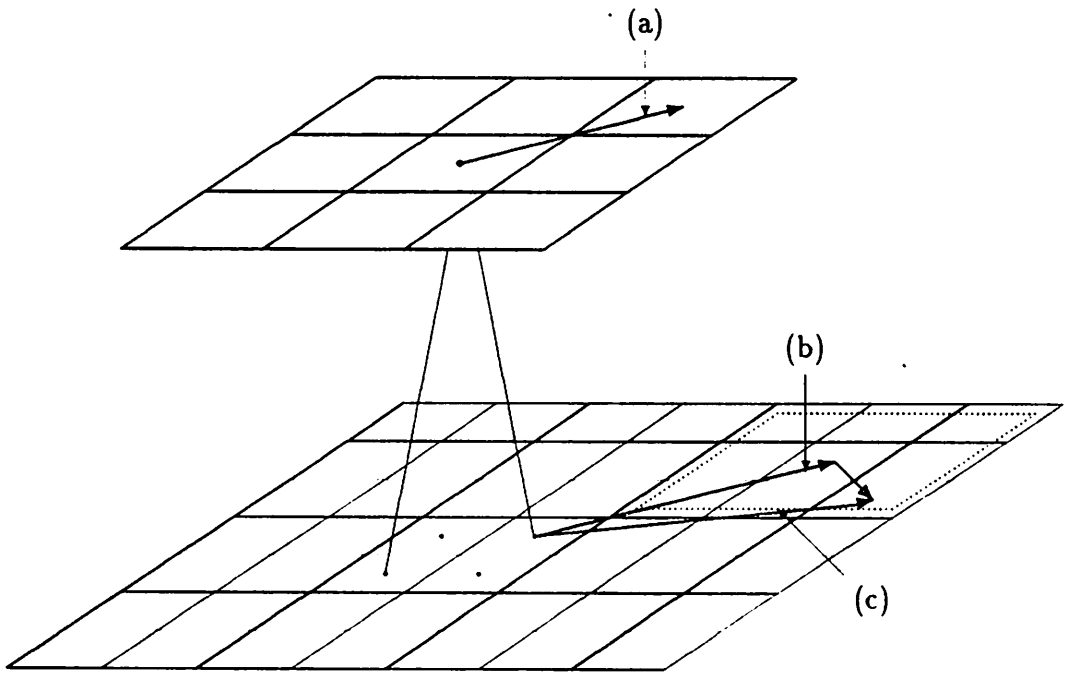


Figure 21: 3 by 3 search

Two levels of the processing cone are shown. At the coarse level, a 3×3 search area is shown with a computed displacement vector (a). At the finer level, the four sons of the central coarse pixel are marked. The projected displacement vector (b) is shown at one of the four sons. A 3×3 search area (dotted lines) about the projected displacement is used to find the updated displacement vector (c).

or equal to one pixel spacing. By convention the grid spacing at level L is $h_L = 1$ and hence at level d it is $h_d = 2^{L-d}$. The level d is chosen such that D is less than or equal to one pixel spacing at that level and larger than one pixel spacing at the next finer level $d+1$, that is $h_{d+1} < D \leq h_d$ or $2^{L-d-1} < D \leq 2^{L-d}$ or taking the base two logarithm $L - d - 1 < \log_2 D \leq L - d$. The last inequalities give us by definition $\lceil \log_2 D \rceil = L - d$, where $\lceil \cdot \rceil$ — the "ceiling" function — gives the smallest integer greater than or equal to $\log_2 D$. So we have: *If D is the maximum row or column displacement, then matching should begin at level $d = L - \lceil \log_2 D \rceil$.*

3.5 Existence and Uniqueness of Matches

The discussion in Section 3.4 shows how the technique of coarse-fine search strategy automatically ensures that the correct match must exist within the 3×3 local search window at each level. The filtering and subsampling processes ensure that the highest frequency at a particular level corresponds to a wavelength of two pixels at that level. Since the search is restricted to 3×3 windows at that level, we have some confidence that the match obtained within this window is unique. Also, the elimination of lower frequencies at that level help provide sufficient variation in the correlation measure within this window. Low frequencies in the image contribute corresponding low frequency content in the correlation function. This makes it more difficult to select the correct peak in the correlation function.

Strictly speaking, this argument is valid only for a one-dimensional version of this process. In the two-dimensional case there is no high frequency content along straight edges. This can lead to nearly constant values of the correlation measure along these edges, thus leading to false matches.

4 Computational Costs

The computational advantages of hierarchical versus single-level search strategies can be measured in two ways. We can consider how many points are searched in arriving at a final match at the finest level. Each ancestor of the final point

matched will contribute to this measure. On the other hand, we can measure the cost of obtaining matches at all points at the finest level. The former measure should be used when matching is confined to a relatively sparse set of interesting points, while the latter is used when matching is done almost everywhere. Since both of these cases are interesting, we will look at both.

First, let us consider the comparative cost of arriving at a single match at the finest level. Let D be the maximum displacement (measured at the finest level). Then the initial coarse search will be performed $\log_2 D$ levels above the finest level. The number of points searched is $9(\log_2 D + 1)$, because the search at each level is restricted to a 3×3 neighborhood and there are $\log_2 D + 1$ levels for search. On the other hand, a correlation process searching at one level through all points closer than the maximum displacement uses $(2D + 1)^2$ points for comparison.

Now consider the cost of computing matches at all points in the finest-level image. Let the finest-level image contain N^2 points and, as above, let D be the maximum displacement. Single-level correlation would then search $N^2(2D + 1)^2$ points. Hierarchical correlation would search $9(N^2 + N^2/4 + N^2/16 + \cdots)$ points, where the summation is over all levels at which matching takes place. However high those levels go, the sum is always less than that of the corresponding geometric sequence, viz., $4N^2/3$. Thus, the number of points searched hierarchically is at most $12N^2$. Table 2 shows a comparison of costs.

In the above comparisons we assumed that the same size sample window is always used (e.g. we use an 8×8 sample window size at all levels). At coarse levels these windows include a larger portion of the image space. In order to capture the same amount of information in the single-level search we would need to use sample windows of size $(8D)^2$.

Our hierarchical method uses lower frequency information at coarser levels and higher frequency information at finer levels, making use of all the useful information in the image. Since we use only the higher frequency information at the finer levels it is not necessary to use large sample windows.

D	S	H	S/H	S/12
1	9	9	1.0	—
2	25	18	1.4	2.1
4	81	27	3.0	6.75
8	289	36	8.0	24.1
16	1089	45	24.2	90.8
32	4225	54	78.2	352.

Table 2: Cost of single-level vs. hierarchical search

This table compares hierarchical and single-level search strategies. D is the maximum displacement, $S = (2D + 1)^2$ is the cost of single-level search, $H = 9(\log\ 2D+1)$ is the cost of *single match* hierarchical search, S/H is their relative cost factor, $S/12$ is the relative cost factor between single-level and hierarchical *full matching* (i.e., at all points).

5 Algorithms

Hierarchical correlation matching is a coarse-to-fine processing cone algorithm. The basic data flow is shown in Figure 22. At each level image data and coarse disparity estimates are used to compute updated disparity estimates. The matching algorithm is the same at all levels. It is a cellular parallel algorithm relative to the first image pyramid. That is, for each point in the first pyramid the computation of a match point is independent of that at other points. A simplified version of this processing is shown in Figure 23.

Near the borders of the images the sample windows will overflow. That is, they will require values at locations beyond the image border where none are defined. If we did not attempt matches in this case, we would severely reduce the area for which matches are computed. We wish to compute a match when there is sample window overflow. This cannot be done by keeping the full window size and filling in the missing pixel values, since there is no adequate way to do this and maintain accurate matches. Instead we choose to crop the window down to the largest size that will

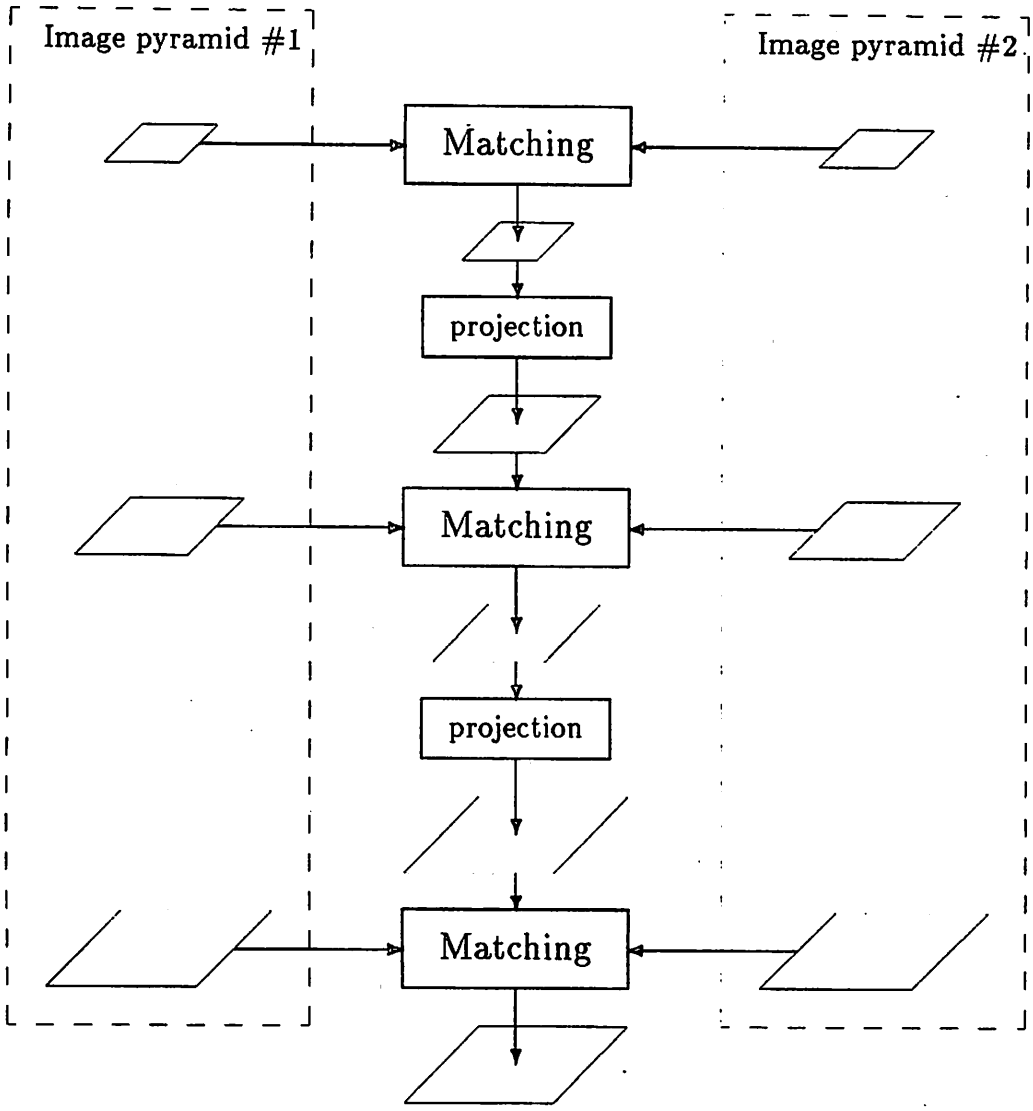


Figure 22: Basic hierarchical correlation: data flow

$(u_f, v_f) \leftarrow \text{Hicorr}\big(f_1, f_2, (u_c, v_c)\big)$

For all pixels (i, j) **Do** {
  $w_1 \leftarrow \text{nbrhd}(f_1, 0, 0)$,
  For $r_{rel} = -1$ to 1 **and** $c_{rel} = -1$ to 1 **Do** {
    $w_2 \leftarrow \text{nbrhd}(f_2, u_c + r_{rel}, v_c + c_{rel})$,
    $corr(r_{rel}, c_{rel}) \leftarrow \sum w_1(\cdot) \times w_2(\cdot)$ },
  $(r_{rel}, c_{rel}) \leftarrow$ position at which $corr(\cdot, \cdot)$ is maximum,
  $(u_f, v_f) \leftarrow (u_f, v_f) + (r_{rel}, c_{rel})$
  }

Figure 23: Basic hierarchical correlation: pixel processing

The basic hierarchical correlation operation takes two frames of image data (f_1 and f_2) and coarse disparity data (u_c, v_c) as input. It outputs an updated disparity field (u_f, v_f). The function $\text{nbrhd}(f, \Delta i, \Delta j)$ returns the sample window centered at the pixel $(i + \Delta i, j + \Delta j)$.

fit both (1) within the first frame when placed about the point being matched, and (2) within the second frame when placed about *each* of the test points in the 3×3 search area.

A further, stronger, overflow condition occurs when the 3×3 search area falls off the edge of the second image. We call this **search area overflow**. At the initial matching level, this occurs at each of the border pixels. At lower levels it can occur when coarse estimates point to border pixels in the second image. In either case, the correct match may be one of the unrepresented pixels which lies outside the border of the second image. This overflow condition is deemed to be more serious than mere sample window overflow. We choose to not compute a match when this situation arises.

Search area overflow is one of three conditions that prevent the computation of a match at a given pixel. The second is "un-matchability" as determined by an interest operator. The third is the lack of an adequate coarse estimate of disparity from the father pixel. This occurs when the father pixel cannot compute a match

for one of these same three reasons.

In any case, when a match cannot be computed, the displacement estimate cannot be refined. This coarse displacement must be accepted as the best that can be done and a flag must be introduced to note the relatively lower accuracy at that point. We thus define the **displacement pyramid** as having three components (at each level of the pyramid): (1) a displacement estimate; (2) an accuracy code; and (3) an error code. The error code is included to help in examining the action of the algorithm. Figures 24 and 25 show how this expanded definition of the displacement pyramid fits into the hierarchical matching algorithm.

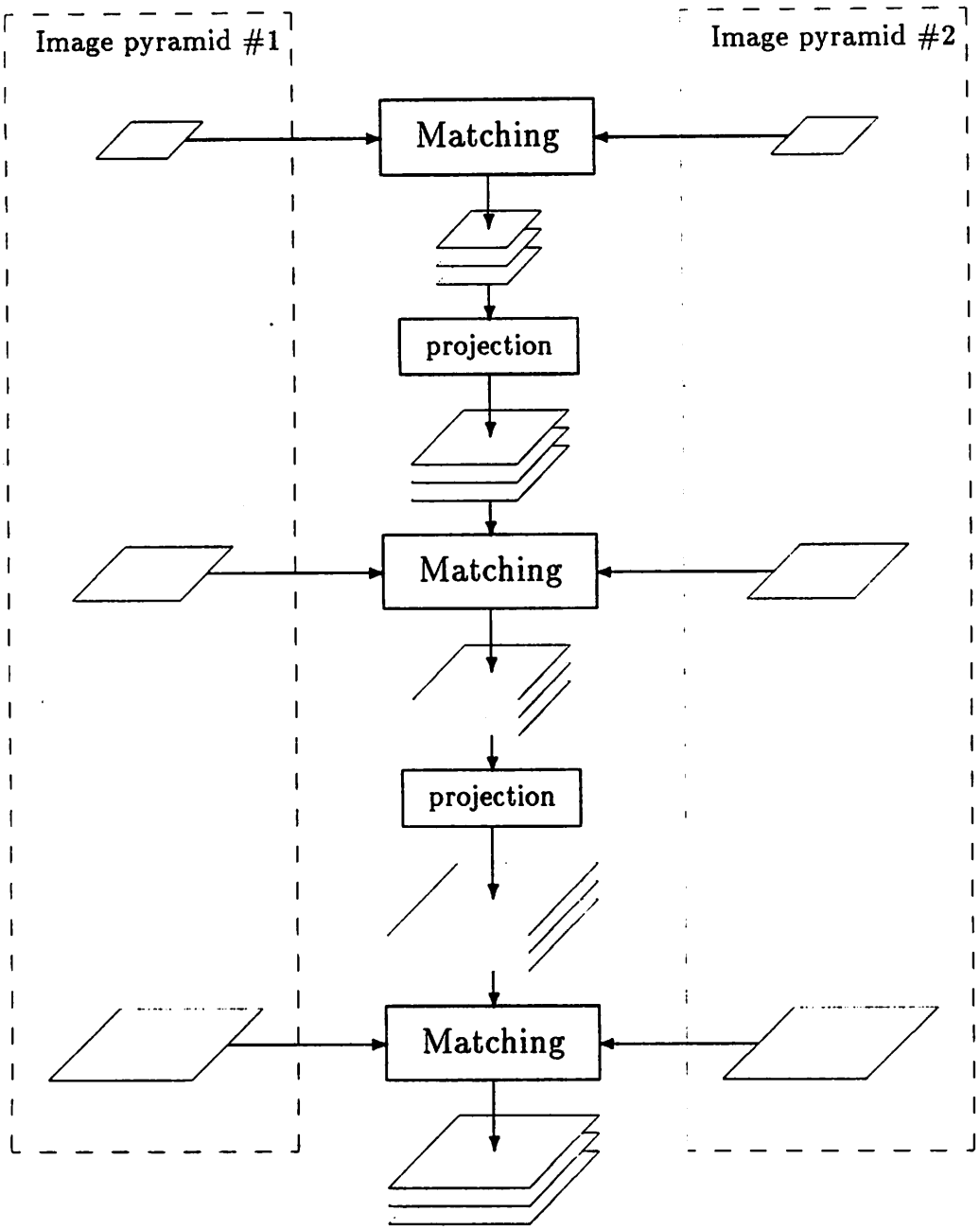


Figure 24: Hierarchical correlation: data flow

$(\mathbf{u}_f, acc_f, err_f) \leftarrow \text{Hicorr}\big(f_1, f_2, (\mathbf{u}_c, acc_c, err_c), matchable\big)$

For all (pixels x) **Do** {
  If $err_c \neq$ NO ERROR **Then** $err_f = err_c$,
  If the search area overflows **Then** err_F = SA_OVERFLOW,
  If ($err_f \neq$ NO_ERROR) OR (NOT$matchable$) **Then** {
    $acc_f \leftarrow acc_c + 1$,
    $\mathbf{u}_f \leftarrow \mathbf{u}_c$,
    FINISHED WITH THIS PIXEL },
  $N \leftarrow$ Maximal sample window indices,
  $w_1 \leftarrow \text{nbrhd}(f_1, \mathbf{0}, N)$,
  For $\delta \in \{-1, 0, 1\}^2$ **Do** {
    $w_2 \leftarrow \text{nbrhd}(f_2, \mathbf{u}_c + \delta, N)$,
    $corr(\delta) \leftarrow C(w_1, w_2)$ },
  $(\delta_M) \leftarrow$ position at which $corr(\cdot)$ is maximum,
  $\mathbf{u}_f \leftarrow \mathbf{u}_c + \delta_M$,
  $acc_f \leftarrow 0$
  }

Figure 25: Hierarchical correlation: pixel processing

The hierarchical correlation operation takes two frames of image data (f_1 and f_2), coarse disparity data ($\mathbf{u}_c, acc_c, err_c$) and matchability flags ($matchable$) as input. It outputs an updated disparity field ($\mathbf{u}_f, acc_f, err_f$). The function nbrhd($f, \Delta, N$) returns the sample window centered at the pixel $\mathbf{x} + \Delta$, where N defines the relative indices of the point to be included in the sample window. C is the correlation measure, typically direct correlation: $C(w_1, w_2) \triangleq \sum w_1(\cdot) \times w_2(\cdot)$.

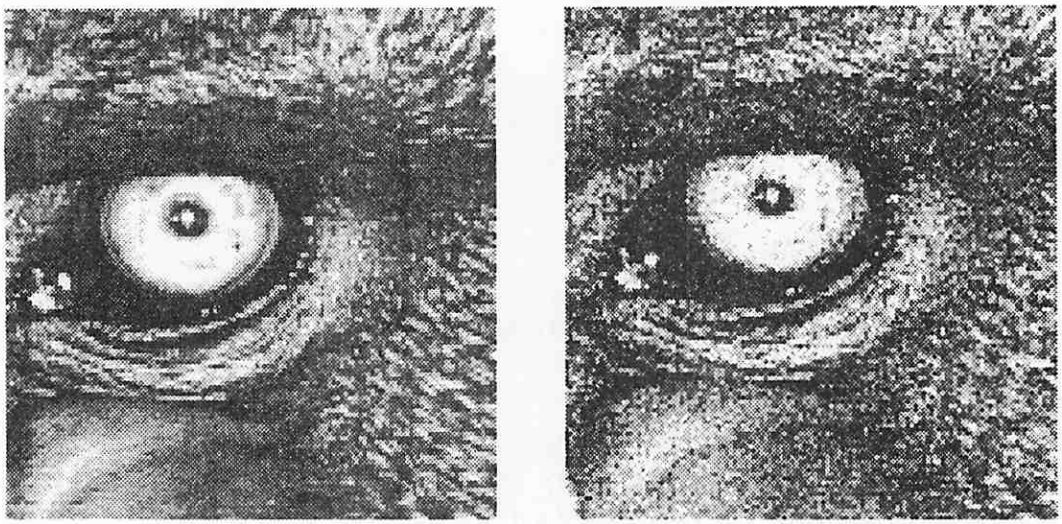


Figure 26: Mandrill eye images used in the first experiment

(a) A 128^2 piece of the larger mandrill image.
(b) A similar piece, translated 5 pixels up and 7 to the right, with white Gaussian noise added (standard deviation = 10% of full range).

6 Experiments

6.1 Mandrill image experiments

In this experiment we took the familiar example USC image of a mandrill and extracted a 128^2 subimage of it (Figure 26a). We created a second image by adding white Gaussian noise to this image and translating it 5 pixels up and 7 pixels to the right with respect to the first image (Figure 26b). The standard deviation of the noise added was 25.0 which is 10% of the intensity range of the image.

We conducted two experiments with these images. The first was the hierarchical matching process. The Laplacian pyramids were constructed using Burt's techniques with the discrete Gaussian reduction operator G_3 described in Chapter II. The matching at each level was done using an 8×8 sample window at all points in the image. The results at the various levels are shown in Figure 27

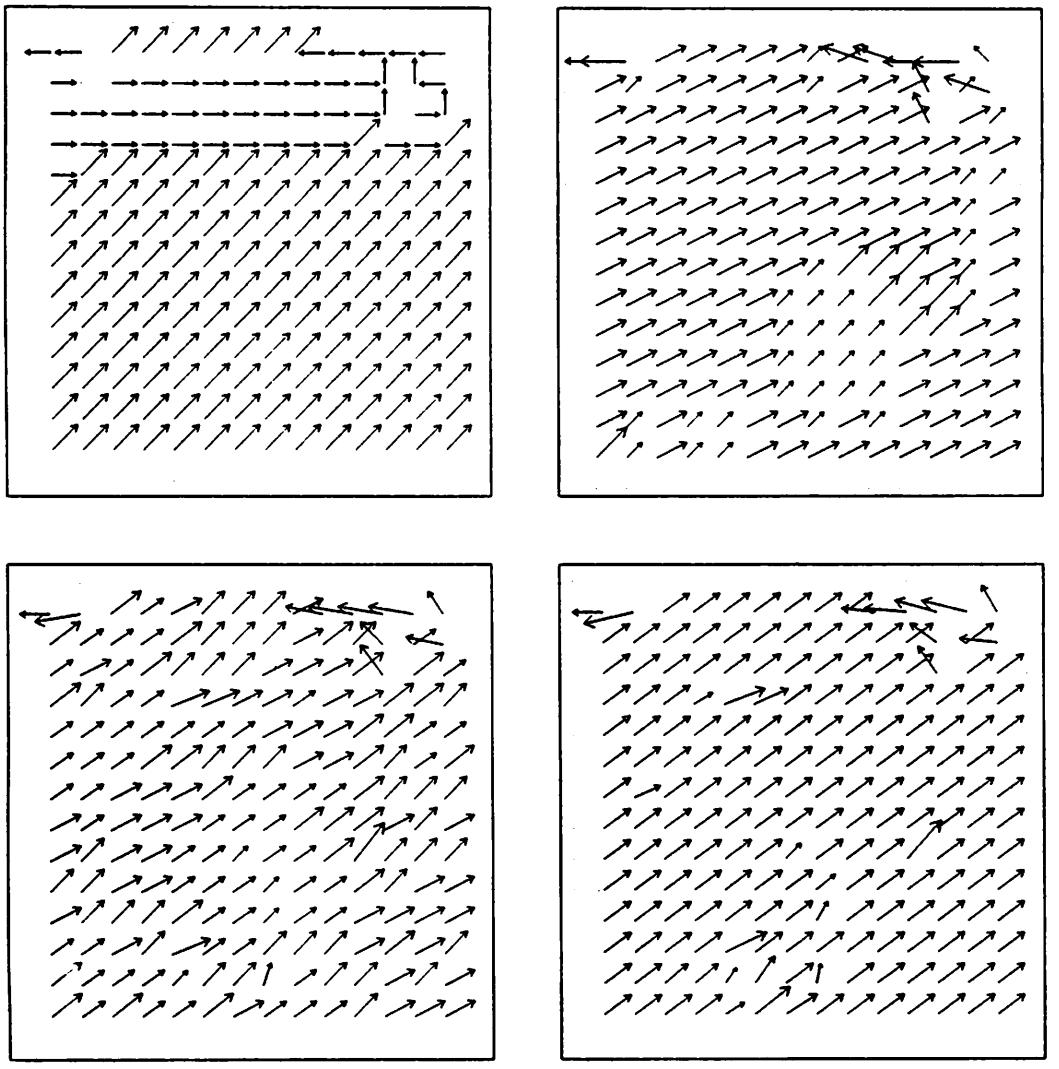


Figure 27: Computed displacement vectors

The displacement vectors at levels 4 through 7 obtained in the Mandrill experiment. Only a 64^2 sample of vectors is shown at each level.

Level 4

	-1	0	1
-1	0	4	**148**
0	8	1	35

Level 5

	-3	-2	-1	0	1	2	3
-2	0	0	4	0	0	31	0
-1	9	0	7	0	153	**527**	1
0	14	0	0	0	0	0	0

Level 6

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-5	0	0	0	0	0	0	0	0	0	0	0	1	0	0
-4	0	0	0	0	5	0	0	0	0	0	0	0	0	0
-3	1	3	0	0	3	10	7	0	4	2	**708**	**359**	31	0
-2	4	1	3	0	5	0	0	0	5	58	**909**	**701**	62	2
-1	0	43	13	0	0	1	13	0	1	0	0	0	5	0
0	1	4	0	0	0	0	0	0	0	0	0	0	0	0
1	3	12	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28: Distribution of displacement vectors

The histograms of the row and column components of the displacements obtained in the Mandrill experiment. Levels 4 through 7 have been shown. Note the peak at (-5,7) in D. At each level, the pixel location nearest the expected disparity level is surrounded by a box. (At level 6 four pixels are equidistant from the expected disparity, so the box surrounds all four.)

Figure 28 continued

Level 7

	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-9	0	0	0	0	0	0	0	0	0	4	2	0	0	0	0
-8	0	0	0	0	0	0	0	0	2	4	2	0	0	0	0
-7	0	0	0	4	0	0	0	0	0	3	3	23	9	4	1
-6	0	4	3	4	0	0	0	0	4	0	0	5	4	17	0
-5	8	1	0	4	1	0	0	0	14	2	0	0	5	0	0
-4	6	0	0	0	1	2	1	0	1	6	0	0	0	0	0
-3	0	0	8	26	51	9	8	0	0	4	3	0	1	9	2
-2	0	0	3	27	6	0	0	0	0	0	0	0	10	3	7
-1	0	0	1	38	45	6	5	0	0	0	0	0	14	2	6
0	0	0	3	10	3	0	0	0	0	0	0	0	0	0	0
1	0	0	1	8	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	10	2	0	0	0	0	0	0	0	0	0	0
3	3	0	24	6	5	0	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
-10	0	0	0	0	0	0	0	0	0	3	0	0	0	0
-9	0	0	0	0	0	0	0	0	0	1	0	0	0	0
-8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7	0	1	0	5	0	11	12	8	1	12	1	0	0	0
-6	0	0	0	7	5	4	12	14	7	9	9	40	0	0
-5	0	3	0	39	7	50	57	10322	57	74	90	19	3	0
-4	0	0	9	56	51	58	7	4	1	35	31	25	0	5
-3	0	11	0	8	22	30	9	24	18	12	27	47	0	0
-2	0	0	0	0	0	0	0	0	0	0	0	12	0	0
-1	0	4	0	0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0

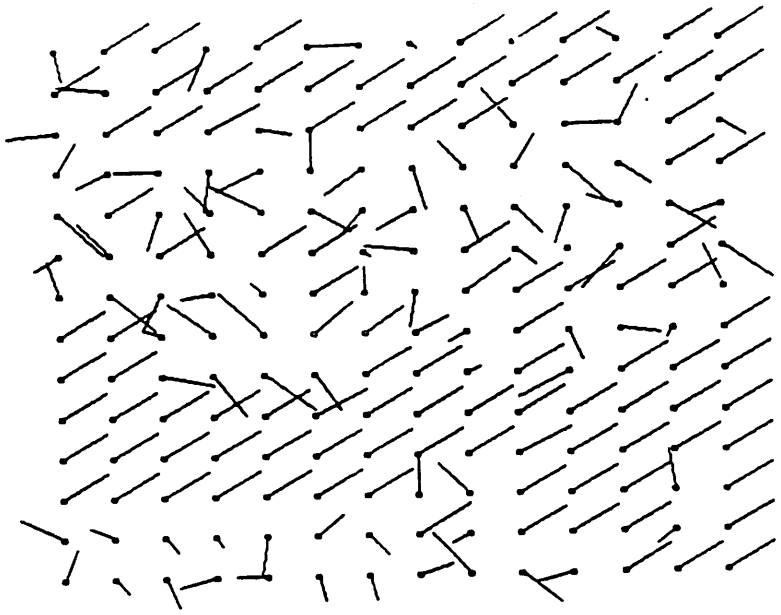
and Figure 28. Figure 27 shows results at levels 4,5,6, and 7. At each level the displacement estimates are shown at a sampling of 64 points.

In Figure 28 two-dimensional histograms of the row versus the column components of the displacements are shown for each of level 4 through 7 (Figure 28, a through d). Note, in Figure 28d, the high count found in the bucket corresponding to the correct displacement of (-5,7). The histogram for level 7 (Figure 28d) indicates that about 87% of the displacement values are exact. This shows that the hierarchical process is quite insensitive to noise.

In the second experiment, we attempted to match these two images using a correlation process all at one level. In doing this we used 8×8 sample windows and searched in a 17×17 search area around each pixel (the actual displacements of -5,7 will fall within this range). The results of this process are shown in Figure 29. Note the greatly reduced accuracy of this method (53% correct). Moreover, the incorrect matches had disparity values distributed uniformly throughout the range of possible values. Compare this to the hierarchical result, where the disparity values were clustered about the correct value. When the single-level method is used with larger disparities, requiring correspondingly larger search areas, incorrect disparities will be found yet farther from the correct value.

7 The Problem of False Matches

The match for a point obtained by the correlation technique may not always correspond to its environmental match. There are three basic reasons for this. First, correlation only deals with translational disparity (in the image plane) and so should break down with increasing rotational and scaling components in the disparity. However, as the experiments in the next section show, small rotations can be dealt with. Secondly, in practical imaging situations there can be significant amounts of noise in the images. Most often this would most adversely affect matching at finer resolutions. A third cause of false matches is the occurrence of occlusions in an image. Two problems can arise here (1) points in one image may have no counterpart



	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
-8	29	6	22	21	18	8	36	28	23	10	25	13	26	17	28	13	15
-7	8	5	10	51	19	8	53	28	10	7	7	16	13	20	6	13	16
-6	22	50	51	58	24	3	13	7	6	30	16	19	9	23	42	75	39
-5	11	18	19	21	15	2	29	4	18	22	5	33	37	20	72	**6279**	138
-4	3	17	19	17	11	12	17	1	36	5	8	19	13	22	29	16	64
-3	32	34	11	6	10	33	17	31	24	12	24	19	17	37	41	15	1
-2	4	8	31	24	49	26	39	5	23	9	8	52	3	9	9	29	11
-1	17	17	2	15	22	42	16	29	22	29	29	3	6	21	2	11	10
0	29	13	13	31	25	21	19	37	3	17	40	10	7	7	7	39	19
1	3	6	46	4	43	6	6	44	4	14	34	17	1	11	11	30	8
2	7	2	30	44	14	24	52	21	5	27	26	27	27	17	4	21	33
3	3	32	19	9	15	21	23	26	35	32	31	26	23	20	17	20	20
4	7	19	54	7	20	21	18	28	18	12	5	8	45	21	53	3	33
5	34	20	3	11	13	17	3	17	4	31	13	13	57	20	11	40	7
6	8	4	57	16	4	11	13	4	12	14	35	15	30	9	24	7	39
7	24	28	11	27	17	129	105	28	3	28	36	17	66	16	8	16	60
8	22	26	86	30	18	33	6	42	21	22	21	5	47	19	21	14	17

Figure 29: Single-level correlation

Results of the variance normalized correlation applied to the Mandrill images (Figure 26): (above) computed displacement vectors at some pixels; (below) the displacement histogram over all pixels. The histogram "bucket" corresponding to the correct match is row -5, column 7.

in the other image; (2) points on an occlusion boundary have neighboring windows which change identity from frame to frame.

7.1 Interest operators

Interest operators are designed to pick out points for which matches can be found with a high reliability. This detection of *matchability* is the key element of an interest operator. They can also be used to restrict processing to a small subset of all the image points to reduce computation costs. On serial machines this is certainly useful. However, on image parallel machines such as the processing cone, we need only be concerned with matchability.

In order that a point in one image image be matchable in another image, the point must be matchable with itself. For this to be true the local autocorrelation function must possess a strict local maxima. A sufficient condition for this is the occurrence of a strong corner at a point. Kitchen and Rosenfeld [Kitchen & Rosenfeld 82] present an analysis of various corner finding algorithms and these algorithms yield very good interest operators. Moravec [Moravec 81] gives an interest operator which attempts to compute the sharpness of an approximate autocorrelation function directly.

In the hierarchical correlation experiments described in the next section we have applied interest operators at the finest level of the image pyramid and then a logical pyramid (pixel values are single bit flags) is formed by using "OR" in the 4×4 reduction operation. Matching is only performed at those points which have a value of TRUE in this pyramid. This method is comparable to Moravec's search strategy [Moravec 81].

A alternative approach could use interest operators applied at all levels of the image pyramid. In this case there can be interesting pixels with uninteresting fathers. For these pixels, we can do one of two things: (1) the search can be done in a larger search area, or (2) a displacement estimate can be obtained based on neighboring pixels.

7.2 Sharpness measures

Interest operators rely on the distinctiveness of a feature as it it appears in the first image only. It is still possible that its counterpart in the second frame has been rotated or scaled, corrupted by noise, or obscured under an occluding object. Under such conditions, we would expect the correlation array to be poorly structured. Ideally, a sharply peaked correlation array provides an obvious choice of match point. Any of the aforementioned conditions should weaken or eliminate the sharp peak decreasing the likelihood of a correct match. Sharpness measures attempt to measure this quality of the correlation array [Williams & Glazer 82]. They can then be used in deciding whether to accept the peak of a correlation array as a match. We have not yet included such measures in our experiments.

A simpler measure of "goodness" of match is the height of the peak of the correlation array. With variance normalized correlation, for which all correlation values lie in $[-1, 1]$, a general threshold could be set. Without this normalization, it is not clear how this can be done.

Burt, Yen, and Xu suggested a confidence measure based on directional sharpness measures [Burt *et al.* 83]. This allowed them to specifically detect motion boundaries by comparing the results of their directional confidence measures. This idea has been further extended and applied to hierarchical correlation algorithms by P. Anandan [Anandan 84, Anandan 87], where it has been used to successfully detect motion boundaries.

8 Further Experiments

8.1 Optic fundus image experiments

The next problem to which we applied the hierarchical matching algorithm was that of registering two fluorescein angiogram images of the optic fundus (see Figure 30). These images were obtained from Paul Nagin at the Tufts New England medical center and are digitized as 128^2 images. The problem is to register two images

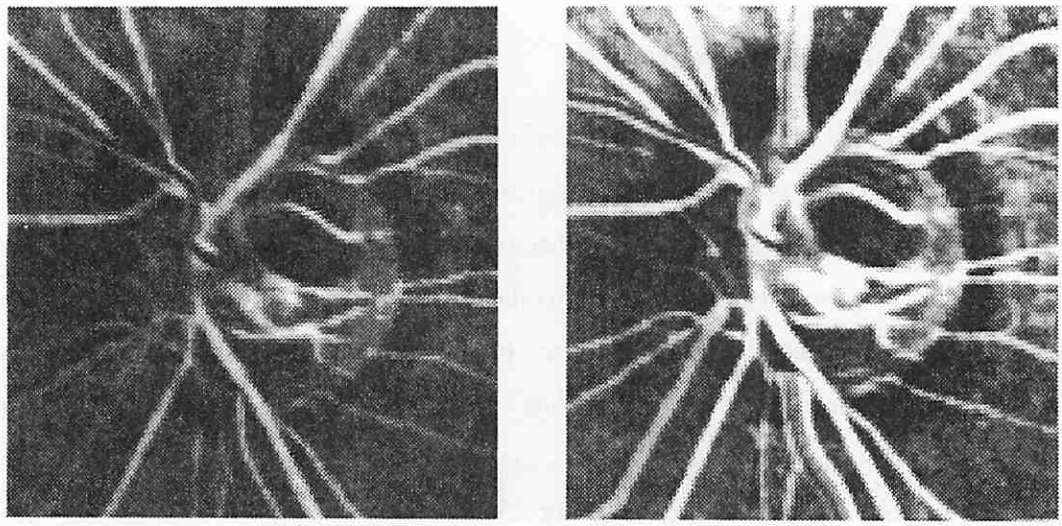


Figure 30: Optic fundus test images

These are real images taken at two successive time instants. Note the large change in mean intensity.

taken at the beginning and at the peak of dye filling. Areas which show very little change are recognized as regions where no filling of the dye is taking place. This measurement can then be used in the prognosis of glaucoma. Due to severe contrast changes over this time interval, it is necessary to register a temporal sequence of 8 to 10 images.

The hierarchical correlation matching algorithm was applied to the two images in Figure 30. A band-pass pyramid was built using G_3 as the reduction filter and bilinear interpolation (Figure 4) as the projection filter. Matching began at level 4 (16×16) working down to level 7 (128×128). Direct correlation was used with 8×8 correlation windows.

In Figure 31a–d we show the results of applying the matching algorithm at the levels 4 to 7. Again, we have subsampled the vector field at the three finer levels for display purposes. The displayed vectors have been drawn with a length equal to .9 times their actual value. This also aids display, as in Figure 31a, where

vectors pointing up 1 pixel do not touch the tail of the vector at the next higher pixel. At the finest level (Figure 31d), we see a roughly rotational disparity field centered about a point off the left side of the image. This is to be expected, as we describe next.

For the images used in this experiment any three-dimensional effects due to the movements of the photographed eye can be safely ignored because of the fixed relative position of the camera and the eye. The misregistration is due to a rigid motion in the plane caused by eye movements and mis-alignments in the digitization process. The problem then is to find the rigid motion which will bring the images into register. To analyze the accuracy of this experiment, the 2D rigid motion vector field that best fits the computed disparity vectors was found.

A rigid motion in the plane can be represented as a rotation and a translation in many ways. For any choice of translation, a corresponding rotation can be found that results in the given rigid motion. In our motion field estimation algorithm, we first estimate a translation component for which we know where the corresponding roation is centered. Then the angle of rotation is estimated use a least-square error method.

If the displacement at a given point in a rigid motion field is subtracted from the field, the result is a rotational field about that point. (For a translational field, the resulting rotation is the null rotation.) We cannot do this with an arbitrary point in the computed field since its displacement vector is not known precisely. Instead, we use the fact that: the average displacement over a discrete set of points is equal to the displacement at the centroid of those points. The discrete set of points we use are the pixel locations at which displacement vectors have been computed. If we choose the centroid of the points as that whose displacement vector will be subtracted out, we obtain a representation of the rigid motion as a translation equal to the mean translation of the points and a rotation about the centroid. Since the translation vector is a mean over a large number of computed displacement vectors, it is a very good estimate of the actual displacement at that point. The centroid is simply that point with coordinates equal to the mean of the respective coordinates

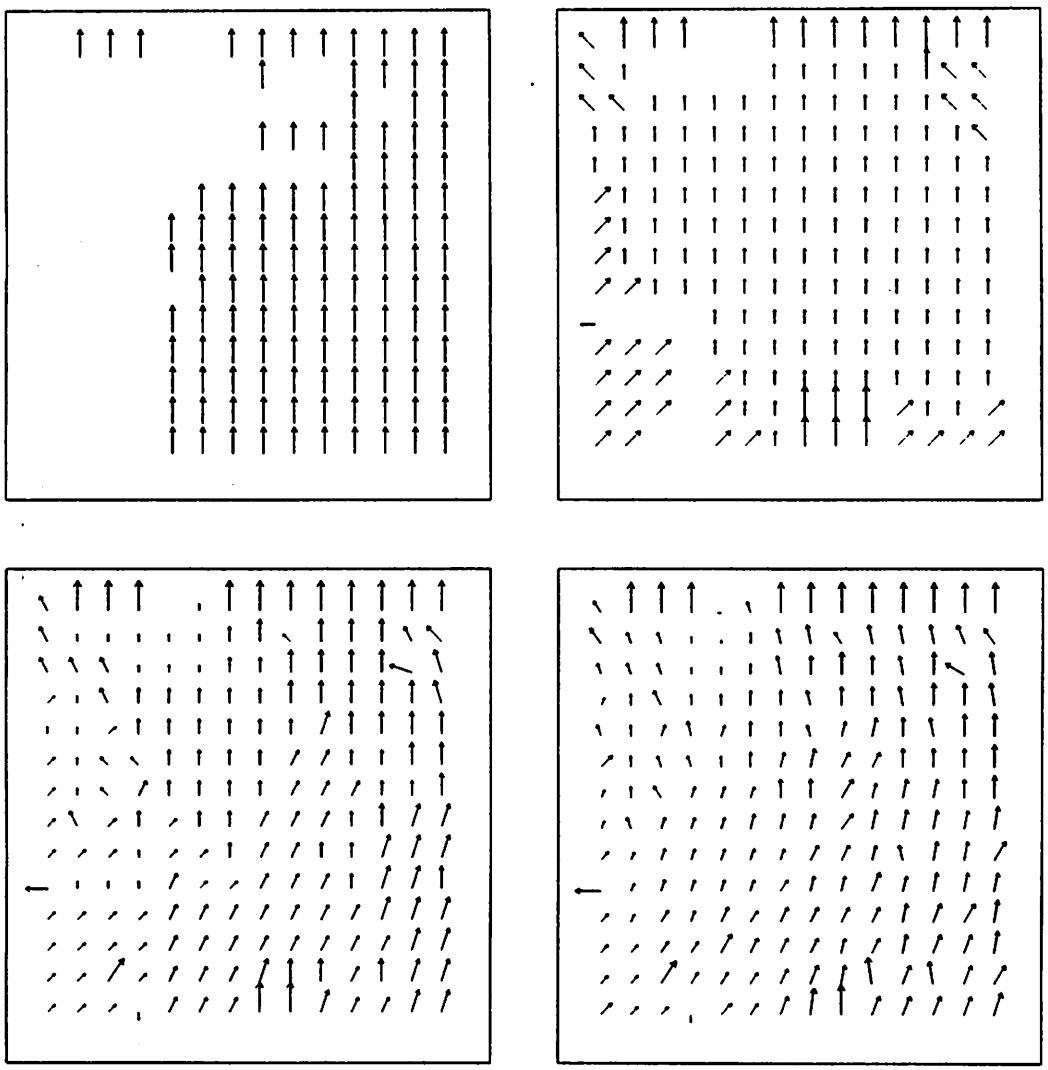


Figure 31: Computed displacement vectors

The displacement vectors at levels 4 through 7 obtained in the optic fundus experiment.

(a) Level 4, all pixels

(b) Level 5, every other row and column, starting at (0,0)

(c) Level 6, every fourth row and column, starting at (1,1)

(d) Level 7, every eighth row and column, starting at (3,3)

of the points in the set. The parameters of the rigid motion field are estimated as follows: (1) the translational component is the mean over the set of points; (2) this vector is subtracted from all the vectors in the discrete set; and (3) the residual vector field is fit by a rotational field centered at the centroid of the points. The angle of this rotation is estimated by a least root-mean-square fit.

This analysis was performed on the computed displacement field of Figure 31d using the vectors at all pixels in the rectangular region from row 12 to 119 and column 8 to 119. This restriction excluded pixels with limited accuracy due to search area overflow. The points used comprise the largest rectangular region containing no error pixels. The centroid of this region is clearly (65.5,63.5). A few more non-error pixels could have been included, requiring a less immediate but still straightforward centroid calculation. The mean disparity over the selected region was computed and that value was subtracted from each vector in the region. A best fit (in the mean square sense) rotational field centered at (65.5,63.5) was then computed.

To measure the accuracy of the matching algorithm we generated a rigid motion vector field using the translation and rotation computed above. The vectors in this field were represented with real numbers; they were not truncated to the nearest integer (pixel spacing). This field was subtracted from the displacement computed by the matching algorithm thus generating an error vector field. A two-dimensional histogram of the row versus the column components of the error vectors is shown in Figure 32. The (0,0) bucket in the histogram represents all error vectors with both components between $-\frac{1}{2}$ and $+\frac{1}{2}$. In general, the (r,c) bucket in the histogram represents all error vectors with row component between $r - \frac{1}{2}$ and $r + \frac{1}{2}$ and column component between $c - \frac{1}{2}$ and $c + \frac{1}{2}$

The central (0,0) bucket counts those pixels at which the computed disparity agrees exactly with the estimated rigid motion field. In this case, it includes 42% of all pixels in the rectangular region of 12,096 pixels. The nine buckets including the central one and its immediate neighbors count those pixels at which the computed disparity was within one pixel spacing (in the row and column direction). In this case, they include 87% of the pixels. The 5×5 set of buckets about bucket (0,0),

	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5
-7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
-5	0	0	0	0	0	0	0	0	13	0	0	0	3	0	0	0
-4	0	0	0	0	0	4	0	12	59	0	0	0	8	0	11	0
-3	0	1	0	0	0	0	0	3	49	20	0	19	36	1	2	0
-2	0	0	7	8	0	0	3	14	27	9	16	42	4	8	4	0
-1	0	0	0	0	0	0	12	15	85	414	1035	280	44	32	17	0
0	0	0	0	0	0	0	0	0	30	885	5074	1340	200	4	0	0
1	0	0	0	0	0	0	12	17	76	120	798	563	95	27	0	0
2	0	0	7	7	6	0	29	2	13	44	22	86	31	5	0	0
3	0	0	0	0	0	14	26	6	53	39	10	6	4	0	0	0
4	0	0	0	0	17	0	11	0	46	0	0	0	4	10	0	0
5	0	0	0	0	4	2	0	7	16	0	0	0	0	0	0	0
6	0	0	0	0	0	0	4	2	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32: Distribution of error vectors

Histogram of the error vectors. The error vectors are obtained by subtracting (1) the field generated by the translational and rotational parameters derived from the computed field, from (2) the computed displacement field. The error vectors have real number components. The (0,0) bucket in the histogram represents all error vectors with both components between $-\frac{1}{2}$ and $+\frac{1}{2}$. In general, the (r,c) bucket in the histogram represents all error vectors with row component between $r - \frac{1}{2}$ and $r + \frac{1}{2}$ and column component between $c - \frac{1}{2}$ and $c + \frac{1}{2}$

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5
-6	0	0	0	0	0	0	0	0	0	0	0	0	0
-5	0	0	0	0	0	2	0	0	0	0	0	0	0
-4	0	0	0	0	1	3	0	0	0	0	0	1	0
-3	0	0	0	0	0	10	2	0	9	1	1	0	0
-2	0	0	0	0	3	3	0	0	3	2	1	0	0
-1	0	0	0	1	3	26	97	228	52	4	0	1	0
0	0	0	0	0	0	5	145	1099	338	36	0	0	0
1	0	0	0	1	0	22	12	111	125	13	1	0	0
2	0	0	0	13	0	2	3	0	3	7	0	0	0
3	0	0	1	4	0	5	1	0	0	0	0	0	0
4	0	8	0	0	0	3	0	0	0	0	2	0	0
5	0	2	0	0	1	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33: Distribution at interesting points

Using the same data as in previous figure, this histogram was obtained by including only the displacement vectors at interesting points. The interesting points were selected using the gradient-weighted planar curvature (*GWPC*) operator used by the Kitchen corner finder. Those points with a *GWPC* magnitude in the top 20% were used.

which counts vectors with errors of two pixel spacings, includes 94% of the pixels.

Figure 33 shows the same type of histogram with the errors counted only at a set of interesting points. In this case 2,418 interesting points (20%) were selected at the finest level by thresholding the absolute value of the gradient weighted planar curvature operator (*GWPC*) used in the Kitchen-Rosenfeld corner finder [Kitchen & Rosenfeld 82]. The *GWPC* operator involves first and second finite differences and includes division by the gradient magnitude, making it sensitive to noise in the image. To reduce noise sensitivity, we chose to use larger finite difference masks. The 5×5 masks developed by [Beaudet 78] were used.

The disparity vectors at the selected interesting points could have been computed with a hierarchical correlation algorithm that uses a interesting point pyramid

built bottom-up (fine-to-coarse) by "OR-ing" up from the interesting points found at the first level.

In Figure 33, it can be seen that the range of error magnitudes is significantly reduced. The match counts go up to 45% (1099/2418) for nearest-pixel accuracy, 91% (2207/2418) for within-one-pixel accuracy, and 97% (2336/2418) for within-two-pixels accuracy. This improvement, expected for interesting points, is not dramatic. In fact, when match results are considered relative to the full range of values of the *GWPC* operator, the correlation between these values and matching accuracy is minor.

Similar results were obtained using another interest operator based on finite differences. The $Det \triangleq I_{xx}I_{yy} - I_{xy}^2$ operator, mentioned in Appendix A and also used by [Kitchen & Rosenfeld 82], was also considered in an analysis comparable to that used with the *GWPC* operator. Again the 5×5 Beaudet masks were used. Unlike the *GWPC* operator, positive and negative values of the *Det* operator signify different types of image structure. Positive values of the *Det* detect "peaks" and "pits", i.e. local maxima and minima, while negative values of *Det* detect "saddle points". We can think of there as being two operators $+Det$ and $-Det$, only one of which returns a value at a given pixel. The $+Det$ operator showed a minor correlation with matching accuracy comparable to *GWPC*. The $-Det$ operator showed a negative correlation: large negative values corresponded to reduced matching accuracy. It appears that the $-Det$ operator is better at picking out noisy points than interesting points.

The minor improvements introduced by the *GWPC* and the $+Det$ interest operators emphasize the fact that the results considered over all pixels, i.e. in the absence of an interest operator, are already quite good. This is likely due to the 8×8 correlation windows used, which in most locations capture significant details. If so, then using larger finite difference masks, say 7×7 or 9×9, or presmoothing with a low-pass filter will not significantly improve results. In any case, the added cost of interest operators does not warrant their use under the given parameters used in our hierarchical algorithm. Sharpness measures, which compute a post-correlation

match confidence [Burt *et al.* 83, Anandan 84], provide a better alternative when improved accuracy is required.

9 Summary

In this chapter we described a hierarchical approach to correlation matching that improves on single-level methods by easing the computational burden and overcoming the problem of false matches. The image pyramids introduced in Chapter II provided the multiresolution image data to be matched. Hierarchical algorithms that fit into the processing cone architecture were developed and run on real data.

Standard single-level correlation matching techniques were described in Chapter III. At the start of this chapter, the inadequacies of such methods were noted: large search areas that call for expensive searches and repeating features that result in incorrect matches. In both cases, the use of coarse-to-fine matching techniques greatly improve a correlation-based algorithm. The reduced costs were shown by the analysis summarized in Table 2 on page 96. The improvement in accuracy was shown by comparing single-level and hierarchical algorithms applied to the same image data. The results of our experiments indicate that the hierarchical algorithm is also insensitive to noise and is able to detect at least small amounts of rotation between images.

In the next chapter a hierarchical gradient-based motion detection algorithm will be presented that extends the gradient-based method to handle larger disparities. These disparities are comparable in size to those detected by the hierarchical correlation algorithm. The computational costs of these two methods will be compared in that chapter.

CHAPTER V

Hierarchical Gradient-Based Methods

1 Introduction

This chapter describes a hierarchical approach to gradient-based motion computation. It generalizes gradient-based techniques to handle cases for which the disparities between points in two images are more than a few pixels in length. The hierarchical control is very similar to that used for hierarchical correlation matching in the previous chapter. Coarse estimates of disparity at high levels of the processing cone are used at lower levels to guide gradient-based algorithms which compute increasingly accurate (higher resolution) disparity vectors.

The success of the gradient-based methods depends on the assumption that the image value (intensity) changes in local neighborhoods can be adequately approximated by the low order terms of their Taylor series expansions. In particular, first order methods depend on the degree to which image structure is locally linear. This assumption is not the case when (1) the optic flow (disparity) is relatively large, or (2) the extent over which the intensity varies linearly is small. In either case, experiments presented later in this chapter show that a gradient-based algorithm operating at a single level of resolution will fail to detect correct disparities. Figures 46–49, in Section 5.1, show a progression of good to bad results for a sequence of image pairs with increasing disparities. Experiments on the same image data show that the hierarchical algorithm can compute the large disparities.

A little more formally, consider the following first order approximation used

in first order gradient-based methods

$$F(x+U, y+V) = F(x,y) + \nabla F \cdot (U,V) \tag{13}$$

Let r be the radius about (x,y) within which this approximation holds (to some minimal ϵ). To use this equation (and those we have derived previously) in the computation of a disparity vector (U,V), we want $\|(U,V)\| < r$. This condition may be violated in two ways: (1) $\|(U,V)\|$ is too large — i.e. large disparities; or (2) r is too small — strong second order (or higher) image structure.

The first problem can be overcome if a coarse estimate of disparity is known and only an update need be computed. If the coarse disparity estimate is a good one then the required update vector is small and will lie in the "linear" neighborhood. This suggests a coarse-to-fine approach to disparity computation comparable to that used in Chapter IV. The second problem is avoided by eliminating higher-order image structure. Such elimination of fast variations in the image can be done by low-pass filtering — smoothing. Of course the smoothing will eliminate detail in the frames and lower the resolution of the computed disparity field. The accuracy of disparity values is reduced and fast variations are lost. In a sense the disparity has itself been smoothed. This suggests that disparity computations on smoothed data can be performed on subsampled image data.

The combination of coarse-to-fine stages with (proportional) subsampling at the coarser stages leads us to conclude that *first order gradient-based methods may be extended to non-trivial disparity computations by formulating a hierarchical generalization of the single-level method that operates on low-pass image pyramids.* (By "non-trivial" we mean disparities more than just a few pixels in length.)

In the hierarchical method, approximate disparities are refined at each level of the processing cone in a coarse-to-fine sequence. At each level, the update vectors are expected to be of bounded magnitude, relative to the image scale (pixel spacing) at that level. This constraint provides a *consistency check* which can be used to "filter" out bad update vectors that might result from noisy image data or violations of the method's assumptions.

The next section summarizes the hierarchical method of applying gradient-based disparity computation. Section 3 presents the technical details. Then in Section 4 a hierarchical gradient-based algorithm is described. Experiments in Section 5 show how single-level methods fail and how the hierarchical method succeeds. In the next chapter, we discuss the computational costs of hierarchical gradient-based methods and compare them to hierarchical correlation.

2 The Hierarchical Method

Initially we are given two input images $F_1(i,j)$ and $F_2(i,j)$ for which the disparity between them must be computed. The disparity will be represented as a vector field $(s,t) = (s(i,j),t(i,j))$ such that a given disparity vector $(s(i,j),t(i,j))$ describes the displacement between a point (i,j) in F_1 and its "match" at $(i+s,j+t)$ in F_2. Low-pass image pyramids $\mathcal{F}_1$ and $\mathcal{F}_2$ are formed for each of the two frames. Processing begins at a level l_{top}, high enough in the pyramid so that the maximum disparity is less than 1 pixel in magnitude. We assume that the coarse disparity field at level $l_{top}-1$ is $(0,0)$, that is $(0,0)$ at all pixels.

The following processing is performed in a coarse-to-fine succession of stages, beginning at level l_{top} and proceeding down the pyramid (increasing level numbers) to the finest level. First, the coarse disparity field at level $k-1$ is projected down to level k, giving an **approximate disparity field** at that level. Then using the two frames $f_1 = \mathcal{F}_1^k$ and $f_2 = \mathcal{F}_2^k$ at level k in the image pyramids and the approximate disparity field $(U,V) = (U(i,j),V(i,j))$, an update vector field $(u,v) = (u(i,j),v(i,j))$ is computed. The **update vector field** is added to the approximate disparity field to give the more accurate updated disparity field $(U+u,V+v)$. The update vector field is computed at each point (i,j) by comparing data in the neighborhood of $f_1(i,j)$ to data in the corresponding neighborhood $f_2(i+U,j+V)$. The update vector field (u,v) is the relative translation between these two neighborhoods. It can be computed using a first order gradient-based method. The relationship between these vectors at a given pixel is shown in Figure 34.

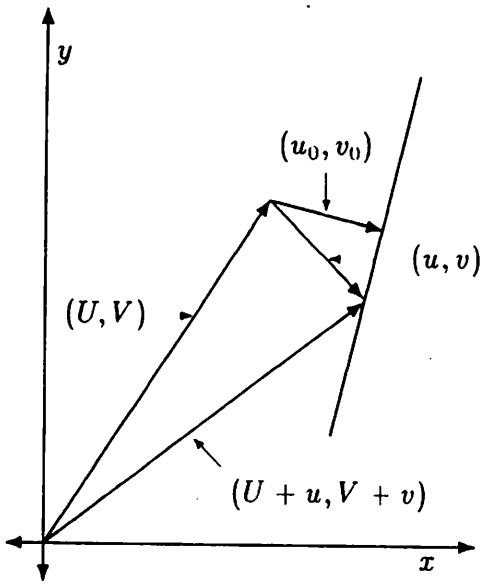


Figure 34: Geometry of updating process

The refinement of the disparity estimate at a given point in the first frame involves the following vectors: (U,V) is the approximate disparity vector; (u,v) is an update vector; and $(U+u,V+v)$ is the updated disparity. The edge flow (u_0,v_0) is the component of (u,v) parallel to the mean spatial gradient of the images at the points being compared. It defines a constraint line which is the locus of points upon which (u,v) is found.

The gradient-based algorithm uses two stages. In the first stage the edge flow is computed at each pixel. It is the component of the update vector parallel to the mean spatial gradient at $f_1(i,j)$ and $f_2(i+U, j+V)$. The edge flow vector, along with the approximate disparity, defines a line which is a locus of points upon which the update vector lies. In Figure 34 the edge flow vector is shown as (u_0, v_0).

In Chapter III, Section 4.3, various methods of computing optic flow from edge flow were discussed. We will use an *optimization* method. In Chapter III, Section 4.2, an application of first order methods to optic flow computation was formulated. An analogous formulation for the case of disparity can be done but it is constrained to work only in the case of small disparities. The next section presents a more general development for use in a hierarchical scheme, in which approximate disparity information is utilized to overcome these constraints.

3 Formal Development

3.1 Computing a Refined Disparity Field

In this section we develop a first order gradient analysis for disparity computation analogous to that given for flow in Chapter III, Section 4.2.

Let $F_1(x,y)$ and $F_2(x,y)$ be two frames of an image sequence and let $(U,V) = (U(x,y), V(x,y))$ be a vector field approximately describing the disparity from F_1 to F_2. We want to find a vector field $(u,v) = (u(x,y), v(x,y))$ which "updates" (U,V), i.e. the new vector field $(U+u, V+v)$ is a better approximation of the disparity from F_1 to F_2. In the following development we will use the notation $\mathbf{x} = (x,y)$, $\mathbf{U} = (U,V)$ and $\mathbf{u} = (u,v)$.

3.1.1 The Constraint Line

First we will find a constraint line for $\mathbf{u}$. We define the change in image value between a point $\mathbf{x}$ in F_1 and a corresponding point in F_2 at the displaced point

$\mathbf{x} + \mathbf{U} + \mathbf{u}$ as:

$$\triangle F(\mathbf{x}) = F_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) - F_1(\mathbf{x}) \tag{14}$$

Consider the following two ways in which a Taylor series expansion of F_2 can be computed:

$$\begin{aligned} F_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) &= F_2(\mathbf{x} + \mathbf{U}) + \mathbf{u} \cdot G_2(\mathbf{x} + \mathbf{U}) \\ &\quad + \frac{1}{2}\mathbf{u}^T \big(H_2(\mathbf{x} + \mathbf{U})\big)\mathbf{u} + \text{higher order terms} \end{aligned} \tag{15}$$

and

$$\begin{aligned} F_2(\mathbf{x} + \mathbf{U}) &= F_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) + (-\mathbf{u}) \cdot G_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) \\ &\quad + \frac{1}{2}(-\mathbf{u})^T \big(H_2(\mathbf{x} + \mathbf{U} + \mathbf{u})\big)(-\mathbf{u}) + \text{higher order terms} \end{aligned} \tag{16}$$

where G_2 is the gradient of F_2 and H_2 is the Hessian of F_2. The gradient and the Hessian of a two-dimensional image are generalizations of the first and second derivative of function of one variable. They are defined in Appendix A. The last two equations can be combined to give:

$$\begin{aligned} F_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) &= F_2(\mathbf{x} + \mathbf{U}) + \mathbf{u} \cdot \frac{1}{2}\big[G_2(\mathbf{x} + \mathbf{U}) + G_2(\mathbf{x} + \mathbf{U} + \mathbf{u})\big] \\ &\quad + \frac{1}{2}\mathbf{u}^T \frac{1}{2}\big[H_2(\mathbf{x} + \mathbf{U}) - H_2(\mathbf{x} + \mathbf{U} + \mathbf{u})\big]\mathbf{u} \\ &\quad + \text{third and higher order terms} \end{aligned} \tag{17}$$

In the second section of Appendix A, it is shown that if $\mathbf{U} + \mathbf{u}$ varies slowly, i.e. $|\nabla(\mathbf{U} + \mathbf{u})|^2$ is small, then the following approximations can be made: $G_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) = G_1(\mathbf{x})$ and $H_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) = H_1(\mathbf{x})$. We then get:

$$\begin{aligned} F_2(\mathbf{x} + \mathbf{U} + \mathbf{u}) &= F_2(\mathbf{x} + \mathbf{U}) + \mathbf{u} \cdot \frac{1}{2}\big[G_2(\mathbf{x} + \mathbf{U}) + G_1(\mathbf{x})\big] \\ &\quad + \frac{1}{2}\mathbf{u}^T \frac{1}{2}\big[H_2(\mathbf{x} + \mathbf{U}) - H_1(\mathbf{x})\big]\mathbf{u} \\ &\quad + \text{third and higher order terms} \end{aligned} \tag{18}$$

Substituting into equation 14 and *ignoring higher order terms* finally gives:

$$\triangle F(\mathbf{x}) = F_2(\mathbf{x} + \mathbf{U}) - F_1(\mathbf{x}) + \mathbf{u} \cdot \frac{1}{2}\big[G_2(\mathbf{x} + \mathbf{U}) + G_1(\mathbf{x})\big] \tag{19}$$

If $\mathbf{U}$ is a good estimate, then $\mathbf{u}$ will be small and hence dropping higher order terms leaves us with a good approximation. In fact the second order term is very small due to the fact that $H_2(\mathbf{x}+\mathbf{U})$ is very close to $H_1(\mathbf{x}) \approx H_2(\mathbf{x}+\mathbf{U}+\mathbf{u})$. That is why we used both of the Taylor series expansions in Equations 15 and 16 and not just one of them.

To simplify our equations we define

$$\hat{F}_2(\mathbf{x}) \triangleq F_2(\mathbf{x}+\mathbf{U}) = F_2(\mathbf{x}+\mathbf{U}(\mathbf{x})) \tag{20}$$

and

$$\overline{F} \triangleq \frac{1}{2}(F_1 + \hat{F}_2) \tag{21}$$

Then the gradient of $\overline{F}$ is equal to $\nabla\overline{F} \triangleq (\overline{F}_x, \overline{F}_y) = \frac{1}{2}(G_1 + \hat{G}_2)$ where G_1 is the gradient of F_1 and $\hat{G}_2$ is the gradient of $\hat{F}_2$. We can then write

$$\triangle F(\mathbf{x}) = \hat{F}_2(\mathbf{x}) - F_1(\mathbf{x}) + \mathbf{u} \cdot (F_x, F_y) \tag{22}$$

If we set $\triangle F$ to zero then equation 22 defines a constraint line upon which the update vector $\mathbf{u}$ must lie.

Two key assumptions in the above derivation were that (1) $\mathbf{U}(\mathbf{x}) + \mathbf{u}(\mathbf{x})$ varies slowly, and (2) second and higher order terms in the Taylor series expansion can be ignored. The first assumption does not hold at motion boundaries. This limitation is the same as that which occurs when gradient-based flow computations are performed. Hence, similar solutions, whatever they might be, can be used.

The (first order) Taylor series approximation only holds within some local neighborhood. If the disparity is greater than the "radius" of this neighborhood then the method doesn't work. The second assumption is warranted because the approximate disparity field $\mathbf{U}(\mathbf{x})$ allows us to combine higher order components from approximately equivalent image neighborhoods. This is a key element of the hierarchical method. Imagine the above derivation was performed for the non-hierarchical case. This is easily done by setting $\mathbf{U} = (0,0)$ — just eliminate all appearances of $\mathbf{U}$ in Equations 14 to 22. The second order terms in Equations 17

and 18 then contain the factors $\left[H_2(\mathbf{x}) - H_2(\mathbf{x}+\mathbf{u})\right] \approx \left[H_2(\mathbf{x}) - H_1(\mathbf{x})\right]$. These terms will NOT be negligible when the actual disparity is large.

Theoretically, the second assumption does not pose a problem for the case of flow computation which is formulated in a continuous three-dimensional XYT space. However, in real imaging systems we are actually processing discrete frames taken with a non-trivial $\triangle t$ and approximating derivatives with finite differences. This introduces a non-infinitesimal jump which may exceed the radius of approximation.

3.1.2 Edge flow

Again we define **edge flow** as the point on the constraint line lying nearest the origin. The edge flow is then

$$\mathbf{u}_0 \triangleq \frac{-(\hat{F}_2 - F_1)}{|\nabla F|}\frac{\nabla \overline{F}}{|\nabla F|} = \left(\frac{-\overline{F}_x(\hat{F}_2 - F_1)}{\overline{F}_x^2 + \overline{F}_y^2}, \frac{-\overline{F}_y(\hat{F}_2 - F_1)}{\overline{F}_x^2 + \overline{F}_y^2}\right) \tag{23}$$

This vector is used as the initial estimate of the update vector $\mathbf{u}$.

Both $\mathbf{u}$ and $\mathbf{u}_0$ lie on the constraint line and $\mathbf{u}_0$ is that point on the line nearest the origin. Thus, $\mathbf{u}_0$ must be smaller (in magnitude) than $\mathbf{u}$, that is $\|\mathbf{u}_0\| \leq \|\mathbf{u}\|$. In a hierarchical scheme, approximate disparity vectors $\mathbf{U}$ insure us that the update vector must be bounded in magnitude. This in turns places a bound on the magnitude of the edge flow vector. In Section 4.4 this fact is used to introduce a consistency check on the computed edge flow. *This is an added benefit of the hierarchical method.*

3.1.3 Computing the Update: An Optimization Problem

Equation 22 provides a constraint on the value of the update vector at each point in the image space (at a given level). The assumption that the change in image value at the matching points vanishes, i.e. $\triangle F = 0$, specifies a line in "update vector" space upon which the update vector should lie. Due to noise in either image, this is likely to be too strong a constraint. A more practical constraint requires that the update vector lie as close to the line as possible.

In the second stage of a first order method, other information is brought to bear to finally arrive at update vectors at each pixel. In the motion problem we have posed, disparity vector fields can vary across the image, but locally they are almost constant. Thus, we must use local disparity information to arrive at a final update value at a given pixel. We will use an optimization method, which while formulated as a global optimization problem, can be represented as a set of local constraints and can be solved by a local, parallel, uniform algorithm. Horn and Schunck's application of such a method to the problem of computing optic flow from edge flow was described in Chapter III, Section 4.3.4. The analysis here is similar, using image differences in place of partial derivatives with respect to time and replacing the optic flow field by an approximate disparity field summed with an update vector field. (Compare the equations in Chapter III, Section 4.3.4, with the equations in this section.)

The variational problem we wish to solve asks for a disparity update field for which (1) the update vector lies as close to the constraint line as possible, and (2) the disparity field is as smooth as possible. Since "close"-ness and "smooth"-ness are measured in different units, a constant of proportionality (α in the equation below) must be included as a parameter to relate these two values.

The problem then posed is: given an approximate disparity field (U, V), and image gradients $\overline{F}_x, \overline{F}_y, \hat{F}_2 - F_1$, find an update vector field (u, v) which minimizes the functional

$$\iint \left[\bar{F}_x u + \bar{F}_y v + (\hat{F}_2 - F_1)\right]^2 + \alpha^2 \left[|\nabla(U+u)|^2 + |\nabla(V+v)|^2\right] dxdy \qquad (24)$$

The first term in the functional is the square of the change of image value, as it is defined by Equation 22. The second term is a roughness measure of $U + u$ and $V + v$, where ∇ is the gradient operator and α a relative weighting factor. Note that it is the disparity field $(U + u, V + v)$ which is required to be smooth and not just the update field (u, v). This is very important in the implementation algorithm (Section 4), which truncates the disparity vector as it is passed down a level in the cone.

The equivalent PDE system (Euler's equations, [Courant & Hilbert 53, Section IV.3.4]) is

$$\alpha^2\Delta(U+u) - \overline{F}_x^2 u - \overline{F}_x\overline{F}_y v = \overline{F}_x(\hat{F}_2 - F_1) \quad (25a)$$

$$\alpha^2\Delta(V+v) - \bar{F}_x\bar{F}_y u - F_y^2 v = F_y(\hat{F}_2 - F_1) \quad (25b)$$

These equations involve first and second partial derivatives in the image data and the vector fields. This is purely local information, that is, at a given point the partial derivatives of a scalar or vector field are determined by their values in the immediate neighborhood of the point. Thus, the discrete solution to these equations presented in the next section is a local computation.

3.2 Discrete Representation and Computation

Update vectors will be found by solving Equation 13. The edge flow, given in Equation 11, will provide an initial estimate. To do this we must formulate a discrete representation of these equations. Finite difference approximations to the "continuous" derivative operators will be used. Equation 13 then becomes a system of linear equations in the variables $(u_{ij}, v_{ij}) \triangleq (u(i,j), v(i,j))$. These equations are solved using an iterative relaxation scheme which is local, uniform. and parallel — the criteria we require to be satisfied for efficient implementation in a cellular/processing-cone architecture.

3.2.1 Partial Derivative Operators

Equations 11 and 13 will be represented by discrete finite difference methods. We must approximate partial derivatives and differential operators by finite difference equations. These can then be represented as image filters as we now show.

For a continuous function $f(x,y)$, f_x is approximated by the first central difference $\delta_x^h f \triangleq \frac{1}{2h}[f(x+h,y) - f(x-h,y)]$. Let $\hat{f}$ be a discrete (sampled) representation of f with $\hat{f}(i,j) = f(ih,jh) = f(x,y)$. Then we can also consider δ_x^h as operating on $\hat{f}$: $\delta_x\hat{f} \triangleq \frac{1}{2h}[\hat{f}(i+1,j) - \hat{f}(i-1,j)]$. Thus, the operator δ_x^h can be represented

as the image filter $\frac{1}{2h}[-1\ 0\ 1]$. We can similarly define $\delta_y = \frac{1}{2h}[-1\ 0\ 1]^T$. These operators approximate $\partial/\partial x$ and $\partial/\partial y$ respectively when the grid spacing equals h. If we "normalize" the grid coordinate system by setting $h = 1$ (as defined in the next section), we get the **normalized first difference** operators $\delta_x \triangleq \frac{1}{2}[-1\ 0\ 1]$ and $\delta_y \triangleq \frac{1}{2}[-1\ 0\ 1]^T$.

In a practical imaging situation, images are likely to be corrupted by noise. This is especially so at high frequencies — low levels in the cone — where image content is low in amplitude and hence the signal-to-noise ratio is also low. Edge detection operators based on differentiating filters are high-pass filters and hence will give false measurements in the presence of noise. Low-pass filtering, to attenuate such noise, reduces this problem at the cost of smoothing out edges. A compromise can be had by smoothing only in the direction perpendicular to the direction of differencing (differentiation). We do this with the following first difference operators:

$$\delta_x \triangleq \frac{1}{2}[-1\ 0\ 1] * \frac{1}{4}[1\ 2\ 1]^T = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\delta_y \triangleq \frac{1}{4}[1\ 2\ 1] * \frac{1}{2}[-1\ 0\ 1]^T = \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Note that, if we ignore the 1/8 scaling factor, these operators are the Sobel edge detectors used early on in machine vision.

Using these definitions, Equation 23 is approximated by

$$\begin{aligned}(u_0, v_0) &= \left(\frac{-\frac{1}{h}\delta_x\overline{F}(\hat{F}_2 - F_1)}{(\frac{1}{h}\delta_x\overline{F})^2 + (\frac{1}{h}\delta_y\overline{F})^2}, \frac{-\frac{1}{h}\delta_y\overline{F}(\hat{F}_2 - F_1)}{(\frac{1}{h}\delta_x\overline{F})^2 + (\frac{1}{h}\delta_y\overline{F})^2} \right) \\ &= \left(\frac{-h\delta_x F(\hat{F}_2 - F_1)}{\delta_x\overline{F}^2 + \delta_y\overline{F}^2}, \frac{-h\delta_y F(\hat{F}_2 - F_1)}{\delta_x\overline{F}^2 + \delta_y\overline{F}^2} \right) \end{aligned} \tag{26}$$

Equation 25 is approximated by

$$\alpha^2 \frac{1}{h^2}\Delta_1(U + u) - (\frac{1}{h}\delta_x\overline{F})^2 u - (\frac{1}{h}\delta_x\overline{F})(\frac{1}{h}\delta_y\overline{F})v = \frac{1}{h}\delta_x\overline{F}(\hat{F}_2 - F_1) \tag{27a}$$

$$\alpha^2 \frac{1}{h^2}\Delta_1(V+v) - (\frac{1}{h}\delta_x F)(\frac{1}{h}\delta_y F)u - (\frac{1}{h}\delta_y F)^2 v = \frac{1}{h}\delta_y F(\hat{F}_2 - F_1) \quad (27b)$$

and multiplying through by h^2 we get

$$\alpha^2\Delta_1(U+u) - (\delta_x\overline{F})^2 u - (\delta_x\overline{F})(\delta_y\overline{F})v = h\delta_x\overline{F}(\hat{F}_2 - F_1) \quad (28a)$$

$$\alpha^2\Delta_1(V+v) - (\delta_x\overline{F})(\delta_y F)u - (\delta_y F)^2 v = h\delta_y F(\hat{F}_2 - F_1) \quad (28b)$$

3.2.2 Normalized Coordinate Systems

The **base coordinate system** is the coordinate system in which the inter-pixel spacing (grid spacing) is 1 unit at the **base level** — the level L of the input images. Using this coordinate system, at level k the grid spacing is $h_k = 2^{L-k}$. Equations 26 and 28 involve the grid spacing h. At the lowest level, $h = 1$ and it can be removed from the equations. The h factor can also be removed from consideration at all other levels of the pyramid if we use a normalized coordinate system at each level. In the **normalized coordinate system** at a given level, the distance between two adjacent pixels is 1 unit. Image operators acting at a single level can be computed without using h_k. However, intra-level computations and comparisons must then take into account the difference in the grid spacings as given by the ratio $r \triangleq h_{k-1}/h_k$.

Consider for example the edge flow equations (Equation 26. If (u_0, v_0) is the edge flow in base coordinates and $(\tilde{u}_0, \tilde{v}_0)$ is the edge flow in normalized coordinates then $h(\tilde{u}_0, \tilde{v}_0) = (u_0, v_0)$. Substitution into equation 26 gives

$$(\tilde{u}_0, \tilde{v}_0) = \begin{pmatrix} -\delta_x F(\hat{F}_2 - F_1) & -\delta_y F(\hat{F}_2 - F_1) \\ \delta_x\overline{F}^2 + \delta_y\overline{F}^2 & \delta_x\overline{F}^2 + \delta_y\overline{F}^2 \end{pmatrix} \quad (29)$$

Similarly the h factor can be removed from equation 28 when we specify that (U, V) and (u, v) are represented in the normalized coordinate system. We will assume from now on that this specification has been made and so we need not use the tilde ($\tilde{}$) notation.

3.2.3 Relaxation Equations

Equation 28 can be solved using the iterative point-Jacobi method. This involves the iterative application of the following update equations derived from equation 28:

$$u^{k+1} := \hat{u}^k + (\hat{U} - U) - \frac{\overline{F}_{\delta x}\left[\overline{F}_{\delta x}(\hat{u}^k + \hat{U} - U) + \overline{F}_{\delta y}(\hat{v}^k + \hat{V} - V) + (\hat{F}_2 - F_1)\right]}{(\alpha^2 + F_{\delta x}^2 + F_{\delta y}^2)} \tag{30a}$$

$$v^{k+1} := \hat{v}^k - (\hat{V} - V) - \frac{\overline{F}_{\delta y}\left[\overline{F}_{\delta x}(\hat{u}^k + \hat{U} - U) + \overline{F}_{\delta y}(\hat{v}^k + \hat{V} - V) + (\hat{F}_2 - F_1)\right]}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)} \tag{30b}$$

where $(\hat{U}, \hat{V}) = (\hat{U}(i,j), \hat{V}(i,j))$ are local averages of (U, V); (u^k, v^k) is the estimate at the kth iteration; $(\hat{u}^k, \hat{v}^k)$ are local averages of (u^k, v^k); and $\overline{F}_{\delta x} \triangleq \delta_x \overline{F}$ and $\overline{F}_{\delta y} \triangleq \delta_y \overline{F}$ are the normalized first difference operators.

3.3 Geometric Interpretation of Update Equations

The relaxation update equations (30) are better understood in terms of their geometric interpretation. They can be thought of as entailing two steps for each pixel: (1) compute the average disparity in the local neighborhood; (2) project this value towards the constraint line. The second step is accomplished by picking a disparity vector that lies on the line segment joining the average disparity and its projection onto the constraint line. The choice is made closer to the constraint line when we have higher confidence in the measurement of this line, namely when the spatial gradient at this pixel is high. The following paragraphs elaborate on these ideas.

First consider the case for which the approximate disparity $(U, V) = 0$. Equation 30 then simplifies to

$$u^{k+1} := \hat{u}^k - \frac{\overline{F}_{\delta x}\left[\overline{F}_{\delta x}\hat{u}^k + \overline{F}_{\delta y}\hat{v}^k + (F_2 - F_1)\right]}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)} \tag{31a}$$

$$v^{k+1} := \hat{v}^k - \frac{\overline{F}_{\delta y}\left[\overline{F}_{\delta x}\hat{u}^k + \overline{F}_{\delta y}\hat{v}^k + (F_2 - F_1)\right]}{(\alpha^2 + F_{\delta x}^2 + F_{\delta y}^2)} \tag{31b}$$

or rearranged into a vector notation

$$(u,v)^{k+1} = (\hat{u},\hat{v})^k - \left[\begin{array}{c}(\hat{u},\hat{v})^k \cdot (F_{\delta x}, F_{\delta y}) + (F_2 - F_1) \\ (\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)\end{array}\right] (\overline{F}_{\delta x}, \overline{F}_{\delta y}) \qquad (32)$$

Note that since $(U,V) = 0$, $\hat{F}_2 = F_2$.

The geometry of this simple relaxation updating equation in shown in Figure 35. The average disparity over the local neighborhood of an image point is shown as the point $(\hat{u},\hat{v})^k$ in (u,v) disparity space. The constraint line is shown with its equation $(u,v)\cdot(F_x,F_y) + (F_2 - F_1) = 0$. The (perpendicular) projection of $(\hat{u},\hat{v})^k$ onto the constraint line is shown as $(\hat{u},\hat{v})^k - (p,q)$ where (p,q) is the vector from that point to $(\hat{u},\hat{v})^k$. It is easy to verify that:

$$(p,q) = \frac{(\hat{u},\hat{v})^k \cdot (\overline{F}_x, \overline{F}_y) + (F_2 - F_1)}{\overline{F}_x^2 + \overline{F}_y^2} (\overline{F}_x, \overline{F}_y) \qquad (33)$$

Now suppose that we were to choose an update vector $(u,v)^{k+1}$ along the line segment joining $(\hat{u},\hat{v})^k$ and $(\hat{u},\hat{v})^k - (p,q)$ using a linear combination of these two vectors, that is

$$\begin{aligned}(u,v)^{k+1} &= (1-t)(\hat{u},\hat{v})^k + t[(\hat{u},\hat{v})^k - (p,q)] \\ &= (\hat{u},\hat{v})^k - t(p,q) \end{aligned} \qquad (34)$$

where $t \in [0,1]$ selects a position between the two end points. When Equation 33 is substituted into this equation, we can make the result equivalent to Equation 32 if we set

$$\begin{aligned} t &= \frac{F_x^2 + F_y^2}{\alpha^2 + F_x^2 + F_y^2} \\ &= \frac{\|(\overline{F}_x, \overline{F}_y)\|^2}{\alpha^2 + \|(F_x, F_y)\|^2} \\ &= \frac{1}{1 + \left(\frac{\alpha}{\|(\overline{F}_x, \overline{F}_y)\|}\right)^2} \end{aligned} \qquad (35)$$

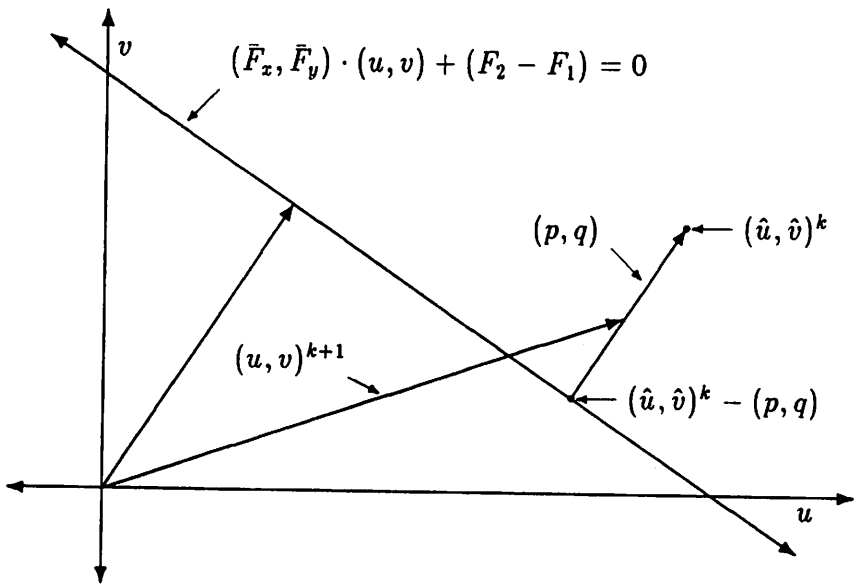


Figure 35: Geometry of simple relaxation updating

$(\hat{u}, \hat{v})^k$ is the average of neighboring pixel flow estimates at iteration k. $(\hat{u}, \hat{v})^k - (p, q)$ is the perpendicular projection of $(\hat{u}, \hat{v})^k$ onto the constraint line. This is the point on the constraint line nearest to $(\hat{u}, \hat{v})^k$. The new flow estimate $(u, v)^{k+1}$ lies between $(\hat{u}, \hat{v})^k$ and $(\hat{u}, \hat{v})^k - (p, q)$.

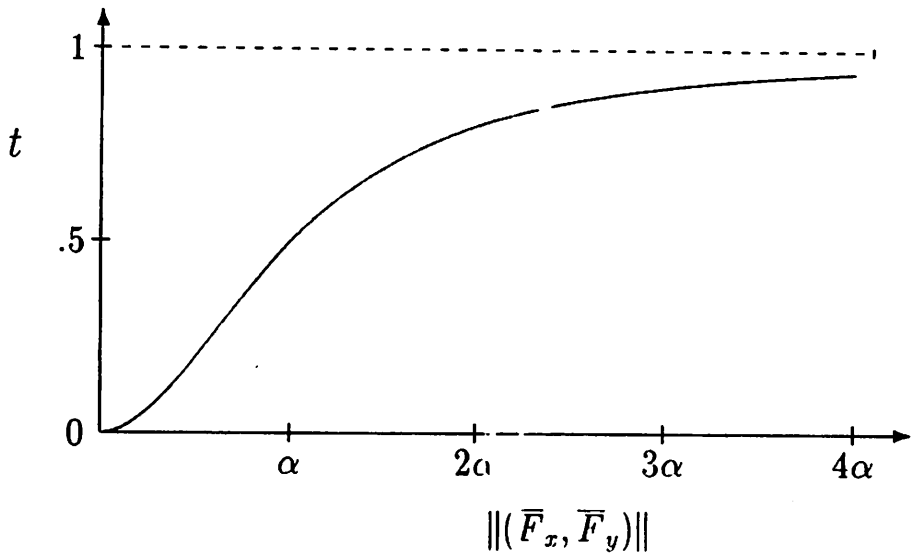


Figure 36: Update weighting factor

In Figure 36, t is graphed as a function of the magnitude of the spatial gradient $\|\overline{F}_x, \overline{F}_y\|$. We see that when the gradient is small t goes to zero and, looking at Equation 34, $(u,v)^{k+1}$ is chosen near $(\hat{u},\hat{v})^k$. On the other hand as the gradient gets very large t goes to 1.0 and $(u,v)^{k+1}$ is chosen near $(\hat{u},\hat{v})^k - (p,q)$. The midway point, giving equal weight to both possibilities, occurs when the magnitude of the gradient is equal to α. The parameter α controls the shape of this function. Larger values of α "stretch" the curve out to the right so that higher values of the spatial gradient are needed to pull the update vector towards the constraint line.

Now we generalize this picture to the case of hierarchical relaxation updating. Figure 37 shows the geometry of updating for a given location in the image. First note that two coordinate systems are shown in this diagram. The main coordinate system, labeled "I", is the (u,v) space with its origin at the point corresponding to zero disparity. This corresponds to the coordinate space shown in Figure 35. The approximate disparity (U,V) is shown with its tail at the origin of this space. The secondary coordinate system, labeled "II", is the (u,v) space with its origin at the approximate disparity. Subscripts of $_{\rm I}$ or $_{\rm II}$ are used to specify which space is

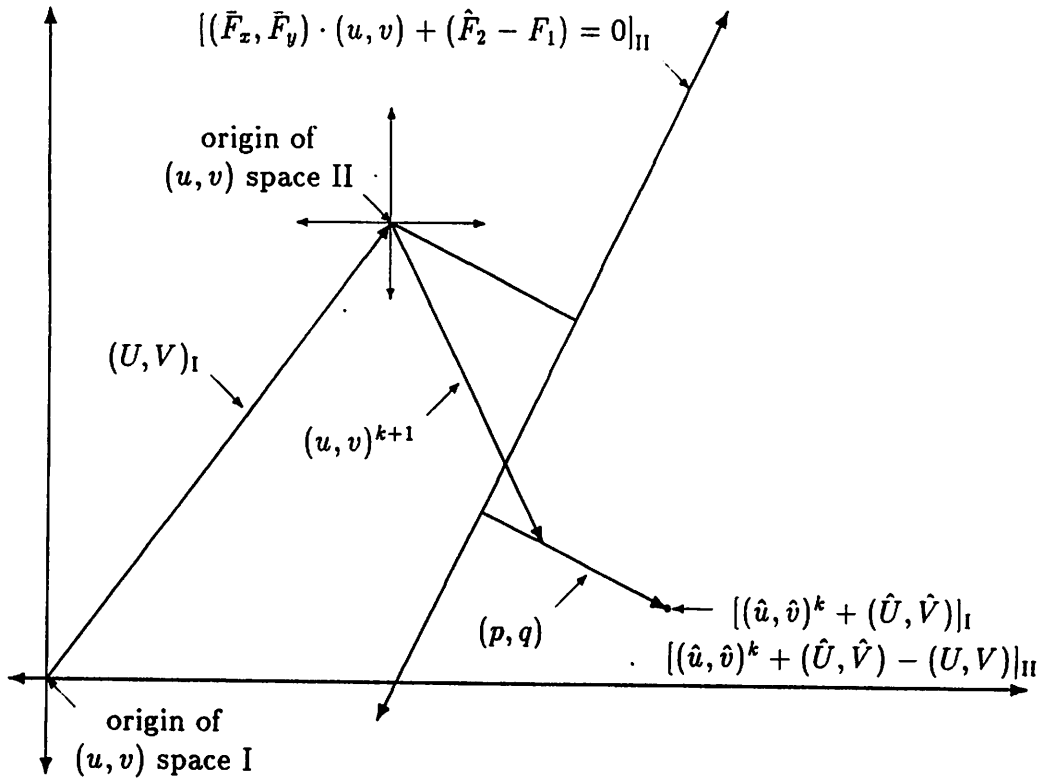


Figure 37: Geometry of hierarchical relaxation updating

being used for a given set of coordinates for a point. For example, the approximate disparity can be represented as $(U, V)_I$ or $(0, 0)_{II}$ and the updated disparity as $(U + u^{k+1}, V + v^{k+1})_I$ or $(u^{k+1}, v^{k+1})_{II}$. Because the coordinate spaces differ only by a translation, the coordinate representation of a vector (as opposed to a point) is independent of the choice of coordinate space.

The selection of an update vector again involves (1) computation of the average disparity in the local neighborhood, and (2) projection of this value towards the constraint line. The average disparity is the average of the approximate disparity vector summed with the update vector at each pixel in the local neighborhood. This is $(u^k \widehat{+ U}, v^k \widehat{+ V})_I = [(\hat{u}, \hat{v})^k + (\hat{U}, \hat{V})]_I$ or in the secondary coordinate system $[(\hat{u}, \hat{v})^k + (\hat{U}, \hat{V}) - (U, V)]_{II}$. The constraint equation is defined on update vectors (u, v) and hence it is defined in the secondary space. This is signified in the figure

by the subscript $_{\text{II}}$ given to the equation. Again (p, q) is the vector pointing to the average disparity from its (perpendicular) projection on the constraint line. In this case it is given by

$$(p,q) = \frac{[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)] \cdot (\overline{F}_x, \overline{F}_y) + (\hat{F}_2 - F_1)}{F_x^2 + F_y^2}(F_x, F_y) \qquad (36)$$

The generalization to the earlier non-hierarchical case is now straightforward. We choose an update vector $(u,v)^{k+1}$ along the line segment joining $[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)]_{\text{II}}$ and $[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)]_{\text{II}} - (p,q)$ using a linear combination of these two vectors, that is

$$\begin{aligned}(u,v)^{k+1} &= (1-t)|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)| \\ &\quad + t|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V) - (p,q)| \\ &= |(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)| - t(p,q) \end{aligned} \qquad (37)$$

where $t \in [0,1]$ selects a position between the two end points. When Equation 36 is substituted into this equation, we can make the result equivalent to Equation 30 if, as before, we set

$$t = \frac{\|(\overline{F}_x, \overline{F}_y)\|^2}{\alpha^2 + \|(\overline{F}_x, \overline{F}_y)\|^2} = \frac{1}{1 + \left(\frac{\alpha}{\|(\overline{F}_x, \overline{F}_y)\|}\right)^2} \qquad (38)$$

This is the same relationship shown in Figure 36, where t is graphed as a function of the magnitude of the spatial gradient $\|\overline{F}_x, \overline{F}_y\|$ with the parameter α controlling the shape of this function. When the gradient is small t goes to zero and $(u,v)^{k+1}$ is chosen near $[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)]_{\text{II}}$. On the other hand as the gradient gets very large t goes to 1.0 and $(u,v)^{k+1}$ is chosen near $[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)]_{\text{II}} - (p,q)$. The midway point, giving equal weight to both possibilities, occurs when the magnitude of the gradient is equal to α. Larger values of α "stretch" the curve out to the right so that higher values of the spatial gradient are needed to pull the update vector towards the constraint line.

4 Hierarchical Disparity Algorithms

4.1 Hierarchical Data Flow

Data flow graphs for the hierarchical disparity algorithm are shown in Figures 38–40. Figure 38 shows the coarse-to-fine flow of control and data, and is essentially equivalent to that used in the hierarchical correlation algorithm as shown in Figure 22. The hierarchical disparity computation is a coarse-to-fine algorithm operating on two low-pass image pyramids. The PROJECTION operator passes the updated disparity field to the next level of the processing cone. The REFINEMENT process computes an updated disparity field given an estimated disparity field and the image pyramid data at the corresponding resolution. A data flow diagram of REFINEMENT is shown in Figure 39. Update vectors are computed using an iterative relaxation process shown in Figure 40.

The hierarchical disparity algorithm is shown in pseudo-code in Figures 41–45. Figure 41 is the "top level" procedure. It builds the Gaussian pyramids over the input images, initializes data, and controls the hierarchical processing within the processing cone.

4.2 Projection

The Projection operator — Figure 42 — passes the coarse disparity vector of the father to its four sons with two changes. First, the components of the vector are multiplied by 2, the inter-grid spacing ratio. This converts between the two normalized coordinate systems. Second, the values are then truncated to the nearest integer. This simplifies the computation of gradients to follow.

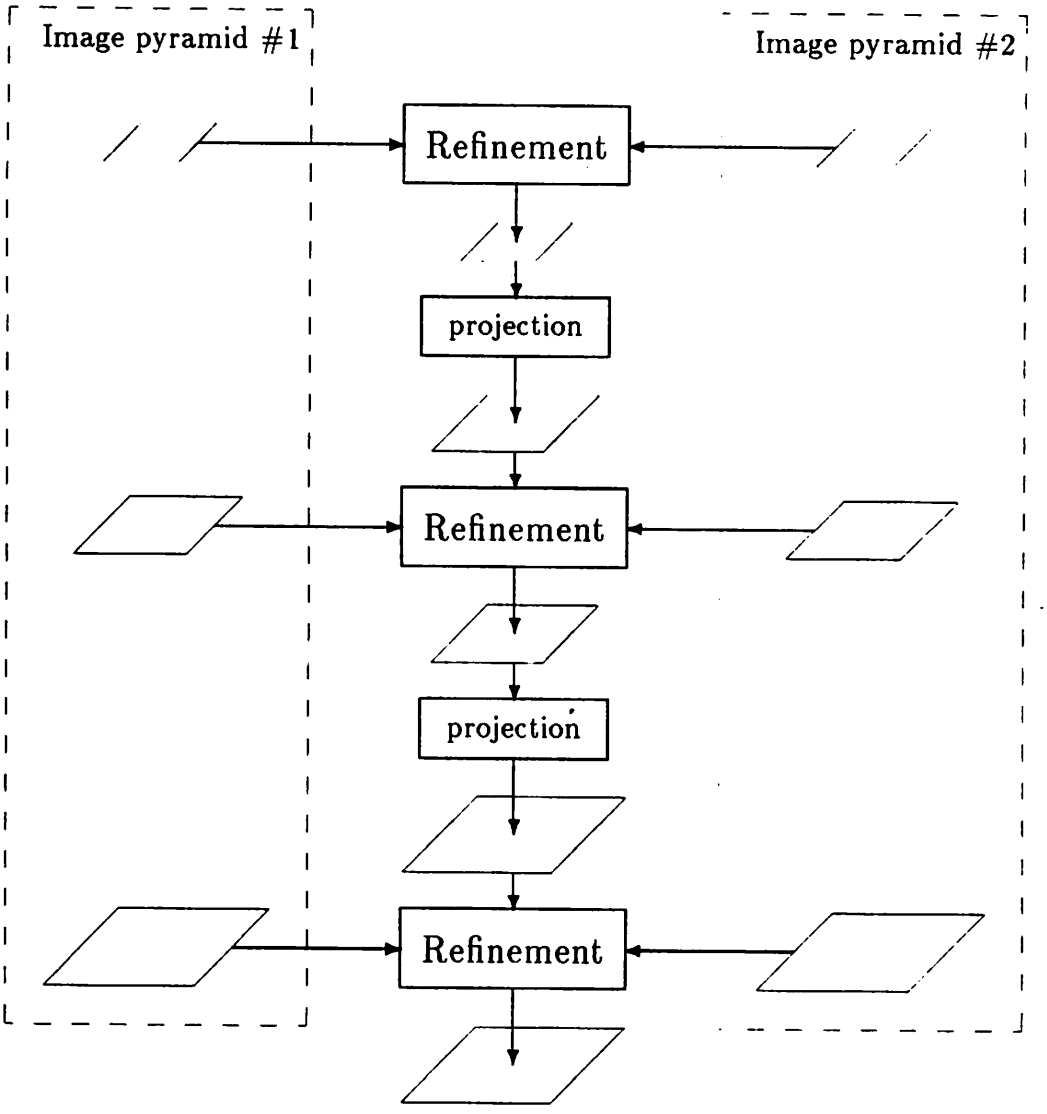


Figure 38: Hierarchical disparity: data flow

The overall hierarchical disparity computation is a coarse-to-fine algorithm operating on two low-pass image pyramids (shown in dotted outline). The REFINEMENT processes compute an updated disparity field given an estimated disparity field and the image pyramid data at the corresponding resolution. The PROJECTION operator passes the updated disparity field to the next level of the processing cone.

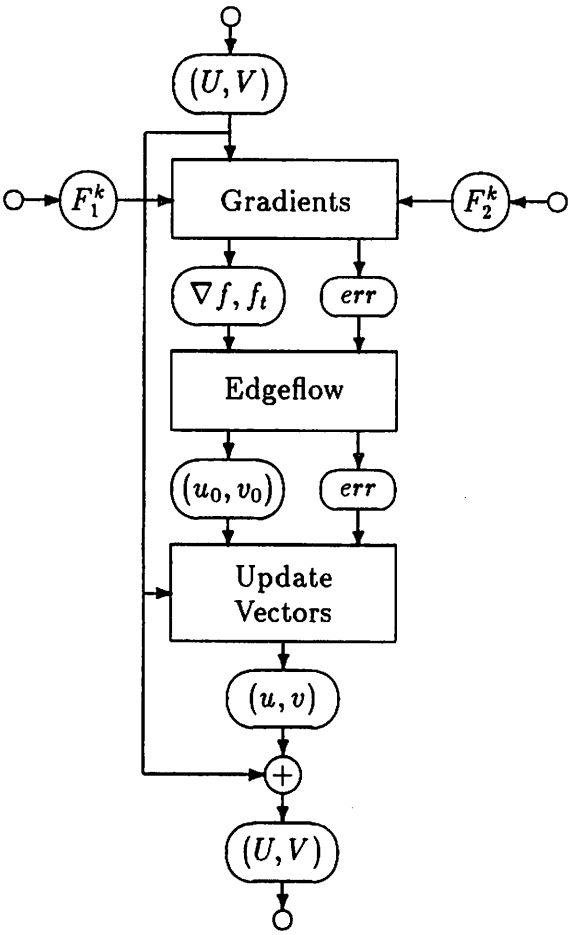


Figure 39: Hierarchical disparity: refinement

The REFINEMENT process performed at each level inputs an estimated disparity (U,V) and image pyramid data $\mathcal{F}_1^k$ and $\mathcal{F}_2^k$. Single-level (non-hierarchical) image operators compute (1) the spatial gradients and image differences $\nabla f. f_t$; (2) the edge flow at each pixel (u_0, v_0); and (3) an update vector (u, v). The update vector is added to the estimated disparity to give the updated (refined) disparity. An array of error flags (*err*), one per pixel, is carried along through the operations at one level. The operators shown may generate error flags or respond to previously generated ones.

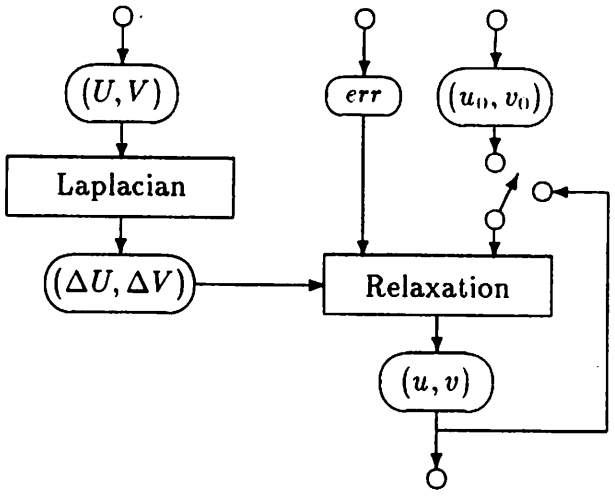


Figure 40: Hierarchical disparity: relaxation

The update vector is computed by iterative application of the relaxation operator specified in Equation 18 (when the error flag is set to NO_ERROR). The edge flow (u_0, v_0) is used as the first estimate of the update vector input to RELAXATION. Thereafter, the current values of (u, v) are used. The LAPLACIAN of (U, V) is only computed once.

4.3 Gradients

The gradients that must be computed are $\delta_x F$, $\delta_y F$, and $\hat{F}_2 - F_1$ (see Equations 17-18), where $\hat{F}_2(\mathbf{x}) \triangleq \hat{F}(\mathbf{x} + \mathbf{U})$ and $\overline{F} \triangleq \frac{1}{2}(F_1 + \overline{F}_2)$. Since δ_x and δ_y are linear operators, $\delta_x \hat{F} = \frac{1}{2}(\delta_x F_1 + \delta_x \hat{F}_2)$ and similarly for $\delta_y \hat{F}$. When refinement is being performed at a pixel (i, j), $\delta_x F_1$ is a first difference of F_1 computed at (i, j) and $\delta_x \hat{F}_2$ is a first difference of F_2 computed at $(i + U, j + V)$. If the approximate disparity (U, V) is real-valued, then we must compute a first difference at a "non-integral" point in F_2. That is, we would require an approximation to the first partial derivative at a point which did not lie on the discrete image grid. This is certainly possible and it can be performed by a weighted sum of values of the neighboring pixels. Such a sum essentially combines first differencing and interpolation. However, it complicates the computation because (1) the weights will vary depending on the interpixel position of the desired measurement and so they must be computed separately at

Input:	F_1, F_2	Images at level L
Parameters:	N	Relaxation iterations (per level)
	α	Noise-level smoothing parameter
	l_{top}	First (highest) computation level
Output:	$(U,V)^k, k = l_{top}, \ldots, L$	Disparity pyramid

$\mathcal{F}_1 \leftarrow \mathcal{G}(F_1)$, ; Gaussian pyramid of frame 1
$\mathcal{F}_2 \leftarrow \mathcal{G}(F_2)$, ; Gaussian pyramid of frame 2
$k \leftarrow l_{top} - 1$, ; start at finest level
$(U,V)^{l_{top}-1} \leftarrow (0,0)$, ; initial coarsest disparity estimate
Until (k is the finest level) **Do** {
  $k \leftarrow k + 1$,
  $(U,V)^k \leftarrow \text{Project}(U,V)^{k-1}$, ; project disparity estimate
  $err \leftarrow$ NO_ERROR at all pixels, ; reset error flags
  $(f_x, f_y, f_t, err) \leftarrow \text{Gradients}(\mathcal{F}_1^k, \mathcal{F}_2^k, (U,V)^k, err)$,
  $(u_0, v_0), err \leftarrow \text{Edgeflow}(f_x, f_y, f_t, err)$,
  $(u,v) = (u_0, v_0)$, ; initial estimate
  Do (N times)
    $(u,v) \leftarrow \text{Relax}\big((u,v), (\Delta U, \Delta V)^k, f_x, f_y, f_t, err; \alpha\big)$,
  $(U,V)^k \leftarrow (U,V)^k + (u,v)$; update the disparity field
  }

Figure 41: Hierarchical disparity algorithm

$(U,V)^k \leftarrow \text{Project}((U,V)^{k-1})$

For all (pixels (i,j)) **Do**
  $(U,V)^k \leftarrow$ nearest_integer($2 \times (U,V)^{k-1}$ of father)

Figure 42: Hierarchical disparity: Projection

each pixel; and (2) the weights will no longer be the simple values like ± 1 or ± 2 but rather more general ones, e.g. real numbers, calling for more general (and hence slower) arithmetic.

To avoid these difficulties, we truncate U and V to the nearest integer. (This operation is included above as part of the projection operator since it need only be done once per father pixel.) This allows us to use the same simple 3×3 operators in F_2 at $(i+U, j+V)$ as are used in F_1 at (i,j). The truncation error is $\pm 1/2$ pixel (normalized coordinates) and thus falls within the range of disparities which the subsequent update computation process can handle.

Had this nearest pixel approximation not proved adequate, a compromise between it and full resolution could be made by truncating to the nearest $1/2$ pixel. That is, U and V would be truncated to the nearest integer or integer plus one half. Such a truncated (U,V) would either land on a grid point, between two horizontally or vertically neighboring grid points, or in the center of a square of four neighboring grid points. For each of these four cases, there corresponds a set of masks for the gradient computations. The weights are then only slightly larger and more unwieldy than the nearest pixel approximation.

The first difference operators δ_x and δ_y were specified above in Section 3.2.1 and include some smoothing. Smoothing is also introduced into the computation of $\hat{F}_2 - F_1$ using the filter

$$G_2 \triangleq \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Thus, the three gradients are computed using three pairs of 3×3 neighborhood operators, one of the pair about a point $f_1(i,j)$ and the other about $f_2(i+U, j+V)$.

These three gradients cannot be computed if either of the 3×3 neighborhoods does not lie completely within its respective image. This is the first error condition that may occur, called **search area overflow**. At pixels where this occurs, the gradients are set to $(0,0,0)$.

$(f_x, f_y, f_t, err) \leftarrow \text{Gradients}\big(f_1, f_2, (U,V), err\big)$

For all (pixels (i,j)) **Do** {
  If ($err \neq$ NO ERROR) **Then**
    $f_x, f_y, f_t \leftarrow (0,0,0)$
  Else if (search area overflows) **Then** {
    $f_x, f_y, f_t \leftarrow (0,0,0)$,
    $err \leftarrow$ SA_OVERFLOW }
  Else {
    $f_x \leftarrow \frac{1}{2}\big[\delta_x \cdot \text{Nbrhd}(f_1; i,j) + \delta_x \cdot \text{Nbrhd}(f_2; i+U, j+V)\big]$,
    $f_y \leftarrow \frac{1}{2}\big[\delta_y \cdot \text{Nbrhd}(f_1; i,j) + \delta_y \cdot \text{Nbrhd}(f_2; i+U, j+V)\big]$,
    $f_t \leftarrow G_2 \cdot \text{Nbrhd}(f_2; i+U, j+V) - G_2 \cdot \text{Nbrhd}(f_1; i,j)$ }
}

Figure 43: Hierarchical disparity: Gradients

Nbrhd$(f; i, j)$ is the neighborhood of pixel values in image f about the central pixel (i, j). δ_x, δ_y, and G_2 are 3×3 weighted sums as specified in the text. The symbol SA_OVERFLOW indicates search area overflow.

4.4 Edge Flow

The edge flow — Figure 44 — is computed as specified by Equation 29. Three error conditions may be encountered at any given pixel. First, if search area overflow has occurred then there are no gradient values available. Second, the spatial gradient may either be zero or too small to give an accurate answer when used as a divisor. The third error condition occurs when the edge flow vector is too large. The derivation of this constraint is presented in the next paragraph. For all three error conditions we set $(u_0, v_0) = (0,0)$.

The third error condition is a consistency check on the computed edge flow. In Section 3.1.2 we noted that because (1) the edge flow must be smaller (in magnitude) than the update vector, and (2) availability of an approximate disparity implies a bound on the size of the update vector, for these reasons an upper bound is placed on the size of the edge flow vector. We can use either of the following relationships:

$$\|(u_0, v_0)\|_2 \leq \|(u, v)\|_2 \tag{39}$$

$$\|(u_0, v_0)\|_2 \leq \sqrt{2}\,\|(u, v)\|_{Max} \tag{40}$$

where $\|(x,y)\|_2 \triangleq (x^2 + y^2)^{1/2}$ is the Euclidean norm and $\|(x,y)\|_{Max} \triangleq \max(|x|,|y|)$ is the Maximum norm. The second inequality follows from the first and from the fact that $\|(u,v)\|_2 \leq \sqrt{2}\,\|(u,v)\|_{Max}$ (since $0 \leq u^2 + v^2 \leq 2[\max(|x|,|y|)]^2$). The second inequality is used when we have bounds on the individual components of the update vector. For example, the truncation of the approximate disparity vectors to the nearest pixel upon projection introduces an error of $\pm\frac{1}{2}$ pixel in each of its components. Were this the only error in that disparity estimate, then the bound on the size of the update vector would be $\|(u,v)\|_{Max} \leq \frac{1}{2}$ and using Equation 40 we get the constraint $\|(u_0, v_0)\| \leq \frac{\sqrt{2}}{2}$. Thus, we have established a maximum allowable value EF_M of the magnitude of the edge flow $\|(u_0, v_0)\|_2$.

$(u_0, v_0), err \leftarrow$ Edgeflow(f_x, f_y, f_t, err)

For all (pixels (i,j)) **Do** {
  If ($err \neq$ NO_ERROR) **Then**
    $(u_0, v_0) \leftarrow (0,0)$
  Else if ($f_x^2 + f_y^2 < \delta$) **Then** {
    $(u_0, v_0) \leftarrow (0,0)$,
    $err \leftarrow$ ZERO GRAD }
  Else if $\frac{f_t^2}{f_x^2+f_y^2} > EF_M^2$ **Then** {
    $(u_0, v_0) \leftarrow (0,0)$,
    $err \leftarrow$ EDGE_FLOW_TOO_BIG }
  Else $(u_0, v_0) \leftarrow \frac{-f_t}{f_x^2+f_y^2}(f_x, f_y)$
}

Figure 44: Hierarchical disparity: Edge flow

EF_M is the maximum allowed magnitude of the edge flow. $f_t^2/(f_x^2 + f_y^2)$ is the square of the magnitude of the edge flow.

4.5 Relaxation

Update vectors are computed by an iterative relaxation process specified by Equation 30 (Figure 45). If any of the possible error conditions exist at a pixel, then no adequate constraint line exists at that pixel. Thus, the update equations cannot be used as is. However, as we noted in Section 3.3, the update equation can be interpreted as specifying that we (1) compute an update vector based on the average of neighboring information, followed by (2) projection of this vector towards the constraint line, then we see that step 1 can still be performed. Thus, we arrive at the update algorithm shown in Figure 45 in which at each pixel one of two things takes place: either (1) update by Equation 30 at non-error points, or (2) update by simple smoothing at points with errors. The latter is a form of interpolation into points (or areas) with no motion-based constraints. If there are large blocks where updating by (1) cannot take place, then vectors in those regions are determined from the constrained vectors which surround the respective regions.

$(u,v) \leftarrow \text{Relax}\Big((u,v),(\Delta U,\Delta V),f_x,f_y,f_t,err;\alpha\Big)$

For all (pixels (i,j)) **Do** {
$(\hat{u},\hat{v})_{ij} \leftarrow$ local mean of (u,v) at (i,j),
If $(err(i,j) =$ NO ERROR$)$ **Then**

$$(u,v)_{ij} \leftarrow (\hat{u},\hat{v})_{ij} + (\Delta U,\Delta V)_{ij} - (f_x,f_y)_{ij} \times \left[\frac{(f_x,f_y)\cdot\Big((\hat{u},\hat{v})_{ij}+(\Delta U,\Delta V)\Big)}{\kappa\alpha^2+f_x^2+f_y^2}\right]_{ij},$$

Else
$(u,v)_{ij} \leftarrow (\hat{u},\hat{v})_{ij} + (\Delta U,\Delta V)_{ij}$,
End if
}

Figure 45: Hierarchical Disparity: Relaxation

This operator performs one iteration of the relaxation process.

The average update vectors $(\hat{u},\hat{v})$ are computed using the following local averaging filter.

$$\frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The discrete Laplacian used to compute $(\Delta U,\Delta V)$ is

$$Lap_2 \triangleq \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & -3 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

5 Experiments

5.1 The Failure of Single-Level Methods

As we noted in the introduction to this chapter, gradient-based disparity computations will fail when the disparities are large relative to the range over which the image "function" is approximately linear. In general, this is a relatively short distance. Such failure takes the form of incorrect disparity estimates based on a model of the local image neighborhood that provides a bad approximation.

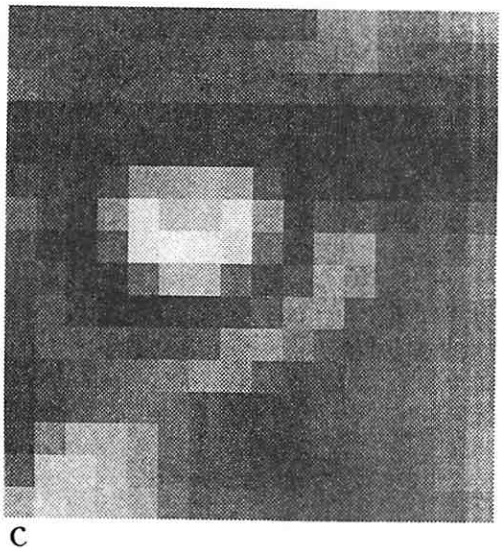
The following series of experiments show the inability of single-level gradient-based methods to compute disparity fields. Four experiments are performed with disparities which start at a low magnitude and increase by a factor of two for each subsequent experiment. We expect the disparity computations with single gradient based methods to become more errorful as the disparities increase in size. This is in fact what the experiments show.

Two 128×128 pieces of the MANDRILL image are used with a disparity of (-5,7) from frame 1 to frame 2, i.e. 5 pixels up and 7 to the right. Low-pass pyramids were built from these images and four separate runs of single-level disparity computation were performed at levels 4 (16^2) through 7 (128^2). Thus, the disparities at each level were $(-.625, .875)$, $(-1.25, 1.75)$, $(-2.5, 3.5)$, and $(-5, 7)$ respectively. The EDGE_FLOW_TOO_BIG error condition was included with maximum allowable edge flow magnitudes of 1.5, 3, 6, and 12 respectively. When this error condition is not included, results are worse.

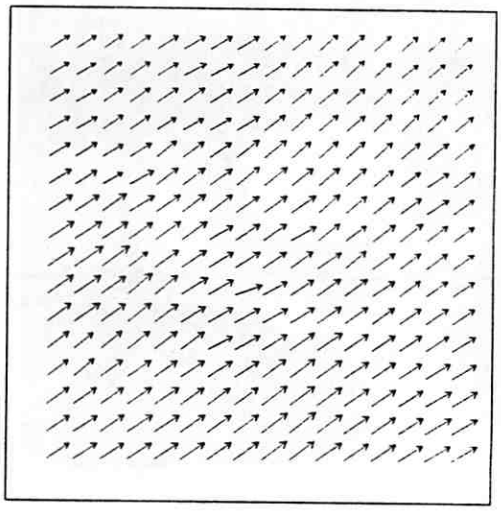
The results are shown in Figures 46–49. In each figure we show (a) the first frame of image data; (b) disparity vectors after 50 iterations; (c) error flags; and (d) statistics of the computed disparities compared to the expected values. The light gray pixels on the border of the error flags display indicate the SA OVERFLOW error condition. The predominant dark pixels throughout the interior of the error flags display indicate the EDGE_FLOW_TOO_BIG condition.

It is clear that as the disparity is increased relative to the pixel spacing, the

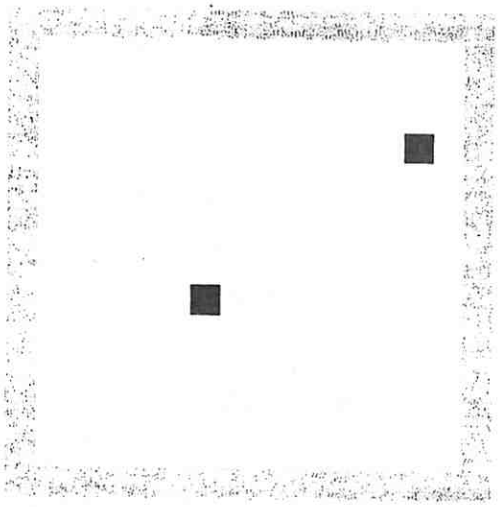
a



b



c



d

Actual	D_{row}	D_{col}
mean	−.5792	.8204
SD	.0655	.0953
minimum	−.7489	.5848
maximum	−.3642	1.037

Expected	−.625	.875

Figure 46: Single-level analysis: level 4

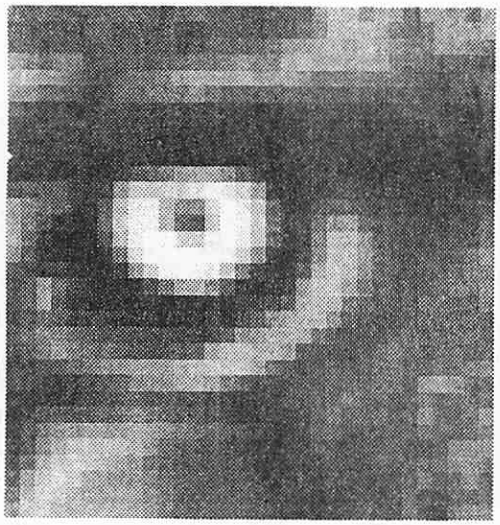
(a) the first frame of image data, 16 × 16

(b) disparity vectors after 50 iterations, shown for each pixel

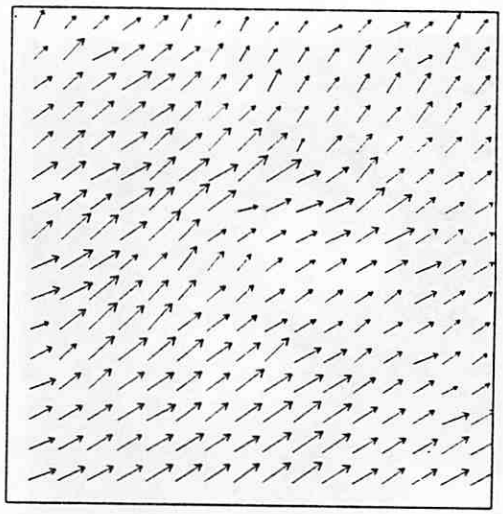
(c) error flags

(d) statistics of the disparities

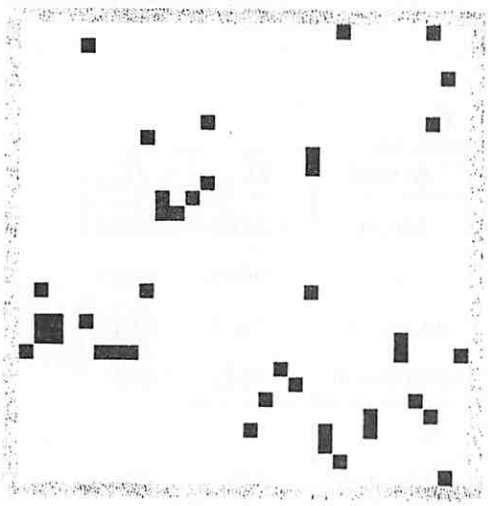
a



b



c



d

Actual	D_{row}	D_{col}
mean	−1.112	1.464
SD	.284	.407
minimum	−2.321	.471
maximum	−0.317	3.184
Expected	−1.25	1.75

Figure 47: Single-level analysis: level 5

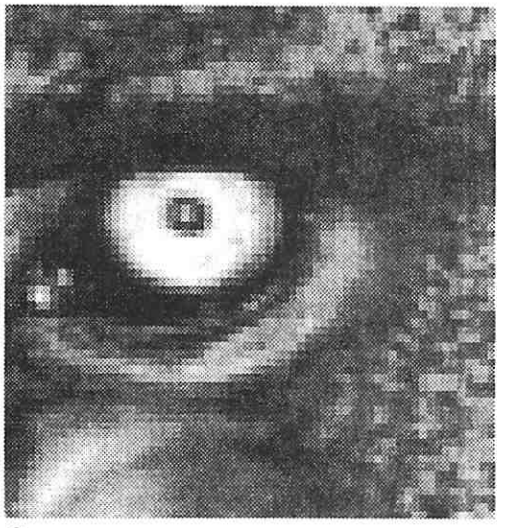
(a) the first frame of image data, 32×32

(b) disparity vectors after 50 iterations, shown for every 2nd pixel in each direction

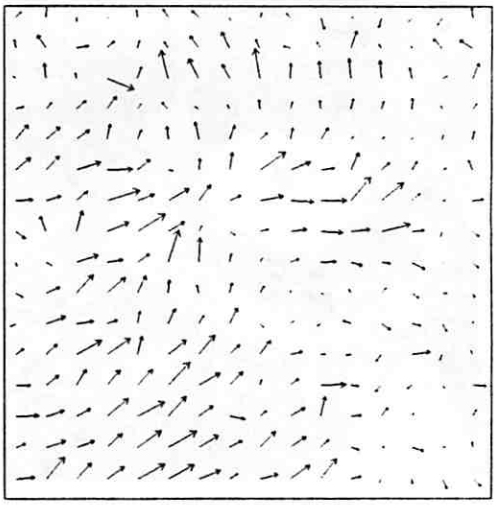
(c) error flags

(d) statistics of the disparities

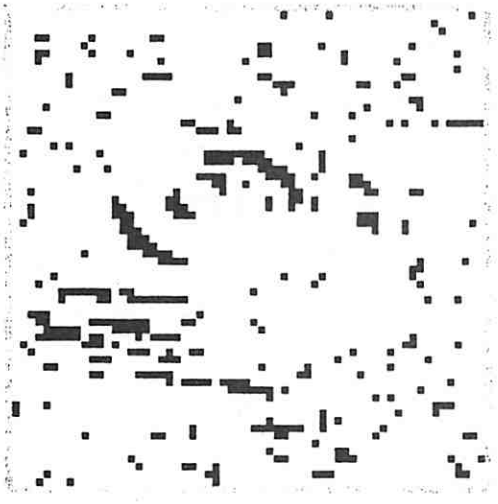
a



b



c



d

Actual	D_{row}	D_{col}
mean	−0.9244	1.007
SD	1.066	1.262
minimum	−5.362	−3.925
maximum	4.538	5.190
Expected	−2.5	3.5

Figure 48: Single-level analysis: level 6

(a) the first frame of image data, 64 × 64

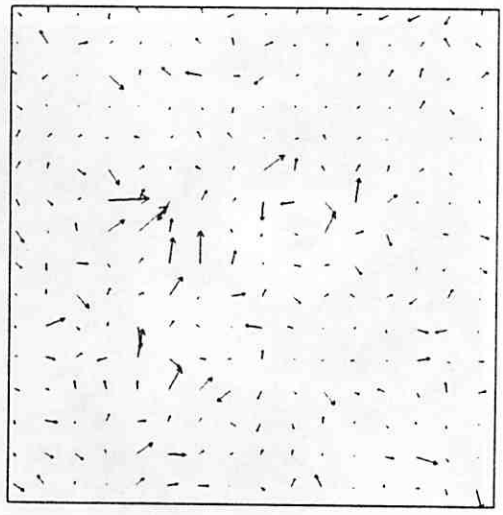
(b) disparity vectors after 50 iterations, shown for every 4th pixel in each direction

(c) error flags

(d) statistics of the disparities

a

b



c

d

Actual	D_{row}	D_{col}
mean	−0.474	0.503
SD	1.737	1.693
minimum	−11.03	−7.417
maximum	−7.282	10.55
Expected	−5.	7.

Figure 49: Single-level analysis: level 7

(a) the first frame of image data, 128×128

(b) disparity vectors after 50 iterations, shown for every 8th pixel in each direction

(c) error flags

(d) statistics of the disparities

performance of single-level gradient-based methods deteriorates quickly. At level 4, with expected disparity equal to $(-.625, .875)$, results are good. At level 5, with expected disparity equal to $(-1.25, 1.75)$, accuracy is significantly diminished. At level 6, with expected disparity equal to $(-2.5, 3.5)$, computed disparities are largely incorrect. At level 7, with expected disparity equal to $(-5, 7)$, computed disparities cluster near $(0, 0)$ and show little relation to the expected disparity.

5.2 Hierarchical Method

The hierarchical gradient-based disparity algorithm is demonstrated in the following experiment. Again 128^2 pieces of the MANDRILL image are used with a disparity of $(-5, 7)$. A low-pass pyramid was built for each input image. Processing was begun at level 4.

In Figure 50 disparity vectors at different stages of the computation are shown. The edge flow at level 4 is shown in Figure 50a. It is labeled as iteration 0 since it is used as the initial approximation. The update vectors after 10 iterations are shown in Figure 50b. Up to this point, this experiment is equivalent to the first single-level experiment of the last section. If relaxation continues at this level to the 50th iteration, the update vectors are then the same as those shown in Figure 46b.

At level 5, Figure 50c shows both the approximate disparity passed down from level 4 and the initial (iteration 0) update vector — the edge flow. Figure 50d shows the initial disparity and the final (iteration 10) update vector. At each pixel shown (every 2nd row and every 2nd column) the approximate disparity is attached by its tail to the pixel in frame 1 that it corresponds to. The tail of the update vectors are attached to the head of the approximate disparity vectors and have an arrowhead at their head. The arrowheads are proportional in size to the update vector and do not show up when that vector is small. The sum of the approximate and the update vectors is the updated disparity estimate, shown in Figure 50e.

Figures 50f–h show the comparable vectors at level 6. The final results at level 7 are shown in Figure 50i.

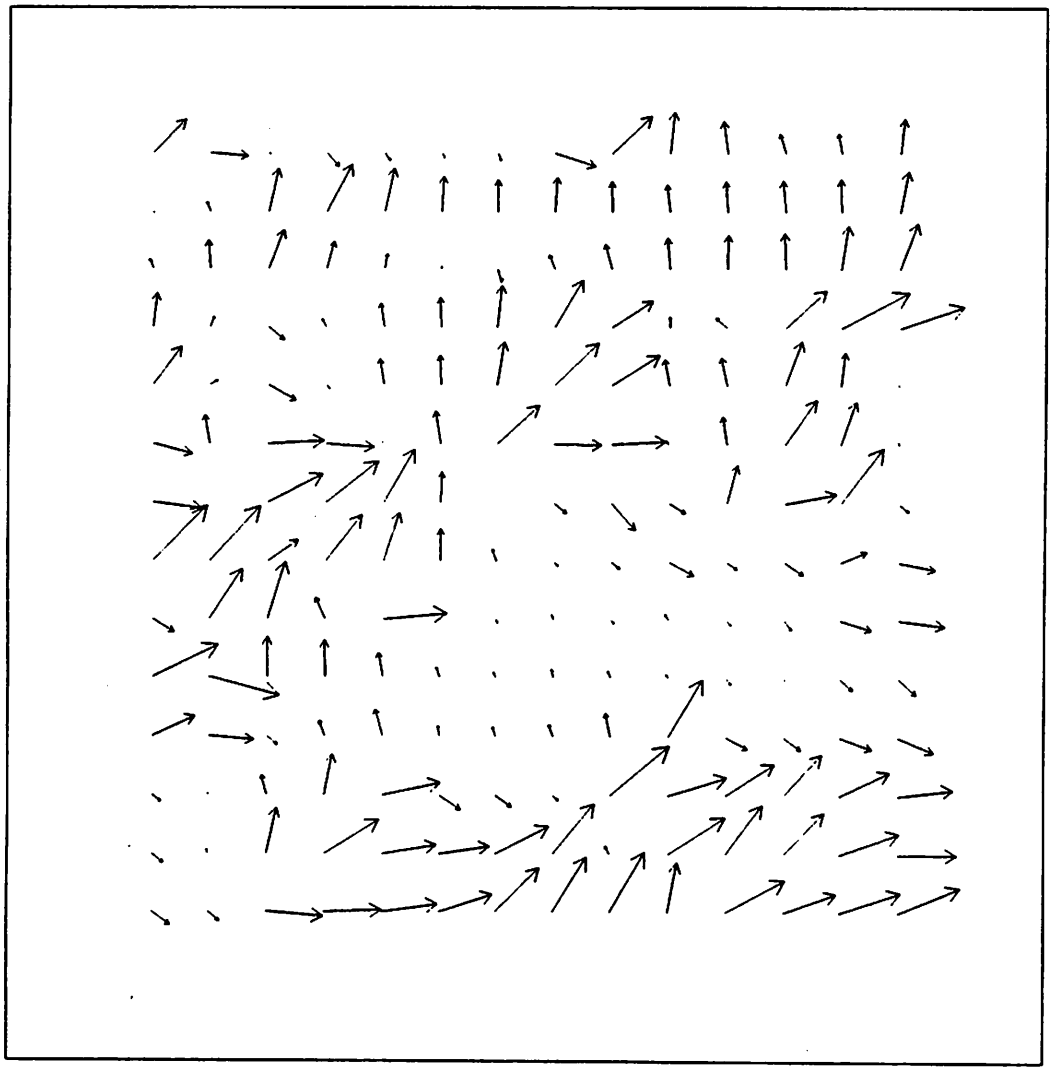


Figure 50: Multilevel experiment: disparity vectors

(a) level 4, iteration 0

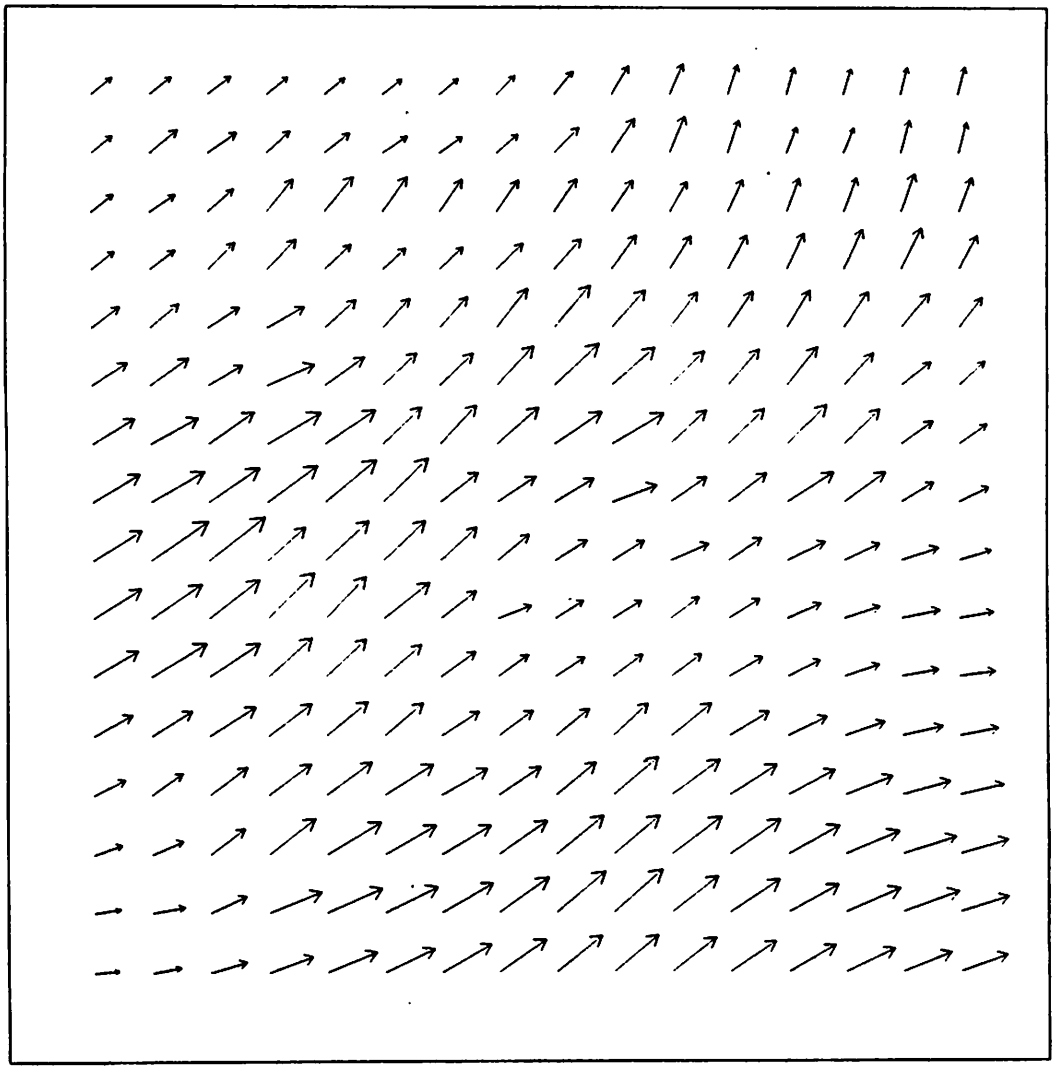


Figure 50 continued

(b) level 4, iteration 10

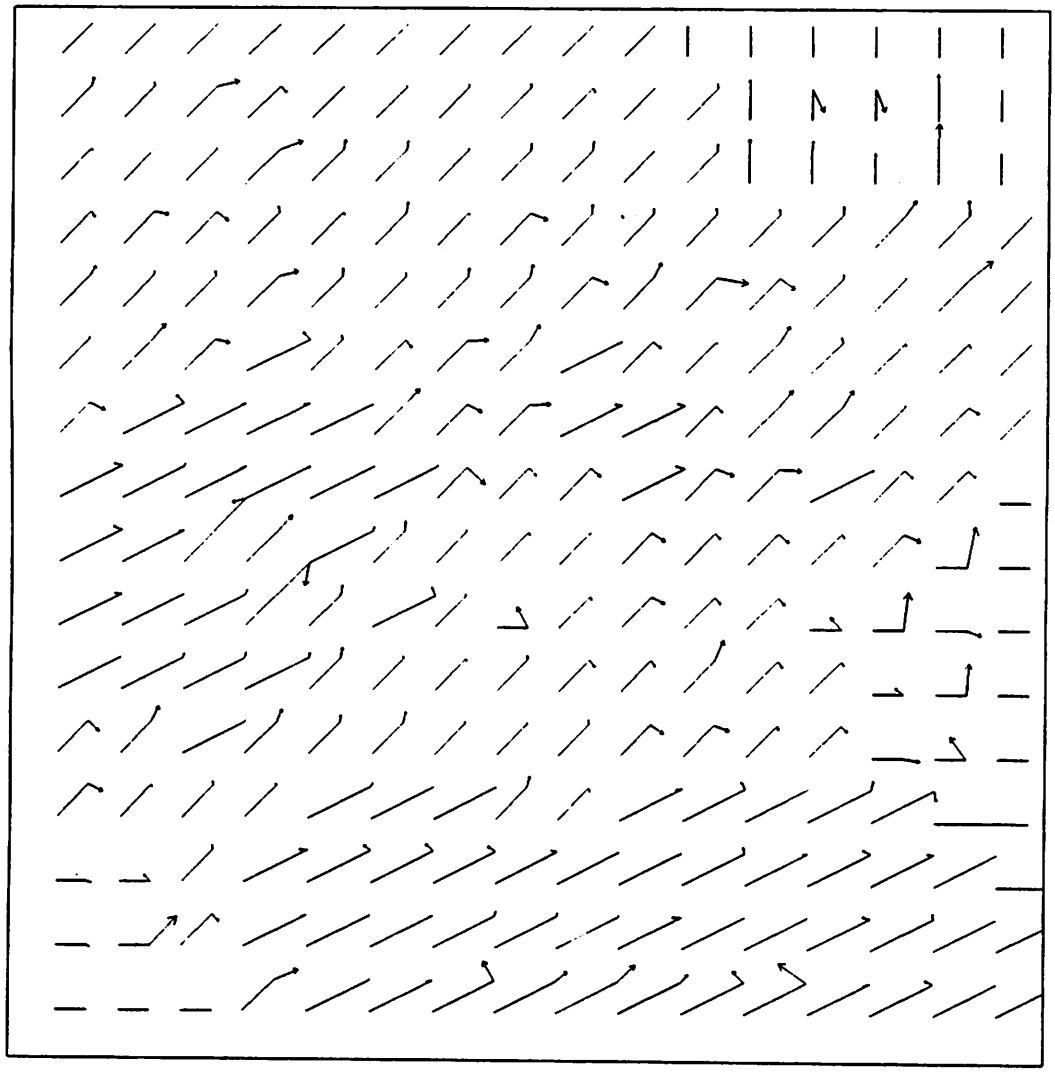


Figure 50 continued

(c) level 5, iteration 0

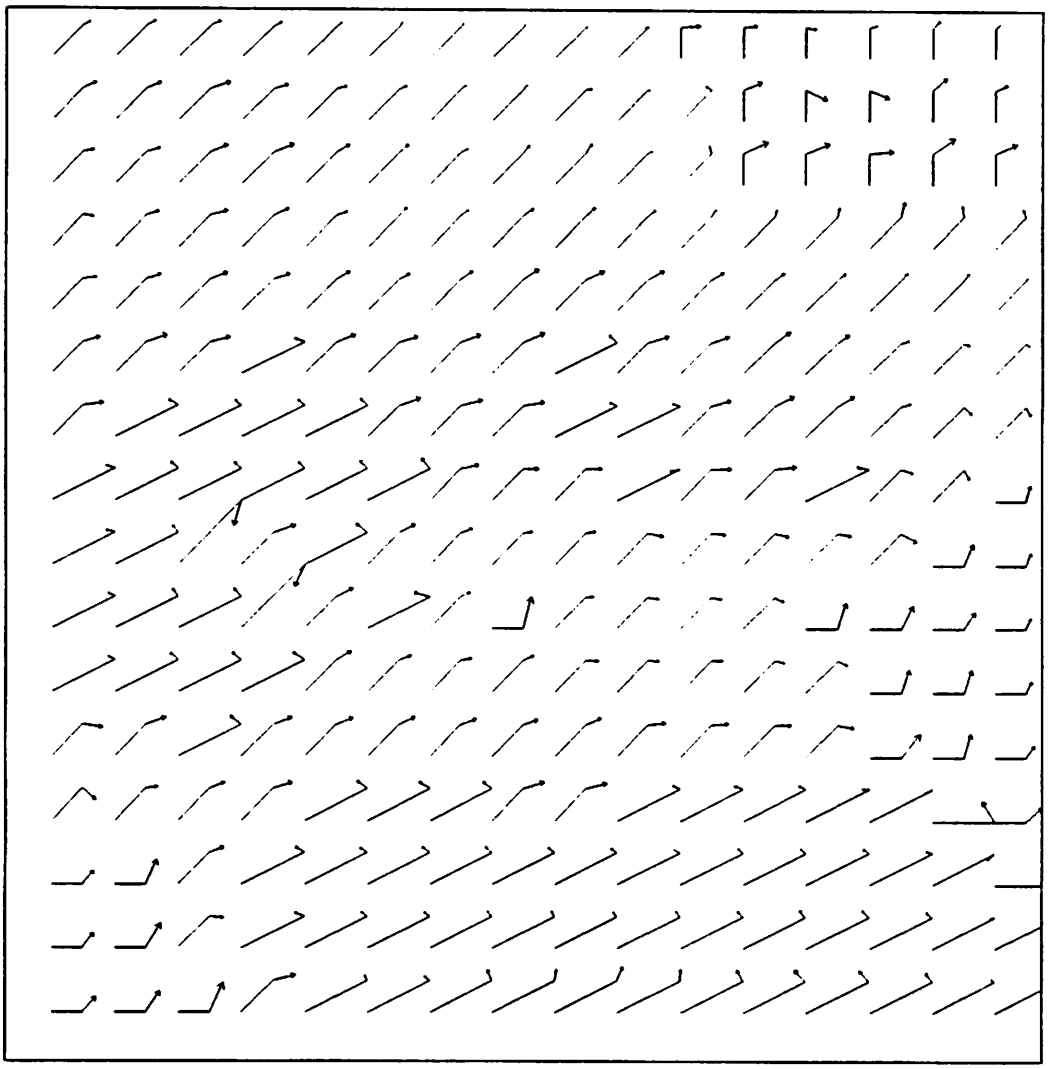


Figure 50 continued

(d) level 5, iteration 10

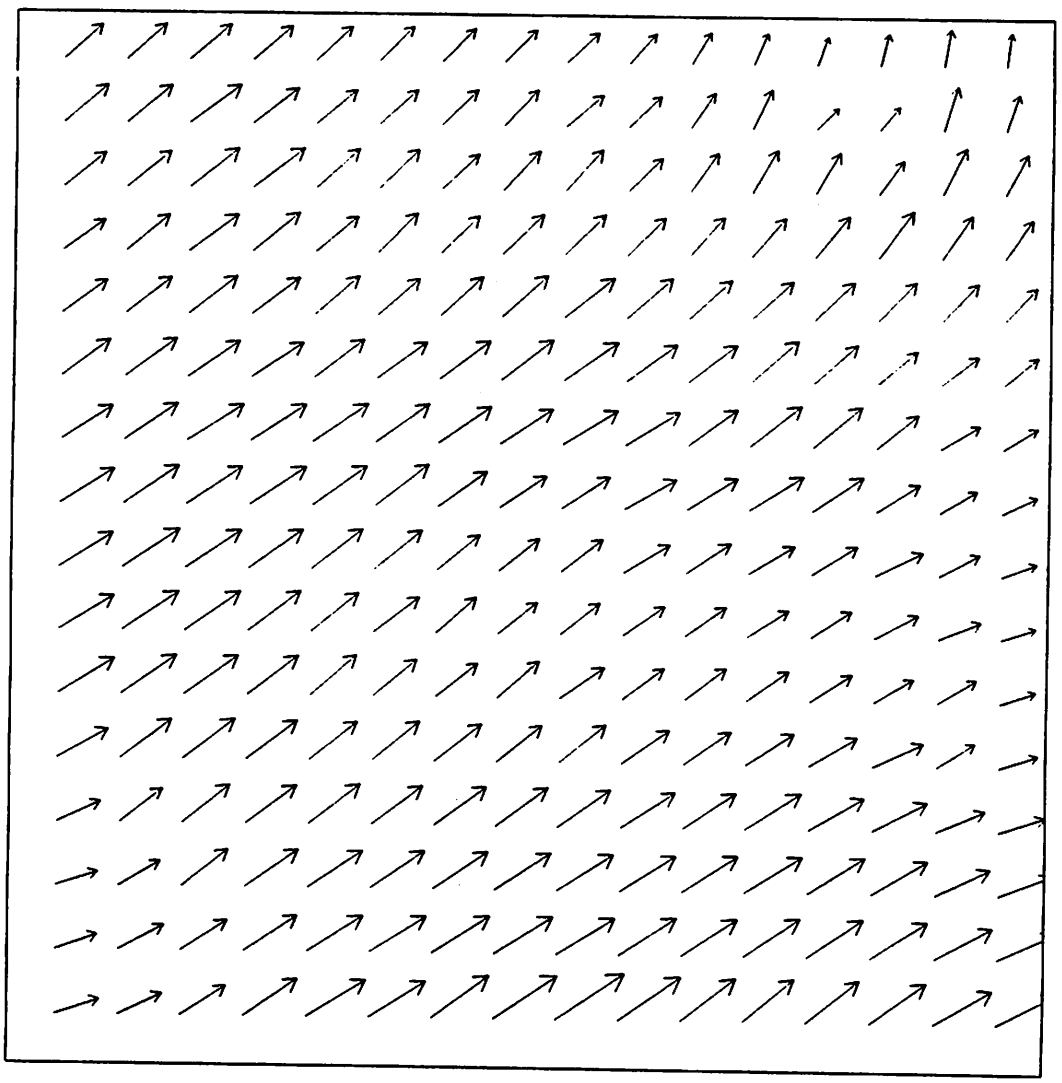


Figure 50 continued

(e) level 5, updated vectors

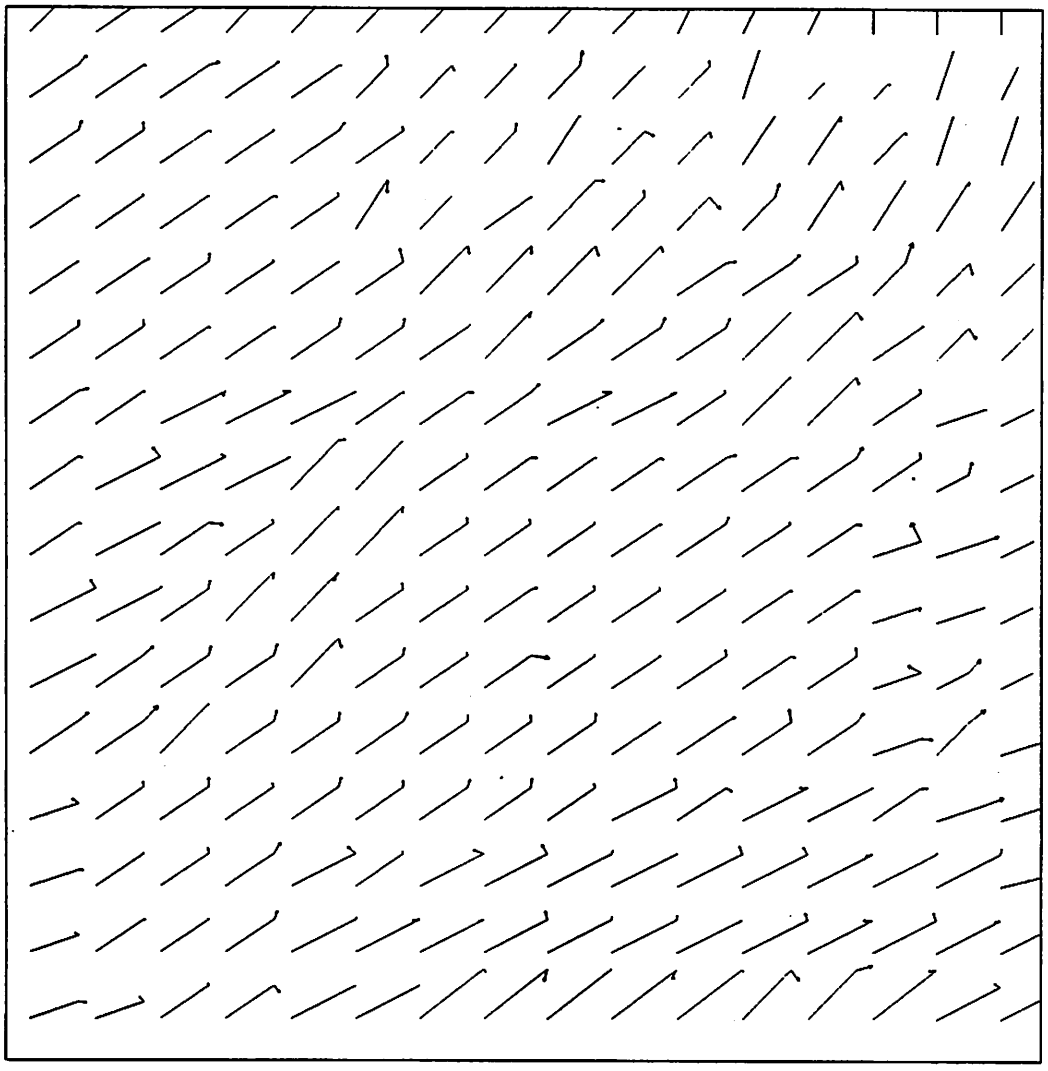


Figure 50 continued

(f) level 6, iteration 0

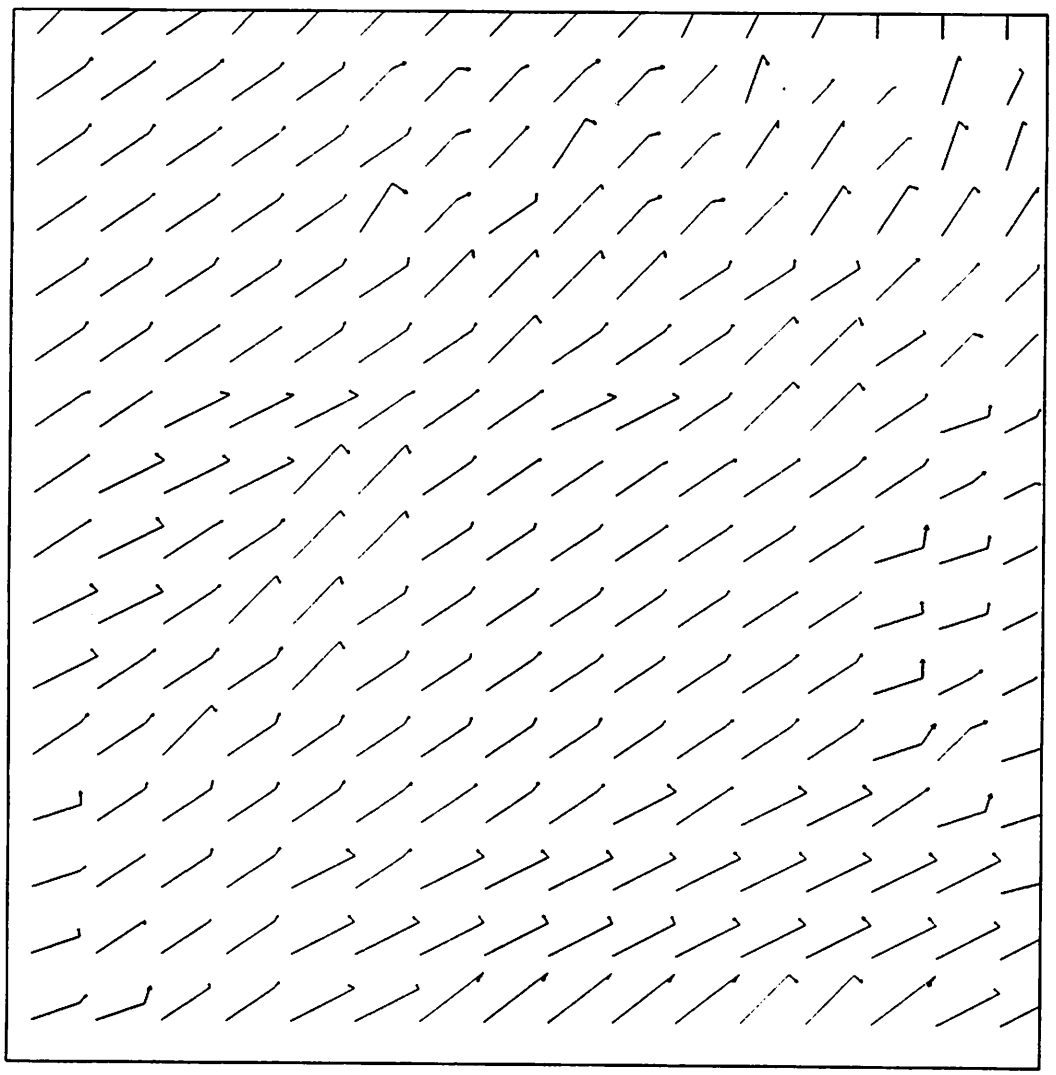


Figure 50 continued

(g) level 6, iteration 10

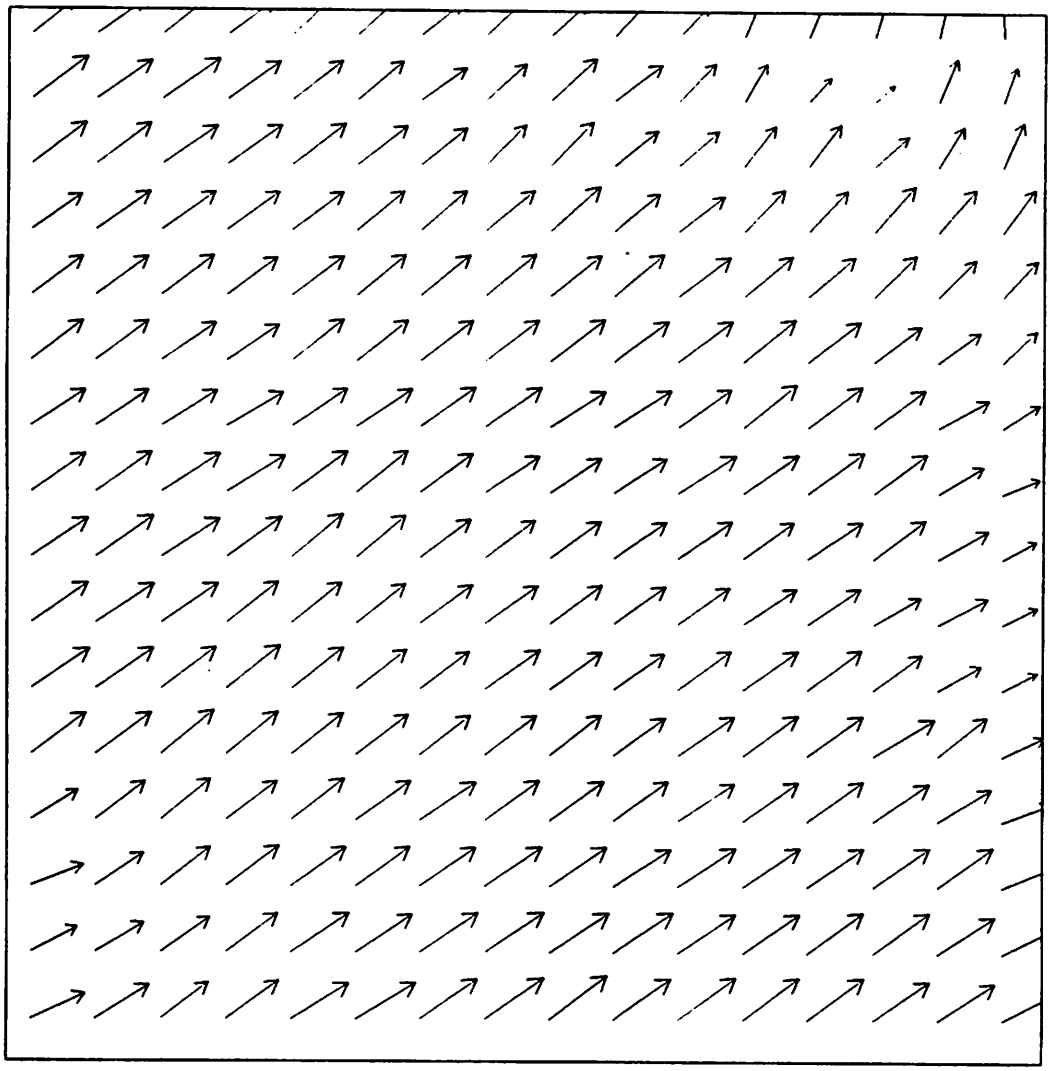


Figure 50 continued

(h) level 6, updated vectors

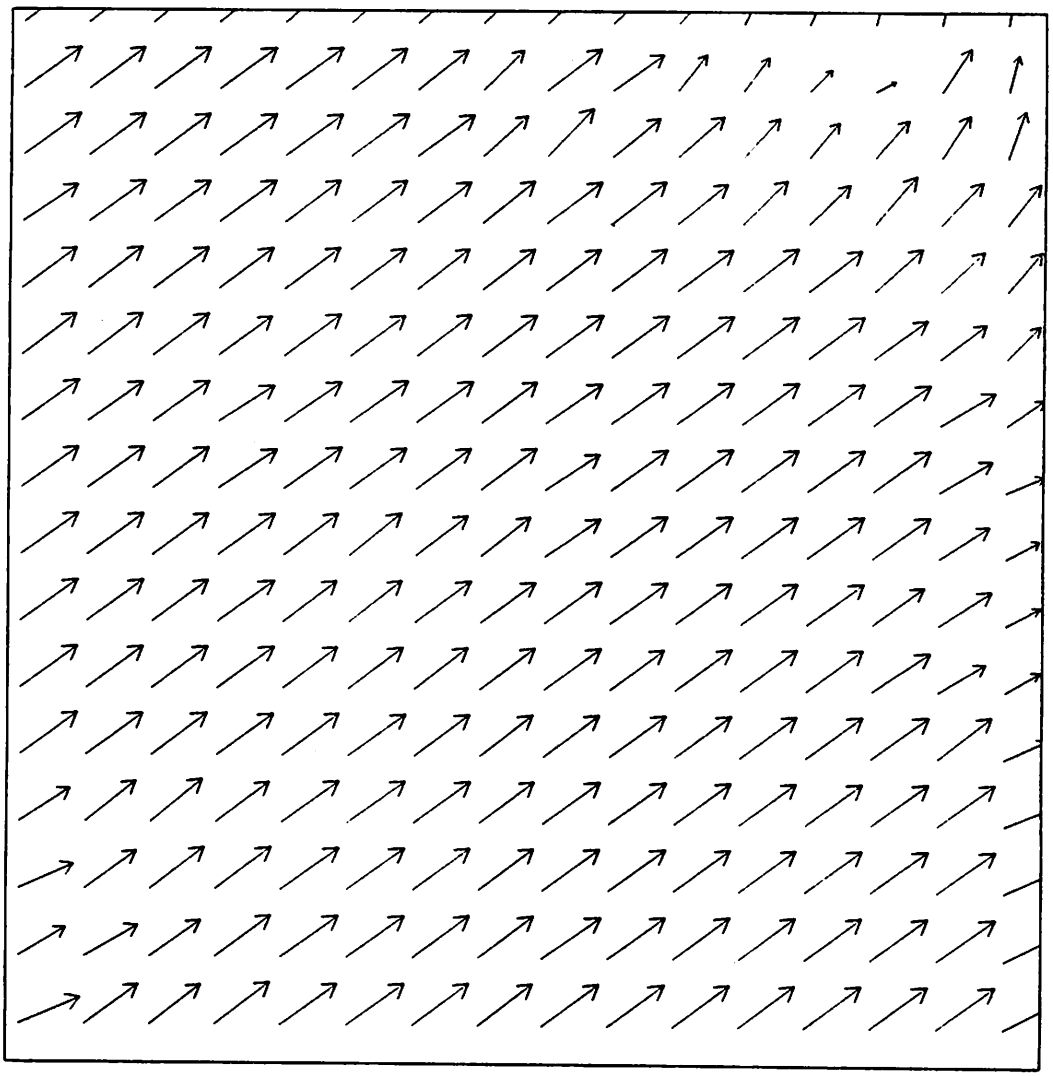


Figure 50 continued

(i) level 7, updated vectors

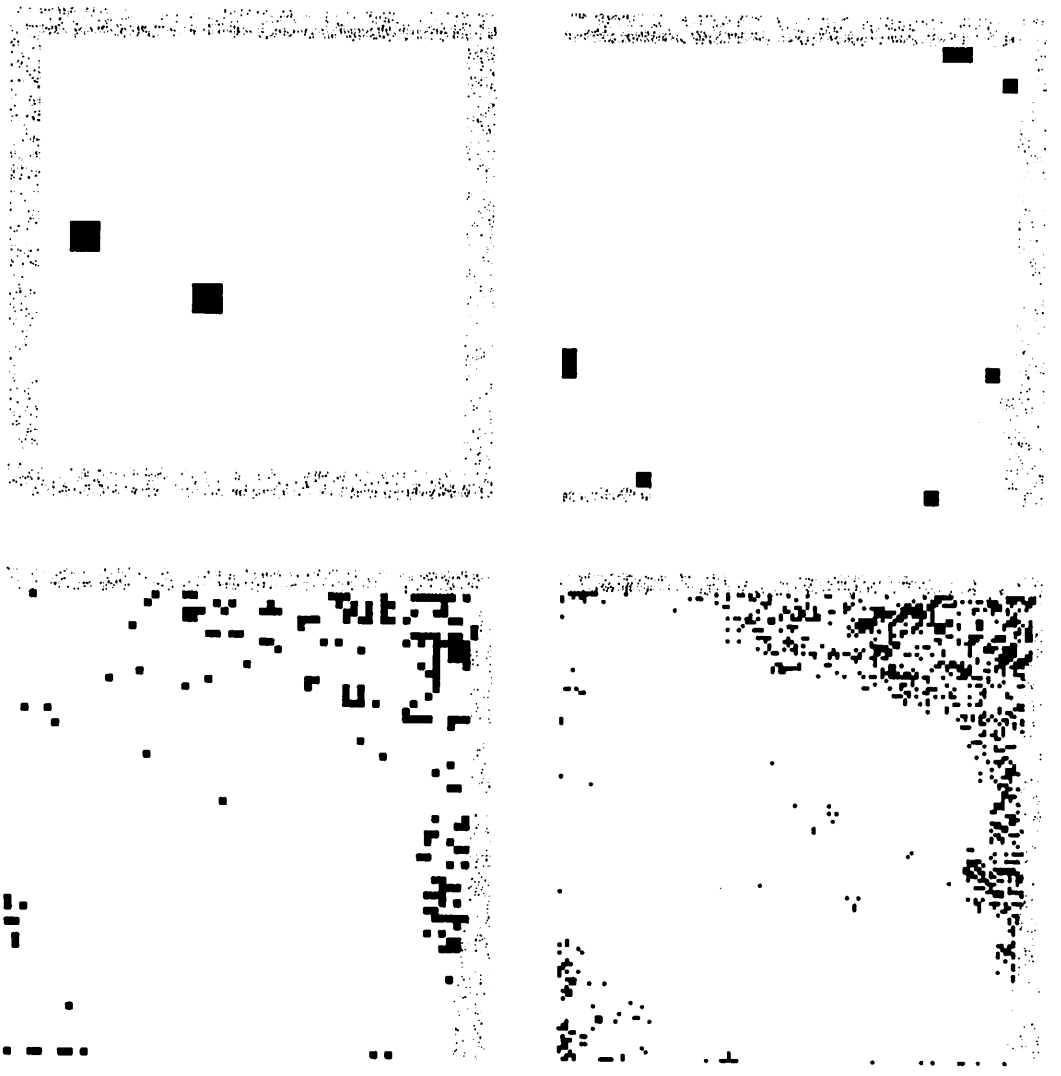


Figure 51: Multilevel experiment: error flags

Error flags at levels 4 through 7 for the multilevel experiment.

In Figure 51 the error flags are shown. Light gray pixels around the borders are SA OVERFLOW errors. Dark pixels are mostly EDGE FLOW TOO BIG errors. There were 2, 8, 188, and 1021 EDGE FLOW TOO BIG errors at levels 4, 5, 6 and 7 respectively and 1 ZERO_GRAD error at level 7.

In Figure 52, histograms of the disparity fields at each of the levels are shown. These histograms are built by counting all disparity vectors that lie within $\pm\frac{1}{2}$ of each pixel. For example, the number at location $(0,0)$ in each histogram is the number of disparity vectors with row and column components both in the range $[-\frac{1}{2},\frac{1}{2})$.

In Figure 52, disparity vectors are included in the histograms only if no error has been detected at their respective pixel locations. The expected disparities are $(-.625,.875)$, $(-1.25,1.75)$, $(-2.5,3.5)$, and $(-5,7)$ at levels 4 through 7 respectively. The center of gravity of these histograms is one measure of accuracy. For the histograms shown, the centers of gravity are $(-.8918,.5309)$, $(-1.782,1.384)$, $(-2.919,3.401)$, and $(-4.929,6.659)$ at levels 4 through 7 respectively. In all cases, both components are within $\pm\frac{1}{2}$ pixel spacing of the expected disparity values.

In Figure 53, the statistics of the row and column components of the disparity vectors are shown. Again only vectors at non-error pixels are included. The mean values are not equal to the above listed centers of gravity of the histograms, because the histograms represent disparity vectors with components truncated to the nearest integer. These can be compared to the corresponding statistics for the single level shown in Figures 46d–49d. The hierarchical method clearly succeeds where the single-level method fails.

5.3 Comparison with Hierarchical Correlation

In the next chapter, the relative computational costs of hierarchical correlation and hierarchical gradient methods will be compared. The next experiment we describe shows that these two algorithms provide comparable matching accuracy.

In Chapter IV, hierarchical correlation was run on the Mandrill subimages

Level 4

	0	1
-1	81	**92**
0	10	11

Level 5

	0	1	2
-2	9	355	326
-1	2	166	**24**
0	0	0	0

Level 6

	0	1	2	3	4
-4	0	2	4	2	0
-3	1	19	66	**1575**	**1561**
-2	1	21	18	**197**	**35**
-1	0	4	3	3	1
0	0	0	0	0	0

Level 7

	0	1	2	3	4	5	6	7	8
-8	0	0	1	2	0	0	0	0	0
-7	0	1	4	2	4	3	0	0	0
-6	0	8	19	15	27	74	62	7	2
-5	2	5	27	60	90	248	1263	**11195**	4
-4	3	24	37	25	65	118	206	95	6
-3	3	15	29	19	10	36	88	40	1
-2	0	1	13	5	8	17	8	5	0
-1	0	0	3	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 52: Hierarchical method : disparity histograms

These are two-dimensional histograms of the disparity fields at levels 4 through 7 for the multilevel experiment. At each level, the pixel location nearest the expected disparity level is surrounded by a box. (At level 6 four pixels are equidistant from the expected disparity, so the box surrounds all four.)

Level 4

Actual	D_{row}	D_{col}
mean	−.4885	.6277
SD	.1445	.1889
minimum	−.7941	.1607
maximum	−.1067	.9798
Expected	−.625	.875

Level 5

Actual	D_{row}	D_{col}
mean	−1.112	1.489
SD	.1800	.2852
minimum	−1.510	.2842
maximum	−0.2302	2.202
Expected	−1.25	1.75

Level 6

Actual	D_{row}	D_{col}
mean	−2.343	3.195
SD	.2549	.4472
minimum	−3.552	−0.1911
maximum	−0.3850	4.007
Expected	−2.5	3.5

Level 7

Actual	D_{row}	D_{col}
mean	−4.872	6.614
SD	.4211	.8675
minimum	−7.862	.0808
maximum	−0.9167	7.852
Expected	−5.	7.

Figure 53: Hierarchical method: disparity statistics

These statistics are computed over those pixels at which there were no error conditions. This included 194, 882, 3513, amd 14007 pixels at levels 4 through 7 respectively (out of a possible 256, 1024, 4096, and 16384). These are the unmarked pixels in Figure 51

with 10% noise added to the second frame. The experiment in the previous subsection did not involve such added noise. That experiment was rerun with one difference, this time 10% noise was added to the second frame. Again the noise was uncorrelated Gaussian with a mean of 0.0 and a standard deviation of 25 (10% of 255). (The noise was not identical to that used in the correlation experiment.) The results are shown in Figure 54 which shows the distribution of the disparity vectors at non-error pixels. The level 6 histogram shows exact hit accuracy of 35% (4153/11982), 3×3 accuracy of 78% (9294/11982), and 5×5 accuracy of 89% (10705/11982).

The results for the comparable hierarchical correlation experiment are shown in Figure 28 on page 104. There we see a significantly higher exact hit accuracy of 87%. However, the "outliers" are much more dispersed. For correlation, the matches at level 7 in Figure 28 are found within a 29×14 rectangular region, while for the gradient-based method, matches are found within a 11×9 region. While the final comparison may be application dependent (i.e. it depends on how the computed vectors will be used), if we give equal importance to exact hit accuracy and to limited outliers, then we can conclude that the two hierarchical methods provide comparable accuracy.

6 Summary

In this chapter we demonstrated that *first order gradient-based methods may be extended to the computation of large disparities by formulating a hierarchical generalization of the single-level method that operates on low-pass image pyramids.* A thorough formal analysis of the gradient-based updating equations was presented as well as the implementation of the method in a hierarchical processing cone algorithm. Experiments were presented both to show how single-level methods fail and how the hierarchical method succeeds.

In the previous chapter, a hierarchical correlation algorithm for motion detection algorithm was presented. The two hierarchical algorithms — correlation and

Level 4

	0	1
-1	60	**92**
0	26	15

Level 5

	0	1	2
-2	2	75	30
-1	16	537	**211**
0	0	3	1

Level 6

	-1	0	1	2	3	4
-4	0	0	4	17	10	0
-3	1	13	66	343	**1912**	**139**
-2	4	22	27	222	**453**	**20**
-1	0	2	5	20	10	0
0	0	0	0	0	0	0

Level 7

	-1	0	1	2	3	4	5	6	7	8	9
-8	0	0	0	0	0	1	1	1	0	0	0
-7	0	0	0	3	9	5	16	6	4	1	0
-6	0	2	8	13	34	84	117	133	285	66	1
-5	0	4	24	45	68	209	350	1015	**4153**	857	12
-4	8	20	32	53	37	141	305	953	1481	351	20
-3	6	24	35	11	23	88	179	241	139	17	3
-2	1	0	9	10	8	40	54	53	56	2	0
-1	0	0	0	4	2	10	30	6	2	0	0
0	0	0	0	0	1	0	0	0	0	0	0

Figure 54: Hierarchical method, noisy data : disparity histograms

These are two-dimensional histograms of the disparity fields at levels 4 through 7 for the multilevel experiment in which 10% noise was added to the second frame. Disparity vectors at non-error pixels are included. At levels 4 through 7, there were 193, 865, 3290, and 11982 vectors included out of a maximum of 256, 1024, 4096, and 16384. At each level, the pixel locations nearest the expected disparity level are surrounded by a box.

gradient-based — were seen to have comparable accuracy in determining disparity vectors. In the next chapter, the computational costs of the two hierarchical algorithms will be determined and compared. The hierarchical gradient-based algorithm was shown to be a less costly algorithm at comparable error levels.

CHAPTER VI

Computational Cost

In this section we analyze the computational costs of hierarchical gradient-based motion analysis and compare them to the costs of hierarchical correlation. A comparison of typical cases reveals hierarchical correlation to be more than four times as costly as the hierarchical gradient-based algorithm. Two types of cost are considered: **arithmetic cost** is the cost of arithmetic computations; and **transfer cost** is the cost of communications.

1 Measuring Computational Costs

Two units of arithmetic cost are used: multiplications cost $1M$ and additions cost $1A$. While $A < M$, their exact relationship cannot be generally given. We expect that in massively parallel cellular arrays and processing cones hardware support for multiplication will not exist and hence $A \ll M$. In any case, our results are independent of the relationship between A and M. All other operations we will consider will be expressed in one of these two units. Specifically subtraction, magnitude compares, and multiplies by powers of 2 (computed by bit shifts) each cost $1A$, while general divisions cost $1M$.

The cost of transferring a data value is given as T, measured in units of time. This is the time it takes to transfer one data value along a communication channel in the processing cone. Recall that these channels exist (1) between nearest neighbors within a given level; and (2) between a father and its four sons. We assume some

standard word size (number of bits) for all data values. Current implementation of cellular array processors have only single bit channels. In this case, T is the amount of time it takes to transfer all the bits of a given data value. When non-standard size data values are transferred, the relative size differences must be factored into the calculation of transfer cost.

Transfers in cellular arrays or processing cones are performed in parallel. Since these are SIMD machines, the same transfer instruction is executed at each processor at any one time. The ability to enable/disable individual processors for any given instruction allows the parallel transfer of data values even when the source to destination "vectors" vary. (For an example of such a transfer algorithm see [Weems 84, pp.293–299]) The overall cost of such a transfer of data values across the image plane is given by the maximum transfer cost over all of the individual transfers that take place in parallel.

In a hierarchical disparity computation algorithm, data must be combined from neighborhoods separated by a distance given by the approximate disparity vector. Transfer costs at a given pixel depend on the approximate disparity vector at that pixel, larger vectors bringing higher costs. The maximum transfer cost over all pixels is the correct measure to use when comparing algorithms.

Our analysis and comparison of the computation costs of the two hierarchical algorithms will be based on the computations that occur in the updating of disparity vectors at a single pixel. We do not need to calculate costs over entire levels of the processing cone nor over the entire cone. We can do this for two reasons. First, because the same processing is performed at all pixels (and moreover it is done in parallel), we need only measure the processing that takes place at a single pixel. Secondly, because the same processing is performed at all levels, we need only measure the processing that takes place at a single level.

An exception to the second rule occurs when transfer costs depend on the maximum magnitude of approximate disparity vectors. This value is highest at the bottom of the cone where fine resolution processing is done and it decreases at higher levels in the cone, to where it is equal to 1 at the initial processing level.

We distinguish between two type of transfer costs. **Disparity related** transfer costs include a term D_k equal to the maximum magnitude of disparity estimates at level k. If D is the maximum disparity magnitude at the bottom level L, then $D_k = D/2^{L-k} = 2^{k-L}D$. **Non disparity related** costs are independent of D_k.

Error handling is not analyzed since, in the algorithms we have presented, the occurrence of errors precludes most pixel processing and hence incurs costs that are strictly less than the errorless case.

2 Costs of the Hierarchical Correlation Algorithm

The hierarchical correlation algorithm that we analyze is shown in Figure 25. First consider the arithmetic costs. Suppose hierarchical correlation is done with a sample window size of $w \times w$. At a given pixel in the first image a neighborhood of w^2 points will be multiplied by nine neighborhoods in the second image. This will cost

$$C_A^C = 9w^2(M + A) \tag{41}$$

(the subscript A stands for "arithmetic cost", while the superscript C stands for "correlation"). If 8×8 neighborhoods are being used the correlation cost is $576(M + A)$. This is by far the largest cost in the algorithm, and for our purposes, we can ignore the rest. (For example, if we assume that integer comparisons cost $1A$ then the selection of the maximum correlation from the nine possibilities only costs $8A$.)

Now consider the communication costs. Two neighborhoods of data must be accessed. In the first image, we must access the $w \times w$ neighborhood centered at the processing pixel. In the second image, we must access the $(w + 2) \times (w + 2)$ neighborhood which contains the nine $w \times w$ neighborhoods centered at pixels in the 3×3 search area about the pixel pointed to by the approximate disparity vector. (The algorithm as shown in Figure 25 suggests that nine transfers of $w \times w$ neighborhoods in the second image are performed. This is highly redundant and inefficient, since these neighborhoods largely overlap.)

Let the transfer costs for these two neighborhoods be $C_{T,1}$ and $C_{T,2}$. In Appendix A it is shown that the cost of accessing the neighborhood in the first image is

$$C_{T,1} \approx \frac{1}{2}w^3 \qquad (\text{for } w \gg 1)$$

The cost of accessing the second neighborhood depends on the maximum magnitude D_k of the disparity vectors (a, b). This value is defined as

$$D_k \triangleq \max_{(x,y)} \|(a,b)\|_{Max} = \max_{(x,y)} |\max(|a|, |b|)|$$

The Max norm is used because the hierarchical search we use is constrained separately to a given maximum disparity in the two directions X and Y.

Now, letting $w_2 \triangleq (w+2)$ be the size of the second neighborhood, then from Equations 2 to 5 in Appendix A, we know that

$$C_{T,2} \geq \binom{D_k^2}{w_2} + \binom{w_2}{2} w_2^2 T \qquad (\text{for } D_k < r) \tag{42}$$

$$\geq \frac{1}{2}w_2^3 T \tag{43}$$

and

$$C_{T,2} \geq \left(D_k + \frac{w_2}{4}\right) w_2^2 \qquad (\text{for } r \leq D_k) \tag{44}$$

$$\geq \left(\frac{3w_2}{4} - \frac{1}{2}\right) w_2^2 \tag{45}$$

Equations 42 and 44 provide stronger bounds but include the value D_k. Equations 43 and 45 eliminate the D_k term, but give weaker bounds. As a final simplification, if we assume no knowledge of the value D_k, we must then use Equation 43, the weakest overall condition. Using this bound, the total transfer cost is then bounded by

$$C_T^C \triangleq C_{T,1} + C_{T,2} \geq \frac{1}{2}(w^3 + (w+2)^3) \tag{46}$$

Even though we have ignored the maximum magnitude disparity D_k, this is still a very large number of transfers. If 8×8 neighborhoods are being used, then communication cost is $C_T^C \geq \frac{1}{2}(8^3 + 10^3)T = 756T$.

3 Costs of the Hierarchical Gradient-Based Algorithm

The overall hierarchical algorithm is shown in Figure 41. Almost all computations take place in the projection, gradients, edge flow, and relaxation operations. These are shown in Figures 42 through 45. We now consider each of these in turn.

The cost of projection (Figure 42 on page 140) is $C_T^{proj} = 2T$ for transfer of the two disparity components and $C_A^{proj} = 2A$ to double them.

The gradient computation (Figure 43 on page 142) is three weighted sums of pixel values in corresponding neighborhoods in the two images. This is most efficiently performed by computing partial weighted sums in each image, and then bringing these sums together. In the next paragraph we show that the arithmetic cost is $C_A^{grad} = 61A$ given the filters we have used and the communication cost is $C_T^{grad} = 64T + 3Td$, where d is the maximum disparity.

We analyze the costs of gradient computation in detail here. First differences were taken using the 3×3 filters on page 128 for δ_x and δ_y. Taking advantage of their anti-symmetry and the simple coefficients, the arithmetic cost of one application of one these is filters is $8A$ ($4A$ to add in the four corners; and $4A$ to add two values, multiply by 2, and add in) and the transfer cost is $10T$ ($2T$ for each of the corners and $1T$ for each of the two sides). Two in each image then costs $32A$ and $40T$. After the single image values are brought together two more additions and divisions by two bring the costs to $36A$ and $40T + 2Td$. Now consider the third gradient which used the filter G_2. Taking advantage of the symmetry of this filter and its simple coefficients, the arithmetic cost of one application is $12A$ ($4A$ to add in the four corners; $6A$ to add four values, multiply them by 2, and add them in; and $2A$ to multiply the center value by 4 and add it in) and the transfer cost is $12T$ ($2T$ for each of the corners and $1T$ for each of the four "sides"). The costs of two such sums plus one more addition when the two values are brought together are $25A$ and $24T + d$. The total arithmetic cost of the three gradients is $C_A^{grad} = 36A + 25A = 61A$. The total transfer cost of the three gradients is

$C_T^{grad} = (40T + 2Td) + (24T + d) = 64T + 3Td.$

The edge flow is computed from the three gradients (Figure 44 on page 144). The arithmetic cost is $C_A^{eflow} = 5M + A$. There is no communication cost, $C_T^{eflow} = 0$.

Finally we consider relaxation (Figure 45 on page 145). Let us write the main update equation as:

$$(u, v) \leftarrow (\tilde{u}, \tilde{v}) - |(f_x, f_y) \cdot (\tilde{u}, \tilde{v})| \, (g_1, g_2)$$

where

$$(g_1, g_2) = \left[\frac{1}{\kappa\alpha^2 + f_x^2 + f_y^2} \right] (f_x, f_y)$$

and

$$(\tilde{u}, \tilde{v}) = (\hat{u}, \hat{v}) + (\Delta U, \Delta V)$$

Since the relaxation step is iterated, computations which don't change between iterations should only be performed once. This includes the computation of $(\Delta U, \Delta V)$ — the discrete Laplacian of the approximate disparity vector — and of (g_1, g_2).

First we consider the one time costs. The discrete Laplacians $(\Delta U, \Delta V)$ are computed using the filter shown on page 145. Taking advantage of the symmetry of this filter and its simple coefficients, the arithmetic cost of one application is $M + 12A$ ($4A$ to add in the four corners; $6A$ to add four values, multiply them by 2, and add them in; $M + A$ to multiply the center value by 12 and subtract it; and finally $1A$ to divide by 4). The transfer cost of one application is $12T$ ($2T$ for each of the corners and $1T$ for each of the four "sides"). The vector (g_1, g_2) is computed once at a cost of $4M + 2A$ ($2M + 2A$ for the denominator; $2M$ for the two terms). The total arithmetic cost for the non-iterated computations is $C_A^{R_n} = 2(M + 12A) + (4M + 2A) = 6M + 26A$. The total transfer cost for the non-iterated computations is $C_T^{R_n} = 2(12T) = 24T$.

Now consider the iterated costs. First consider $(\tilde{u}, \tilde{v})$. These two local averages of the components on the update vector are computed using the filter on page 145. Taking symmetries and the simple coefficients into account, the arithmetic cost of one application is $M + 10A$ ($4A$ to add in the four corners; $6A$ to add four values,

multiply them by 2, and add them in; and $1M$ to divide by 12) and the transfer cost is $12T$ ($2T$ for each of the corners and $1T$ for each of the four "sides"). The costs for $(\tilde{u}, \tilde{v})$ are then $2M + 20A$ and $24T$. Finally we combine everything using one inner product, a scalar times a vector, and a vector addition, adding $(2M+A)+2M+2A = 4M + 3A$ arithmetic cost and no further communication cost. The total arithmetic cost for the iterated computations is $C_A^{R_i} = (2M + 20A) + (4M + 3A) = 6M + 23A$. The total transfer cost for the iterated computations is $C_T^{R_i} = 24T$.

Now let R be the number of iterations that are performed. The total arithmetic cost of R iterations is $C_A^{relax} = C_A^{R_u} + RC_A^{R_i} = (6M + 26A) + R(6M + 23A)$. The total transfer cost is $C_T^{relax} = C_T^{R_u} + RC_T^{R_i} = (24T) + R(24T) = (R + 1)24T$. For example, with $R = 10$ as in our experiments, this gives $C_A^{relax} = 66M + 259A$.

Finally, we add up all the costs to get

$$\begin{aligned} C_A^{GB} &= C_A^{proj} + C_A^{grad} + C_A^{eflow} + C_A^{relax} \\ &= 2A + 61A + (5M + A) + (6M + 26A) + R(6M + 23A) \\ &= 90A + 11M + R(6M + 23A) \end{aligned} \tag{47}$$

and

$$\begin{aligned} C_T^{GB} &= C_T^{proj} + C_T^{grad} + C_T^{eflow} + C_T^{relax} \\ &= 2T + (64T + 3Td) + 0 + (R + 1)24T \\ &= (90 + 3d + 24R)T \end{aligned} \tag{48}$$

4 Comparison of the Two Hierarchical Methods

In Table 3, we show representative costs of the two hierarchical algorithms. These costs are calculated from Equations 41, 46, 47, and 48. The computational costs of hierarchical correlation are high for two reasons: (1) many multiplications and additions are needed for the correlation inner product; and (2) many pixel values must be transferred over multi-pixel distances in the image plane. Also, note that

Hierarchical Correlation		
w	C_A^C	C_T^C
6	$324(M+A)$	$364T$
8	$576(M+A)$	$756T$
10	$900(M+A)$	$1364T$
12	$1296(M+A)$	$2236T$

Hierarchical Gradient-Based			
R	d	C_A^{GB}	C_T^{GB}
5	0	$120A+126M$	$210T$
5	10	$120A+126M$	$240T$
10	0	$150A+241M$	$330T$
10	10	$150A+241M$	$360T$

Table 3: Cost Comparison of Hierarchical Algorithms

relative to the parameter w, the arithmetic costs of correlation increase as the square, and the transfer costs increase as the cube. In contrast, both costs of the gradient-based method increase linearly relative to the parameters R and d.

In general, the hierarchical gradient-based algorithm is less costly than hierarchical correlation. The costs are only comparable when very small 6×6 sample windows are used in the correlation algorithm and a relatively large disparity is used in the gradient-based. However, since we have used the weakest bounds in the correlation case, derived by assuming $d = 0$, it is actually more expensive than the tables show. If we compare the cases of $w = 10$ versus $R = 10$ and $d = 0$, we see four times more transfer costs and at least four times more arithmetic cost for the hierarchical algorithm.

CHAPTER VII

Multilevel Optic Flow Relaxation

1 Introduction

The computational cost of using relaxation to compute optic flow is very dependent on the number of iterations needed. This number is in turn dependent on the image data. We will demonstrate how the lack of fine details in an image necessitates more iterations to compute optic flow. In this chapter we show how a hierarchical relaxation scheme in the processing cone can be used to reduce this cost and make it relatively independent of the image data. Multilevel relaxation techniques will be presented which essentially solve for coarse components (low frequencies) of the solution at coarse levels of the processing cone and fine components (high frequencies) at fine levels.

In general, relaxation operators satisfy our goals of obtaining algorithms which 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 due to the slow propagation of global information by the local update processes. In the last ten years, multilevel relaxation techniques have been introduced to overcome this problem of asymptotically slow convergence. [Brandt 77a, Brandt 77b, McCormick & Trottenberg 83]. Multilevel relaxation is an algorithmic extension of iterative relaxation. By representing the spatial domain at multiple levels of resolution these algorithms apply the basic local iterative up-

date to a range of neighborhood sizes. Local updates on coarser grids introduce a more global propagation of information.

Recently, the general usefulness of multilevel relaxation in low-level computer vision has been noted and applications have been made [Glazer 82, Terzopolous 82, Terzopolous 84]. In the next section we briefly describe the general class of problems for which multilevel relaxation algorithms can be used. These problems are characterized by (1) variational formulations of global measures of constraint satisfaction; (2) discrete formulations of the continuous problem, e.g. finite difference approximations; and (3) relaxation algorithms involving the iterative application of local neighborhood operators.

In Section 3, multilevel relaxation is formally described. A simple example of multilevel relaxation is given in Section 4. The remainder of this chapter is then devoted to the application of multilevel relaxation to the computation of optic flow within the processing cone. The multilevel optic flow relaxation algorithm is presented in Section 5. In Section 6, the single-level and multilevel algorithms are compared. Some multilevel cases are seen to diverge at coarse levels of the processing cone. The next two sections contain a further analysis of the multilevel optic flow equations. In Section 7, a *local mode analysis* is performed to provide insight into the convergence properties of multilevel optic flow relaxation. In Section 8, the relationship between spatial variation in the image data and convergence/divergence of relaxation is shown. Finally in Section 9, a fixed cycling scheme is employed to successfully avoid the problem of divergent relaxation.

2 Optimization Problems in Low-Level Vision

The iterative methods used to compute disparity fields in Chapter V and the related optic flow algorithm of Horn and Schunck are computationally similar to a large class of other low-level vision algorithms. Generally speaking, these al-

gorithms solve problems in the dense interpolation or approximation of sparsely-known, noisy or partially constrained data. A few examples are image smoothing [Narayanan *et al.* 82] — the approximation of noisy data; surface interpolation [Grimson 81b, Ikeuchi 80, Ikeuchi & Horn 81] — the interpolation of sparse data; and optic flow computation [Horn & Schunck 81] — the approximation of partially constrained data. In all cases, global requirements of total smoothness are specified and algorithms that involve only local neighborhood operations can be designed. Smoothness constraints and problem-dependent constraints are expressed formally as variational problems — integrals to be minimized over the entire image domain — or as equivalent partial differential equations (PDE's). Variational problems specify a global measure to be optimized. The equivalent PDE's provide local constraints that must hold at each point.

Using finite difference approximations, discrete approximations of the variational problem or of the equivalent PDE are defined over a discrete grid of points. Variational problems can be represented as summations over the discrete image grid. PDE's over the image space can be represented as a set of equations, one per grid point, involving the values at neighboring points.

Discrete variational problems can be solved by gradient methods, which take the form of iterative algorithms commonly associated with optimization theory. (These gradient methods are not related to the gradient-based methods for motion computation.) For linear PDE problems, the set of finite difference equations can be represented as a single matrix equation in which the solution image is the "vector" to be solved for. These can be solved by iterative algorithms employing local neighborhood updating.

Before examining the above ideas in more detail, the next section presents some specific examples of recent work.

2.1 Four examples

Narayanan *et al.* [Narayanan *et al.* 82] 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

$$\sum_{x,y} [R^2 + \alpha(F-G)^2] \tag{49}$$

where $R(x,y)$ is a roughness measure of G, computed for a neighborhood of each point and α is a relative weighting factor. To the extent that G is not smooth, the first term R adds relatively larger values to the error measure. The second term adds to the error measure in proportion to the 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 method was used to minimize the above cost functional. A general discussion of optimization methods can be found in [Luenberger 73] or [Gill *et al.* 81].

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 (interpolating) or near (approximating) points of known depth [Grimson 81b]. Approximation was formulated as minimization of the functional

$$\iint (S_{xx}^2 + 2S_{xy}^2 + S_{yy}^2)\, dxdy + B \sum_P [S(x,y) - C(x,y)]^2 \tag{50}$$

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 [Luenberger 73, Gill *et al.* 81]. Interpolation was formulated as minimization of the functional

$$\iint (S_{xx}^2 + 2S_{xy}^2 + S_{yy}^2)\, dxdy \tag{51}$$

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 [Luenberger 73, Gill *et al.* 81].

Ikeuchi approached the problem of shape from shading as follows [Ikeuchi 80, Ikeuchi & Horn 81]. Given a brightness image $I(x,y)$, a reflectance map $R(f,g)$, and an occluding contour $\partial\mathcal{A}$, find the surface orientation $(F(x,y), G(x,y))$ in $\mathcal{A}$

which minimizes the functional

$$\sum_{x,y} [I - R(F,G)]^2 + \alpha|(\nabla F)^2 + (\nabla G)^2| \tag{52}$$

where ∇ is a discrete gradient operator. The first term of the functional penalizes for deviation from the image irradiance/reflectance equation $I(x,y) = R(f,g)$. The second term penalizes for the 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 as described in Chapter III, Section 4.3.4 [Horn & Schunck 81]. Recall that given a dynamic image $F(x,y,t)$, their approach is to find the optic flow vector field $(u(x,y), v(x,y))$ which minimizes the functional

$$\iint (F_x u + F_y v + F_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2)\, dxdy \tag{53}$$

The first term in the functional is the square of the rate of change of image brightness (measured in 3D space-time of the dynamic image space). When $F_x u + F_y v + F_t = 0$, 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 α 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 5.

2.2 The General Framework

We will now consider a framework of that makes clear the similarities and differences among the methods described above. Figure 55 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. Single-grid and multilevel solutions to this problem are presented in Section 4.

2.2.1 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 [Courant & Hilbert 53, Section IV.3]. (Figure 55A → Figure 55B). 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 55. Both Grimson and Horn & Schunck began their analyses with variational problem statements.

2.2.2 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 55. The integral in a variational problem becomes a summation over a discrete set of grid points (Figure 55A → Figure 55C). In Figure 55C this gives a summation, over the 2D grid indices i and j, of first finite difference approximations of the horizontal and vertical partial derivatives. Narayanan *et al.* began their analysis with such a discrete minimization representation, without any continuous statement of the problem.

In Figure 55D 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 2D convolution with the 5-point Laplacian mask. This system of equations can also be generated from the discrete functional by 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 55C to Figure 55D. It is analogous to using the first variation of the continuous functional to derive the corresponding PDE; Ikeuchi takes this path.

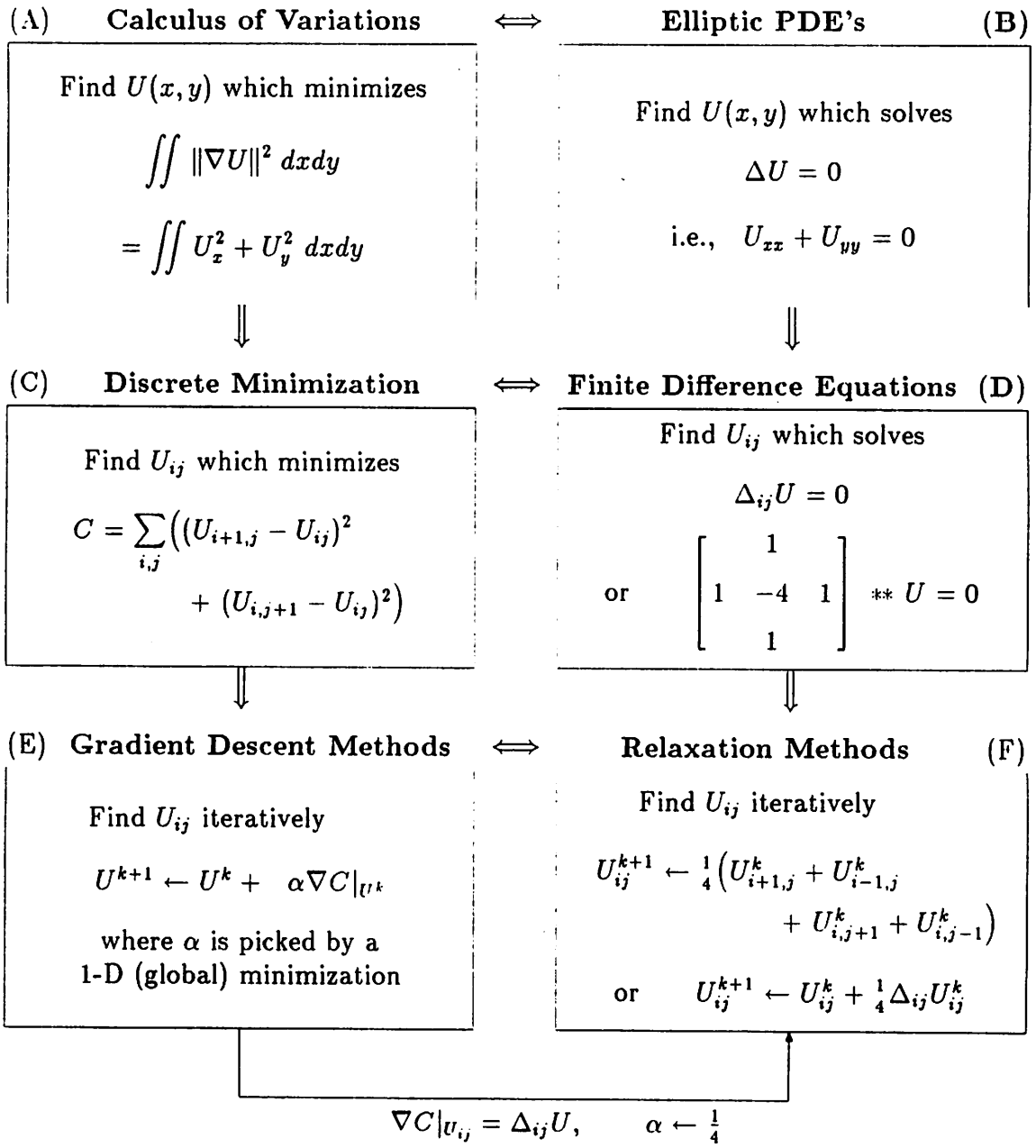


Figure 55: General framework of approaches and algorithms

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

2.2.3 Algorithms

Finally, we come to specific algorithms for solving these discrete problems. Consider first the system of finite difference equations in Figure 55D. 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 or "forcing terms", including the boundary conditions. The fact that each equation only relates the values at nearby points makes the matrix very sparse. Any technique for solving sparse matrix equations could then be applied. Although many methods exist for solving such a system of equations (see Section 3.1), 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 55C. This is a problem that can be solved using the techniques of mathematical programming (also known as non-linear programming and cost minimization) [Luenberger 73, Gill *et al.* 81]. 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. The direction vector in these problems is actually an $n \times n$ plane of data. The one-dimensional search computes the amount of the direction vector to be added to the current solution estimate. Step 1 is a local operation but step 2 is global. (This is explained in the next paragraph.) 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 55E where the current iteration U^k is updated by an optimally chosen 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 55 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 [Grimson 81b, Narayanan *et al.* 82]. 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.

2.2.4 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 [Terzopolous 82]. While not equivalent to finite difference methods, FEM's 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 55.

3 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 homogeneous. 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 [Brandt 77b, p.281]. Grimson's algorithm, requiring second order smoothness, takes thousands of iterations to approach final solutions. (This number is based on our unreported experiments. See also [Terzopolous 82].) In the problem domain of elliptic PDE's this slowness has been overcome by using multilevel relaxation algorithms [Brandt 77a, Brandt 77b].

In this approach, a standard iterative algorithm is applied on grids of different resolution. Each grid covers the entire data domain and they are organized in a hierarchical structure. This is in fact the processing cone architecture. 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 $O(d)$ [Brandt 77b, p.278].

3.1 Solving the Set of Grid Point Equations

All of the discrete problems we have discussed result in the formulation of a system of equations, one per grid point, each of which describes how the value at that grid point is related to the values at neighboring grid points. For linear problems, we get a linear system of equations which we can represent as a matrix equation $A\mathbf{x} = \mathbf{b}$. For an $n \times n$ grid, there are $N = n^2$ equations and A is an $N \times N = n^2 \times n^2$ matrix. Since any one equation only relates the values at grid points in a small neighborhood, the matrix is sparse and given a suitable grid point numbering scheme it is banded with a bandwidth of order $O(n)$. Solving for $\mathbf{x}$ is equivalent to inverting the matrix A. In this section, we discuss the general range of algorithms for solving such problems. See [Smith 78, Hageman & Young 81, Rice 83, Birkhoff & Lynch 84] for further details.

Two major classes of algorithms are used to solve such matrix equations: direct methods and iterative methods. **Direct methods** give the solution in a finite number of steps. They solve for A^{-1} by a sequence of operations on $n^2 \times n^2$ matrices. Well known direct methods include *Gaussian elimination* and *LU decomposition*. For these methods, full storage of the n^4 entries of A is required and $N^3/3 = n^6/3$ additions and multiplications must be performed. For banded matrices with bandwidth $O(n)$, *band elimination* can be used to reduce the storage requirements to $O(Nn) = O(n^3)$ and the operation count to $N^2 + O(Nn) = n^4 + O(n^3)$. For symmetric positive definite matrices, *Cholesky decomposition* can be used to halve storage and computation costs in both the general case and for banded matrices.

There are many refinements and extensions of these algorithms which further reduce costs.

In general, direct methods are *not local* because the values at distant points in the grid may be combined and they are *not parallel* since only single variables are changed at a time. Moreover the manipulation of $n^2 \times n^2$ matrices *cannot be implemented in retinotopic architectures*. Thus, we are led to the other major class of algorithms, the iterative methods.

Iterative methods compute an infinite sequence of better and better approximations $\mathbf{x}^1, \mathbf{x}^2, \ldots$ by iterating some process $\mathbf{x}^{s+1} \leftarrow S[\mathbf{x}^s]$. The operation S is determined by the matrix A, but it does not involve any operations on $n^2 \times n^2$ matrices; only values of $\mathbf{x}$ (an $n^2 \times 1$ vector) are manipulated. On standard sequential computers, iterative methods were preferred over direct methods in the days when memory was costly. As core memory got larger and virtual memory techniques were developed, direct methods came into play. Iterative methods are still cost effective on sequential machines when matrices are large and sparse.

Our interest in iterative methods is due to the further constraints on algorithms that our highly parallel architecture necessitates. Since these methods operate in the discrete $n \times n$ image space, they are *naturally mapped to cellular grid architectures*. At any one step in the iteration, elements of $\mathbf{x}$ are updated to better satisfy some of the equations. Since these equations are similar for all grid points and since they only relate neighboring grid points, the application of S to $\mathbf{x}$ is both *uniform* and *local*. Moreover, certain iterative methods, e.g. the Jacobi method, are also *parallel* because they allow all elements of $\mathbf{x}$ to be update simultaneously. By selecting such a method we satisfy our goal of obtaining a uniform, local, and parallel algorithm operating in a retino-topic architecture. The problem remains that such iterative methods may be very slow in that they require many iterations to reach an adequate approximate solution.

3.2 The Multilevel Method

The slow convergence of relaxation algorithms 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. The greater the range over which information must be sent, the slower the smoothing will be. Looking ahead to Figure 58 in which interpolation from given boundary data is performed by local averaging at each grid point, we can see how points far from the constraining boundary data take a long time to reach their final values.

A longstanding technique used to overcome this problem is to initialize relaxation with a coarsely computed estimate of the solution. In a processing cone, coarse approximations to the problem can be computed at a coarse level and the coarse solution can be projected down to a finer level to serve as the initial estimate for relaxation at the fine level. Such an approach can be applied at multiple levels of resolution in a coarse-to-fine strategy.

While significantly speeding up the early stages of relaxation, ultimate convergence is still slow. Projection of coarse solutions for use as initial fine estimates does provide a "head start" over single-level methods. However, after a few iterations of relaxation at the finer level, error components remain that are attenuated only very slowly in subsequent iterations. Brandt [Brandt 77a, Brandt 77b] significantly extended the multilevel approach by showing how the coarser grid can be used in the process of improving the first approximation. His algorithms take the problem of error removal and create coarse level approximate problems which can be solved more quickly on coarser grids. Brandt combined the techniques of projecting estimates and coarsening "residual equations" to develop iterative schemes that move up and down a cone of varying resolution grids.

The extent to which one pass of relaxation computes a new approximation closer to the actual solution is dependent on the spatial frequency content of the error. 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 [Brandt 77a, Brandt 77b] whereby the error reduction is considered as a function of spatial frequency.

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 multilevel cyclic algorithms running in the processing cone. 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.

The basic operators that are involved in a multilevel relaxation scheme are:

relax : Reduce error components at a given level.

This step is defined by the discrete approximation to the PDE to be solved and the particular iterative update scheme chosen.

project : Send coarse solutions down.

The essential step here is the projection of corrections by interpolation from coarse to fine grids.

reduce : Send problems to be solved up.

Coarse equations whose solution is the correction at a fine level are created by this process.

3.3 Formal Development

In the basic multilevel method the domain of definition $\mathcal{A}$ of the problem is represented discretely by a set of uniform square grids $G^0, \ldots, G^k, \ldots, G^M$ with mesh-sizes $h_0, \ldots, h_k, \ldots, h_M$. Each grid covers the full domain and we assume that the interlevel mesh size ratio is $\rho \triangleq h_i/h_{i+1} = 2$. Brandt advises that this ratio always be used, since it provides close to optimal convergence rates, and the simplest intergrid architecture [Brandt 77a, p.353].

Suppose we have a PDE with suitable boundary conditions which we wish to solve over $\mathcal{A}$, such as $\Delta U(x,y) = 0$. We represent this as:

$$LU(\mathbf{x}) = F(\mathbf{x}) \qquad \text{for all } \mathbf{x} \in \mathcal{A} \tag{54}$$

where L is a general differential operator and U and F are vector valued functions over $\Re^n$. U is the solution we seek. At each level k we can form the finite difference approximation to the PDE's;

$$L^k U^k(\mathbf{x}^k) = F(\mathbf{x}^k) \qquad \text{for all } \mathbf{x}^k \in G^k \tag{55}$$

If L is a linear operator, then Equation 55 is a matrix equation relating values at neighboring grid points.

Following Brandt's development [Brandt 77a], let u^M be an approximate solution of the G^M problem with $L^M u^M = f^M$. The error is $U^M - u^M$ and we define the **residual** to be $r^M \triangleq F^M - f^M$. The residual is the difference between the desired and the actual right-hand-side. Then

$$L^M U^M - L^M u^M = F^M - f^M \triangleq r^M \tag{56}$$

If L is linear and if we let $v^M \triangleq U^M - u^M$, then v^M satisfies the **residual equation**:

$$L^M V^M = r^M \qquad (\triangleq F^M - f^M) \tag{57}$$

and the exact discrete solution is $U^M = u^M + v^M$. If u^M is computed by some relaxation iterations on G^M then r^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 r^M is smoothed out and convergence has slowed down.

The coarsened residual equation is

$$L^k V^k = I_M^k r^M \tag{58}$$

where I_M^k is a reduction operator which computes a coarse (level k) version of r^M. Relaxation on the coarse grid produces an approximation v^k of the correction V^M. An improved level M solution is then obtained by projecting v^k to level M and adding this interpolated correction to u^M, i.e.

$$u^M \leftarrow u^M + I_k^M v^k \tag{59}$$

3.3.1 Interpolation: Projections and Reductions

The notation I_m^k represents the generic operation that transfers data from grid G^m to grid G^k. We will always transfer data between adjacent levels using suitably defined I_k^{k+1} and I_k^{k-1}. The operator I_k^{k+1} is a projection down one level in the processing cone. The values on the finer grid are interpolated from those on the coarser. The accuracy of a given interpolation algorithm can be measured by the **order of interpolation** which is a measure of how the error of interpolation is reduced as the mesh-size goes to zero ($h \rightarrow 0$). In particular nearest-neighbor interpolation, in which interpolated values are equal to the nearest known value, is of order $O(h^{-1})$. While linear interpolation is of order $O(h^{-2})$. Interpolation of order $O(h^{-n})$ is called **n-th order interpolation.**

The order of interpolation must be greater than or equal to the order of the differential equations [Brandt 77a, p.377]. The *order of a differential equation* is the highest degree of all differential operators in the equation. *The differential equations we are solving are all second order, hence we use bilinear projections.* In this case, the value at a fine pixel is computed from the values of the four nearest coarse pixels

by bilinear interpolation. With even pyramids, the bilinear projection masks are

$$I_k^{k+1} = Bil \triangleq \frac{1}{16}\begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix} \tag{60}$$

and the three 90^o rotations. (See Figure 4 on page 39.) The level sub- and superscripts have not been included in *Bil* since the operation is level independent.

The operator I_k^{k-1} is a reduction. The simplest is **injection** in which the father receives the value of the nearest son. For even pyramids there are four nearest sons, so strict injection cannot be done. The simplest even pyramid reduction is the average of the sons' values:

$$I_k^{k-1} = A_{2\times 2} \triangleq \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{61}$$

3.3.2 Relaxation

If we are to approximate the residual equation on a coarse grid then the residual r^k must have little high frequency component. *Relaxation sweeps* are used to smooth the error component. The *Gauss-Seidel* iterative algorithm is a common example. In this algorithm, during one sweep the points $\mathbf{x}^k$ in G^k are scanned one by one in some fixed order. At each point the old value of $u^k(\mathbf{x}^k)$ is replaced by a new value which satisfies Equation 55. Having completed such a sweep, the equations are not yet solved since they are coupled together. Standard (non multilevel) iterative algorithms apply a long sequence of such relaxation sweeps. They can be very slow since relaxation has little effect on smooth errors which can have very small local errors (small residuals) relative to their own magnitude. However, in multilevel methods relaxation sweeps are only used to smooth the error. This can be done in only a few iterations.

The Gauss-Seidel algorithm is not parallel since grid points are updated in a sequential order. New values at a given point will in general depend on new values at some neighboring points. The **Jacobi method** involves similar update equations at each node, but they are applied at all grid points in parallel. For the example

$-\Delta^k U^k = F^k$ with

$$\Delta^k = \frac{1}{h_k^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}$$

the grid point equation expands to:

$$-4U^k(i,j) + U^k(i-1,j) + U^k(i+1,j) + U^k(i,j-1) + U^k(i,j+1) = h_k^2 F^k(i,j)$$

which if solved for $U(i,j)$ gives

$$U^k(i,j) = \frac{1}{4}\Big((U^k(i-1,j) + U^k(i+1,j) + U^k(i,j-1) + U^k(i,j+1)\Big) - \frac{1}{4}h_k^2 F^k(i,j)$$

To use this last equation as an update equation we replace U^k by U^k_{new} on the left hand side and U^k by U^k_{old} on the right hand side. Switching back to the kernel notation the update equations are:

$$U^k_{new} \leftarrow \tilde{U}^k_{old} - \frac{1}{4}h_k^2 F^k$$

where

$$\tilde{U}^k_{old} \triangleq \frac{1}{4}\begin{bmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{bmatrix} * U^k_{old}$$

is a local averaging operator.

The above derivation can be formalized and applied to general systems of equations as follows. Let the system of equations be given by $A\mathbf{u} = \mathbf{b}$ and let $D = diag(A)$ be the diagonal matrix of A. Then $D^{-1}A\mathbf{u} = D^{-1}\mathbf{b}$ and $\mathbf{u} = (I - D^{-1}A)\mathbf{u} + D^{-1}\mathbf{b}$. Defining $B \triangleq I - D^{-1}A$ and $k \triangleq D^{-1}b$, we get the update equation:

$$\mathbf{u}_{new} \leftarrow B\mathbf{u}_{old} + \mathbf{k}$$

3.3.3 Cycle C Algorithm

A simple multilevel relaxation algorithm based on these ideas (called Cycle C [Brandt 77a]) is shown in Figure 56. 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

$$L^k V^k = r^k \tag{62}$$

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

$$r^k = I_{k+1}^k (r^{k+1} - L^{k+1} v^{k+1}) \tag{63}$$

The equation on G^k is thus defined in terms of the approximate solution on G^{k+1}. On the finest grid, the equation is the original one ($L^M V^M = r^M \leftrightarrow L^M U^M = F^M$) so we have:

$$r^M = F^M \quad , \quad v^M = u^M \tag{64}$$

When a correction v^k is arrived at, it is projected down to the finer level, adding on to the current estimate v^{k+1}. The projection is:

$$v^{k+1} \leftarrow v^{k+1} + I_k^{k+1} v^k \tag{65}$$

3.3.4 Convergence Measures

Convergence is measured using the **residual norm** — the Euclidean L_2 norm of the residual. It is defined as

$$\sqrt{\frac{1}{N} \sum (r^k - L^k u^k)^2}$$

where the sum is over all N grid points. (Note that it has the form of the *root-mean-square-error*.) 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 [Brandt 77a]). 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 "target" threshold is set, when

```
k ← M                                   ; start at finest level
r^k ← F^M                               ; initial right-hand-side
v^k ← u^M                               ; initial solution estimate
Until v^M has converged Do
    Begin
        v^k ← Relax|L^k · = r^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
            End if
        Else if convergence is slow Then
            If k > 0 Then
                k ← k − 1                       ; go up one level
                v^k ← 0                         ; initial estimate
                r^k ← I^k_{k+1}(r^{k+1} − L^{k+1} v^{k+1})   ; new right-hand-side
            End if
        End if
    End
```

Figure 56: Cycle C algorithm, multilevel relaxation

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 [Brandt 77a]). Brandt claims robustness of such algorithms to the extent that variations of these two parameter settings produce little qualitative change in performance.

3.3.5 Fixed Cycling Schemes

The two parametrized decisions that are used in controlling the cycles are global computations, i.e. they are computed from the current solution estimate over the entire grid. This is due to the fact that the decisions are based on the residual norm which is an integral (continuous case) or a summation (discrete case) over the entire domain. 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.*

3.3.6 Full Approximation Storage

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 [Brandt 77a, Brandt 77b]. *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 approximation u^k at level k is the sum of the correction v^k and its base approximation u^{k+1};

$$u^k = I_{k+1}^k u^{k+1} + v^k \qquad k = 0, 1, \ldots, M-1 \tag{66}$$

We can rearrange the general residual equation (Equation 56), in which L

may be non-linear, to get $L^k U^k = L^k u^k + r^k$. If we set u^k in this equation to an initial estimate computed from the finer level, i.e. $u^k_{init} = I^k_{k+1} u^{k+1}$ and use the r^k also obtained from the finer level in Equation 63, then substituting in we get

$$\begin{aligned} L^k U^k &= L^k u^k_{init} + r^k \\ &= L^k(I^k_{k+1} u^{k+1}) + I^k_{k+1}(r^{k+1} - L^{k+1} v^{k+1}) \end{aligned} \tag{67}$$

This is a new equation $L^k U^k = R^k$, with the right hand side $R^k = L^k u^k_{init} + r^k$. In this form R^k depends on r^k which in turn is computed from the residual r^{k+1} and correction v^{k+1} from level $k+1$. To complete the transformation to the FAS method, we must represent R^k in terms of R^{k+1} and u^{k+1} alone. This is done as follows. First, note that at level $k+1$ we have $u^{k+1} = u^{k+1}_{init} + v^{k+1}$ and $R^{k+1} = L^{k+1} u^{k+1}_{init} + r^{k+1}$. Apply L^{k+1} to the first equation and then subtract the result from the latter equation. This gives

$$R^{k+1} - L^{k+1} u^{k+1} = r^{k+1} - L^{k+1} v^{k+1} \tag{68}$$

Substituting Equation 68 into Equation 67, the correction equations can be rewritten as

$$L^k U^k = R^k \tag{69}$$

where

$$R^k = L^k(I^k_{k+1} u^{k+1}) + I^k_{k+1}(R^{k+1} - L^{k+1} u^{k+1}) \qquad k = 0, \ldots, M-1 \tag{70}$$

and

$$R^M = F^M \tag{71}$$

When a correction u^k is arrived at, it is projected down to the finer level, updating the current estimate u^{k+1}. The projection of the correction can be thought of as the addition of a correction term, i.e. $u^{k+1} \leftarrow u^{k+1} + I^{k+1}_k(v^k)$. This is done explicitly in the non-FAS algorithm. For the FAS algorithm v^k is given implicitly as $u^k - I^k_{k+1} u^{k+1}$ (see Equation 66). Thus, the projection is:

$$u^{k+1} \leftarrow u^{k+1} + I^{k+1}_k(u^k - I^k_{k+1} u^{k+1}) \tag{72}$$

$$\leftarrow I^{k+1}_k u^k + (I - I^{k+1}_k I^k_{k+1}) u^{k+1} \tag{73}$$

```
k ← M                                          ; start at finest level
R^k ← F^M                                      ; initial right-hand-side
v^k ← u^M                                      ; initial solution estimate
Until u^M has converged Do
    Begin
        u^k ← Relax[L^k · = R^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
            End if
        Else if convergence is slow Then
            If k > 0 Then
                k ← k - 1                                  ; go up one level
                u^k ← I^k_{k+1} u^{k+1}                    ; initial estimate
                R^k ← I^k_{k+1}(R^{k+1} - L^{k+1}u^{k+1}) + L^k u^k   ; new right-hand-side
            End if
        End if
    End
```

Figure 57: Cycle C/Full Approximation Storage, multilevel relaxation

This is the FAS version of Equation 65. In the second form of the right hand side, the operator $I - I_k^{k+1} I_{k+1}^k$ is a high-pass filter similar to that used to build low-pass pyramids. Thus, we see that *the projection updates u^{k+1} by replacing its low frequency component by a projection of u^k.*

For linear problems equations 62–64 are equivalent to 69–71. One advantage of the FAS method is that equations 69–71 apply equally well to nonlinear problems [Brandt 77a, 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. The FAS generalization of the Cycle C algorithm is shown in Figure 57.

3.4 Normalized Coordinate Systems

We will use normalized coordinate systems at each level of the processing cone to make the update operators independent of the mesh size for the given level. This idea was introduced in Chapter V, Section 3.2.2. The **base coordinate system** is determined by the finest level (L) of the processing cone. The interpixel spacing at the finest level is 1 unit in the base coordinate system. The interpixel spacing at a coarser level k, when measured in the base system is $\rho^{L-k} = (h_{L-k}/h_{L-k+1}) \times \ldots (h_{L-2}/h_{L-1}) \times (h_{L-1}/h_L)$. In the **normalized coordinate system** at a given level, the distance between two adjacent pixels is 1 unit. Image operators which involve finite differences can be then be computed without using h_k terms, h_k being the distance between pixels. Intra-level computations and comparisons must take into account the difference in the grid spacings as given by the ratio $\rho \triangleq h_{k-1}/h_k$. This adjustment is described below.

If regular image data (intensity, brightness, etc.) is moved in the cone, i.e. projected or reduced, then no adjustment for the normalized coordinate systems is necessary. However, if image gradient data is moved, then adjustments must be made. Whether such adjustments must be made and what they are is completely determined by the *order* of the quantity we are moving between levels. This is now defined.

Image operators that approximate spatial derivatives involve the mesh size h_k. The order of the derivative corresponds to the power of h_k which appears in the denominator of the operator (in the non-normalized form). See, for example, the discrete Laplacian on page 192, in which the term $1/h_k^2$ appears. Now, consider the units we use to measure the quantities being manipulated. Let I represent the unit of image data and L the unit of length. Application of a differential operator results in a change of units. For example, if a first difference is applied to image data, then the resultant units are I/L. If the Laplacian is applied to the original data, then the resultant data is in units of I/L^2. The change in units is directly related to the order of the differential operator.

We define the **order** of a quantity as the negative of the exponent of its length

units. We also define the **order** of a finite difference operator as the order of the differential operator it approximates. As noted above this is equal to the exponent of h_k in the non-normalized form. When a finite difference operator is applied, the order of the result is equal to the sum of the order of the operator and the order of the operand.

When data of order n is reduced, an extra factor of ρ^n must be applied. Let u^k be measured in units of I/L^n and let $\tilde{u}^k$ represent the same quantity in the normalized coordinate system. Then the relationship between these two measures is $\tilde{u}^k = h_k^n u^k$ where h_k is the mesh-size (i.e. the length per pixel-spacing). If A is a local averaging reduction operator, then reduction of u^k would be given by $u^{k-1} \leftarrow Au^k$. If normalized coordinates are used at levels k and $k-1$, then by substitution $\tilde{u}^{k-1}/h_{k-1}^n \leftarrow A(\tilde{u}^k/h_k^n)$ and the reduction becomes $\tilde{u}^{k-1} \leftarrow (h_{k-1}/h_k)^n A\tilde{u}^k = \rho^n A\tilde{u}^k$.

For example, if we reduce order 2 data with the 4×4 averaging operator (Equation 61 on page 191) the net reduction operator is a simple 4×4 sum.

Projections involve an adjustment in the opposite direction. When data of order n is projected, an extra factor of ρ^{-n} must be applied. The demonstration of this is analogous to that for reductions. These results are now summarized.

Reduction and Projection of Order n data

If data u of order n is represented in a pyramid using normalized coordinates at each level, then it is reduced using

$$u^{k-1} \leftarrow \rho^n A u^k \tag{74}$$

and it is projected using

$$u^{k+1} \leftarrow \frac{1}{\rho^n} P u^k \tag{75}$$

where A and P are (non-normalized) reduction and projection operators respectively and we have dropped the tildes ($\tilde{}$).

4 An Example: Interpolation

4.1 First Order Smoothing: Laplace's Equation

Consider the problem of interpolating data into a region in such a way that (1) the values are predefined along a boundary curve, and (2) the data is as "smooth" as possible. Smoothness can be defined many ways (see the examples in Section 2.1). We will use here simple **first order smoothness**, viz. minimum total squared gradient magnitude. Thus, the problem is to find a solution $U(x,y)$ which minimizes the integral

$$\iint_{\mathcal{A}} \|\nabla U\|^2 \, dxdy \tag{76}$$

and which satisfies the boundary condition

$$U(x,y) = C(x,y) \quad \text{for} \quad (x,y) \in \partial\mathcal{A} \tag{77}$$

where $\partial\mathcal{A}$ is the boundary of $\mathcal{A}$. The Euler equation for this variational problem is Laplace's equation:

$$\Delta U(x,y) = 0 \quad \text{on the domain} \quad \mathcal{A} \tag{78}$$

This specific problem is given as an example in Figure 55.

4.2 Discrete Representation and Computation

4.2.1 Boundary Handling: The *Mask* Plane

The example experiment computes the solution to a Dirichlet type problem: the solution values are known along the boundary at the finest level. We refer to these locations as the *fixed points* since their values remain unchanged. At coarser levels, grid points which have fixed points as descendants, become fixed points at their level. The fixed points are represented by a *Mask* pyramid. At each level of this pyramid a *mask* plane contains 1's at the fixed points and 0's elsewhere. The finest level of the *Mask* pyramid is given at the start. The remainder of the pyramid is built by OR-ing up in the processing cone starting with the finest level. That is, the

value of a father is set to 1 if any of its sons is set to 1. In the following experiment, at each level only the border pixels are masked to 1.

4.2.2 Normalized Coordinate Systems

We use normalized coordinate systems at each level of the processing cone to make the update operators independent of the mesh size for the given level. This idea was presented in Section 3.4. The data passed between levels are the solution estimates u which are of order 0, and the right-hand-sides which are of order 2.

4.2.3 Relaxation

The particular relaxation operator used is determined by the discrete approximation to the Laplacian that we chose and by the iterative relaxation scheme. The following experiment uses the simple five-point discrete Laplacian operator Δ_1:

$$L^k = \Delta_1^k = \frac{1}{h_k^2}\Delta_1 = \frac{1}{h_k^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}$$

The relaxation scheme is Gauss-Seidel relaxation. One iteration of Gauss-Seidel relaxation based on the above discrete Laplacian is computed by the following update equation:

$$\begin{aligned} u_{s+1}(i,j) \leftarrow \frac{1}{4}\big(&u_{s+1}(i-1,j) + u_{s+1}(i,j-1) \\ &+ u_s(i+1,j) + u_s(i,j+1)\big) - \frac{1}{4}\tilde{r} \end{aligned}$$

where s is the iteration number and $\tilde{r}^k \triangleq h_k^2 r^k$. The right-hand-side data r^k is of order 2 (see Section 3.4). In the normalized coordinate system at level k we use $\tilde{r}^k$. This allows the update equation to be independent of the resolution, having no h_k term.

At each level there will be fixed points, where the u must remain constant in value. Thus, the following equation is used in updating u at each point (i,j):

$$u_{s+1}(i,j) \leftarrow \begin{cases} \frac{1}{4}\big(u_{s+1}(i-1,j) + u_{s+1}(i,j-1) & \text{if } mask(i,j) = 0 \\ \qquad + u_s(i+1,j) + u_s(i,j+1)\big) - \frac{1}{4}\tilde{r} & \\ u_s(i,j) & \text{if } mask(i,j) = 1 \end{cases}$$

4.2.4 Projection

The interpolation operators used for projection and reduction are bilinear interpolation $I_{k-1}^{k} = Bil$ (Equation 60) and average-of-the-sons $I_{k}^{k-1} = A_{2\times 2}$ (Equation 61). Using Equation 72 on page 196, the projection of corrections is then given by

$$u^k \leftarrow \begin{cases} u^k + Bil(u^{k-1} - A_{2\times 2}u^k) & \text{if } mask^k = 0 \\ u^k & \text{if } mask^k = 1 \end{cases}$$

The values at fixed points (as indicated by $mask$ values of 1) are not updated.

4.2.5 Reduction

Coarse solution estimates are passed up using:

$$u^{k-1} \leftarrow A_{2\times 2}u^k$$

After which the coarse residuals are given by:

$$r^{k-1} \leftarrow A_{2\times 2}(r^k - \Delta^k u^k) + \Delta^{k-1}u^{k-1}$$

This is Equation 70 from page 196. Now we convert this equation to the normalized form. If we multiply through by h_{k-1}^2 we get:

$$h_{k-1}^2 r^{k-1} \leftarrow \frac{h_{k-1}^2}{h_k^2} A_{2\times 2}\big(h_k^2 r^k - h_k^2 \Delta^k u^k\big) + h_{k-1}^2 \Delta^{k-1} u^{k-1}$$

which by simple substitution gives:

$$\begin{aligned} \tilde{r}^{k-1} &\leftarrow \rho^2 A_{2\times 2}(\tilde{r}^k - \Delta_1 u^k) + \Delta_1 u^{k-1} \\ &\leftarrow \rho^2 A_{2\times 2}\tilde{r}^k - \rho^2 A_{2\times 2}\Delta_1 u^k + \Delta_1 u^{k-1} \end{aligned}$$

where $\tilde{r}^{k-1} \triangleq h_{k-1}^2 r^{k-1}$ and $\tilde{r}^k \triangleq h_k^2 r^k$. This form of the reduction is used since the data is represented over normalized coordinate systems at each level. Note that the discrete Laplacians do not involve h_k or h_{k-1} and the local averaging operator has been adjusted for the fact that $\tilde{r}$ and Δu are of order 2. For our even pyramids we have $\rho = h_{k-1}/h_k = 2$, and the local neighborhood operators are:

$$4A_{2\times 2} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

and

$$4A_{2\times 2}\Delta_1 = \begin{bmatrix} & 1 & 1 & \\ 1 & -2 & -2 & 1 \\ 1 & -2 & -2 & 1 \\ & 1 & 1 & \end{bmatrix}$$

4.3 Performance Measures

4.3.1 Work Units

On a sequential computer, which processes the algorithm one pixel at a time, iterations at finer levels have a higher computational cost, proportional to the larger grid size. In this case, iteration count is not a good measure of overall computational cost. Brandt introduced **work units** as a simple but accurate measure of the computational cost of multilevel algorithms. The cost of iterations at each level are measured in work units. The cost at a given level will be proportional to the number of nodes in the grid at that level. At the finest level M, we define the cost of one iteration to be one work unit, that is $w_M \triangleq 1$. At level k, one iteration costs $w_k \triangleq (1/4)^{M-k}$ work units.

If multilevel relaxation is run in an actual cellular array computer or in a processing cone computer, then all nodes in a given level are updated at the same time. Thus, the cost of an iteration is constant and not proportional to the number of grid points involved. In this case, the iteration count does provide an accurate measure of computational cost, and work units as defined above are not an appropriate measure. We will show convergence in terms of both iteration count and

work units in the first examples to illustrate the work units measure. Thereafter, in the later optic flow experiments, we will only measure convergence in terms of iteration count, given our emphasis on parallel processing cone computations.

4.3.2 Convergence

In Section 3.3.4, the *residual norm* measure of convergence was defined and the **convergence rate** was defined as the ratio of consecutive residual norms from one iteration to the next:

$$C_R \triangleq \frac{Err^{k+1}}{Err^k}$$

We also define the **convergence factor** as the reduction in the Log (base 10) of the residual norm from one iteration to the next:

$$C_F \triangleq \log(Err^k) - \log(Err^{k+1})$$

Clearly, $C_F = -\log(C_R)$. In the semi-log plots of residual norm shown for the following experiments, C_F corresponds to the slope of the graphs (or more precisely, the slope of tangents to the graphs).

The convergence factor and the convergence rate measure how much the residual norm is being reduced from iteration to iteration. A complementary way of measuring performance is the number of iterations it takes to get a certain amount of reduction in the residual norm. For this measure, we define the **convergence time** as the number of iterations it takes to reduce the residual norm an order of magnitude at the given rate of convergence. This is simply given by

$$C_T \triangleq \frac{1}{C_F}$$

Since C_F measures reduction in the Log per iteration, $1/C_F$ measures iterations per reduction in the Log by 1. This corresponds to reduction by an order of magnitude.

4.4 Experiments

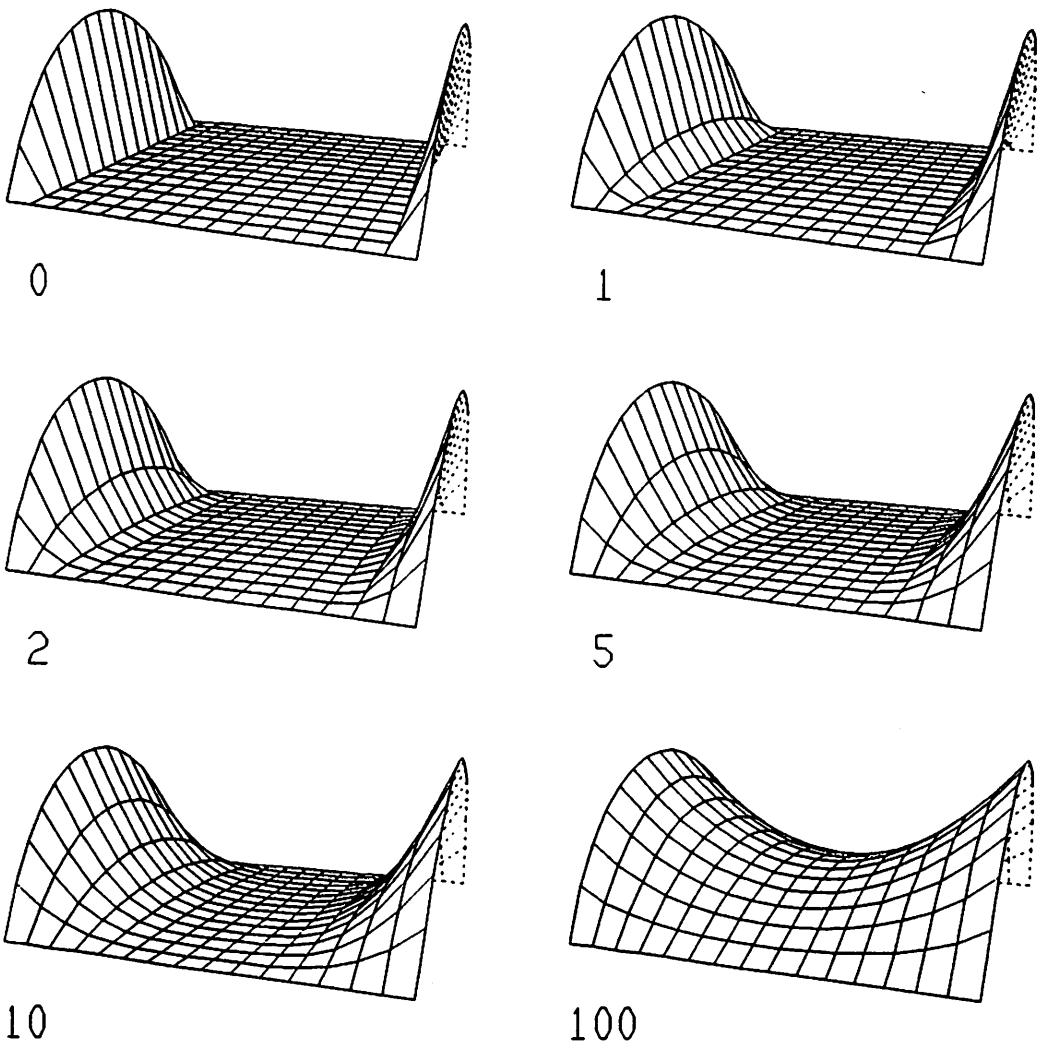


Figure 58: Single-level relaxation on Laplace's equation
Initial data (iteration 0) and iterations 1,2,5,10, and 100.

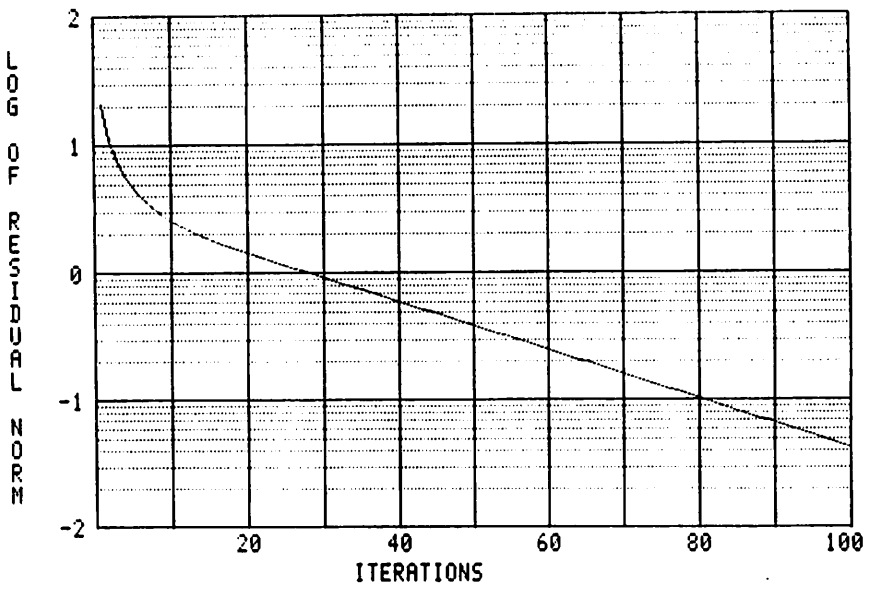


Figure 59: Single-level relaxation on Laplace's equation: error graph

Semi-log graph of residual norm vs. iteration number

Figures 58 and 59 show the results of performing Gauss-Seidel relaxation on a finite difference approximation to equation 78 (as formulated in the lower right hand box of Figure 55). Intermediate stages in the iteration are shown as surface plots in Figure 58 with the initial solution estimate (all zeroes in the interior of $\mathcal{A}$; $\mathcal{A} - \partial\mathcal{A}$) labeled as iteration 0. The residual norm is plotted against iteration number in Figure 59. 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 Figures 60, 61, and 62. In Figure 60, surface plots show various stages of the computation at different levels. The residual norm is plotted in Figure 61, and also in Figure 62 where it is superimposed with the results of single-level relaxation.

In Figure 61a, the residual norm is plotted against iteration number, while in Figure 61b it is plotted against work units. In both of these graphs each iteration

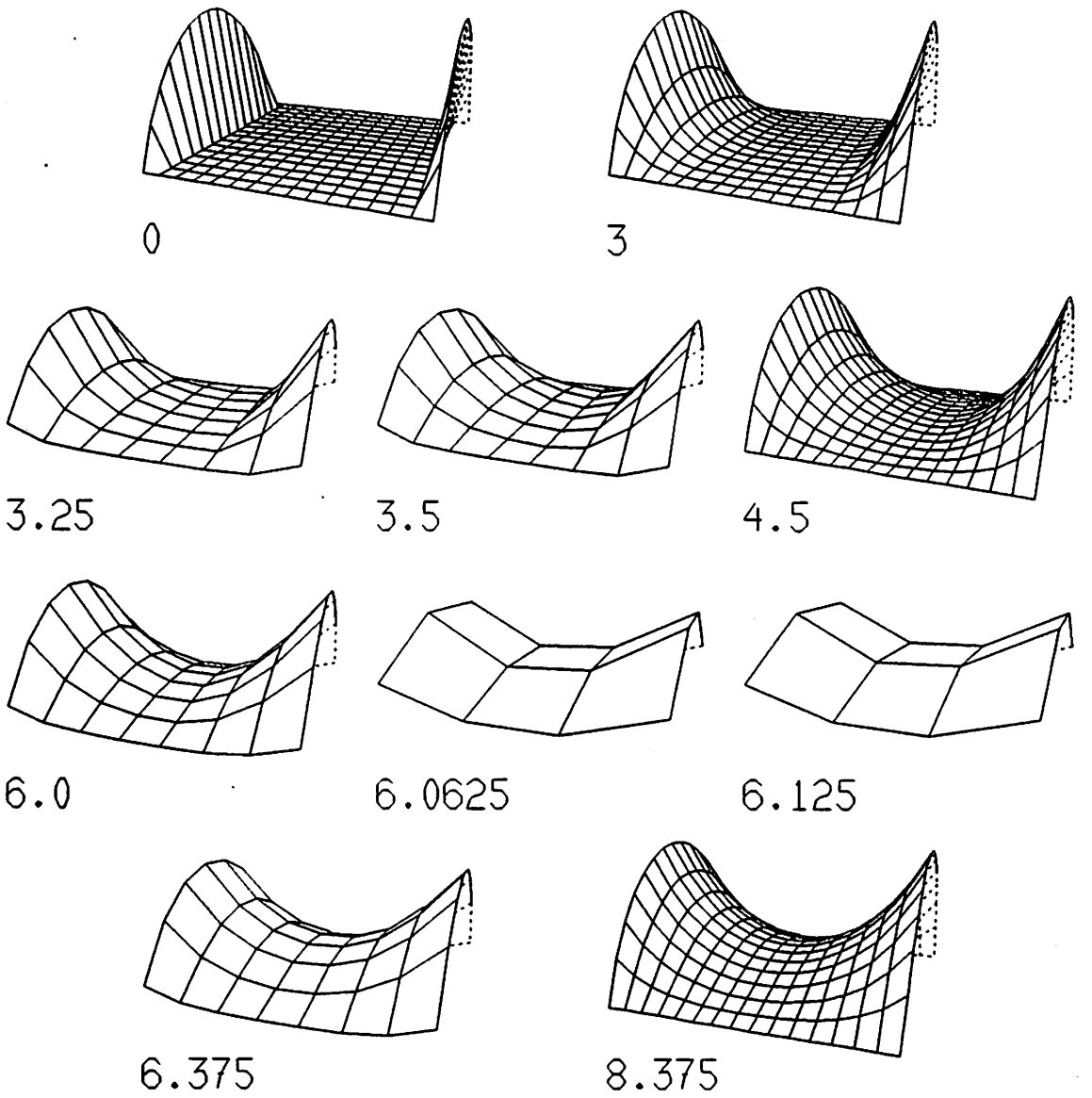


Figure 60: Multilevel relaxation on Laplace's equation

Selected intermediate iterations labeled by work number

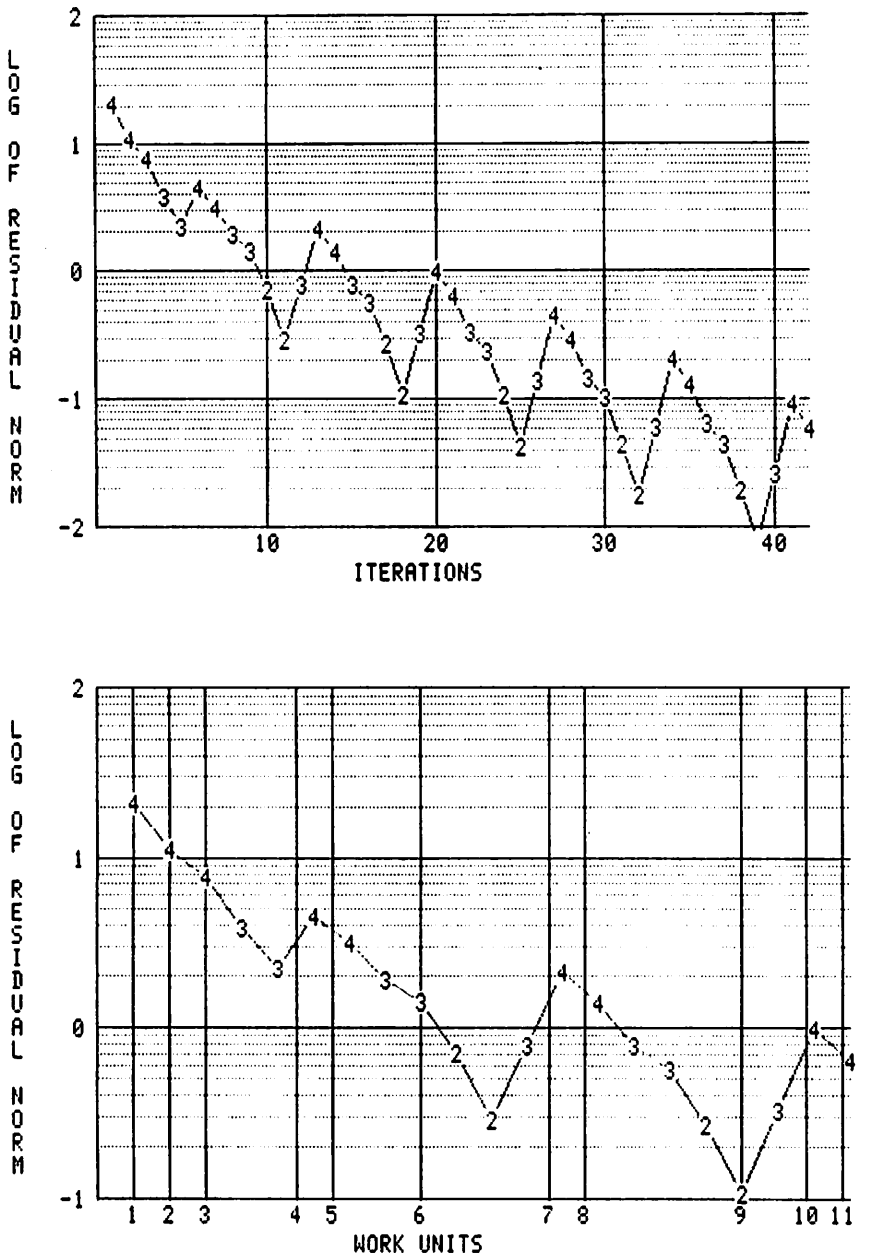


Figure 61: Multilevel relaxation on Laplace's equation: error graphs

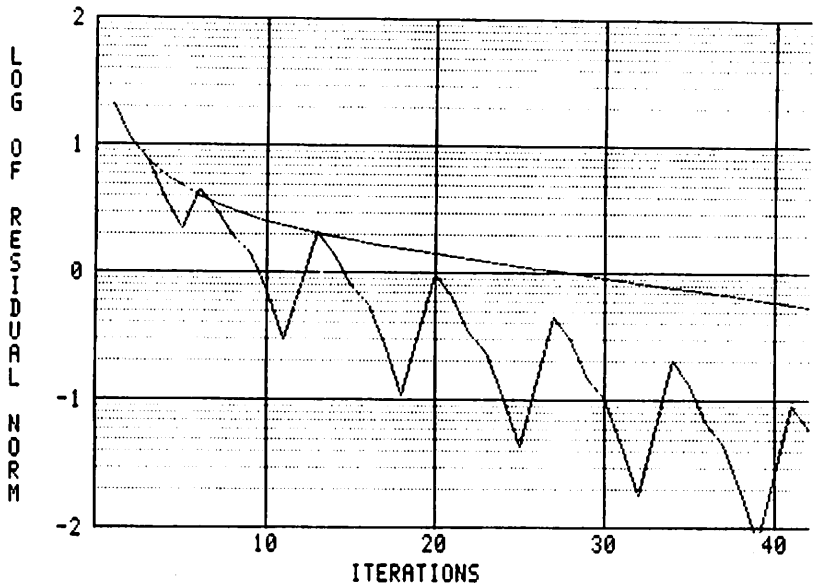


Figure 62: Multilevel vs. single-level relaxation

This graph compares the results of the experiments shown in Figure 59 and Figure 61.

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.

In the single-level experiment (Figures 59 and 62), at iteration 40, the convergence rate is .958 and the convergence time is 53.1 iterations. In the multilevel experiment we can measure convergence in terms of work units or iterations. This has been done using the residual norms at iterations 14 and 21 in Figure 61a, with corresponding work units of 8.375 and 11.25 respectively in Figure 61b. In terms of work units, the convergence rate is .761 and the convergence time is 8.44 work units. In terms of iterations, the convergence rate is .894 and the convergence time is 20.6 iterations.

This last result is quite important in that it shows us that *while multilevel algorithms greatly speed up relaxation on sequential machines, they also provide*

significant gains on parallel computers such as cellular arrays or processing cones. This is clearly seen in Figure 62 — a comparison of single and multilevel relaxation. Residual norm for both experiments is plotted against iteration number. The multilevel algorithm clearly converges faster.

5 Multilevel Optic Flow Computation

5.1 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 $F(x, y, t)$. In Chapter III, Section 4.3.4, Horn & Schunck's variational principle was described and in Section 2.1 it was included as one of the four examples of the uses of optimization methods in low-level vision. It includes a first order smoothness constraint on u and v and the velocity line constraint. The resulting variational problem is to find the optic flow field satisfying Equation 53 on page 180. The equivalent PDE system (Euler's equations) is [Courant & Hilbert 53, Section IV.3.4]

$$\alpha^2 \Delta u - F_x^2 u - F_x F_y v = F_x F_t \tag{79a}$$

$$\alpha^2 \Delta v - F_x F_y u - F_y^2 v = F_y F_t \tag{79b}$$

We can represent this in the form $LU = F$ as follows:

$$\begin{bmatrix} \alpha^2\Delta - F_x^2 & -F_x F_y \\ -F_x F_y & \alpha^2\Delta - F_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} F_x F_t \\ F_y F_t \end{bmatrix} \tag{80}$$

This elliptic system of PDE's 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 and y).

5.2 Discrete Representation and Computation

5.2.1 Normalized Coordinate Systems

Again, we use normalized coordinate systems at each level of the processing cone to make the update operators independent of the mesh size for the given level. If we examine the equations in the previous section we see the following quantities: (1) optic flow estimates (u, v) of order -1 (since it is measured in units of length per time); (2) spatial derivatives F_x and F_y of order 1; and (3) a time derivative F_t of order 0. The L operator is of order 2, as given by each of its elements, viz. the Laplacian and the products of the spatial derivatives. The order of the equations themselves is 1 (L applied to (u, v)) and thus this is the order of the residual. In the course of a multilevel processing, the optic flow estimates and the residuals will be transferred between levels. Reductions and projections must take their respective orders into account.

Optic flow vectors in normalized coordinates will be represented as $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v})$. They are related to the unnormalized form by $h_k\tilde{\mathbf{u}} = (h_k\tilde{u}, h_k\tilde{v}) = (u, v) = \mathbf{u}$.

Spatial gradients in normalized coordinates will be represented as $(\tilde{F}_x, \tilde{F}_y)$. They are related to the unnormalized form by $(\tilde{F}_x/h_k, \tilde{F}_y/h_k) = (F_x, F_y)$.

5.2.2 Relaxation

The particular relaxation operator used is determined by the discrete approximation to the Laplacian that is used and by the iterative relaxation scheme. We choose Jacobi relaxation because it is a parallel updating method. The discrete five-point Laplacian Δ_1 used in the interpolation example (page 201) is not a good choice for use with Jacobi relaxation because elimination of the highest frequency error is not guaranteed. This is shown by Local-Mode-Analysis A.1 in Appendix A. (Local mode analyses are defined in Section 7.) Local-Mode-Analysis A.2 in Appendix A shows that the following discrete Laplacian does provide good convergence for mul-

tilevel relaxation using the Jacobi scheme.

$$\Delta_2^k = \frac{1}{h_k^2}\Delta_2 = \frac{1}{h_k^2}\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This discrete Laplacian is used in all of the following experiments. The discrete version of Equation 80 is then

$$L^k \mathbf{u}^k = \mathbf{r}^k \tag{81}$$

where

$$L^k = \begin{bmatrix} \frac{\alpha^2}{h_k^2}\Delta_2 - F_x^2 & -F_xF_y \\ -F_xF_y & \frac{\alpha^2}{h_k^2}\Delta_2 - F_y^2 \end{bmatrix} \tag{82}$$

$\mathbf{u}^k = [u^k\ v^k]^T$ and $\mathbf{r}^k = [r_1^k\ r_2^k]^T = [F_xF_t\ F_yF_t]^T$.

The Jacobi update equations are derived as follows. The discrete Laplacians $\Delta_2 u$ and $\Delta_2 v$ are represented as $3(\hat{u} - u)$ and $3(\hat{v} - v)$ where $\hat{u}$ and $\hat{v}$ are local averages given by:

$$\hat{u} = \frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix} * u \qquad \hat{v} = \frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix} * v \tag{83}$$

The expressions $3(\hat{u} - u)$ and $3(\hat{v} - v)$ are substituted into Equations 81, 82 and the equations solved for u and v. The resultant equations are used as Jacobi update equations by specifying that at the $(s+1)$-st update iteration, a new value (u_{s+1}, v_{s+1}) at the central pixel is computed from the local average $(\hat{u}_s, \hat{v}_s)$ of old values at the neighbors. The following update equation results:

$$u_{s+1} \leftarrow \hat{u}_s - F_x[F_x\hat{u}_s + F_y\hat{v}_s + F_t]/(3\alpha^2 + F_x^2 + F_y^2) \tag{84a}$$

$$v_{s+1} \leftarrow \hat{v}_s - F_y[F_x\hat{u}_s + F_y\hat{v}_s + F_t]/(3\alpha^2 + F_x^2 + F_y^2) \tag{84b}$$

We convert to the normalized form by substituting in the normalized form of the flow vectors and the spatial gradients, viz. $(h_k\tilde{u}, h_k\tilde{v}) = (u, v)$ and

$(\tilde{F}_x/h_k, \tilde{F}_y/h_k) = (F_x, F_y)$ and dividing the equations by h_k to get:

$$\tilde{u}_{s+1} \leftarrow \hat{u}_s - \tilde{F}_x[\tilde{F}_x\hat{u}_s + \tilde{F}_y\hat{v}_s + F_t]/(3\alpha^2 + \tilde{F}_x^2 + \tilde{F}_y^2) \quad (85a)$$

$$\tilde{v}_{s+1} \leftarrow \hat{v}_s - \tilde{F}_y[\tilde{F}_x\hat{u}_s + \tilde{F}_y\hat{v}_s + F_t]/(3\alpha^2 + \tilde{F}_x^2 + \tilde{F}_y^2) \quad (85b)$$

where $\hat{u}$ and $\hat{v}$ are changed in meaning to local averages of $\tilde{u}$ and $\tilde{v}$ respectively.

5.2.3 Projection

The interpolation operators used for projection and reduction are bilinear interpolation $I_{k-1}^k = Bil$ (Equation 60) and average-of-the-sons $I_k^{k-1} = A_{2\times 2}$ (Equation 61). Substituting into Equation 72 on page 196, corrections are projected using

$$\mathbf{u}^k \leftarrow \mathbf{u}^k + Bil(\mathbf{u}^{k-1} - A_{2\times 2}\mathbf{u}^k)$$

Using flow vectors represented in normalized coordinates, the projection equation is

$$\tilde{\mathbf{u}}^k \leftarrow \tilde{\mathbf{u}}^k + 2Bil(\tilde{\mathbf{u}}^{k-1} - \frac{1}{2}A_{2\times 2}\tilde{\mathbf{u}}^k)$$

The scale factors 2 and $\frac{1}{2}$ for projection and reduction are used because $\mathbf{u}$ is of order -1. For example, a velocity or displacement of 2 pixel positions at a given level, when reduced, is equivalent to 1 pixel position at the coarser level.

5.2.4 Reduction

Coarse solution estimates are passed up using:

$$\tilde{\mathbf{u}}^{k-1} \leftarrow \frac{1}{2}A_{2\times 2}\tilde{\mathbf{u}}^k \quad (86)$$

The coarse residuals are given by:

$$\mathbf{r}^{k-1} \leftarrow A_{2\times 2}(\mathbf{r}^k - L^k\mathbf{u}^k) + L^{k-1}\mathbf{u}^{k-1} \quad (87)$$

where L^k and L^{k-1} are defined by Equation 82. This is Equation 70 from page 196. If we multiply through by h_{k-1} we get:

$$h_{k-1}\mathbf{r}^{k-1} \leftarrow \frac{h_{k-1}}{h_k}A_{2\times 2}\left(h_k\mathbf{r}^k - h_kL^k\mathbf{u}^k\right) + h_{k-1}L^{k-1}\mathbf{u}^{k-1}$$

which is finally converted to the level-independent form of the update equation used ($h_{k-1}/h_k = 2$):

$$\tilde{\mathbf{r}}^{k-1} \leftarrow 2A_{2\times 2}(\tilde{\mathbf{r}}^k - \tilde{L}^k\tilde{\mathbf{u}}^k) + \tilde{L}^{k-1}\tilde{\mathbf{u}}^{k-1} \tag{88}$$

$$\leftarrow 2A_{2\times 2}\tilde{\mathbf{r}}^k - 2A_{2\times 2}\tilde{L}^k\tilde{\mathbf{u}}^k + \tilde{L}^{k-1}\tilde{\mathbf{u}}^{k-1} \tag{89}$$

where the normalized right-hand-sides are $\tilde{\mathbf{r}}^{k-1} = h_{k-1}\mathbf{r}^{k-1}$ and $\tilde{\mathbf{r}}^k = h_k\mathbf{r}^k$, and the normalized operator is given by

$$\tilde{L}^k = h_k^2 L^k = \begin{bmatrix} \alpha^2\Delta_2 - \tilde{F}_x^2 & -\tilde{F}_x\tilde{F}_y \\ -\tilde{F}_x\tilde{F}_y & \alpha^2\Delta_2 - \tilde{F}_y^2 \end{bmatrix} \tag{90}$$

Equation 89 can be broken into its component equations as follows:

$$\begin{aligned} \tilde{r}_1^{k-1} \leftarrow\ & 2A_{2\times 2}\tilde{r}_1^k - 2A_{2\times 2}(\alpha^2\Delta_2 u^k - \tilde{F}_x^2 u^k - \tilde{F}_x\tilde{F}_y v^k) \\ & + (\alpha^2\Delta_2 u^{k-1} - \tilde{F}_x^2 u^{k-1} - \tilde{F}_x\tilde{F}_y v^{k-1}) \end{aligned} \tag{91a}$$

$$\begin{aligned} \tilde{r}_2^{k-1} \leftarrow\ & 2A_{2\times 2}\tilde{r}_2^k - 2A_{2\times 2}(\alpha^2\Delta_2 v^k - \tilde{F}_y^2 v^k - \tilde{F}_x\tilde{F}_y u^k) \\ & + (\alpha^2\Delta_2 v^{k-1} - \tilde{F}_y^2 v^{k-1} - \tilde{F}_x\tilde{F}_y u^{k-1}) \end{aligned} \tag{91b}$$

Reduction is given by Equations 86 and 91.

6 Experiments

In this section we demonstrate the increased convergence rate obtained by multilevel relaxation over that for single-level relaxation in computing optic flow.

6.1 Single-level Relaxation

The two frames of test data for the first experiment are shown in Figure 63. Also shown is the gradient data, with F_x and F_y displayed as a vector field. In the first experiment the motion is translational: 1/2 pixel to the right and 1 pixel up. One percent uniform noise has been added to both test images and is uncorrelated between frames. This test data is designed to be as close as possible to that used

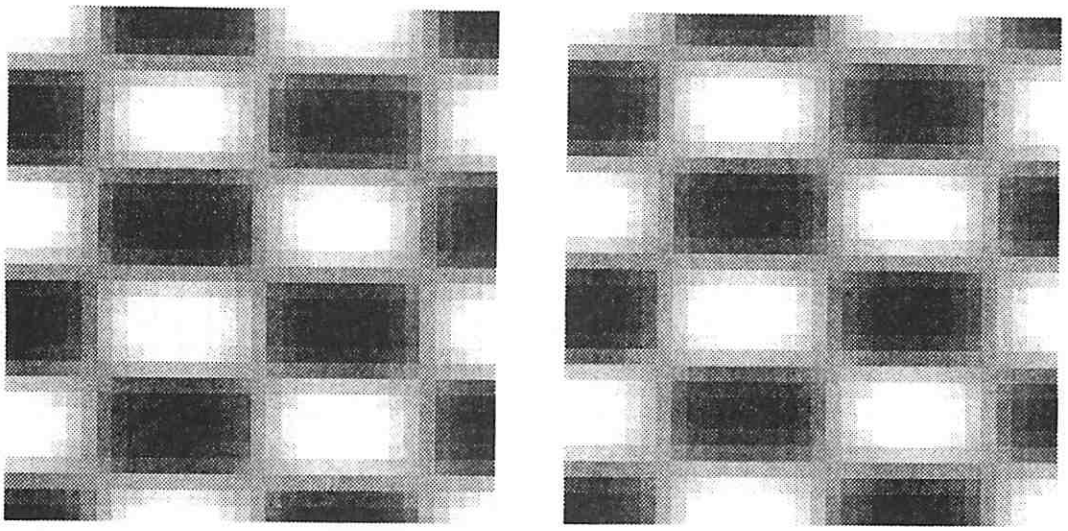


Figure 63: Optic flow test data

(Left) First frame, a cross product of sinusoids with 1% uniform noise added.
(Right) Second frame, the first frame displaced 1 pixel up, 1/2 pixel to the right.

in [Horn & Schunck 81]. The single grid Horn & Schunck algorithm is shown in Figures 64 and 65. Figure 64 shows a portion of the image plane at various stages in the iteration. The initial estimate (shown as iteration 0) is given by the edge flow vectors computed from equation 92.

$$\begin{pmatrix} -F_x F_t & -F_y F_t \\ F_x^2 + F_y^2 & , \; F_x^2 + F_y^2 \end{pmatrix} \tag{92}$$

An error graph is plotted in Figure 65. In Figure 65a, the residual norm is plotted against iteration number. In Figure 65b, the mean error magnitude is plotted. This is defined as the mean of the length of the error vector at each pixel, where the length is given by the Euclidean norm, and the error vector is the difference between the computed flow vectors and the ideal flow vector $(D_{row}, D_{col}) = (-1.0, 0.5)$. Note that the ideal flow is NOT equal to the actual solution to the discrete representation of the optic flow equations. There are two reasons for this. First, there is an inherent error involved in a discrete representation due to approximations in the finite difference operators and the finite precision arithmetic. This is known as *truncation*

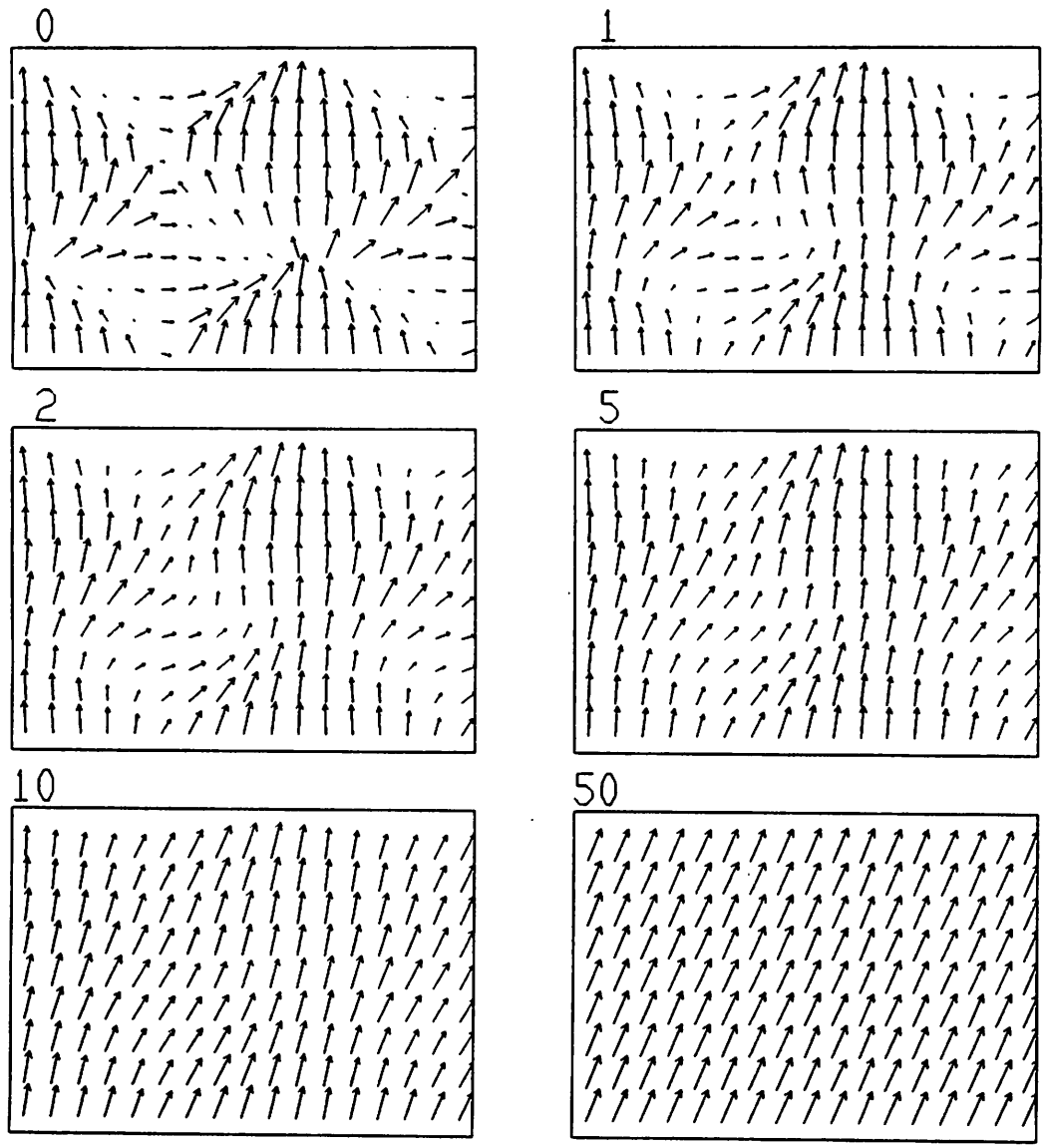


Figure 64: Single-level optic flow computation

Iterations 0,1,2,5,10 and 50. Only a portion of the image plane is shown.

error. Secondly, we have added noise to the data, thereby further corrupting the approximate coefficients derived from the spatial and temporal gradients. In any case, we still see the computed flow vector coming very close to the ideal, with error on the order of .015 pixels. (For further accuracy statistics, see the single-level vs. multilevel comparison in Table 4.)

6.2 Multilevel Relaxation

The results of the multilevel algorithm are shown in the next two figures. Again the Cycle C/FAS hierarchical control strategy is used since it provides a hierarchy of solution fields at the varying levels of resolution of the cone. While reductions to levels coarser than level 4 were allowed, that was the highest level that the algorithm chose to go to. In Figure 66 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. Figure 67a plots residual norm vs. iteration. By the 22nd iteration, little further reduction in the residual norm is seen. Figure 67b plots mean error magnitude vs. iteration, showing accuracy at least as good as that obtained by single-level relaxation after many more iterations. (The mean error magnitudes in the error graphs are given in the "non-normalized" base coordinate system to aid in comparison)

Table 4 compares the actual computed optic flow versus the expected value of $(-1, .5)$ at each pixel. The tables show that after 22 iterations, the multilevel algorithm has reached a solution which, in terms of relative error, is more than twice as good as that reached by the single-level method in 50 iterations. Note, also, that this comparison is based on iteration count. A comparison based on work units, the relevant measure for the case of serial computation, would clearly be even better.

6.3 Non-Translational Motion

Next we present an experiment using a rotational motion field. Again, this test data is designed to be as close as possible to that used in [Horn & Schunck 81].

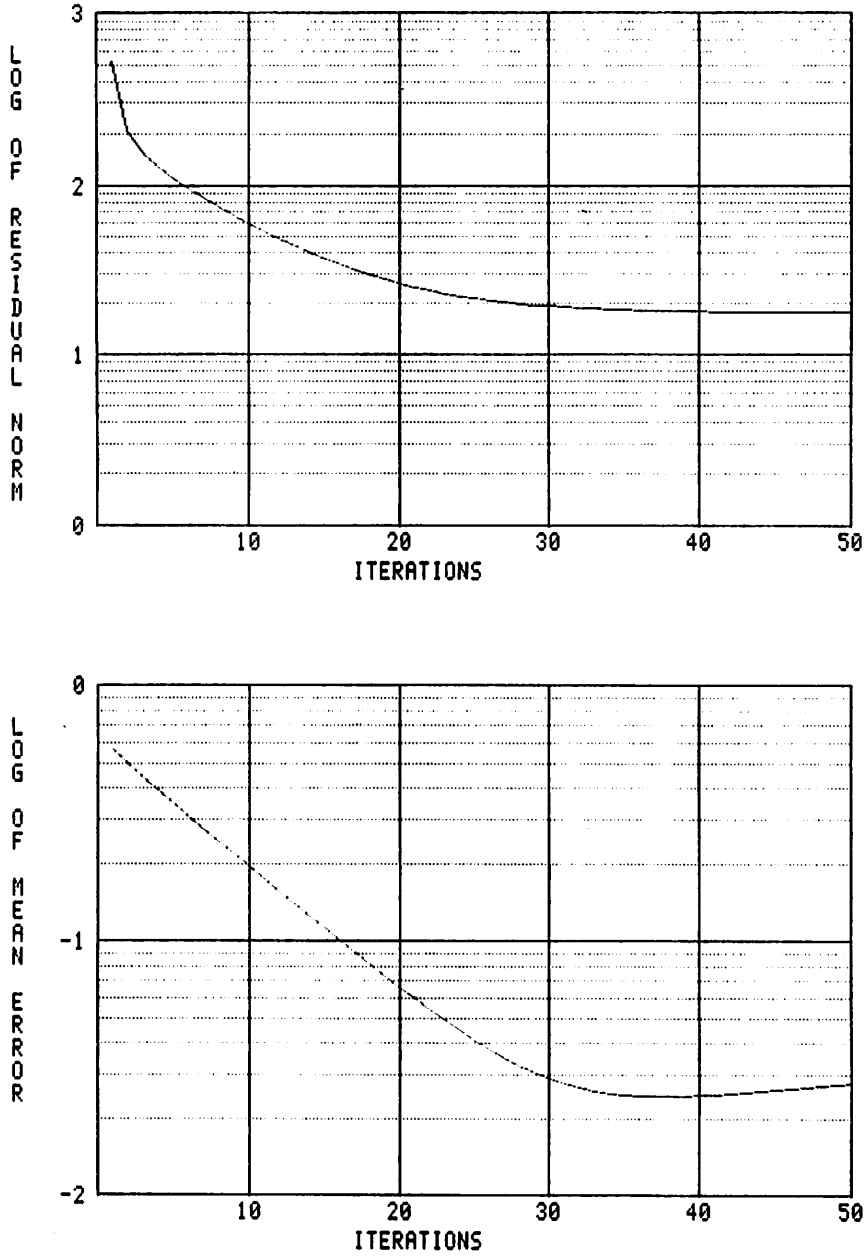


Figure 65: Single-level optic flow computation: error graphs

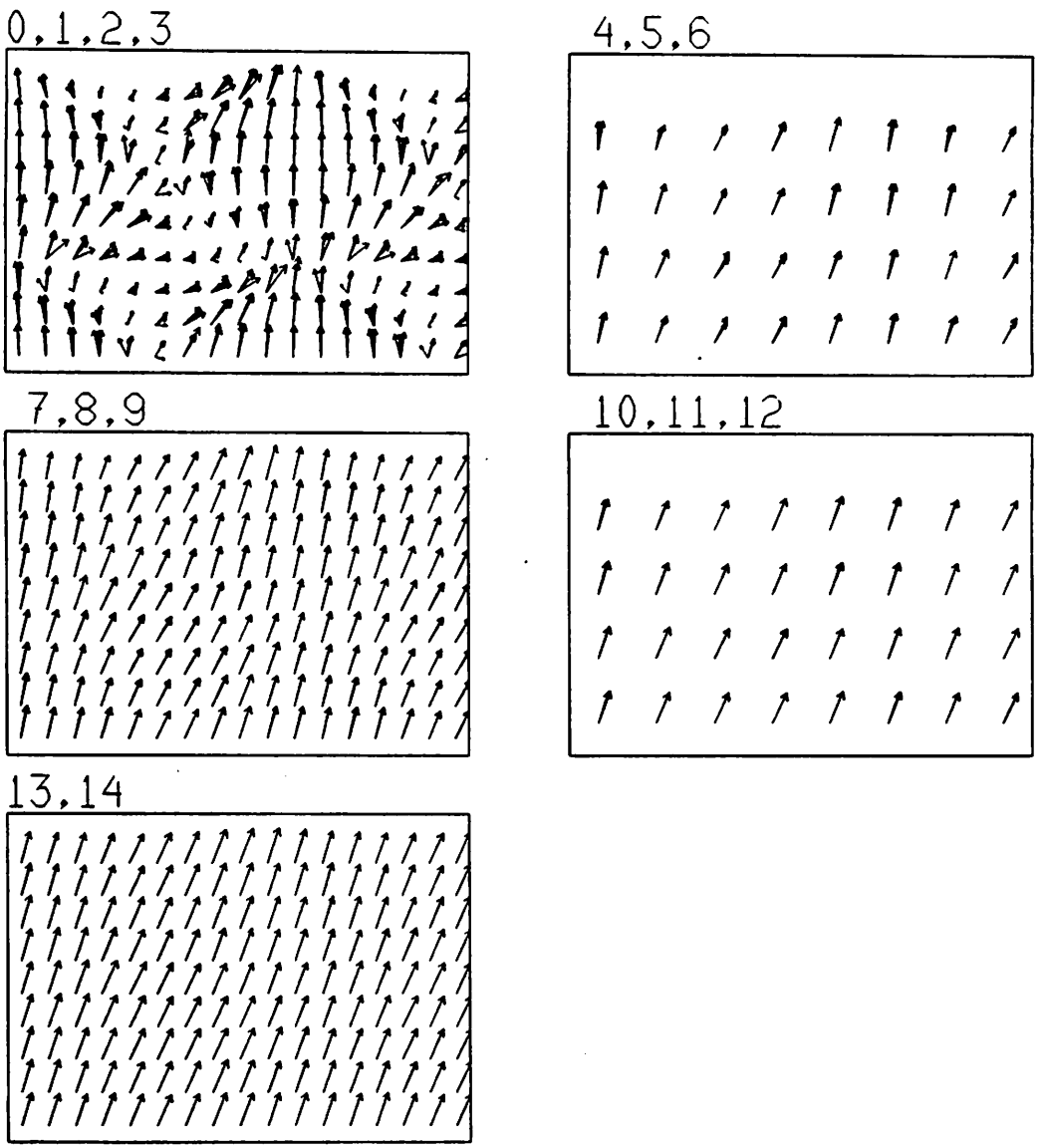


Figure 66: Multilevel optic flow computation
iterations 0,1,2,3 at level 5
iterations 4,5,6 at level 4
iterations 7,8,9 at level 5
iterations 10,11,12 at level 4
iterations 13,14 at level 5

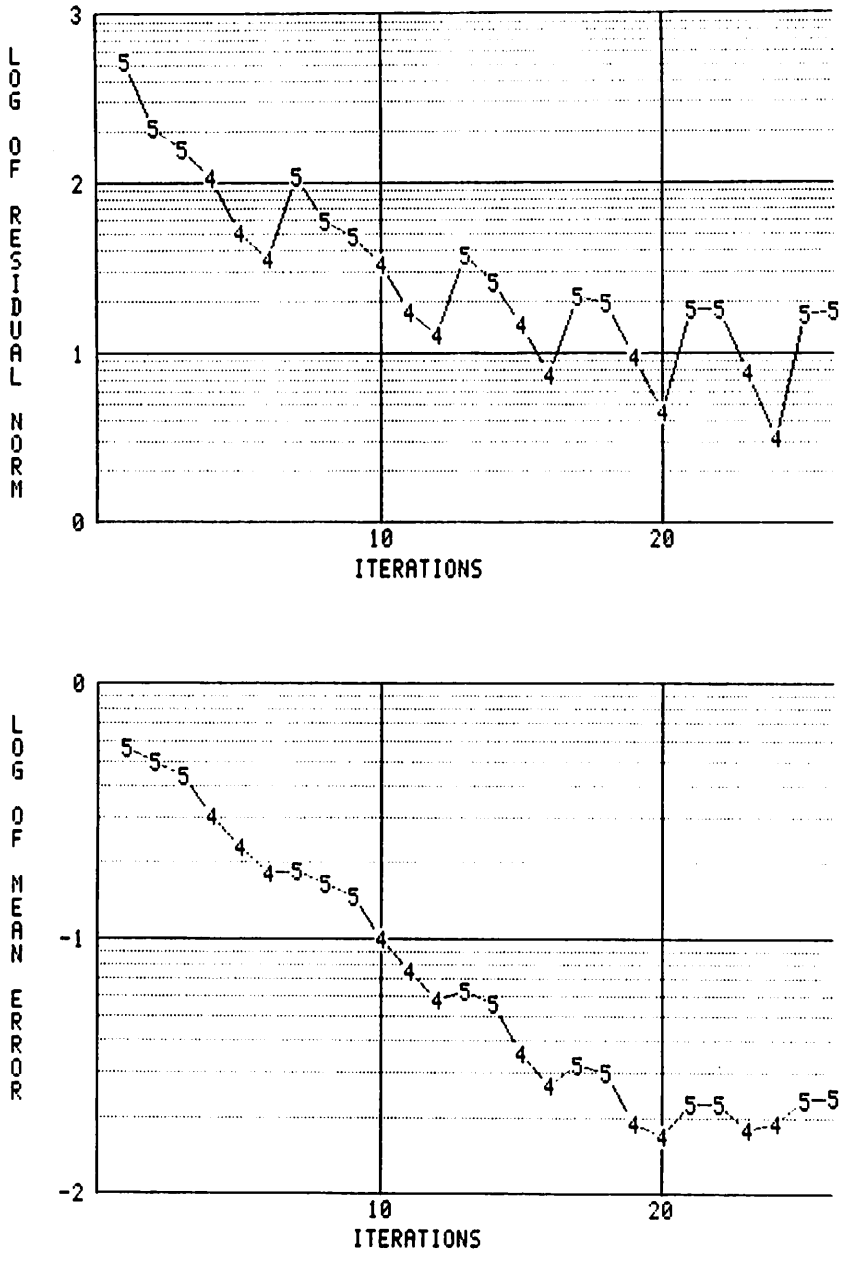


Figure 67: Multilevel optic flow computation: error graphs

Single-level, iteration 50

Actual	D_{row}	D_{col}
mean	−1.007	.5205
SD	.01478	.01667
minimum	−1.056	.4595
maximum	−0.9584	.5715
Expected	−1.0	.5

Multilevel, iteration 22

Actual	D_{row}	D_{col}
mean	−1.003	.5082
SD	.01554	.01967
minimum	−1.066	.4541
maximum	−0.9582	.5694
Expected	−1.0	.5

Table 4: Optic flow field statistics

These tables compare the actual computed optic flow against the expected values. The table on the left shows the results after 50 iterations of the single level relaxation algorithm. The relative errors of the mean actual value in the two components are .7% and 4.1%. The table on the right shows the results after 22 iterations of the multilevel optic flow algorithm. The relative errors of the mean actual value in the two components are .3% and 1.6%.

The first image frame is the same as the first frame in Figure 63. The second frame is a rotated version of the first frame, where the rotation angle is .05 radians (2.86 degrees) and the center of rotation is the point 1/4 in from the left and bottom sides. In Figure 68 the edge flow for a rotational motion is shown. This vector field is used as the initial estimate of optic flow.

Figure 69 shows later stages in the multilevel relaxation, and Figure 70 shows the error graphs for this experiment. The computed flow fields shown look good to the eye and the fine-level field at iteration 13 has reached a mean error magnitude of less than a tenth of a pixel. However, at level 3 the algorithm has diverged and does not return to level 5 where, as the downward trending data suggests, we would expect an even more accurate fine-level field to have been obtained compared to the final time the algorithm was at level 5. We will consider this problem further.

In this experiment, the highest level allowed was level 3. In a similar experiment not shown, when control was allowed to pass to level 2, divergence occurred and was more extreme. If coarsening is only allowed up to level 4, then divergence

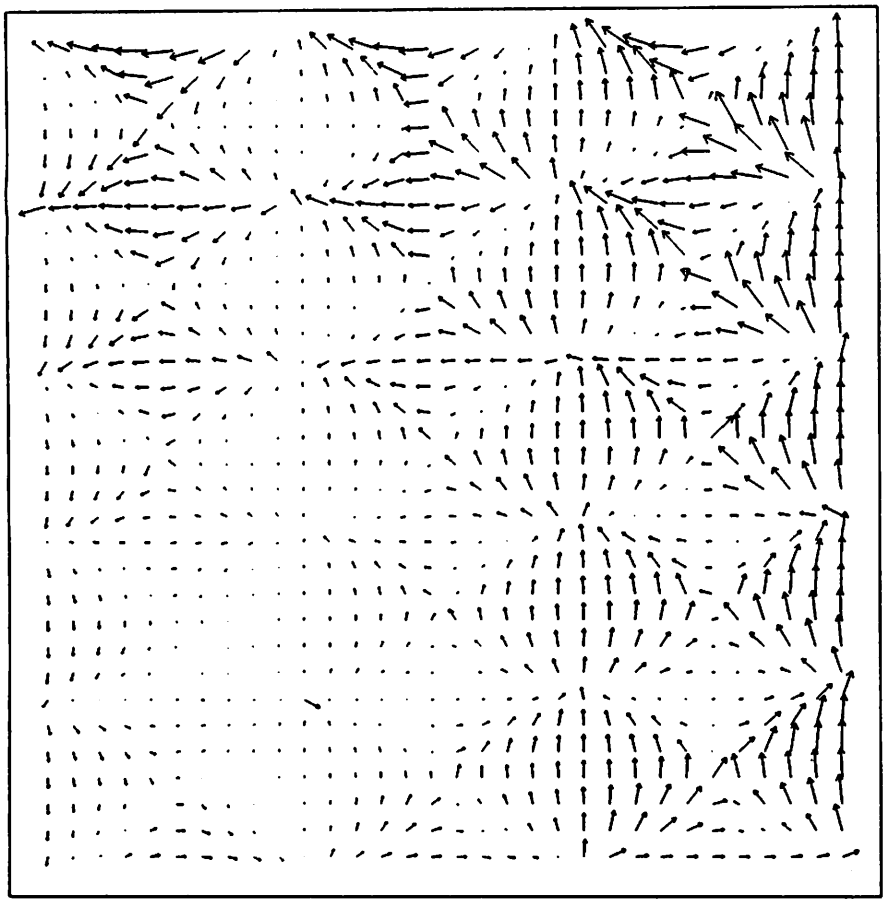


Figure 68: Rotational edge flow

The edge flow field for the rotational motion data as given by Equation 92. This vector field is used as the initial estimate of optic flow.

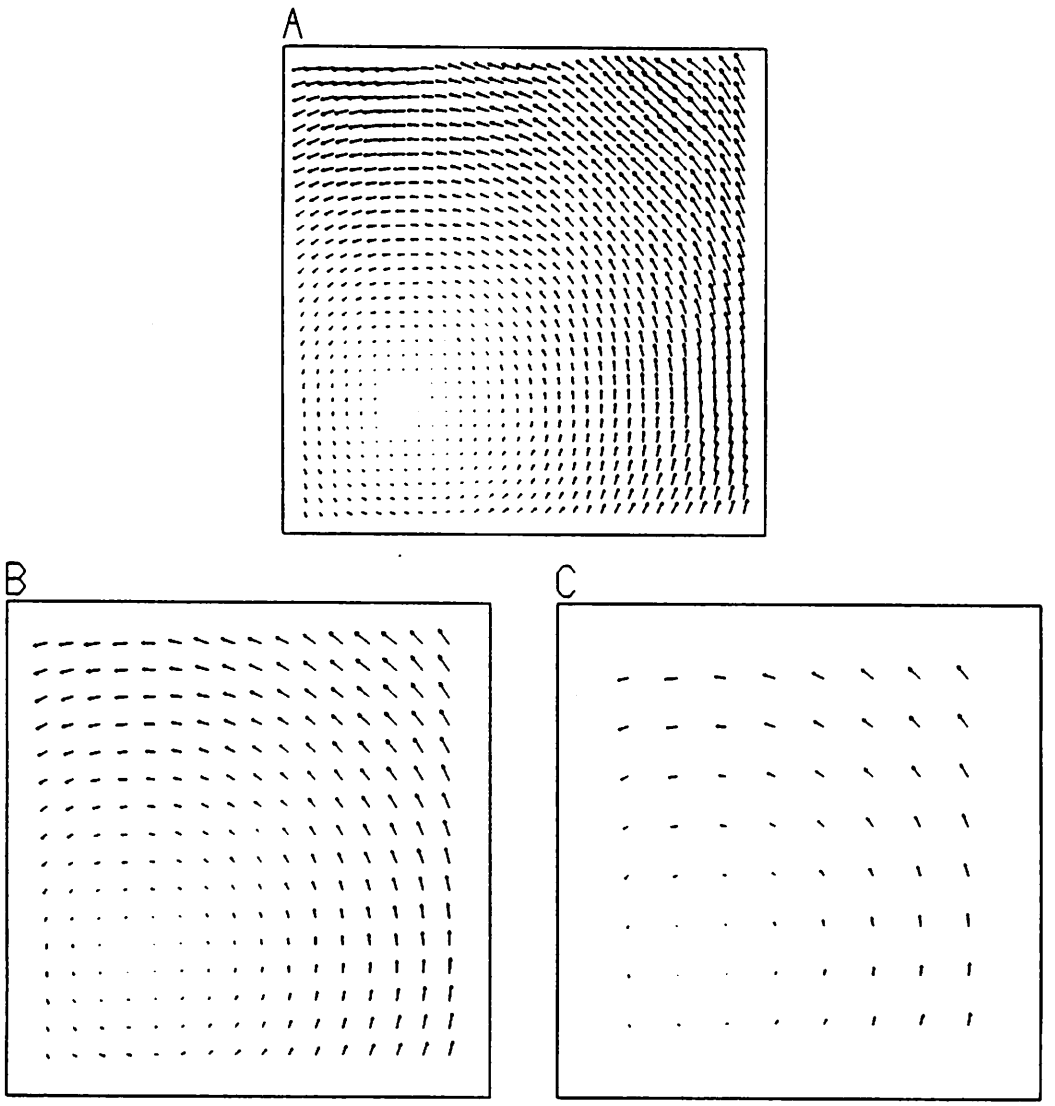


Figure 69: Multilevel flow computation: rotational motion

(a) iteration 13 at level 5

(b) iteration 16 at level 4

(c) iteration 20 at level 3

does not occur and more accurate level 5 fields are obtained. However, in that experiment convergence at level 4 eventually slowed to the point where the target residual norm was not reached after tens of iterations, leaving control "stuck" at level 4. This behavior will be examined later in Section 8.2.

In these last experiments on rotational motion fields, the basic Cycle C/FAS algorithm fails to work. Divergence at coarse levels prevents good fine-level solutions from being attained. In the next sections, we further analyze multilevel optic flow relaxation equations to show how this divergence is caused by local variations in the image data, where "local" is measured relative to the grid resolution. This being the case, hierarchical control must be prevented from passing too high in the processing cone. We will see that instituting a fixed cycling scheme, that is, a fixed pattern of travel up and down the cone, solves the divergence problem.

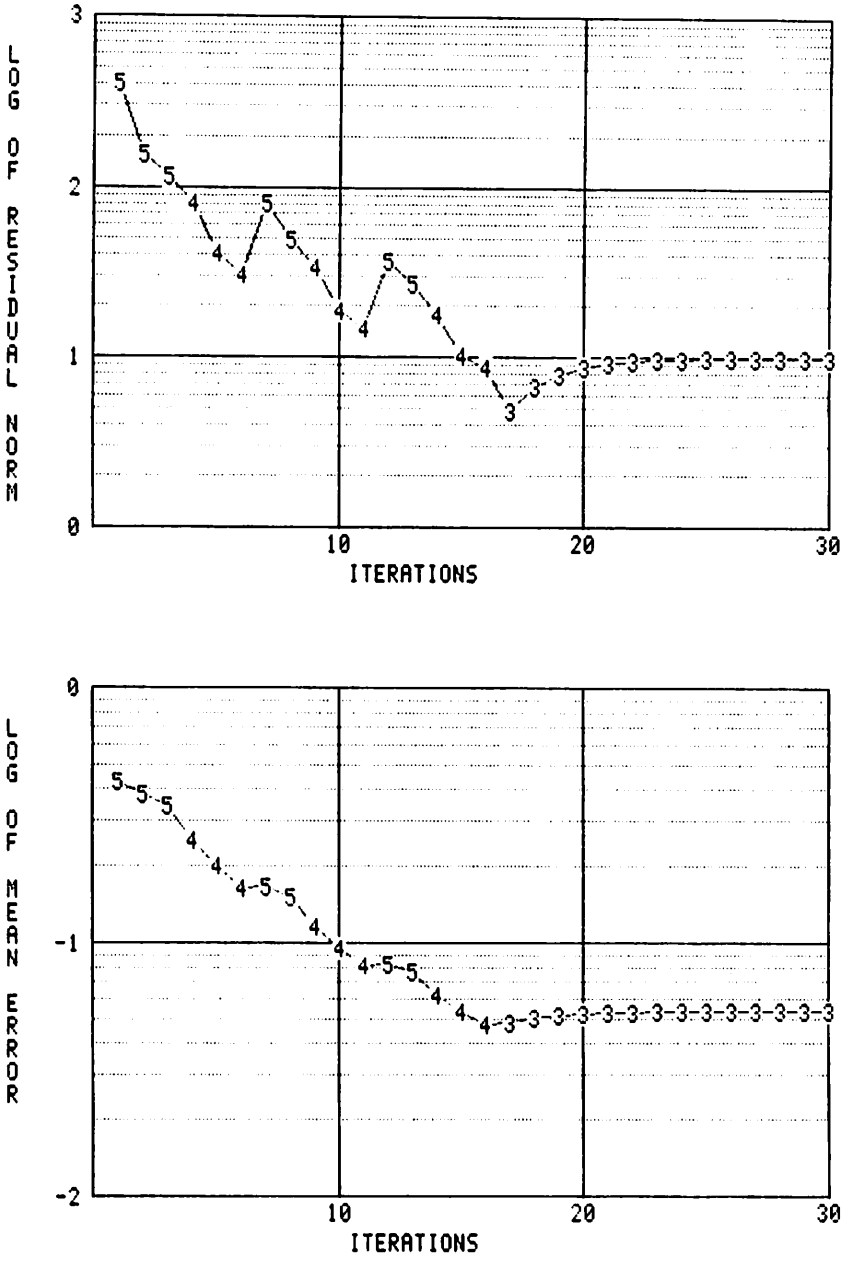


Figure 70: Multilevel flow computation: rotational motion, error graphs

7 Local Mode Analysis

In this section, we describe Brandt's notion of *local mode analysis* and we consider the local mode analysis of optic flow relaxation. This further analysis of multilevel optic flow relaxation serves to justify our formulation and to provide added insight into the nature of its convergence properties. Brandt's purpose was to provide a good approximate measure of the convergence rate of multilevel algorithms by considering the error reduction performed at each level in a hierarchical algorithm. In our analysis for optic flow relaxation, besides obtaining the approximate convergence rates, we show how the error reduction is related to both the direction and the magnitude of the gradient. It will be shown that when the gradients are weak (i.e. low magnitude) convergence behavior is equivalent to simple smoothing; and that when gradients are strong convergence will be accelerated towards the constraint line.

Multilevel relaxation algorithms use relaxation to smooth out error components. At a given level, relaxation is effective in smoothing out those components of the error which are high frequencies at that level. The overall rate of convergence for a multilevel algorithm is directly related to the rate at which high frequency error terms are reduced at each level, and it is independent of the rate of reduction of low frequency terms, which as we know can be very slow. Brandt introduced *local mode analysis* to approximate the rate at which high frequency error terms are reduced in the approximate solution [Brandt 77a]. This involves defining a *local convergence rate* to measure error reduction of the high frequency error terms.

Local mode analysis involves a Fourier analysis of the error terms. In this analysis, the convergence rate for individual error frequencies (as defined by the Fourier Transform of the error function) is computed. The local convergence rate is then defined as the maximum individual convergence rate over the higher frequencies. These higher frequencies are selected to be those in the baseband at a given level which are not also in the baseband of the next coarser level. That is, the frequencies which are representable at the given level but not at a coarser level.

The details of local mode analysis can be found in Section 1 of Appendix A where a thorough analysis is performed on the simple case of Laplace's equation. The main result appears in Section 2 of Appendix A, where we perform a local mode analysis of optic flow relaxation. We briefly summarize the results of the former analysis before examining the optic flow analysis in detail.

7.1 Local Mode Analysis of Laplace's Equation

After a given relaxation iteration, the error is equal to the difference between the exact solution (to the discrete system of equations) and the current approximation. Individual frequency components of the error are given by the Fourier transform of the error function $\delta(m,n)$

$$\delta(m,n) = \int_{|\theta|\leq\pi} A_\theta e^{i(\theta_1 m+\theta_2 n)}\, d\theta$$

where $\theta = (\theta_1, \theta_2)$ and $|\theta| = \max(\theta_1, \theta_2)$. The convergence rate for the individual θ component is then defined as

$$\mu(\theta) \triangleq \left|\frac{\overline{A}_\theta}{A_\theta}\right|$$

where A_θ is the Fourier coefficient before a given iteration of relaxation and $\overline{A}_\theta$ is the Fourier coefficient after the iteration. The values of $\mu(\theta)$ range from 0 to 1, with low values indicating fast convergence and high values slow convergence. The **local convergence rate** is defined as the maximum individual convergence rate over the high frequencies, that is:

$$\bar{\mu} \triangleq \max_{\pi/2\leq|\theta|\leq\pi} \mu(\theta)$$

The results of local mode analysis depends on the choice of finite difference approximations, and on the choice of iterative update scheme. In Appendix A, with Δ_2 used to approximate the discrete Laplacian and Jacobi relaxation scheme used, the individual convergence rate was found to be

$$\mu(\theta) \triangleq \left|\frac{\overline{A}_\theta}{A_\theta}\right| = \frac{1}{3}|\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2| \qquad (93)$$

and hence the local convergence rate is given by

$$\bar{\mu} \triangleq \max_{\pi/2 \le |\theta| \le \pi} \mu(\theta) = \mu(\pi, \pi) = \frac{1}{3}$$

Note that at low frequencies (θ near $(0,0)$), individual convergence rates can go 1, indicating no reduction of the error term. This is of no consequence in a multilevel algorithm since low frequency error is handled by coarser levels.

7.2 Local Mode Analysis of Optic Flow Relaxation

In the local mode analysis of Laplace's equation, the error component at frequency θ before and after relaxation were represented by the Fourier coefficients A_θ and A_θ respectively. In the case of optic flow, the error component is a vector field, and there are two Fourier coefficients (A_θ, B_θ) before and two coefficients $(\overline{A}_\theta, \overline{B}_\theta)$ after relaxation, one for each of the vector components.

The general relationship between these coefficients is given by the *convergence matrix* — a 2×2 matrix relating error components at a given spatial frequency before and after relaxation. This generalizes the single valued individual convergence rate seen in the case of Laplace's equation. The local convergence rates can then be defined in terms of the eigenvalues of the convergence matrix.

First we state the results of the local mode analysis of optic flow relaxation. (The proof is found in Appendix A.)

Local Mode Analysis VII.1 *Suppose we use the discrete Laplacian Δ_2 in a Jacobi relaxation scheme to do multilevel optic flow relaxation. If (A_θ, B_θ) are the Fourier coefficients of the θ component of the error before relaxation, and (A_θ, B_θ) are the coefficients after relaxation, then these coefficients are related by the* **convergence matrix** *M as follows:*

$$\begin{bmatrix} \overline{A}_\theta \\ \overline{B}_\theta \end{bmatrix} = M \begin{bmatrix} A_\theta \\ B_\theta \end{bmatrix}$$

$$= \frac{\mu(\theta)}{3\alpha^2 + F_x^2 + F_y^2} \begin{bmatrix} 3\alpha^2 + F_y^2 & -F_xF_y \\ -F_xF_y & 3\alpha^2 + F_x^2 \end{bmatrix} \begin{bmatrix} A_\theta \\ B_\theta \end{bmatrix}$$

where $\mu(\theta) = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2)$. *The eigenvalues of* M *are*

$$\lambda_{min} = \mu(\theta)\frac{3\alpha^2}{3\alpha^2 + F_x^2 + F_y^2} \qquad \lambda_{max} = \mu(\theta)$$

The eigenvector associated with λ_{min} *is* $\nabla F \triangleq (F_x, F_y)$, *the gradient of* F. *The eigenvector associated with* λ_{max} *is* $\nabla_\perp F \triangleq (-F_y, F_x)$, *a vector perpendicular to the gradient of* F.

The convergence matrix generalizes on the convergence factor $\mu(\theta) \triangleq |\overline{A}_\theta / A_\theta|$ defined in the case of Laplace's Equation. (In a non-rigorous sense it is a measure of $[\overline{A}_\theta \ \overline{B}_\theta]^T / [A_\theta \ B_\theta]^T$.)

Various remarks are now in order:

1. When the convergence matrix M operates on error components, λ_{max} provides an upper bound on how much that error component is reduced in magnitude. λ_{max} is, by definition, the *spectral radius* of M; and, since M is symmetric, it is equal to the *spectral norm* of M; and thus for any vector $\mathbf{x}$ we know that $\|M\mathbf{x}\| \leq \lambda_{max} \|\mathbf{x}\|$ where $\|\cdot\|$ is the Euclidean norm [Varga 62].

2. The convergence matrix tells us how error is reduced at different spatial frequencies. The action of M on a component (A_θ, B_θ) of the error vector δ is given by: (1) break (A_θ, B_θ) into component vectors parallel to ∇F and $\nabla_\perp F$; (2) multiply those components by λ_{min} and λ_{max} respectively; (3) add the results of (2) together. More formally, if $(A_\theta, B_\theta) = x_1 \nabla F + x_2 \nabla_\perp F$, then $M(A_\theta, B_\theta) = (\lambda_{min} x_1)\nabla F + (\lambda_{max} x_2)\nabla_\perp F$ and, iterating n times, $M^n(A_\theta, B_\theta) = (\lambda_{min}^n x_1)\nabla F + (\lambda_{max}^n x_2)\nabla_\perp F$

3. The term $\mu = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2)$ is determined by the particular discrete Laplacian Δ_2 that is used. Note that it is equivalent to the individual convergence rate derived in the local mode analysis of Laplace's equation.

4. If the gradient is zero, then $\lambda_{min} = \lambda_{max} = \mu(\theta)$, the convergence matrix is equal to $\mu(\theta)$ times the identity matrix, and relaxation serves to multiply the

error vectors by $\mu(\theta)$. Convergence in the two components of the optic flow is identical to that seen for simple smoothing by Laplace's equation. This is consistent with the fact that the optic flow update equation reduces to simple smoothing (as expressed by Laplace's Equation) when the gradient vanishes.

5. If the gradient is not zero, then the convergence rate perpendicular to the gradient is still $\mu(\theta)$, but the convergence rate parallel to the gradient is faster in proportion to the gradient magnitude (squared). Thus, error components perpendicular to the gradient still converge at the simple smoothing rate, while error components parallel to the gradient converge at a faster rate, where the decrease is proportional to the magnitude of the gradient.

If we measure overall convergence by the maximum eigenvalue $\lambda_{max} = \mu(\theta)$, we must conclude that the local convergence rate is the same as that for simple smoothing. The minimum eigenvalue λ_{min} is associated with convergence towards the constraint line, since the direction parallel to the gradient is perpendicular to the constraint line. For strong gradients (large magnitude), λ_{min} is small implying fast convergence. This is a quantitative measure of the enforcement of the velocity line constraint at a point.

There is a second aspect to convergence of optic flow relaxation which is not captured by the local mode analysis. The local mode analysis only depends on the gradient at the central pixel. Any effect that locally varying gradient directions have on convergence are not reflected directly in the convergence matrix. Local mode analysis does not give us any measure of convergence based on the accumulation of information from multiple constraint lines with varying directions. In particular, it does not lend any insight into the divergence we have seen at coarse levels. In the next section, we study the relationship between coarse divergence and (1) variations in the flow field and (2) variations in the image data. The latter is shown to directly influence both overall speed of convergence and the occurrence of divergence.

8 Analysis of Coarse Approximations

In this section we consider the divergence of the multilevel optic flow relaxation algorithm at coarse levels of the processing cone. This divergence may be due to variations of the disparity field or in the image data itself. We first show that the former is not the case and then go on to analyze the relationship between local structure in the image data and convergence/divergence at varying resolutions.

8.1 Variation of the Disparity Field

The multilevel algorithm does not exhibit divergence at coarse levels in the case of pure translational motion. This suggests the possibility that coarse divergence may be related to variation in the disparity field to be computed. Clearly, if a disparity field with large enough variation was chosen, the local smoothness property on which the optic flow relaxation method is based would be violated, and we would not expect a relaxation technique to convergence to a reasonable solution. However, this does NOT explain the occurrence of divergence at coarse levels only as seen in Figure 70. As we show next, the variation of rotational disparity fields is the same at all resolution levels. Thus, with high enough variation, we would have to see divergence at all levels, not just at coarse levels.

Consider, in the continuous case, the rotational disparity field $(u, v) = (u(x, y), v(x, y))$ centered at (x_c, y_c) with angle β. It is given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\beta - 1 & \sin\beta \\ -\sin\beta & \cos\beta - 1 \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix}$$

The variation of the field (u, v) is given by the gradients of u and v. Considering their magnitudes, we see that $|\nabla u|^2 = |\nabla v|^2 = 2(1 - \cos\beta)$ — a constant value. Now consider a discrete rotational field formed by sampling the continuous field $(u(x, y), v(x, y))$ at grid points. If finite differences are used to measure the gradient, the result is exact (up to the finite arithmetic precision) because the individual components u and v are linear functions of x and y. Thus, the local variation of a

rotational disparity field as given by the finite difference gradient magnitude of u and v is a constant value independent of grid level.

8.2 Variation of the Image Data

We are now left with variations in the image data to explain coarse level divergence. Intuitively, the reason seems to be that the optic flow PDE is not accurately represented at coarse levels because of the spatially varying coefficients. This idea is taken up more fully in the next section. For now, we point out that the coefficients are directly computed from the image data, being products of first differences. Spatial variation in the coefficients is due to spatial variation in the underlying image data. In this section, the relationship between spatial variations in the image data and convergence/divergence of optic flow relaxation is shown by experiment.

Consider the spatial variation of the image data in the previous experiments. The test image in Figure 63 is a product of sinusoids in the x (column) and y (row) directions with wavelengths of 12.8 and 21.33 respectively at the finest level. If a low-pass pyramid is built from this data, the corresponding wavelengths at coarser levels would be 10.67, 5.33, 2.67 at levels $k = 4, 3, 2$ respectively for λ_y and 6.4, 3.2, (1.6) at levels $k = 4, 3, 2$ respectively for λ_x. The level 2 x-wavelength of 1.6 is not actually representable since the maximum wavelength representable at any level is $\lambda = 2$. These numbers suggest that the coefficients for the optic flow PDE cannot be adequately represented at level 2 and that even at level 3 the variation may be too high.

If divergence at some coarse level is related to the degree of local variation in the image data, where "local" is measured relative to the pixel spacing at the coarse level, then we expect that when the test data has less variation (i.e. lower spatial frequencies) convergence will be obtained at coarser levels. This is in fact the case as experiment will show. Before demonstrating this for rotational motion fields, we revisit the case of translational motion for both single-level and multilevel experiments, this time with data of less spatial variation. These experiments on translational motion show the relationship between relative frequency content and

convergence/divergence independent of any variation in the motion field. We start with a single-level relaxation experiment that shows how convergence is slower for lower frequency data.

The test data for these experiments is equivalent to that used in the prior experiments (Figure 63) with one important exception: the wavelengths of the sinusoids are doubled to 25.6 and 42.67 in the x and y directions respectively. First, we again consider the case of purely translational motion, using the same translational field of $(-1.0, 0.5)$. In Figure 71 the error graphs are shown for single-level optic flow relaxation. This should be compared to the comparable single-level experiment shown in Figure 65. Convergence is clearly much slower for the low frequency data. This must be attributed to reduced local variation in the direction of the image gradients.

The next experiment shows how multilevel relaxation fares on the low frequency data. Figure 72 contains the error graphs for this experiment. Compare them to the corresponding experiment for higher frequency data shown in Figure 67. (Note the different ranges of the iteration count axis in these two experiments.) Two points are easily made. First, for the case of lower frequency data, the hierarchical control strategy found it necessary to go to level 3 twice, while for the earlier experiment level 4 proved sufficient. (In both of these experiments, the hierarchical control parameters would have allowed reduction up to level 2.) Secondly, in both cases, convergence (in terms of the residual norm) was reached after about 20 iterations. It is interesting to note that while the single-level relaxation takes longer to converge on the lower frequency data, the multilevel algorithm made use of a yet coarser level to achieve comparable convergence in the same number of iterations used for multilevel solution of the high frequency case.

Now, we come back to the case of rotational motion, this time with the lower frequency data. The error graphs are shown in Figure 73 and should be compared to the comparable higher frequency data experiment shown in Figure 70. As expected, for the lower frequency data, we no longer see divergence at level 3. We do have the remaining problem that at later visits to level 3 convergence eventually slows

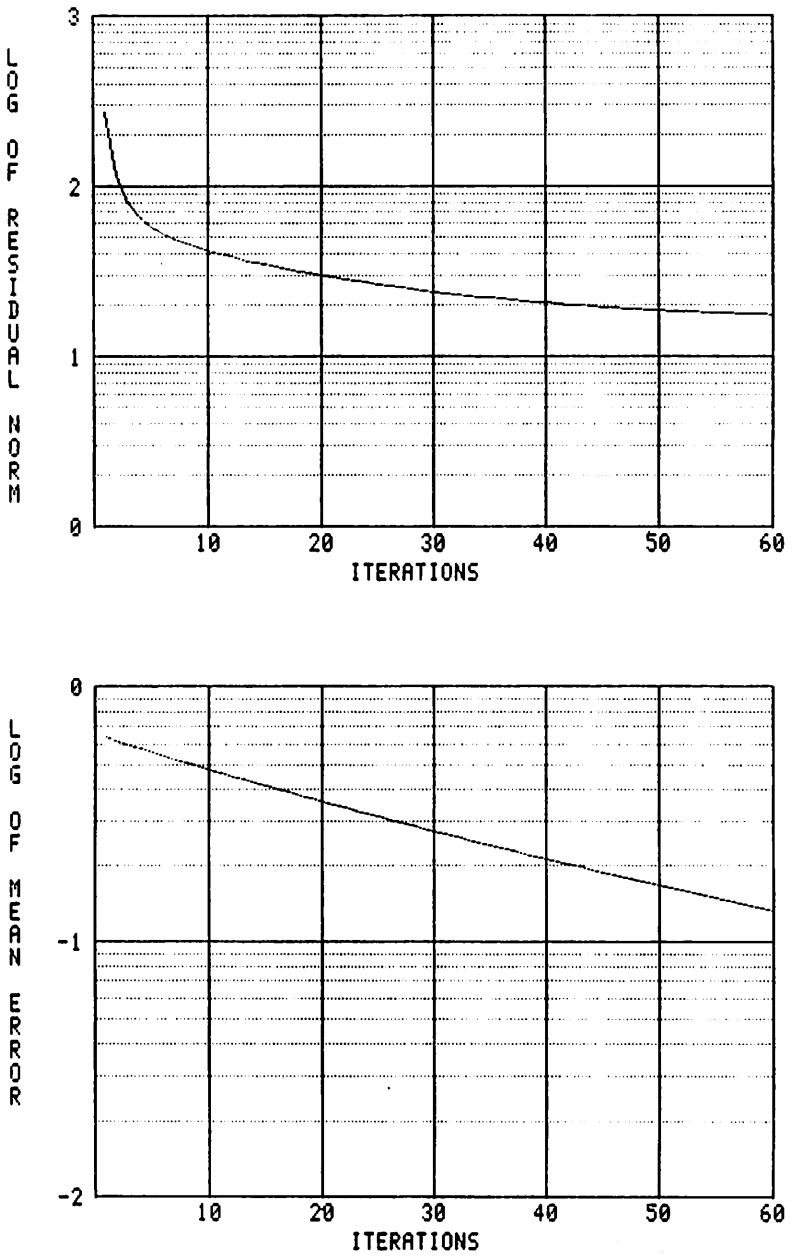


Figure 71: Single-level relaxation: low frequency data

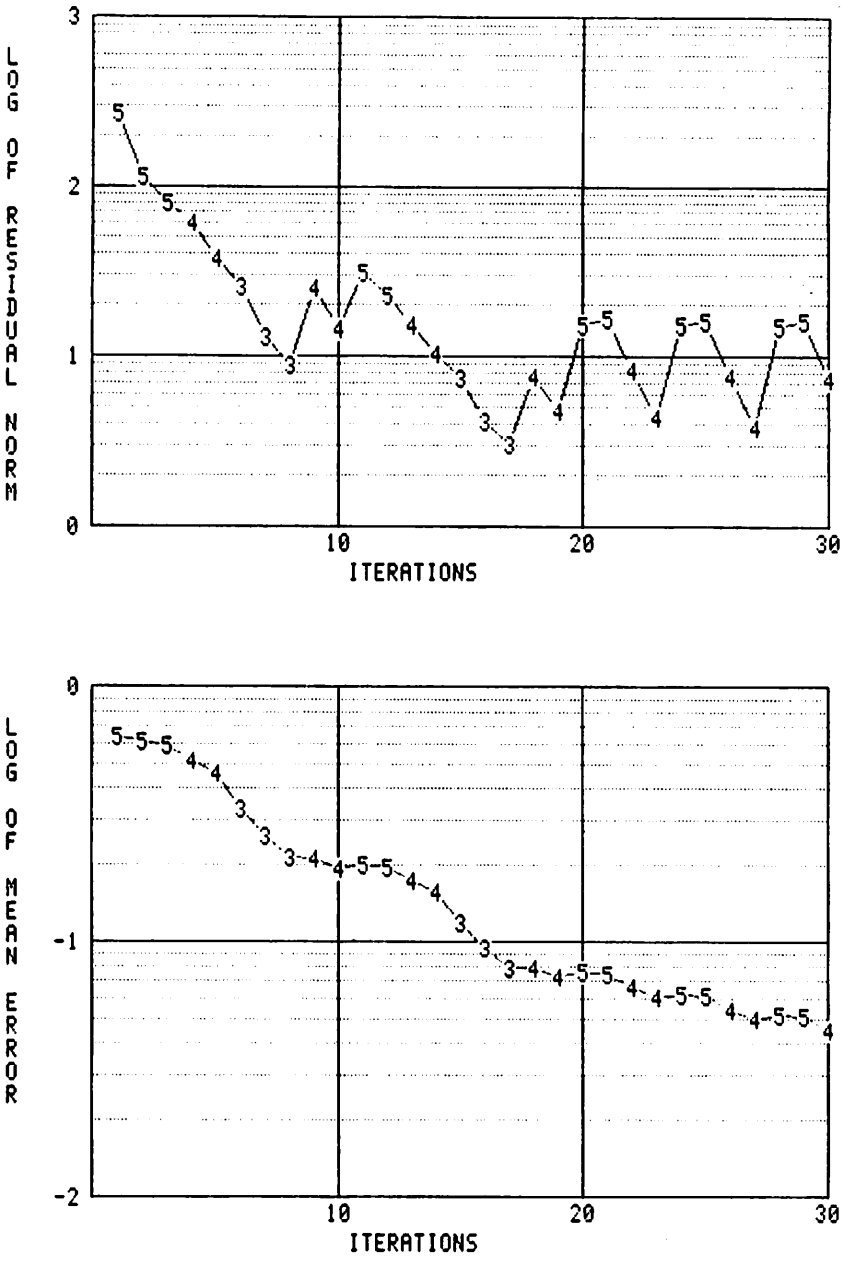


Figure 72: Multilevel relaxation: low frequency data

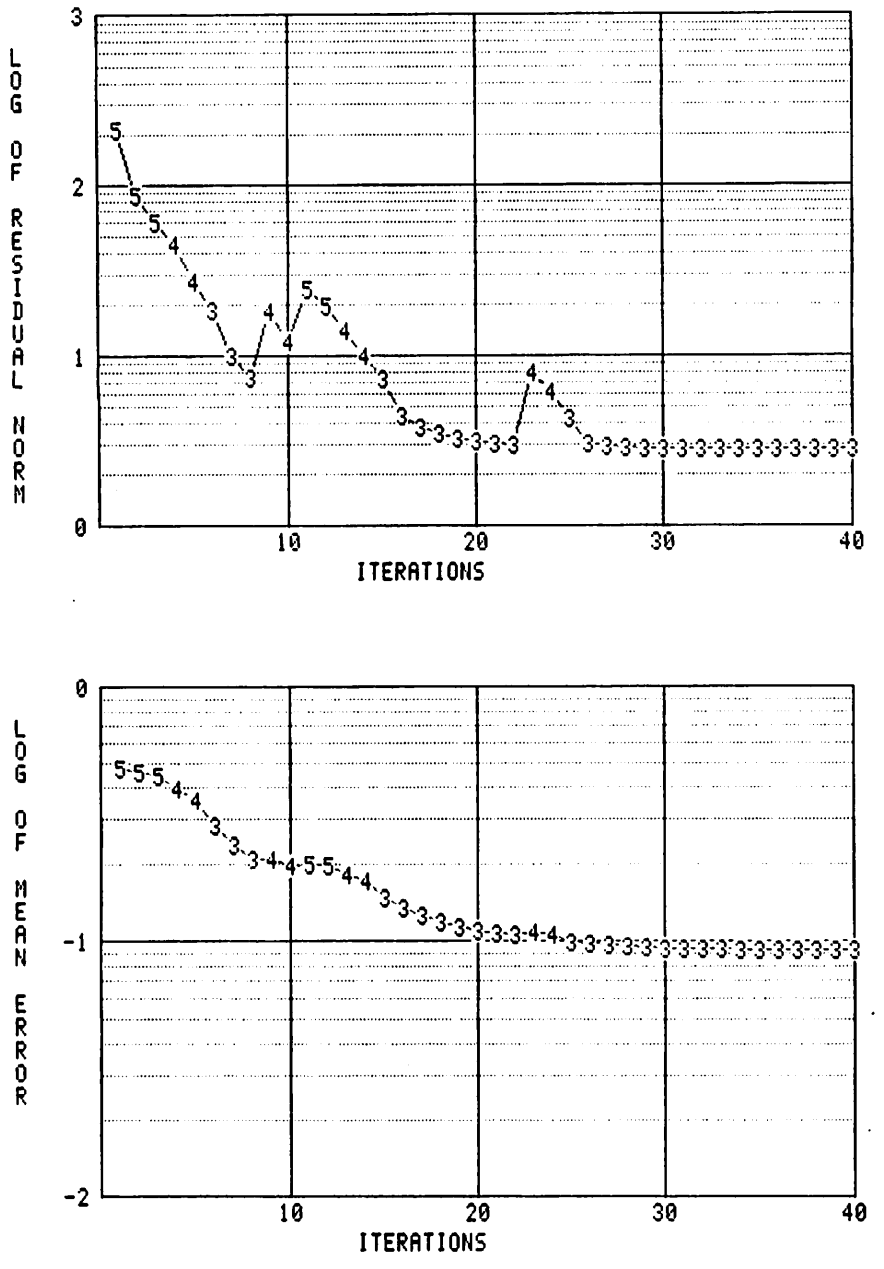


Figure 73: Multilevel relaxation: low frequency data, rotational field

to where the target residual norm (for that visit) may not be reached. This also occurred in an experiment with high frequency data, but that time at level 4. In both cases, a simple limit on consecutive iterations at one level would prevent getting stuck. This will be implemented in the fixed cycle scheme in Section 9.

9 Further Experiments

In the last section, we established the need to avoid coarse level relaxation when fast local variations in the image data at those coarse levels causes divergence. In this section, we show how coarse level divergence is avoided by changing the hierarchical control strategy to a fixed cycling scheme, i.e. the decisions to move up or down (reduce or project) are made not by a dynamic tracking of the residual norm, but rather by a fixed pattern of up/down moves.

The fixed pattern hierarchical control strategy is important to us for two reasons. The first relates to the divergence problem and the related problem of slowed convergence at coarse levels. By limiting the number of iterations during any one visit to a given level, hierarchical control cannot get stuck at coarse levels when convergence slows at later stages of the overall relaxation process. Moreover, if in fact any divergence at coarse level is seen, then limiting iterations also limits the extent to which this divergence corrupts finer-level solutions.

The second benefit of a fixed cycling scheme is that it is no longer necessary to measure the rate of convergence of the residual norm. This is very important for multilevel algorithms run in the parallel processing cone architecture. The measurement of residual norm is a global computation, since it is defined as a summation over an entire level of residuals at individual pixels. A dynamic control strategy which makes reduce/project decisions based on the residual norm must at each iteration collect individual residuals together somewhere. In the locally connected processing cone, such a global operation introduces high communication costs.

In the first two experiments, we use the "high frequency, rotational field" image, that is, the first frame is that shown in Figure 63, and the second frame

is a .05 radian rotation of the first frame image. Recall that both images have one percent uniform noise added in, uncorrelated between the two frames. The Cycle C/FAS experiment shown in Figure 70 used this data.

The fixed pattern of hierarchical control that we have used in all of the next experiments is a "three up, two down" pattern, that is, after a reduction to a coarser level three iterations are performed and after a projection to a finer level two iterations are performed. For the first experiment, reduction is only allowed up to level 4 and hierarchical control simplifies to an alternating pattern of two iterations at level 5 (after an initial three) and three iterations at level 4. The error graphs for this experiment are shown in Figure 74.

For the second experiment, reduction is allowed up to level 3 and the "three up/two down" pattern of hierarchical control is seen. The error graphs for this experiment are shown in Figure 75. For both of these experiments, we see a fine-level rotation field extracted with high accuracy at the finest level — level 5. In the first case, at iteration 33, the mean error magnitude goes to .031 pixel-lengths at level 5. In the second case, at iteration 43, the mean error magnitude goes to .037 pixel-lengths at level 5.

In both cases, the resultant vector fields at all levels are clearly seen to be very good approximations to rotational fields. This is shown for the second experiment in Figure 76. For comparison, refer back to the initial solution estimate (iteration 0 at level 5) for this experiment shown in Figure 68 on page 222. Solution estimates at later stages in the relaxation are shown in Figure 76. Iteration numbers correspond to those shown in the graphs in Figure 75. The rotation field is seen clearly at all levels.

In the third experiment, the .05 radian rotation is applied to the lower frequency data. This is the same image data as used in the experiment shown in Figure 73. In that prior experiment, convergence flattened at level 3 and control did not pass back to the finer levels. In the experiment shown in Figure 77, return to the finest level is assured by the fixed pattern of hierarchical control. Convergence to less than .1 pixel-lengths at the finest level is attained by the 32nd iteration

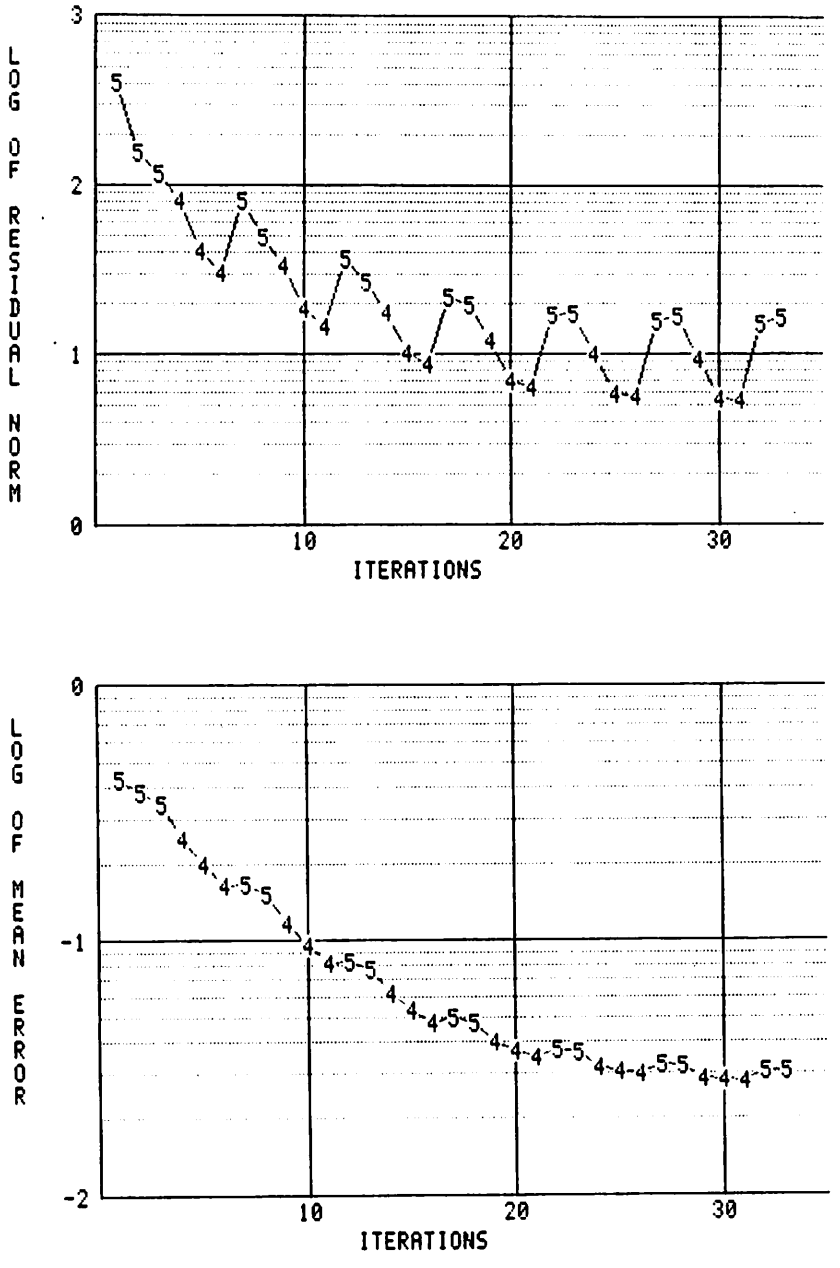


Figure 74: Fixed up/down hierarchical control pattern

The higher frequency data shown in Figure 63 is used with the second frame a 0.05 radian rotation of the first frame data. Reduction was allowed only up to level 4.

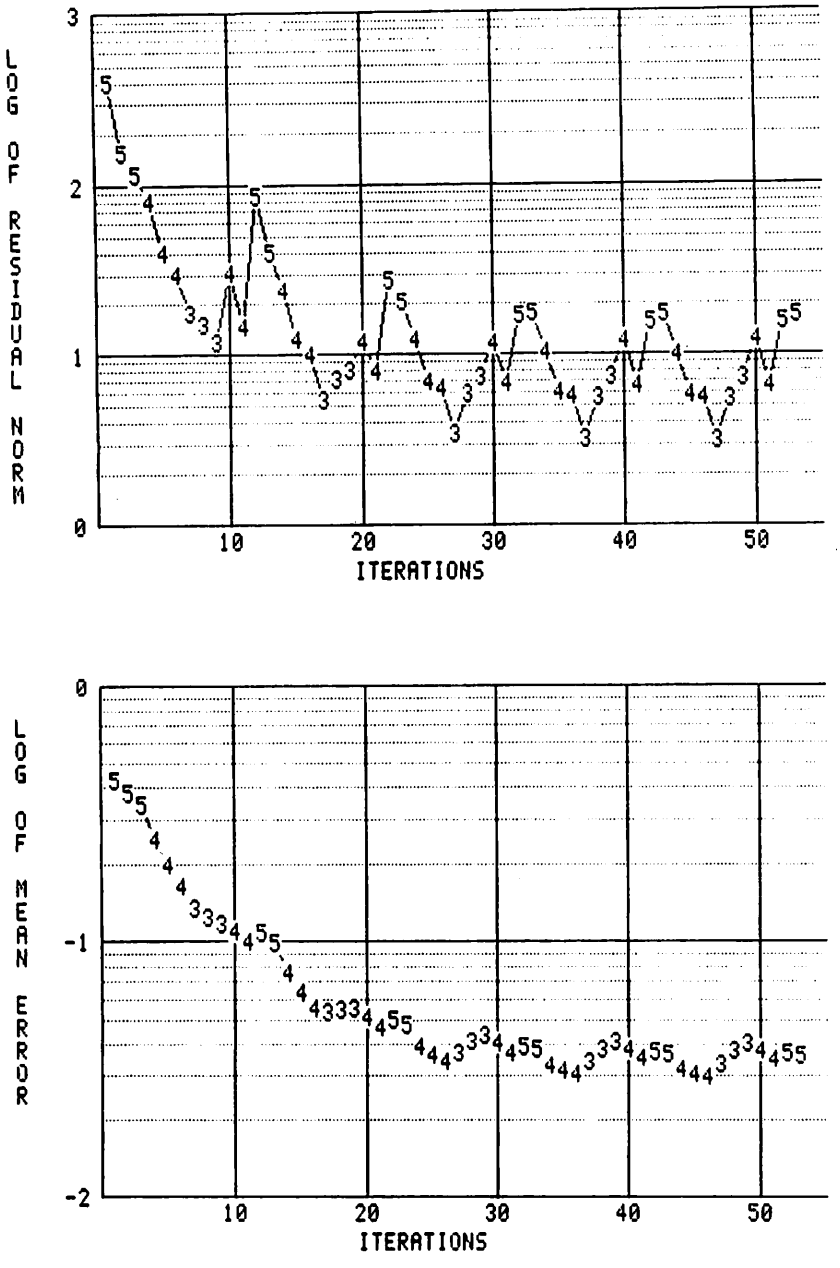


Figure 75: Fixed up/down control, to divergent levels

The image data for this experiment is identical to that used for Figure 74. Reduction was allowed up to level 3.

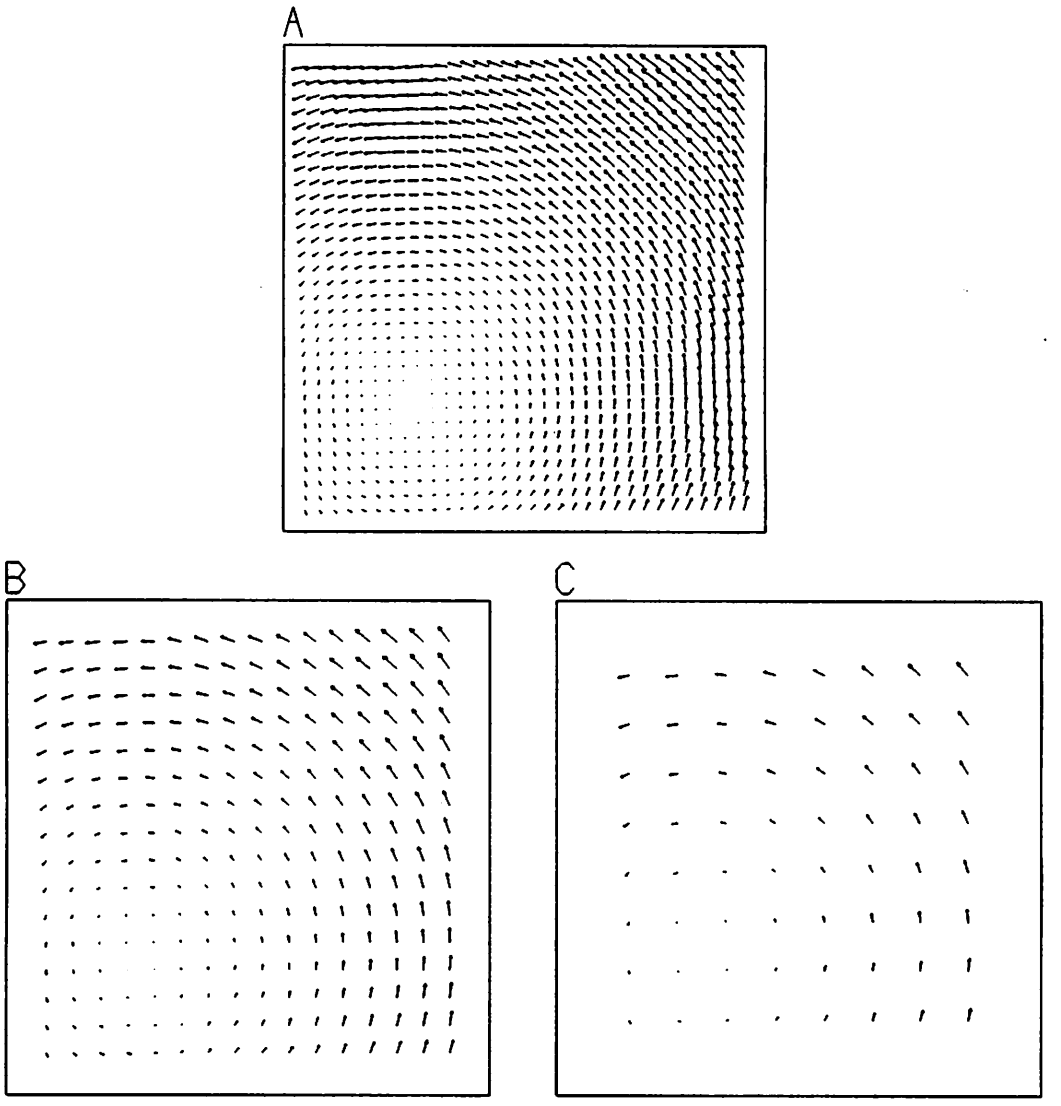


Figure 76: Rotational solution fields

These vector fields are solution estimates for the experiment with error graphs shown in Figure 75.

(a) The fine-level (level 5) solution at iteration 23. The mean error magnitude at this iteration (Figure 75b) is .048 pixel-lengths.

(b) The level 4 solution at iteration 21. The mean error magnitude at this iteration is .047 level-5-pixel-lengths.

(c) The level 3 solution at iteration 19. The mean error magnitude at this iteration is .056 level-5-pixel-lengths.

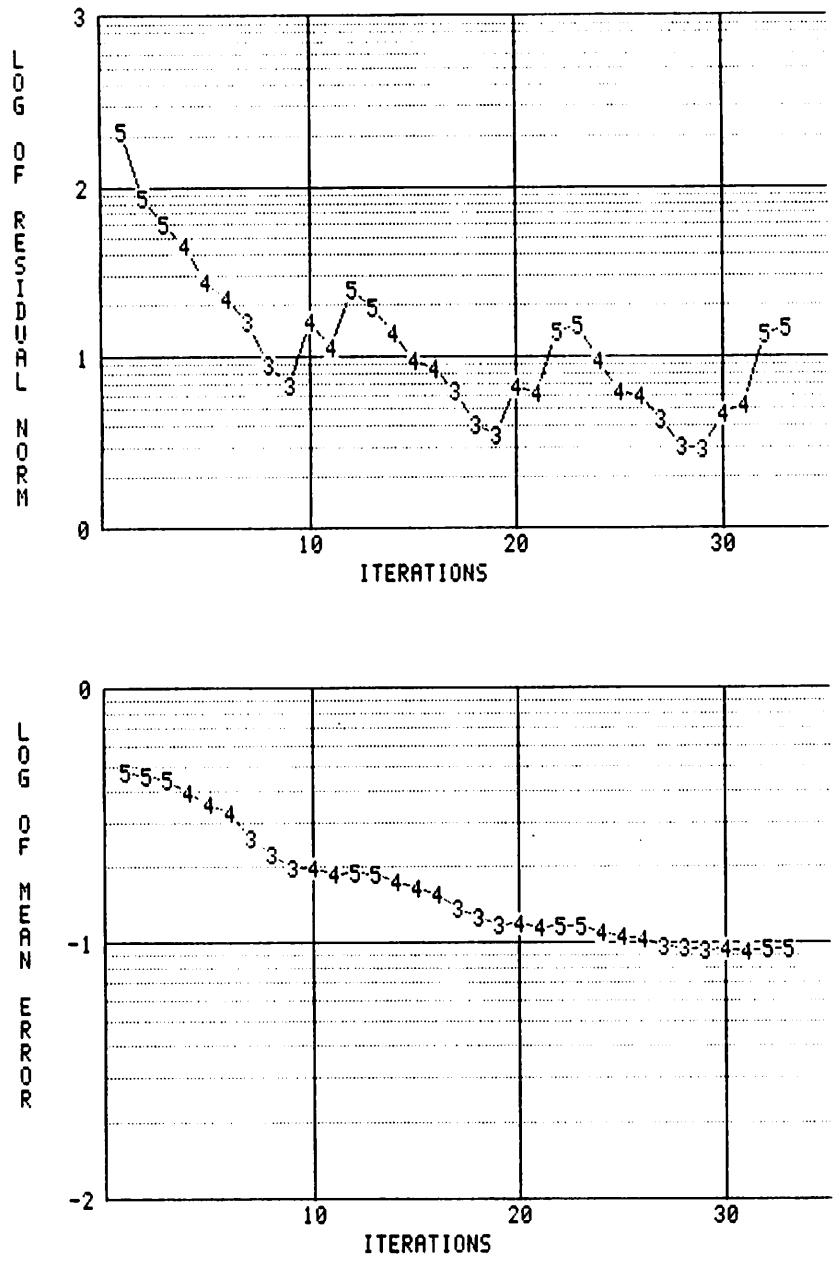


Figure 77: Fixed up/down control, low frequency data

The image data for this experiment is the low frequency data with a rotational flow field, the same data used in the experiment shown in Figure 73. Reduction was allowed up to level 3.

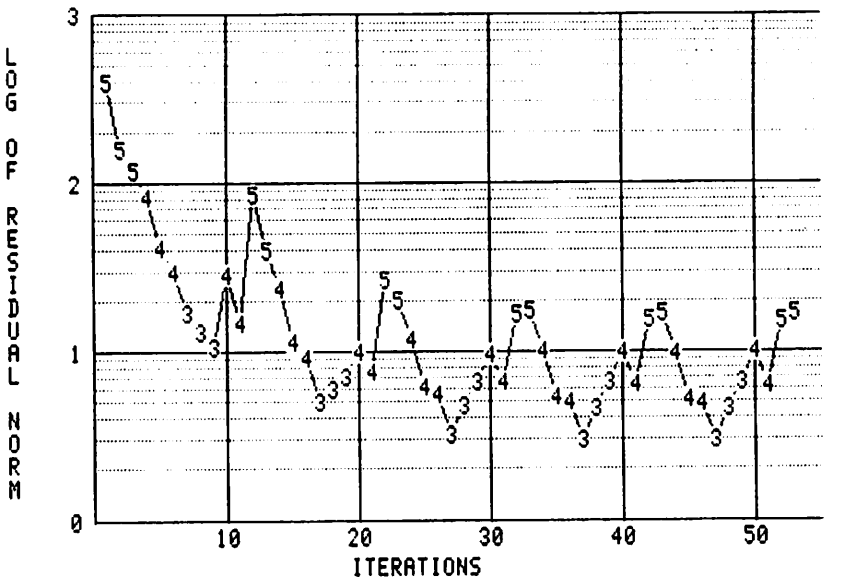


Figure 78: Multilevel relaxation: motion in depth

The higher frequency data shown in Figure 63 is used with the second frame, a 5% up-scaling of the first frame data. Reduction was allowed only up to level 4. Residual norm is plotted vs. iteration number.

during the fourth visit to level 5.

In the final experiment, multilevel relaxation with a fixed hierarchical control strategy is applied to another type of non-translational motion field. In this case, the second frame was computed to be a scaling up of the first frame by 5% about the point 1/4 in from the left and bottom sides. (Such a scaling corresponds to depth motion, i.e. translation perpendicular to the image plane.) Figure 78 shows results comparable to those for the rotational case shown in Figure 75. The edge flow used as the initial optic flow estimate is shown in Figure 79.

The resultant vector fields at all levels are clearly seen to be very good approximations to scaling fields. Solution estimates at later stages in the relaxation are shown in Figure 80. Iteration numbers correspond to those shown in the graphs in Figure 78. The scaling field is seen clearly at all levels.

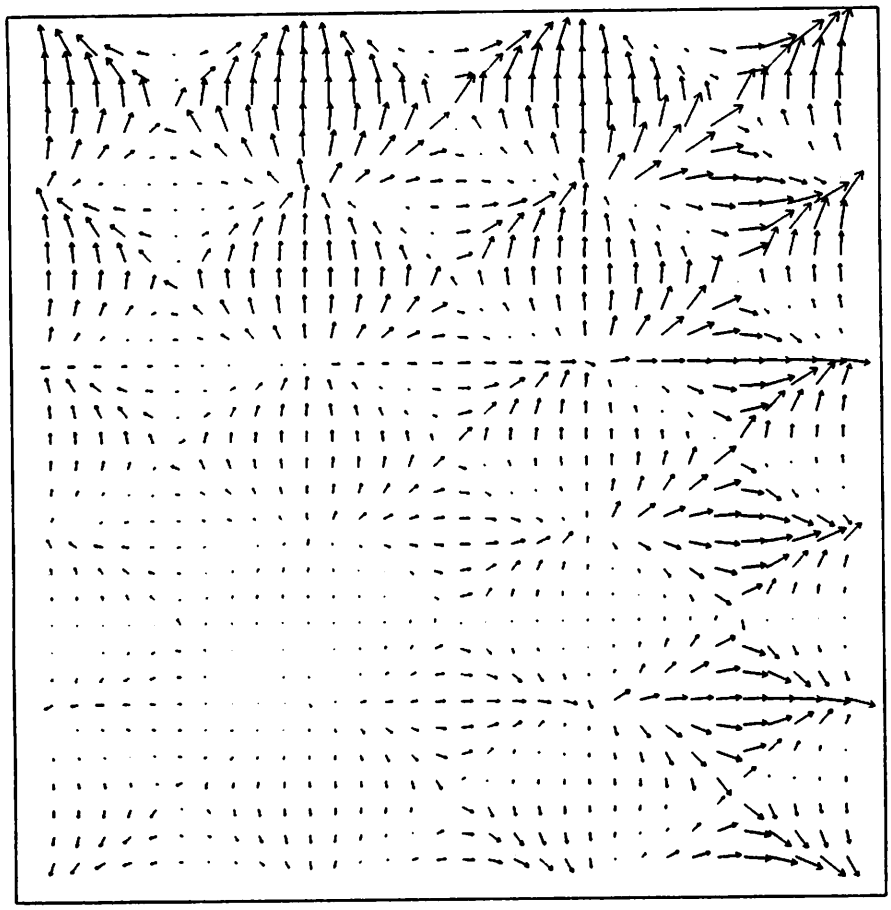


Figure 79: Scaling edge flow

The edge flow field for the depth motion data as given by Equation 92. This vector field is used as the initial estimate of optic flow.

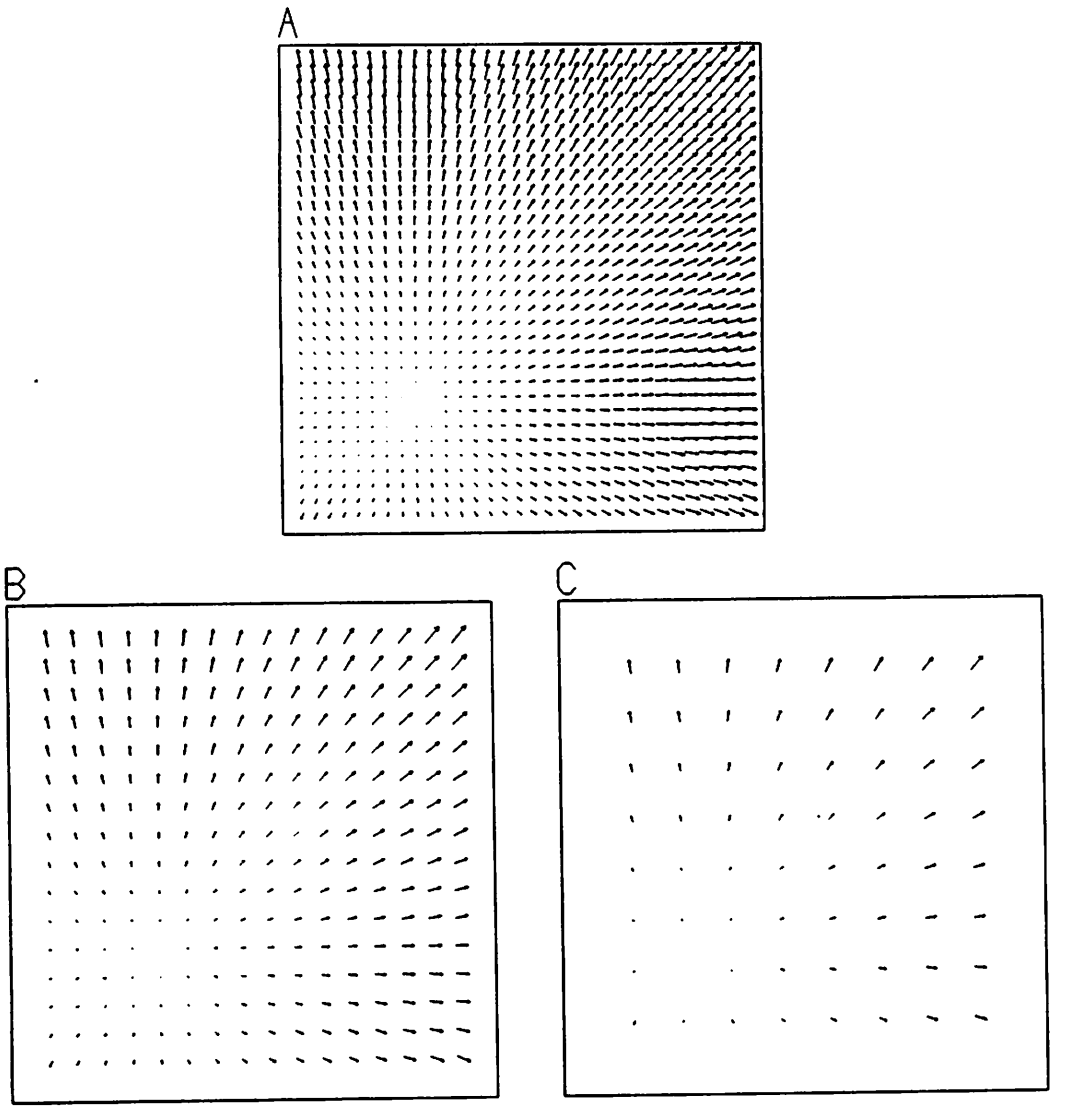


Figure 80: Scaling solution fields

These vector fields are solution estimates for the experiment with error graphs shown in Figure 78.

(a) The fine-level (level 5) solution at iteration 23.

(b) The level 4 solution at iteration 21.

(c) The level 3 solution at iteration 19.

10 Summary

In this chapter we have shown how multilevel relaxation can be used to compute optic flow within the processing cone, using significantly fewer iterations than single-level relaxation. After a relatively straightforward application of Brandt's ideas, we arrived at a multilevel optic flow relaxation algorithm. While exhibiting the expected increased convergence rate over single-level relaxation, some experiments presented a problem of divergence at coarse levels. A local mode analysis of the update equations showed how (1) convergence is at least as fast as simple smoothing; and (2) when the gradient is strong, convergence is accelerated towards the constraint line. However, the local mode analysis did not account for the coarse-level divergence seen in experiments. This divergence was then shown to be due to spatial variation in the image data. That is, when significant variations were on a scale comparable to the pixel spacing at a given coarse level, then divergence was seen. Finally, fixed cycling schemes were introduced as hierarchical control strategies to overcome the divergence problem. Further experiments showed the success of such a method.

CHAPTER VIII

Summary

1 Review

In this thesis we have presented algorithms for the efficient computation of image motions using multiresolution methods in a hierarchical processing architecture. The major contributions lie in three areas: (1) fast construction of the multiresolution image pyramids; (2) hierarchical coarse-to-fine motion detection algorithms in which motion estimates are progressively refined; and (3) multiresolution relaxation algorithms for optic flow computation.

All of the algorithms we have presented are designed to run in a hierarchical multiresolution processing architecture. This architecture is characterized by two main components. First, the *processing cone* is the general machine architecture within which the programs run. Secondly, the *image pyramids* are the multiresolution image representations upon which our hierarchical motion detection algorithms operate.

Pyramid building—the computation of an image pyramid from a single resolution original image—involves low-pass filtering to reduce resolution and interpolation to increase it. A family of *discrete Gaussians* was presented which provides good anti-aliasing characteristics, efficient computation, and a good hierarchical Gaussian approximation. Frequency space analysis, using Fourier Transform theory, was used to compare alternative filters.

In our review of motion computation methods, we emphasized two classes of

algorithms: correlation matching and gradient-based matching. These algorithms best satisfy our goals of designing local, parallel, and uniform algorithms, thereby obtaining optimal utilization of the processing cone's hierarchical and parallel architecture. Both of these methods have drawbacks which are eliminated by embedding them in a hierarchical coarse-to-fine algorithm. The following hierarchical extensions were developed.

Hierarchical correlation overcomes two disadvantages of correlation matching: large search areas which require expensive searches, and repeating features which cause incorrect matches. Coarse-to-fine control provides speed and efficiency when search spaces are large, and accuracy when repeating details can be confused. The hierarchical correlation algorithm operates on band-pass pyramids built using the techniques described in Chapter II. It was shown how direct correlation on these pyramids is essentially equivalent to mean normalized correlation on low-pass pyramids. Matching is performed at each level of these pyramids, starting at the coarsest resolution and proceeding to the finest. The search area at each level can be restricted to a 3×3 set of candidate points. The reduced cost of hierarchical correlation is due to the small search areas. Hierarchical and single-level methods were shown to cost roughly 25 times less when the maximum disparity is 8 and 90 times less when the maximum disparity is 16.

Hierarchical gradient-based algorithms extend single-level algorithms to the computation of large disparities by formulating a hierarchical generalization of the single-level method, which operates on low-pass image pyramids. A complete formal analysis of the hierarchical method was presented, including (1) the basic equations for computing refined disparity vector, (2) the discrete representations and computations for solving these equations, and (3) a geometric interpretation of the resultant relaxation algorithm. Implementation in a hierarchical processing algorithm was detailed. Experiments showed both the failure of single-level methods (for large disparities) and the success of hierarchical methods.

The two hierarchical algorithms were shown by experiment to have comparable accuracy. Comparison of the computational costs, both arithmetic and data

transfer, show that the gradient-based algorithm are, in general, less costly. The computational costs of hierarchical correlation are relatively high for two reasons: (1) many multiplications and additions are needed for the correlation inner product; and (2) many pixel values must be transferred over multi-pixel distances in the image plane.

The two types of hierarchical motion detection algorithms share a coarse-to-fine control strategy used to progressively refine individual motion estimates. The third and final major area of contribution of this thesis, *multilevel relaxation for optic flow computation*, involves an altogether different use of the multiresolution hierarchy, with both coarse-to-fine and fine-to-coarse flow of control used in an overall scheme that provides efficient computation of otherwise slow relaxation processes. A formal application of multilevel methods to the computation of optic flow was presented and experiments showed the expected increased convergence rate over single-level relaxation. However, some experiments presented a problem of divergence at coarse levels. A *local mode analysis* of the relaxation equations showed that convergence is at least as fast as simple smoothing, and that, when the gradient is strong, convergence is accelerated towards the constraint line. The local mode analysis does not account for coarse-level divergence. Divergence was then shown to be due to spatial variation in the image data. Finally, fixed up/down cycling schemes were used to overcome the divergence problem.

2 Future Directions

2.1 Pyramid Building

The choice of G_3 as the discrete Gaussian low-pass filter used to build our low-pass pyramids was partly based on optimality of the 1D filter g_3 for generating hierarchical Gaussian filters. This optimality is measured relative to other four element filters. The ability of larger discrete Gaussians to generate good hierarchical Gaussian filters is an open question. In particular, we could ask whether g_k is

optimal among all length $k+1$ filters.

The other question is the size of the low-pass filter used to build the pyramid. A larger filter—one with a larger spatial extent— gives more high frequency attenuation. While this further decreases aliasing artifacts, it reduces actual detail in the images. Motion detection based on local image structure may be fooled by aliasing artifacts on the one hand, or by the reduction of detail on the other. Thus there is a size-of-discrete-Gaussian vs. motion-detection-accuracy tradeoff that must be determined and optimized.

2.2 Hierarchical Correlation

In Chapter IV our experiments show that hierarchical matching provides an excellent method for the computation of displacement fields. However, a thorough evaluation of the effects of various parameters and algorithmic options on the accuracy of the computed fields has yet to be carried out. These include:

- *The size of the sample window*: Can windows as small as 3×3 provide adequate matches ?
- *The "shape" of the sample window*: How do center-weighted windows (e.g., Gaussian windows [Burt *et al.* 82, Quam 84]) improve accuracy when the disparity field is not smoothly varying ?
- *The method of computing the bandpass pyramids*: This issue was mentioned in the previous section.
- *The use of normalized correlation*: Bandpass filtering and the small 3×3 search areas seem to eliminate the need to do normalized correlation. This may not remain true if sample windows smaller than 8×8 are used.

While the absolute accuracy (i.e., percent correct) of the displacement field is important, it is the subsequent use of the motion information in specific applications that ultimately determines acceptable error tolerances. For example, in image registration applications, the statistical distribution of errors is more significant than

errors at specific points. On the other hand, some of the structure from motion algorithms require highly accurate displacements at a few specific points. Hence, a study of the *evaluation criteria* of the displacement fields with careful consideration of the application domains is an important area of future work.

Another important research problem is a systematic study of interest operators and sharpness measures. It is necessary to understand how they relate to the degree of confidence in the displacements obtained at image points. Sharpness measures appear to be especially useful when motion boundaries need to be detected. A good example of this is found in P. Anandan's confidence measures [Anandan 87].

When interest operators and/or sharpness measures are used, good motion estimates may be sparsely distributed, and some sort of interpolation is then necessary to obtain dense displacement fields. Typically, these involve applying smoothing or interpolation processes to the displacement fields (see [Anandan 87]). In the processing cone, interpolation is best done by an iterative smoothing process (see Chapter VII, Section 4) since this is a local, uniform, and parallel algorithm. As we discussed in Chapter VII, this may require many iterations of the basic relaxation step. In this case, multilevel relaxation can be applied to increase the convergence rate. Hierarchical matching and interpolation could then be performed together in the processing cone.

2.3 Hierarchical Gradient-Based Methods

Basic gradient-based motion computation algorithms, including our hierarchical extension, assume that the image data and the motion vector field are smoothly varying, that is, they have no sharp changes or discontinuities. This is not the case at a motion boundary—an occlusion boundary at which the image motion is different on either side. At a motion boundary two problems arise. First, the discontinuity in the image data due to the occlusion edge prevents the correct computation of the first differences, since these computations require image data over a neighborhood. This causes incorrect estimation of the velocity constraint line. The second problem is that, at discontinuities in the motion vector field, the

smoothness assumption is invalid, hence the relaxation stage is not appropriate. If smoothing takes place across this boundary, then vectors describing unrelated motions are combined giving an "average" field that is not incorrect on either side.

To deal with motion boundaries, the basic relaxation method must be extended. This involves three additions to the algorithm: (1) detect motion boundaries, (2) ignore incorrect constraint lines at these boundaries, and (3) avoid smoothing across these boundaries. The only significant problem is the first one since the second step has already been addressed in our hierarchical algorithm and the third has a straightforward solution.

The second step (ignore incorrect constraint lines) was needed in the hierarchical algorithm because bad constraint line estimates can also be caused by noise in the underlying image data. The solution there was to detect bad constraint lines by their large edge flow vectors (hence the EDGE-FLOW-TOO-BIG error condition), and to avoid use of the constraint line in the update equation. The latter was straightforward because the update operation cleanly splits into two steps (1) computation of neighborhood average vectors, followed by (2) projecting them towards the corresponding velocity constraint lines. When the constraint line estimate is incorrect, the second step is skipped and update reduces to merely smoothing.

The third addition (avoidance of smoothing across motion boundaries) is readily implemented by "truncating" the local neighborhood used to compute the local average of the motion vector field. That is, only those vectors belonging to pixels on one side of the motion boundary are averaged together.

It remains to specify how the motion boundaries can be detected. At a motion boundary, the constraint line estimate is likely to be bad. Hence, we could use the EDGE-FLOW-TOO-BIG error to detect this. The disadvantage of this is that bad constraint lines are also detected due to noise in the image data. While we should not use the constraint line in the update operation irrespective of what caused the bad estimate, it is not necessary to truncate the smoothing neighborhood if there is not a motion boundary. In fact, in a noisy image, where many constraint lines are judged invalid, this might greatly reduce the sharing of motion information hence

slowing or even preventing computation of the vector field.

The essential problem here is that it is not easy to distinguish between motion boundaries and noisy image data by examining the image data. Instead we suggest that the detection of motion boundaries be delayed until some aggregation of motion information has taken place. To do this, we consider the application of Terzopolous' method for detecting discontuities in a multigrid relaxation algorithm for surface interpolation [Terzopolous 84]. Terzopolous does this by first solving the problem with no discontinuities, then flagging zero-crossings with large gradient magnitudes as (potential) discontinuites, and recomputing the problem using the smooth solution as an initial estimate but not smoothing across discontinuities.

Terzopolous' method cannot be directly applied to the optic flow relaxation problem because actual motion boundaries cause incorrect constraint line estimates which should not be used in the relaxation updating. However, we have already discussed how this problem can be avoided, namely, incorrect constraint line estimates can be detected and consequently not used in the update equation. This could then be combined with Terzopolous' discontinuity detection method, giving the following extended algorithm: (1) detect bad constraint line estimates and flag the pixels; (2) relax for a few iterations, not using constraint lines where they are flagged as bad; (3) look for sharp changes in the current estimated solution field, mark these locations as possible motion edges; (4) relax for a few iterations, not using bad constraint lines, and not smoothing across possible motion edges; repeat steps 3 and 4.

2.4 Multilevel Optic Flow Relaxation

We have shown experimently that convergence/divergence in multilevel optic flow relaxation is related to spatial variation in the image data. For the test images we have used, variation has been measured by the spatial frequency of the sinusoidal data. In general, spatial frequency is measured globally. In our locally-connected parallel processing cone, such global measures are not efficiently computed. A locally computable measure is desired.

We can suggest two lines of investigation along which suitable measures might be found. First, consider the notion that at higher (coarser) levels of the pyramid the variation of the coefficient data (which itself derives from variation of the image data) cannot be adequately represented, and that this in turn causes divergence in the relaxation update at those levels. (The *coefficients* of the optic flow PDE are the elements of the two-dimensional differential operator.) An analysis we have performed of coefficient pyramids suggests that the appropriateness of coarse representations can be measured by: (1) the local variance of the spatial gradients; (2) the laplacian of the image data; or, combining these two measures, (3) the local variance of the laplacian of the image. These suggestions are based on the idea that the discrete representation of the optic flow PDE is valid wherever the low-pass pyramid built from the high resolution coefficients is similar to a coefficient pyramid obtained by computing the coefficients separately at each level of the low-pass image pyramid.

The second method (using the local image structure to determine when coarse relaxation will diverge) considers the variation in the spatial gradient at each level. Variation in the direction of the gradient is directly related to the rate of convergence. Specifically, if the variation of the gradient direction is high, then nearby velocity constraint lines have very different directions, and convergence to actual optic flow is faster. The exact relationship is not known quantitatively. In any case, we are led to consider various measures of local variations in the gradient. Any such measure should be related to the Hessian $H = \nabla(\nabla F)$ of the image data $F(x, y)$, since the Hessian, being a generalized second derivative of the two-dimensional image function, is equivalently a first derivative of the gradient $G = \nabla F$.

The first two measures that suggest themselves are the trace $Tr(H) = F_{xx} + F_{yy}$ and the determinant $Det(H) \triangleq F_{xx}F_{yy} - F_{xy}^2$ of the Hessian matrix (see Appendix A). Note that the trace is equal to the Laplacian. These two measures are invariants of the Hessian; they do not change under coordinate rotations. When $\|\nabla F\|$ is small, the *Det* is approximately equal to the Gaussian curvature of the image "surface", and the trace is approximately the mean curvature of the surface. High curvature

in either of these measures is a sign of variation in the direction of the gradient.

A more direct approach involves computation of the local variation of the gradient direction, by deriving an expression for the *gradient of the direction of the gradient*—the *GDG*. The direction of the gradient is given by the angle $\theta = \arctan(F_y/F_x)$. The gradient of this is

$$\nabla\theta = (\theta_x, \theta_y) = \frac{1}{F_x^2 + F_y^2}\left(F_x F_{xy} - F_y F_{xx}, F_x F_{yy} - F_y F_{xy}\right) = \frac{G_\perp^T H}{\|G\|^2}$$

where $G_\perp = (-F_y, F_x)$. To get a single number describing the local variation of the gradient direction, we can take the magnitude of $\nabla\theta$. An alternative is to apply $\nabla\theta$, which is a linear operator, to a specific direction vector, to get a directional first derivative of θ. If this direction vector is chosen parallel to $G_\perp$, then it is easily shown that this turns out to be equal to the Kitchen interest operator [Kitchen & Rosenfeld 82]:

$$\begin{aligned} K &\triangleq -\frac{F_{xx}F_y^2 - 2F_{xy}F_xF_y + F_{yy}F_x^2}{(F_x^2 + F_y^2)^{3/2}} \\ &= -\frac{\overline{G}_\perp^T H \overline{G}_\perp}{|G|} = -\nabla\theta \cdot \overline{G}_\perp \end{aligned}$$

where $\overline{G}_\perp$ is the unit vector perpendicular to the gradient $G = \nabla F$. The numerator is the second directional derivative of F in the direction perpendicular to the gradient.

The determinant $Det(H)$, the *GDG* and Kitchen's operator all involve the multiplication of differentials, the measurements of which are susceptible to noise. Thus, their use in detecting general areas of local variation is questionable. On the other hand, the trace of the Hessian ($Trace(H)$), which is in fact equal to the Laplacian, does not involve such multiplication. Moreover, the Laplacian was also suggested as a measure in the discussion of coefficient pyramids at the beginning of this section. The Laplacian is also the easiest to compute of all the measures we have mentioned. For these reasons, the Laplacian (applied to the image data) should first be studied as measure for detecting when optic flow relaxation will diverge at coarser levels.

APPENDIX A

Spatial Frequency Representation and Transfer Functions

The spatial frequency representation of image structure allows us to formalize the concept of gross vs. detailed structure. The qualitative gross vs. detailed distinction is represented quantitatively as the *spectrum* of spatial frequencies. The Fourier Transform of an image explicitly describes the relative frequency content of an image. The Fourier Transform of a convolution mask of an image filter — its Transfer Function — describes the way in which the filter passes specific spatial frequencies. These ideas are developed in this appendix with an emphasis upon those of most relevance to us. More detailed and complete developments can be found in [Bracewell 78] and [Dudgeon & Mersereau 84].

1 Image Filters

A **discrete image** $f(m,n)$ is a function $f : Z^2 \rightarrow \Re$ where $Z^2 \triangleq \{(m,n) \mid m,n \text{ integers}\}$. Z^2 represents a regular rectangular grid of sample points with a spacing between adjacent points of $h = 1$. In Section 7 the generalization to arbitrary grid spacings h is made. An **image filter** φ takes any discrete image f to another discrete image φf. A **linear** image filter satisfies:

$$\begin{aligned} \varphi(f_1 + f_2) &= \varphi f_1 + \varphi f_2 \\ \varphi(af_1) &= a\varphi f_1 \qquad \text{for any } a \in \Re \end{aligned}$$

An image f can be shifted by some translation vector (m_0, n_0) to give a new image $g(m,n) = f(m - m_0, n - n_0)$. A filter φ is **shift invariant** if for any such f and g, φg is a shifted version of φf, that is, $(\varphi g)(m,n) = (\varphi f)(m - m_0, n - n_0)$.

A **linear shift invariant** (LSI) filter is both linear and shift invariant. Its action on an image is equivalent to convolution. We show that now. First define the **delta function** as

$$\delta(m,n) \triangleq \begin{cases} 1 & \text{if } m = 0, n = 0 \\ 0 & \text{otherwise} \end{cases}$$

Now define the **impulse response** of an LSI filter φ as $h = \varphi\delta$. Any function f can be represented as a weighted sum of delta functions as follows:

$$f(m,n) = \sum_{(u,v)\in Z^2} f(u,v)\delta(m-u, n-v)$$

Now apply an LSI filter to f:

$$\begin{aligned} \varphi f(m,n) &= \sum_{(u,v)\in Z^2} f(u,v)\varphi[\delta(m-u,n-v)] && \text{(by linearity)} \\ &= \sum_{(u,v)\in Z^2} f(u,v)h(m-u,n-v) && \text{(by shift invariance)} \end{aligned}$$

. The right hand side of the last line is by definition the convolution of f and h (see Section 5), so we have: *if φ is an LSI filter with impulse response $h(m,n)$ then $\varphi f = f * h$.* Thus, we also call h a **convolution mask** or "mask" for short.

2 The Fourier Transform

The spatial frequency representation is defined formally by the Fourier Transform (FT) and the Inverse Fourier Transform (IFT). A discrete image $f(m,n)$ and its transform $F(s,t)$ are related by

$$\begin{aligned} \mathcal{F}(f) &= F(s,t) = \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} f(m,n)e^{-i2\pi(ms+nt)} \\ \mathcal{F}^{-1}(F) &= f(m,n) = \int_{-\frac{1}{2}}^{+\frac{1}{2}}\int_{-\frac{1}{2}}^{+\frac{1}{2}} F(s,t)e^{i2\pi(sm+tn)}\,dsdt \end{aligned}$$

$F(s,t)$ is a complex function over the two-dimensional s,t *frequency domain.* $F(s,t)$ is defined and continuous when f is absolutely summable, that is, when $\sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}|f(m,n)| = S < \infty$. $F(s,t)$ is also defined when f is square summable, that is, when $\sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}|f(m,n)|^2 = S < \infty$, but F(s,t) may have discontinuities in this case.

We call the value $F(s,t)$ the **Fourier coefficient** of the image $f(m,n)$ at the spatial frequency (s,t). In a sense, this coefficient defines the relative "content" of the image at that specific frequency. In the IFT integral, $f(m,n)$ is expressed as a "weighted" sum of spatial frequencies, where the Fourier coefficients $F(s,t)$ are the weights, and the complex functions $\varepsilon_{st}(m,n) \triangleq e^{i2\pi(sm+tn)}$ are the spatial frequency functions.

When, as in our case, the images are real-valued, the Fourier Transform can be interpreted as defining the image as a combination of strictly real-valued spatial frequency functions. If the image f is real-valued, then it is easy to show (see Section 4.2) that $F(s,t) = F^*(-s,-t)$ where the z^* is defined as the *complex conjugate* of z. (We thus say that such an FT is *Hermitian about the origin* in the frequency domain.) Now noting that $\varepsilon_{-s,-t}(m,n) = \varepsilon^*_{st}(m,n)$, and using the polar form $F(s,t) = \rho e^{i\theta}$ we can show that

$$\begin{aligned} F(s,t)\varepsilon_{st}(m,n) + F(-s,-t)\varepsilon_{-s,-t}(m,n) &= 2\mathrm{Re}[F(s,t)\varepsilon_{st}(m,n)] \\ &= \rho\cos(2\pi(ms+nt)+\theta) \end{aligned}$$

This final equation shows us that the pair of Fourier coefficients $F(s,t)$ and $F(-s,-t)$ represent the amplitude (ρ) and phase shift (θ) of the component of the (real-valued) image f at the spatial frequency (s,t).

3 The Baseband

The Fourier Transform of a discrete image defined over the sampling grid Z^2 is periodic with two-dimensional period $(1,1)$, that is, $F(s,t) = F(s+p,t+q)$ for any $(p,q) \in Z^2$. This periodicity is a function of the sampling grid. The Fourier Trans-

form is completely specified by its values in one period. In fact the Inverse Fourier Transform is given by an integral over $B_1 \triangleq [-\frac{1}{2}, \frac{1}{2}]^2$, a rectangular area which tiles the (s,t) frequency space with the period $(1,1)$. We call B_1 the **baseband** corresponding to the sampling grid.

4 Basic Properties

4.1 Linearity

If $\mathcal{F}(f) = F(s,t)$ *and* $\mathcal{F}(g) = G(s,t)$, *then* $\mathcal{F}(f+g) = F(s,t) + G(s,t)$.

If $\mathcal{F}(f) = F(s,t)$ *and* $a \in \Re$, *then* $\mathcal{F}(af) = aF(s,t)$.

4.2 Symmetry Theorems

If a real-valued function is symmetric about the origin ($f(m,n) = f(-m,-n)$), then its Fourier Transform is purely real-valued. This follows from the more general fact that a *function is Hermitian about the origin* ($f^*(m,n) = f(-m,-n)$) *if and only if it has a real-valued transform.* (The definition of image functions is readily extended to include complex valued functions: $f : Z^2 \to C$.)

Proof of $\Rightarrow$ case:

$$\begin{aligned} F^*(s,t) &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} f^*(m,n) e^{i2\pi(ms+nt)} \\ &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} f(-m,-n) e^{-i2\pi(-ms-nt)} \\ &= \sum_{m'=-\infty}^{+\infty} \sum_{n'=-\infty}^{+\infty} f(m',n') e^{-i2\pi(m's+n't)} \\ &= F(s,t) \end{aligned}$$

Proof of $\Leftarrow$ case:

$$\begin{aligned} f^*(m,n) &= \int_{-\frac{1}{2}}^{+\frac{1}{2}} \int_{-\frac{1}{2}}^{+\frac{1}{2}} F^*(s,t) e^{-i2\pi(ms+nt)}\, dsdt \\ &= \int_{-\frac{1}{2}}^{+\frac{1}{2}} \int_{-\frac{1}{2}}^{+\frac{1}{2}} F(s,t) e^{i2\pi(-ms-nt)} \end{aligned}$$

$$= f(-m,-n)$$

A similar proof shows that a *function is real-valued if and only if its Fourier Transform is Hermitian.*

4.3 Separability

If a function is separable then its Fourier Transform is also. Moreover the transform is equal to the product of the (one-dimensional) transforms of the "components" of the function. More formally, if $f(m,n) = f_1(m)f_2(n)$ then $F(s,t) = F_1(s)F_2(t)$ where $F = \mathcal{F}(f)$, $F_1 = \mathcal{F}(f_1)$ and $F_2 = \mathcal{F}(f_2)$. The one-dimensional Fourier Transform and its inverse are defined as:

$$\mathcal{F}(f) = F(s) = \sum_{n=-\infty}^{+\infty} f(n)e^{-i2\pi ns}$$

$$\mathcal{F}^{-1}(F) = f(n) = \int_{-\frac{1}{2}}^{+\frac{1}{2}} F(s)e^{i2\pi ns}\,ds$$

4.4 Stretching Theorem

Define the **stretch** of a function $f(m,n)$ by an integral amount $a \in \mathcal{N}$ as:

$$S[f;a](m,n) \triangleq \begin{cases} f(m/a, n/a) & \text{if } m/a, n/a \in Z \\ 0 & \text{otherwise} \end{cases}$$

The the **stretching theorem** is: *if* $\hat{f} = S[f;a](n)$ *and* $F(s) = \mathcal{F}(f)$, *then* $\hat{F}(s,t) = \mathcal{F}(\hat{f}) = F(as,at)$. The derivation is:

$$\begin{aligned} \hat{F}(s,t) &= \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} \hat{f}(m,n)e^{-i2\pi(ms+nt)} \\ &= \sum_{m\in aZ}\sum_{n\in aZ} f(m/a,n/a)e^{-i2\pi[(m/a)(as)+(n/a)(at)]} \\ &= \sum_{m'\in Z}\sum_{n'\in Z} f(m',n')e^{-i2\pi[(m')(as)+(n')(at)]} \\ &= F(as,at) \end{aligned}$$

5 The Convolution Theorem

The convolution of two discrete functions $f(m,n)$ and $g(m,n)$ is given by:

$$f * g(m,n) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)g(m-u,n-v)$$

It is easy to show that convolution is both associative $f * (g * h) = (f * g) * h$ and commutative $f * g = g * f$.

The **Convolution Theorem** tells us that convolving two functions is the same as multiplying their transforms. That is, *if* $\mathcal{F}(f) = F(s,t)$ *and* $\mathcal{F}(g) = G(s,t)$, *then* $\mathcal{F}(f * g) = F(s,t)G(s,t)$. The derivation is:

$$\begin{aligned}
\mathcal{F}(f * g)(s,t) &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \left[\sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)g(m-u,n-v) \right] e^{-i2\pi(ms+nt)} \\
&= \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v) \left[\sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} g(m-u,n-v)e^{-i2\pi(ms+nt)} \right] \\
&= \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)e^{-i2\pi(us+vt)}G(s,t) \\
&= F(s,t)G(s,t)
\end{aligned}$$

A change of polarity in the exponent of the exponentials above and replacement of the summations with a double integral over the baseband proves the **Multiplication Theorem**: multiplying two functions is the same as convolving their transforms. That is, *if* $f(x,y)$ *has the Fourier Transform* $F(s,t)$ *and* $g(x,y)$ *has the Fourier Transform* $G(s,t)$, *then* $f(x,y)g(x,y)$ *has the Fourier Transform* $F(s,t) * G(s,t)$.

6 Transfer Functions

The Convolution Theorem tells us that convolving two functions is the same as multiplying their transforms. That is, if $f(m,n)$ has the Fourier Transform $F(s,t)$ and $h(m,n)$ has the Fourier Transform $H(s,t)$, then $h * f$ has the Fourier Transform $H(s,t)F(s,t)$. Thus, an LSI image filter, whose operation can be described as

a convolution, can also be described as multiplication in the frequency domain. We define the **transfer function** of an image filter as the Fourier Transform of its impulse response. The transfer function $H(s,t)$ of a mask $h(m,n)$ is complex valued and can be represented as a gain and phase shift for each spatial frequency, viz. $H(s,t) = A(s,t)e^{i\theta(s,t)}$.

If the mask is real-valued and symmetric about the origin, i.e. $h(m,n) = h(-m,-n)$, then its transfer function is real-valued (Section 4.2). In this case $e^{i\theta(s,t)}$ must be 1 or -1 and hence the phase shift $\theta(s,t)$ is zero or π respectively. A phase shift of π can be represented as a negative gain. Shifting the phase of a spatial frequency function by π radians is equivalent to multiplying it by -1 since $e^{\pi} = -1$. Thus, a gain of A and a phase shift of π is equivalent to a gain of $-A$ and phase shift of 0. We use this representation so that we need only show the amplitude of transfer functions of real-valued symmetric masks.

7 General Discrete Image Spaces

Discrete images are functions defined over a regularly spaced grid of points. To this point, we have used the grid Z^2, a regular rectangular array of points with unit spacing. In order to consider *multiresolution* representations we must generalize our definition of discrete images to include varying sample spacing in the grid. First we define the grid with sample spacing h as $(hZ)^2 = hZ \times hZ$ where $hZ \triangleq \{hn \mid n \in Z\}$. h can be any positive real number. Then a **discrete image** is a function $f : (hZ)^2 \rightarrow \Re$. The definitions of *linear shift invariant*, *delta function*, *impulse response*, and *convolution* have straightforward extensions. For example **convolution** is defined as

$$f * g(m,n) = \sum_{u \in hZ} \sum_{v \in hZ} f(u,v)g(m-u, n-v) \qquad m,n \in hZ$$

The Fourier Transform and its inverse are now given by:

$$\mathcal{F}(f) = F(s,t) = \sum_{m\in hZ}\sum_{n\in hZ} f(m,n)e^{-i2\pi(ms+nt)}$$

$$\mathcal{F}^{-1}(F) = f(m,n) = \int_{-\frac{1}{2h}}^{+\frac{1}{2h}}\int_{-\frac{1}{2h}}^{+\frac{1}{2h}} F(s,t)e^{i2\pi(sm+tn)}\,dsdt \qquad m,n\in hZ$$

The baseband is now $B_h \triangleq [-\frac{1}{2h},\frac{1}{2h}]^2$.

All of the basic properties in Section 4 still hold. For the Stretching Theorem, we generalize the definition of the **stretch** of a function $f(m,n)$ over $(hZ)^2$ by an integral amount $a\in N$ to:

$$S[f;a](m,n) \triangleq \begin{cases} f(m/a,n/a) & \text{if } m/a,n/a\in hZ \\ 0 & \text{otherwise} \end{cases}$$

The Convolution Theorem also remains in effect. Its proof in Section 5 is easily generalized by replacing the sums over Z with sums over hZ. Finally, we can define Transfer Functions as before and interpret the action of an LSI image filter as multiplication by a frequency filter.

8 Subsampling and Aliasing

If we wish to sample an image on a coarser grid, then certain high frequencies can no longer be represented, namely, those frequencies in the baseband of the finer grid which are not in the baseband of the coarser grid. If a coarse image is formed by subsampling a finer image then frequencies which lie outside the coarse baseband appear as lower frequencies within that baseband. This is known as *aliasing*. In this section (1) the relationship between the Fourier Transforms of an image and a subsampled version of that image is derived, (2) we show how aliasing occurs in the processing cone, and (3) ideal anti-aliasing filtering is described.

8.1 Subsampling Theorem

The subsampling theorem says that an image g derived by sampling values of another image f on a finer grid has a transform G which is equal to a sum of shifted

versions of the transform F of the higher resolution image.

Theorem: *If $f : Z^2 \rightarrow \Re$ is an image over a grid with unit spacing, $g : (nZ)^2 \rightarrow \Re$ is an image over a grid with spacing $n \in Z$ such that $g(nk, nl) = f(k,l)$ for all $(k,l) \in Z^2$, and $F(s,t) = \mathcal{F}(f)$ and $G(s,t) = \mathcal{F}(g)$ are the Fourier transforms, then*

$$G(s,t) = \frac{1}{n^2} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} F(s - \frac{k}{n}, t - \frac{l}{n})$$

We prove the one-dimensional case. The two-dimensional case is a straightforward generalization.

Proof:

$$\begin{aligned} \sum_{k=0}^{n-1} F(s - \frac{k}{n}) &= \sum_{k=0}^{n-1} \sum_{m=-\infty}^{+\infty} f(m) e^{-i2\pi m(s - \frac{k}{n})} \\ &= \sum_{m=-\infty}^{+\infty} f(m) e^{-i2\pi ms} \sum_{k=0}^{n-1} e^{i2\pi \frac{m}{n} k} \end{aligned}$$

The inner summation is easily shown to be equal to n if $m/n \in Z$ (i.e. if n divides m, also written as $n|m$) and 0 otherwise. So we have

$$\sum_{k=0}^{n-1} F(s - \frac{k}{n}) = n \sum_{\substack{m=-\infty \\ n|m}}^{+\infty} f(m) e^{-i2\pi ms}$$

The summation on the right is equivalent to $\sum_{m \in Z}$ and we can substitute $g(m)$ in for $f(m)$. So we finally have

$$\frac{1}{n} \sum_{k=0}^{n-1} F(s - \frac{k}{n}) = G(s)$$

□

8.2 Subsampling in the Processing Cone

In the processing cone, subsampling up one level corresponds to the case $n = 2$ in the sampling theorem. Consider the two bottom levels of the cone L and $L-1$ where the grid spacings are $h_L = 1$ and $h_{L-1} = 2$. The baseband for level L is

$B_1 = [-\frac{1}{2}, \frac{1}{2}]^2$ and for level $L-1$ it is $B_2 = [-\frac{1}{4}, \frac{1}{4}]^2$. The sampling theorem tells us how the spatial frequency content of an image at level L determines that of a subsampled image at level $L-1$.

Let f be an image at the finest level L ($h_L = 1$) and let g be a subsampling of f one level up $L-1$ ($h_{L-1} = 2$). If $F(s,t) = \mathcal{F}(f)$ and $G(s,t) = \mathcal{F}(g)$, then by the subsampling theorem $G(s,t) = \frac{1}{4}[F(s,t) + F(s-\frac{1}{2},t) + F(s,t-\frac{1}{2}) + F(s-\frac{1}{2},t-\frac{1}{2})]$. Consider $(s,t) \in B_2 = [-\frac{1}{4}, \frac{1}{4}]^2$. If we define $\hat{s}$ as

$$\hat{s} \triangleq \begin{cases} s - \frac{1}{2} & \text{if} \quad 0 \leq s \leq \frac{1}{4} \\ s + \frac{1}{2} & \text{if} \quad -\frac{1}{4} \leq s < 0 \end{cases}$$

and we similarly define $\hat{t}$, then $(\hat{s},t)$, $(s,\hat{t})$ and $(\hat{s},\hat{t})$ are all in $B_2 - B_1$. Using the periodicity of F, we have $G(s,t) = \frac{1}{4}[F(s,t) + F(\hat{s},t) + F(s,\hat{t}) + F(\hat{s},\hat{t})]$. Thus, the spatial frequency content of the subsampled image g at some $(s,t) \in B_2$ is determined by the content of f at (s,t), and at three other frequencies $(\hat{s},t)$, $(s,\hat{t})$ and $(\hat{s},\hat{t})$ in $B_1 - B_2$.

8.3 Ideal Anti-Aliasing Filtering

Aliasing is the addition of spatial frequency content at (s,t) in the image g due to content at the frequencies $(\hat{s},t)$, $(s,\hat{t})$ and $(\hat{s},\hat{t})$ in the image f. Aliasing can be eliminated if ideal low-pass filtering is performed. Such a filter passes all frequencies within the coarse baseband and none outside of it. For the case of subsampling in the cone, the spatial filter with this property is the product of two sampled sinc functions.

The one-dimensional (continuous) sinc function is defined as $\text{sinc}(x) \triangleq \frac{\sin(\pi x)}{\pi x}$. A two-dimensional separable outer product of sincs is given by $\text{sinc}_2(x,y) \triangleq \text{sinc}(x)\text{sinc}(y)$. Define the rectangular pulse $\Pi(x,y)$ to be equal to 1 when $(x,y) \in B_1 = [-\frac{1}{2}, \frac{1}{2}]^2$ and to 0 otherwise. Then $\Pi(2x,2y)$ is equal to 1 over $B_2 = [-\frac{1}{4}, \frac{1}{4}]^2$ and 0 elsewhere. The Fourier Transform of the (continuous) function $\text{sinc}_2(x)$ is $\Pi(s,t)$. The Fourier Transform of the (continuous) function $\frac{1}{2}\text{sinc}_2(x/2)$ is $\Pi(2s,2t)$.

The sampled sinc function $\frac{1}{4}\text{sinc}_2(m/2, n/2)$ has the Fourier Transform

$$\mathcal{F}(\frac{1}{4}\text{sinc}_2(m/2,n/2)) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \Pi(2(s-k),2(t-l))$$

$$\triangleq \begin{cases} 1 \text{ if } \begin{matrix} k-\frac{1}{4} \le s \le k+\frac{1}{4} \\ l-\frac{1}{4} \le t \le l+\frac{1}{4} \end{matrix} \text{ for some } (k,l) \in Z^2 \\ 0 \text{ otherwise} \end{cases}$$

This is just a periodic repetition over $\Re^2$ of the function $\Pi(2s, 2t)$ over the base-band B_1. The discrete one-dimensional sinc function $\text{sinc}(n/2)$ is non-zero whenever n is even and non-zero. The discrete two-dimensional sinc function $\text{sinc}_2(m/2, n/2)$ is non-zero whenever both m and n are even and non-zero.

APPENDIX B

Edge Flow from Edges in Three Dimensional XYT Space

Discontinuities (edges) in a static image $I(x,y)$ result from discontinuities in the imaging factors, e.g., occlusion of objects, shadowing (occlusion of lighting), surface markings, and internal edges of objects. In a dynamic image $I(x,y,t)$ these factors vary in time and we get discontinuities in $I(x,y,t)$. These discontinuities can be viewed as two-dimensional surfaces swept out by the edges in successive time-slices of $I(x,y,t)$. We refer to these two-dimensional discontinuities in a three-dimensional space as **2D-edges**, while we call edges in a two-dimensional image **1D-edges**. These 2D-edges contain information about both the location of 1D-edges in a single frame and the velocity of these edges in the xy image plane.

A small local piece of a 1D-edge is essentially linear and a small local piece of a 2D-edge is essentially planar. Hereafter *1D-edge* and *2D-edge* will refer specifically to these small local pieces. A 1D-edge is the intersection of a 2D-edge with an xy plane (a plane of constant t). Conversely, successive occurrences of a 1D-edge sweep out a 2D-edge. The direction (orientation) of the 1D-edge in the xy plane and its velocity determine the orientation of the 2D-edge.

Let a 1D-edge E be represented in polar coordinates (ρ,θ) as in Figure 81. In the xyt coordinate system this normal form defines the "static" unit vectors $U_\perp = (\cos\theta, \sin\theta, 0)$ and $U_\parallel = (\sin\theta, -\cos\theta, 0)$ in the xy plane which are perpendicular and parallel to E respectively.

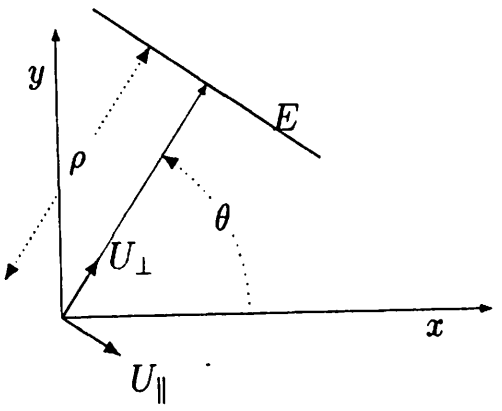


Figure 81: Normal form representation of edges

The edge E has the normal form of the line on which it lies, viz., (ρ, θ) where ρ is the length of the perpendicular dropped to the origin and θ is the angle between that perpendicular and the x axis.

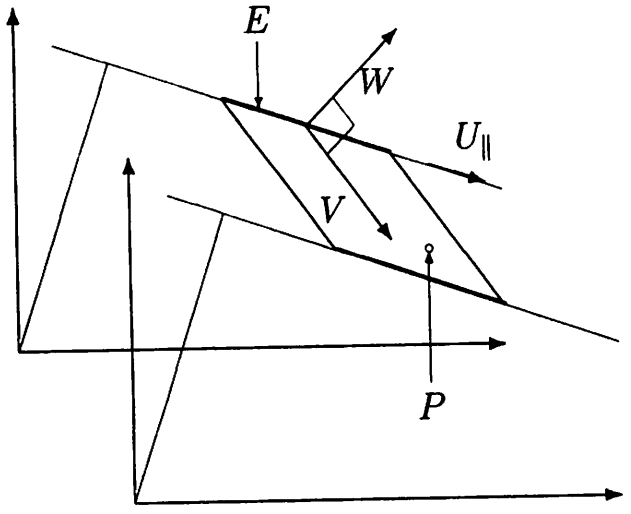


Figure 82: Moving edges in xyt space

A 1D-edge E moving with velocity V sweeps out a 2D-edge P. P is normal to the vector $W = V \times U_\parallel$.

Let the velocity of E be the vector $V = (V^x, V^y, 1) = d/dt(x, y, t)$. The 2D-edge P swept out by E moving with velocity V is parallel to the vectors $U_\parallel$ and V (see Figure 82). Thus, it is perpendicular to their cross product $W = V \times U_\parallel = (\cos\theta, \sin\theta, -V^x \cos\theta - V^y \sin\theta)$. This vector defines the orientation of P. Note that the third component of W is equal to $-V \cdot U_\perp$. The inner product $V \cdot U_\perp$ is the length of the projection of V onto $U_\perp$.

If we are given $P = (P^x, P^y, P^t)$ a vector perpendicular to the 2D-edge P, it must be a multiple of W, viz., $pP = W$. By noting that the projection of W into the xy plane is equal to $U_\perp$, a unit vector, we see that $p = 1/\sqrt{(P^x)^2 + (P^y)^2}$. We then have $-V \cdot U_\perp = pP^t$, from which we get

$$V \cdot U_\perp = \frac{-P^t}{\sqrt{(P^x)^2 + (P^y)^2}} \tag{94}$$

This is the magnitude of the component of the velocity of the 1D-edge perpendicular to that edge. The component of velocity parallel to the 2D edge cannot be computed from the 3D edge. This local ambiguity in the detection of optic flow can be demonstrated geometrically in the xy plane (see Figure 83).

The geometric interpretation of local ambiguity in three-dimensional xyt space is seen in the fact that the locus of velocity vectors V that would generate P (along with a given $U_\parallel$) is the intersection of P with the plane $t = 1$. This is a line of solutions, so one degree of freedom remains to be specified.

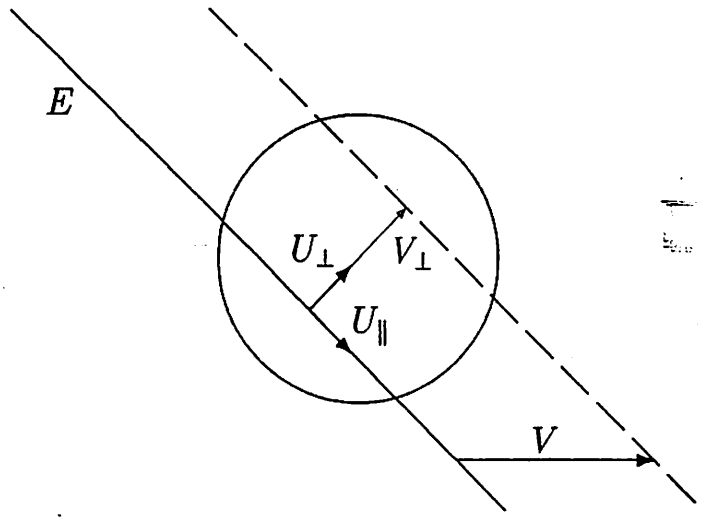


Figure 83: The local ambiguity of a moving edge

The 2D edge E is shown moving with velocity V. $U_\perp$ is the unit vector perpendicular to E and $U_\parallel$ is the unit vector parallel to E. Only the component of V parallel to $U_\perp$, namely $V_\perp$, can be detected. We have that $V_\perp = (V \cdot U_\perp)U_\perp$.

APPENDIX C

The Pseudo-Intersection Method

When two or more edge flow vectors that are not all parallel are given for a single optic flow vector, the velocity constraint lines intersect at the actual flow vector. In practical applications these lines cannot be expected to intersect at a unique point. Imaging errors, discrete sampling, various noise sources, etc. introduce alterations in the data and produce incorrect measurements of the edge flow vectors. Given the measured velocity lines we can look for the point that best "fits" these lines. This is the dual problem to that of best fitting a set of points by a line. As in that problem, there are different ways of specifying what is "best". Such measures are typically some function of the distances between the point(s) and the line(s). The least square error (LSE) reflects a standard intuitive definition of "best fit" and is easy to compute. We define the **pseudo-intersection** as that point which minimizes the total of the squared distances between the point and each of the lines. Figure 18 in Chapter III shows an example of the pseudo-intersection of a set of velocity lines. In this appendix, we show how to compute the pseudo-intersection of a set of velocity constraint lines given their corresponding edge flow vectors.

Suppose edge flow is given in polar form for n velocity constraint lines, viz. $(\rho_i \cos\theta_i, \rho_i \sin\theta_i)$ for $i = 1 \ldots n$. (See Figure 81 in Appendix B.) The ith velocity line L_i is given by the equation $u\cos\theta_i + v\sin\theta_i = \rho_i$. The distance of any point (u, v) in velocity space to L_i is

$$d(L_i, (u, v)) \triangleq u\cos\theta_i + v\sin\theta_i - \rho_i \tag{95}$$

The **pseudo-intersection** is the point which minimizes the total square error

$$E = \sum_{i=1}^{n} \Big(d(L_i, (u,v)) \Big)^2 = \sum_{i=1}^{n} (u \cos\theta_i + v \sin\theta_i - \rho_i)^2 \tag{96}$$

Taking the partial derivatives of E with respect to u and v we get

$$\frac{\partial E}{\partial x} = 2 \sum_{i=1}^{n} (u \cos\theta_i + v \sin\theta_i - \rho_i) \cos\theta_i \tag{97a}$$

$$\frac{\partial E}{\partial y} = 2 \sum_{i=1}^{n} (u \cos\theta_i + v \sin\theta_i - \rho_i) \sin\theta_i \tag{97b}$$

Setting these to zero and distributing out u and v gives two linear equations in two unknowns or equivalently the matrix equation

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} d \\ e \end{bmatrix} \tag{98}$$

where

$$a = \sum_{i=1}^{n} \cos^2\theta_i \qquad b = \sum_{i=1}^{n} \cos\theta_i \sin\theta_i \qquad c = \sum_{i=1}^{n} \sin^2\theta_i \tag{99}$$

$$d = \sum_{i=1}^{n} \rho_i \cos\theta_i \qquad e = \sum_{i=1}^{n} \rho_i \sin\theta_i \qquad f = \sum_{i=1}^{n} \rho_i^2$$

A solution exists if and only if the coefficient matrix is invertible, which is the case if and only if its determinant $ac - b^2$ is non-zero. If $ac - b^2 \neq 0$ then these equations can be solved and the solution is

$$(u,v) = \frac{1}{ac - b^2}(cd - be, ae - bd) \tag{100}$$

If we substitute the solution back into Equation 96 and expand the summation we get

$$E = au^2 + 2buv + cv^2 - 2du - 2ev + f \tag{101}$$

The **root-mean-square-error** $E_{RMS} \triangleq \sqrt{E}/n$ is a good measure of the quality of the fit and is independent of the number of lines fit.

Next we show that *a solution exists if the velocity constraint lines are not all parallel.* We will determine under what conditions the determinant $ac - b^2 = 0$.

Define the two vectors in $\Re^n$, $U \triangleq (\cos\theta_1, \ldots, \cos\theta_n)$ and $V \triangleq (\sin\theta_1, \ldots, \sin\theta_n)$. By Schwartz's Inequality, $(U \cdot V)^2 \leq (U \cdot U)(V \cdot V)$. Now noting that $a = (U \cdot U)$, $b = (U \cdot V)$, and $c = (V \cdot V)$ we get $ac - b^2 \geq 0$. We also know that the equality holds in Schwartz's Inequality only when the two vectors are multiples of each other. That is $ac - b^2 = 0$ if and only if $kU = V$ for some $k \in \Re$. This is saying that for all $i = 1, \ldots, n$, $k\cos\theta_i = \sin\theta_i$ or equivalently $\tan\theta_i = k$. This is true if and only if all of the velocity constraint lines are parallel.

Since the computation of (u, v) as given by Equation 100 involves division by the determinant $ac - b^2$, it is sensitive to errors when that value is small. Geometrically, this corresponds to the case where the velocity constraint lines are close to being parallel. Consider just two lines and their intersection. If the lines are nearly parallel, then small changes in the position or orientation of either will produce very large changes in the location of their intersection. This suggests another measure of the quality of the fit based on the determinant. We define the **dispersion** $D \triangleq \sqrt{ac - b^2}/n$. It can be shown that this number ranges between 0 and $\frac{1}{2}$. It is zero only when all of the lines are parallel and it is $\frac{1}{2}$ when the input lines form two equal sized classes perpendicular to each other. Note that the dispersion is independent of the ρ_i.

APPENDIX D

The Gradient and the Hessian

In the first section of this appendix, we define the gradient and the Hessian of two-dimensional images. In the second section, we show how they are related at corresponding points in an image pair, the correspondence being given by a displacement field.

1 Definitions

If an image is considered as a function of two real variables — the spatial image coordinates — then the gradient and the Hessian of that image function describe the local structure of an image in terms of its first and second derivatives. For a function of two variables the notion of first derivative (of a function of one variable) generalizes to the gradient, while the second derivative generalizes to the Hessian. The gradient and the Hessian are both defined in terms of partial derivatives of the image function with respect to the x and y spatial variables. Partial derivatives of a function at some point are computed from the values of that function in the neighborhood of the point. The gradient and the Hessian thus contain information about the the local structure of an image function. This idea will be formalized in this section.

We can formally consider a continuous image $F(x, y)$ as a map $F : \Re^2 \rightarrow \Re$. The **gradient** of F is then given by $\nabla F \triangleq (F_x, F_y)$ where the subscripts represent partial differentiation. The gradient is a vector valued function over the two-

dimensional image coordinate space, i.e., $\nabla F : \Re^2 \rightarrow \Re^2$. At a given point (x, y) the gradient of the image is a two-valued vector. The following interpretations of the gradient at a given point are interesting:

1. The gradient vector points in the direction of maximum increase of the image function F as we move away from the given point. The magnitude of the gradient vector is proportional to the rate of increase.

2. The gradient defines a linear functional over the two-dimensional space of direction vectors, which when applied to a unit vector, "returns" the corresponding directional first derivative of F at the given point. This linear functional is just the inner product of the direction vector with the gradient vector.

3. The gradient specifies that function which is the best linear (first order) approximation to F at the given point. Namely

$$F(x, y) \approx F(x_0, y_0) + (x - x_0, y - y_0) \cdot \nabla F$$

This is based on the Taylor series expansion of F about the point (x_0, y_0).

The **Hessian** of F is given by $\nabla(\nabla F) \triangleq \begin{bmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy} \end{bmatrix}$. The following interpretations of the Hessian at a given point are interesting:

1. The Hessian defines a quadratic form over the two-dimensional space of direction vectors, which when applied to a unit vector, "returns" the corresponding directional second derivative of F at the given point. This quadratic form is given by $\frac{1}{2}\mathbf{d}^t \nabla(\nabla F)\mathbf{d}$, where $\mathbf{d}$ is a direction vector.

2. The Hessian along with the gradient specifies that function which is the best quadratic (second order) approximation to F at the given point. Namely

$$F(x, y) \approx F(x_0, y_0) + \mathbf{d} \cdot \nabla F + \frac{1}{2}\mathbf{d}^t \nabla(\nabla F)\mathbf{d}$$

where the direction vector is

$$\mathbf{d} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

While the four partial derivatives that comprise the Hessian can change with a change of coordinates in the image space, there are some interesting invariants that can be computed from the Hessian, which relate to local image structure.

- The *eigenvalues* λ_M, λ_m — maximum and minimum respectively. The associated *eigenvectors* are called the *principal directions*. They give the directions along which the second directional derivatives of F attain their maximum and minimum — these values being equal to the eigenvalues. The following table summarizes the correspondence between the eigenvalues and image structure.

	$\lambda_m < 0$	$\lambda_m = 0$	$\lambda_m > 0$
$\lambda_M < 0$	maximum (peak)	—	—
$\lambda_M = 0$	ridge	planar	—
$\lambda_M > 0$	saddle	gully	minimum (pit)

- The determinant $Det \triangleq |\nabla(\nabla F)| = F_{xx}F_{yy} - F_{xy}^2$. The determinant is also equal to $\lambda_M\lambda_m$. If we consider the graph $z = F(x, y)$ as a surface, then when ∇F is small, the determinant of the Hessian is approximately the Gaussian curvature of the surface. *Det* takes on large positive values at local maxima and minima, and large negative values at "saddle points". This suggests its use as a simple interest operator, picking out image points with significant identifying structure [Beaudet 78, Kitchen & Rosenfeld 82].

- The trace $Tr(\nabla(\nabla F)) = F_{xx} + F_{yy}$. This is the Laplacian of F. The trace is also equal to $\lambda_M + \lambda_m$. If we consider the graph $z = F(x, y)$ as a surface, then when ∇F is small, the trace of the Hessian is approximately the mean curvature of the surface.

2 Gradients and Hessians at Corresponding Points

In this section we show that when the disparity field that relates two areas of a pair of images is slowly varying, then the gradient and the Hessian at the corresponding locations in these areas are approximately equal (gradient to gradient and Hessian to Hessian).

The disparity vector (u, v) at a point (x, y) in the first frame F_1 defines the corresponding point $(x+u, y+v)$ in the second frame F_2. That is $F_2(x+u, y+v) = F_1(x, y)$. In general the gradient $G = \nabla F$ and the Hessian $H = \nabla(\nabla F)$ of the two images will NOT be equal at these two points because they depend on the behavior of F_1 and F_2 in neighborhoods of (x, y) and $(x+u, y+v)$ respectively. If the disparity vector field $(u(x, y), v(x, y))$ is constant in a neighborhood of (x, y), then clearly we would have $G_2(x+u, y+v) = G_1(x, y)$ and $H_2(x+u, y+v) = H_1(x, y)$. We would also expect that if (u, v) is approximately constant over a neighborhood of (x, y), then we would have $G_2(x+u, y+v) \approx G_1(x, y)$ and $H_2(x+u, y+v) \approx H_1(x, y)$. This is case and we next show the exact relationship.

Proposition D.1 *Let $F_1(x, y)$ and $F_2(x, y)$ be continuous image functions and $(u(x, y), v(x, y))$ a continuous disparity field such that $F_2(x+u, y+v) = F_1(x, y)$. Let F_1, F_2, u, and v all have first and second order partial derivatives. If we define $G_1 \triangleq \nabla F_1$, $G_2 \triangleq \nabla F_2$, and $H_1 \triangleq \nabla(\nabla F_1)$, $H_2 \triangleq \nabla(\nabla F_2)$, then*

$$G_1 = G_2 + \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} G_2 \tag{102}$$

and

$$H_1 = H_2 + \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} H_2 + \nabla(\nabla u)\frac{\partial F_2}{\partial x} + \nabla(\nabla v)\frac{\partial F_2}{\partial y} \tag{103}$$

Proof: Given that $F_2(x + u(x, y), y + v(x, y)) = F_1(x, y)$, taking partial derivatives and using the chain rule gives:

$$\frac{\partial F_1}{\partial x} = \frac{\partial F_2}{\partial x}(1 + u_x) + \frac{\partial F_2}{\partial y} v_x \tag{104}$$

$$\frac{\partial F_1}{\partial x} = \frac{\partial F_2}{\partial x}u_y + \frac{\partial F_2}{\partial y}(1+v_y) \tag{105}$$

or converting to matrix notation:

$$\nabla F_1 = \begin{bmatrix} 1+u_x & v_x \\ u_y & 1+v_y \end{bmatrix} \nabla F_2 \tag{106}$$

$$= \nabla F_2 + \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \nabla F_2 \tag{107}$$

This immediately gives Equation 102.

Then taking the second partials we get:

$$\begin{aligned}
\frac{\partial^2 F_1}{\partial x^2} &= \frac{\partial^2 F_2}{\partial x^2}(1+u_x) + \frac{\partial F_2}{\partial x}u_{xx} + \frac{\partial^2 F_2}{\partial xy}v_x + \frac{\partial F_2}{\partial y}v_{xx} \\
\frac{\partial^2 F_1}{\partial xy} &= \frac{\partial^2 F_2}{\partial xy}(1+u_x) + \frac{\partial F_2}{\partial x}u_{xy} + \frac{\partial^2 F_2}{\partial y^2}v_x + \frac{\partial F_2}{\partial y}v_{xy} \\
\frac{\partial^2 F_1}{\partial xy} &= \frac{\partial^2 F_2}{\partial x^2}u_y + \frac{\partial F_2}{\partial x}u_{xy} + \frac{\partial^2 F_2}{\partial xy}(1+v_y) + \frac{\partial F_2}{\partial y}v_{xy} \\
\frac{\partial^2 F_1}{\partial y^2} &= \frac{\partial^2 F_2}{\partial xy}u_y + \frac{\partial F_2}{\partial x}u_{yy} + \frac{\partial^2 F_2}{\partial y^2}(1+v_y) + \frac{\partial F_2}{\partial y}v_{yy}
\end{aligned} \tag{108}$$

This can be rearranged into

$$\begin{aligned}
\begin{bmatrix} \frac{\partial^2 F_1}{\partial x^2} & \frac{\partial^2 F_1}{\partial xy} \\ \frac{\partial^2 F_1}{\partial xy} & \frac{\partial^2 F_1}{\partial y^2} \end{bmatrix} &= \begin{bmatrix} 1+u_x & v_x \\ u_y & 1+v_y \end{bmatrix} \begin{bmatrix} \frac{\partial^2 F_2}{\partial x^2} & \frac{\partial^2 F_2}{\partial xy} \\ \frac{\partial^2 F_2}{\partial xy} & \frac{\partial^2 F_2}{\partial y^2} \end{bmatrix} \\
&+ \begin{bmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{bmatrix} \frac{\partial F_2}{\partial x} + \begin{bmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{bmatrix} \frac{\partial F_2}{\partial y}
\end{aligned} \tag{109}$$

which by the definition of the Hessian is equivalent to

$$\nabla(\nabla F_1) = \begin{bmatrix} 1+u_x & v_x \\ u_y & 1+v_y \end{bmatrix} \nabla(\nabla F_2) + \nabla(\nabla u)\frac{\partial F_2}{\partial x} + \nabla(\nabla v)\frac{\partial F_2}{\partial y} \tag{110}$$

which is clearly equivalent to Equation 103. □

We define the disparity field (u, v) to be **slowly varying** or **locally translational** or **locally constant** when both $\| \nabla u \|$ and $\| \nabla v \|$ are small and both $\|\nabla(\nabla u)\|$ and $\|\nabla(\nabla v)\|$ are small. From Equations 102 and 103, we see that when $|u_x|, |v_x| \ll 1$ then $G_1 \approx G_2$ and when we further have that $|u_{xx}|, |u_{xy}|, |u_{yy}| \ll 1$ then $H_1 \approx H_2$.

APPENDIX E

Data Transfer Costs in the Processing Cone

In this appendix we briefly discuss data transfer costs in the processing cone and then derive specific costs for accessing all values in a given $w \times w$ neighborhood of pixels. These transfer cost estimates appear as Equations 116 through 119.

On a single level of the processing cone direct communication channels exist between pixels along row and column nearest neighbor links. The cost of transferring a data value is given as T, measured in units of time, where all data values are assumed to be of equal size. When non-standard size data values are transferred, the relative size differences must be factored into the calculation of transfer cost.

The cost of transferring a data value from pixel (x_1, y_1) to pixel (x_2, y_2) is equal to Td where $d = \|(x_2, y_2) - (x_1, y_1)\|_{CB}$ and $\|\cdot\|_{CB}$ is the **city block norm** defined as $\|(x, y)\|_{CB} \triangleq |x| + |y|$. For example, a 3×3 filtering operation would have a transfer cost of $12T$. It takes T units to get the value at each nearest neighbor, and $2T$ for each of the "corner" values.

1 The Cost of Accessing $w \times w$ Neighborhoods

Hierarchical correlation requires that pixel values from two $w \times w$ neighborhoods be brought together for pair-wise multiplication. Let w be an odd number and let $r = (w - 1)/2$ be the "radius" of the neighborhoods. Suppose at a given pixel

(x_0, y_0), the approximate disparity vector is (a, b). Then the two neighborhoods from which pixel values must be accessed are $\{(x_0 + x, y_0 + y) : |x|, |y| \leq r\}$ in the first image, and $\{(x_0 + a + x, y_0 + b + y) : |x|, |y| \leq r\}$ in the second image. The cost of transferring the second neighborhood is

$$\begin{aligned} C_{T,2} &= \sum_{x=-r}^{r} \sum_{y=-r}^{r} \|(x_0 + a + x, y_0 + b + y) - (x_0, y_0)\|_{CB} \\ &= \sum_{x=a-r}^{a+r} \sum_{y=b-r}^{b+r} \big(|x| + |y|\big) \\ &= \sum_{x=a-r}^{a+r} \sum_{y=b-r}^{b+r} |x| + \sum_{y=b-r}^{b+r} \sum_{x=a-r}^{a+r} |y| \\ &= w \sum_{x=a-r}^{a+r} |x| + w \sum_{y=b-r}^{b+r} |y| \end{aligned}$$

(The factor T has not been explicitly included in the above equations nor in any of the following measures of cost.) Substituting in $(a, b) = (0, 0)$ we get the cost of transferring the first neighborhood:

$$\begin{aligned} C_{T,1} &= w \sum_{x=-r}^{r} |x| + w \sum_{y=-r}^{r} |y| \\ &= 2w \sum_{x=-r}^{r} |x| \end{aligned}$$

The following lemma allows us to further evaluate these sums and directly express the cost $C_{T,1}$ in terms of w and r and the cost $C_{T,2}$ in terms of w, r, a, and b.

Lemma E.1 *Let a and r be integers with $r \geq 0$. Then*

$$\sum_{x=a-r}^{a+r} |x| = \begin{cases} |a|(2r+1) & \text{if } r \leq |a| \\ a^2 + r^2 + r & \text{if } |a| < r \end{cases}$$

and, in both cases, the sum is greater than or equal to $r^2 + r$.

This lemma is proven at the end of this Appendix.

Now using Lemma E.1 we have:

$$C_{T,1} = 2w(r^2 + r) \approx \frac{1}{2} w^3 \tag{111}$$

The approximation $r^2 + r \approx \frac{1}{4}w^2$ is valid as long as $w \gg 1$, since $w^2 = (2r+1)^2 = 4r^2 + 4r + 1 = 4(r^2 + r) + 1$. The average cost per pixel in the $w \times w$ neighborhood is $\frac{1}{2}w^3/w^2 = \frac{1}{2}w$. This agrees with the fact that the average city-block distance of pixels from the origin over the $w \times w$ neighborhood is $\frac{1}{2}w$.

Evaluation of $C_{T,2}$ requires that we know the magnitudes of a and b relative to r. When a and b are relatively large, that is $|a|, |b| \geq r$, then we would have

$$C_{T,2} = \big(|a| + |b|\big)w^2 = dw^2$$

where $d = \| (a,b) \|_{CB}$. The average cost per pixel in the $w \times w$ neighborhood is $dw^2/w^2 = d$. This agrees with the fact that d is the average city-block distance of pixels from the origin over the $w \times w$ neighborhood centered at (a,b).

On the other hand if $|a|, |b| < r$, then

$$C_{T,2} = (a^2 + b^2)w + 2w(r^2 + r) \approx (a^2 + b^2)w + \frac{1}{2}w^3$$

The average cost per pixel is $(a^2 + b^2)/w + \frac{1}{2}w$.

Now $C_{T,2}$ is generally given by

$$C_{T,2} = \begin{cases} \big(|a| + |b|\big)w^2 & \text{if } r \leq |a| \text{ and } r \leq |b| \\ |a|w^2 + w(b^2 + r^2 + r) & \text{if } r \leq |a| \text{ and } |b| < r \\ w(a^2 + r^2 + r) + |b|w^2 & \text{if } |a| < r \text{ and } r \leq |b| \\ w(a^2 + b^2) + 2w(r^2 + r) & \text{if } |a| < r \text{ and } |b| < r \end{cases}$$

The above estimates of $C_{T,2}$ pertain to transfer costs for a single pixel. At a given level of the processing cone, the computation of disparity updates occurs in parallel at all pixels (x,y) at that level. This includes the access of pixel values from the neighborhood centered at $(x + a(x,y), y + b(x,y))$ in the second image. Transfer cost should be measured as the maximum transfer cost over all pixels. We thus define:

$$C_{T,2} \triangleq \max_{(x,y)} C_{T,2}$$

Also, let D_k be the maximum disparity magnitude at level k. That is

$$D_k \triangleq \max_{(x,y)} \|(a,b)\|_{Max} = \max_{(x,y)} [\max(|a|, |b|)]$$

The *Max* norm is used because the hierarchical search we use is constrained separately to a given maximum disparity in the two directions X and Y.

By definition of D_k, there is an (a,b) such that either $a = D_k$ or $b = D_k$. We will pick without loss of generality (a,b) with $|a| = D_k$. First consider the case in which $D_k < r$. Then for all pixels (x,y), $|a(x,y)| < r$ and $|b(x,y)| < r$, and we have

$$\begin{aligned}\overline{C}_{T,2} &= w\Big(\max_{(x,y)}(a^2+b^2)\Big) + 2w(r^2+r) \\ &\approx w\Big(\max_{(x,y)}(a^2+b^2)\Big) + \frac{1}{2}w^3 \\ &\geq wD_k^2 + \frac{1}{2}w^3\end{aligned}$$

Which finally gives us

$$\overline{C}_{T,2} \geq (\frac{D_k^2}{w} + \frac{w}{2})w^2 \qquad (\text{for } D_k < r) \tag{112}$$

and if we wish to eliminate the D_k term entirely

$$\overline{C}_{T,2} \geq \frac{w}{2}w^2 = \frac{1}{2}w^3 \qquad (\text{for } D_k < r) \tag{113}$$

Note that this is the same as our estimate of $C_{T,1}$ in Equation 111.

Next consider the case in which $D_k \geq r$. Again we pick w.l.o.g. (a,b) with $|a| = D_k$, then we have

$$\overline{C}_{T,2} \geq \begin{cases} D_k w^2 + |\bar{b}|w^2 & \text{if } r \leq |\bar{b}| \\ D_k w^2 + w(\bar{b}^2 + r^2 + r) & \text{if } |\bar{b}| < r \end{cases}$$

where

$$\bar{b} \triangleq \max_{\substack{(x,y) \\ |a(x,y)|=D_k}} |b(x,y)|$$

Ignoring $\bar{b}$, we know from Lemma E.1 that

$$\overline{C}_{T,2} \geq D_k w^2 + w(r^2+r)$$

Or, using the approximation $r^2 + r \approx \frac{1}{4}w^2$

$$\overline{C}_{T,2} \geq D_k w^2 + \frac{1}{4}w^3 = (D_k + \frac{w}{4})w^2 \qquad (\text{for } r \leq D_k) \tag{114}$$

Now if we wish to eliminate the D_k term, we use the fact that $D_k \geq r = (w-1)/2$ to get

$$\overline{C}_{T,2} \geq (\frac{3w}{4} - \frac{1}{2})w^2 \qquad (\text{for } r \leq D_k) \tag{115}$$

Finally, we combine the two costs $\overline{C}_{T,1}$ and $\overline{C}_{T,2}$ to get an overall cost $\overline{C}_T \triangleq \overline{C}_{T,1} + \overline{C}_{T,2}$. First we use Equations 111, 112, and 114 to get

$$C_T \geq (\frac{D_k^2}{w} + w)w^2 \qquad (\text{if } D_k < r) \tag{116}$$

and

$$\overline{C}_T \geq (D_k + \frac{3w}{4})w^2 \qquad (\text{if } D_k \geq r) \tag{117}$$

On the other hand if we wish to eliminate the D_k term from the bounds, then from Equations 111, 113, and 115, we get the weaker conditions:

$$\overline{C}_T \geq w^3 \qquad (\text{if } D_k < r) \tag{118}$$

and

$$\overline{C}_T \geq (\frac{5w}{4} - \frac{1}{2})w^2 \qquad (\text{if } D_k \geq r) \tag{119}$$

If we have no knowledge of the value of D_k, then the condition given in Equation 118 must be used. The value used as the bound in this case is simply twice the value of $C_{T,1}$.

Proof of Lemma E.1 : We will use the fact that for integers m and n such that $1 \leq m \leq n$, we have

$$\sum_{x=m}^{n} x = \sum_{x=1}^{n} x - \sum_{x=1}^{m-1} x = \frac{n(n+1)}{2} - \frac{(m-1)m}{2}$$

We consider three cases:

Case 1a : $r \leq a$

Then $x \geq a - r \geq 0$ for all terms in the summation.

$$\begin{aligned}\sum_{x=a-r}^{a+r} x &= \frac{(a+r)(a+r+1)}{2} - \frac{(a-r-1)(a-r)}{2} \\ &= \frac{1}{2}[(a+r)^2 + (a+r) - (a-r)^2 + (a-r)] \\ &= a(2r+1)\end{aligned}$$

Case 1b : $a \leq -r$

Then $x \leq a + r \leq 0$ for all terms in the summation.

$$\begin{aligned}\sum_{x=a-r}^{a+r} |x| = \sum_{\hat{x}=-a-r}^{-a+r} \hat{x} &= \frac{(-a+r)(-a+r+1)}{2} - \frac{(-a-r-1)(-a-r)}{2} \\ &= \frac{1}{2}[(-a+r)^2 + (-a+r) - (-a-r)^2 + (-a-r)] \\ &= -a(2r+1)\end{aligned}$$

The results of cases 1a and 1b can be combined to give the first condition of the desired result.

Case 2 : $|a| < r$, or equivalently, $-r < a < r$

Then $a - r < 0$ and $0 < a + r$ and the summation must be broken into two pieces.

$$\begin{aligned}\sum_{x=a-r}^{a+r} |x| &= \sum_{x=a-r}^{-1} |x| + \sum_{x=1}^{a+r} |x| \\ &= \sum_{\hat{x}=1}^{-a+r} \hat{x} + \sum_{x=1}^{a+r} x \qquad (\hat{x} = -x \geq 0) \\ &= \frac{(-a+r)(-a+r+1)}{2} + \frac{(a+r)(a+r+1)}{2} \\ &= \frac{1}{2}[(-a+r)^2 + (-a+r) + (a+r)^2 + (a+r)] \\ &= a^2 + r^2 + r\end{aligned}$$

This is the desired second condition.

Finally we show that the sum is greater than or equal to $r^2 + r$. This is obvious for the second condition. For the first condition, just note that $|a|(2r+1) \geq r(2r+1) = r^2 + (r^2 + r) \geq r^2 + r$. This completes the proof. □

APPENDIX F

Local Mode Analysis of Optic Flow Relaxation

In this appendix, we perform the local mode analysis [Brandt 77a] for optic flow relaxation. Local mode analysis is used to approximate the rate at which high frequency error terms are reduced in the approximate solution. This is an important number in multilevel relaxation since the purpose of relaxation at a given level is the smoothing of the high frequency error terms at that level. At intermediate levels, these frequencies are a band of the overall frequency space.

Since local mode analysis ignores lower frequencies, it can proceed without taking into account global aspects of the problem such as boundary conditions and the variation of coefficients. This allows a simple Fourier analysis of the error terms. In this analysis, the convergence rate for individual error frequencies (as defined by the Fourier Transform of the error function) is computed. The local convergence rate is then defined as the maximum individual convergence rate over the higher frequencies. These higher frequencies are selected to be those in the baseband at a given level which are not also in the baseband of the next coarser level. That is, the frequencies which are representable at the given level but not at a coarser level.

For Laplace's equation, the individual convergence rate and the local convergence rate are single numbers. For the optic flow equation, the two components of the optic flow field are coupled. We will show how convergence is in general defined by a two by two matrix. Eigenanalysis of this matrix allows us to define convergence

rates in terms of directions parallel to and perpendicular the (local) gradient of the image data.

The main content of this Appendix is the local mode analysis of optic flow relaxation. Prior to this, in the first section, we perform local mode analysis for the simpler case of first order smoothing based on Laplace's equation. This simpler case helps in understanding the more complicated optic flow analysis. The first analysis performed includes more of the detailed steps in its presentation, including the definition of *local convergence rate.*

1 Local Mode Analysis of Laplace's Equation

Brandt has shown how local mode analysis is performed for Laplace's equation when Lap_1 is the discrete Laplacian and Gauss-Seidel relaxation is used (see [Brandt 77a, pp.339-340] or [Brandt 80a, pp.122-123]). We show two similar analyses using Lap_1 and Lap_2 as the discrete Laplacians and Jacobi relaxation.

The two discrete Laplacians are defined as

$$Lap_1 \triangleq \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \qquad Lap_2 \triangleq \frac{1}{h^2}\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Local Mode Analysis F.1 *When Laplace's equation is solved using Lap_1 as the discrete Laplacian and the Jacobi relaxation scheme, then the local convergence rate is $\bar{\mu} = 1$.*

This is not a satisfactory result. It tells us that there is no reduction in the highest frequency component of the error. It is also clear that convergence is extremely slow for other frequencies near this highest one. Brandt's analysis of Gauss-Seidel relaxation using Lap_1 gives a local convergence rate of $\mu = .5$ [Brandt 77a]. If we wish to use a parallel updating scheme like Jacobi while insuring convergence for Laplace's equation, we must either select another relaxation scheme or another discrete Laplacian. For example a weighted Jacobi scheme can be used

to attain a local convergence of $\mu = .8$ [Brandt 80b]. On the other hand, using Lap_2 for the discrete Laplacian with Jacobi relaxation gives $\mu = 1/3$ as stated in the next result.

Local Mode Analysis F.2 *When Laplace's equation is solved using Lap_2 as the discrete Laplacian and the Jacobi relaxation scheme, then the local convergence rate is $\bar{\mu} = 1/3$.*

The local mode analysis of the first two results are now given. The first proof, for the Lap_1 case, provides a detailed example of local mode analysis.

Proof of Local Mode Analysis F.1 : On the discrete grid (m, n), let $U_{m,n}$ be the solution to a discrete representation of Laplace's equation. Using Lap_1, Laplace's equation is given by

$$4U_{m,n} - \left(U_{m-1,n} + U_{m,n-1} + U_{m+1,n} + U_{m,n+1}\right) = 0 \tag{120}$$

Now let $u_{m,n}$ be an approximate solution before the relaxation and let $\bar{u}_{m,n}$ be the approximation after. Then the update equation is

$$\bar{u}_{m,n} \leftarrow \frac{1}{4}\left(u_{m-1,n} + u_{m,n-1} + u_{m+1,n} + u_{m,n+1}\right) \tag{121}$$

And this can be rearranged to

$$4\bar{u}_{m,n} - \left(u_{m-1,n} + u_{m,n-1} + u_{m+1,n} + u_{m,n+1}\right) = 0 \tag{122}$$

Now let the errors before and after relaxation be given by $\delta_{m,n} \triangleq U_{m,n} - u_{m,n}$ and $\bar{\delta}_{m,n} \triangleq U_{m,n} - \bar{u}_{m,n}$. Subtracting Equation 122 from Equation 120 we get

$$4\bar{\delta}_{m,n} - \left(\delta_{m-1,n} + \delta_{m,n-1} + \delta_{m+1,n} + \delta_{m,n+1}\right) = 0 \tag{123}$$

Expand the error functions into Fourier integrals

$$\delta_{m,n} = \int_{|\theta| \le \pi} A_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \tag{124}$$

$$\bar{\delta}_{m,n} = \int_{|\theta| \le \pi} \overline{A}_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \tag{125}$$

where $\theta = (\theta_1, \theta_2)$ and $|\theta| = \max(\theta_1, \theta_2)$. (The representation of spatial frequency as (θ_1, θ_2) corresponds to the alternative representation as $2\pi(s,t)$ in Appendix A. The region define by $|\theta| \leq \pi$ in (θ_1, θ_2) space corresponds to the baseband $[-\frac{1}{2}, \frac{1}{2}]^2$ in the (s,t) space.)

Substitute these Fourier integrals into Equation 123 and combine the integrals:

$$0 = \int_{|\theta| \leq \pi} e^{i(\theta_1 m + \theta_2 n)} \left[4\overline{A}_\theta - A_\theta (e^{-i\theta_1} + e^{i\theta_1} + e^{-i\theta_2} + e^{i\theta_2}) \right] d\theta \tag{126}$$

Since this must hold for all (m,n), the integrand must be 0 and we have:

$$4\overline{A}_\theta - 2A_\theta(\cos\theta_1 + \cos\theta_2) = 0 \tag{127}$$

The convergence rate for the θ component is then defined as

$$\mu(\theta) \triangleq \left| \frac{\overline{A}_\theta}{A_\theta} \right| = \frac{1}{2} |\cos\theta_1 + \cos\theta_2| \tag{128}$$

The **local convergence rate** is defined as the maximum individual convergence rate over the high frequencies, that is:

$$\bar{\mu} \triangleq \max_{\pi/2 \leq |\theta| \leq \pi} \mu(\theta) = \mu(\pi, \pi) = 1 \tag{129}$$

□

Proof of Local Mode Analysis F.2 : Using the same line of analysis as the above proof, this time with Lap_2 used as the discrete Laplacian, we arrive at the following relationship between u and $\bar{u}$ (analogous to Equation 122):

$$12\bar{u} - \begin{pmatrix} u_{m-1,n-1} & + & 2u_{m-1,n} & + & u_{m-1,n+1} \\ + \; 2u_{m,n-1} & + & & + & 2u_{m,n+1} \\ + \; u_{m+1,n-1} & + & 2u_{m+1,n} & + & u_{m+1,n+1} \end{pmatrix} = 0 \tag{130}$$

(The summation in shown in an array-like notation to emphasize its relationship to the local averaging used in updating.) A similar relationship holds between the

errors δ and $\bar{\delta}$. After performing the Fourier analysis, we arrive at the following relationship between A_θ and $\overline{A}_\theta$:

$$12\overline{A}_\theta - \begin{pmatrix} e^{-i(\theta_1+\theta_2)} & + & 2e^{-i\theta_1} & + & e^{-\theta_1+\theta_2} \\ + & 2e^{-i\theta_2} & + & & + \quad 2e^{i\theta_2} \\ + & e^{i(\theta_1+\theta_2)} & + & 2e^{i\theta_1} & + \quad e^{\theta_1+\theta_2} \end{pmatrix} A_\theta = 0 \tag{131}$$

The convergence at frequency θ is

$$\mu(\theta) \triangleq \left|\frac{A_\theta}{A_\theta}\right| = \frac{1}{12}|4\cos\theta_1 + 4\cos\theta_2 + 2\cos(\theta_1+\theta_2) + 2\cos(\theta_2-\theta_1)| \tag{132}$$

$$= \frac{1}{3}|\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2| \tag{133}$$

and straightforward calculations show that the local convergence rate is

$$\bar{\mu} \triangleq \max_{\pi/2 \le |\theta| \le \pi} \mu(\theta) = \frac{1}{3} \tag{134}$$

□

2 Local Mode Analysis for Optic Flow Relaxation

In the local mode analysis of Laplace's equation, the error components at frequency θ before and after relaxation were represented by the Fourier coefficients A_θ and $\overline{A}_\theta$ respectively. In the case of optic flow, the error component is a vector field, and there are two Fourier coefficients both before and after relaxation, one for each of the vector components. The general relationship between these coefficients is given by a 2×2 matrix. Convergence can be expressed in terms of the eigenvalues and eigenvectors of this matrix.

Local Mode Analysis F.3 *Suppose we use the discrete Laplacian Lap_2 in a Jacobi relaxation scheme to do multilevel optic flow relaxation. If (A_θ, B_θ) are the Fourier coefficients of the θ component of the error before relaxation, and (A_θ, B_θ)*

are the coefficients after relaxation, then these coefficients are related by the **convergence matrix** M *as follows:*

$$\begin{bmatrix} \overline{A}_\theta \\ B_\theta \end{bmatrix} = M \begin{bmatrix} A_\theta \\ B_\theta \end{bmatrix}$$

$$= \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \begin{bmatrix} 3\alpha^2 + F_y^2 & -F_xF_y \\ -F_xF_y & 3\alpha^2 + F_x^2 \end{bmatrix} \begin{bmatrix} A_\theta \\ B_\theta \end{bmatrix}$$

The eigenvalues of M *are*

$$\lambda_{min} = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \quad 3\alpha^2 \quad 3\alpha^2 + F_x^2 + F_y^2 \tag{135}$$

$$\lambda_{max} = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \tag{136}$$

The eigenvector associated with λ_{min} *is* (F_x, F_y)*, the gradient of* F*. The eigenvector associated with* λ_{max} *is* $(-F_y, F_x)$*, a vector perpendicular to the gradient of* F*.*

The convergence matrix generalizes the convergence factor $\mu(\theta) \triangleq |A_\theta/A_\theta|$ defined in the first section. (In a non-rigorous sense it is a measure of $[\overline{A}_\theta\ \overline{B}_\theta]^T/[A_\theta\ B_\theta]^T$.) The relationship between the magnitudes of $[\overline{A}_\theta\ \overline{B}_\theta]^T$ and $[A_\theta\ B_\theta]^T$ can be understood by considering the eigenvalues and eigenvectors of M.

If the gradient vanishes, then $F_x^2 + F_y^2 = 0$ and $\lambda_{max} = \lambda_{min}$. In this case convergence in the two components of the optic flow is identical to that seen for simple smoothing by Laplace's equation. This is consistent with the fact that the optic flow update equation reduces to simple smoothing when the gradient vanishes.

If the gradient is significant, then we see that error components perpendicular to the gradient still converge at the simple smoothing rate. On the other hand error components parallel to the gradient converge at a faster rate, where the increase is proportional to the magnitude of the gradient.

The term $\frac{1}{3}(\cos\theta_1+\cos\theta_2+\cos\theta_1\cos\theta_2)$ that appears in the convergence matrix is identical to that which appears in Equation 133. It is reasonable to suspect that the existence of this term is dependent only on the particular discrete Laplacian applied to the optic flow relaxation update. This is in fact the case. If Lap_1 is used

in the local mode analysis of optic flow relaxation, then the θ-dependent term in the convergence matrix is $\frac{1}{2}(\cos\theta_1 + \cos\theta_2)$ as in Equation 128.

Proof of Local Mode Analysis F.3 : Let (U, V) be the exact solution to the discrete form of the optic flow equation at some level of the cone. In Chapter VII (Equation 79 on page 210), the equations for (U, V) are shown to be

$$\alpha^2 \Delta U - F_x^2 U - F_x F_y V = F_x F_t \quad . \tag{137a}$$

$$\alpha^2 \Delta V - F_x F_y U - F_y^2 V = F_y F_t \tag{137b}$$

We have assumed that ΔU is approximated by the discrete Laplacian $Lap_2 U$. $Lap_2 U$ is equal to $3(\hat{U} - U)$, where $\hat{U}$ is the local average defined just below. Substituting in to the above equations and solving for the central value U gives:

$$U_{m,n} = \hat{U} - [a\hat{U} + b\hat{V} + d]/(3\alpha^2 + a^2 + c^2) \tag{138a}$$

$$V_{m,n} = \hat{V} - [b\hat{U} + c\hat{V} + e]/(3\alpha^2 + a^2 + c^2) \tag{138b}$$

where $a = F_x^2$, $b = F_x F_y$, $c = F_y^2$, $d = F_x F_t$, and $e = F_y F_t$; and where the local averages $\hat{U}$ and $\hat{V}$ are given by:

$$\hat{U} = \hat{U}_{m,n} = \frac{1}{12}\begin{pmatrix} U_{m-1,n-1} + 2U_{m-1,n} + U_{m-1,n+1} \\ + 2U_{m,n-1} + 2U_{m,n+1} \\ + U_{m+1,n-1} + 2U_{m+1,n} + U_{m+1,n+1} \end{pmatrix} \tag{139a}$$

$$\hat{V} = \hat{V}_{m,n} = \frac{1}{12}\begin{pmatrix} V_{m-1,n-1} + 2V_{m-1,n} + V_{m-1,n+1} \\ + 2V_{m,n-1} + 2V_{m,n+1} \\ + V_{m+1,n-1} + 2V_{m+1,n} + V_{m+1,n+1} \end{pmatrix} \tag{139b}$$

Now let (u, v) be an approximate solution before the relaxation and let $(\bar{u}, \bar{v})$ be the approximation after. The update equation using Jacobi relaxation and the Lap_2 discrete operator is given by simple substitution into Equation 138:

$$\bar{u}_{m,n} = \hat{u} - [a\hat{u} + b\hat{v} + d]/(3\alpha^2 + a^2 + c^2) \tag{140a}$$

$$\bar{v}_{m,n} = \hat{v} - [b\hat{u} + c\hat{v} + e]/(3\alpha^2 + a^2 + c^2) \tag{140b}$$

where $a = F_x^2$, $b = F_xF_y$, $c = F_y^2$, $d = F_xF_t$, and $e = F_yF_t$; and where the local averages $\hat{u}$ and $\hat{v}$ are given by:

$$\hat{u} = \hat{u}_{m,n} = \frac{1}{12}\begin{pmatrix} & u_{m-1,n-1} & + & 2u_{m-1,n} & + & u_{m-1,n+1} \\ + & 2u_{m,n-1} & + & & + & 2u_{m,n+1} \\ + & u_{m+1,n-1} & + & 2u_{m+1,n} & + & u_{m+1,n+1} \end{pmatrix} \tag{141a}$$

$$\hat{v} = \hat{v}_{m,n} = \frac{1}{12}\begin{pmatrix} & v_{m-1,n-1} & + & 2v_{m-1,n} & + & v_{m-1,n+1} \\ + & 2v_{m,n-1} & + & & + & 2v_{m,n+1} \\ + & v_{m+1,n-1} & + & 2v_{m+1,n} & + & v_{m+1,n+1} \end{pmatrix} \tag{141b}$$

Let the errors before and after relaxation be given by $(\delta, \varepsilon) \triangleq (U, V) - (u, v)$ and $(\bar{\delta}, \bar{\varepsilon}) \triangleq (U, V) - (\bar{u}, \bar{v})$. Then by simple rearrangement of Equations 138 and 140, followed by subtraction of the two sets of equations we get:

$$(3\alpha^2 + a^2 + c^2)(\hat{\delta} - \bar{\delta}_{m,n}) = [a\hat{\delta} + b\hat{\varepsilon} + d] \tag{142a}$$

$$(3\alpha^2 + a^2 + c^2)(\hat{\varepsilon} - \bar{\varepsilon}_{m,n}) = [b\hat{\delta} + c\hat{\varepsilon} + e] \tag{142b}$$

where

$$\hat{\delta} = \hat{\delta}_{m,n} = \frac{1}{12}\begin{pmatrix} & \delta_{m-1,n-1} & + & 2\delta_{m-1,n} & + & \delta_{m-1,n+1} \\ + & 2\delta_{m,n-1} & + & & + & 2\delta_{m,n+1} \\ + & \delta_{m+1,n-1} & + & 2\delta_{m+1,n} & + & \delta_{m+1,n+1} \end{pmatrix} \tag{143a}$$

$$\hat{\varepsilon} = \hat{\varepsilon}_{m,n} = \frac{1}{12}\begin{pmatrix} & \varepsilon_{m-1,n-1} & + & 2\varepsilon_{m-1,n} & + & \varepsilon_{m-1,n+1} \\ + & 2\varepsilon_{m,n-1} & + & & + & 2\varepsilon_{m,n+1} \\ + & \varepsilon_{m+1,n-1} & + & 2\varepsilon_{m+1,n} & + & \varepsilon_{m+1,n+1} \end{pmatrix} \tag{143b}$$

Expand the error functions into Fourier integrals

$$\delta_{m,n} = \int_{|\theta| \le \pi} A_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \qquad \varepsilon_{m,n} = \int_{|\theta| \le \pi} B_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \tag{144}$$

$$\bar{\delta}_{m,n} = \int_{|\theta| \le \pi} \overline{A}_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \qquad \bar{\varepsilon}_{m,n} = \int_{|\theta| \le \pi} \overline{B}_\theta e^{i(\theta_1 m + \theta_2 n)}\, d\theta \tag{145}$$

Substituting into Equation 142, combining the integrals, and requiring the integrand to be 0, then gives:

$$(3\alpha^2 + a^2 + c^2)(EA_\theta - 12\overline{A}_\theta) = aEA_\theta + bEB_\theta \tag{146a}$$

$$(3\alpha^2 + a^2 + c^2)(EB_\theta - 12B_\theta) = bEA_\theta + cEB_\theta \tag{146b}$$

where

$$E = \begin{pmatrix} e^{-i(\theta_1+\theta_2)} + 2e^{-i\theta_1} + e^{-\theta_1+\theta_2} \\ + 2e^{-i\theta_2} + + 2e^{i\theta_2} \\ + e^{i(\theta_1+\theta_2)} + 2e^{i\theta_1} + e^{\theta_1+\theta_2} \end{pmatrix}$$

$$= 4\cos\theta_1 + 4\cos\theta_2 + 2\cos(\theta_1+\theta_2) + 2\cos(\theta_2-\theta_1)$$

$$= 4(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \quad (147)$$

Note that this same E term appeared in the local mode analysis of Laplace's equation with Lap_2 in Equation 131. Equation 146 can be rearranged into the following matrix equation

$$\begin{bmatrix} \overline{A}_\theta \\ B_\theta \end{bmatrix} = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \begin{bmatrix} 3\alpha^2+c & -b \\ -b & 3\alpha^2+a \end{bmatrix} \begin{bmatrix} A_\theta \\ B_\theta \end{bmatrix} \quad (148)$$

This is the relationship which defines the convergence matrix M.

The eigenanalysis of M is straightforward. Let

$$A = \begin{bmatrix} 3\alpha^2+c & -b \\ -b & 3\alpha^2+a \end{bmatrix} \quad (149)$$

The characteristic polynomial of A is the determinant of the matrix $A - \lambda I$ (I is the identity matrix).

$$\det(A-\lambda I) = (3\alpha^2+c-\lambda)(3\alpha^2+a-\lambda) - b^2 \quad (150)$$

$$= \lambda^2 - (6\alpha^2+a+c)\lambda + 3\alpha^2(3\alpha^2+a+c) + (ac-b^2) \quad (151)$$

The term $ac - b^2$ drops out, being equal to zero (since $a = F_x^2$, $b = F_xF_y$, and $c = F_y^2$). By inspection we soon see that the roots of the characteristic polynomial, i.e. the eigenvalues of A, are $\lambda_{min} = 3\alpha^2$ and $\lambda_{max} = 3\alpha^2 + a + c$.

Hence the eigenvalues of M are

$$\lambda_{min} = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2)\frac{3\alpha^2}{3\alpha^2+a+c} \quad (152)$$

$$\lambda_{max} = \frac{1}{3}(\cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2) \quad (153)$$

The eigenvectors of M are equivalent to those of A. They are the non-trivial solutions to the matrix equation $A\mathbf{x} = \lambda\mathbf{x}$. The eigenvector associated with λ_{min} is found to be (a, b) or equivalently (F_x, F_y). The eigenvector associated with λ_{max} is found to be $(-c, a)$ or equivalently $(-F_y, F_x)$. □

References

[Adiv 85] Adiv, G., Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects, *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(2):384–401.

[Aggarwal *et al.* 81] Aggarwal, J.K., Davis, L.S., and Martin, W.N., Correspondence Processes in Dynamic Scene Analysis, *Proc. IEEE* **69**(5):562–672, 1981.

[Anandan 84] Anandan, P., Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion, *SPIE Vol. 521 Intelligent Robots and Computer Vision*, Cambridge, Mass., 1984. Also in expanded form as COINS Tech. Report 84-32, U.Massachusetts, Amherst, 1984.

[Anandan 87] Anandan, P., Measuring Visual Motion from Image Sequences, Ph.D. Thesis and COINS Tech. Report, U.Massachusetts, Amherst, 1987

[Antonisse 82] Antonisse, H.J., Image Segmentation in Pyramids, *Computer Graphics and Image Processing* **19**:367-383, 1982.

[Barnard & Thompson 80] Barnard, S.T., and Thompson, W.B., Disparity Analysis of Images, *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**(4):333–340, 1980.

[Barnea & Silverman 72] Barnea, D.I., and Silverman, H.F., A Class of Algorithms for Fast Digital Image Registration, *IEEE Trans. Computers* **21**(2):179–186, 1972.

[Beaudet 78] Beaudet, P.R., Rotationally Invariant Image Operators, *Proc. Int. Conf. on Pattern Recognition* pp. 579-583, 1978.

[Birkhoff & Lynch 84] Birkhoff, G., and Lynch, R.E., *Numerical Solution of Elliptic Problems*, SIAM, Philadelphia, 1984.

[Bracewell 78] Bracewell, R.N, *The Fourier Transform and Its Applications*, McGraw-Hill, New York, 1978.

[Brandt 77a] Brandt, A., Multi-Level Adaptive Solutions to Boundary-Value Problems, *Mathematics of Computation* **31**(138):333–390, 1977.

[Brandt 77b] Brandt, A., Multi-Level Adaptive Techniques (MLAT) for Partial Differential Equations: Ideas and Software, in *Mathematical Software III*, J.R. Rice (Ed.), Academic Press, 1977.

[Brandt 80a] Brandt, A., Multi-Level Adaptive Finite-Element Methods: I. Variational Problems, In *Special Topics of Applied Mathematics*, Freshe, J., Pallaschke, D., and Trottenberg, U. (Eds.), North Holland, New York, 1980.

[Brandt 80b] Brandt, A., Multigrid Solvers on Parallel Computers, ICASE Report No. 80-23, NASA Langley Research Center, Hampton, Virginia, 1980. Also in *Elliptic Problem Solvers*, M.H. Schultz (Ed.), Academic Press, New York, 1981.

[Burt 80] Burt, P.J., Tree and Pyramid Structures for Coding Hexagonally Sampled Binary Images, *Computer Graphics and Image Processing* **14**(3):271-280, 1980.

[Burt 81] Burt, P.J., Fast Filter Transforms for Image Processing, *Computer Graphics and Image Processing* **16**(1):20–51, 1981.

[Burt *et al.* 81] Burt, P.J., Hong, T.H., and Rosenfeld, A., Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation, *IEEE Trans. Systems, Man, and Cybernetics* **11**(12):802-809, 1981.

[Burt 82a] Burt, P.J., Pyramid-Based Extraction of Local Image Features with Applications to Motion and Texture Analysis, *Proc. SPIE Conf. on Robotics and Industrial Inspection*, San Diego, 1982.

[Burt 82b] Burt, P.J., Fast Algorithms for Estimating Local Image Properties, Tech. Rep. IPL-TR-022, Image Processing Lab., Electrical, Comp., and Systems Eng. Dept., Rensselaer Polytechnic Institute, 1982.

[Burt & Adelson 82] Burt, P.J., and Adelson, E.H., The Laplacian Pyramid as a Compact Image Code, Tech. Rep. IPL-TR-025, Image Processing Lab., Electrical, Comp., and Systems Eng. Dept., Rensselaer Polytechnic Institute, 1982. Also in *IEEE Trans. on Communications*, **31**(4):532-540, 1983.

[Burt *et al.* 82] Burt, P.J., Yen, C., and Xu, X., Local Correlation Measures for Motion Analysis: A Comparative Study, *Proc. Pattern Recognition and Image Processing* pp. 269–274, 1982. Also IPL-TR-024, ECSE Dept., Rensselaer Polytechnic Institute, 1982.

[Burt *et al.* 83] Burt, P.J., Yen, C., and Xu, X., Multi-Resolution Flow-Through Motion Analysis, *Proc. Computer Vision and Pattern Recognition* pp. 246-252, 1983.

[Burt 83] Burt, P.J., Some Useful Properties of Pyramids, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Chen & Pavlidis 79] Chen, P.C., and Pavlidis, T., Segmentation by Texture Using a Co-occurence Matrix and a Split-and-Merge Algorithm, *Computer Graphics and Image Processing* **10**:172-182, 1979.

[Cibulskis & Dyer 84] Cibulskis, J.M., and Dyer, C.R., An Analysis of Node Linking in Overlapped Pyramids, *IEEE Trans. Systems, Man, and Cybernetics* **14**(3):424-436, 1984.

[Cornelius & Kanade 83] Cornelius, N. and Kanade, T., Adapting Optical-Flow to Measure Object Motion in Reflecting and X-ray Image Sequences, *Proc. ACM SIGGRAPH/SIGGART Interdisciplinary Workshop on Motion: Representation and Perception*, pp. 50-58, 1983.

[Courant & Hilbert 53] Courant, R. and Hilbert, D., *Methods of Mathematical Physics, Volume I*, Interscience Publishers, Inc., New York, 1953.

[Crowley 81] Crowley, J.L., A Representation for Visual Information, Ph.D. dissertation, Carnegie-Mellon Univ., 1984.

[Crowley & Parker 84] Crowley, J.L. and Parker, A.C., A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform, *IEEE Trans. Pattern Analysis and Machine Intelligence* **6**(2):156-169, 1984.

[Crowley & Stern 84] Crowley, J.L. and Stern, R.M., Fast Computation for the Difference of Low-Pass Transform, *IEEE Trans. Pattern Analysis and Machine Intelligence* **6**(2):212-221, 1984.

[Dudgeon & Mersereau 84] Dudgeon, D.E. and Mersereau, R.M., *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[Dyer & Rosenfeld 81] Dyer, C.R., and Rosenfeld, A., Parallel Image Processing by Memory-Augmented Cellular Automata, *IEEE Trans. Pattern Analysis and Machine Intelligence* **3**(1):29-41, 1981.

[Dyer 81] Dyer, C.R., A VLSI Pyramid Machine for Hierarchical Parallel Image Processing, *Proc. Pattern Recognition and Image Processing* pp. 381-382, 1981.

[Davis & Roussopoulos 80] Davis, L.S., and Roussopoulos, N., Approximate Pattern Matching in a Pattern Database System, Information Systems 5(2):107-119, 1980.

[Enkelmann 86] Enkelmann,W., Investigations of Multigrid Algorithms for the Estimation of Optical Flow Fields in Image Sequences, *Proc. Motion Workshop* pp. 81-87, Kiawah Island, S.C., May 1986.

[Fennema & Thompson 79] Fennema, C.L. and Thompson, W.B., Velocity Determination in Scenes Containing Several Moving Objects, *Computer Graphics and Image Processing* 9(4):301-315, 1979.

[Fountain 83] Fountain, T.J., A Survey of Bit-Serial Array Processor Circuits, In: *Computing Structures for Image Processing*, M.J.B. Duff (Ed.), Academic Press, 1983.

[Glazer 81] Glazer, F., Computing Optic Flow, *Proc. 7th. Int. Joint Conf. on Artificial Intelligence*, pp. 644-647, Vancouver, BC, 1981.

[Glazer 82] Glazer, F., Multilevel Relaxation in Low Level Computer Vision, COINS Tech. Report 82-30, U.Massachusetts, Amherst, 1982. Also in: *Multiresolution Image. Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Glazer *et al.* 83] Glazer, F., Reynolds, G., and P. Anandan, Scene Matching by Hierarchical Correlation, *Proc. Computer Vision and Pattern Recognition* pp. 432-441, 1983.

[Gill *et al.* 81] Gill, P.E., Murray, W., and Wright, M.H., *Practical Optimization*, Academic Press, New York, 1981.

[Grimson 81a] Grimson, W.E.L., A Computer Implementation of a Theory of Human Stereo Vision, *Trans. R. Soc. Lond. B*, **292**:217-253, 1981.

[Grimson 81b] Grimson, W.E.L., A Computational Theory of Visual Surface Interpolation, A.I. Memo 613, MIT AI Lab, Cambridge, MA, June, 1981.

[Grossky & Jain 83] Grossky, W.I., and Jain, R., Region Matching in Pyramids for Dynamic Scene Analysis, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Hackbusch 85] Hackbusch, W., *Multi-Grid Methods and Applications*, Springer-Verlag, New York, 1985.

[Hageman & Young 81] Hageman, L.A. and Young, D.M., *Applied Iterative Methods*, Academic Press, New York, 1981.

[Hanson & Riseman 74] Hanson, A. and Riseman, E.M., Preprocessing Cones: A Computational Structure for Scene Analysis, COINS Tech. Report 74C-7, U.Massachusetts, Amherst, 1974.

[Hanson & Riseman 80] 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.

[Haralick & Lee 83] Haralick, R.M., and Lee, J.S., The Facet Approach to Optic Flow, *Proc. Image Understanding Workshop (DARPA)*, pp. 84–93, June, 1983.

[Hartman & Tanimoto 84] Hartman, N.P., and Tanimoto, S.L., A Hexagonal Pyramid Data Structure for Image Processing, *IEEE Trans. Systems, Man, and Cybernetics* **14**(2):247-256, 1984.

[Haynes & Jain 83] Haynes, S.M. and Jain, R., Detection of Moving Edges, *Computer Vision, Graphics and Image Processing* **21**(3):345-367, 1983.

[Hill *et al.* 83] Hill, F.S., Walker, S., and Gao, G., Interactive Query System Using Progressive Transmission, Computer Graphics **17**(3):323-330, 1983.

[Hong *et al.* 82] Hong, T.H., Narayanan, K.A., Peleg, S., Rosenfeld, A., Silberberg, T., Image Smoothing and Segmentation by Multiresolution Pixel Linking: Further Experiments and Extensions, *IEEE Trans. Systems, Man, and Cybernetics* **12**(5):611-622, 1982.

[Horn & Schunck 81] Horn, B.K.P. and Schunck, B.G., 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.

[Horowitz & Pavlidis 76] Horowitz, S.L., and Pavlidis, T., Picture Segmentation by Tree Traversal Algorithms, *Journal of the ACM* **23**:368-388, 1976

[Ikeuchi 80] Ikeuchi, K., Numerical Shape from Shading and Occluding Contour in a Single View, A.I. Memo 566, MIT AI Lab, Cambridge, MA, Feb., 1980.

[Ikeuchi & Horn 81] Ikeuchi, K. and Horn, B.K.P., Numerical Shape from Shading and Occluding Boundaries, *Artificial Intelligence* **17**(1-3):141-184, 1981.

[Kahn 85] Kahn, P., Local Determination of a Moving Contrast Edge, *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(2):402-409.

[Kelly 71] Kelly, M.D., Edge Detection in Pictures by Computer Using Planning, In: *Machine Intelligence Vol. 6*, pp. 397-409, Meltzer, B. and Mitchie, D. (eds.), Edinburgh University Press, American Elsevier, New York, 1971.

[Kent *et al.* 84] Kent, E.W., Shneier, M.O., and Lumia, R.L., PIPE: Pipelined Image Processing Engine, National Bureau of Standards, Tech Report, 1984.

[Kitchen & Rosenfeld 82] Kitchen, L., and Rosenfeld, A., Gray-Level Corner Detection, *Pattern Recognition Letters* **1**(2):95-102, 1982.

[Klinger & Dyer 76] Klinger, A. and Dyer, R.D., Experiments on Picture Representation Using Regular Decomposition, *Computer Graphics and Image Processing* **5**(1):68-105, 1976.

[Kohler & Hanson 82] Kohler, R., and Hanson, A., The VISIONS Image Operating System, *Proc. Int. Conf. on Pattern Recognition* pp. 71-74, Munich, 1982.

[Lawton 83] Lawton, D., Processing Translational Motion Sequences, *Computer Vision, Graphics and Image Processing* **22**(1):116-144, 1983.

[Lawton 84] Lawton, D., Processing Dynamic Image Sequences From A Moving Sensor, Ph.D. Thesis and COINS Tech. Report 84-05, U.Massachusetts, Amherst, 1984.

[Lucas & Gibson 84] Lucas and Gibson, Image Pyramids and Partitions, *Proc. Int. Conf. on Pattern Recognition* pp. 230-233, 1984.

[Lucas & Kanade 81] Lucas, B.D., and Kanade, T., An Iterative Image Registration Technique with an Application to Stereo Vision, *Proc. 7th. Int. Joint Conf. on Artificial Intelligence*, pp. 674-679, Vancouver, B.C., Canada, 1981

[Luenberger 73] Luenberger, D.G., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[McCormick & Trottenberg 83] McCormick, S. and Trottenberg, U. (ed.), Multigrid Methods, special issue of *Applied Mathematics and Computation*, **13**(3 & 4), November, 1983.

[Marr & Poggio 79] Marr, D. and Poggio, T., A Computational Theory of Human Stereo Vision, *Proc. R. Soc. Lond. B*, **204**:301–328, 1979.

[Marr & Ullman 79] Marr, D., and Ullman, S., Directional Selectivity and its Use in Early Visual Processing, A.I. Memo 524, MIT AI Lab, Cambridge, MA, June, 1979.

[Marr & Hildreth 80] Marr, D. and Hildreth, E., Theory of Edge Detection, *Proc. R. Soc. Lond. B*, **207**:187–217, 1980.

[Milman & Parker 77] Millman, R.S., and Parker, G.D., *Elements of Differential Geometry*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

[Moravec 81] Moravec, H.P., *Robot Rover Visual Navigation*, UMI Research Press, Ann Arbor, Michigan, 1981.

[Mutch & Thompson 83] Mutch, K.M., and Thompson, W.B., Hierarchical Estimation of Spatial Properties from Motion, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Nagel 83a] Nagel, H.-H., Displacement Vectors Derived from Second Order Intensity Variations in Image Sequences, *Computer Vision, Graphics and Image Processing* **21**:85–117, 1983.

[Nagel 83b] Nagel, H.-H., Constraints for the Estimation of Displacement Vector Fields From Image Sequences, *Proc. 8th. Int. Joint Conf. on Artificial Intelligence*, pp. 945-951, Karlsruhe, West Germany, 1983.

[Nagel & Enkelmann 84] Nagel, H.-H., Enkelmann,W., Towards the Estimation of Displacement Vector Fields by "Oriented Smoothness" Constraints, *Proc. Int. Conf. on Pattern Recognition* pp. 6-8, Montreal, 1984.

[Nagin *et al.* 82] Nagin, P.A., Hanson, A.R., and Riseman, E.M., Studies in Global and Local Histogram-Guided Relaxation Algorithms, *IEEE Trans. Pattern Analysis and Machine Intelligence* **4**(3):263-277, 1982.

[Narayanan *et al.* 82] Narayanan, K.A., D.P. O'Leary and A. Rosenfeld, Image Smoothing and Segmentation by Cost Minimization, *IEEE Trans. Systems, Man, and Cybernetics* **12**(1):91–96, 1982.

[Neveau *et al.* 85] Neveau, C.F., Dyer, C.R., and Chin, R.T., Object Recognition Using Hough Pyramids, *Proc. Computer Vision and Pattern Recognition* pp. 328-333, 1985.

[Oppenheim & Schafer 75] Oppenheim, A.V., and Schafer, R.W., *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[Pietikainen & Rosenfeld 81] Pietikäinen, M., and Rosenfeld, A., Image Segmentation by Texture Using Pyramid Node Linking, *IEEE Trans. Systems, Man, and Cybernetics* **11**(12):822-825, 1981.

[Pietikainen *et al.* 82] Pietikäinen, M., Rosenfeld, A., and Walter, I., Split-and-Link Algorithms for Image Segmentation, *Pattern Recognition* **15**(4):287-298, 1982.

[Prager 79] Prager, J.M., Segmentation of Static and Dynamic Scenes, Ph.D. Thesis and COINS Tech. Report 79-7, U.Massachusetts, Amherst, 1979.

[Prager & Arbib 83] Prager, J., and Arbib, M., Computing the Optic Flow: The MATCH Algorithm and Prediction, *Computer Vision, Graphics and Image Processing* **24**:271-304, 1983.

[Preston & Uhr 82] Preston, K., and Uhr, L., *Multicomputers and Image Processing*, Academic Press, New York, 1982.

[Quam 84] Quam, L.H., Hierarchical Warp Stereo, *Proc. Image Understanding Workshop (DARPA)*, pp. 149-155, October 1984.

[Rice 83] Rice, J.R., *Numerical Methods, Software, and Analysis*, McGraw-Hill, NewYork, 1983.

[Rosenfeld 83] Rosenfeld, A. (Ed.), *Multiresolution Image Processing and Analysis*, Springer-Verlag, 1983.

[Rosenfeld & Kak 82] Rosenfeld, A., and Kak, A., *Digital Picture Processing*, Academic Press, New York, 1982.

[Rosenfeld & Thurston 71] Rosenfeld, A., and Thurston, M., Edge and Curve Detection for Visual Scene Analysis, *IEEE Trans. Computers* **20**(5):562-569, 1971.

[Rosenfeld & Vanderbrug 77] Rosenfeld, A., and Vanderbrug, G.J., Coarse-Fine Template Matching, *IEEE Trans. Systems, Man, and Cybernetics* **7**(2):104-107, 1977.

[Samet 84] Samet, H., The Quadtree and Related Hierarchical Data Structures, *Computing Surveys* **16**(2):187–260, 1984.

[Schunck 83] Schunck, B.G., Motion Segmentation and Estimation, Ph.D. Thesis, MIT, Dept. of EE & CS, 1983.

[Sloan & Tanimoto 79] Sloan, K.R., and Tanimoto, S.L., Progressive Refinement of Raster Images, *IEEE Trans. Computers* **28**(11):871-874, 1979.

[Smith 78] Smith, G.D., *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford University Press, Oxford, 1978.

[Stout 83] Stout, Q., Sorting, Merging, Selecting, and Filtering on Tree and Pyramid Machines, *Proc. Int. Conf. on Parallel Processing* pp. 214-221, 1983.

[Snyder *et al.* 80] Snyder, W.E., Rajala, S.A., and Hirzinger, G., Image Modeling: The Continuity Assumption and Tracking, *Proc. Int. Conf. on Pattern Recognition* pp. 1111-1114, Miami Beach, 1980.

[Tanimoto & Pavlidis 75] Tanimoto, S., and Pavlidis, T., A Hierarchical Data Structure for Picture Processing, *Computer Graphics and Image Processing* **4**(2):104–119, 1975.

[Tanimoto 78] 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.

[Tanimoto & Klinger 80] Tanimoto, S., and Klinger, A. (Editors), *Structured Computer Vision: Machine Perception through Hierarchical Computation Structures*, Academic Press, New York, 1980.

[Tanimoto 83a] Tanimoto, S.L., Algorithms for Median Filtering of Images on a Pyramid Machine, In: *Computing Structures for Image Processing*, Duff,M.J.B. (Ed.), Academic Press, 1983.

[Tanimoto 83b] Tanimoto, S.L., A Pyramidal Approach to Parallel Processing, *Proc. 10th Int. Symp. on Computer Architecture* pp. 372-378, 1983.

[Tanimoto 84] Tanimoto, S.L., Sorting, Histogramming, and Other Statistical Operations on a Pyramid Machine, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Terzopolous 82] Terzopolous, D., Multi-Level Reconstruction of Visual Surfaces: Variational Principles and Finite Element Representations, A.I.

Memo 671, MIT AI Lab, Cambridge, MA, 1982. Also in: *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Springer-Verlag, 1983.

[Terzopolous 84] Terzopolous, D., Multiresolution Computation of Visual-Surface Representations, Ph.D. Thesis, MIT, Dept. of EE & CS, 1984.

[Thomas & Martin 84] Thomas, B.H., and Martin, W.N. Pyramid Structures in Dynamic Scenes, *Proc. Int. Conf. on Pattern Recognition* pp. 1008-1010, 1984.

[Thompson & Barnard 81] Thompson, W.B. and Barnard, S.T., Low-Level Estimation and Interpretation of Visual Motion, Computer **14**(8):20–28, 1981.

[Uhr 72] Uhr, L., Layered "Recognition Cone" Networks that Preprocess, Classify, and Divide, *IEEE Trans. Computers* **21**(7):758-768, 1972.

[Uhr 78] Uhr, L., "Recognition Cones", and Some Test Results; The Imminent Arrival of Well Structured Parallel-Serial Computers; Positions, and Positions on Positions. In: *Computer Vision Systems*, Hanson, A. and Riseman, E.M. (eds.), Academic Press, New York, 1978.

[Varga 62] Varga, R.S., *Matrix Iterative Analysis*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1962.

[Weems 84] Weems, C.C., Image Processing on a Content Addressable Array Parallel Processor, Ph.D. Thesis and COINS Tech. Report 84–14, U.Massachusetts, Amherst, 1984.

[Williams & Glazer 82] Williams, T., and Glazer, F., Comparison of Feature Operators Use in Matching Image Pairs, In: *Image Sequence Processing and Dynamic Scene Analysis*, T.S. Huang (ed.), Springer-Verlag (NATO ASI Series), New York, 1983.

[Wong & Hall 78] Wong, R.Y. and Hall, E.L., Sequential Hierarchical Scene Matching, *IEEE Trans. Computers* **27**(4):359–366, 1978.

[Yachida 81] Yachida, M., Determining Velocity Map by 3-D Iterative Estimates, *Proc. 7th. Int. Joint Conf. on Artificial Intelligence*, pp. 716–718, Vancouver, B.C., Canada, 1981.