SEGMENTATION OF STATIC AND DYNAMIC SCENES

by

John M. Prager

COINS Technical Report 79-7

May 1979

Author's current address: IBM Scientific Center
545 Technology Square
Cambridge, Massachusetts 02139

This work was supported in part by:

The Office of Naval Research
Grant No. N00014-75--C-0459

and

The National Institute of Health
Grant No. R01 NS 09755-07 COM

ABSTRACT

Segmentation of Static and Dynamic Scenes

May, 1979

John M. Prager

B.A., University of Cambridge

M.A., University of Cambridge

Ph.D., University of Massachusetts

Directed by: Professor Michael Arbib

One of the most important, and difficult, problems in scene analysis is to perform a segmentation of the visual image. We are interested in solving this problem, paying special attention to current neurophysiological and psychophysical data, to gain insight from how animals and humans perform this task. In the real world, though, there is constant motion, both of the observer and the environment. Thus we are led to the analysis of dynamic scenes.

In fact, we claim that segmentation of dynamic scenes is in many ways easier than that of static scenes, since there is information about object boundaries that results purely from the motion in the scene. We extend the concept of optic flow to form a low-level representation of the observed motion, and develop a program called MATCH to generate this flow-field. We apply MATCH to a sequence of frames of real data and show that by using predictive mechanisms we can greatly reduce the computational effort required to analyse the third and subsequent frames.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Artificial Intelligence and Brain Theory are two fields of scientific endeavour with much in common, for they both deal with perception, learning and cognition. Artificial Intelligence (AI) is concerned with understanding the principles behind these concepts, especially with respect to embodying them in computer programs. Brain Theory (BT), on the other hand, is concerned with how these functions are performed in living organisms. Regrettably, there has been very little contact between these two fields up to now (but see Arbib 1976).

In this thesis, an attempt is made to bring together AI and BT within the domain of visual scene analysis. For performing the necessary computations on the visual data, algorithms will be presented which may be integrated into a complete robot vision system. The motivation for, and structure of these algorithms, however, is derived from a study of psychophysical and neurophysiological data on biological visual systems. In particular, all of the proposed algorithms are inherently parallel in nature, as distinct from the great majority of current AI techniques. It should be noted, however, that parallelism is being given increasing attention within the field of AI.

Scene analysis in Artificial Intelligence has ranged from programs which worked in very limited domains, for example, on collections of polyhedra viewed under favourable lighting conditions (Waltz 1972), to large, generally applicable systems, such as the VISIONS project (Hanson and Riseman 1978a,b) which seeks to understand natural outdoor scenes. All of these systems share one very important subproblem, which has not yet been satisfactorily resolved. This is the problem of segmentation, that is, the division of the scene under consideration into its fundamental components.

Most of the work to date has been on static scene analysis, that is, on analysing single images or views of scenes. Indeed, in Chapter 2 we present a system for the segmentation of static scenes based upon the extraction of boundaries. The focus of this thesis, however, is on the segmentation of moving scenes: the visual data will be in the form of a sequence of snapshots or frames, taken with only a brief interval between each frame, as with a regular movie camera. The data may be of moving objects viewed by a stationary camera, a stationary environment viewed by a moving camera, or the general case of a moving camera and moving environment. It is felt that the system presented here is particularly well suited to the second of these conditions, which will apply, for example, to an exploratory robot on the Lunar or Martian surface.

Visual scene analysis is commonly discussed in terms of low-level and high-level analyses. A low-level analysis, which generally includes segmentation, does not usually rely on, or produce directly, any semantic knowledge about the scene in question. A high-level analysis will invoke domain-specific knowledge in order to produce an interpretation of the scene. These processes are usually kept separate, although the

IGS system (Tenebaum and Barrow 1976), for example, does not make this distinction; in this system, semantic information is integrated into the segmentation process. Since only the early stages of biological visual processing (which correspond to low-level machine vision systems) are at all well understood, segmentation is felt to be an appropriate subject for study in the light of both AI and BT.

1.2 Overview of the Thesis

In Chapter 1 we introduce the problem of motion analysis by machine and refer to work already completed (or in progress) in this field. We contrast and compare this problem with those of change-detection and stereoscopic vision, pointing out the essential similarity of motion analysis and stereopsis on interchanging the dimensions of space and time.

We describe the aims of the research to be the production of a system which can segment its dynamic visual input. The analysis is entirely "low-level"; no use of semantics is made and no identification or interpretation of the input is attempted. The visual input is in the form of frames of data as in a movie film; this data is digitized onto a square grid, each grid position recording the intensity of the incident light. In contrast to "high-level" interpretive programs, the processing is performed in an entirely local fashion. The system will consist of retinotopic arrays of processors and memory cells, each processor in an array or "layer" being identical to all others in the array, all operating in parallel and with no executive control.

Later in this thesis we will demonstrate how dynamic scene analysis may be performed at a lower cost per frame than static scene analysis. Both because of the intrinsic interest in static scene analysis, and so that the comparison with dynamic scene analysis may be made more meaningful, Chapter II will be devoted to the problems of static scene analysis. In that chapter, we will present a system for the segmentation of static scenes using a boundary analysis approach.

The magnitude of the problems in scene analysis is evidenced by the complexity and/or number of processes needed to perform a segmentation of a static scene. In our case, there are several stages to the segmentation process. The raw image data are firstly preprocessed to remove certain kinds of noise; this output is then differentiated to generate an edge-representation. These tentative edges then undergo a relaxation process to boost or inhibit them according to context. Finally, the remaining edges are labelled as extended line-segments, some of which may then be eliminated in a post-processing phase.

Some of the discussions in Chapter II will serve a dual (dialectic) purpose. The same pre-processing steps as are used in the static case may be applied to good effect to frames of motion data, prior to the feature-extraction and matching processes. Furthermore, although the particular iterative processes described in Chapters II and V are quite different, they are both examples of relaxation processes. Chapter II, therefore, provides background for the processing described in Chapter V.

In Chapter III, we look at some of the data which has been discovered in physiological and psychological studies. We first examine the

anatomy and physiology of mammalian visual systems, and discuss the extent to which this study will influence our development of a machine vision system. Two aspects of the visual system are treated in particular. The evidence for the existence of two parallel visual systems in mammals is presented. The X and Y systems are described and it is shown how they might subserve shape and motion analysis respectively. It is appropriate to present here also the psychological data which suggest strongly that motion and shape perception in man are mediated by two separate, parallel systems. The implications of this for machine vision are discussed.

The organization of receptive fields is examined since we hope to gain an indication of the kind of elementary features extracted by the visual system. In particular, we investigate reasons for large receptive fields, so dominant in biological systems.

In Chapter IV we look at a variety of psychophysical experiments and phenomena which relate to motion perception. We attempt to infer something of the mechanisms involved and postulate certain basic computational functions in motion perception. We first look at Gibson's work on Optic Flow, and see that it is a very suitable form of representation of the movement in terms of changes between successive frames. We examine the properties of Optic Flow in terms of the simulated motion of a robot through a forest. It is shown how the flow that results may easily be resolved into translation and rotation components using entirely local, parallel computations.

We next develop a feature-extraction process, which is to extract from the frames of motion data certain 'feature-points', i.e. the stimuli to be matched. A metric or distance between different 'feature-types' is defined; this measure determines the likelihood of matching the different feature-types to each other. The 'similarity' of feature-types is related to the ease with which the underlying patterns of data can transform into each other during motion.

The studies by Johansson and his colleagues on moving dots and the use of the 'common vector' concept are examined. Certain phenomena such as those of induced motion and Johansson's common vector suggest the need for an 'Attraction Function'. This is related to Burt's 'Field Velocity'. Certain 'filling-in' properties of the human visual system are studied in conjunction with Weisstein's phantom moving contour experiments.

In Chapter V, we present a series of programs exhibiting properties and features introduced in earlier chapters. The central component is a program called MATCH which takes as input sets of feature-points extracted from frames of the input data and matches the points in successive frames. Experiments are performed with the program using different receptive-field organizations and different input conditions. This is followed by a discussion of the use of 'moving-memories' to propagate the internal representation of the motion from one frame to the next. The results of applying these processes to a sequence of frames of real data are presented. The chapter ends with a discussion of occlusion, and how the existence of boundaries may be inferred from the occurrence of occlusion. The integration of outputs from separate segmentation

processes is also discussed.

In the final chapter, we review what has been achieved, and examine where work remains to be done.

1.3 Nomenclature

1.3.1 The Data.

We describe here the terminology used for the data and data-structures employed in this work. The word 'scene' will be used to refer to that part of the environment which is being looked at, but not to any one particular view. Such a view, as represented by a photograph, will be called an 'image'. The sequence of snapshots or pictures that are taken by a movie camera are all images, but will sometimes be called 'frames'. The word 'frame' will generally be used when a picture is being considered with respect to other pictures in the sequence, and 'image' when it is considered alone.

The imaging device referred to or implied during the course of this thesis will sometimes be an eye and sometimes a camera. Since the data-structures used here for machine vision may be likened to those used in early biological visual processing, at least at an abstract level, common terminology may be used, so that a camera may be thought of as having a retina, for example. Just as the (mammalian) visual system is replete with retinotopic maps of the visual input, so our program will use arrays of data which are also 'retinotopic'. These arrays will sometimes be called 'layers' to emphasize the analogy.

The data in these arrays will be the digitized light intensity or grey-level recorded by a camera. The sizes of these arrays may typically be from 128 x 128 to 1024 x 1024 points or bigger. The intensity recorded at any point or 'pixel' (picture-element) may be thought of as representing the light impinging upon a retinal (rod or cone) receptor. It may at times be convenient to think of a set of retinotopic arrays as a single array with vector-valued pixels; thus a colour image may be a single array with three values at each pixel, representing the blue, green and red components of the incident light.

1.3.2 The Processing.

The computations in which we are interested will be performed by arrays of 'processing elements' (or PE's) in register with the data arrays. These PE's do not necessarily have an exact neural correlate; indeed, the precise localization of, and distinction between data and processing in the brain has by no means been worked out yet. It will be sufficient for us to imagine that the computations will be carried out by neuron-pools arranged retinotopically, as discussed above.

In general, a given PE may receive input from many points in many different layers; however, two particular configurations will be of great interest to us. For certain computations, such as uniform scaling of intensity, the input to a PE will be solely from points in direct topological correspondence with it. This kind of computation involves a representation of the incident light at a point and possibly some global parameters, but does not permit interactions between neighbouring points. The other kind of computation in which we are interested

concerns precisely this interaction. In this situation a PE will receive input from a small region of points, in one or more layers, centering around the topological map of the PE in those layers. This kind of connectivity pattern is necessary when the computation must proceed until the data is in a locally consistent state, which is usually the goal of a relaxation process (see Section 2.4).

Throughout the thesis, the set of points which act as input to a PE will be known as its 'receptive field'. The set of points to which the output of the PE is sent will be known as its 'projective field'. In the case of relaxation, when the computation proceeds until certain conditions are met, the output from one iteration of the process will act as input for the next, so the projective and receptive fields will lie in the same layer(s).

Although neighbouring PE's may share data through overlapping receptive fields, the actual computations occur independently of each other and will be thought of as being carried out in parallel over the whole layer. In the results reported here of running programs on the computer this parallelism is of course simulated, although it is anticipated that an array processor will eventually be used for image analysis of this kind.

1.4 Motion Analysis and the Stimulus-Matching Problem

It was mentioned earlier in this chapter that one of the major goals in scene analysis is segmentation, and that the objective of this work is to study the segmentation of dynamic scenes. It might seem that the easiest and most appropriate solution to the moving-scene case is to segment completely each frame of the data, to identify the components and to match them up from one frame to the next. Now, while the last of these stages would presumably be a fairly simple matter, especially if the foregoing identification is at the object level, the segmentation and identification problems are not. Segmentation and identification in static scene analysis is still a subject of intense study as provably efficient and accurate techniques do not yet exist. Besides, to perform a segmentation from scratch for each frame would be computationally prohibitive (at least if anything like real-time processing is required), and furthermore, would completely miss a very important point: that there is segmentation information inherent in the motion that is a property of the relationship between successive frames. It is by studying invariants and changes between frames that we will be able to perform a segmentation both efficiently and accurately.

In order to do this we must select certain attributes of the images that we will study from frame to frame. We have argued above against using presegmented regions, and will use instead certain features of the images. A detailed discussion of the extraction of these features will be presented later, but for the moment we will consider a feature to be an easily detectable attribute of a small region of an image. Our features will lie predominantly in areas of high variance, since that is where the visual information is concentrated, and may be typified by 'edges', 'corners' and 'blobs' (small, circular-shaped regions of significantly different intensity from their surroundings). Points

in the images where features are detected are known as 'feature-points'.

Now, in order to chart the progress of these feature-points from one frame to the next, it is necessary to know to which feature-point in frame N (if any) a feature-point in frame N-1 corresponds. This problem lies at the heart of motion analysis, and is known both as the Stimulus-Matching Problem (Burt 1976) and the Correspondence Problem (Duda and Hart 1973).

1.5 Related Fields

There are several closely related fields in which the stimulus-matching problem is the basic problem. Because of the applicability of solutions in any of these fields to any of the others, the relevant work done in stereopsis (depth perception), change-detection and on the video-telephone will be reviewed here along with the previous work in motion analysis. After a brief discussion of these fields and the problems common to most or all of them, a survey of the more important work done in them to date will follow.

Stereopsis, or binocular vision, computes the distance of points in the scene from an observer, also known as 'depth' by analysing two images formed simultaneously by two imaging devices, separated along a line roughly perpendicular to their line-of-sight. The relative distance between corresponding points in the image plane is known as 'disparity', and is inversely related to depth. Many animals with two eyes with a region of binocular overlap use this mechanism for the measurement of depth in that region.

It is interesting to note that in many respects stereopsis may be considered to be a special case of motion analysis. The latter in general consists of arbitrary camera/eye movements and arbitrary motions in the environment. If the camera/eye moves perpendicular to its line-of-sight at such a velocity that it travels in the time between the taking of each frame the displacement between the two camera/eyes in the stereopsis setup, then the correspondence problem for those two frames is the same for the stereo and motion cases, as long as the environment moves relatively slowly.

Change detection is a problem that is pertinent to aerial reconnaissance. Given two flights over the same territory, it is often desired to know what changes have taken place on land between the flights. Since it is impossible to take pictures from exactly the same altitude and under exactly the same lighting and camera conditions, the detection of change must await the registration of the two images. This in turn depends on finding corresponding (static) objects in the two images.

The detection of movement in television signals has applications in, for example, automatic surveillance, but its most important use may be in the videotelephone. In order to achieve efficient transmission of large amounts of video information over low band-width lines, techniques known as 'conditional replenishment' are being developed to transmit only those parts of pictures which change. To extract this information, change in position of points from frame to frame must be computed.

In order to make the review of existing systems more meaningful,

we will firstly discuss in general terms the characteristics which are to be noted in these systems, and which distinguish one system from the next. We will also look at the possible sources of error, since some systems make a special effort to eliminate such errors. It is hoped that the review will benefit when approached from this perspective.

1.5.1 System Characteristics

What is moving. It was mentioned earlier that the general problem of motion and stereopsis is of a moving camera in a moving environment. This is a much harder problem than if either were stationary, since almost every point in the scene will appear to be moving. With a stationary camera, any motion can be attributed to the intrinsic motion of the object. This is the case which is usually studied. If the camera moves through a stationary environment, (or in stereopsis), then most points in the scene will appear to move, but their motion will be a function of their depth; if the velocity of the camera is known, then the depth may be deduced exactly. In the case of general motion it is impossible to assign either depth or velocity to any point in the scene, although the ratio depth/velocity may be found. The optics and equations involved in the case of a linear retina are discussed in Chapter IV.

Number of frames. In change detection and stereopsis, only two images are usually compared, while motion will be tracked in a long sequence of frames. Having more than two frames in sequence can be advantageous for two reasons. Firstly, any movement detected between frames N and N+1 can be extrapolated under the assumption of no or slow acceleration. In this way, an immediate ballpark estimate is deduced for the locations of corresponding points in frame N+2, thus cutting down search time. The other advantage is that any mistakes made in initial matching between frames can be spotted by their failure to allow consistent extrapolations over time. These mistakes cannot easily be detected with only two frames.

Numeric vs. Symbolic. The data produced by commonly used digitization techniques is in the form of arrays of numeric values coding the light intensity projected to the 'retina'. The data may undergo transformations, but as long as the topological structure is preserved, the format remains the same, and is known variously as 'numeric', 'iconic', and 'retinotopic'. These terms all imply that the location in space or on the retina of a point resulting in a data value is encoded implicitly by the data item's position in the array. On the other hand, the data may be represented by a graph- or list-structure; this is known as a symbolic representation, where coordinates are given explicitly. The matter of when (if at all) data may pass from a numeric to a symbolic representation is a subject of on-going debate in the scene-analysis community.

Parallel vs. Serial. Related intimately to the mode of representation is the question of whether the computations may be carried out in parallel over the image. In retinotopic representations, it is evidently possible, and algorithms for parallel computation tend to be elegant and simple, since there is no problem of "what do I do next?". Graphical and other symbolic representations do not lend themselves so naturally to parallel operations.

This distinction is also related to the location of control of the processing. In the retinotopic, parallel case, the control is distributed, or even non-existent. Symbolic processing, on the other hand, is usually performed under central executive control. Indeed one of the characteristics of symbolic processing is the need for back-tracking; such a concept rarely applies to parallel processing.

Levels of correspondence. One of the basic problems that must be faced in motion analysis is the extraction of suitable features to match between frames. There is a range of complexity of these features. This range may be characterized by small windows of intensity values at one end, through features such as 'edges' and 'blobs' to regions representing entire objects or large parts of them, at the other. The sophistication of the motion analysis required depends upon what kinds of features are to be matched.

Authenticity of data. Allied to the problem of level of representation is that of the 'authenticity' of the data. Those systems which only use simulated data will generally fail to address the problems (listed below) encountered by those systems working on real data. Use of simulated data is potentially misleading, since programs which require points to be present in exactly the same form in all frames (in which they are visible) may fail miserably when run on real data.

Use of neighbourhood to guide search. Unless the scene consists almost entirely of long thin objects - quite a pathological case - it will generally be true that the majority of the neighbours of any point in an image belong to the same object as does the point. Under the assumption of rigid motion, or even slow deformation, most points in the image will move with a vector velocity very similar to that of their neighbours. This is a potential means of greatly reducing the search effort for matches for these points; this observation is used to varying degrees by the systems described below.

Occlusion. If each point in one frame was guaranteed to appear in the next frame, with no extraneous points, then the stimulus-matching problem would be much less difficult - it would essentially be reduced to a problem of optimization subject to constraints. Indeed, Ullman (1978) models the general problem exactly this way. However, with occlusion, the problem is much more complex. Some of the systems we will meet cannot deal with occlusion in any way. The sophistication of a system can often be judged by how it handles this important problem.

1.5.2 Sources of Error

There are many sources of error and inaccuracy when dealing with real data. The more important ones are listed below.

Registration error. This error occurs when the two frames being compared are not centered around exactly the same point, with their axes aligned. This is the predominant problem in change detection, and occurs in motion analysis when the camera jogs, and in stereopsis if the cameras are not aligned properly.

Perspective distortion. In a perspective projection, points in space are projected onto the retina through the centre of the lens of the imaging device. In parallel or orthographic projection, the lines of projection are perpendicular to the retina. The projections are

somewhat different, especially at the periphery, but these errors are usually ignored. It should be noted, though, that parallel projection to parallel retinas is not useful for stereopsis.

Changes in illumination. This problem is of much greater importance in change detection, where the photographs may be taken months apart, than in stereopsis or motion analysis, where they are taken simultaneously or within a fraction of a second. The changes can generally be corrected through density and contrast normalization techniques.

Topological changes. The shape of an object's projection changes due either to the object's movement, especially rotation, or the camera's movement. These changes are typically kept to a minimum by using high sampling rates.

Digitization errors. These are generally of two kinds. The first is due to the quantization of the images, both spatially into rectangular grids of, say, 256 x 256 pixels, and in intensity, into, say, 64 grey levels. (The frame sampling rate is a temporal quantization.) The finer the resolution one achieves, the better the performance one expects to get, but at the cost of greatly increased execution time. The second kind of error is simply noise, due to random electrical events, residual images in the camera, and from other sources. Some techniques exist to reduce these errors. Two particular algorithms are presented in Chapter II.

1.6 Literature Review

1.6.1 Motion Analysis

One of the earliest instances of motion analysis was in the tracking of clouds in satellite photography. The first two papers we will discuss are concerned with the tracking of clouds under conditions of occlusion. Both use idealized models of the clouds. Aggarwal and Duda (1975) use artificially generated polygons, whose vertices are perturbed randomly to add some verisimilitude to the problem. Chow and Aggarwal (1977) digitize scenes of models of clouds and aeroplanes painted in white on a black background; random digitization errors are introduced naturally by this method, but the contrast of the figures is such that there is no problem in finding the boundaries. Both systems therefore use the same kind of input, so may readily be compared.

Aggarwal and Duda assume that their polygonal objects are rigid but are otherwise arbitrary. The features they use on tracking these polygons are vertices; they distinguish two kinds, namely, 'true' and 'false' vertices. True vertices are the corners of the original polygons, while false vertices occur where the polygons intersect. The angles of true vertices never change between frames (except maybe for minor perturbations) while false vertices often do. If the length of an edge segment changes by more than some threshold value, then at least one of its vertices is false. If the angular velocity of two edges incident at a vertex are not comparable, then the vertex must be false. These observations form the basis of their algorithm, which is

now briefly described.

An attempt is made to match polygons between frames on the basis of their real vertices, and the similarity of the shape of the polygon. This is not always possible, due to the occurence of false vertices. Under the further assumption that at most one topological change occurs between frames, vertices can be classified accurately, and polygons matched. This assumption is not foolproof, but is increasingly realistic with increasingly smaller time-intervals between frames. A model is maintained of the ongoing motion; at each stage the updated model is a function of the previous frame, the current frame and the current model. The model allows the tracking of the polygons, and, in fact, permits the generation of a complete description of a polygon, even if only several partial views of it were available. The model is used in a predictive mode to find the match for vertices; two 'consistency conditions' must be obeyed. (1) Knowing the current estimate of the velocity of a vertex, the location of its match must be within a 'radius of uncertainty' of its predicted position in the next frame. (2) The angles of a vertex and its match must be approximately the same.

The primary goal of Chow and Aggarwal (1977) is to remove the requirements of a) polygonality and b) at most one topological change between frames. However, to do this, they assume that there is no occlusion for the first frames, so that the program can generate for itself a complete description of the objects. The objects are described in terms of position, area and principal axes. These are simple, global features, and are not sufficient for correcting mis-matches, as will be seen below. The assumption of no occlusion for the first two frames guarantees a match between objects, so that their velocity can be estimated and used in further matching, via a predictive model. This latter process is rather simplistic. While occlusion takes place, the objects are assumed to have constant velocity; their positions are predicted, and as long as the predicted scene matches the actual scene, the program is happy; when a mis-match occurs, it halts. It has no way of making corrections to its model under these conditions.

Both of these systems work well under the assumptions listed, including the implicit one that the objects can be easily segmented prior to analysis. It is by no means clear, however, that these assumptions are realistic. Objects may change topologically, perhaps to the extent of having holes open up in their interior - clouds in particular may do this. There is no guarantee of constant velocity, nor on precise limits on the rate of topological change of images. In the general case, objects may rotate out of the plane; it is not apparent that systems like these can be extended to cover general motion.

The two systems just described require a segmentation of the scene prior to analysis. We are more interested in systems which use motion to aid segmentation, which is what Potter (1977) claims to do. For the first frame he selects for future match one point in ten along both X- and Y-directions. For each of these points he scans to the left, to the right, up and down until he finds a discontinuity in intensity. The lengths of the arms of the 'cross-template' so-formed become the descriptor of the point, and these, along with the intensity of the cen-

tre-point of the cross, are the features used in the match. The match-process is simply a search for a matching template in the next frame. Points are grouped on the basis of similarity of velocity.

In the case of finding no match for his templates, Potter allows his templates to expand and shrink, within certain limits, so that his system can detect motion in the Z-direction. However, his system cannot handle rotations, nor is there any capability for dealing with occlusion. In looking for the end-points of the arms of his templates in his software-generated images, he is looking for discontinuities in intensity, which he interprets as indications of parts of the boundaries of the objects. For his artificial data, it would make more sense to form complete boundaries by thresholding the discontinuities in intensity, and dispense with the templates altogether. On the other hand, it is unclear whether with real data he would consistently find the boundaries of an object in the same position relative to its centre, if he could find them at all.

Some interesting work has been done by Nagel's group in Germany on analysing motion in real images taken of traffic scenes. They use a stationary camera, and analyse scenes with 'sparse' movement. Their work concentrates on separating the objects from the background. They make no mention of the occlusion of one moving object by another, and it is in general unclear how their systems would handle moving objects just passing close by each other.

Jain et al. (1977) define a 'geopixel' as a 6 x 4 window on the image and compute for each geopixel the mean and mean-square grey-level.

They use the maximum-likelihood ratio test between frames to test for identity of geopixels. If there is no match, they look at the next frame. If there is still no success, they signal a mis-match. This process is iterated for N frames. Mis-matched points are clustered, an enclosing rectangle is drawn, and the whole procedure is repeated for the next N frames. The rate of expansion of this rectangle gives an estimate of the velocity of the motion. This is a fairly simple technique, but seems to be reasonably effective, as far as it goes.

Dreschler and Nagel (1978) is represented as an advance over Nagel (1976) in that certain restrictions on the movement are relaxed. They require that images of a given moving object be more similar to each other than to other moving objects, that images of moving objects exceed a certain (minimal) size and that there exists a reference frame in which the image of a moving object does not overlap the image in the current frame. By differencing the current and reference frame, two areas of large grey-value difference are found. The average contrast of each boundary is found in the intensity image. The larger-valued is taken to represent the boundary of the object in the current frame. (The other is the boundary of its 'motion-shadow', i.e. where it was in the reference frame.) Features such as area, perimeter, mean and variance of intensity, eccentricity and fractional fill of minimal bounding rectangle are evaluated for all these 'object-candidates'. A minimal-spanning-tree clustering algorithm is used to group the object-candidates; the image of a moving object will appear as an elongated cluster, due to slight changes in features from frame to frame. As well as fail-

ing to deal with occlusion, this system suffers on occasion from camera jitter, which causes prominent edges to appear in the difference image. These may be removable by the clustering algorithm, but it seems unlikely that the system could be used directly with a moving camera, since registration of frames is an implicit assumption in their algorithm. The main trouble with their algorithm, though, is the reference frame. Its specification is done manually at the moment. To do it automatically would require a prior solution to the motion analysis problem!

A rather novel feature is used by Radig (1978). As well as the intensity he uses the slope of small windows on the intensity profile of the image. He transforms an image into a graph in which the nodes contain information about the position, intensity and slope of small windows (equivalent to a 3-D gradient), and in which the arcs connect spatially adjacent neighbours. He filters first by restricting the ranges of these components, then on the basis of arc-consistency. He ends up with regions of isolated sub-graphs with internal local consistency, and evaluates features for each region. He next filters by histogramming and marking as disallowed values below a threshold. Arcs cannot exist between allowed and disallowed nodes, so he then splits the graphs into sub-graphs and iterates the process.

When the system converges, the sub-graphs remaining represent regions which are to be matched over frame-sequences by similarity of extracted features. In this way he can track the path of individual regions over time. He attempts a treatment of occlusion, using incomplete region-sequences not overlapping in time as candidates. He computes the similarity of the regions at the tail of the earlier sequence and head of the later sequence. If they are comparable, occlusion is inferred to have taken place. This technique only works strictly after the fact, however. He has no method of tracing what happens during the actual occlusion process - he has no model of occlusion - nor of even knowing for sure that occlusion is taking place.

Radig's system is computationally very expensive, taking two hours of computer time for 66 frames covering a time-period of 2 - 3 seconds. The system of Jain et al. is slightly faster, taking 25 - 30 seconds per frame, but neither system can reasonbaly be used in real-time motion analysis unless tremendous reductions of data and speed-ups of computation are effected.

1.6.2 Stereopsis

Space exploration, in particular the Mars program, has given rise to much work in computer depth perception and comparison of pictures. Some of that work is reviewed here.

Prior to the Viking missions, Levine et al. (1973) did some experiments in depth perception using desert scenes to simulate the Martian environment. Their goal is to solve the correspondence problem between points in a stereo pair of images, using one as a reference. The points which they choose as reference for the purpose of correlation are selected on the basis of high texture, since there is little potential information available in regions of uniform texture. They use an adaptive correlation window and some heuristic strategies to get effi-

ciency and accuracy. They let the significant dimension of the window depend upon the variance at the centre-point, and perform a coarse search followed by a fine search. The retinal disparity found between corresponding points gives the depth.

Levine et al. use two significant assumptions to reduce the search for corresponding points: (1) a "Proximity Assumption" that neighbouring points in one image will be neighbours in the other, if they belong to the same object, and (2) an "Ordering Assumption" that points in any row in one image are in the same order in the corresponding row of the other image. These are related to the "Continuity Assumption" of Thompson (1975): that the (vector) disparity of neighbouring points will be similar. Thompson uses this assumption to find the approximate locations of matches. He also uses correlation and hill-climbing to find the optimal match. His program is an extension of Hannah (1974) to which he has added a scaling technique to correct for perspective distortion.

The system described by Gennery (1977) and Moravec (1977) for obstacle avoidance by robot vehicles is somewhat similar to that of Thompson's. Moravec describes an interesting technique to reduce the search for matching points. The algorithm starts with a reduced (averaged) version of the image and the maximum correlation value is found for all NxN windows in the shrunken image. This procedure is iterated with images reduced one step (a factor of 2 in each dimension) less each time. The maximally correlating NxN window is expanded to 2Nx2N and this area is now searched. This procedure is very similar in principle to that of the "top-down" line-finder of Hanson and Riseman (1974). The advantage of this technique is that the number of correlations needed in a typical problem are drastically reduced over any brute-force technique. It is not clear, though, whether it is any better than any other directed searches.

Nevatia's (1976) depth analysis program is unusual in that he uses a sequence of images. Just as motion analysis can draw upon the slow rate of change apparent between frames closely spaced in time, so he uses a sequence of views taken at intermediate angles. He can track the path of regions between consecutive pairs of these images, so the correspondence between regions in the widely spaced initial and final images are more easily found. Rather than using cross-correlation between regions, he uses mean-square difference (for purposes of efficiency) and a hill-climbing technique.

A study by Ganapathy (1975) is also concerned with wide-angle stereo views. His objects, however, are artificially generated polyhedra, and his match-candidates are vertices. In the light of the general shape of real-world objects, especially naturally-occuring objects such as rocks and trees, it is unlikely that his system would have application in a robot system. He does perform a limited set of experiments on real data, albeit generated from polyhedra, but he indicates that much work in this area is still to be done.

Quite different from any of the approaches already described are the neural models of Trehub (1978), Dev (1975) and Nelson (1975). Dev and Nelson proposed rather similar models using facilitation and inhibition between binocular units. Dev's model will now be briefly de-

scribed. Using the ideas of Julesz (1971) she constructed a cooperative model for detecting disparity. Her model consisted of five layers of neural-like elements, representing disparities of -2, -1, 0, 1, or 2 units in the left-right direction. Inter- and intra-layer connections provided support for adjacent activity within a layer and inhibition between the same positions in different layers. Presented with a pair of stereo, linear, binary (simulated) images, the system would converge with high activity in those layers in the correct retinotopic positions to represent the 'objects' at the corresponding depth in space. While this parallel, distributed system conforms very much to the spirit expressed in this thesis, there are several problems with it, expressed well by Levine et al. in their criticism of Julesz:

> "....This forms the basis for an algorithm which is restricted to inputs which do not contain areas of uniform brightness, which are digitized into two grey-levels, and which do not contain excessive differences in depth. The latter condition is imposed in order to limit the number of difference fields. Apart from the severe restrictions on the input data, the main drawback of this method is that it requires an analysis of the complete set of difference fields, a significant task for complicated images with large depth variations."

Of course, having parallel hardware would put a completely different complexion on things. However, for making this kind of algorithm suitable for present-day machine vision systems, the three-layer model of Richards (1971) for those regions in front of, inside and behind the focal plane, might be appropriate. It might be noted in conclusion that Marr and Poggio (1976) suggest a cooperative model which is a variant of the Dev model (see Burt (1977) for a comparison).

1.6.3 Change Detection

Much of the work in change detection is from aerial and satellite photography. The first system we will describe here is that of Quam (1971) which was used in comparing pictures taken in the Mariner 6 and 7 fly-by missions. Quam's work laid the basis for that of Hannah and Thompson.

Quam first geometrically normalized the two images to a common point of view, and then photometrically normalized them to eliminate differences due to different camera characteristics and different reflectances in the scene. In order to perform these normalizations, he needed accurate geometric models of camera and scene, the photometric response of the camera, a reflective model of the scene, and finally to know that some points in the scene did not move so that the ensuing cross-correlation might be effective. The next step was registration by maximizing the cross-correlation between the pictures. In performing the registration he introduced the concept of 'misregistration vectors', which were the "predominantly translational alignment errors" after registration. He termed as 'blunders' the misregistration vectors which did not fit into any systematic model; these were caused by the appearance of clouds and shadows. He models the misregistration as a polynomial function of position in the image, and for each misregistration vector calculates the 'residual' as the difference between the empirical and predicted values. Blunders are removed by finding residuals which are large and do not cluster with other residuals.

The papers by Lam and Howt (1972) and Henrikson (1972) describe

change detection systems for aerial photography. Their requirements are unusual because (1) the photographs are taken by side-looking radar, and (2) they need real-time operation. Their purpose is to let the computer detect the changes, but allow humans to determine their significance. After preprocessing, images are registered by correlation, and then differenced. Detection of change is simple in this case - objects are bright against a dark ground. Their registration problem also is often simple, since most points in their images will not have moved.

Price (1976, 1977) uses a technique for change detection in some ways similar to the methods used by Nagel's group for motion analysis. He forms a complete segmentation of the images using Ohlander's (1975) algorithm. He generates features such as area, location, eccentricity, orientation, etc. for each region as do Dreschler and Nagel (1978) and compares each region in one frame with every one in the other. The matching process is guided somewhat by using a region's neighbours as part of its feature-description. The best-matching region is considered to be the corresponding one, even if there is no correct match. The time taken to perform his change-detection is very large. Even though 'planning' is used in the segmentation process, 24 minutes is a rather long time for a single segmentation.

Work is in progress at Bell Laboratories for the efficient coding of moving picture sequences for transmission and storage. The goal of Limb and Murphy (1975) is to estimage the velocity of moving objects, so that, for example, they only need send those parts of a picture which have changed from frame to frame (see Haskell (1975)). The calculate a very simple approximation to velocity using measures called the "frame difference signal" and the "element difference signal". The first compares corresponding points between frames, the second neighbouring points within one line of a frame. They must assume that all movement is in one direction. Indeed, only horizontal motion is measured directly, while vertical motion is measured statistically. Frames are delayed in order to make comparison between successive frames possible. By delaying individual lines as well, and generating a "line difference signal", they could measure vertical velocity directly, and hence more accurately. They are concerned with real-time operation and have built a system which operates entirely in hardware. However, their system cannot cope with rotations or multiple motions, much less occlusion.

1.6.4 Some Psychophysical Experiments

There are many points of contact between our ideas and those of Ullman (1977, 1978) and Stevens (1977). We will now briefly describe the work of these two researchers, and make further references at appropriate points during the rest of the thesis.

Ullman's (1977) thesis was directed at the problem of interpreting the motion and three-dimensional structure of a collection of moving dots, observed at discrete times. He obtains the result that three frames and five points (belonging to one object) are usually sufficient to infer the motion of the object. To do this, however, he must find the matches of the points between frames. He argues against correlation,

but for the extraction of units representing physical entities, such as edges, on which the correspondence is to be performed. Based upon psychophysical observations, he derives a scheme for predicting between which points apparent motion will be perceived. He defines the 'affinity' between potential matches to be, inter alia, a decreasing function of distance and orientation difference. He then computes the 'correspondence strength' between points, which is a function of their affinity for each other, and for neighbouring points. The greater the resultant correspondence strength, the greater the probability that motion takes place between the points.

He uses real-world constraints to impose structure from motion later on, but, unlike us, he does not use these constraints to help form the correspondence between two frames to predict that between subsequent frames. His algorithm is not iterative, however, but computes the correspondence in a single pass. It would appear that this is because he observes that in psychophysical experiments with just two or three dots, the movement of just one dot does not influence the movement of another (Ullman (1978)). He attempts to generalize from this, but it is not clear to us that the same effects would be observed from a large field of dots, or from just a few dots but much closer together.

In Ullman (1978) he uses a different approach to forming the matches, namely that of linear programming. He employs the ideas of parallelism, locality, and simplicity, making his paradigm similar to ours. His algorithm requires, however, that one processor be assigned to each element in the image. Although we do just that in the initial development of our algorithm MATCH, we find we can dispense with this constraint. His method is to minimize the sum of the displacements of the elements over possible matches, and he proves that in the case of translatory or radial 'flow', the correct match minimizes the total displacement of elements over 1 - 1 mappings. He says nothing about rotation, however; as we will see, our method handles rotation with no added complexity.

Stevens (1977) observed that by superimposing a pseudo-random dot pattern that has been rotated, translated or expanded, over the original pattern, one perceives circular, linear or radial Moire-like patterns. These patterns were found to have locally parallel structure, from which it was deduced that the human visual system is performing a correspondence between these dots in these situations. These percepts were found to be relatively independent of the dot density, but did depend on the amount of movement. He found experimentally that parallel groupings could be formed if, on the average, no more than two or three 'incorrect' dots were nearer a given dot than the 'correct', corresponding one.

He used the construct of 'virtual lines' as a representation of the local structure. These lines were drawn from a given dot to all other dots within a small circular neighbourhood, whose size is a function of dot density. His goal was to select one of these on the basis of its similarity of orientation to that of the local structure of the pattern. All of the virtual lines were considered, but the shorter ones were weighted more. He computed the local structure by constructing a histogram of orientations of these virtual lines in a small neighbourhood.

The peak of the histogram corresponded to the orientation of the pattern locally, and for each dot the incident virtual line which was closest to this orientation was chosen.

In further experiments, he superimposed two differently transformed copies of a pattern on itself. The original and one of the copies were of the same but variable intensity, the other of fixed, bright intensity. When all three patterns were of the same intensity, the locally parallel structure was difficult to perceive. When the elements of the variable pair were of lower intensity than those in the third pattern, then these were perceived as forming the Moire-like patterns, the bright dots forming the background. It may be concluded that the visual system will try if at all possible to form locally parallel pairings of points, with a preference to matching points of similar intensity. This may be generalized to a preference for similarity of any appropriate feature, such as orientation in the case of edges.

Some relations between Stevens' work and our own will now be outlined. As he does, we will attempt to match points in a manner that produces locally parallel directions of match, and we use the influence of the local neighbourhood of a point to do this. Our optic flow representation may be likened to his use of virtual lines. We favour matches between nearest neighbours, as does Stevens, but we compute 'distance' in a three-dimensional space composed of one feature-based and two spatial dimensions, whereas he only suggests using similarity of intensity in his histogram computation, but does not actually put it into effect.

While Stevens forms a histogram of orientations and computes the maximum in a dot's neighbourhood, we will use a weighted average. Although different quantitatively, it may readily be appreciated that these techniques give qualitatively similar results. We find the averaging method more elegant due to its lack of discontinuity.

Stevens concedes that his method is not very effective for considerable displacements between dots. In fact, were he to move rectangular grids of points by multiples of the grid-width, a paradigm we use extensively, he would find that his algorithm fails miserably. This is presumably largely due to the fact that, like Ullman, his algorithm is non-iterative, and does not let information propagate outside of small neighbourhoods.

1.7 Summary

We have seen many different systems using the various techniques of pixel-differencing, cross-correlation, vertex-matching and region-matching. Some handle occlusion, some do not. Differing restrictions are placed on the degrees of freedom of the motion permitted. Most motion analysis systems used estimated velocity from previous frames to predict positions in succeeding frames, though only a few used similarity of neighbourhood motion to constrain the search.

The system we will describe in Chapter V will have the following form. A very fast, partial segmentation will be performed by extracting features such as edges and corners. Matching will take place via a relaxation procedure which uses two consistency conditions similar to those of Aggarwal and Duda (1975). The internal representation or model

of the motion will be a vector velocity field originating from Gibson's idea of Optic Flow. The vectors here correspond to Quam's misregistration vectors, and indeed we will find that 'blunders' occur in motion too, only we will use them as an indication of occlusion. We can find regions (objects) by looking for discontinuities in the velocity field, but will not go so far as modelling occlusion at the 'level' of object. We will instead find occluding boundaries and so be able to segment all of our moving objects. We will use predictions from previous frames to guide analysis of successive frames, thus incurring a greatly reduced computational cost per frame.

CHAPTER II

SEGMENTATION OF STATIC SCENES

Introduction

We describe in this chapter a set of programs that are used to perform a boundary analysis of static images of outdoor scenes. Each process can operate in parallel across the whole image, since the value computed at a pixel is a function only of a small neighbourhood around it. Due to the modular nature of these programs, they can easily be "unplugged" and replaced by more sophisticated versions, or left out altogether if desired. The relaxation process in particular (described in Section 2.2.4) was designed to be a minimal functional implementation of the ideas behind the algorithm. A more sophisticated elaboration of these ideas is described in Hanson and Riseman (1978a). The techniques described in this paper have been developed for use in the VISIONS scene analysis system (Hanson and Riseman 1978b).

Overview of the System

The goal of the system is to form a segmentation of a static image, that is, to delineate the boundaries of the objects in the scene. This is done via differentiation using edge masks, a process which finds discontinuities in the intensity image. Prior to this stage, however, preprocessing is required to clean up the raw data somewhat. After differentiation, a relaxation process will consolidate the edges formed on the basis of local consistency requirements. Individual edges are then

linked together during the binding stage to form extended line segments. Finally, properties of these line segments such as confidence are computed, allowing removal of low-confidence lines.

Conceptually there are four stages to our line-finding process. Each of these is implemented as one or more computational modules.

(1) Preprocessing: this stage cleans up the raw data.

- (1a) UNMIX corrects for "mixed pixels" introduced in digitization.
- (1b) CONDITIONAL AVERAGE smooths out random noise and fine microtexture.

(2) Generating the Edge Representation: this is the heart of the whole process.

- (2a) DIFFERENTIATION finds the apparent edge-strength at each point in the image.
- (2b) SUPPRESSION removes "multiple edges" formed by spatial differentiation of boundaries which are composed of a gradient across many pixels.
- (2c) RELAXATION drives the probability of an edge at each point to 1 or 0 on the basis of local support or inhibition.

(3) Grouping: this stage joins edges into line segments and finds features of these lines.

- (3a) BIND joins contiguous edges together to form line segments, and each line segment is given a unique label.
- (3b) FEATURE EXTRACTION produces features such as length, contrast and location for each line segment.

(4) Postprocessing.

- (4a) TRIM1 removes selected line segments (e.g. short, low contrast lines).
- (4b) CONFIDENCE generates a confidence for each of the remaining segments that it actually is a meaningful line segment.
- (4c) TRIM2 removes low confidence lines by thresholding on their confidence level.

2.1 Stage 1 - Preprocessing

We start with the image digitized into three arrays containing the red, green, and blue components of the scene. These arrays are now averaged to form the black and white intensity image. Prior to differentiation, this image undergoes two preprocessing stages which provide the black and white image upon which the line-finding process works.

2.1.1 UNMIX (Step 1a)

The first process is deisgned to eliminate what is known as the "mixed-pixel" problem. This problem occurs whenever images are digitized, and is due to the fact that boundaries in the image will not in general fall in register with the digitization grid. Thus, the intensity recorded at a pixel might overlap two regions, and therefore represent a weighted average of them (see Figure 2.1).

The procedure must test to see if a two-step intensity gradient occurs at the same place in all of the three colour images. If it does, then a mixed pixel is assumed to have been formed (of course this assumption might not be correct and then errors would be introduced). It is consequently "unmixed" by assigning to it the values of its nearest neighbour along the direction of the gradient. This has the effect of shifting the boundary by a fraction of a pixel (see Figure 2.1c).

2.1.2 CONDITIONAL AVERAGE (Step 1b)

The second process is a variation of a smoothing process described in Lev et al. (1977) which helps eliminate noise in the image. In this

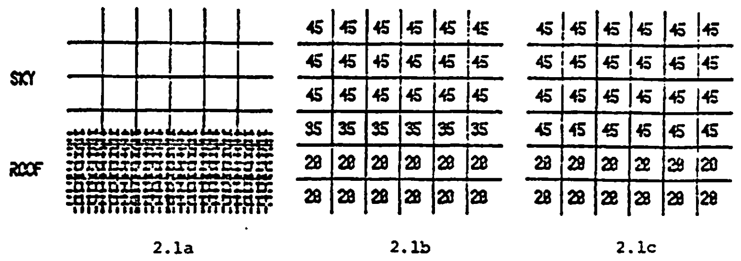


Figure 2.1. The "mixed-pixel"problem. 2.1a: Digitization grid superimposed upon a portion of an image. 2.1b: Intensity values recorded in this grid. 2.1c: "UNMIX" correction applied to 2.1b.

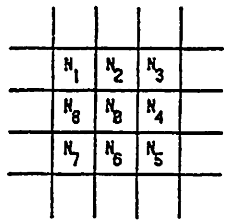


Figure 2.2. Neighbourhood considered in "CONDITIONAL AVERAGE".

process, the intensity value at each point is replaced by the average of itself and its neighbours, except that if the difference between the value of the point and a neighbour is greater than a certain value T, then that neighbour is not included in the average.

For the neighbourhood $\{N_i\}$ of the point N_0 in Figure 2.2, its updated value is given by:

$$N_0' = \frac{1}{n} \sum_{N_i \in S} N_i$$

where $S = \{N_i : |N_i - N_0| < T\}$ and n is the cardinality of S. Note that S always contains N_0. This procedure has the following effects:

(1) Within a homogeneous region, it smooths out small amounts of noise (small relative to T).

(2) Near a region boundary whose contrast is greater than T, it includes no points across the boundary in the average. This allows a smoothing of the points on either side of the boundary without blurring the boundary as a nondiscriminatory averaging process would do.

(3) Within an intensity gradient, the process averages a point with roughly as many other points that have small intensity as greater. This will smooth noise within the gradient, but it will not destroy the gradient.

(4) In a textured region, if the texture elements differ little in intensity (relative to T), they will be smoothed into a homogeneous region. If the texture elements differ by more than T, then no averaging will occur, except perhaps within the texture elements themselves.

After experimental testing of the differential operator (to be described later) on images that have selectively undergone the UNMIX and/or CONDITIONAL AVERAGE passes, it was subjectively concluded that an application of both processing techniques gives the cleanest results (i.e., no loss of any important lines, or addition of extraneous ones).

Examples of the combined use of these preprocessing stages on intensity images are shown in Figure 2.3.

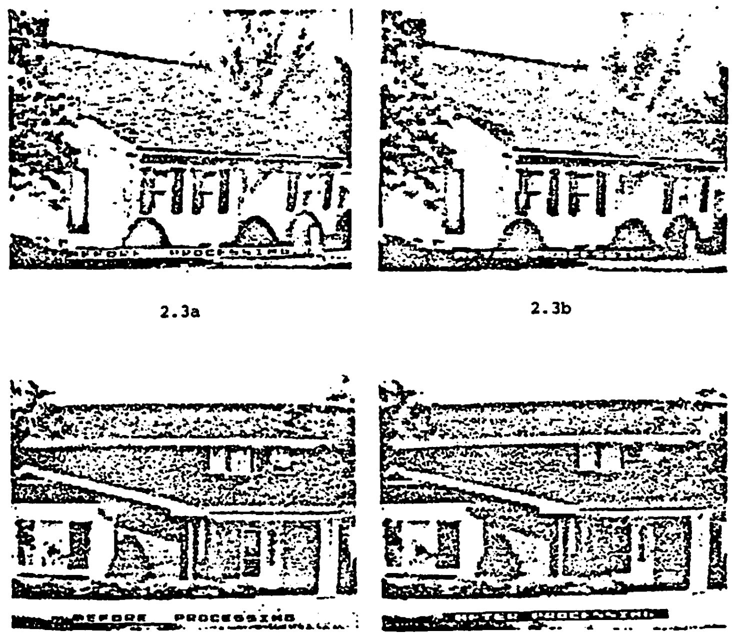
2.2 Stage 2 - Generating the Edge Representation

Representation of the Edge Image

The input consists of an array of numbers representing the light intensity of each position in the image. Since each of these pixels is to be in some region, it is reasonable to constrain the boundaries of regions to fall only between pixels. Representing the image on a rectangular grid and constraining edges to lie between pixels imposes a boundary that consists entirely of horizontal and vertical edges (as in Brice and Fennema 1970; Yakimovsky 1976). This greatly facilitates further processing, in particular the relaxation phase described in 2.2.3.

We are attempting to form segmentations of static images via boundary-based analyses. To find the boundaries we will be looking at discontinuities in the gray-scale in the images. We will now discuss briefly some considerations affecting the level at which the analysis will take place. By 'level' is meant how local or global a view is taken when extracting edges. A more global approach might attempt to extract entire boundaries, perhaps by a least-squares technique, while the more local approaches would extract the edge segments or components of the boundaries, which must then be joined together to form extended boundaries.

The overriding consideration in our case is that we are examining local, parallel techniques for scene analysis. In particular, we are



2.3a 2.3b

2.3c 2.3d

Figure 2.3. Preprocessing. Figures 2.3b and d show the results of applying UNMIX and CONDITIONAL AVERAGE to the images in Figures 2.3a and c respectively. Notice the sharpening of most of the boundaries and the smoothing of much of the texture, especially in the roof of the house in Figure 2.3b.

using analyses which 'see' through windows on the images as small as 3 x 3 (or 5 x 5 in the case of UNMIX). This leads us to the use of local edge-detecting mechanisms (masks), and local reinforcement techniques (relaxation) discussed in this section and the next. Indeed, the masks we will use are very small, taking into account only the most immediate neighbourhood of the edges concerned.

While this (temporary) neglect of other nearby information might seem to have serious drawbacks, it should be remembered that we are dealing with relatively low resolution images (256 x 256). A 1 x 2 mask applied to such an image is equivalent to an 8 x 16 mask on a 2048 x 2048 image. This latter size of mask is wholly unacceptable in our case, since it would cover an appreciable amount of the image, over which there is more useful information than could be captured using such masks. Indeed, the number of possible edges or combinations of them within an 8 x 16 window (and hence the number of masks needed to detect them) is too large to contemplate for current computing equipment.

Our data is digitized on a rectangular grid, and we find it most convenient to restrict our edges to lie along pixel boundaries. Thus line-segments, whatever their global orientation, will appear locally as a series of horizontal and vertical segments. For example, a line of orientation 73.5° (arbitrarily chosen) will appear as the zigzag line shown in Figure 2.4 where the ratio of vertical to horizontal segments tends to 3.38 = tan 73.5°.

It is clear that, at a local level, it is not possible to deduce a line's orientation except very grossly. That is, a line's local orien-

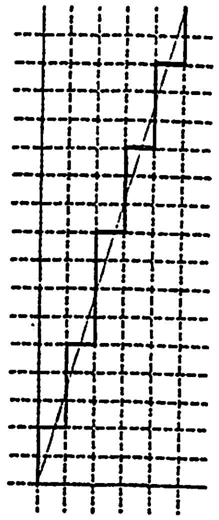


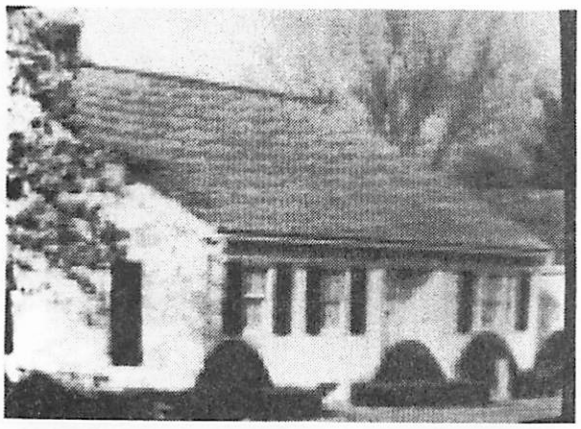
Figure 2.4. Digitization of a Slanting Edge. This figure shows how a line oriented at 73.5° to the horizontal would appear in a digital representation (heavy lines). The ratio of vertical to horizontal segments tends to 3.38 = tan 73.5°.

tation (horizontal or vertical) is not a good determinant of its global orientation. This idea influenced the edge relaxation process described in Section 2.2.3, where parallel or perpendicular continuations of edge segments are considered equally acceptable. With higher resolution, images and larger edge masks capable of determining the orientations of lines more accurately, the local orientation of the lines will play a more important part in the relaxation process.

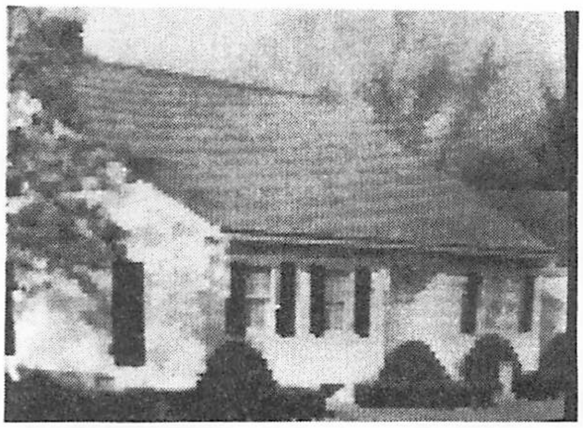
2.2.1 DIFFERENTIATION (Step 2a)

Differentiation is the most drastic transformation that the data undergoes, so it needs careful attention. Ideally, edges should be placed only between regions that differ with respect to some feature (in our case intensity), and nowhere else. In practice, problems occur due to texture within a region, blurred edges, gradients, etc. However, our simple preprocessing of the data will reduce the impact of these problems. Let us consider the three cases separately.

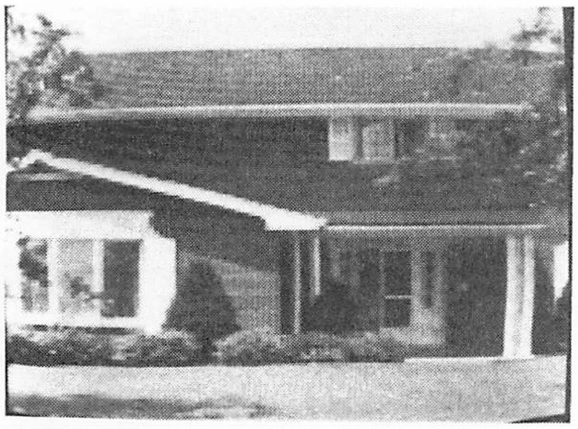
(1) Texture within a region. Fine low contrast micro-texture will have been largely eliminated by the conditional averaging process. Very distinct texture elements of high contrast will be prominent, and so will produce edges. At this point it is not the task of the differentiator to determine whether the edges are boundaries of texture elements or the boundaries of a textured region. Texture edges may be removed by a subsequent process which eliminates short and/or low contrast boundaries, or one that detects texture patterns if necessary. This will be performed when more reliable global information is available. Alterna-



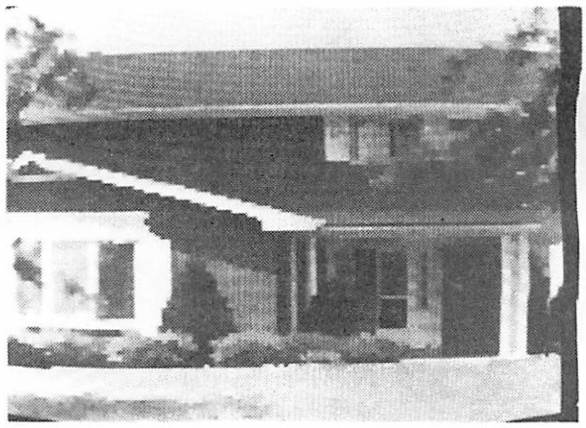
2.3a



2.3b

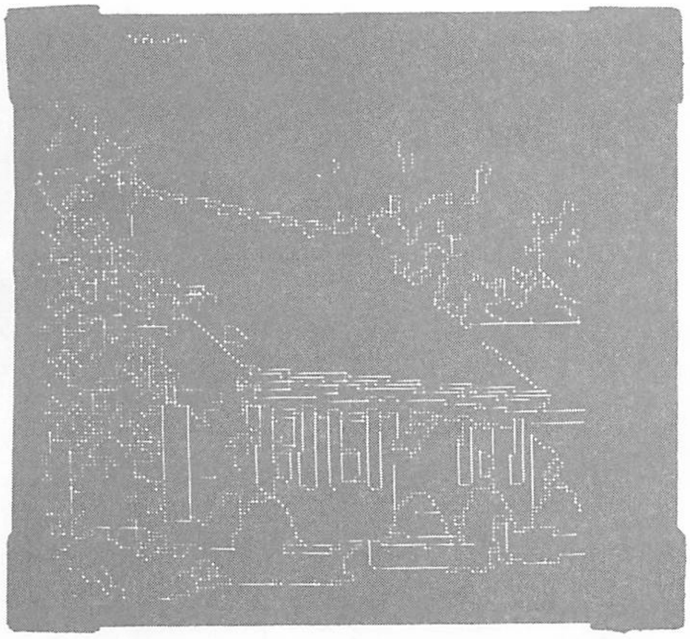


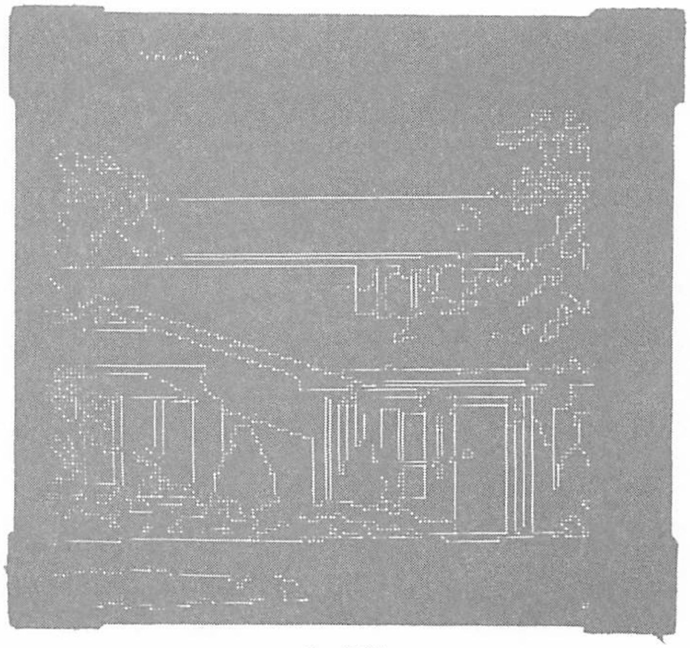
2.3c



2.3d

Figure 2.3. Preprocessing. Figures 2.3b and 2.3d show the results of applying UNMIX and CONDITIONAL AVERAGE to the images in Figures 2.3a and 2.3c respectively. Notice the sharpening of most of the boundaries and the smoothing of much of the texture, especially in the roof of the house in Figure 2.3b.


2.22a


2.22b

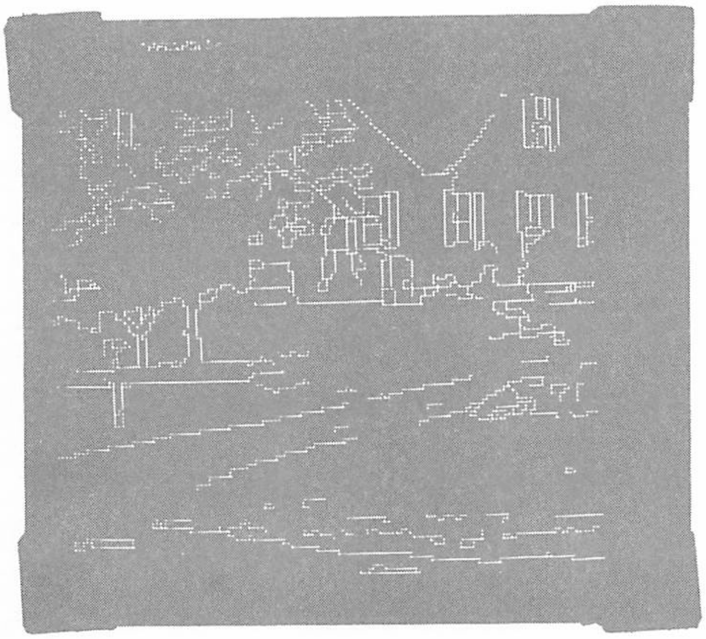
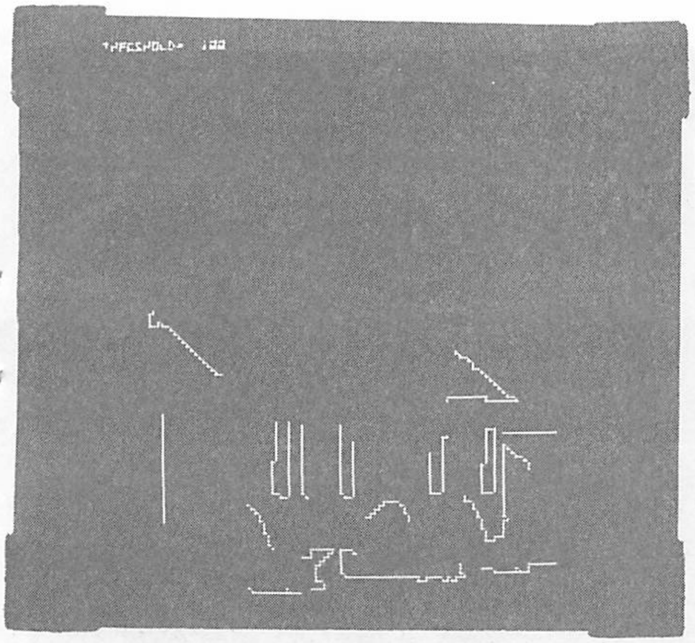
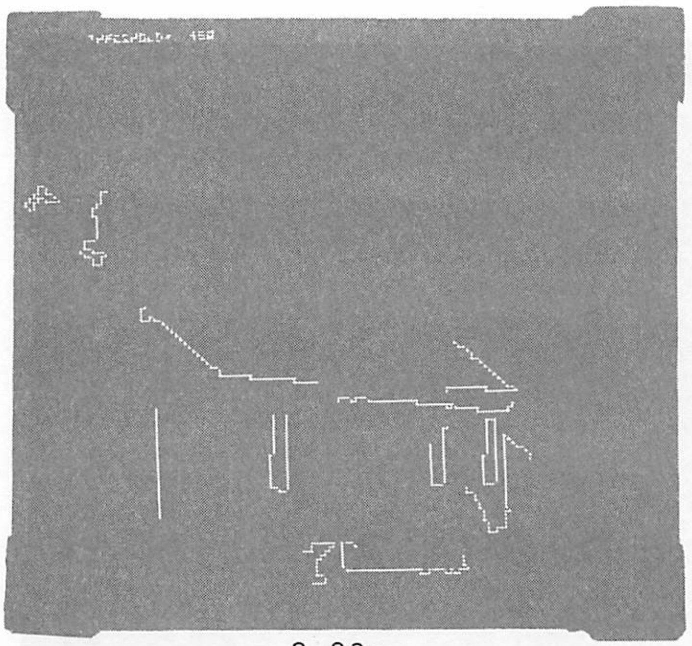

2.22c

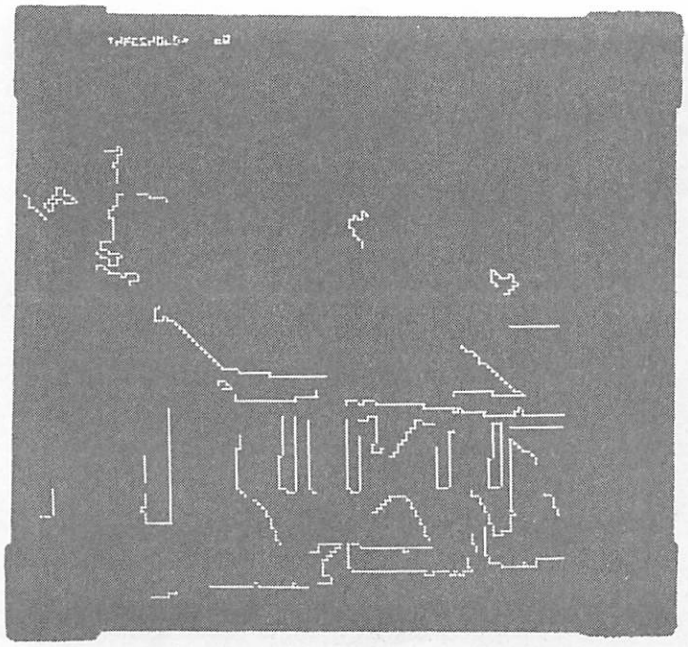
Figure 2.22. Edge images which are used to compare the two confidence measures described in the text. In figures 2.23 - 2.25 these images are shown thresholded at different levels. Due to the different scales produced by the two measures, an exact comparison is impossible, but sequences of comparable threshold values were chosen for the two algorithms.



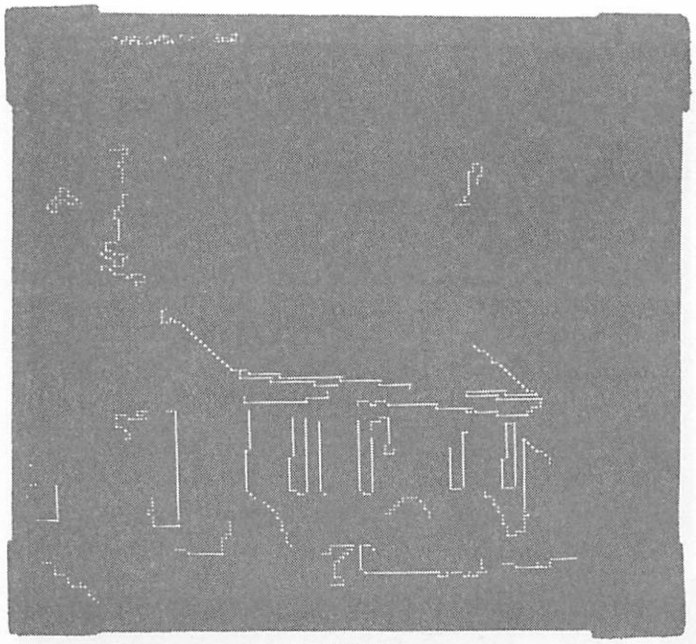
2.23a



2.23e

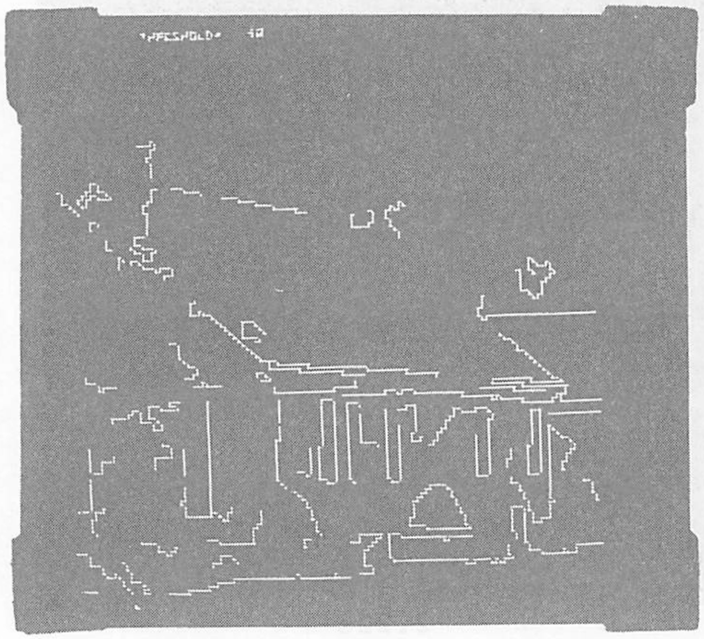


2.23b

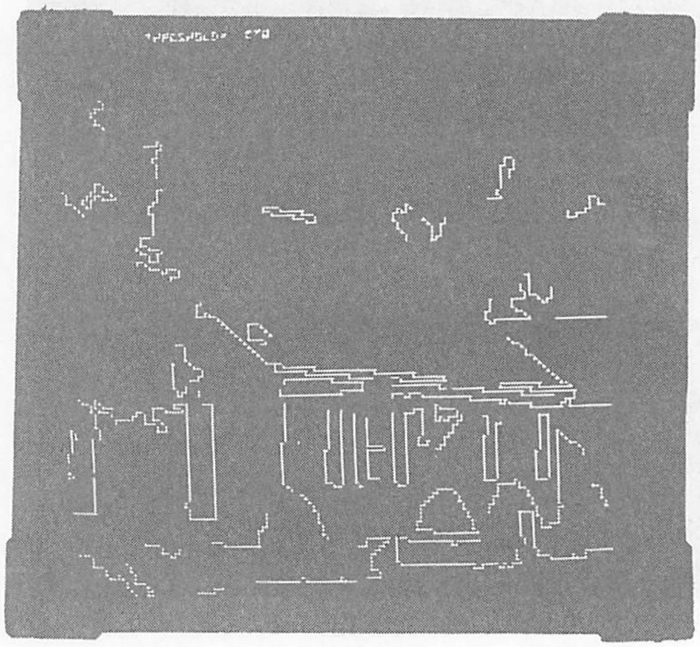


2.23f

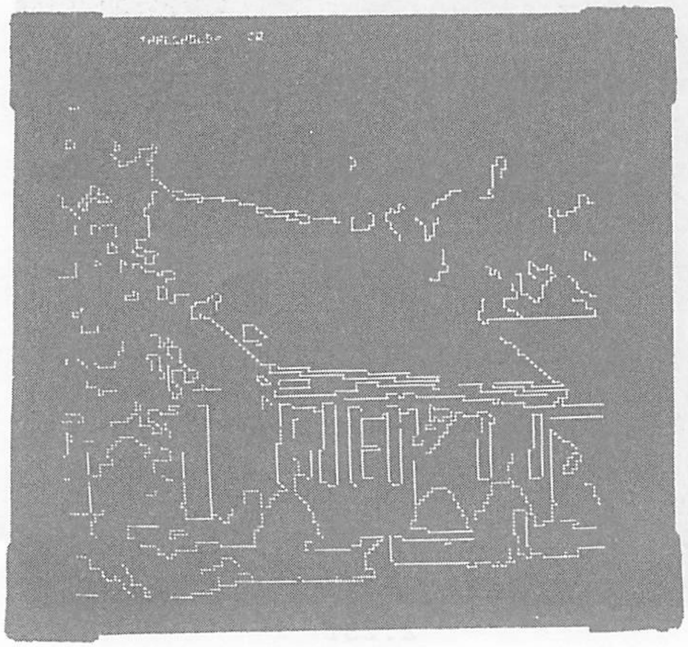
Figure 2.23. Successive thresholdings of the edge image in Figure 2.22a. Figures 2.23a - 2.23d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 2.25e - 2.25h show the same with heuristic confidence measures.



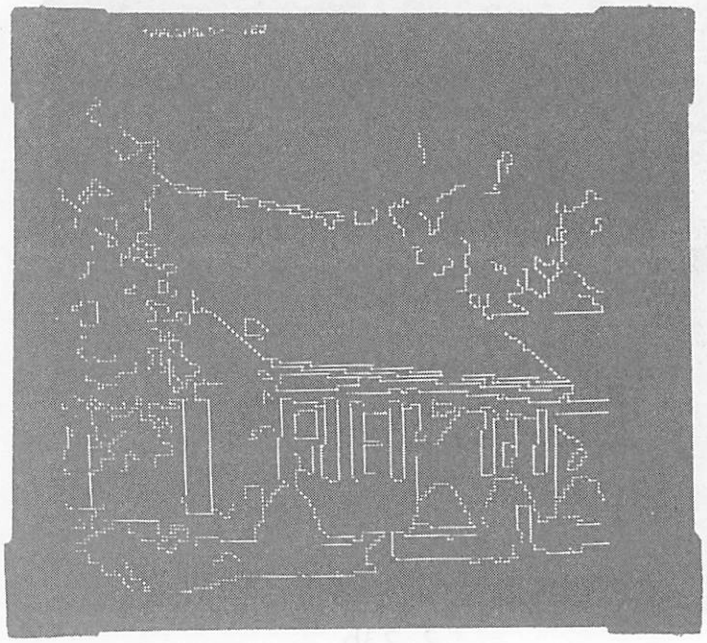
2.23c



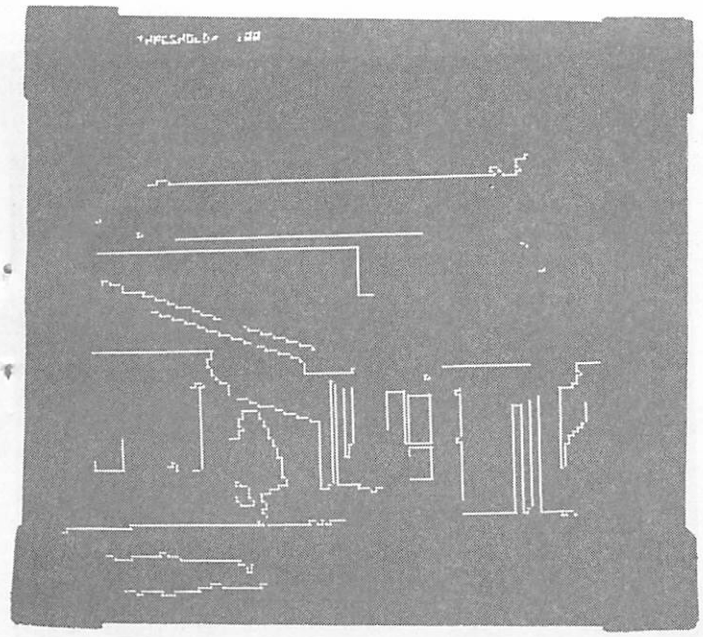
2.23g



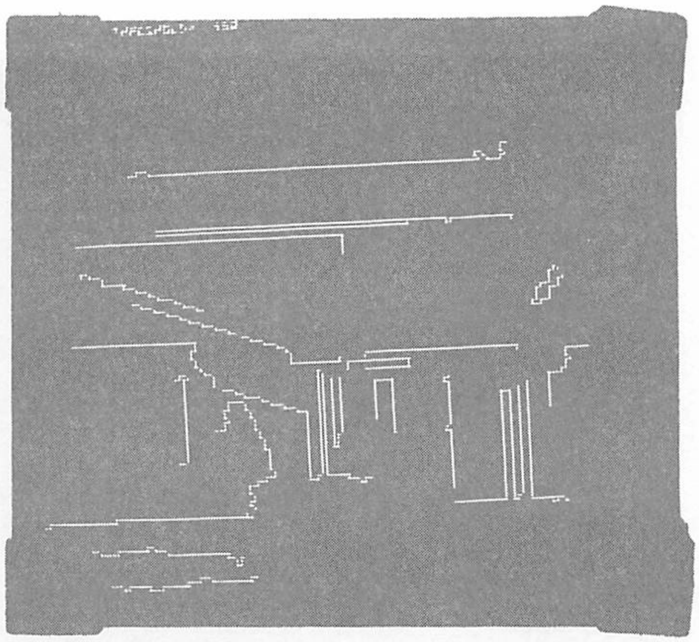
2.23d



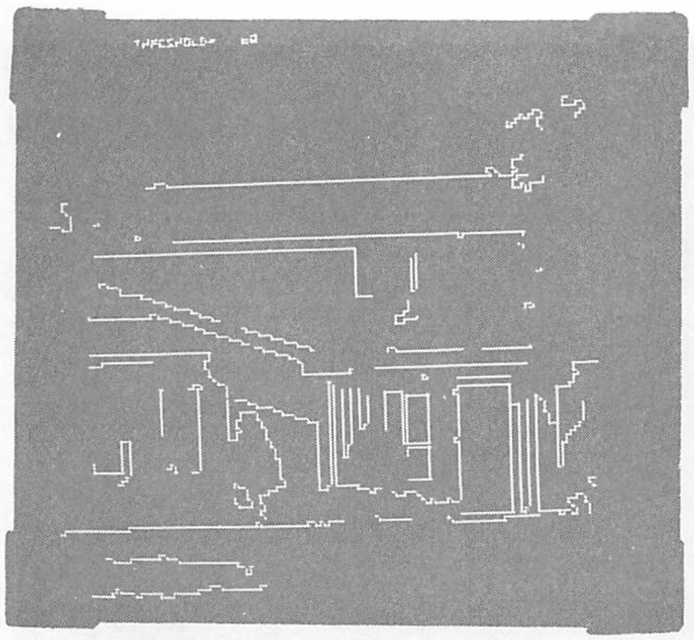
2.23h



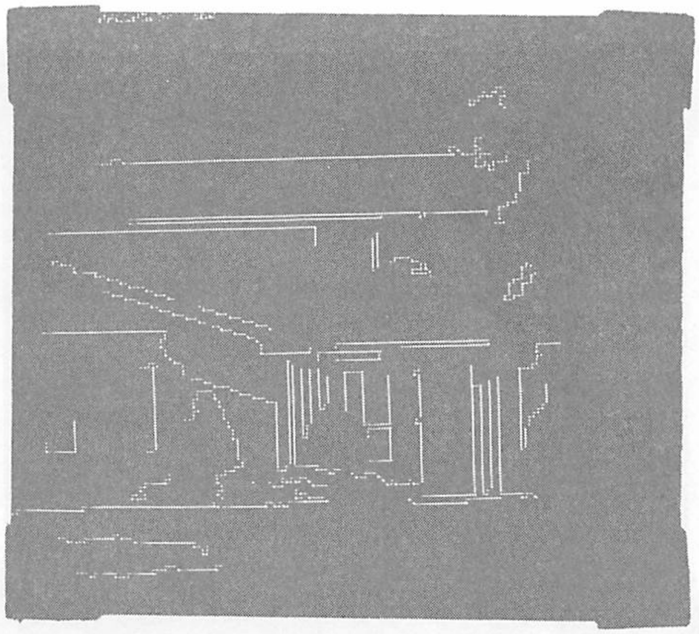
2.24a



2.24e

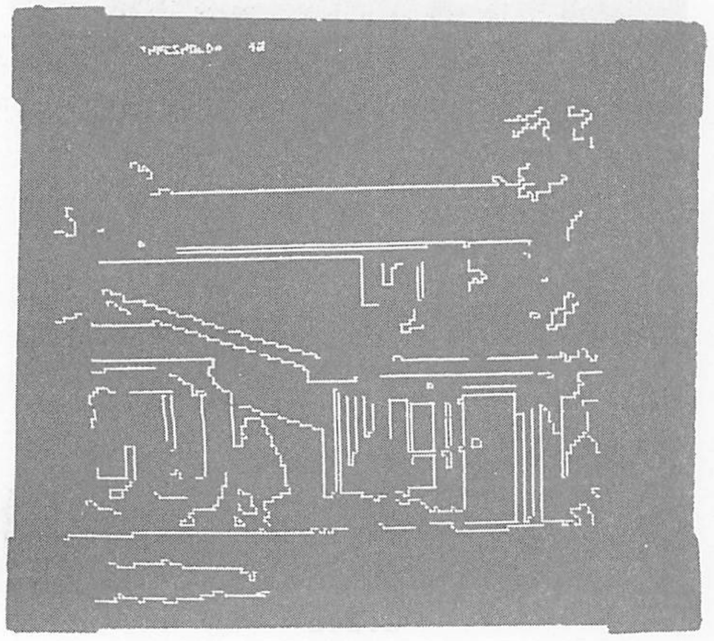


2.24b

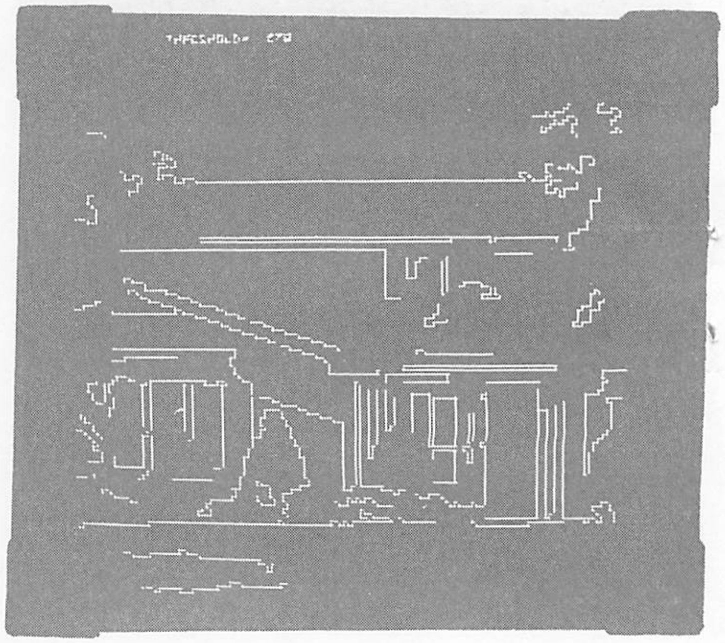


2.24f

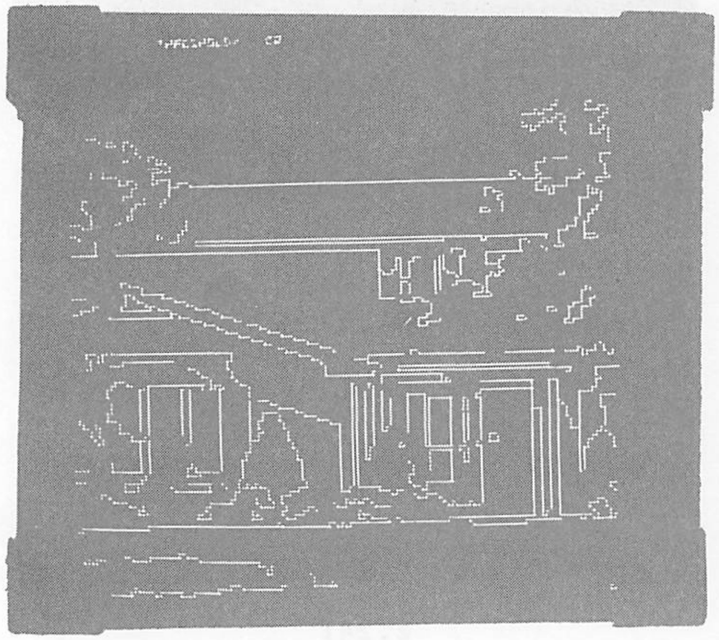
Figure 2.24. Successive thresholdings of the edge image in Figure 2.22b. Figures 2.24a - 2.24d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 2.24e - 2.24h show the same with heuristic confidence measures.



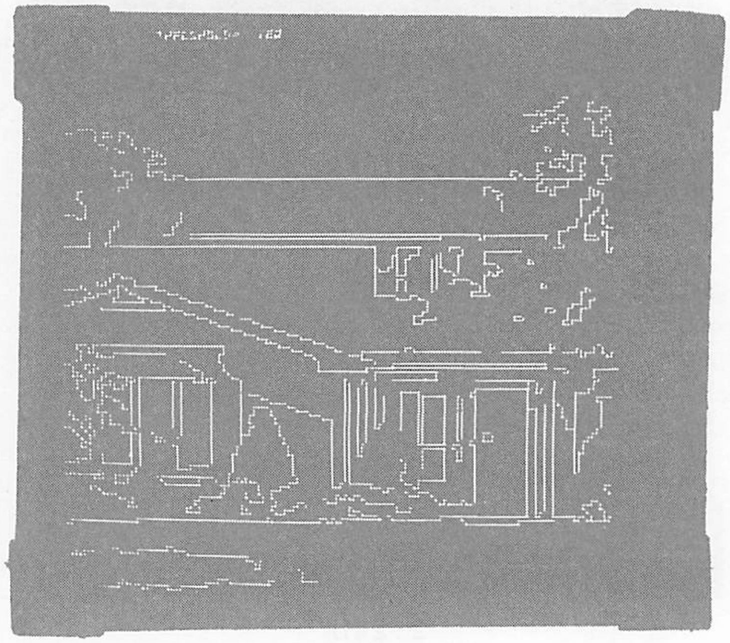
2.24c



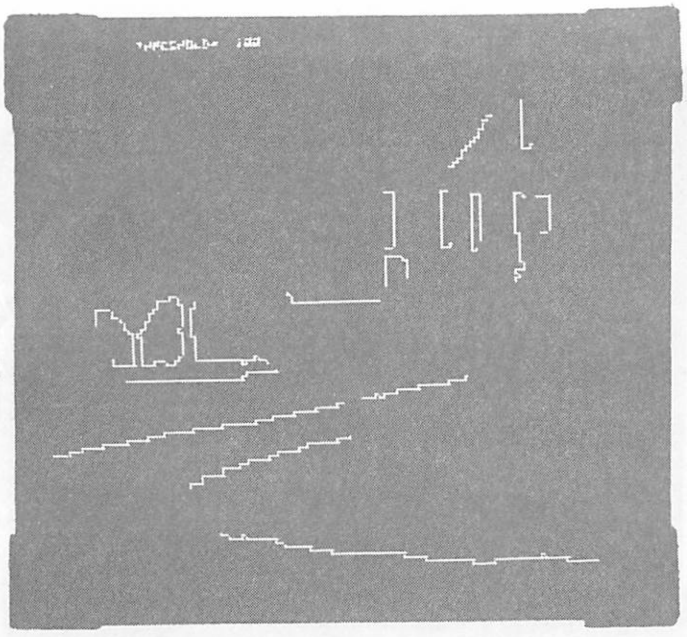
2.24g

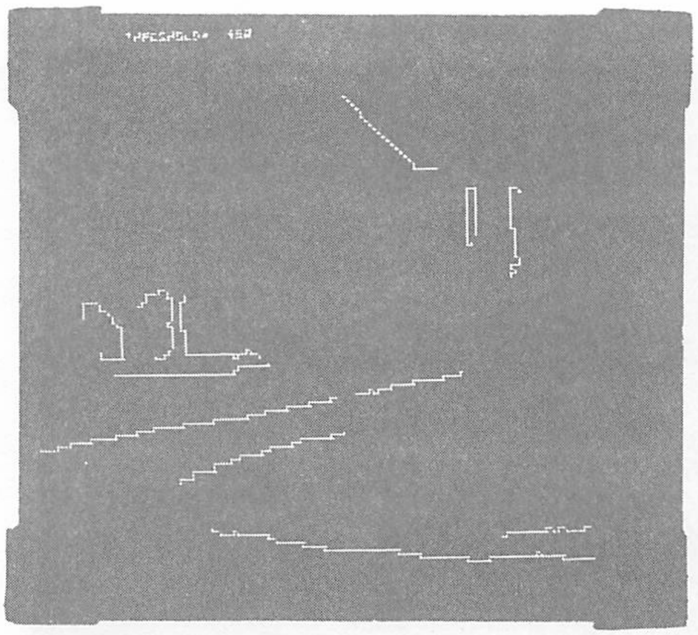


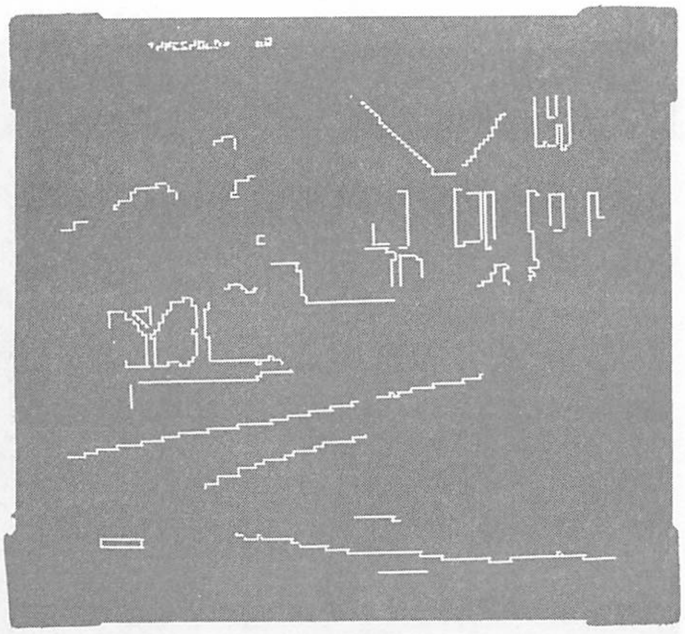
2.24d



2.24h


2.25a


2.25e


2.25b

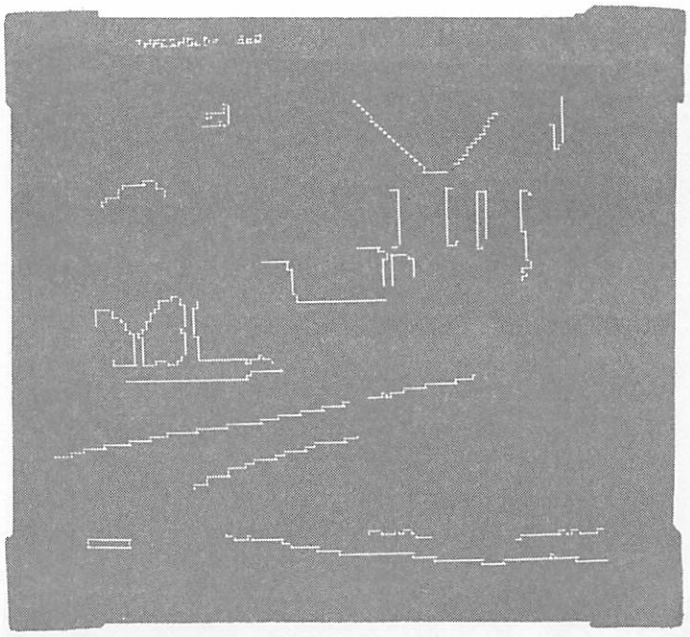
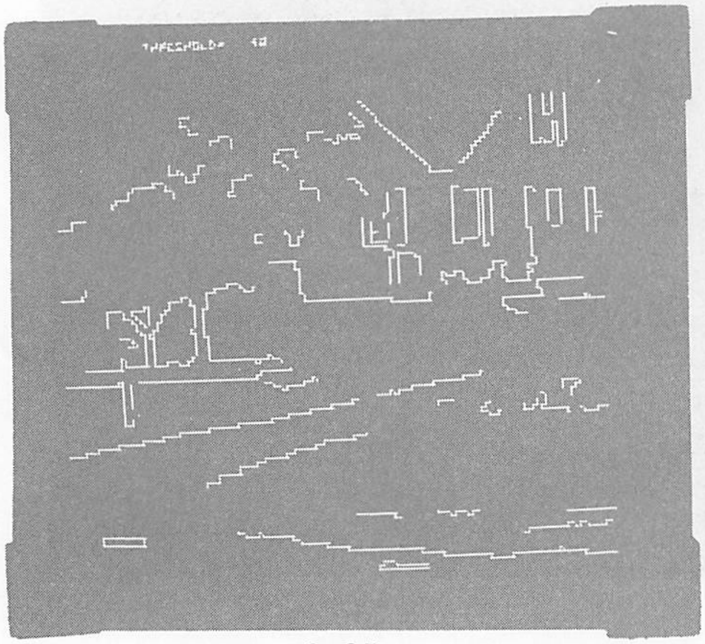
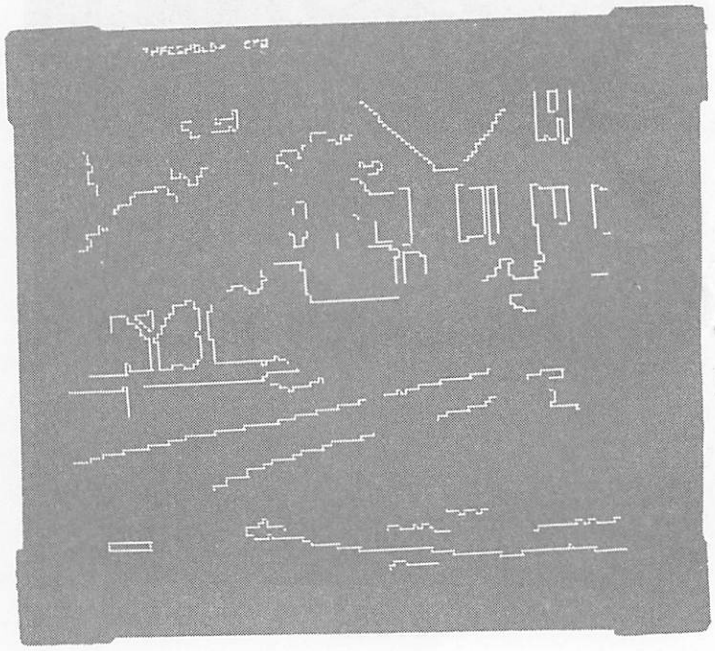
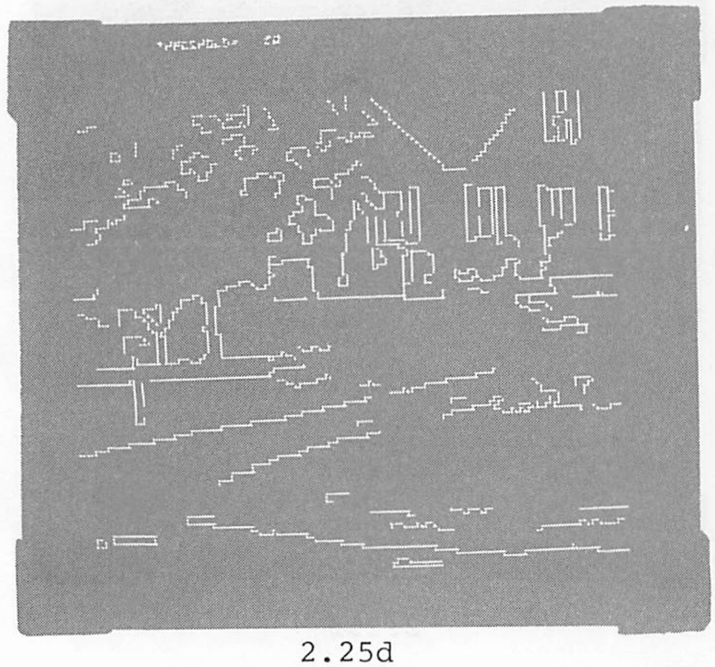
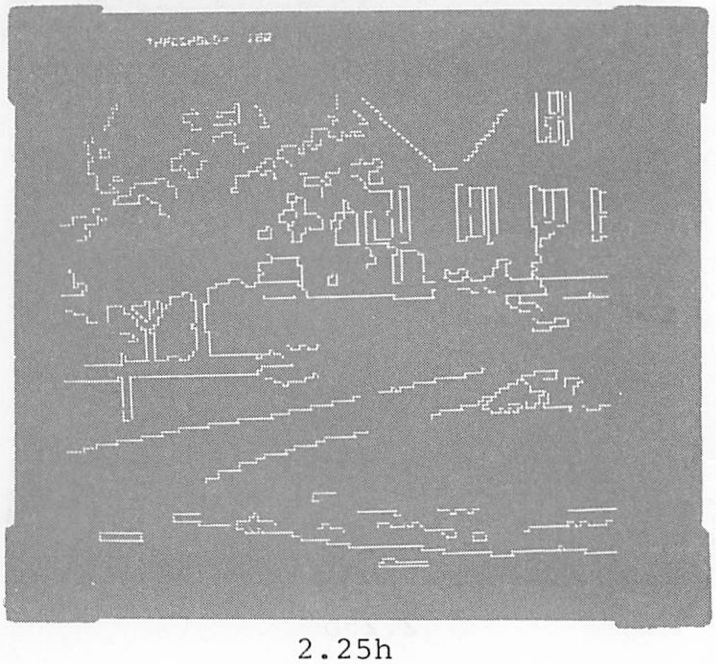

2.25f

Figure 2.25. Successive thresholdings of the edge image in Figure 2.22c. Figures 2.25a - 2.25d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 2.25e - 2.25h show the same with heuristic confidence measures.


2.25c


2.25g


2.25d


2.25h

tively, texture regions may be clearly defined with the aid of a region-growing process. (See Hanson and Riseman 1974 and also Section 5.5 below for a discussion.)

(2) Blurred edges. Many of these will have been corrected (or reduced) by the "UNMIX" process. Some of those that were introduced through noise or some other means and were not corrected will give rise to two adjacent parallel edges; one of the pair will be eliminated through SUPPRESSION.

(3) Gradients. This problem is a more general version of (2), where the change in intensity occurs over several pixels. One procedure for detecting gradients is to use a hierarchy of increasing-sized masks (Riseman and Arbib 1977; Hanson and Riseman 1974; Marr 1975; Rosenfeld and Thurston 1971). While this procedure can be shown to work in simple cases, it is difficult to make it work in general. Many masks of different sizes at varying distances from a boundary can detect that boundary, and it is difficult to organize them consistently. In addition genuine gradients are indistinguishable in a digitized image from three or more parallel one-pixel-wide regions with intensity monotonically varying across them. For an example refer to Figure 2.5. Since there is no way that this distinction can be made without using very high-level knowledge, the system will treat all such cases as gradients. We will accept the fact that such one-pixel-wide regions will be lost.

The Differentiation Operator

The standard technique for differentiation is to convolve edge

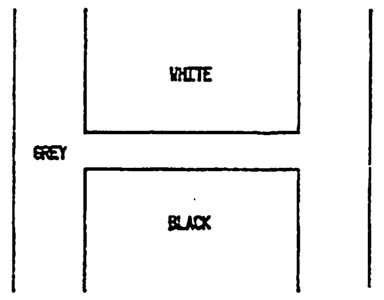


Figure 2.5. An example of a 1-pixel-wide region that is intermediate in intensity between that of its neighbours. The grey strip is a real region, and not a gradient.

masks with the image. It can be generalized to apply a set of masks, and to compute the output as some function of the results of these masks, often the maximum response.

For sharply defined boundaries, the simplest mask possible is all that is necessary (see Figure 2.6a). We will call this a 1 x 2 mask. In this and subsequent diagrams of masks, a heavy line indicates the edge position to which a mask's output is associated. On long straight boundaries a better response might be achieved using a 3 x 2 mask (see Figure 2.6b), since the information from three 1 x 2 masks in a line is used to average out the presence of occasional noise points. However, the relaxation processes that follow should be able to fill in such an edge, and we believe it will render the benefits of the 3 x 2 mask unnecessary; its limitations are described below.

Diagonals and corners can be detected by using diagonal masks (Figure 2.6c). This mask might be used in the computation of the edge strength of a vertical section of a diagonal. Note that application of this mask alone would give a positive response when applied to a horizontal edge (see Figure 2.7a and 2.7b).

Therefore, the difference between its output and that of its mirror image should be used. This will give strong response to diagonals, but not to horizontal or vertical edges. Other masks may be used to detect gradients; for example, the mask depicted in Figures 2.6d and 2.6f can be generalized into a hierarchy of mask sizes (Hanson and Riseman 1974; Marr 1975).

The more varied the collection of masks, the more guarantee there

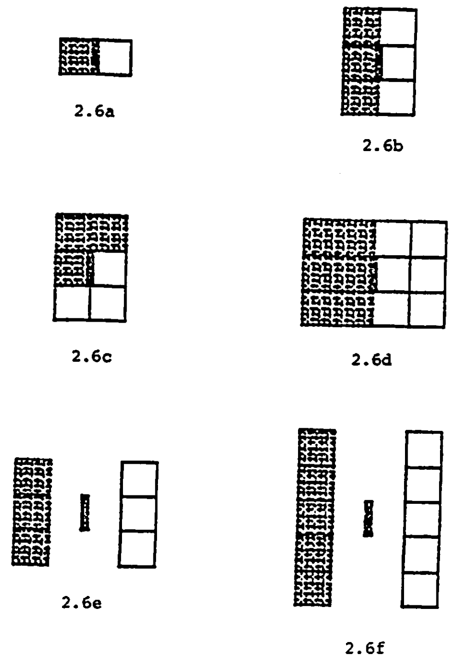
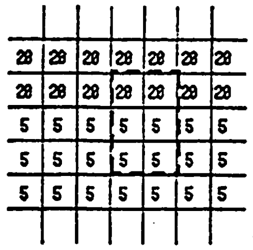
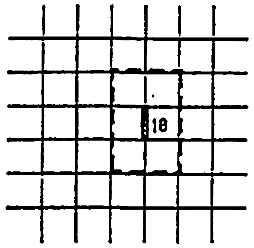


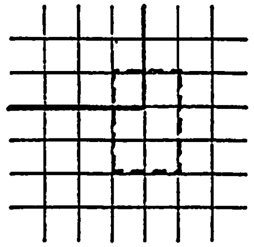
Figure 2.6. Typical masks. 2.6a 1 x 2, 2.6b: 3 x 2 straight, 2.6c: 3 x 2 diagonal, 2.6d: 3 x 4, 2.6e: expanded 3 x 2, 2.6f: expanded 5 x 2.



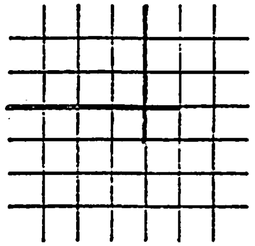
2.7a



2.7b



2.7c



2.7d

Figure 2.7. 2.7a shows a 3 x 2 diagonal mask on a horizontal boundary between two regions. 2.7b shows the output of this mask (if used above). The response is seen to be (20 + 20 + 5 - (5 + 5 + 5))/3 = 10, which is significant, since it happens to be 2/3 of the difference between the regions. 2.7c shows a 3 x 2 straight mask overlapping the corner of a dark region. 2.7d shows the spur produced by this mask and a similar horizontal mask in a neighbouring position.

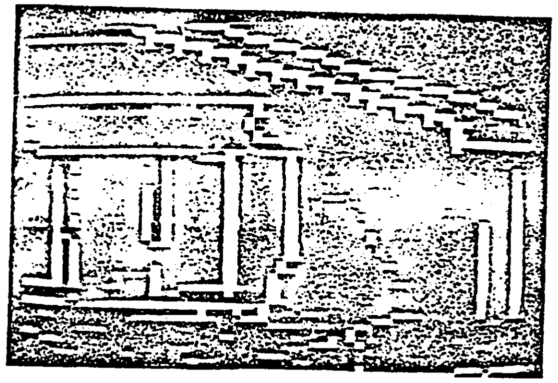
is of detecting the edge. However, using large masks has unfortunate consequences in positions where no edge is desired. Figure 2.7c shows a 3 x 2 straight mask superimposed upon a corner of a region. In this position there will be a response, albeit weak, giving rise to a "spur" in the differentiated version. A horizontal mask will cause the same problem, giving a result as shown in Figure 2.7d.

Larger masks will give more and longer spurs, which cause serious problems. During relaxation, it is possible for them to grow lines where none ought to exist; if they are close to other spurs formed similarly, they can get linked together during grouping processes. The result of the whole process begins to get quite ill-defined.

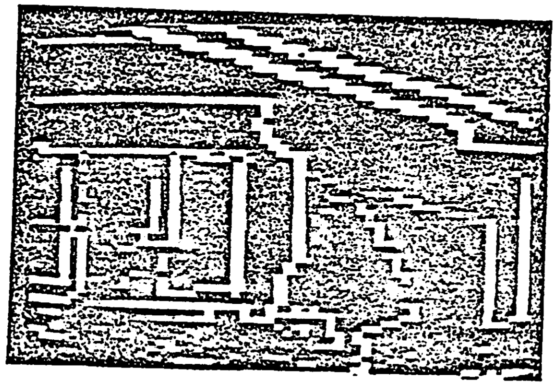
A comparison of these different combinations of some of these masks is presented in Figure 2.8. It can be seen that the 1 x 2 gives fairly good results; the absence of spurs is quite noticeable in contrast with some of the other masks. For this reason, we decided to use a simple 1 x 2 mask as our differentiation operator.

2.2.2 SUPPRESSION (Step 2b)

The weakness of a 1 x 2 mask is that it will be prone to missing boundaries of wider gradients. However, most of the boundaries in the several scenes examined in this paper were detected. Of course, the problem of gradients still must be dealt with since the system will be blind to edges, such as wide shadows on a cylindrical surface. Ideally, the total strength of the wide gradient edge ought to be collected (Riseman and Arbib 1977), which is the goal of employing masks of in-



2.8a



2.8b

Figure 2.8. Effects of using large masks. Figure 2.8a shows a portion of the house scene of Figure 2.3b differentiated using a 3 x 2 mask. The spurs that are so prominent here are not produced by a 1 x 2 mask, as shown in Figure 2.8b.

creasing size. Rather than deal with some of the problems discussed in the last section, here instead we seek means to suppress multiple parallel indications of edges.

While the UNMIX procedure will eliminate some narrow gradients, others will inevitably remain and give rise to parallel multiple indications of the same edge. These can be removed by what is known as multiple edge suppression (Riseman and Arbib 1977). Consider the image in Figure 2.9a representing brightness, and its derivative in 2.9b representing the strength of the gradient. The suppression technique works as follows: consider three pixels in a vertical line as in Figure 2.10. Let p, q, r be the horizontal gradient at the lower boundary of these three cells. If it happens that either

$$|q| \le |r|$$

or

$$|q| < |p|$$

with q the same sign as r or p respectively, then q is set to zero. A similar operation is applied to vertical gradients in a horizontal row of cells. Hence, in Figure 2.9b the row of 10's will be set to zero, resulting in Figure 2.9c. This type of suppression is restricted to the cases where the pair of edges have the same gradient sign. Therefore, in Figure 2.10 the suppression process does not remove either of the boundaries of the one-pixel-wide region.

An improved version of the SUPPRESS procedure requires an increase in the values of the local maxima of gradients by the sum of those values that were suppressed in a direction perpendicular to the gradient.

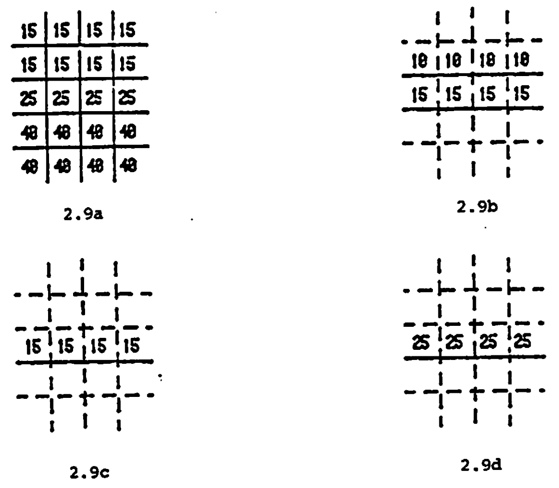


Figure 2.9. 2.9a shows typical intensity values. 2.9b shows 2.9a after differentiation. In 2.9c the top line in 2.9b has been suppressed. In 2.9d the top line in 2.9b has been suppressed and its strength added in to the bottom line.

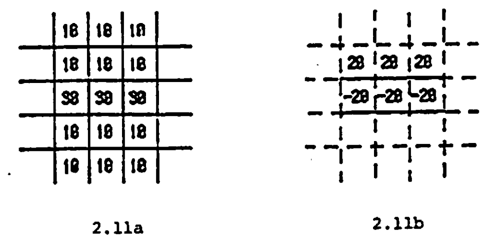


Figure 2.11. 2.11a: A light strip on a dark backgound. 2.11b: The resulting edges and their strengths. Since they are of differing signs, no suppression takes place.

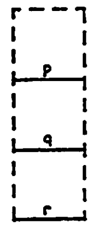


Figure 2.10. Three horizontal edges in a vertical column (see text).

Thus in Figure 2.9b, the 10's will be set to zero, and the 15's will be set to 25 as in 2.9d. This more accurately reflects the strength of the boundary since it is between regions of intensity 15 and 40.

2.2.3 RELAXATION (Step 2c)

The edge strengths produced by the differentiation process depend upon the local contrast in the image. Weak edges may arise from low-contrast boundaries, gradients extending over many pixels, or from texture internal to a region, or indeed from noise. The output of the differentiation process is thus usually far from being clean. If the strengths of edges are viewed as probabilities, or confidences, of the existence of edges, usually few of them would be considered to have probabilities of 0 or 1. An edge probability that is neither 0 nor 1 effectively is an ambiguous interpretation of the entity concerned. However, the local context around each edge contains information for updating the probability so that ultimately the ambiguity will be reduced and interpretations will be locally consistent. A _relaxation_ process may be used to update these confidences in parallel. We will firstly discuss what is meant by relaxation.

Let $X = \{x_i\}$ be a set of units corresponding to features of a system M, and let $S = \{s_i\}$ be a (possibly infinite) set of states that the x_i might individually assume. Let $f: X \to S$ be an assignment of states to the x_i. Now, if M is a physical system, then the laws of physics will determine whether f is globally consistent, that is whether the assignment corresponds to some physical reality. However, the set of

such mappings S^X may be so large as to defy directly testing whether a fiven f belongs to the subset of 'realistic' mappings.

The nature of M may result in certain relationships amongst subsets of X. These relationships too will be subject to real-world constraints. Thus if x_i and x_j obey a generalized adjacency relation (i.e. they interact directly) then although individually they may assume any values of S, as a pair they are restricted to a certain subset of S^2. Testing the validity of an assignment of states to just two (or some other small number) of the units may be computationally feasible, whereas the testing of global consistency is not.

In order to take advantage of these local conditions, techniques known as relaxation processes have been defined. In such a process, the assignment of states to units is performed so that all local constraints are satisfied. This is generally done in an iterative manner, where in some sense each unit 'closes in' on its final state. A point of faith in all applications of relaxation is that local consistency everywhere will guarantee global consistency. In those cases where the local constraints are well-defined and accurately reflect all possible physical situations, this tends to be true, but in more complicated cases, such as scene analysis, global consistency is by no means guaranteed. Indeed, there may even be no guarantee of convergence.

We will look now at three different paradigms, all of which fall under the general description of relaxation given above. They differ in the main by the kind of states that may be assumed in the units x_i in the problem, and in the way these states are changed from one iteration to the next.

The first problem we will discuss is that of the numerical solution to simultaneous equations with large numbers of variables. In fact, it was from an early solution to such problems that the method got its name (Southwell 1935). Here the x_i are the variables, and s_i the real numbers. The basic operation (ignoring modifications such as over-relaxation) is as follows. 'Reasonable' initial guesses are made for the x_i. One variable, say x_1 is adjusted so that the first equation holds exactly. A second variable, say x_2, is then adjusted so that the second equation holds exactly, using the new value for x_1. This process is continued cyclically until all the discrepancies or 'residuals' are within certain tolerances. The individual equations correspond to the 'local' constraints. However, since all variables may appear in any equation, the constraints are in some sense global. It is for this reason that, if the equations are neither under-determined nor ill-conditioned, this kind of relaxation usually does find the unique solution. In Chapter V we examine a variation of this application where (1) only a small subset of all variables take part in each equation, so they may truly be said to be local, and (2) the updating is performed at all units in parallel.

The second method to be discussed here is not usually known as relaxation, but as it conforms to the general description given above, it will be included here. In this instance, S is the power set of some set T, say, of labels. The goal is to associate a unique label of T with each x_1. Initially, however the association is made with a subset of T.

As the process proceeds, candidates are discarded according to the local constraints. The method is thus simply an iterative constraint-satisfaction process. Two examples of its use will be mentioned. In their Interpretation-Guided Segmentation (IGS) system, Tenenbaum and Barrow (1976) initially associated an entire set T of object-descriptors with each pixel in an image. By imposing certain spatial adjacency constraints they were able in most cases to eliminate all but one label for each pixel. Waltz (1972) used a similar approach for determining the 3-D interpretation of a collection of polyhedra by examining constraints on edge- and vertex-types.

The third type of relaxation to be covered here is that of Rosenfeld et al. (1976). In their scheme, the states are n-tuples of real numbers $(p_1,\ldots,p_n)$, where n is usually small, say in the range 2 to 10. There will be an associates set Λ of n 'target-states' or labels λ_i, where p_i represents the probability or confidence that λ_i is the correct state of the unit concerned. The p_i are therefore usually in the range 0 - 1, and in each n-tuple they sum to 1. The updating takes place on the basis of compatibility of labels. This compatibility is expressed in the form of pre-defined 'compatibility coefficients' between labels on interacting, i.e. adjacent units. This kind of relaxation is useful in edge-detection techniques for scene analysis, since at each point in the image it may be assumed that either no line, or a line at one of a small set of orientations may exist. The local constraints manifest themselves in terms of local continuity of line-segments (and in some applications continuity of derivative). We will now look at this application of relaxation to scene analysis in more detail.

The set of labels Λ can be a set of edge-descriptors, such as "horizontal edge", "vertical edge", etc., and will typically include a special label, the "null edge" label, which is an assertion that there is no edge at that point. Λ is chosen so that the labels are mutually exclusive, since it is desired that ultimately one label will be present with probability one, the others with probability zero at each point. The labels may be regarded as competing at each position in the image during relaxation. In this process, each probability for every label at every position is then updated in parallel according to its compatibility with the labels at neighbouring positions (in some predefined neighbourhood). Under quite restricted conditions convergence can be guaranteed (Zucker et al. 1976), although not necessarily to any meaningful global interpretation.

A consideration of the relaxation model described in detail in Rosenfeld et al. (1976) (and summarized above) leads one to conclude that the heart of the scheme is in the setting of the (many) compatibility coefficients. Not only are there many of these which need to be set, but due to heavy interdependence of effects there is no direct correlation between the setting of an individual weight and the performance of the system. Thus, tuning can be very difficult, since it requires optimization of many variables simultaneously. Furthermore, there is no guarantee that it is possible to set weights such that all the desired effects can be achieved simultaneously. While it is fairly straight-forward to set the weights so that some of the more obvious

cases are taken care of, there is rarely enough leeway to adjust them so that the more "awkward" cases, such as when part of an edge's neighbourhood supports it and the other part inhibits it (case 0-1 described below), are managed correctly. Indeed, it is difficult to determine where the system is failing, or how it is achieving its results.

It appears that one source of these difficulties arises when the updating process employs a single formula that is used to take care of the various very different cases that arise. In the next section, an alternative scheme is proposed which will deal with each of the aforementioned problems separately, in a clearly structured manner.

A Different Representation for Relaxation

In the scheme just described, several labels are competing for each position in the image. Thus for a point on a diagonal boundary, both horizontal and vertical labels will be competing. In our representation, we can allow both labels to coexist at a pixel since we are placing edges at interpixel boundaries, not on top of the pixel. Therefore, at each vertical pixel boundary the only labels we need to consider are "vertical edge" and "no edge", and similarly for horizontal edges. In this way, the set of two probabilities at each edge location $\{P_i(\lambda) | \lambda \in \Lambda\}$ can reduce to a single parameter P_i. The probability of an edge at position i is P_i, while the probability of the null label "no edge" at position i is $1 - P_i$. Relaxation is very much simplified as a result of this representation.

We will use the notation of Figure 2.12 to describe the edge-con-

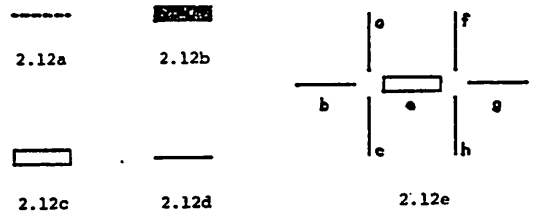


Figure 2.12. Notation. 2.12a: Edge position with no edge. 2.12b: Edge position with edge. 2.12c: Edge to be updated. 2.12d: Edge of unknown strength. 2.12e: Configuration of edges around central edge e.

figurations under consideration.

An open rectangle will represent the edge to be updated,

a dotted line will represent an edge position with no edge present,

a thick solid line the presence of an actual edge, and

a thin solid line an edge of undermined strength.

Now let us describe the algorithm. Associated with each edge-position will be a value indicating the probability or confidence that an edge exists at that position. Every edge-position has two end-points at which that edge could continue, and every end-point will be classified as one of four "vertex-classes" according to the strengths of the incident edges. The vertex classes of the end-points of the edge-position under examination will then determine how the edge-strength is to be updated.

Cases for Updating Edges

An iterative procedure for updating the probabilities is described below. We will denote the configuration of continuing edges at an end-point by an integer n in the range 0 to 3, representing the number of such edges in the pattern. Recall the discussion of Section 2.2 in which we saw that the local orientation of an edge was a poor determinant of the edge's global orientation, and that edges of all orientations consist of sequences of horizontal and vertical segments. Failing any additional information, we will suppose that parallel and perpendicular continuations of an edge are equally likely. A configuration of n edges at a vertex of edge e will be considered equivalent no matter which of the three possible edge positions to that side of e that they take. These four equivalence classes of continuing edge patterns are depicted in Figure 2.13.

Now it will be remembered that few edges are present or absent with any certainty. Therefore the usual case is that each equivalence class has a probability of being true which is a function of the probabilities of the individual edges. The determination of which classes, or <u>vertex types</u> are present, computed as a function of the probabilities of the three edges to either side is now discussed.

Computation of Neighbourhood Patterns

We would like to classify the configuration of edges to each side of e as one of the four vertex types of Figure 2.12a-d. Consider one end-point, say the left one, in Figure 2.12e. We will assume that the numerical values associated with edges are in the range 0 - 1, representing probabilities of the presence of an edge.

Since we are treating perpendicular continuation as equivalent to straight-line continuation (i.e. a and c have exactly the same effect on e as does b), we can assume without loss of generality that $a \geq b \geq c$.

Assuming independence of the edges (unfortunately, often a bad assumption), a probability-theory based argument would give for vertex types 0 - 3:

$$\text{Conf(type 0)} = (1 - a)(1 - b)(1 - c)$$

$$\text{Conf(type 1)} = a(1 - b)(1 - c)$$

$$\text{Conf(type 2)} = ab(1 - c)$$

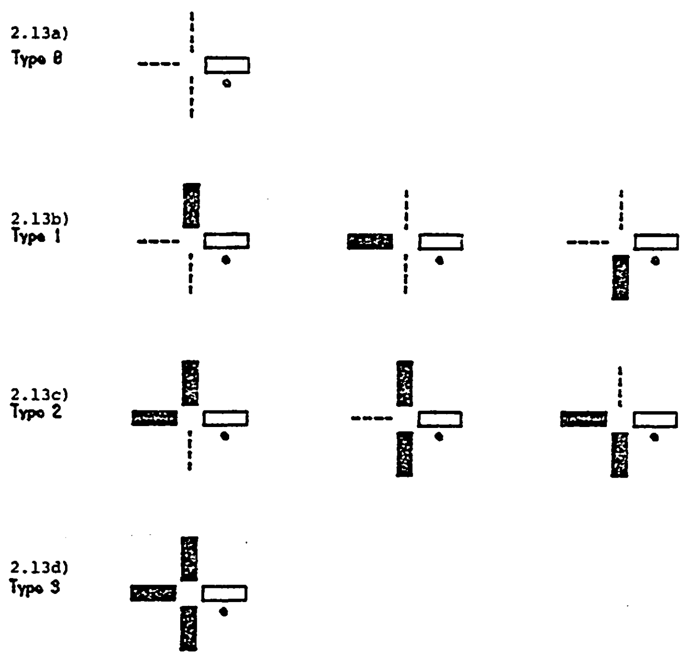


Figure 2.13. Classification of vertex-type of left-hand endpoint of edge e.

Conf(type 3) = abc.

The case with the highest confidence is then chosen as being the "state" of the left side of edge e.

However, there are cases where, for example, b and c are very low and a is considerably larger than them, but perhaps not close to 1. In such situations we would like a strong indication of a type 1 vertex (see Figure 2.14a). The remedy would be as follows: instead of subtracting a, b, and c from 1 to form the no-edge confidences, we can subtract from m, where m = max(a, b, c) = a in this case. m thus represents very locally the confidence-level of a high-confidence edge. Thus we have

$$\mathrm{Conf}(0) = (m - a)(m - b)(m - c)$$

$$\mathrm{Conf}(1) = a(m - b)(m - c)$$

$$\mathrm{Conf}(2) = ab(m - c)$$

$$\mathrm{Conf}(3) = abc.$$

There is one difficulty with this formulation; that is that it forces Conf(0) to be zero at all times. It could occur that a is much larger than b or c, but also be very close to zero (see Figure 2.14b); our formula would calculate a larger value for Conf(1) than Conf(0) when type 0 should actually be selected. This can be easily fixed by anchoring m to some minimum value q. We need a lower bound for m because there is always a chance that a stronger edge could be present. This will guarantee type 0 to be the most probable vertex type when all incident edges have very low strengths. Thus the final definition of m is

$$m = \max(a, b, c, q) \text{ and}$$

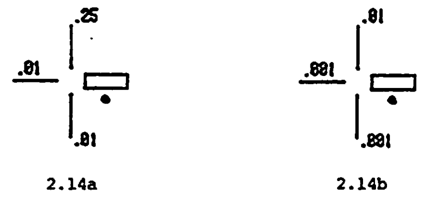


Figure 2.14. Two configurations depicting low-probability neighbours of e. Vertex-type 1 is indicated in 2.14a, and vertex type 0 in 2.14b.

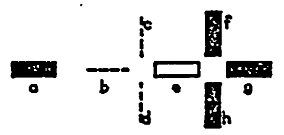


Figure 2.16. Edge e is classified as being type 0-3. If its strength is high, it is likely that edge a will join up with it. The desirability of this effect is not so clear if e is weak.

$$\mathrm{Conf}(0) = (m - a)(m - b)(m - c)$$
$$\mathrm{Conf}(1) = a(m - b)(m - c)$$
$$\mathrm{Conf}(2) = ab(m - c)$$
$$\mathrm{Conf}(3) = abc.$$

Since we will select vertex type i, where $\mathrm{Conf}(i) = \max_j[\mathrm{Conf}(j)]$, we only need to know the relative sizes of the Conf(i), so we do not need to normalize these probabilities. q can be calculated as a function of edges in some neighbourhood of e in the image. A suitable such function might be $\mu - \sigma$, where μ is the average edge strength and σ its standard deviation. We found that a constant value for q of about .1 performed very well over several images.

Calculation of Direction of Update

The following notation is used to depict the neighbourhood characteristics (or state) of an edge: the symbols i - j denote that configuration i is at one vertex of central edge e, and j is at the other. Obviously, i - j = j - i, so we need only consider the ten cases of i - j where $i \le j$, shown in Figure 2.15. It will be remembered here that we are only interested in the possibility of the continuation of an edge, not the direction of such a continuation (cf. Figure 2.4).

Now in states 0-0, 0-2, and 0-3 one can quite confidently say that there is no good support for e, and in 1-1, 1-2, and 1-3 one can quite confidently say that there is. However, if e is in state 0-2, for example, it is conceivable that the situation is really as in Figure 2.16. In such a case, the current strength of e itself may be a determining

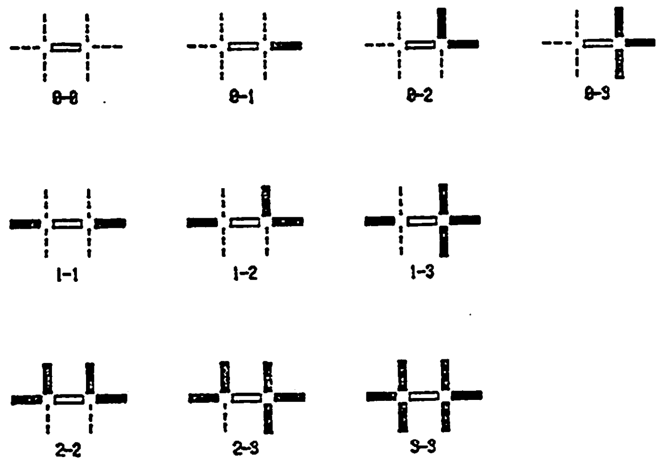


Figure 2.15. Representative combinations of vertex-types. This figure depicts all possible cases, subject to symmetry and the equivalences noted in the text.

factor for the case that b and e should be grown in to complete the line.

Two points may now be made. First, in some of the above cases it is clear how edge e should be updated. Therefore, the updating process should explicitly increment or decrement the edge. Secondly, as information may need to organize and propagate for some period of time over some distance in the image, updating increments (decrements) should not drive the probabilities to one (zero) too quickly. Rather, the increase (decrease) whould be some small amount on each iteration. In this manner the influence of regions which are initially locally consistent will spread into "less confident" regions, much as "islands of reliability" played a part in the HEARSAY speech analysis system (Lesser and Erman 1977).

So in cases 1-1, 1-2, and 1-3 we will let e increase (see Figure 2.17a); and in cases 0-0, 0-2, and 0-3 we will let e decrease (see Figure 2.17b). In all other cases the context is not very clear. Leaving aside case 0-1 for the moment, we see that in none of the cases 2-2, 2-3, 3-3 (see Figure 2.17c) is the presence or absence of e critical for the continuation of a neighbouring edge since they have alternative directions for continuation. It will not introduce or eliminate "cracks" - edges terminating at an indeterminate point. Whether e should exist or not depends largely upon its contrast strength, as well as continuity properties on either border, and little else, at least until more global views and higher level knowledge is available.

Case 0-1 is really the only problem. The neighbourhood on one side

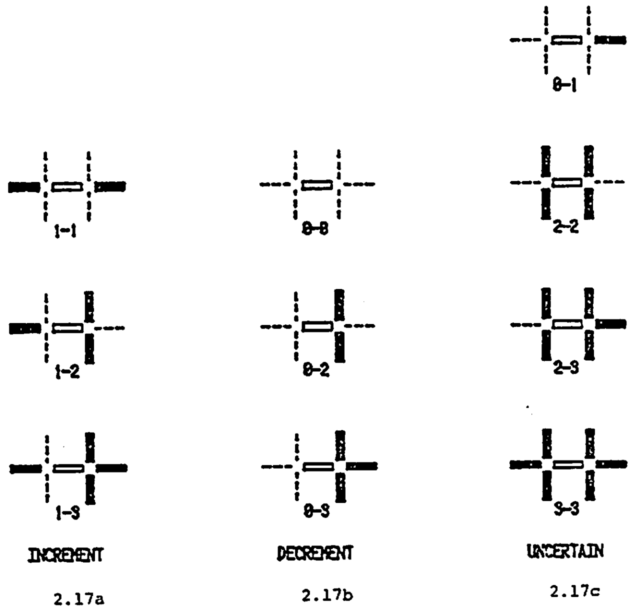


Figure 2.17. Cases for updating edge. In Figure 2.17a are depicted all those cases where the central edge should be incremented, and in 2.17b those where the edge should be decremented. Figure 2.17c shows the uncertain cases.

strongly supports e, yet the other suggests that e should be absent. As no sensible decision can be made, no action is taken here (or in cases 2-2, 2-3, and 3-3). This is a very important decision: it implies that in the updating process, the 0-1 case remaining constant will prevent a line from growing into noise or from being eaten away at its terminating point.

Performing the Update

The operation of the system in updating an edge may now be described. Let C_e^t be the confidence (in the range 0 - 1) of edge e at time t. The decision to increment, decrement or leave the edge-strength alone has been made on the basis of the vertex-types at either end-point of the edge. The updating formula depends on the situation as follows:

Increment case: $C_e^{t+1} = \mathrm{Min}(1, C_e^t + k)$

Decrement case: $C_e^{t+1} = \mathrm{Max}(0, C_e^t - k)$

Uncertain case: $C_e^{t+1} = C_e^t$

where k is a constant. That is, we add k, subtract k or do nothing to the current confidence, making sure the value stays within the range 0 - 1. Zucker and Mohammed (1978) argue against a fixed-increment updating procedure because, in general, the fixed-point of the relaxation algorithm might be skipped over. This is not a problem in our case.

A large value for k gives fast convergence, but does not permit information to propagate very far before the confidences of edges converge to 0 or 1. For small k the opposite is true. By experimentation on several images, values in the range .15 to .20 were found to be most

suitable. Typical results of using this relaxation process are given in Figure 2.18.

2.3 Stage 3 - Grouping

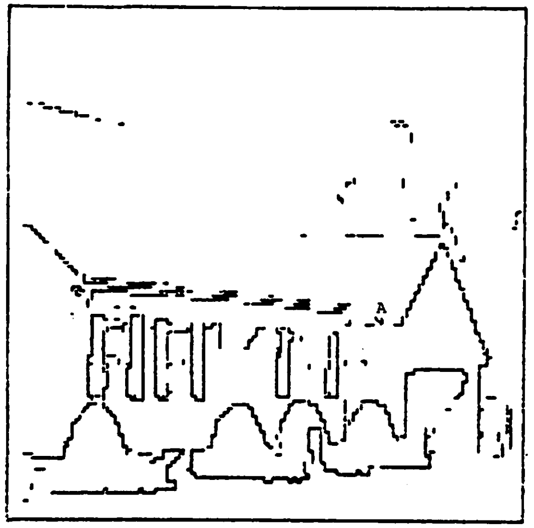
2.3.1 BIND (Step 3a)

The next stage is to decide which neighbouring edges link up to form extended line segments. It is clear that those points in the current representation which have 1, 3, or 4 edges entering them, i.e., vertices of degree 1, 3, or 4, are natural termination points for these line segments (see Figure 2.19). Breaking boundaries in these places will tend to form segments which lie between only two regions. This is a highly desirable effect, since there will then be less variation of properties (such as intensity) on either side of the segments. This was a major design consideration in the RSE representation of low-level output in the VISIONS system (Hanson and Riseman 1976).

The first stage of the binding process, then, is to mark all vertices which terminate segments as in the configurations in Figure 2.19. Following this computation, it is straightforward to track all segments between vertices and assign a unique label (line-number) to each boundary segment.

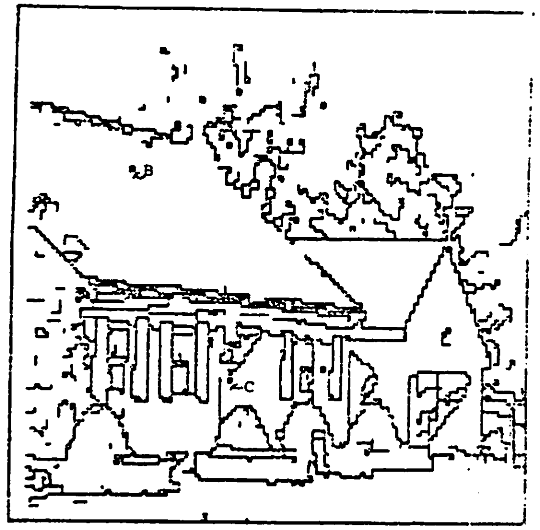
2.3.2 FEATURE EXTRACTION (Step 3b)

For each unique line-segment a set of properties can be established, some requiring recourse to the original intensity image, or at least, the intensity image that was differentiated. Typical properties to be



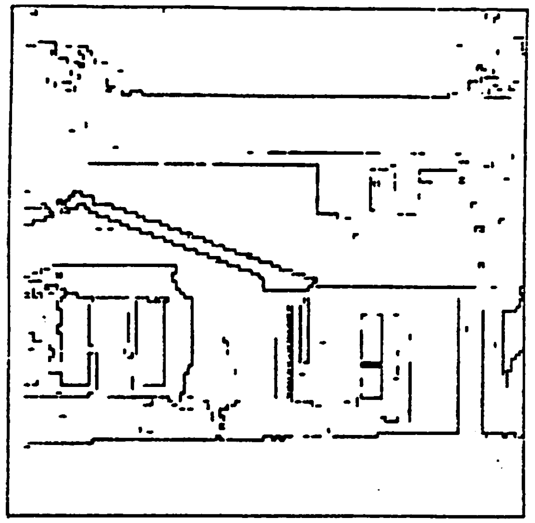
2.18a

Figure 2.18. Differentiation and relaxation. Figure 2.18a shows the data in Figure 2.3a differentiated. Edge strengths have been thresholded at .25 for display purposes only. This form of display was chosen to demonstrate that by way of thresholding prior to relaxation at a value chosen so that most wanted edges and few unwanted edges appear, most of the desired edges have gaps in them. It is relaxation which both fills in the gaps, and removes many unwanted lines. It will be remembered that in a threshold image, at any point with no visible edge there may be a sub-threshold edge present. The gap in the top of the garage roof (point A in Figure 2.18a) consists of such edges.



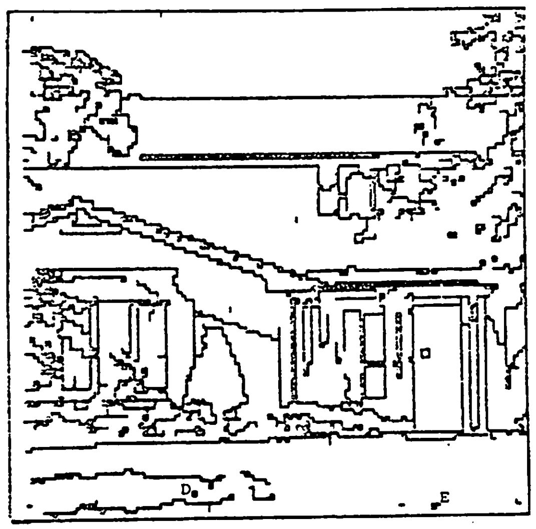
2.18b

Figure 2.18b. Results after 5 iterations of relaxation applied to Figure 2.18a. Small 'bubbles' such as points B and C in Figure 2.18b and points D and E in 2.18d were present in the pre-relaxation data (Figures 2.18a and 2.18c) but were sub-threshold. Due to the mutual reinforcement of the component edges, they came through as strong edges. They are removed in a later stage of processing.



2.18c

Figure 2.18c. Differentiated version of Figure 2.3b. Edge strengths have been thresholded at .25 for display purposes only.



2.18d

Figure 2.18d. Results after 5 iterations of relaxation applied to Figure 2.18c.

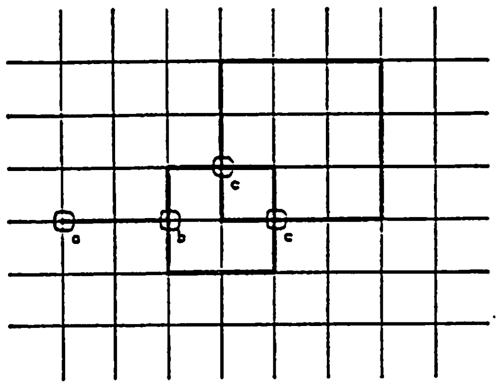


Figure 2.19. Three kinds of vertices. (a) Order-1. (b) Order-3. (c) Order-4.

associated with the segment label are:

(1) coordinates of end-points,

(2) N-length (defined as the number of edges that comprise the line),

(3) E-length (defined as the Euclidean distance between the end-points),

(4) frequency with which the edges that comprise the line change direction,

(5) mean and variance of contrast across the line, computed along its length,

(6) mean and variance of difference between neighbouring points on either side of the boundary computed along its length.

Properties 2 and 5 can be used to give a measure of confidence that the extracted segment represents a meaningful unit of a visible boundary. The greater the length of, and the smaller the variance of the contrast across the line, the more confident one may be of the line. Property 6 gives an indication of the homogeneity of a thin peripheral strip of the regions that the line bounds. Properties 1, 2, 3, and 4 can be used to compute a measure of the straightness of the line. These properties are important for later use in the interpretation phases of processing.

2.4 Stage 4 - Postprocessing

2.4.1 TRIM1 (Step 4a)

Most unwanted line segments can be eliminated on the basis of low confidence (see Section 2.4.3), but it turns out that certain kinds of low-confidence edges result as a consequence of the idiosyncracies of the particular relaxation procedure employed. In particular, spurious edges are sometimes formed because multiple edges on a gradient are mistakenly believed to be distinct boundaries (a problem with any process where the gradients in the data are significantly wider than the largest mask), and any noise points remaining despite the earlier preprocessing stages get bounded by 'bubbles' (see below); these unwanted edges are best removed by a distinct process. Since they can be detected by their 'topological' nature, they can be removed before the confidence generation process in Section 2.4.2. This improves the latter procedure, as is explained in that section.

TRIM1 detects two kinds of unwanted edges: short edge-segments with at least one order-one vertex - called 'spurs', and some or all of the edges surrounding one-pixel regions - called 'bubbles'. For example, all edges marked with a cross in Figure 2.20 will be removed. Results of applying this process are shown in Figure 2.21.

2.4.2 CONFIDENCE GENERATION (Step 4b)

The confidence associated with a line segment will be based upon a measure of how dissimilar the points in the regions on either side of the line are to each other. While each line segment has been generated from edges which in turn were formed on the basis of local discontinuities, the line confidence will reflect the global difference between the regions in the neighbourhood of the line.

Some of the lines generated by the processes described up to this point are true, strong boundaries, and hence are desirable components of

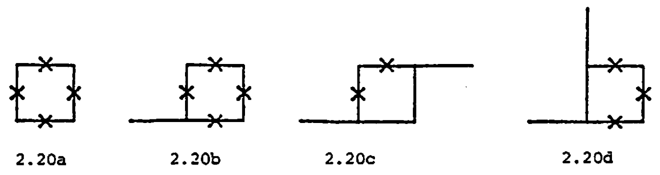


Figure 2.20. All edges marked with a cross will be removed to eliminate "bubbles" from the image. In Figure 2.20c, it is an arbitrary choice whether the left and upper, or right and lower sides of the bubble are removed.

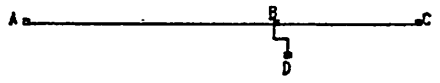
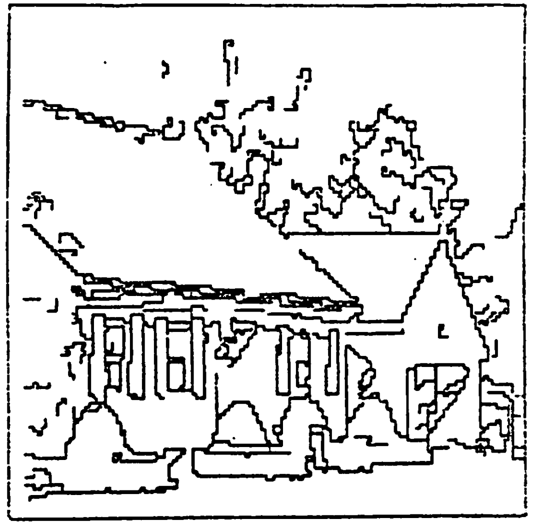
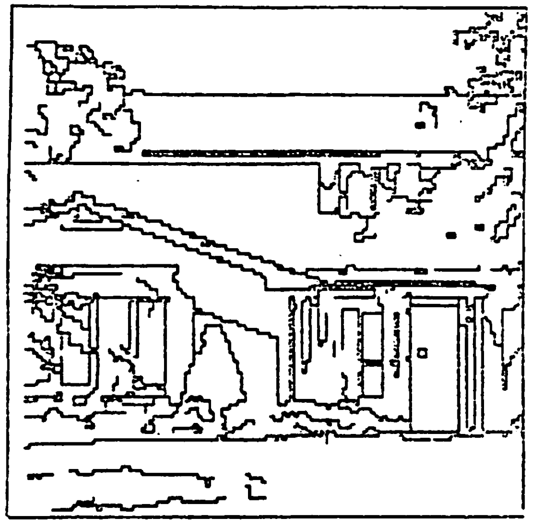


Figure 2.26. BD should be eliminated as a low-confidence edge. AB and AC should be merged to form AC which, being longer, will have greater conficence than AB or BC.



2.21a

Figure 2.21. Postprocessing. In Figure 2.21a the short edges and most of the smallest (1-pixel) regions have been removed from the data in Figure 2.18b.



2.21b

Figure 2.21b. Postprocessing applied to the data in Figure 2.18d.

a segmentation. Some others, though, are less clearly desirable; this set is characterized by lines of short length and/or low contrast. To experiment with the elimination of these lines we devised a simple heuristic confidence measure based upon length and contrast.

A Heuristic Confidence Measure

Let i be the length of a line segment, and L^* the length of what one would call a "long" line - say 1/4 the width of the image. Let C_i be the average contrast across the line and C^* the maximum average contrast one would expect to see in an image - say 3/4 of the difference between the extremes of the intensity scale. Then define

$$\hat{L}_i = \text{normalized length} = \text{Min}(L_i, L^*)/L^*$$
$$\hat{C}_i = \text{normalized contrast} = \text{Min}(C_i, C^*)/C^* \; .$$

Our heuristic confidence measure for the line is given by

$$f = \hat{L}_i + \hat{C}_i - \hat{L}_i\hat{C}_i \; .$$

This value is 1 if either $\hat{L}_i$ or $\hat{C}_i$ alone equal 1, zero if both are zero, and is monotone increasing in $\hat{L}_i$ and $\hat{C}_i$ for all other values.

A Statistical Confidence Measure

In order to test the effectiveness of this simple heuristic, we develop an alternative confidence measure based upon a statistical analysis. While much more elaborate computationally, this analysis is more justifiable on mathematical grounds than the other one.

For any line segment L we consider sets of pixels S_1 and S_2 on either side of L. We now test the hypothesis H_1 that the points in S_1

and S_2 belong to different regions, against H_0 that points in S_1 and S_2 belong to the same region. This statistical test is an extension of Yakimovsky (1976), but the manner in which it is employed differs in an important respect. Yakimovsky used the test to form the edges which comprise his boundaries. For each edge calculation, a predetermined neighbourhood was used for selecting the points in S_1 and S_2. Each edge was processed independently of each other, so the sets S_1 and S_2 taken each time did not necessarily accurately reflect the region structure formed by the boundary as a whole. In our case, on the other hand, the entire boundary is available for processing. This allows a better determination of the pixels which are to contribute to the analysis.

Let S_0 be $S_1 \cup S_2$. Under hypothesis H_0, S_0 is one region and will be modelled by the normal distribution $N(\mu_0, \sigma_0)$; under hypothesis H_1, S_1 and S_2 can be modelled by $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$ respectively. A maximum likelihood analysis leads to

$$P(L) = \frac{(\sigma_0{}^2)^{n_0}}{(\sigma_1{}^2)^{n_1}(\sigma_2{}^2)^{n_2}}$$

as a measure of the confidence of L, where $|S_i| = n_i$, $i = 0,1,2$, and $n_0 = n_1 + n_2$ (Yakimovsky 1976).

Let $S_i = \{(X_{ij}, Y_{ij}, I_{ij}), j = 1,\ldots,n_i\}$, $i = 0,1,2$, where I_{ij} is the intensity of pixel (X_{ij}, Y_{ij}) in set S_i. The model described above assumes the I_{ij} are normally distributed about mean μ_i, independent of their location in the image, i.e., it is supposed that the readings I_{ij} are governed by the probability distributions

$$\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(I_{ij} - \mu_i)^2}{2\sigma_i^2}}$$

independently of (X_{ij}, Y_{ij}). We can improve the test procedure by using a more sophisticated statistical model of the regions on either side of the putative segment L.

Rather than assume that there are no spatial dependencies in the gaussian distribution of the I_{ij} in each region, we will improve the model by assuming that μ_i can vary linearly across the region. This leads us to the concept of a dynamic mean μ^*, which is a function of the spatial coordinates, and represents the expected value of the intensity at a given point in the region. Our model should be better than the simpler one since object brightnesses in real world images are not usually constant but vary across the surfaces concerned. We will suppose that at each point (X_{ij}, Y_{ij}) the intensity is normally distributed about the dynamic mean $\mu_i^*(j) = \mu_i + a_i X_{ij} + b_i Y_{ij}$. This assumes that there is a constant gradient of intensity (strength a_i in the X-direction, b_i in the Y-direction) across the region, which will be fairly realistic, at least over small parts of the region. To ensure the accuracy of the analysis, then, we see that the sets S_1 and S_2 should be composed of pixels lying within a short distance of L.

The values of a_i and b_i for $i = 0,1,2$ will be determined by a least-squares fit. In those cases where there are no intensity gradients, a_i and b_i will evaluate to zero, showing that our approach covers

all those cases where the simpler analysis would have been sufficient.

For each of the sets $S_i = \{(X_{ij}, Y_{ij}, I_{ij})\}$ we perform the following computations:

(1) determine constants a_i, b_i, μ_i which minimize

$$\sum_{S_i} [I_{ij} - (\mu_i + a_i X_{ij} + b_i Y_{ij})]^2$$

(2) determine the variances $(\sigma_i^*)^2$ by

$$(\sigma_i^*)^2 = \sum_{j \in S_i} \frac{(I_{ij} - \mu_i^*(j))^2}{n_i} .$$

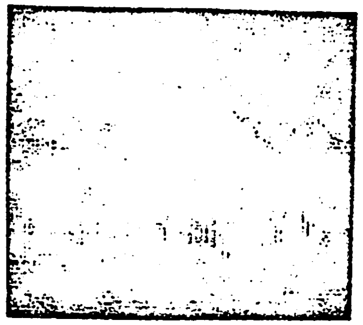
The measure of the confidence of L is now given by

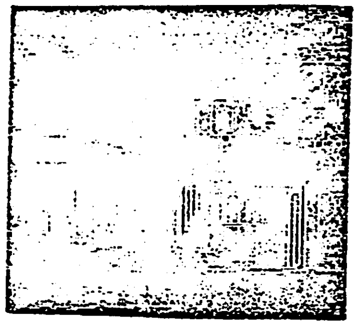
$$P(L) = \frac{(\sigma_0^*)^{2n_0}}{(\sigma_1^*)^{2n_1}(\sigma_2^*)^{2n_2}}$$

or, alternatively, but preserving the ordering of the confidences of the various line segments,

$$P(L) = n_0 \log\sigma_0^* - n_1 \log\sigma_1^* - n_2 \log\sigma_2^* .$$

In Figures 2.22 - 2.25 we compare this confidence measure P with the heuristic one f described earlier. The pictures display all those line segments with confidence greater than a suitably scaled threshold value. It will be seen that both methods do equally well regarding the assignment of high confidence measures to long high-contrast lines, such as the sides of windows or houses, and low-confidence values to many of the boundaries of texture elements in the trees and shrubbery.


2.22a


2.22b

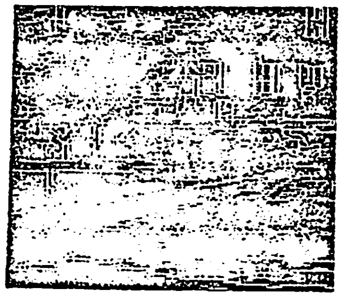
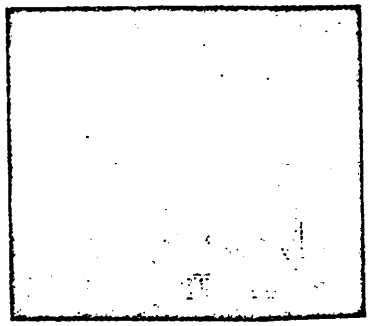
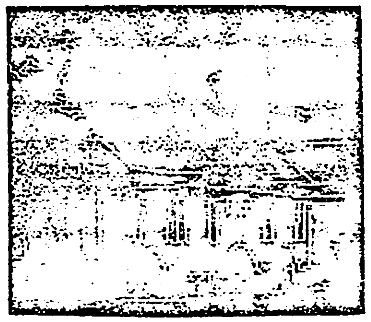

2.22c

Figure 2.22. Edge images which are used to compare the two confidence measures described in the text. In Figures 2.23 - 2.25 these images are shown thresholded at different levels. Due to the different scales produced by the two measures, an exact comparison is impossible, but sequences of comparable threshold values were chosen for the two algorithms.


2.23a

2.23e


2.23b

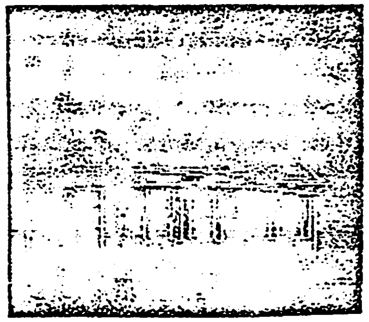
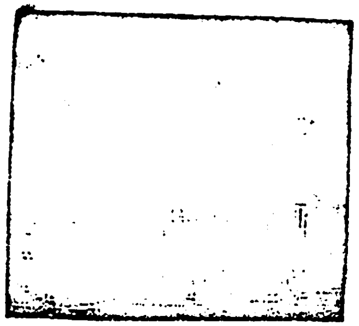
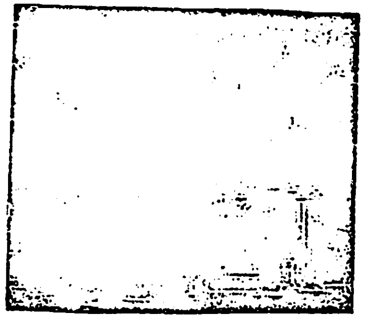
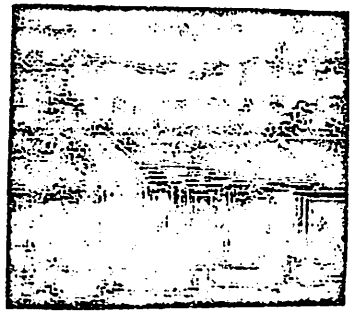
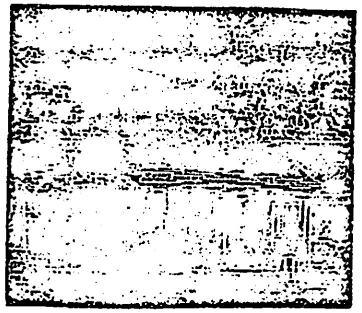

2.23f

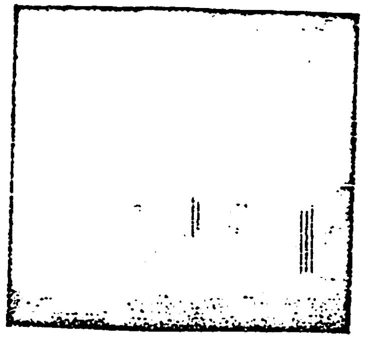
Figure 2.23. Successive thresholdings of the edge image in Figure 2.22a. Figures 2.23a - 2.23d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 2.23e - 2.23h show the same with heuristic confidence measures.

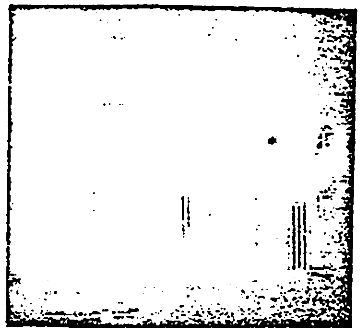

2.23c

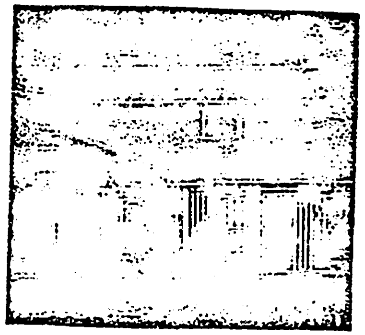

2.23g


2.23d


2.23h


2.24a


2.24e


2.24b

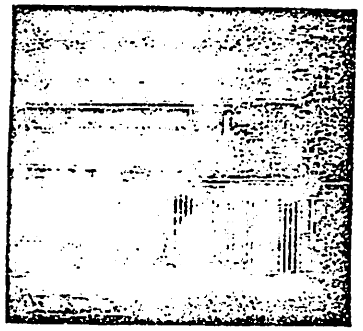
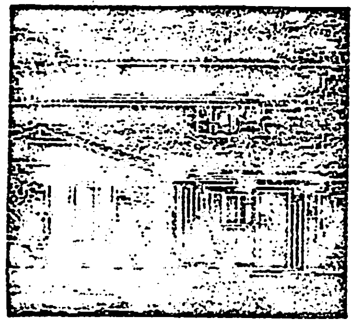

2.24f

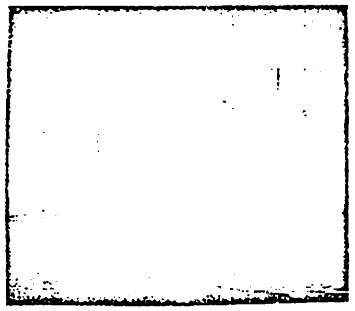
Figure 2.24. Successive thresholdings of the edge image in Figure 2.22b. Figures 2.24a - 2.24d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 2.24e - 2.24h show the same with heuristic confidence measures.

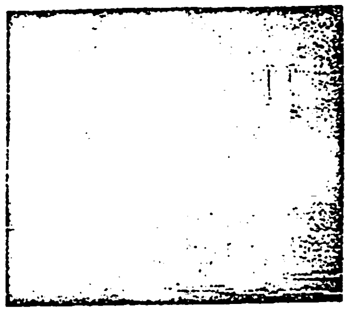
2.24c

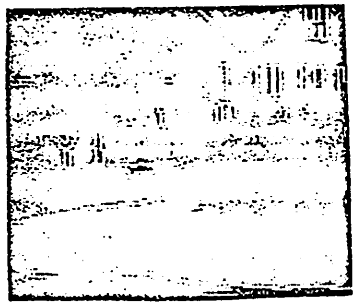
2.24g

2.24d


2.24h

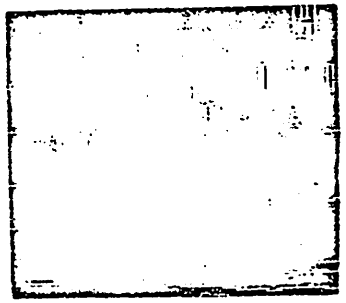

2.25a

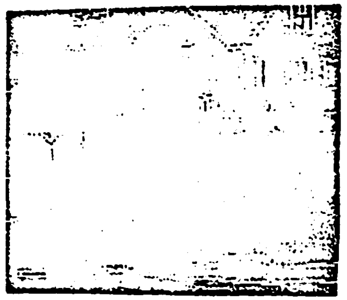

2.25e


2.25b

2.25f

Figure 2.25. Successive thresholdings of the edge image in Figure 2.22c. Figures 2.25a - 2.25d show line segments with statistical confidence measures under successively decreasing thresholds. Figures 2.25e - 2.25h show the same with heuristic confidence measures.


2.25c


2.25g

2.25d

2.25h

This does not mean that the 'sophisticated' analysis is not good. On the contrary, it sets a standard with which our heuristic measure may be compared. The heuristic shows itself to produce very acceptable results, and if it proves to be generally reliable, it is recommended over the other method due to computational efficiency.

2.4.3 TRIM2 (Step 4c)

Line segments can be removed on the basis of their relative confidence ratings by removing lines with confidences less than some threshold T. This process should be performed conservatively (with a low threshold) for the following reason. Consider Figure 2.26 and suppose that AB and BC have average confidences, while that of BD is relatively low. Any reasonable threshold should get rid of BD, but if it is set too high there is a danger that either AB or BC could be removed as well. This is to be avoided because as soon as BD is gone, ABC might become a single line segment with a much higher confidence than that of either AB or BC alone.

The process of removing weak and uncertain line segments may be performed iteratively. Instead of immediately adjusting threshold T to what is thought to be an optimal level, it can first be set to a lower level T'. Lines of confidence < T' will be removed, stage III grouping reapplied, T' increased somewhat and the whole process repeated until the desired level is reached. In fact, regrouping could occur after each line segment is removed.

2.5 Conclusions

Not many segmentation systems using relaxation have been tried on outdoor scenes such as the ones examined in this paper. An application which does use real data, though, is that of Zucker et al. (1977); here they examined with some success the data in LANDSAT images. Their use of the probabilistic relaxation (the third kind described in Section 2.2.3) had a tendency to form boundaries two or three pixels thick, however. This did not seem to pose a great problem with their data, which had relatively few boundaries, but it is unclear how their techniques would fare on very detailed images.

Bajcsy and Tavakoli (1976) used a sequence of processes to recognize roads and intersections, again from satellite images, although they did not employ relaxation. The processes they used include, with the equivalents in our system noted in parentheses, a Strip Detector (DIFFERENTIATION), a Road Grower (RELAXATION), a Thinning Operator (SUPPRESSION) and a Short Segment Eliminator (TRIM). Interestingly, they thinned their edges (roads) after the extended segments had been formed through the road-growing process, whereas we perform multiple-edge-suppression first, followed by relaxation to join the individual edges. Their system was successful in detecting most of the major roads in the images they processed for that paper. However, due to the differences in the details of the algorithms and the types of images used, it is difficult to make a more pointed comparison between their work and ours.

In fact, it is in general a very difficult problem to make explicit measurements of the success of a segmentation system. It has long been

argued in the scene-analysis community that a set of standard images should be constructed and distributed so that different techniques may more easily be compared, and this is just now being done. Unfortunately this venture may be premature, since little work has been done on the very important problem of establishing criteria for judging the performance of a segmentation system.

For a scene-analysis system such as the one described in this paper, it may only be possible to make assessments in the light of the system in which it is embedded. For example, the success of this system might be measured by how good a model the VISIONS system can make from it. Even if that were possible, however, it would not be clear to what extent the success was due to any particular component of the system. The matter is complicated by the fact that the system which uses the segmentation might have been tuned to accept the kind of segmentation peculiar to that subsystem. We are left with a situation where the only judgements that can be made, for the time being at least, are subjective ones. This was done explicitly in the case of the two confidence measures, and is essentially the only way that, for example, different relaxation schemes may be compared.

We have presented a boundary-analysis system consisting of four stages: pre-processing, generating the edge representation, grouping and post-processing. The output of the system is a set of line segments with a list of attributes, such as length and confidence. This output may be used in conjunction with region information for making semantic hypotheses about object identities.

Segmentation is a difficult problem, as is borne out by the complexity of the processing it appears to require. We have chosen an implementation in which this complexity takes the form of a series of processing stages, each of which performs a fairly simple, direct function. The advantages of this modular approach are twofold. Firstly, the effects of each stage are reasonably well understood, so that the state of the data at every point is fairly well-defined. Thus each stage can be tailored to suit the data upon which it must act.

The second advantage is that any stage can be omitted, or replaced by a more sophisticated variant, if desired. This greatly facilitates system development; in particular, it makes experimentation with different algorithms very straightforward.

C H A P T E R III
BIOLOGICAL IMPLICATIONS

Introduction

In this chapter, we will investigate two aspects of biological systems which may lend themselves very well to cooperative analysis by neurophysiologists, neural modellers and machine vision experts. We will look at current physiological knowledge and suggest certain directions for further explorations and experiments to be performed. First, we will examine the data on 'multiple visual systems' which suggest the separation of detection of motion, tracking of motion and static 'form analysis'. The implications of this kind of division of function to a control system for directing the gaze will be discussed. We will then examine the receptive field organization of several 'layers' in biological visual systems, and briefly investigate possible reasons for the existence of large receptive fields, concentrating on the idea of 'population encoding'. We will suggest that the encoding of information by populations of neurons may be the device by which visual systems achieve much higher visual acuity (along several dimensions) than is possible otherwise.

3.1 Evidence for Separate Form and Movement Channels

We will present below evidence that there are, *inter alia*, three main visual pathways from the retina to higher brain centres in mammals, two projecting to the cortex, and one to the superior colliculus. We will also present evidence from psychology that there are separate channels mediating form and movement perception. We will attempt to show that these are the retino-cortical projections, while the superior colliculus plays a part in the detection of and orientation to novel visual stimuli.

It is widely argued that there are two major, separate and parallel (but not necessarily non-interacting) channels in the visual systems of the higher mammals for what might be called 'form' and 'movement' perception. These claims are made from studies of, for example, behavioural changes after brain lesions/ablations, and from adaptation studies (Tolhurst (1973); Kulikowski and Tolhurst (1973); Frisby and Clatworthy (1974); Ingle (1967); Nelson (1974); Schneider (1967); Trevarthen (1968)). Furthermore, neurophysiologists have revealed several pathways along which information travels from the retina (Enroth-Cugell and Robson (1966); Fukuda and Stone (1974); Ikeda and Wright (1975); Stone and Dreher (1973); Stone and Fukuda (1974)). We will discuss these findings in some detail, and investigate their relationships. This study may be used to suggest the allocation of functions to different components in a machine vision system.

3.1.1 Neurophysiological Evidence

In the following discussion, we will generalize over several species, in particular the cat and the rhesus monkey. There is overall a very close correspondence between features in the respective visual systems (see, e.g. Schiller and Malpeli (1977)), but one must be careful

not to draw any hasty conclusions, because some differences do exist. For example, cells in the monkey superior colliculus exhibit little or no directional selectivity, while about 3/4 of those in the cat exhibit it strongly . As this might be due to the different ecological niches that these animals occupy (Cynader and Berman 1972), it is important to remember that there is a difference between "vital for the cat" and "vital for vision in a higher organism".

Enroth-Cugell and Robson (1966) found that they could classify most retinal ganglion cells into one of two types, called X and Y cells. They and other workers, e.g. Hoffman (1973) have derived a large set of properties used to distinguish these cell-types:

X cells are sustained or tonic; Y cells are transient or phasic.

Y cells are more sensitive to stimuli or low spatial frequency than X cells.

Y cells are less sensitive to stimuli of high spatial frequency than X cells.

Y fibres (i.e. fibres from Y-cells) are faster conducting than X fibres.

Y cells respond best over a very different stimulus speed range than the best range for X cells.

Y cells have larger receptive fields than X cells.

Y cells are proportionally denser in the periphery, while X cells are denser in the center of the visual field.

Signals from photoreceptors in X cells' receptive fields add linearly; this is not true for Y cells.

The segregation of the two channels is maintained from retina to cortex (Stone 1972; Stone and Dreher 1973), and there is evidence that simple cells may be the targets of the X fibres, and (some) complex cells the targets of the Y fibres (Maffei and Fiorentini 1973; Hoffman and Stone 1971).

The Y fibres from the retina bifurcate, the second branch leading to the superior colliculus. A brief comparison of the collicular and cortical systems will now be given, to demonstrate that they are significantly different in a functional sense.

Cells in the superior colliculus are well known to be very sensitive to moving contours. They are almost entirely binocularly driven (Cynader and Berman 1972; Sterling and Wickelgren 1969), while cortical cells contain proportionally more monocular cells.

Cortical cells have high orientation specificity (Hubel and Wiesel 1965, 1968), while collicular cells don't show such selectivity (Cynader and Berman 1972; Sterling and Wickelgren 1969; Goldberg and Wurtz 1972). Colour sensitive cells are found in the cortex (de Monasterio and Gouras 1975), but there is evidence that none exist in the superior colliculus (Humphrey 1970).

There is considerable evidence that while the cortex mediates pattern discrimination, the superior colliculus mediates orientation in space. By differentially ablating visual cortex and superior colliculus in the hamster, Schneider (1967) has shown an anatomically-based dissociation between the animal's ability to learn and identify a stimulus (cortex intact), and its ability to orient to the stimulus in space (superior colliculus intact). Trevarthen (1968) also proposes the existence of two visual systems; one for orientation and the other for object identification, using the superior colliculus and cortex respectively as major units in the processing. Our conclusions are that the situation is more complicated than that, and will be presented below.

3.1.2 Psychophysical Evidence

Frisby and Clatworthy (1974) examined the effect of similarity of stimulus on degree of adaptation. The test stimulus was a moving grating whose orientation and/or movement axis could be the same as or different from that of the adapting stimulus, giving four combinations. The fact that only one dimension of similarity (e.g. the same orientation but different axis, or vice versa), gave an intermediate amount of adaptation was used to argue for two channels in the visual system; one for movement, the other for form perception.

It has been shown that motion after-effects (MAE's) can be mediated either by monocularly or binocularly driven neurons (Gates 1934; Barlow and Brindley 1963; Freud 1964; Walls 1953). It has been proposed that this is due to these neurons being at different sites along the same pathway (Anstis and Moulden 1950). However, Favreau (1976) has shown that interocularly transferred and dichoptic MAE's exhibit substantially different decay rates, which would tend to rule out this thesis, and suggest instead that the neural populations must be operating in parallel on different pathways.

Similarly, Pantle and Sekuler (1969) found that subjects experienced different thresholds for test gratings after adaptation to gratings at different contrasts. By varying the contrast of the adaptation grating, they discovered two components in the response to test gratings. One component depended on the grating's orientation and was independent of its movement. The other component was dependent on the direction of movement. These two components were different functions of contract. These results suggested the functional separation of orientation and direction-sensitive elements in the visual system.

Again, Kulikowski and Tolhurst (1973) showed that it is unlikely that motion detection is performed by neural elements mediating very fine spatial discriminations. In particular, there is evidence that motion perception takes place in (at least) two different channels.

We see, then; that it has been suggested a number of times in the literature that there exist two distinct visual systems in higher mammals. However, it is apparent that there are at least three: two within the cortex and one in the superior colliculus. We will propose now the roles that these three systems play in visual and, in particular, motion perception.

3.1.3 The Collicular System

We will start with the superior colliculus. Schneider (1967) has shown that it plays a crucial role in a hamster's orientation ability in space. Fukuda and Stone (1974) have shown that Y cells project to the superior colliculus, but X cells do not. Their targets in the superior colliculus thus have large receptive fields, located primarily in the periphery of the visual field and sensitive to moving boundaries, but not particularly so to the exact shape of the boundaries. This is exactly the kind of input characteristics required of a system which is

to determine, by an analysis of the optical events on the retina, the orientation and perhaps speed of moving objects.

It is also well-known that the superior colliculus plays a part in eye-movements. Mohler and Wurtz (1976) found a "visually triggered movement cell" at the border between superficial and intermediate layers of the superior colliculus. This type of cell would increase its discharge rate before saccades made to a visual stimulus, but not before spontaneous saccades of equal amplitude made in the light or dark. The size of the movement fields of these cells was about the same as the size of the visual fields of the superficial layer cells (see also Schiller and Koerner 1971). It is strongly suggested that the superior colliculus is involved in making saccades to novel stimuli entering the visual field, but in the role of "agent, not initiator" (Wurtz and Mohler 1976). The results did not rule out the possibility that it may play the part of "co-initiator".

Schiller and Koerner (1971) suggest that the superior colliculus may well also be involved in the smooth pursuit of visual targets, as well as making saccades to the targets. However, this may be just because the only access the visual cortex has to the oculomotor system is through corticotectal projections (Palmer and Rosenquist 1975). The findings that cortical cells project to collicular cells which "see" the same area of the visual field (McIlwain 1977) would support this view.

It may be concluded, then, that the branch of the Y fibres to the superior colliculus is centrally important in indicating movement in the periphery of the visual field (to which saccades may be made). It will be noted that the exact shape of such a stimulus is unimportant at this stage; the visual system merely needs to know its position relative to the observer.

3.1.4 The Cortical System

It is our further contention that the X and Y systems projecting to the cortex subserve two quite different functions: form perception and tracking of moving objects respectively. This is not to say that the X and Y systems are totally separate; they undoubtedly interact, but we propose that they are the major informational channels of their respective systems.

While Hubel and Wiesel's (1962) hierarchical model of processing in the visual cortex may be partially valid, it is generally regarded nowadays as not being strictly accurate. Hoffman and Stone's (1971) data suggest that the X and Y systems feed differentially into the simple, complex and hypercomplex cells. Maffei and Fiorentini's (1973) data suggest that the direction sensitive elements may be in the Y system, and the orientation sensitive elements in the X system. This would lend credence to the suggestion that the Y system is involved in motion detection and the X system in pattern recognition. The high concentration of X cells in the fovea would support this view. In fact, so would the relative scarcity of the Y cells in the fovea, since because tracking is concerned with detecting deviations from the current fixation point, it would require detectors off-centre.

This would not explain how fine tracking of very slowly moving ob-

jects is possible, as they would rarely trigger the Y cells. We should note, therefore, that it has been observed that the fovea is more sensitive to very slow motion than is the periphery (Lichtenstein 1963) although the opposite is true for high velocities.

We may thus make the following conclusions:

1. Movement detection in the periphery (or a saccade to such movement) is mediated by the superior colliculus.
2. Tracking objects at moderate speeds is performed by the Y system in the cortex.
3. Tracking objects at very slow velocities is performed through the X system.

As it would be very unreasonable to suppose that these systems operate in disjoint but contiguous ranges of speeds and eccentricities, there must be some overlap. Indeed, it is assumed that there is a continuum of effect, with each system always contributing something to the analysis; the major contributors in each range being as given above.

The maintenance of a stationary percept of the world during fixation drifts must be accomplished somewhere: the above argument would suggest it to be in the cortex (X system) since the stimulation would be similar to that from slowly moving objects. From studies of brain lesions which indicate the location of sensory and motor functions relevant to central vision to be primarily in the cortex, Trevarthen (1968) concludes that "determination of fixation drifts.... is essentially a cortical function".

3.1.5 Implications for a Motion Analysis System

The existence and properties of these separate channels suggest how motion detection may be handled by a machine. Simulation experiments with models of the human visual system can provide insight into the biological mechanisms, and in turn suggest further experiments in a way that the more *ad hoc* Artificial Intelligence approaches to motion detection cannot. We therefore suggest that a motion analysis system should incorporate as far as possible what is known of the human visual system; in particular its multi-channelled nature. This will mean that one should not attempt to construct a single processing module which will perform "motion analysis" per se. Rather, one should be permitted to use different mechanisms for different kinds of motion analysis (e.g. orientation and tracking), possibly with separate submechanisms for subsets of operational ranges (e.g. speed). A central problem to be overcome in this paradigm is to ensure smooth transitions when different modules become active as the stimulus parameters change.

One may conceive of a basic system consisting of two modules. One module will detect novel motion, that is, when an object enters the visual field or when an object within it accelerates. This will be an analogue of the collicular system fed by the Y fibres. The primitive receptive elements will be ones which fire when a boundary (of a specific contract and/or direction) enters their receptive fields.

It might be thought that these elements should be 'velocity detectors' - neural elements which directly signal speed; there is, however, no firm evidence for their existence in biological systems. On the other hand, Pantle and Sekular's (1968) results indicate that there are several channels responsive to separate but overlapping velocities. Thus if there is a set of channels C = {Ci, i = 1...n} fed by populations of neurons responding maximally to different velocities, and if fi(v) is the response to a moving boundary in channel i, then the n-tuple (f1(v), f2(v),..., fn(v)) will in some sense be the signature S(v) of the velocity v (see Erickson 1974). S(v) should contain enough information to determine v, as long as the response profiles of the different channels overlap sufficiently to cover the entire range of velocities detectable by the system. Benefits of population encoding are looked at in some detail in the next section.

The goal of the system will be to determine at what point in the field of view motion has taken place, and what is the direction and velocity of this motion. Such information is presumably necessary and sufficient for a saccade to be made to the vicinity of the object.

Considering that neurons in the superior colliculus are not particularly selective for the specific contours of a moving object (Cynader and Berman 1972; Goldberg and Wurtz 1972; Sterling and Wickelgren 1969), we suggest that the system not be expected to derive more than a very gross estimation of the size and shape of the moving object (at this point).

The second system should be concerned with the tracking or pursuit of a moving object. It will be assumed that the object is fixated, or nearly so, and is being tracked at approximately the right velocity. What will be observed, then, is a gradual drift away from or jitter about the fixation point. The primitive receptors will respond to the presence of edges at different orientations in their receptive fields. They will not be velocity detectors, but will of course respond to moving objects, since such movement will entail the object's entering the receptive field. This system, then, will detect movement on the basis of change-of-position of the object, rather than a direct perception of its velocity.

In order to perform an analysis of movement in digitized data based upon change-of-position, it is necessary to determine what points (in the general sense) in one frame correspond to points in the subsequent frame. For this reason, the Stimulus Matching problem becomes of crucial importance in motion analysis, and will therefore be discussed in great detail in Chapter V.

3.2 Receptive Fields and Feature Detectors

Given the existence of the various biological visual systems described in the previous section, one might well ask how they accomplish their tasks. It would be useful to discover what language the brain uses in its visual analysis, i.e. what are the primitives with which it describes the visual world, at the early stages of processing at least. Once again we turn to neurophysiology and look at some results obtained in attempts to trace the flow of visual information from the eye to higher centres in various animals.

One of the earliest of these experiments was performed by Lettvin et al. (1959). They recorded responses along fibres leading to the

frog's optic tectum as the animal was presented with certain visual stimuli. They discovered four distinct classes of response to particular stimuli in localized regions of the visual field. The four kinds of stimulus were (1) local sharp edges and contrast, (2) curved edges of dark objects, (3) moving edges, and (4) local dimming. Each class of fibre terminated in a different layer, the layers being retinotopic maps of the visual field and in register with each other.

In their famous series of experiments, Hubel and Wiesel (1962,1965, 1968) found somewhat similar effects in the cat and monkey. They recorded responses of cells in the visual cortices of these animals on presentation of selected visual stimuli. Again, they found that each cell had an area in the visual field to which it was responsive, and that the cells in different layers formed registered retinotopic maps. They classified the cells into three groups according to their responses. "Simple" cells also responded to edges, but were less specific. Both the light and dark edges of a moving stimulus caused a response in the same region, the activating region showed no subdivision into spatially separate zones, and the receptive fields were larger. The third class of cells were called "hypercomplex". The responses of these cells were inhibited by stimuli extended along the axis of orientation, thus making the cells maximally excited by corners and ends of bars.

Now, Hubel and Wiesel suggested that these cells formed a hierarchy with the simple cells feeding the complex cells which in turn fed the hypercomplex. Subsequent results, such as those of Hammond and McKay

(1975) who showed that kinetic contours excited complex cells but not simple ones, have shown that this model is not entirely valid. The results of Schiller et al. (1976a, b), however, suggest that some complex cells may receive input from simple cells, while others do not.

Whatever the precise mechanism may be, it is clear that there are many retinotopic maps of the visual field present, each one exhibiting some kind of transformation of the visual information. It is not known yet exactly what the visual system does next with this information, but the very fact that this processing occurs suggests that a similar procedure may be useful in machine vision.

Cells which are responsive to certain kinds of stimulus may be regarded as being detectors for these stimuli - the magnitude of the response increasing with the 'fitness' of the stimulus. Indeed, the type 2 cells of Lettvin et al. have been called "bug detectors", since the properties of the stimuli to which they are maximally sensitive characterize flies and other food objects for the frog.

These properties of biological visual systems suggest how machine vision systems may be structured. A machine vision system may be organized as a series of arrays or layers of data, each containing some retinotopic transformation of the visual data. To achieve this, we can imagine registered arrays of processing elements to perform these computations, each such element receiving its input from a finite, localized receptive field in one or more layers. [N.B. In an implementation on a serial machine, this will of course be simulated.] One of these processes will extract edge- and corner-like features from the visual image.

Now, the retinal ganglion cells in higher mammals, e.g. the cat and monkey, have centre-surround organization (see, e.g. Kuffler 1953) and it is suggested (Schiller et al. 1976b) that the edge-specificity of cortical cells comes about by lateral inhibition between cells whose direct input is solely from cells in the Lateral Geniculate Nucleus. (These cells themselves have centre-surround receptive fields.) This multi-stage process can be avoided by applying certain edge-operators or templates directly on the visual image, performing a kind of spatial differentiation. This approach was taken in Chapter II and will be taken again, in a slightly different form, in Chapter IV.

3.2.1 Large Receptive Fields

Most of the receptive fields of cortical cells receiving input from in or near the fovea are in the range 1/2 - 5 degrees across (see, e.g. Hubel and Wiesel 1962). Much larger fields are found too, especially near the periphery. These sizes are much greater than that of visual acuity. A good discussion of the geometric properties of visual systems with large receptive fields is given in McIlwain (1976). The question arises, what benefit is there in these large receptive fields? We are particularly interested in the role such large fields play in the analysis of motion.

Possible Reasons for Large Receptive Fields

(1) Redundancy. In a large receptive field (RF) situation as exists in mammals, neighbouring RFs overlap to a large extent. Thus the

projective field of retinal points on the ganglion cell layer is large. Many cells are simultaneously responding to a single point of stimulation. So, if due to noise or any other reason a small number of these cells fail to fire, others will still signal the existence of the stimulus.

(2) Intensity Encoding. As (1) above, but the thresholds of those cells in the projective field of a retinal receptor cover a wide range. If this is the case, out of those which can potentially fire given a stimulus, the number which do respond will be a measure of the intensity of the stimulation. The use of population encoding as a general device in sensory perception is discussed below.

(3) Fatigue. Slow-moving stimuli (or ones that are being tracked) will remain within the RF of a responding cell for an appreciable period of time. If the cell fatigues, and lateral inhibitory connections are present, inhibition on neighbouring cells will be reduced, so their responses when the stimulus enters their RFs will be enhanced. This process may act as a dynamic gain control in a feedback control circuit, the purpose being to get the eyes (back) on target as quickly as possible.

(4) Slow Movement. If a stimulus moves slowly across a large RF, the ganglion cell will fire continuously, but may not receive information of the right kind to determine that the stimulus is moving at all. Thus, no sudden offset-onset will occur in neighbouring cells as the stimulus passes over their RFs, so no neural network which simply tests for this situation (cf. Barlow and Levick 1965) will work. In fact, it

may be a desirable thing for an organism not to notice very slow movements. [N.B. As it happens, this is not a situation unique to biological vision systems. In machine vision systems, the average pixel-width is a lot larger than that of a retinal receptor, in terms of visual angle. Thus if a dynamic image is sampled frequently enough, inter-frame movements are very small, and 'objects' may remain 'in' a given pixel for a number of frames.]

(5) Large RFs have the advantage that even if objects in the visual field are sparsely tectured, neighbouring cells (ganglion or cortical) will still signal movement across the entire width of a moving surface. In this manner, region-growing becomes feasible as a means of grouping points for segmentation.

3.2.2 Population Encoding

Receptive fields of the ganglion cells may be anywhere from about 5 degrees to 30 degrees across, and hence must overlap to a considerable extent. Thus instead of a stimulus being encoded (accurately) by a small number of signals, it may be carried by many signals, none of which alone describes it very precisely. We suggest that it is by means of population encoding that the visual system achieves accuracy otherwise unattainable.

There are several attributes of a stimulus that the visual system may wish to determine, including location, orientation, direction of motion, speed and colour. We claim that it is plausible that all of these computations are performed with the aid of population encoding, a view supported by Erickson (1974).

Let us consider the two very different organizations mentioned above. In general, a system employing the first organization consists of a large number of very finely tuned neurons. There would possibly be a need for as many neurons as discriminations are to be made. Moreover, if one of the neurons which were to fire in a given situation did not do so, then much of the information would be lost. Now, in the second organization, the neurons are much more broadly tuned and have large receptive fields. Many neurons will fire for any (appropriate) stimulus, and it is the relative strengths of these signals which will ultimately describe the stimulus.

We can see that the small receptive field case certainly seems to have serious problems, but how would population encoding achieve the accuracy required? We will address two issues. Firstly, how could the broadly-tuned neurons do nearly as well as finely-tuned neurons? The answer is that the recognition of a stimulus would be by means of what Erickson (1974) calls an "across-fiber pattern", i.e. the pattern of activity evoked across the population of responding neurons - that which we called a 'signature' in Section 3.1 above. He demonstrates how by examining the activity in a collection of neurons, a number of stimuli could easily be distinguished, where no one neuron could make a single discrimination.

It is likely that this kind of mechanism underlies the perception of colour. Through combinations of just three hues (red, green and blue) at different intensities, humans can perceive a whole spectrum of

of colours. In fact, not even all three are necessary for the impression of the entire visual frequency range (McCann and Benton 1969). Some interactive process between the responses of the photoreceptors undoubtedly takes place to achieve this.

We will dwell for a minute on some more recent experiments reported by Land (1977) in order to suggest an experiment conspicuous by its absence. The experimental paradigm used by Land and his colleagues consisted of matching up coloured pieces of paper in a multicoloured collage with standard colour 'chips' when viewed under different coloured lighting conditions. They found that the perceived colour of an area was independent of the light flux reaching the eye from that area. Instead it was a function of how the ratio of the components of the light in certain wavebands compared with the light reflected from a white surface. They calculated the ratio of the reflectance of the given surface and the white one by forming the sequential product of such ratios across all boundaries along any path between the surfaces. They use a simple technique for discovering which area has the highest reflectance, and this is taken to be white; in every experiment, at least one of the coloured surfaces was white. They do not report performing the experiment with no white present. If perceptions of colour are still accurate in such a modified situation, then their theories would be confounded. If, on the other hand, an area (such as pale yellow) had the highest lightness of all surfaces in the three relevant wavebands, and if every colour was perceived as, in this case, "too blue" (i.e. green as blue-green, yellow as white) then their theories would be confirmed.

The second problem we will look at asks how broadly-tuned neurons achieve better accuracy than single very finely-tuned neurons. This leads to the further question of, given the responses of several broadly -tuned neurons, how to compute the relevant parameters of the stimulus to which they are responding. While it is true that in some circumstances the visual system may not need to 'decode' the population code, in other circumstances it might. We will investigate below a method by which the responses of a collection of units, taken in pairs, may be translated into the unique stimulus which gave rise to the responses.

Consider the problem of the determination of an object's velocity. The existence of separate 'channels' for different velocity ranges is supported by the adaptation experiments of Pantle and Sekuler (1968). At the neural level, though, while some neurons are more sensitive to certain velocity ranges than others, they are generally very widely tuned (Pettigrew et al. 1968). To illustrate how to extract precise information from a collection of signals, each with potentially ambiguous information, suppose that each neuron has a response curve like that in Figure 3.1. Given a certain response from one neuron, the stimulus could fall within two parameter subranges. Given the response from another, differently tuned neuron in addition, and knowing that they both signal the same stimulus, greatly narrows down the possible stimulus ranges. Figure 3.2 shows how the accuracy is improved given the response and error margins from just two neurons.

The mechanism by which the neuron populations arrive at an accurate determination of stimulus parameters is unknown, but an iterative method

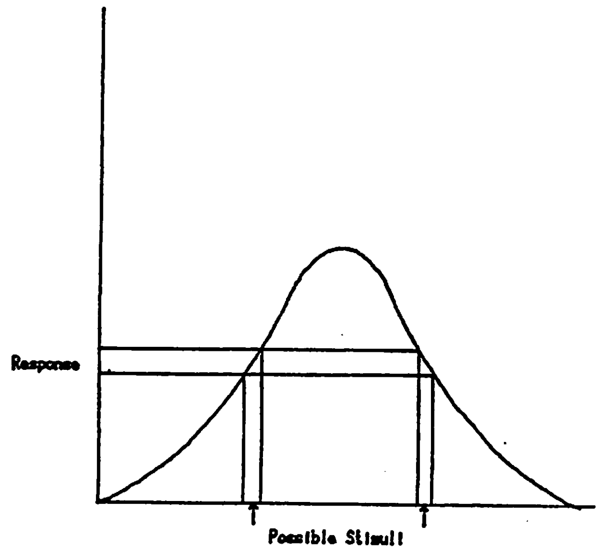


Figure 3.1. Ambiguous Response from a Single Stimulus. Given a response of a feature-detector with possible error margins indicated by the horizontal strip, the stimulus could have fallen within either range indicated by the vertical strips.

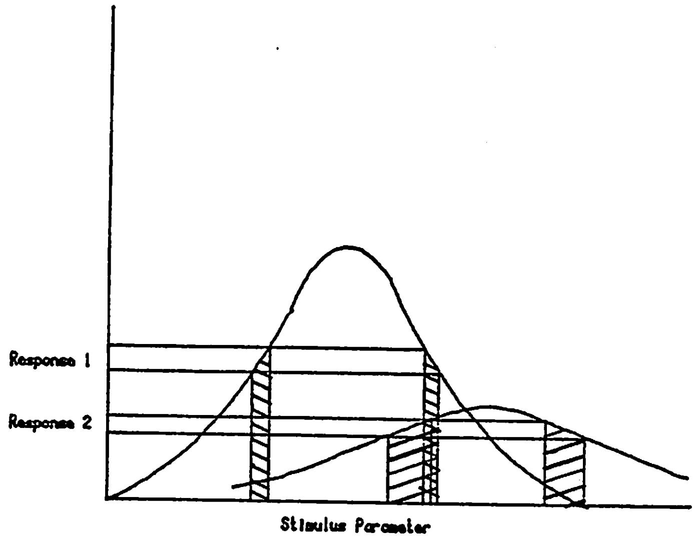


Figure 3.2. Reduction of Ambiguity Using Two Detectors. Suppose two detectors (response curves shown) respond to a given stimulus. Then although each response alone is ambiguous, taken together only one stimulus range is possible. Note that the width of this range is narrower than that from either response alone. The larger the number of differently-tuned detectors that respond, the more exactly the stimulus can be pinpointed.

seems entirely feasible. The general procedure would be to arrive at a global consensus, that is everywhere locally consistent with the input, via cooperative mechanisms. We are interested in the analysis of motion, and therefore would like to see if the inherent ambiguities in motion may be resolved this way. We present below one possible iterative solution, not in any way claimed to be the biological mechanism.

A 'Velocity Detector' Processing Network

In this section we will consider a neural-like architecture which may be appropriate for the computation of velocity. Let us suppose that there is a layer or array, each unit of which computes the velocity signalled at the corresponding patch of retina (its receptive field). Neighbouring units with similar indications of velocity would link up to form a segmentation of the object. Boundaries of several objects with different velocities may be found simultaneously this way.

The problem with this approach is in the local determination of velocity. Suppose that for each small patch of retina there is a set of computational elements $\{x_i\}$ each of which responds maximally when an edge at orientation α_i passes over the patch in a direction perpendicular to that of the edge. Suppose there is a mechanism for selecting the maximum response (cf. Didday 1970). Suppose further that an edge of orientation α_j passes over this patch moving with velocity v in a direction of Θ where $|\alpha_j - \Theta|$ is not necessarily $\frac{\pi}{2}$. Then the response will be as if the edge were moving in a direction $\alpha_j \pm \frac{\pi}{2}$ with velocity $v\sin(\alpha_j - \Theta)$ (see Figure 3.3). So at a local level there is not enough

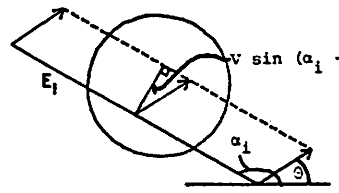


Figure 3.3. The Local Ambiguity of a Moving Edge. Suppose that edge E_i is moving with speed V in direction θ, and let α_i be the orientation of E_i. Then a movement detector in the circular area in the figure will only be able to deduce that the component of the velocity perpendicular to the edge is $V \sin(\alpha_i - \theta)$.

information to determine both v and θ, but only a relationship between them. By using interactions between neighbouring elements, the ambiguity may be resolved.

We will assume that a moving object has a polygonal boundary consisting of a set of edges E_i at orientations α_i relative to some arbitrary reference orientation. We will suppose that the object is moving with velocity v in direction θ. We will also suppose that the detectors are 'sufficiently' dense, and that each one responds maximally to edges moving in a certain direction. If at each small patch P_i the maximum response of all local detectors is computed, then this will be (at least approximately) equal to the component of the object's velocity in a direction perpendicular to the edge E_i crossing P_i. The actual velocity and direction of the object is not computable at this point.

Let the maximum response at P_i be R_i, so

$$R_i = v\sin(\alpha_i - \theta). \qquad (1)$$

We will assume that α_i and R_i are known locally at P_i. Although (1) also is insufficient for solution of v and θ, given the response R_j for edge E_j, where $\alpha_i \neq \alpha_j$, then v and θ may be determined as follows. For E_j,

$$R_j = v\sin(\alpha_j - \theta) . \qquad (2)$$

Solving (1) and (2) we get

$$v^2 = (R_i^2 + R_j^2)\,\mathrm{cosec}^2(\alpha_j - \alpha_i) - 2R_iR_j \cot(\alpha_j - \alpha_i)\,\mathrm{cosec}(\alpha_j - \alpha_i)$$

$$\tan\theta = \frac{R_i\sin\alpha_j - R_j\sin\alpha_i}{R_i\cos\alpha_j - R_j\cos\alpha_i} .$$

These are fairly complicated equations which may not easily be solved in biological systems, and besides, their solution is not well defined for $\alpha_i \approx \alpha_j$, especially with noisy data. We will therefore seek an alternative approach. We will allow the computational elements to interact with each other so that they arrive at a consensus via an iterative process.

Let x_i be a detector which responds maximally to edges at angle α_i. Suppose a moving object passes over the retina with velocity v and direction θ. Suppose x_i estimates the velocity and direction to be u_i and θ_i respectively from response r_i.

For two neighbouring detectors x_i and x_j,

$$r_i = v\sin(\alpha_i - \theta) \qquad (3)$$

$$r_j = v\sin(\alpha_j - \theta) \ .$$

For consistency,

$$r_i = u_i\sin(\alpha_i - \theta_i) \qquad (4)$$

$$r_j = u_j\sin(\alpha_j - \theta_j)$$

u_i, u_j, θ_i and θ_j are time-varying, but must always obey (2). Let us see how x_i and x_j may cooperate to achieve consistency, i.e.,

$$u_i = u_j$$

$$\theta_i = \theta_j \ .$$

If $u_i \neq u_j$, where $u_i > u_j$, say, then the following is done. We set u_j to equal u_i, and adjust θ_j to maintain the relation (4). So θ_j becomes

$$\alpha_j - \sin^{-1}(r_j/u_i) \ .$$

Now suppose $u_i = u_j$, but $\theta_i \neq \theta_j$. We have from (3) and (4)

$$\frac{r_i}{r_j} = \frac{\sin(\alpha_i - \theta)}{\sin(\alpha_j - \theta)} = \frac{\sin(\alpha_i - \theta_i)}{\sin(\alpha_j - \theta_j)} \ . \qquad (5)$$

Set $\delta_i = \theta_i - \theta$

$\delta_j = \theta_j - \theta$

$\beta_i = \alpha_i - \theta_i$

$\beta_j = \alpha_j - \theta_j$.

From (5)

$$\frac{\sin(\beta_i + \delta_i)}{\sin(\beta_j + \delta_j)} = \frac{\sin\beta_i}{\sin\beta_j}$$

Expanding,

$$\sin\beta_j\sin\beta_i\cos\delta_i + \sin\beta_j\cos\beta_i\sin\delta_i = \sin\beta_i\sin\beta_j\cos\delta_i + \sin\beta_i\cos\beta_j\sin\delta_j .$$

For small δ_i,δ_j we have

$$\cos\delta_i \approx \cos\delta_j \approx 1$$

$$\sin\delta_i \approx \delta_i$$

$$\sin\delta_j \approx \delta_j \ .$$

Hence

$$\delta_i/\delta_j \approx \frac{\tan\beta_i}{\tan\beta_j} = \Delta, \text{ say.}$$

We will use Δ to estimate ϕ. Consider Figure 3.4, which depicts a posi-

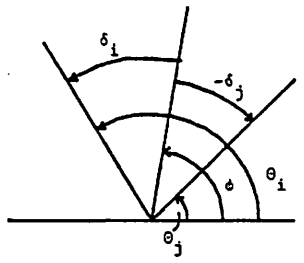


Figure 3.4. Relationship between angles used in cooperative computation of velocity (see text).

tive δ_i and a negative δ_j. It may be seen from the Figure that

$$\phi = \theta_j + \frac{-\delta_j}{\delta_i - \delta_j}(\theta_i - \theta_j)$$

$$= \frac{\delta_i\theta_j - \delta_j\theta_i}{\delta_i - \delta_j}$$

$$= \frac{\Delta\theta_j - \theta_i}{\Delta - 1} \quad .$$

θ_i and θ_j are now set to this value. u_i and u_j are then re-evaluated from

$$u_i = r_i \operatorname{cosec}(\alpha_i - \theta_i)$$

$$u_j = r_j \operatorname{cosec}(\alpha_j - \theta_j)$$

using the new values for θ_i and θ_j.

This procedure was tested for several polygonal figures. The updating process was applied to each pair of edges taken in series around the perimeter of the objects. Table 3-1 shows the results of presenting the system with an equilateral triangle moving at a constant speed of 10 units, moving in thirteen directions at 10° intervals (see Figure 3.5). Three detectors were simulated. The estimations of each detector after 10 iterations are given in the Table. The success of the method is evident.

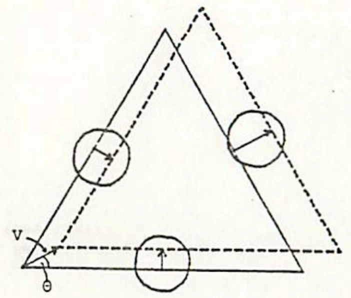


Figure 3.5. Motion of a Triangular Figure. Three detectors (in the circles) seek to determine the object's velocity. Locally, they can only compute the components of the velocity perpendicular to the sides. Through interaction, they can arrive at a consensus and compute the correct velocity (V, θ). The results of running a simulation for several values of θ are shown in Table 3.1.

Angle	Detector 1		Detector 2		Detector 3	
θ	Speed	Direction	Speed	Direction	Speed	Direction
0	10.00	0.00	10.00	0.00	10.00	0.00
10	10.34	9.67	9.98	9.30	10.01	9.98
20	10.02	19.95	10.00	20.00	10.00	20.00
30	10.00	30.00	10.00	30.00	9.85	30.00
40	10.00	40.00	10.00	40.00	10.00	40.00
50	10.00	50.00	10.00	50.00	10.00	50.00
60	10.00	60.00	10.00	60.00	10.00	60.00
70	10.00	69.93	10.00	70.00	10.00	70.00
80	10.00	80.00	10.00	80.00	10.00	80.00
90	10.00	90.00	10.00	90.00	10.00	90.00
100	10.00	100.01	10.00	100.00	10.00	100.00
110	10.00	110.01	10.00	109.97	10.00	110.00
120	10.00	120.00	10.00	120.00	10.00	120.00

Table 3.1. Results of a Motion Detection Experiment. Three interacting processors were used to detect the passage of the sides of an equilateral triangle as it moved in different directions. Each detector 'saw' the motion of one side, but could only determine the component of the object's velocity perpendicular to the side it saw. By interacting with the other processors, each under the assumption that they were all responding to the motion of the same object, each processor was able to compute the object's speed and direction. The processors' estimates after 10 iterations are listed here.

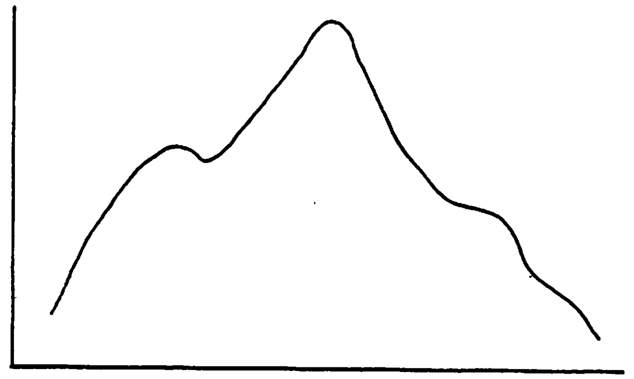
As far as may be determined, there is no direct evidence of this kind of cooperation in biological visual systems. It would be very interesting if an experiment could be devised where the response is recorded of a cortical cell which is the only one in its neighbourhood receiving visual stimulation. This response may be compared with that which occurs when neighbouring cells too get stimulation. In this manner, it might be possible to deduce if the neighbouring cells can influence the neuron to interpret the response from an edge at orientation a, say, to be from an edge at orientation b, a ≠ b. Of course, this argument presupposes that these cooperative processes take place in the cortex, but since we believe that they take place early in visual processing, the cortex is a reasonable place to look. It is possible, though, that the computation takes place in a cortico-geniculate interaction.

3.2.3 Correlation vs. Feature Extraction

Let us consider two approaches to motion analysis, given two successive frames of motion data.

Method 1. For each point in frame 1, compute its correlation with frame 2. The result may be histogrammed as a 2-d distribution; a typical 1-d slice is sketched in Figure 3.6a. The highest peak (usually) corresponds to the displacement between the frames.

Method 2. Extract features from each frame (such as spots of high/low intensity, edges, bars, points of high curvature). Now match these features as before. In general, the distribution will be much sharper and/or have fewer peaks, since there is a lot more information in each


3.6a

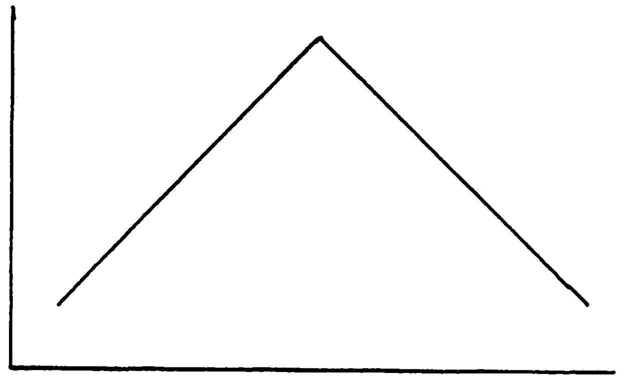

3.6b

Figure 3.6. Effects of Lateral Inhibition. Laterally inhibitive networks have the property of sharpening curves and eliminating all but the strongest peaks. Thus such a network when applied to the curve in 3.6a might produce that in 3.6b.

extracted feature-point (see Figure 3.6b).

Presumably, if one goes as far as extracting entire regions as features, then the correlation function may have only one peak, but to do this may be computationally infeasible (since the extraction of such regions may be the goal of the process).

Unless the whole scene has moved by the same displacement, the immediate correlation method will only work when based upon small regions. The question arises, how small is small? Moreover, if the match is being made on intensity level, then there are almost always ambiguous matches within a small displacement. If, for example there are 64 intensity levels, then within a window of size greater than 8 x 8 there is bound to be at least one intensity value represented more than once. Furthermore, the effects of noise will increase the ambiguity dramatically. With extracted features, however, if a good match can be found in a local neighbourhood, there is much less chance that this match is incorrect.

Now, it has been widely suggested that the visual system employs Fourier analysis (see, e.g. Campbell and Robson 1968; Cornsweet 1970; Blakemore and Campbell 1968), although this view is not held universally. While this may or may not be true, the following analysis of the use of feature extraction versus correlation uses the Fourier transform to reveal an interesting property of receptive field size. We will examine Methods 1 and 2 described above in the one-dimensional case.

Let $f(x)$ and $f'(x)$ be two consecutive frames. Then their cross-correlation is given by

$$a(e) = \int f(x)f'(x + e)dx .$$

It will be possible to sharpen the correlated images by effecting contrast enhancement through lateral inhibition, modelled by a linear filter $r(x)$. Thus

$$c(d) = \int a(e)r(d - e)de.$$

For example, the lateral inhibitory networks of Ellias and Grossberg (1975) and Grossberg (1976) would, when applied to the function in Figure 3.6b, produce an output like that in 3.6b.

We will see if we can achieve the same effect by applying a feature-extraction process (centre-surround) prior to the correlation. Suppose that the feature-extraction process is performed by convolution with a mask $h(x)$, so that

$$g(y) = \int f(x)h(y - x)dx$$

$$g'(y) = \int f'(x)h(y - x)dx.$$

Correlating these two processed images gives

$$b(d) = \int g(y)g'(y + d)dy.$$

We ask whether an $h(x)$ exists such that $b(d) = c(d)$, i.e. that the diagram of Figure 3.7 commutes. In the frequency domain, these equations become, where capital letters indicate Fourier transforms,

$$A(\omega) = F(\omega)F'(-\omega)$$

$$C(\omega) = A(\omega)R(\omega)$$

$$G(\omega) = F(\omega)H(\omega)$$

$$G'(\omega) = F'(\omega)H(\omega)$$

$$B(\omega) = G(\omega)G'(\omega).$$

Putting $C(\omega) = B(\omega)$ gives

$$R(\omega) = H(\omega)H(-\omega)$$

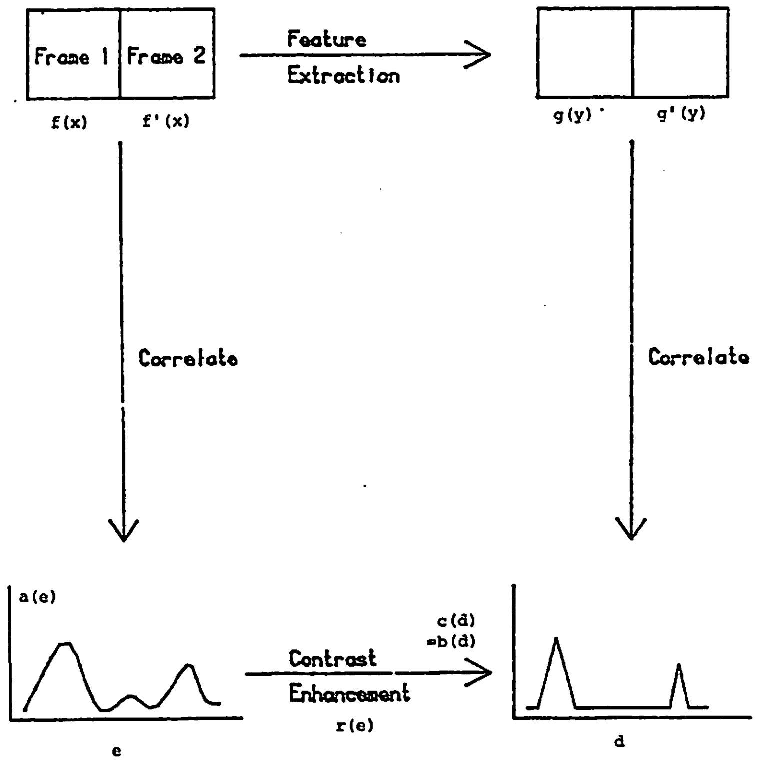


Figure 3.7. Two Algorithms for Velocity Detection. Given a pair of frames, it is desired to determine the gross motion that has taken place by the objects in them. Method 1 correlates the frames and sharpens the resultant curve. Method 2 extracts features from the images and correlates these features. In the text, the conditions under which these algorithms are identical are discussed. In this figure, one-dimensional data only is considered.

or

$$R(\omega) = H(\omega)H(\omega)$$

if $H(\omega)$ is even.

For many parameter settings, $h(x)$ will be the same shape as $r(x)$ but may be much wider. For example, if the upper curve in Figure 3.8 represents $h(x)$, then the lower curve would be $r(x)$. The vertical line-segments indicate minima. In this example, then the centre of the inhibitory region of $h(x)$ is about one and a half times as wide as that of $r(x)$. That is, performing the feature-extraction prior to correlation, rather than contrast enhancement afterwards, requires a much larger receptive field size.

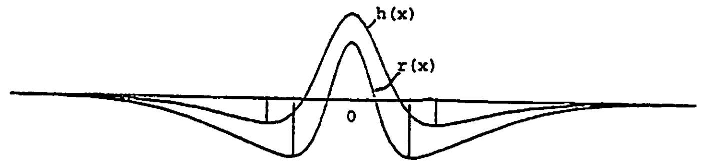


Figure 3.8. Effect of Using Feature Extraction. A process using feature extraction requires a wider receptive field (h(x)) than one using correlation then contrast enhancement (r(x)).

CHAPTER IV

ANALYSIS OF A MOTION-DETECTING SYSTEM

4.1 Introduction

There are very many ways in which a system may be constructed to perform motion-analysis. We have seen in Chapter I that a large number of different algorithms and methodologies are being tested, with greater or lesser success. To a large extent, though, these systems are all 'ad hoc': the authors often fail to provide justification for their methods, other than the fact that they work reasonably well in the limited domains in which they are tried out.

It is by no means clear in most cases whether these systems can be extended to work outside their proven domains. Now, presumably one of the goals of scene-analysis is to produce a 'complete' machine vision system - one which can address the dynamic problems of detecting and tracking motion under both high and low velocities (quite different problems) as well as the more static considerations of object and scene identification. It is unlikely that this can be achieved unless careful consideration is given to the necessary extensions and integrations of the separate subsystems as they are being designed.

We are not about to describe such a complete, integrated vision system - that is outside the scope of the work presented here. We are, however, going to lay some foundations for the design of a vision system, and to construct, in the next chapter, part of a motion-analysis system based upon these principles.

Overview

We begin the chapter by introducing the concept of 'Optic Flow'. Optic Flow is a central concept in this thesis, and since the low-level representation of perceived motion which we will use in Chapter V may be identified with it, we will look at it here in some detail. We will see how it may be generated from dynamic visual input, and what information it conveys. We will claim that an optic flow field is a very convenient but 'raw' representation of motion in the visual world. The effect of the visual system 'filling-in' those areas deprived of moving visual stimulation will also be discussed.

In the next section we develop a set of edge-operators for the extraction of feature-points from the moving-image data. These operators may be thought of as analogues of biological feature-detectors. The feature-points themselves will be the objects of the Stimulus-Matching process described in Chapter V.

Properties of co-moving objects will next be examined in the light of moving-dot experiments. We will see how the definition of 'common vector' may be extended to cover the general case, and how along with the 'field velocity' concept it may be used to explain various perceptual effects. We will also demonstrate its use in extracting the rotational and translational components of a moving body.

4.2 Optic Flow

Gibson (1966) has stated that the dynamic visual environment is full of information just waiting to be 'picked up'. We agree that there

is a great deal of information there, but feel that at least some processing is required to extract it, before the organism (or machine) can make use of it.

Suppose that we have available two consecutive frames of a movie sequence. If we superimpose these frames, identifying points common to each and joining them up with small arrows, we generate a graphic description of a vector velocity field depicting what Gibson has called the 'Optic Flow'. Optic Flow is one of the most important kinds of visual information. It may be represented as a retinotopic array, each point containing a vector signifying the projected motion from the 3-d world onto the corresponding part of the retina. We will examine in this section those properties of the motion that may be deduced from the flow, and how a segmentation of the visual world may be performed directly from the flow field.

The information inherent in flow is present when the observer is at rest and objects move about him, when only the observer is moving, or in any combination. The only difference is that when the observer moves, there is non-zero optic flow associated with the ground and stationary objects, unlike when he is still. In either case, the optic flow on a moving object is substantially different from that on the background. Indeed, when the observer moves, the optic flow is constant on surfaces at equal distance from him; the optic flow on a surface extending in depth would change smoothly over the surface.

Segmentation from the Optic Flow

The optic flow produced by the relative motion of an object and observer will vary smoothly across the surface of the object. This flow will in general be different from the flow on neighbouring objects in the image. Therefore, by finding discontinuities in the flow, a segmentation of the image may be formed.

In order to find boundaries of objects in space, Nakayama and Loomis (1974) proposed 'convexity detectors' which consisted of centre-surround receptive fields sensitive to different velocities. Their purpose is simply revealed to be to differentiate the generated optic flow. The discontinuities so found in the velocity field may be interpreted as a segmentation of the scene with respect to depth. Intensity gradients on the surfaces of objects will not show up in this representation, since the velocity field associated with an object varies continuously over its surface. However, if an object is moving with respect to its background, or if due to the observer's motion it appears to move with respect to the background, then at the object's boundaries there will be discontinuities in the velocity field. The points on either side of the boundaries found this way will lie on objects at different depths from the observer.

It should be stressed that this approach is in distinction from spatial brightness differentiation, commonly used in static scene analysis, which looks solely at contrast in the image to perform its segmentation. In differentiating the optic flow, there is no danger of confusing the internal surface structure of objects (texture, lighting effects, different colourations) with the object's boundaries, a common problem in scene analysis.

4.2.1 Properties of Optic Flow

We will describe the kinds of flow that may be observed in a real-world situation from the viewpoint of a moving robot in an arbitrary environment. (In our later simulations, the robot will move through an artificial forest.) The robot will be able to deduce certain properties of the environment from the perceived flow, and thus be able to avoid collisions and navigate to his intended goal. We are not interested here in the problems of motor control.

We will centre our coordinate system at the robot. It may move only in the (horizontal) x-z plane. Its line-of-sight is also horizontal, but is independent of his direction of motion. The robot will have a planar retina parallel to the x-y plane, which will be indexed by the coordinates ξ and η. Using perspective projection, we have

$$\xi = \frac{ax}{z} \qquad (1)$$

$$\eta = \frac{by}{z}$$

where a and b are scale constants (see Figure 4.1), and may for convenience be dropped if desired. It will be noted that to avoid an inverted image we are using a 'virtual retina'.

Before we proceed with the analysis, we should note that Lee (1974) examined the case of a moving observer with an infinite cylindrical retina. The particular choice of retina employed will simplify some of the

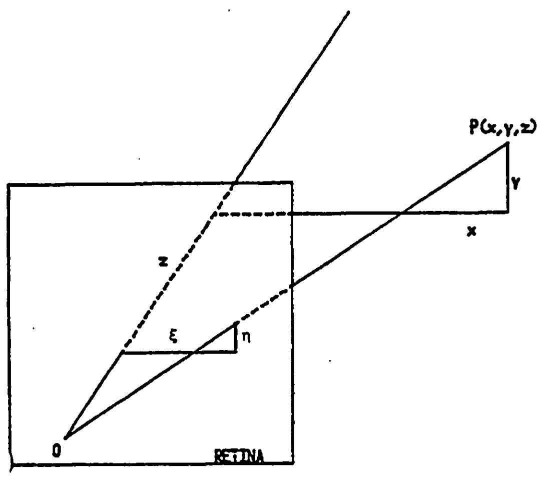


Figure 4.1. Use of a Planar Retina. O is the origin of coordinates at the observer's eye. The retina is parallel to the x-y plane. In order to avoid an inverted image, a 'virtual' retina in front of O is used. Point P(x,y,z) in space projects to point (ξ, η) on the retina.

equations and relations derived, but may complicate others. Some of our results are equivalent to his, but are expressed in terms of our coordinate system. Our choice of retina is more natural, and its attendant optic flow exhibits certain rather attractive properties (as we will see).

The projection of Figure 4.1 only applies when the robot is looking along the z-axis. To avoid complicated expressions when the direction of gaze is different from this, use will be made of the fact that optic flow is a measure of the relative velocity of observer and environment. Thus the same flow will result from the robot moving with velocity v, or with the robot and the environment as a whole moving with velocity -v. The effect of a shift in the robot's direction of motion is accomplished by a rotation of the environment, which since the robot is at the origin, is simply a rotation of the coordinate system.

We can therefore keep the robot fixed at the origin of coordinates and looking in a constant direction (the z-axis), and move points in space to simulate its own motion. It is clear that we can also take care of those cases where points in the environment have their own intrinsic motion.

Linear Motion: No Acceleration

Let P(x, y, z) be an arbitrary point in the robot's field of view. Suppose that the path of P relative to the robot is given by the straight line

$$\begin{aligned} z &= mx + c \\ y &= y_0 \end{aligned} \qquad (2)$$

From (1),

$$z = \frac{by}{\eta} ,$$

$$x = \frac{\xi by}{a\eta} .$$

Substituting for x, y and z we get, for the projection of this line on the retina,

$$mby_0(\xi - \frac{a}{m}) = -ac\eta \qquad (3)$$

which is again a straight line. Thus,

Result.1. Flow lines deriving from linear motion are straight.

Now suppose that the path of P is at an angle $\alpha (0 \leq \alpha < \pi)$ to the x-axis (remembering that the robot always looks along the z-axis) and P', the projection of P on the x-z plane, comes to a distance d of the robot at its 'closest encounter'. (The point here is that if P is on a vertically oriented object, P' will be the point on the object that comes closest to the robot.) We find it more convenient to parametrize the linear motion with d and α instead of m and c. Since

$$m = \tan\alpha$$

and $$c = -d\sec\alpha$$

(2) can be re-written

$$z = x\tan\alpha - d\sec\alpha \qquad (4)$$

(see Figure 4.2), so that (3) becomes, again substituting x and z from (1),

$$\eta = \frac{by}{d} \sin\alpha(\xi - a\cot\alpha) . \qquad (5)$$

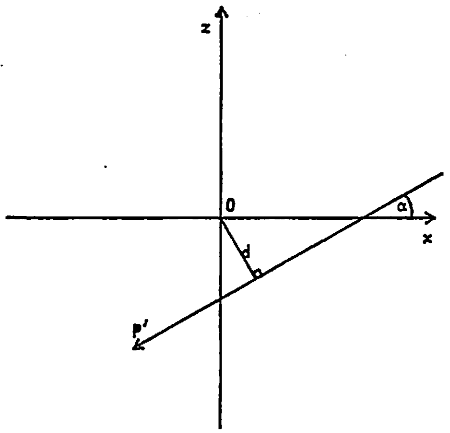


Figure 4.2. Path of Point in x-z Plane. The relative linear motion between observer at O and point P' is given parametrically by the line $z = x \tan\alpha - d \sec\alpha$.

If the environment is (in reality) stationary, then relative to the robot, all points on it will move on lines of the form (4), with α the same for each. Thus all lines of the form (5) pass through the point $(a\cot\alpha, 0)$. This is the projection of the point-at-infinity to which the robot is heading. Because flow-lines seem to emanate from this point, it is known as the Focus of Expansion (FOE). These lines through the FOE will have slope

$$\frac{d\eta}{d\xi} = \frac{by}{ad}\sin\alpha. \qquad (6)$$

Any object moving with respect to the environment will produce flow of the form (5), but with a different value of α from stationary points, so will have its own FOE. This may be summarized as:

Result 2. Under linear motion, flow lines are straight and emanate from the single focus of expansion, whose location depends on the direction of relative velocity.

Now, (6) has the following implications:

Result 3. If the robot is looking in a direction perpendicular to his motion ($\alpha = 0$), then all flow lines are parallel to the x-axis, and are (from (5)) of the form $\eta = by/d$.

Result 4. If an object is on a direct collision course with the robot ($d = 0$), then the corresponding flow lines are vertical, and are of the form $\xi = a\cot\alpha$.

These results are represented in Figure 4.3.

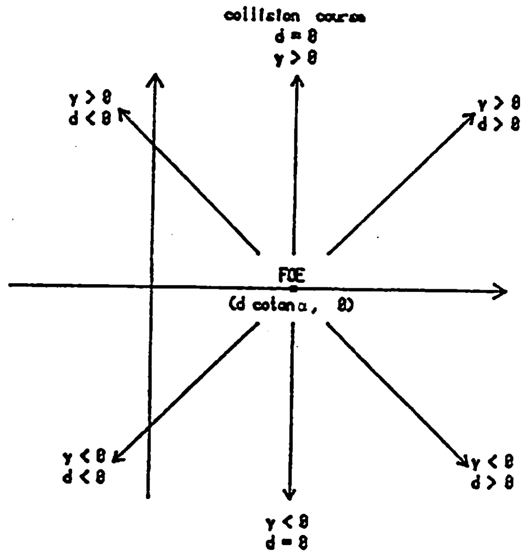


Figure 4.3. Relationship between y, d and direction of optic flow.

Collision Avoidance

Let the robot have a width of 2k. Then an object whose course is plotted by (4) will not collide with the robot if

$$d > k. \tag{7}$$

Suppose that the object's velocity in the x-z plane is v. Then

$$\begin{aligned} x' &= v\cos\alpha \\ z' &= v\sin\alpha. \end{aligned} \tag{8}$$

[Note on sign convention. (8) implies that for positive v, and $0 \le \alpha < \pi$, z' will be positive, meaning that the object is receding from the robot. In most cases of interest to us, the objects visible to the robot will be moving towards it, which would require a negative v according to our convention. This should be borne in mind in the equations that follow.]

From (1) we get

$$\begin{aligned} \xi &= \frac{ax}{z} \; \frac{-axz'}{z^2} \\ \eta &= \frac{-byz'}{z^2}. \end{aligned} \tag{9}$$

For simplicity, now consider the case when the robot is looking where it is going, that is, when $\alpha = \pi/2$. In this case, (4) reduces to

$$x = d \tag{10}$$

with

$$x' = 0$$

$$z' = v .$$

Substituting in (9), we get

$$\xi' = \frac{-adv}{z^2}$$

$$\eta' = \frac{-byv}{z^2}.$$

But since, from (1),

$$\xi = ad/z$$

we get

$$\begin{aligned} \xi' &= \frac{-\xi^2 v}{da} \\ \eta' &= \frac{-by\xi^2 v}{a^2 d^2} \end{aligned} \tag{11}$$

Since we also have from (1) and (10)

$$d = b\xi y/a\eta$$

we have finally

$$\xi' = \frac{-\xi^2 v}{da} \tag{12}$$

$$\eta' = \frac{-\eta^2 v}{by} \tag{13}$$

For no collision, (7) tells us $d > k$, so (12) gives

$$\xi' < \frac{\xi^2 v}{ak}. \tag{14}$$

Let us see what this means. Consider a point P whose projection P' passes a distance d from the robot to its right ($\alpha = 0$ in Figure 4.2). From (5), its flow is

$$\eta = by\xi/ad$$

and since v is negative, so is $\dot{z}'$, so, from (9) or (11), ξ' and η' are positive. Also, from (5), the FOE is at the origin on the retina. The flow, at point (ξ, η) is given by (ξ', η'), where ξ' and η' are computed from (9).

We know that ξ' and $(-v)$ are both positive. (14) tells us then that if ξ' is sufficiently small in magnitude, then no collision will occur. The rationale here is that many points in space project to a given point (ξ, η) on the retina. All of these subtend the same angle from the robot's direction of gaze. Those with small associated flow will be those furthest away, and so will be those which, when they eventually pass, do so with the widest berth.

For those objects that pass to the left of the robot, d is negative in (4) and Figure 4.2. Thus from (12), ξ' is negative, and the collision avoidance condition (7) becomes

$$|d| < k$$

so that (14) becomes, in general,

$$|\xi'| < \left|\frac{\xi^2 v}{ak}\right| .$$

Thus

Result 5. At each point on the retina, a simple test on the optic flow (15) is all that is needed to see if a collision will occur.

For completeness we may note that if there are hazards in the vertical direction, like overhanging trees, then if the robot reaches to a height h above its 'eye', then (13) gives

$$\eta' < \frac{-\eta^2 v}{bh}$$

for collision avoidance.

Segmentation

Consider the case where the robot moves along the z-axis with velocity v in a stationary environment. For any point P(x, y, z) in the environment, we have from (1) and (9)

$$\xi' = \frac{-\xi v}{z} \qquad (17)$$

$$\eta' = \frac{-\eta v}{z}$$

The retinal velocity uret is then

$$\text{uret} = \frac{v}{z}\sqrt{\xi^2 + \eta^2}$$

$$= \frac{rv}{z}$$

where r is the eccentricity of P's retinal projection. Thus we clearly see that for neighbouring points on the retina, flow velocity is inversely proportional to depth. Segmentation may therefore be performed by finding discontinuities in the optic flow.

Inference from Optic Flow

Equations (15) and (16) give conditions for collision avoidance. To do this correctly, the robot must know the ratios v/ak and v/bh. However, as v is likely to vary considerably from time to time, while a, k, b and h are unlikely to do so, it would seem plausible that the robot learn the products ak and bh adaptively, and v from vestibular or other proprioceptive information.

From (5) we find that the FOE is at (acotα, 0) on the retina. We may infer:

Result 6. Knowing a, the robot can deduce the angle between its directions of gaze and motion.

Alternatively, he may know this from proprioceptive information, but will not know proprioceptively the angle with which a moving object is coming at it.

The slope of the flow lines give information of the object's height y and 'close-encounter' distance d by means of (6). It is only possible to deduce from this the ratio y/d, though. That is, two objects, one of which is twice as tall and will pass twice as far away as the other will induce identical flow lines. However, these flow lines will be traversed at different rates.

The components of the velocity of a point along a flow line are given by (12) and (13). Therefore we have

Result 7. Knowing the velocity of a point along a flow line, the robot can deduce its height y and passing distance d independently.

Rotation

Now suppose that the robot rotates on the spot, or alternatively, that P(x, y, z) moves in a circle around it. Recall equation (1):

$$\xi = ax/z$$
$$\eta = by/z.$$

If the polar coordinates of P are (R, θ), we have

$$x = R \sin \theta \qquad (18)$$
$$z = R \cos \theta \ .$$

Combining (1) and (18) we get

$$\xi = a \tan \theta$$
$$\eta = by \sec \theta / R$$

from which we get

$$\eta^2 = \frac{b^2 y^2}{R^2} \left(1 + \frac{\xi^2}{a^2}\right) \qquad (19)$$

or

$$\frac{\eta^2}{B^2} - \frac{\xi^2}{A^2} = 1 \qquad (20)$$

where

$$A = a$$
$$B = by/R.$$

Equation (20) represents a hyperbola, or rather a family of hyperbolas parametrized by B. All points in space with the same value for R/y will give rise to the same hyperbola. For a given hyperbola, its vertices are at (0, ±B) (see Figure 4.4). We have

Result 8. For any point (ξ_0, η_0) on the retina, only one hyperbola of the family (20) passes through it. This hyperbola is given by

$$\frac{\eta_0^2}{B_0^2} - \frac{\xi_0^2}{A^2} = 1. \qquad (21)$$

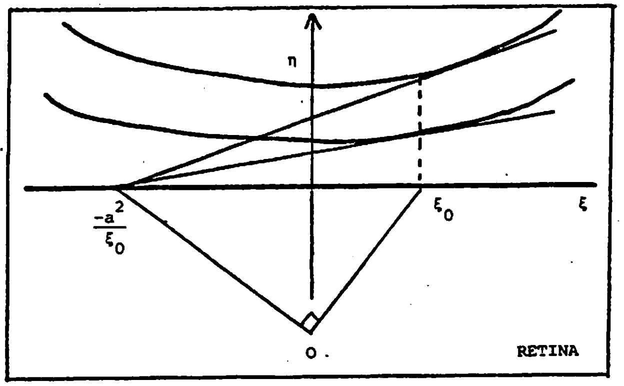


Figure 4.4. Hyperbolae Induced on Retina by Rotational Motion. During pure rotational motion, all points with abscissa ξ_0 have instantaneous FOE at $(-a^2/\xi_0, 0)$. The angle between any point in the x-z plane and its FOE is 90 degrees.

Differentiating (20) we get

$$\frac{\eta\eta'}{B^2} = \frac{\xi\xi'}{A^2}$$

so the gradient is

$$\frac{d\eta}{d\xi} = \frac{\xi B^2}{\eta A^2}$$

The tangent at (ξ_0, η_0) is thus

$$(\eta - \eta_0) = \frac{\xi_0}{\eta_0} \frac{B_0^2}{A^2} (\xi - \xi_0).$$

or, substituting B_0^2 from (21), we get after rearranging,

$$\eta = \frac{\eta_0\xi_0}{a^2+\xi_0^2} (\xi + \frac{a^2}{\xi_0}) .$$

We now have

Result 9. The tangents to all points on the hyperbolas with abscissa ξ_0 pass through the point $(-a^2/\xi_0, 0)$ (see Figure 4.4).

Suppose that the robot turns with constant angular velocity ω.

From (17) we have

$$x' = R\omega \cos \theta \qquad (22)$$

$$z' = -R\omega \sin \theta$$

Substituting (17) and (22) in (9) we get

$$\xi' = a\omega \sec^2 \theta \qquad (23)$$

$$\eta' = \frac{-by}{R} \omega \sec \theta \tan \theta$$

or using our original retinal coordinate system

$$\xi' = \frac{\omega}{a}(1 + \xi^2)$$

$$\eta' = \frac{\omega\xi\eta}{a} \quad .$$

Now consider equation (19). From the flow-line itself, the robot can only deduce the ratio y/R. Therefore we have

Result 10. Unlike the translation case, the velocity of the flow (23) does not give y or R separately.

Finally, consider any point on the line $\xi = \xi_0$ in Figure 4.4. It corresponds to a point in space in a direction θ from the robot's direction of gaze, where from (1)

$$\tan \theta_1 = \xi_0/a.$$

The point's instantaneous FOE is the point $(-a^2/\xi_0, 0)$ which corresponds to a direction θ_2, where from (1) again

$$\tan \theta_2 = -a/\xi_0.$$

The angle between these directions has tangent

$$\frac{\xi_0/a + a/\xi_0}{1 - \xi_0 a/a\xi_0}$$

which is infinite, so $\theta_1 - \theta_2$ must be a right angle. Our final result of this analysis is

Result 11. The FOE for any point in the visual field during pure rotation is always the same point, the two points subtending a right angle at the observer. (See Figure 4.4.)

Results of a Simulation

In summary, we present some outputs from a simulated robot system. The robot is placed in the middle of an artificial forest populated by trees consisting solely of vertical trunks. The trees are randomly placed and have varying radii. On the trunks are simulated texture markings consisting of short vertical lines. For computational reasons, it is not possible to generate a sufficient number and variety of texture markings to give any suggestion of the richness of optical texture on real trees.

The operator of the system can ask for any of three 'views'. He may ask for a 'bird's eye view' of the forest, to determine the location of the robot relative to the trees. He may ask for the view the robot has of the forest at any instant, or the optic flow generated as the robot takes a step. [N.B. In this simulation, the stimulus-matching problem is trivial, since each point in the simulated environment is uniquely labelled.] Further details of the system are given in Appendix II.

A typical bird's eye view is shown in Figure 4.5. The corresponding view in front of the robot is given in Figure 4.6, and the flow generated as the robot takes a step in Figure 4.7. Notice the common focus of expansion of all the flow-lines. The robot may now be turned to face (and move towards) the tree just to the right of straight-ahead; the resultant flow is shown in Figure 4.8.

If we make a tree move, as in Figure 4.9, we see that it has its own focus of expansion. If this tree is made to move on collision-

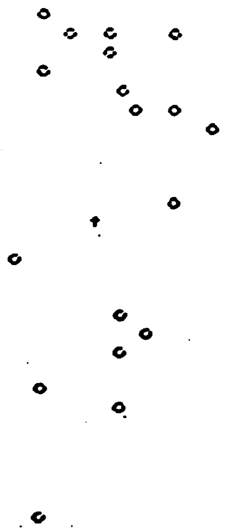


Figure 4.5. Bird's-eye-view of Forest (1). Although appearing uniform here, the trees (represented by circles) have randomly distributed radii. The arrow indicates the robot.

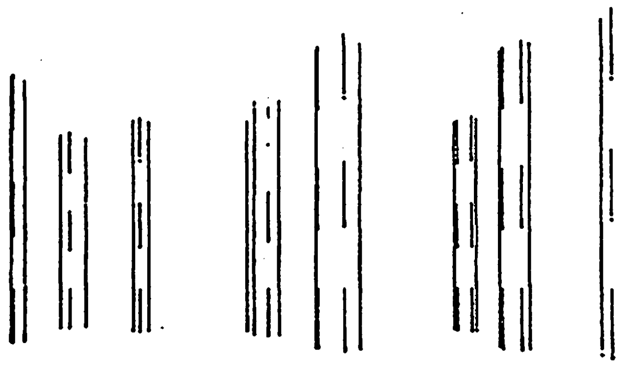


Figure 4.6. Static View of Forest. The trunks of the trees seen in an approximately 120° wide angle in front of the robot are shown here. Dashed lines indicate the simulated texture markings.

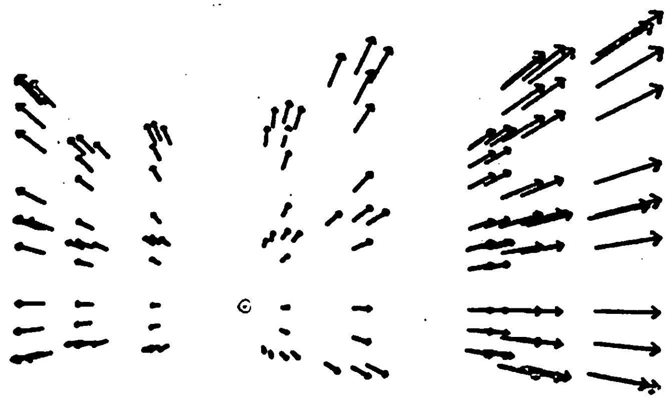


Figure 4.7. Optic Flow (1). Flow is generated from the end-points of the tree-boundaries and texture edges. Notice that the extrapolations of all flow lines will pass through the FOE (circled).

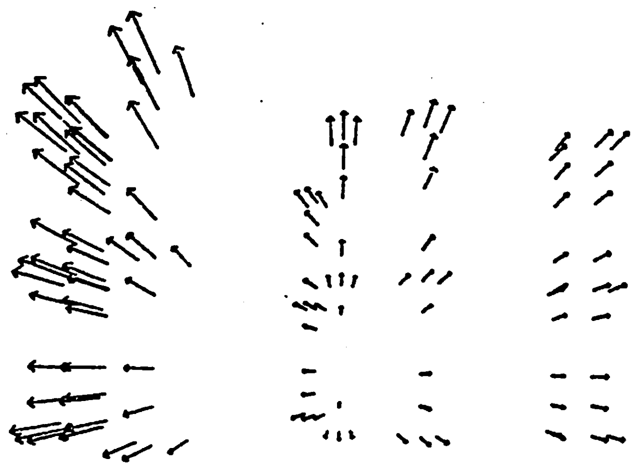


Figure 4.8. Optic Flow (2). Flow generated as the robot heads towards the tree just to the right of the FOE in Figure 4.7. The flow-lines on this tree are now vertical (and near-vertical).

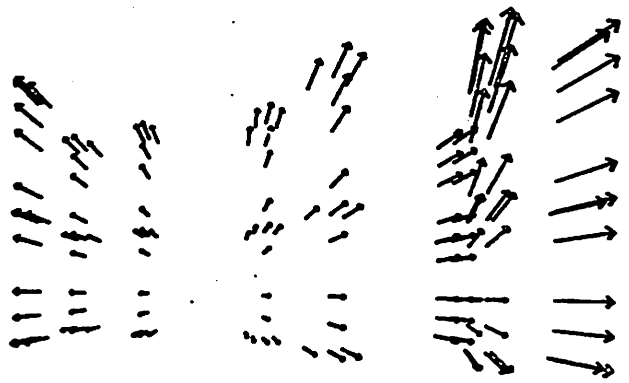


Figure 4.9. Optic Flow (3). Flow generated as a tree moves simultaneously with the robot. The moving tree has a different FOE from the other (stationary) trees.

course with the robot, then its flow lines will be (near) vertical (see Figure 4.10). Note that being on collision course with an object does not mean moving towards the instantaneous position of the object, but rather, the relative velocity is in the direction of the object (see the bird's eye view in Figure 4.11). Finally, if the robot turns on the spot, environmental points describe hyperbolas on the retina (see Figure 4.12).

4.2.2 Computing Optic Flow

We have seen that optic flow represents the passage of points in the scene during time. The computation of this flow consequently depends on successfully matching up points from one frame to the next (the Stimulus Matching problem).

A word should be said here about 'velocity detectors'. It has been hypothesized that one of the functions of the retina is to signal directly the speed of edges, spots, etc. moving across it. Were this to be the case, the optic flow could be directly 'read out' - indeed, some authors, e.g. Clocksin (1978) suppose that this is possible. However, biological velocity detectors have not been convincingly demonstrated. That evidence which does exist, however, does not provide us with velocity detectors with nearly sufficient resolution to produce the necessary information directly. A discussion of how, in such conditions much more accurate information may be achieved by cooperative processes was presented in Chapter III. In fact, as far as their implementation in machine vision systems is concerned, their existence would merely push the

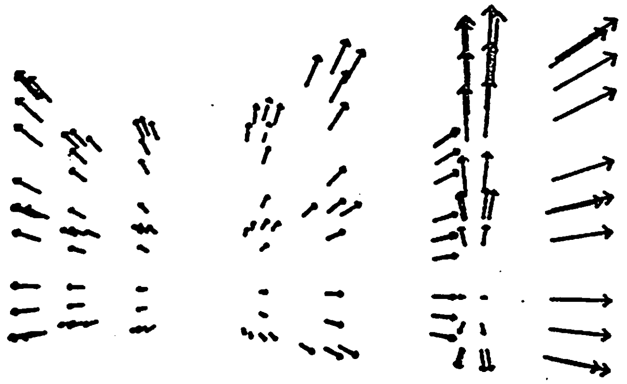


Figure 4.10. Optic Flow (4). The tree that moved in Figure 4.9 is now moving on collision course with the robot. Even though it is not directly in front of the robot, the flow lines on the tree are vertical.

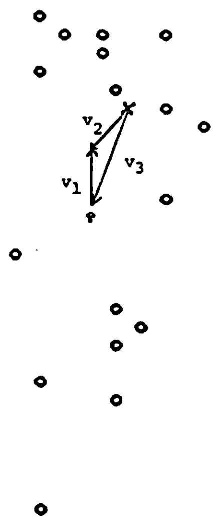


Figure 4.11. Bird's-eye-view of Forest (2). The moving tree indicated by an 'X' is on a collision course with the robot. The individual velocities of the robot (V_1) and the tree (V_2) are marked. Their relative velocity (V_3) is on a line joining them, indicating that they will collide.

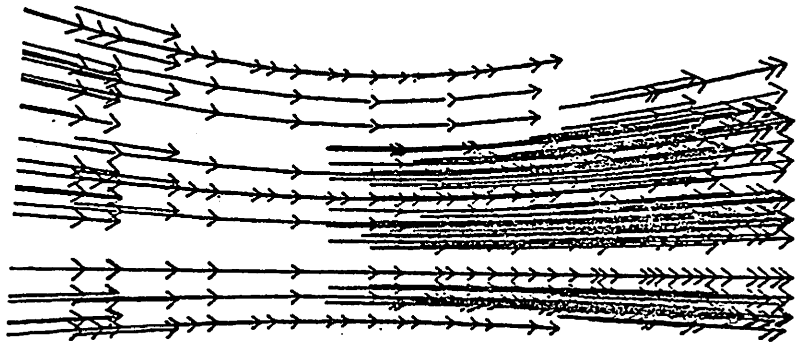


Figure 4.12. Optic Flow (5). Flow generated by rotation is in the form of vertical-axis hyperbolas. In this instance, the robot has turned through 100° anticlockwise in 10° increments. The heavy flow on the right is due to the many trees 'north' of the robot (see Figure 4.5) passing out of its field of view. The flow on the left is due to the trees 'west' and 'south-west' of the robot coming into view.

problem down one level, the question now becoming "how do the velocity detectors work?", the solution to which returns us to the Stimulus Matching problem!

Consistency Conditions

Although one particular method of computing optic flow will be the subject of the next Chapter, some general comments about this problem will be made now. The underlying structure of the physical world is manifested in a number of so-called 'consistency conditions' which apply to the optic flow. These conditions place restrictions on the optic flow in such a way that solving the Stimulus Matching problem is possible in a computationally feasible way.

We will list six consistency conditions and then describe each of them briefly.

1. Feature Compatibility.
2. Boundedness of Distance Travelled.
3. Path Consistency.
4. Neighbourhood Match.
5. Common Motion.
6. Environmental Model.

1. Feature Compatibility. The features associated with corresponding points in two frames must be similar. The transformation between the features must be a possible consequence of motion during this time-step (and the digitization process).

2. Boundedness of Distance Travelled. Assuming a bound on real-world velocities puts a bound on the magnitude of optic flow vectors

(possibly as a function of eccentricity). Thus matches will be found 'nearby'.

3. Path Consistency. Under the assumption that jerky movements happen relatively rarely, then the motion of a given feature-point between frames N and N+1 will be the same (to a first-order approximation) as that between frames N-1 and N.

4. Neighbourhood Match. The neighbourhood of a point may be described as the geometric arrangement of other nearby feature points around it. This neighbourhood will tend to be preserved from one frame to the next.

5. Common Motion. Nearby points on the same object will move with the same (or similar) velocities.

6. Environmental Model. As we saw earlier, the type of motion undergone places constraints on the optic flow produced. For example, translatory motion in a static environment produces linear flow emanating from the FOE.

We will see in Chapter V how some of these consistency conditions may be embodied in a relaxation process used to generate the flow. It might be noted here that it is the failure of these consistency conditions which gives a good indication of the occurrence of occlusion and disocclusion, and of the location of boundaries.

4.2.3 Completion of the Optic Flow Field

We have discussed the generation of optic flow by matching up extracted feature points from one frame to the next. The feature extrac-

tion process selected these points due to their 'special' nature; however a (possibly zero) velocity may be associated with every point in the visual field, so every point should have an associated optic flow element. Of course, the density of the flow field is limited by the resolution of the visual process. Assuming, though, that feature points are not found 'everywhere', the problem remains of filling in the remainder of the field. We will describe two possible approaches, one of which will be carried out in Chapter V.

A popular concept in AI is that of multiple levels of representation. An analysis of low-level data (such as lines, dots, corners) can cause invocations of higher-level entities (such as objects) to 'cover' the low-level ones. Hypotheses of higher-level constructs can cause an analysis of lower-level representations for supporting data. There can be many levels altogether in a system; HEARSAY-II (Erman and Lesser 1975) and VISIONS (Hanson and Riseman 1978b) for example, both use several.

One of the main contributions of feedback from higher levels is that filling-in can occur. After recognition of the shape (or possibly even the identity) of an object the optic flow may be completed within the object's boundaries. These higher levels of representation will not be used in the work described here, so we will seek an alternative low-level approach.

The continuity of some feature f over a region can be described by Laplace's equation $\nabla^2 f = 0$, or its discrete counterpart, over interior parts of the region. Weisstein (Weisstein and Maguire 1978; Weisstein et al. 1977) has demonstrated that humans can perceive 'phantom' patterns; that is, they can fill in missing portions of patterns under certain conditions. The patterns experimented with were all periodic, as are some of the solutions to Laplace's equation. A typical pattern is presented in Figure 4.13a.

Now, it is just the continuity of the velocity field over an object which is expressed by the consistency condition 5 (Common Motion). As we will see in Chapter V, one component of our system will be a process which seeks to generate optic flow for which Laplace's equation holds. In so doing, this process will be able to fill-in the optic flow in those areas of the visual field lacking 'significant' features.

A Proposed Experiment

It was mentioned above that Weisstein's experiments were performed with simple periodic patterns. It would be interesting to see if the class of patterns amenable to the 'phantom contour' phenomenon included those not obeying Laplace's equation, for example identical with the top but not completely out of phase with it. This experimental paradigm, or extensions of it, may be very useful in exposing the brain's interpolation mechanisms.

4.2.4 Decomposition of Flow Field into Components

In Chapter V we will see how the optic flow may be generated from the feature points on moving objects. The movements may be any combination of translation and rotation in space. For purposes of approach/avoidance, the translatory component of a moving object may be more im-

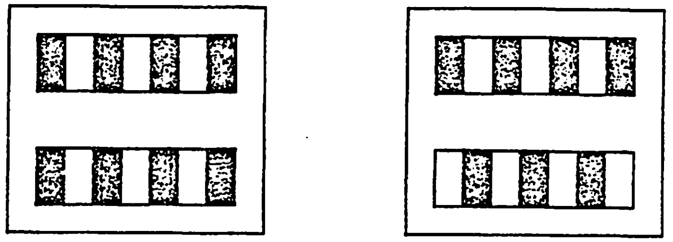


Figure 4.13. 'Phantom-Contour' Experimental Patterns. The gap in the centre of Figure 4.13a was 'filled-in' in Weisstein's phantom-contour experiments. Note the regularity of the pattern. The pattern in Figure 4.13b differs in that it does not have vertical uniformity. It would cast some light on the 'filling-in' process to see if this pattern is also subject to phantom contours.

portant to the observer than the combined motion. We will show below how an arbitrary (but assumed smooth) vector velocity field may be resolved into translation and rotation components in a local, parallel, distributed manner.

Suppose that there are N vectors v_i, $i = 1,\ldots,N$ in the flow field. Let

$$v_i = s_i + r_i , \qquad (1)$$

where the s_i are the translation components, and the r_i the rotation components. We will wish to find, via a relaxation process, s_i and r_i such that

$$s_i = s_j \quad \text{for all } i, j$$

$$\Sigma r_i = 0.$$

In this manner, the translation component will be the common vector discussed in Section 4.4.1 below.

For purposes of analysis, we will imagine that the s_i and r_i are functions of continuous time, and so can be modelled by differential equations. In the computer simulations that follow, though, they will be updated during several iterations of a relaxation process. The s_i and r_i will be initialized (see later) and the relations (1) will be preserved throughout.

The uniformity of the desired translation component can be expressed by the requirement that each translation component tends toward the average of its neighbours, thus

$$\frac{ds_i}{dt} = k \left[\sum_j s_j - Ns_i \right] . \qquad (2)$$

The desire that the rotation vectors sum to zero can be expressed by making each component adjust itself in a direction to make this true, thus

$$\frac{dr_i}{dt} = -k' \sum_j r_j . \qquad (3)$$

Let us put $\Sigma v_i = V$. Then using (1), (3) becomes

$$\frac{-ds_i}{dt} = k' \left[\sum_j s_j - V \right] . \qquad (4)$$

We may combine (2) and (4) to get, for weights λ and μ

$$\begin{aligned} \frac{ds_i}{dt} &= \lambda\left[\sum_j s_j - Ns_i \right] + \mu\left[-\sum_j s_j + V \right] \\ &= (\lambda - \mu) \sum_j s_j - \lambda N s_i + \mu V. \end{aligned} \qquad (5)$$

Summing (5) over i, we get

$$\begin{aligned} \frac{d\,\Sigma s_j}{dt} &= N(\lambda - \mu)\, \Sigma s_j - \lambda N\, \Sigma s_j + \mu NV \\ &= -N\mu\, \Sigma s_j + \mu NV. \end{aligned}$$

Hence

$$\Sigma s_j = Ae^{-N\mu t} + V. \qquad (6)$$

Thus in the limit, Σs_j tends to V, which is equivalent to Σr_j tending to zero, as desired. If we initially set $r_i(0) = v_i$, $s_i(0) = 0$, for all i, then the constant A in (6) equals -V, so we have

$$\Sigma s_j = V(1 - e^{-N\mu t})$$

$$\Sigma r_j = Ve^{-N\mu t} .$$

This is a straightforward case of exponential decay. However, we have not yet examined the effect on the individual values r_i and s_i. Substituting (7) in (5), we get

$$\frac{ds_i}{dt} = -\lambda N s_i + \mu V + V(\lambda - \mu)(1 - e^{-N\mu t}). \tag{8}$$

Clearly, since (8) represents an uncoupled system of first-order linear equations, the equilibrium point does not depend directly on $s_i(0)$. However, due to the forcing term $\mu V + V(\lambda + \mu)(1 - e^{-N\mu t})$, the equilibrium point will depend on V, which is a function of $s_i(0)$, so the initial conditions will determine the outcome.

With initial conditions $s_i(0) = 0$, (8) has solution

$$s_i(t) = \frac{V}{N}(1 - e^{-N\mu t}) \tag{9}$$

Thus we see that the individual values s_i tend uniformly to V/N, the common velocity, again by exponential decay.

The above analysis shows that if the summations in (3) and (4) are carried out over the entire field, the estimated translation and rotation components proceed monotonically to the actual values. We present below results of computer simulations where each element in the flow field only interacts with its immediate neighbours. Thus if there are n_i points in the neighbourhood N_i of point i, the updating equation for point i becomes, from (5)

$$s_i(t+1) = s_i(t) + (\lambda - \mu)\sum_{j\epsilon N_i} s_j - \lambda n_i s_i + \mu \sum_{j\epsilon N_i} v_j.$$

We set $r_i(0) = v_i$, $s_i(0) = 0$ for all i.

Figure 4.14 depicts the flow generated by rotating the grid about

(a) Combined Motion
(b) Rotational Motion
(c) Translational Motion

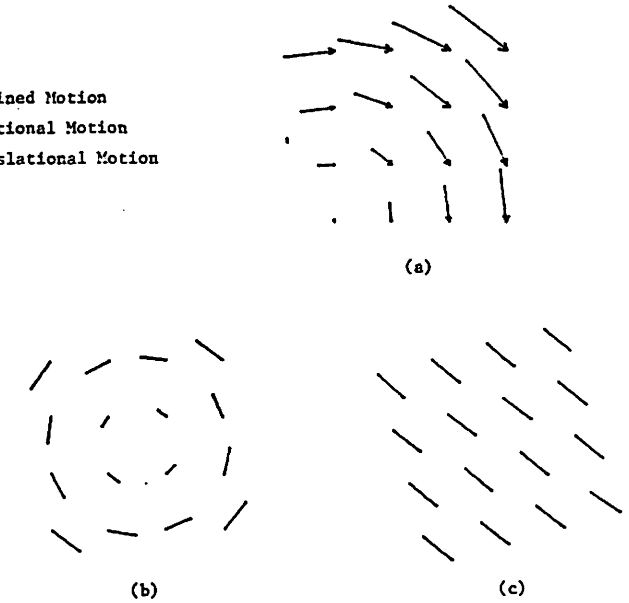


Figure 4.14. Resolution of Flow-Field into Components. Figure 4.14a shows the flow generated by the rotation of a grid of points about the bottom left-hand corner. Figures 4.14b and c show the rotation and translation components of the motion respectively, as generated by the relaxation process described in the text.

a point near its bottom left-hand corner. Figures 4.14b and c show the translation and rotation components, again after 60 iterations with $\lambda = \mu = .05$. It is clear that all the translation components are equal, and that the rotation components sum to zero as required.

4.3 Extraction of Features

We examined in the previous section the computation of optic flow from two successive frames in a moving-image sequence. The generation of optic flow depends upon the matching-up of what we call 'feature-points' in the images. These feature-points are the locations of optical events in the images, that is, points having certain distinguishing characteristics. The characteristics in which we are interested are functions of the gradient of image intensity. Our feature-points correspond to Marr's (1975) 'place-tokens', the components with which he constructs his Primal Sketch. They are also equivalent to Ullman's (1977) 'correspondence tokens'.

We will be looking for edges and corners of objects, which manifest themselves (ideally) as sharp discontinuities in the intensity. These discontinuities may be detected by the standard practice of applying edge-masks, as discussed in Section 2.3. This technique is somewhat analagous to the use of on-off receptive fields in biological visual systems.

Many such operators have been defined, from the relatively simple (Kirsch 1971; Roberts 1965) to the more complex (Hueckel 1973). For a very good comparative review of edge-operators, see Bullock (1974).

While these have been fairly successful at picking out local edge elements, they have not been so good at producing clean boundaries of the objects in the images examined. However, we saw in Chapter II how such boundaries may be extracted by using several stages of (rather time-consuming) processing. As far as our motion analysis is concerned, we intend to track the motion of these extracted edge and corner elements - our 'feature-points' - and so we will find simple differentiation again sufficient.

A word or two about colour would be appropriate here. Most of the receptive-field-mapping experiments that were performed by neurophysiologists were done with black and white stimuli, because these provide the greatest contrast. However, it has been shown that some ganglion cells in the monkey, for example, have colour-opponent centre-surround receptive fields (de Monasterio and Gouras 1975). This suggests that we may find colour a useful feature to use. Now, all of the differentiation operators described earlier were designed to work on black and white images. By the use of filters, the red, green and blue (RGB) components of the images may be generated; the question is, is there any benefit in using colour features? Robinson (1977) investigated the relative usefulness of differentiating the RGB images as well as certain other representations of colour-space. Segmentations were performed on all of the component images. By a subjective analysis, it was concluded that the component most related to simple black and white intensity carried the most information. Consequently, we will stick with black and white images, although we recognize that in so doing some information will be lost.

In Chapter V we will show how by extrapolating the optic flow into the next time-step, we can greatly diminish the time taken by the relaxation process which computes the flow, thus drastically lowering the computational cost per frame. Since feature-points must be extracted from every frame, this saving will be to no avail unless the edge detection is a very fast process. Therefore, as in Chapter II, we will use extremely simple edge-masks. The 'broken-wheel' experiments of Ullman (1977) suggest that individual edge elements are used in the correspondence process rather than the extended linked line-segments. So again, we find that the preliminary processing need not be very elaborate, a situation we consider very desirable.

Not only do we wish to extract these feature-points, but we want to label each of them with a 'feature-type'. These feature-types will characterize the kind of feature present at the feature point. In our case, orientation of edge/corner will be the feature-type used. These labels will serve to aid the relaxation process of Chapter V in deciding whether feature points from consecutive frames match each other. For this purpose, a 'feature-metric' is required in order to associate a 'distance' between different feature-types. We may again compare our ideas with those of Ullman (1977). He uses the concept of 'affinity' between correspondence tokens to determine whether correspondence takes place. His affinity is, amongst other things, a decreasing function of orientation difference between line-segments.

We will use masks to extract edges at eight equally separated orientations. Because of the spatially quantized nature of digitized images, diagonal edges appear locally as corners, indicating the need for both edge and corner masks. Feature points will be those points at which the output of at least one mask exceeds threshold T. We consider the mask to be a detector which 'fires' when this occurs. For convenience, we will use separate masks to detect edges at orientations 180 degrees apart, that is, the same apparent orientation but with different direction of contrast.

Horizontal and vertical edges are detected by the masks in Figure 4.15a-d. Related 'corner' masks are shown in Figure 4.15e-l, but we will show that their effect can be closely simulated by a combination of horizontal and vertical masks. Let us suppose that the shaded regions of the masks 'detect' low intensity, and the open regions high intensity. Thus the mask in Figure 4.15a when applied to the portion of the image shown in Figure 4.15m will fire if $(b + c) - (a + d) > 2T$.

We will define the feature-type of a mask to be the inclination of a line, parallel to the edge detected by the mask and with dark on the left. Thus the feature-type of the mask in 4.15a is 90 degrees.

Consider the detection of 45 degree edges. This may be achieved by the masks in Figures 4.15f and 4.15j. We may therefore want a 45 degree feature point if either mask fires, i.e. if

$$3b - a - d - c > 3T$$

or

$$a + b + c - 3d > 3T.$$

If neither fires, then

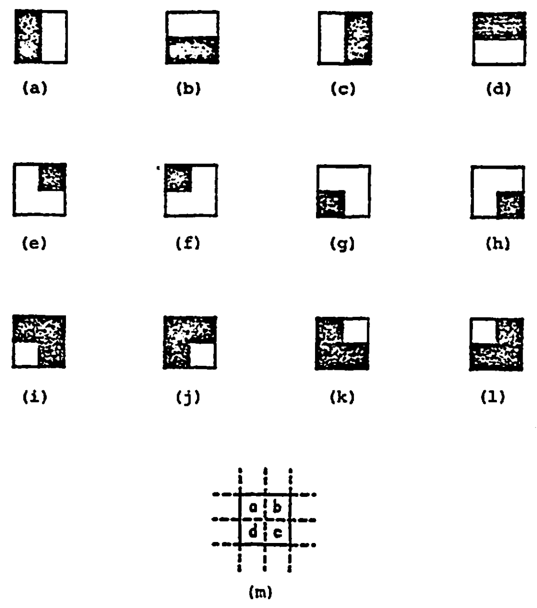


Figure 4.15. 2 x 2 Masks. Figures 4.15a - l depict horizontal, vertical and diagonal 2 x 2 masks. The effects of masks e - l may be achieved by combining outputs of masks a - d. Figure 4.15m shows the labelling of an arbitrary 2 x 2 portion of an image.

$$3b - a - d - c \le 3T$$

$$a + b + c - 3d \le 3T$$

hence

$$b - d \le 3T/2.$$

Now, the 0 degree mask (4.15d) will fire if

$$b + c - a - d > 2T,$$

and the 90 degree mask (4.15a) will fire if

$$a + b - c - d > 2T.$$

If both fire, then we must have

$$b - d > 2T.$$

Consequently, the firing of both these masks guarantees that either 45 degree mask would have fired. Although the conditions for firing are more strict, they are not significantly so, so we will not use diagonal masks, but employ solely the horizontal and vertical ones. The edges detected, and the (combinations of) masks which detect them are given in Table 4.1.

A great advantage of using just horizontal and vertical masks is that the joint firings of masks listed in Table 4.1 are the only ones possible, since a mask and its 180-degree counterpart will never simultaneously fire (for positive T). Therefore at any point in the image, only one feature-type (if any) will be present. The distance between any two feature-types may be simply defined to be the difference between them, modulo 360 degrees.

We should note that it will commonly happen that horizontal edges 'transform' into vertical ones, and vice versa. For example, consider

Edge	Mask(s)	Figure(s)
0	0	a
45	0 & 90	a & b
90	90	b
135	90 & 180	b & c
180	180	c
225	180 & 270	c & d
270	270	d
315	0 & 270	a & d

Table 4.1. Edges at the orientations listed in the left-hand column are detected by the masks in the middle column. These masks are depicted by the corresponding parts of Figure 4.15, indicated in the right-hand column.

the solid diagonal edge of Figure 4.16. As it moves to another position (dotted) along the direction of the diagonal, this effect clearly takes place. However, this is just a variant of the Stimulus-Matching Problem (Burt 1976), where the global movement of, for example, the square in Figure 4.17 is not evident from a very local view. Information from the end-points of the line-segments and cooperative effects from the co-moving individual elements in the line-segments will combine to produce the correct matching.

4.4 Common and Field Velocities

In this section we study the concepts of Common Velocity and Field Velocity. Both of these arose out of experiments in the psychophysicist's laboratory, but are extended so that they may be of use in real-world situations.

4.4.1 Common Velocity

Johansson and his colleagues have performed several studies on the motion of dot patterns (see, e.g. Johansson 1950 Ch. V; Borjesson and von Hofsten 1973), and have coined the term 'common vector' to denote a common component of the dots' velocities. The common vector may be formally defined as that unique vector which when subtracted from the motion vector of each dot leaves residual vectors which sum to zero (Borjesson and von Hofsten 1973). Thus for a set of dots rotating about an arbitrary point, the common vector is the translation of the centroid, and the residual vectors represent the rotation of the other points a-

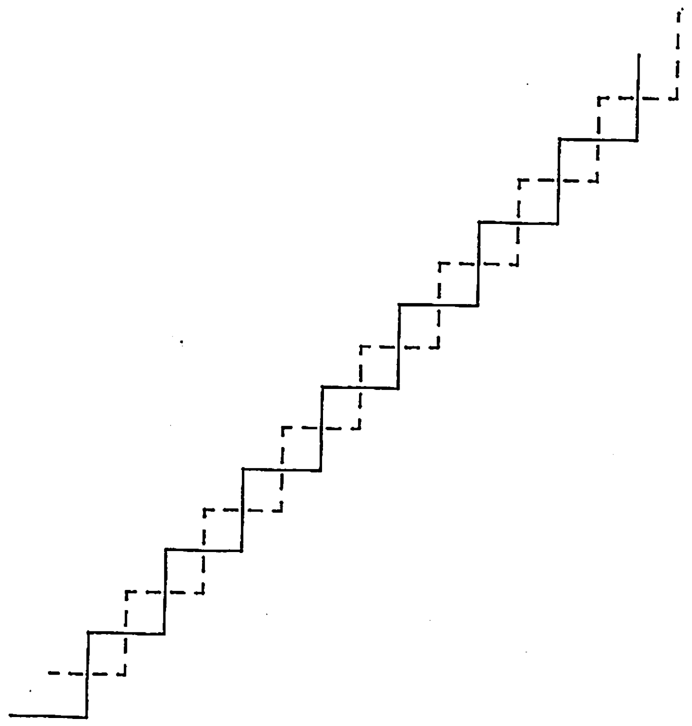


Figure 4.16. Moving Diagonal Edge. A digitized diagonal edge will appear as a sequence of vertical and horizontal segments, as shown by the solid line in this figure. As it moves diagonally upward and to the right by one unit (dotted line), horizontal edges appear to change into vertical edges, and vice versa.

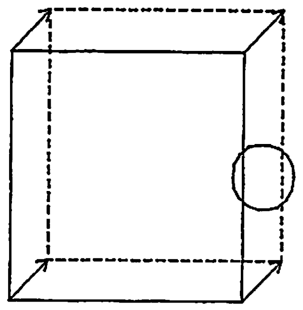


Figure 4.17. Stimulus Matching Problem. From a local view, e.g. within the circle, the global motion of the squre is not apparent.

round the centroid. We will use the terms common vector and common velocity interchangeably.

Johansson was concerned with maximally simplified stimulus events - on the threshold in terms of information. If the extraction of the common vector is an important process in the perception of moving objects, then since biological organisms do not deal with such simplified conditions outside of the laboratory, then this process must be applicable to real stimuli.

In some of the experiments which they performed, movement in depth was perceived from an objective motion of points in the plane, the common vector being the translatory component. However, such a common vector can not in general be defined uniquely for more than three dots. Their experiments resulted in the perception of a rigid object rotating in depth only because three dots moving in the plane have six degrees of freedom, as do rigid bodies moving in 3-space. But for an arbitrary number of independent motions (dots in the plane) some other definition of common vector is required.

Suppose that the visual system tries to superimpose a motion on the set of stimulus points that leaves the smallest residue or difference between the actual motions and the would-be common motion. While it seems that the visual system knows about 3-d motion (but not about 4-d) we will suppose that it only knows about 2-d motion. (The crucial point here is that there are more constraints than degrees of freedom. The simplification facilitates the mathematics, and can possibly be justified by an argument that the third component will be close to zero.)

Suppose that there is a collection of n stimulus objects (dots) with velocities V_i in the plane and weights W_i, where the weights are normalized to sum to one. The weights may be, for example, functions of brightness, size, eccentricity, etc. Suppose that the system tries to find the common vector C which minimizes $\Sigma\, W_i(V_i - C)^2$. This C is easily seen to be $C = \Sigma\, W_i V_i$, that is, the weighted mean of the velocities.

It is clear that this definition is in accord with that of Borjesson and von Hofsten. If C is that vector which leaves residuals which sum to zero, then $\Sigma\, (V_i - C) = 0$, so $C = \Sigma\, V_i/n$ by their definition, i.e., the simple (non-weighted) mean.

To summarize so far, Johansson and colleagues introduced the concept of common velocity as a frame of reference used when stimuli are undergoing various motions. We have shown that the weighted mean of the stimulus velocities gives a value which minimizes the mean-square difference of the other velocities from it; this is in agreement with their definition of common velocity as being that velocity which when subtracted from all stimulus velocities leaves residuals which sum to zero.

4.4.2 The Attraction Function

The trouble with using common velocity as a frame of reference in real-world scenes with multiple motions is that one would like to have it coincide with the actual velocity of an object or collection of objects, so that the object(s) may be viewed as stationary within that frame of reference. The common velocity, computed as above, will not in general do this. This is because is is essentially an average of the motions of different objects.

If the human does indeed perform this kind of common velocity calculation, then it would seem likely that there is a mechanism whereby the resulting value could be adjusted to coincide with the actual velocity of at least one of the moving objects. It would not seem unreasonable to expect that the adjustment is in the direction of the 'nearest' actual velocity.

Let us define an Attraction function $A(x; S)$, where S is a set of points $\{x_i\}$ in some space with metric d, and x is some arbitrary point in that space, to compute that x_i in S which minimizes $d(x, x_i)$. If more than one x_i satisfies this condition, then one may be chosen at random from this subset.

To see the use of the Attraction function, consider the following classic example of induced motion. Suppose that a large frame of 'size' M moves with velocity V, and inside it is a small stationary point of 'size' $m(< M)$. The small dot will appear to move backwards with velocity V, the frame stationary.

We have the common velocity vector $C = MV/(M + m) > V/2$. If we define $S = \{V, 0\}$, then $A(C; S) = V$, which is the velocity of the frame - our 'frame of reference'.

The velocity V above is an example of a 'Field Velocity' as described by Burt (1976). The field velocity may be regarded as the velocity of the environment, or in the case of observer motion, that apparent velocity induced by the motion of the observer.

Consider the 'railway-carriage phenomenon': when one is sitting

in a stationary railway carriage and a train on an adjacent track begins to move, the impression is that the the train on which the observer sits is moving. This may be explained using the field velocity concept.

Essentially, two velocities are presented in the field of view: V, that of the moving train, and 0, that of the stationary window-frame and carriage. If the moving train fills a sufficient portion of the field of view that the common velocity evaluates to a figure $> V/2$, then the Attraction function will compute the field velocity to be V. Thus the moving train becomes the frame of reference, and, relative to it, the stationary train seems to move with a velocity of -V.

The Continuous Attraction Function

We notice that the Attraction function defined above is not a continuous function. It might be interesting and profitable to formulate a continuous Attraction function.

Let $S = \{X_i, i = 1,\ldots,n\}$ be an arbitrary set of n vectors X_i, and let X be the vector whose "nearest match" in S is desired, according to some metric d. Thus we wish to define $\hat{A}(X; S)$, where $\hat{A}$ is in some sense close to A, but is continuous. Bearing in mind that the common vector may be defined as the weighted mean of the stimulus velocities, consider the formula

$$\hat{A}(X; S) = \frac{\Sigma a_i X_i}{\Sigma a_i} .$$

This is the weighted average of the X_i, where the weights, known as Attraction Weights, are functions of the distances between the X_i and X.

Suppose that f is a very fast-decreasing function of one argument, for example,

$$f(y) = e^{-y^2}$$

Then if d is the Euclidean metric,

$$\hat{A}(X; S) = \frac{\sum_i X_i e^{-|X - X_i|^2}}{\sum_i e^{-|X - X_i|^2}} .$$

Now, if for a particular i, $|X - X_i|$ is significantly less than $|X - X_j|$ for $j \neq i$, then the weight $e^{-|X - X_i|^2}$ will be significantly greater than $e^{-|X - X_j|^2}$ for $j \neq i$, and so $\hat{A}$ will be very close to X_i.

This definition of $\hat{A}$ has an interesting property in the case that the 'nearest' to X is not well-defined. Suppose that two values X_i and X_k have 'minimum distance' to X, that is, $|X - X_i| \approx |X - X_k| << |X - X_j|$, $i \neq j \neq k$ (for our purposes, $|X - X_i|$ and $|X - X_k|$ need not be exactly the same). Then the weights $e^{-|X - X_i|^2}$ and $e^{-|X - X_k|^2}$ will be approximately the same, but will be very much greater than $e^{-|X - X_j|^2}$ for $i \neq j \neq k$, so that $\hat{A}$ will evaluate to the mean of X_i and X_k. (Compare the frog sometimes snapping at the 'average' fly in Didday (1970).)

This may be a significant feature for the following reason. We will see in Chapter V the incorporation of the Attraction function in the relaxation process which performs stimulus-matching. If two points at X_i and X_k respectively are equally good candidates for match with some point X, then it may be a wise thing to temporarily 'hedge one's bets' rather than to choose one or the other arbitrarily. If an arbitrary decision is made and it turns out to be wrong, then because of the

way in which information travels in a relaxation process, the effects may spread and the system may never recover, or if it does, it may take a long time to do so.

It is interesting to note that the continuous Attraction function $\hat{A}$ may be described algorithmically by a program that has a linear flow of control. The discontinuous nature of A arises out of the fact that a program to compute it would require conditional branching. It is possible that the continuous function might be favoured in a particular application, since with the right equipment it may be computed in a single step.

4.4.3 Extension of the Field Velocity Concept

Psychophysical experiments concerned with induced motion in the laboratory and the railway-carriage phenomenon have given rise to the idea of a field velocity - the common motion of the 'majority' of the points in a scene which becomes the reference by which all other motions are judged. We will show that there is no need for the field velocity to be spatially uniform.

Consider the case of a visual organism moving on a flat surface with certain fixed objects at some distance - say a dense forest to which the observer is both looking and moving over an open plain. The optic flow produced would be something like that in Figure 4.18. This flow is an easily generated and maintained part of the organism's internal representation of the current state. This velocity, by no means uniform, can be used as the field velocity. It will be constantly sub-

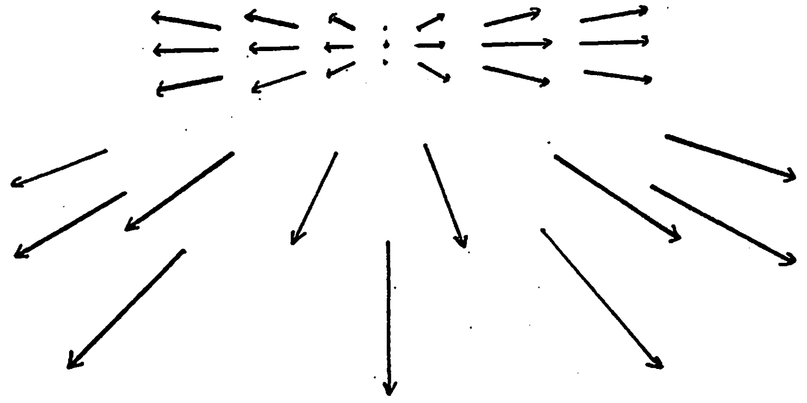


Figure 4.18. Diagrammatic Flow Field. This figure depicts the kind of flow field one might expect to be generated while travelling over an open field towards a forest.

tracted from the instantaneous optic flow, and the lack of any discrepancy will imply that there is no imminent action to be taken by the organism. Any irregularities, however, such as bumps on the ground, or one tree being much nearer than the others will be detected instantly, so that the organism may take evasive action. The important point here is that the field velocity may be represented by a velocity field, not just a uniform velocity. This extension allows the concept to be of more general applicability than before.

CHAPTER V

SYNTHESIS OF A MOTION-DETECTING SYSTEM

5.1 Introduction

At the heart of the problem of the analysis of moving-image data lies the matching-up of corresponding points in consecutive frames. This has been termed the Correspondence Problem (Duda and Hart 1970) and also the Stimulus-Matching Problem (Burt 1976). It has been tackled in a variety of ways, but most often with simplifying assumptions, such as no rotation, no occlusion, binary images, single motions, etc. (see, e.g. Aggarwal and Duda 1975; Potter 1977; Dreschler and Nagel 1978; Radig 1978). The correspondence problem also arises in stereopsis, where two frames taken at the same time but from different positions need to be matched (see, e.g. Levine et al. 1973; Thompson 1975; Nevatia 1976; Ganapathy 1975). For either application, it has aroused much interest as a problem in cooperative computation in neural-like circuits (Julesz 1971; Dev 1975; Marr and Poggio 1976; Nelson 1975; Trehub 1978).

In this chapter, we explore the use of a cooperative algorithm, but using computations which are not necessarily neural-like. The 'control-structure' of the algorithm, however, is based on concepts from neurophysiology. Our data is in the form of retinotopic arrays or 'layers' being processed in parallel by arrays of computational elements. These elements have certain 'receptive fields', that is they obtain their input from localized neighbourhoods in the layers of data. Due to the overlap of these neighbourhoods, these elements communicate with each

other. Since the computations are iterative and proceed until certain local criteria or 'consistency conditions' are met, the process may be regarded as an example of a relaxation process (see Chapter II for a general discussion of relaxation processes).

We discussed in Chapter IV how Optic Flow is a good low-level representation of motion. We describe here processes to compute the Optic Flow from the motion of dot-patterns. We will, however, reserve the phrase 'Optic Flow' to refer to the veridical flow associated with the motion, and use the term 'vector velocity field' to denote our estimations of the flow field.

We will demonstrate at the end of this chapter that by using extrapolation techniques we may greatly decrease the computational cost per frame over a series of frames of data; a possibility that has not received too much attention elsewhere to this data. We will use real data in this demonstration.

5.1.1 Overview

In this chapter we follow the development of a motion-analysis system known as MATCH. We generally suppose that two arrays of layers P and Q of feature-points extracted from consecutive frames of data are available to the system. The task of MATCH is to generate and maintain the vector velocity field layer D representing the motion of these points.

We start by specifying the consistency conditions which are used to constrain the motions of the points. These conditions are incorporated into a relaxation equation which is used to generate the vector velocity field. This is performed on two consecutive frames of feature-points, with the initial assumption of no occlusion. Experiments are firstly carried out under three conditions in which the velocity field elements are in register with the points in the Q layer.

(i) Velocity field components correspond only to extracted feature-points. This is the basic experimental paradigm.

(ii) Velocity field components in the background are 'filled in'. This arrangement simulates the occurrence of homogeneous regions of a moving surface, over which no feature-points are extracted, but for which it is desired to associate a flow-field.

(iii) Velocity field components in the background correspond to a stationary background object. This arrangement simulates the different relative motions of two adjacent objects in the image.

(iv) We next extend the system to cover the (more realistic) case where the stimulation (set of feature-points) is not in register with the discretized vector velocity field. We see that this requires the addition of an extra layer C between the D and P and Q layers (see Figure 5.1).

(v) The discussion then turns to the problems of analysing sequences of more than two frames of data. We see that it greatly facilitates the analysis to impose certain nominal restrictions upon velocities manageable by the system. For very slow velocities, where stimulus points remain within receptive-fields (pixels) for more than one frame, an averaging process is applied to smooth the inferred motion.

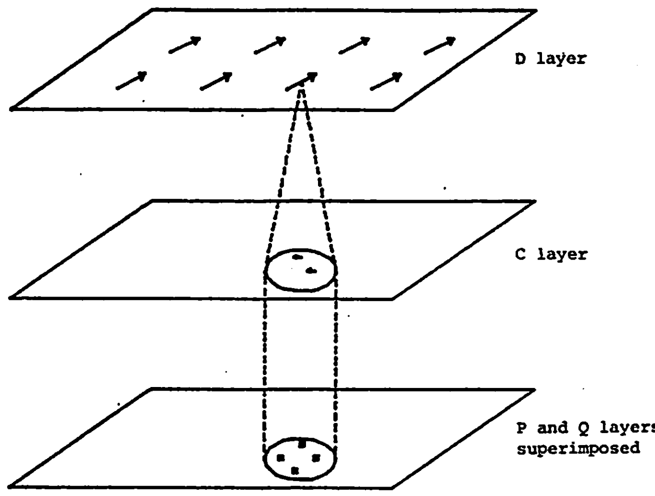


Figure 5.1. Organization of Data Arrays. P and Q are two consecutive frames of extracted feature-points. They are superimposed in this figure, the open squares representing points in P, the filled squares points in Q. The points drawn all fall within the receptive field of a single element in the D-layer. The C-layer contains 'corrections' to the D-layer derived from trying to match points in P with points in Q, using the vectors in D as 'estimates'. These corrections will in turn provide one of two updating components for the elements in the D-layer. The other component (not drawn) is from a weighted average of surrounding elements, used to smooth the flow. In the early experiments described in this chapter, the flow vectors in the D-layer are in register with the feature-points in Q, so the C-layer is not needed explicitly. Note that, in the general case, the receptive fields of adjacent D-layer elements may overlap.

In order to track the motion beyond the second frame, the internal representation of the motion, the vector velocity field, needs to be propagated. To do this, we introduce an algorithm for performing the tracking. We present results of running the system on sequences of frames of real data. The feature-points are extracted in the manner described in Chapter IV.

(vi) Finally, we describe how the system manages occlusion and disocclusion, and discuss how the segmentations produced as a result of occlusion-detecting processes may be integrated with other segmentation processes.

5.2 The Basic Stimulus Matching Algorithm

The frames of data consist of arrays of intensity values. At some of the positions in these arrays the feature-detectors have determined that one of a set of features is present. In this section we study the matching of these feature-points, or simply points, in two successive pre-processed images.

5.2.1 Specifying the Consistency Conditions

Let us define a pre-processed image P to be a set of pairs {(PX(i), PT(i)) i = 1,...,n}. The pair (PX(i), PT(i)) is an assertion that feature PT(i) is found at location PX(i). We will sometimes use the abbreviated notations (X_i, T_i) for (PX(i), PT(i)) if P is understood, or P_i if the location and feature attributes do not need to be referred to individually.

Suppose that P and Q are two successive pre-processed frames from a movie sequence, Q being the more recent. The goal of MATCH is to find the Vector Velocity Field associated with P and Q.

We will not initially address the problem of occlusion directly. From the observation that, in general, if a point becomes occluded or disoccluded at a given instant, then it will stay as such for some (possibly considerable) time interval, we may conclude that the phenomenon occurs sufficiently rarely, in the temporal sense, at a given spatial location that it can be treated as noise. In performing the correspondence, we make use of the fact that the vast majority of points which are present in P are also present in Q, and vice versa. It will be noted, though, that the disappearance or appearance of points may be taken as a strong cue for the occurrence of occlusion or disocclusion and hence the existence of a boundary in the vicinity. This point will be addressed in detail in Section 5.5.

As we have just observed, almost every point in Q will have a predecessor in P. Thus we will be able to associate with each such point Q_i at location QX(i) a vector DX(i), and to find some (PX(j), PT(j)) in P such that

(1) $PX(j) + DX(i) = QX(i)$

(2) $PT(j) \approx QT(i)$.

Due to noise, illumination and topological changes between frames, and other factors, we will only find approximate equality in (2) whenever we process real data. In any case, while these (approximate) conditions are necessary for a correct match between Q_i and P_j, they are not sufficient. There is no guarantee that PX(j) really did undergo movement DX(i) to reach point QX(i). In fact, given any two successive frames P and Q there is never any guarantee that a point in P and a point in Q correspond to each other. However, given certain assumptions about the motion, we may be able to say with a high degree of confidence that the two points correspond.

The most important assumption that we make is that we are viewing real-world rigid motion. Under this assumption, points on the surface of a moving object move with a velocity which is very close to that of neighbouring points. That is, the Vector Velocity Field is continuous across the object. For the moment we will put aside boundary problems and assume that all the points in P and Q belong to only one object.

Consider Figures 5.2a and 5.2b. The symbol + corresponds to points of P, the Δ to points of Q, and the arrows to the associated velocity field DX. Now, if, for example, the feature T for all these points is the same, then conditions (1) and (2) are satisfied for both 5.2a and 5.2b. However, if we make the assumption that these points belong to an object which has undergone rigid motion, then 5.2 b is clearly correct, and 5.2a incorrect. In fact, the crossing phenomenon of 5.2a is never observed in apparent motion situations (Kolers 1972). We can express this fact by adding a third condition:

(3) DX(i) is (approximately) equal to the average of the value of DX at neighbouring points.

This condition will not hold across boundaries of objects moving with respect to their background. This phenomenon is explored in Exper-

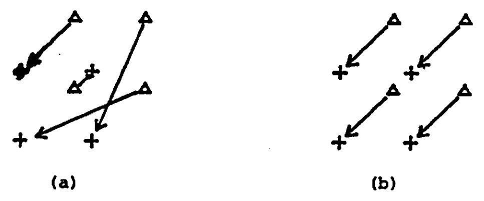


Figure 5.2. The Correspondence Problem. The +'s and Δ's represent a group of four feature-points in two consecutive frames. Two possible correspondences are shown. The correspondence in Figure 5.2b is preferred due to the uniformity of the generated flow. In fact, the crossing of the flow in Figure 5.2a is never observed in apparent motion situations.

iment 8 (Section 5.3.3) below, where it is seen that the discontinuity in the flow is not sufficient to prevent the correct correspondence from being formed. A more drastic form of discontinuity occurs when points disappear due to occlusion; this is discussed in Section 5.5.

To see why we use the average in (3), consider a smooth surface with vector velocity D(x, y) associated with each point at coordinates (x, y). By assumption, D(x, y) is continuous. If the gradient of the velocity field is not too large, then the field is approximately linear in x and y. It follows that D(x, y) is, to first order, the average of neighbouring values. It is sufficient that we only require approximate equality in (3) for two reasons:

(i) the points in P and Q will not necessarily be equally spaced, and

(ii) if the motion contains a rotation component for example, the assumption of linearity will be inaccurate (see Figure 5.3).

We can use conditions (1), (2) and (3) to generate the Vector Velocity Field. This is done essentially by a relaxation process which makes corrections to the value DX at each point according to the degree to which (1), (2) and (3) are in error. The derivation of the relaxation equations may be seen by examining the action necessary under a slight perturbation from the state of convergence.

Suppose that DX(i), for some Q_i, is perturbed from the equilibrium state, while all the other points remain unaffected. Let P_j be the correct match point for Q_i. Unless the perturbation is too large, the quantity QX(i) - DX(i) will still be closer to PX(j) than PX(k), for

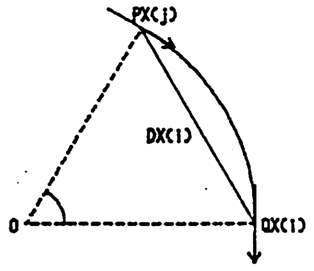


Figure 5.3. Effects of Non-linear Motion. DX(i) is the displacement associated with QX(i) which was formed when PX(j) rotated about O. DX(i) is not equal to the tangential velocity at either PX(j) or QX(i).

any $k \neq j$. So, using the Attraction function described in Section 4.3, A[EX(i); P] will pick out P_j, where EX(i) = (QX(i) - DX(i), TX(i)) is the estimated match for Q_i. By condition (1), then, the correction to DX(i) will be

$$QX(i) - A[EX(i);\ P] - DX(i). \qquad (4)$$

By condition (3), though, the correction to DX(i) would be

$$\frac{\Sigma\, w_{ij} DX(j)}{\Sigma\, w_{ij}} - DX(i) \qquad (5)$$

that is, the weighted average of DX at neighbouring values where the summations are carried out over all Q_j in some neighbourhood of Q_i. The w_{ij} are decreasing functions of the distance between PX(i) and PX(j) since we wish nearer points to have a stronger effect.

Thus the combined effect will be to impose a correction to DX(i) of size

$$\lambda\left[\frac{\Sigma\, w_{ij} DX(j)}{\Sigma\, w_{ij}} - DX(i)\right] + \mu[QX(i) - A[EX(i);\ P] - DX(i)] \qquad (6)$$

for suitable λ and μ (see Section 5.2.3 for a discussion of the setting of these parameters and, in particular, conditions under which it is computationally useful to violate the natural condition $\lambda + \mu = 1$).

5.2.2 Defining the Relaxation Process

So far, we have developed a correction (6) to DX without reference to condition (2). We may incorporate this constraint by letting the distance between feature points depend also upon the difference between the features. Thus if we use the feature-metric described in Section

4.3, which we will denote by F, then for the continuous Attraction function $\hat{A}$, the attraction weight a_{ij} for points (X_i, T_i) and (X_j, T_j) becomes

$$a_{ij} = e^{-(X_i - X_j)^2 - kF(T_i,\ T_j)^2} \qquad (7)$$

where the parameter k determines the relative importance of the feature-types in the matching process. A value of about .015 was used in the experiments described later in the chapter.

In order to appreciate how fast the a_{ij} fall off with distance, suppose that three points P_1, P_2 and P_3 have identical feature-types to a point P_0, and are of distances 1, 2 and 3 grid-units respectively from P_0. Then $a_{01} = .37$, $a_{02} = .018$, $a_{03} = .0001$, compared with $a_{00} = 1$. The exponential function was used simply because it has the right shape. Whether other functions, such as $\dfrac{1}{1+(X_i-X_j)^2+kF(T_i,T_j)^2}$ are more effective depends on their relative ease of computation (which will in turn depend on whether the substrate for implementation is biological or mechanical), time to convergence and other factors. This issue was not explored any further here.

Our formulation for the continuous Attraction function is then

$$\hat{A}[(X_j,\ T_j);\ S] = \frac{\sum_{i=1}^{n} a_{ij}(X_i,\ T_i)}{\sum_{i=1}^{n} a_{ij}} \qquad (8)$$

and the discontinuous Attraction function is simply

$$A[(X_j,\ T_j);\ S] = \text{that } (X_i,\ T_i) \text{ which maximizes } a_{ij}$$

where $S = \{(X_i,\ T_i),\ i = 1,\ldots,n\}$.

The μ-component of equation (6) is now fully defined. In (5) we wish the w_{ij} to behave like the attraction weights, that is to fall off quickly with distance, but we are not interested in the similarity of the features associated with the points. So we simply set

$$w_{ij} = e^{-(X_i - X_j)^2}$$

to give

$$DX(i)_{t+1} = DX(i)_t + \lambda B_1(i) + \mu B_2(i) \qquad (9)$$

where the correction align DX(i) with neighbouring field elements is

$$B_1(i) = \frac{\Sigma\, w_{ij} DX(j)}{\Sigma\, w_{ij}} - DX(i) \qquad (9a)$$

and the correction to extend DX(i) to reach to the nearest feature-point in P is

$$B_2(i) = QX(i) - DX(i) - A[EX(i); P] \; . \qquad (9b)$$

5.2.3 Setting of Update Parameters λ and μ

Our earlier analysis indicated that for small perturbations, either of the correction components (9a) or (9b) of (9) will provide a fairly accurate adjustment. This implies that for speedy convergence, $\lambda + \mu$ should equal 1. If the system is starting up, with $DX(i, 0) = 0$ for all i, and if there was a sufficiently small movement between the first two frames that the attraction function will pick the right point each time, then putting $\mu = 1$ will guarantee immediate convergence. In other

cases, however, such as when every DX(i) is correct except one, which is terribly wrong, then we would wish λ to be 1.

It was found experimentally (see Experiment 1 described below), that setting $\lambda = \mu = 1$ for a few iterations and then reducing them both to .5 worked very well. The parameters λ and μ correspond to time-constants in differential equations. The initial use of values for λ and μ which sum to a value greater than 1, which is equivalent to a state of instability (i.e. the corresponding differential equation will diverge), is important in getting over the 'inertia' in the start-up condition. As we will see later, when subsequent frames are presented, the DX array only needs minor modifications from one frame to the next, so setting $\lambda = \mu = .5$ is completely satisfactory here. The 'best' time to reduce λ and μ to .5, that is the earliest time which guaranteed convergence, turned out to vary slightly between experimental conditions, but it also turned out to matter very little exactly when this took place. It also happened under some conditions that convergence was a little quicker if λ was reduced before μ.

In cases where only two frames exist (e.g. stereopsis), and it is desired to let the relaxation continue to convergence, then we may let λ continue decreasing to zero after it has been already reduced to .5. This procedure leaves 9b (the Attraction function) the only contributing updating component, and therefore guarantees that a match will be found for each point in Q.

Experiment 1. Setting of Time-Constants λ and μ

To demonstrate the usefulness of changing λ and μ dynamically, the following simplified experiment was performed. A relaxation process was tested using three equally spaced points a, b and c along a horizontal line as the first frame P, which was displaced by 0.6 times the inter-point distance to the left to form the second frame Q. Denote the displaced points by A, B and C, and let the points in P be depicted as circles, and those in Q by squares, as shown in Figure 5.4. No feature-types were associated with the points, so they all appeared identical. In the averaging process, only immediate neighbours were considered.

A vector D_S is associated with each square S (S = A, B, C), and indicates, as the iterations proceed, the output of the correspondence process as it tries to find a match for the squares. Due to the linear nature of the problem, all of the vectors will be horizontal, and may be represented by positive or negative displacements. The triple $D = (D_A, D_B, D_C)$ will be used to denote the entire velocity field at a given time.

The functioning of the relaxation process can be examined by following the state of D over a number of iterations. In Table 5.1 the progress of the system is shown over 20 iterations where λ = μ = 0.5, and in Table 5.2 the results for λ = μ = 1.0. Table 5.3 shows the results of running the process with λ = μ = 1.0 for the first 12 iterations, then setting them both to 0.5 for the remainder. Only in this last case does the system converge correctly. It is clear that it is necessary to use initially large values for λ and μ, but that their con-

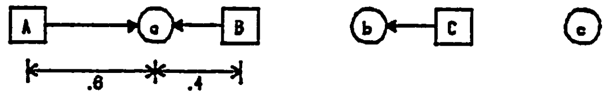


Figure 5.4. Experimental Setup of Three Points in a Line. Three points in a, b and c (indicated by circles) were displaced .6 units to the left. The new positions of the points (relabelled A, B and C) are indicated by squares. Initially hypothesized matches (nearest neighbours) are indicated by arrows.

Iter.	D_A	D_B	D_C
1	0.60000	-0.40000	-0.40000
2	0.10000	-0.06667	-0.40000
3	0.51667	-0.45556	-0.23333
4	0.11389	-0.00185	-0.51111
5	0.54213	-0.53117	-0.14537
6	0.06235	0.08637	-0.59290
7	0.61151	-0.63410	-0.06037
8	-0.02280	0.20645	-0.68687
9	0.71462	0.22581	0.04666
10	0.35559	0.70322	-0.31042
11	0.77361	0.14625	0.10682
12	0.28622	0.79605	0.61971
13	0.85491	0.37128	0.69817
14	0.35818	0.86684	0.44156
15	0.85423	0.28869	0.81264
16	0.31718	0.96320	0.33802
17	0.92301	0.17627	0.91259
18	0.22663	1.09436	0.23184
19	1.03386	0.02325	1.03126
20	0.09469	0.27287	0.09600

Table 5.1. Successive States of Experimental System (1). The state of the triple (D_A, D_B, D_C) is shown over 20 iterations for $\lambda = \mu = .5$ throughout. The system did not converge.

Iter.	D_A	D_B	D_C
1	0.30000	-0.20000	-0.20000
2	0.32500	-0.21667	-0.30000
3	0.32708	-0.23194	-0.32917
4	0.32378	-0.23900	-0.34028
5	0.32120	-0.24258	-0.34482
6	0.31965	-0.24437	-0.34685
7	0.31882	-0.24526	-0.34780
8	0.31839	-0.24571	-0.34827
9	0.31817	-0.24593	-0.34849
10	0.31806	-0.24604	-0.34861
11	0.31800	-0.24610	-0.34866
12	0.31798	-0.24613	-0.34869
13	0.31796	-0.24614	-0.34870
14	0.31796	-0.24615	-0.34871
15	0.31795	-0.24615	-0.34871
16	0.31795	-0.24615	-0.34872
17	0.31795	-0.24615	-0.34872
18	0.31795	-0.24615	-0.34872
19	0.31795	-0.24615	-0.34872
20	0.31795	-0.24615	-0.34872

Table 5.2. Successive States of Experimental System (2). The state of the triple (D_A, D_B, D_C) is shown over 20 iterations for $\lambda = \mu = 1.0$ throughout. The system did not converge.

Iter.	D_A	D_B	D_C
1	0.60000	-0.40000	-0.40000
2	0.10000	-0.06667	-0.40000
3	0.51667	-0.45556	-0.23333
4	0.11389	-0.00185	-0.51111
5	0.54213	-0.53117	-0.14537
6	0.06335	0.08637	-0.59290
7	0.61151	-0.63410	-0.06037
8	-0.02280	0.20645	-0.68687
9	0.71462	0.22581	0.04666
10	0.35559	0.70322	-0.31042
11	0.77381	0.14625	0.10682
12	0.53001	0.47115	0.36327
13	0.55029	0.52740	0.50860
14	0.56942	0.56438	0.55900
15	0.58345	0.58213	0.58085
16	0.59140	0.59107	0.59075
17	0.59562	0.59554	0.59545
18	0.59779	0.59777	0.59775
19	0.59889	0.59888	0.59888
20	0.59944	0.59944	0.59944

Table 5.3. Successive States of Experimental System (3). The state of the triple (D_A, D_B, D_C) is shown over 20 iterations, where $\lambda = \mu = 1.0$ for the first 11 iterations, then .5 for the remainder. In this case, the system did converge.

tinued use at these levels prevents convergence, so that they should eventually, that is after some number N iterations, be decreased. In the particular experiment just described, convergence was achieved when N took any value from 12 to 20. With experiments performed with 4 x 4 grids, to be described later, the smallest possible value for N varied somewhat from run to run, but using N = 6 was found to work very well in general. It is felt that the dynamic modification of λ and μ is an interesting problem, and one which may provide an important contribution to efforts to solve the correspondence problem. However, no further experiments were done in this vein beyond those described above.

5.3 Experiments with MATCH

Some experiments with the basic version of MATCH, as defined so far, are described below. The original, discontinuous Attraction function was used throughout these experiments. The data were simulated.

A rectangular grid P of points was generated, and to each point was associated random real numbers between 1 and a variable limit R (which we will call the 'feature-spread'), representing the feature-type. The feature-metric used is simply the absolute difference between the feature numbers. The set of points Q is generated from P by causing P to undergo whatever motion or combination of motions was specified by the experimenter. The points in P are considered to lie in a plane perpendicular to the experimenter's line-of-sight, and may undergo any rigid 3-d motion. A description of this computer system is given in Appendix Three.

A regularly spaced grid of points was used because it can produce the potentially most ambiguous motions, viewed at a local level at least. If, for example, a large rectangular grid is moved sideways by one inter-element distance (IED), then on regarding the interior of the grid it would seem as if it had not moved at all. Only the fact that in some cases the features do not match up, and the more obvious situation at the boundary, indicate that motion has taken place. Even so, it is never clear at a very local level exactly in which direction motion has taken place, and the interaction of neighbouring processing elements is required to resolve the problem. Using a rectangular grid was therefore thought to be a useful test-bed for the stimulus-matching relaxation process, and was used extensively in testing algorithms.

Experiments were performed using a wide variety of motions between P and Q. In all translation cases, if the displacement was less than half an IED, then convergence was immediate, as the earlier discussion predicted.

If the motion was a rotation about line-of-sight, then the system oscillated a little before settling down, even if the displacements were all less than half an IED. This was because the displacements DX(i) between a point Q_i and its 'partner' P_j is not exactly equal to the tangential velocity of P_j (see Figure 5.3). We see that the DX array is a displacement field, and is only an approximation to the Vector Velocity Field. Because of this, 9a will not be exactly zero at the same time that 9b is, so the relaxation process will adjust the value of DX(i) very slightly away from its 'correct' value QX(i) - PX(j) in order to

reduce 9a and so smooth out the displacement field. This error is small, but noticeable. In images which have a higher density of feature-points, such as real-world images, the smoothing action of 9a will be less drastic. Besides, the displacements undergone by these points are far greater than would be expected in a movie sequence where frames are typically 1/30 second apart. In any case, the error is removed by letting the contribution of 9a tend to zero, as described in Section 5.2.3.

<u>Experiment 2. Comparison of Simultaneous and Sequential Updating Algorithms</u>

At this point it may be beneficial to compare the relaxation approach we have used with a variant which, it might be expected, would easily overcome the problems of inexact matching described above. Our relaxation process updates each velocity field vector by two components, the one adjusting the vector to the average of its neighbours, the other to match up with its nearest neighbour. These updates occur simultaneously. The suggested variant would make the adjustments sequentially. That is, at each even iteration, for example, every velocity field vector would be replaced by or adjusted towards the average of its neighbours, and at every odd iteration it would be adjusted to close in on the (possibly new) nearest neighbour.

This modified algorithm has one minor and one major drawback, though. Consider first the case of rotation about the line-of-sight, and suppose that the system is close to convergence, that is, both updating components 9a and 9b are everywhere close to zero. Consider how the alternating algorithm would work. At an even iteration, every velocity vector would be corrected to reach to the nearest feature-point, which is, by assumption, the 'correct' one. This state of convergence is not stable, though, because the velocity field is not locally uniform, so that the next iteration will further adjust the vectors in an attempt to smooth the field. The overall effect will be oscillatory behaviour, although the states encountered are all near the desired one. This problem will vanish, though, if the two iterations are considered to comprise a single step. In this case there will be no oscillation.

The major drawback with the proposed algorithm is that in certain circumstances, it will never 'get off the ground'. We will consider the situation discussed earlier in the section dealing with setting of parameters λ and μ. We have three points a, b and c equally spaced along a horizontal line, and these points are displaced to the left by 0.6 times the inter-dot distance, as depicted in Figure 5.4. As before, the circles represent the points before moving, the squares A, B and C the points after moving. Again, we will suppose that the points are all identical, and that only immediate neighbours are considered in the averaging operations. Let us associate an initially zero vector with each square.

We saw in Section 5.1.3 how the simultaneous-update algorithm performed; we will now examine the alternating-update algorithm. As before, due to the simplicity of the experiment, we can represent the state of the system, i.e. the entire velocity field by a triple of values D, representing the velocity field components of the three points, in left-to-right order.

In the first iteration, each vector will match its corresponding square with the nearest circle, as shown in Figure 5.4. The state will thus be

$$(.6, -.4, -.4).$$

That is, A is matched with a, B is matched with a, and C is matched with b. The averaging in the next iteration will result in the state

$$(.1, -.67, -.4).$$

Now, the nearest circle to the point .1 units to the right of square A is circle a. The nearest circle to the point .67 units to the left of square B is also circle a. The nearest circle to the point .4 units to the left of square C is circle b. So all of the associations made in the third iteration will be the same as for the first, resulting in the same state of the system, namely

$$(.6, -.4, -.4).$$

The system will therefore oscillate, and never come close to finding the right correspondence.

This example was used to highlight the problems with this algorithm which appear also in more complicated situations, but in more subtle ways. Because of the failings of this algorithm, we will continue to use the originally proposed algorithm which makes both kinds of update simultaneously.

We should note, though, that the simultaneous-update algorithm is also prone to failure in certain circumstances. If updating components 9a and 9b at a point are equal but opposite, then the net effect will be zero, and the system will be stable there. This is a particular case where the dynamic modification of time=constants λ and μ (and hence the updating contributions) might play an important role.

5.3.1 Some Results with the Basic Experimental Setup

Experiment 3. Translation of a Grid of Points

Figure 5.5 shows the action of presenting the system with a 4 x 4 grid of feature points (P), and the same grid shifted down and to the left by one IED (Q). A value of 5 for the feature spread R was used here. Note that since this value is very small (see discussion under Experiment 4 below), the feature-types have little influence on the matching process, so that after the first iteration, for example, each point is matched with its nearest neighbour. The graph in this and subsequent figures shows the total discrepancy from convergence at each iteration, i.e. the sum of the distances from each 'free-floating' arrowhead to its correct 'target' point.

Experiment 4. Translation with Feature-Types of Varying Distinctiveness

The same run as in Experiment 3 was made with different random numbers as feature-types in the range 1 - R, and different values of the range-limit R. The purpose of this experiment was to see to what extent the relaxation process depended on the distinctiveness of feature-types to sort out the right matches.

Table 5.4 shows the number of times the system converged out of ten trials with selected values of R. It is seen that when R was equal to 10 or greater, the system most often converged. In these cases the

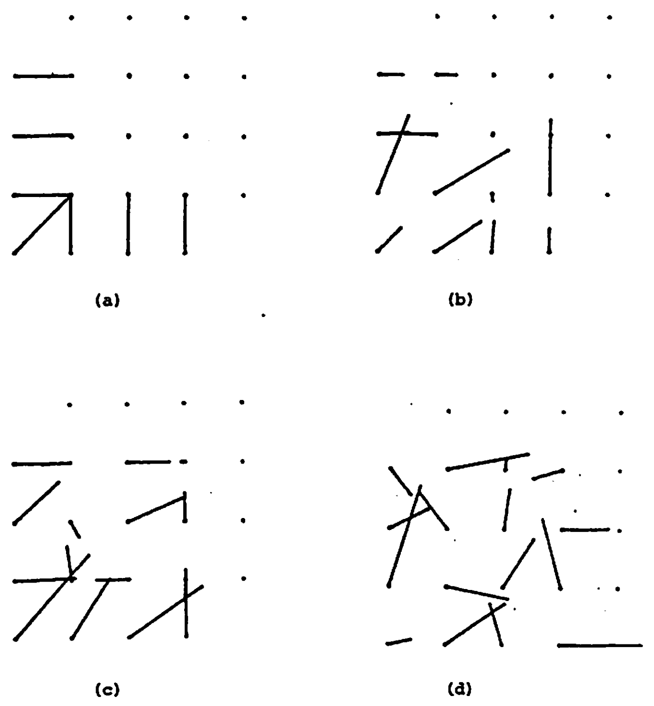
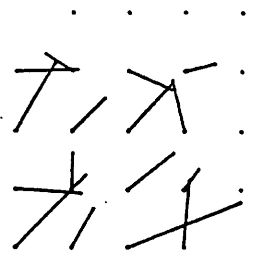
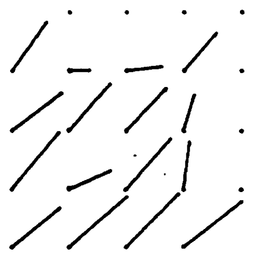
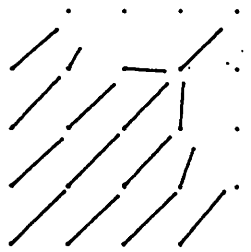


Figure 5.5. Translation in the Plane. A 4 x 4 grid of points was displaced down and to the left by one grid unit. Figures a - l show the generated flow after the first 12 iterations. A random spread of 5 was used. Due to the smallness of this value, the feature-types associated with the points did not influence the match process very much, as is evident in figure a, where initial matches were made with nearest neighbours. Figure m shows the state of the field after 20 iterations. The graph in this and succeeding figures depicts the total discrepancy from convergence at each iteration. This graph is generated by a supervisory process which knows the correct correspondence (the MATCH process itself does not know this).


(e)


(f)


(g)

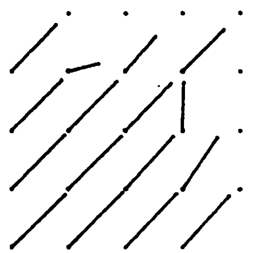
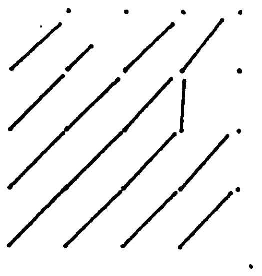
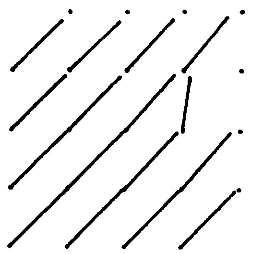
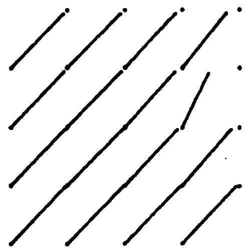

(h)

Figure 5.5e-h


(i)


(j)


(k)

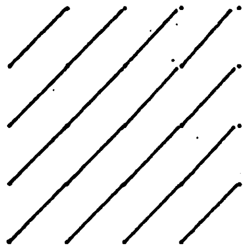
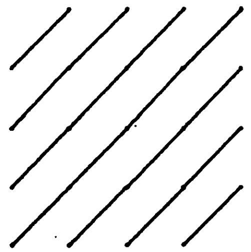

(l)

Figure 5.5i-l



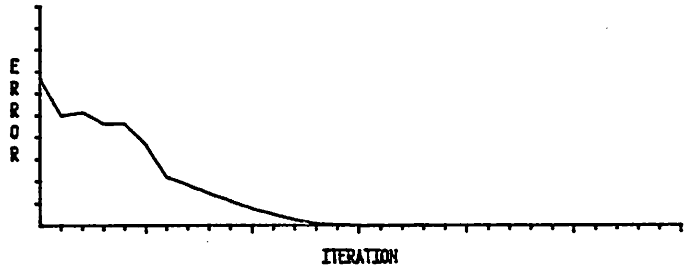


Figure 5.5m

RANDOM SPREAD	SUCCESSES (out of 10)
5	2
7	6
10	6
13	7
15	9
18	8
20	10

Table 5.4. Effect of Random Spread on Convergence. This table shows the number of times out of 10 the system converged with selected values of the random spread R, when a 4 x 4 grid of points was shifted down and to the left by 1 IED.

average difference between two randomly chosen feature-values would be about 3. Putting this value in (7), we see that, for k = 1/64, this difference was equivalent to about 3/8 of the inter-dot distances in the grid. Now, this is a relatively small value, indicating that the system only needs a slight bias from the feature-type comparisons in order to work well. Indeed, when R was less than this value, the system often converged, even in some conditions when R was equal to 1, i.e. when every point had the same feature-type. To better appreciate the meaning of a given magnitude of R, R was increased until the system found the correct match for every feature-point at the first iteration. It was found that values around 400 were necessary to achieve this.

Experiment 5. Non-translatory and/or Non-Planar Motion

Further runs were made with motions other than translation in the plane. In Figures 5.6, 5.7, 5.8, 5.9 and 5.10 the motions were rotation about the line-of-sight (LOS), rotation about a line parallel to the LOS, rotation about a line perpendicular to the LOS, translatory movement along the LOS, and combined rotation about and translations along the LOS. After 10 iterations, λ was gradually reduced to zero, as discussed before, with μ always equal to 1 - λ. In all of the non-translatory cases, this was necessary to get exact matches. Without this modification, though, the errors were quite small, and non-qualitative. Notice the distinctive flow-patterns formed by these motions.

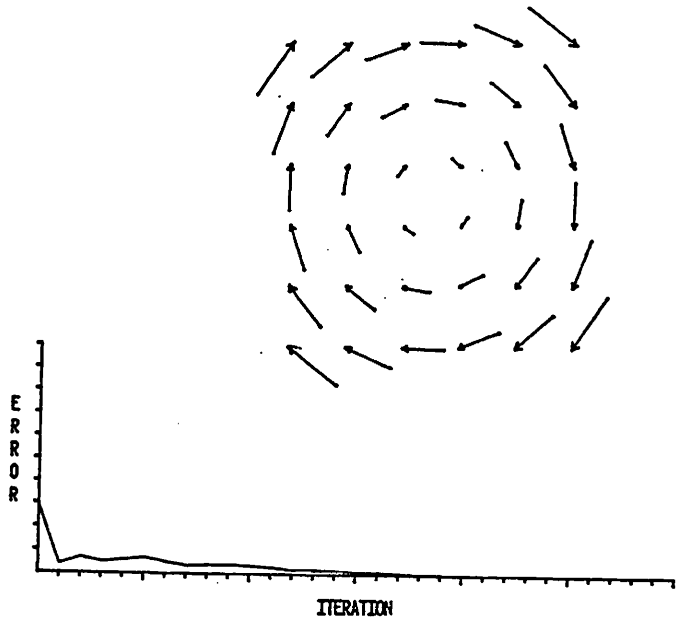


Figure 5.6. Rotation in the Plane. Figure 5.6 shows the flow generated when a 6 x 6 grid of points is rotated.

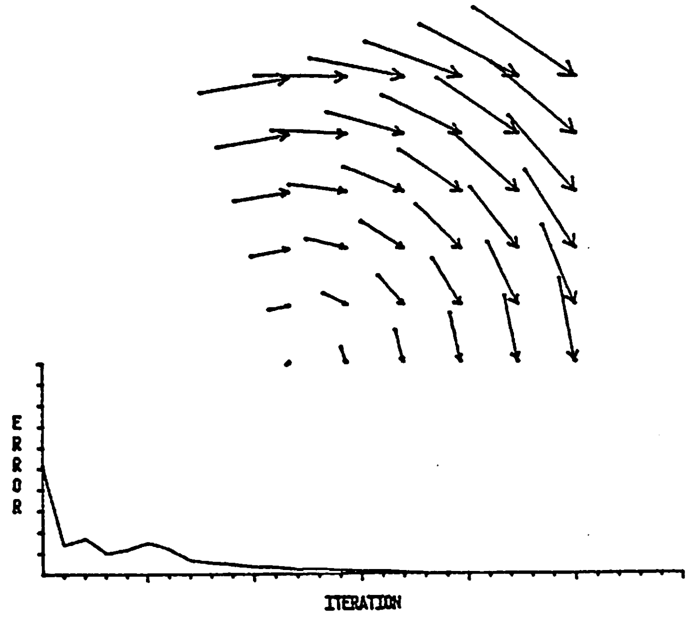
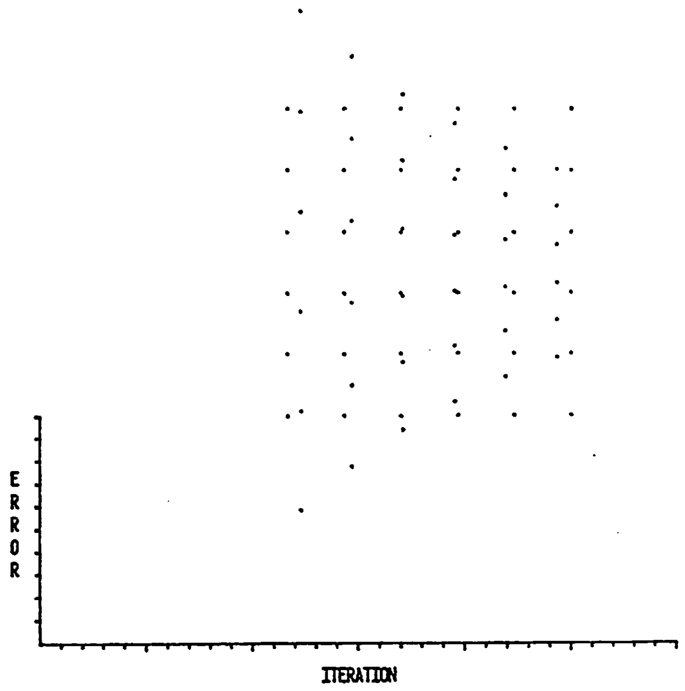
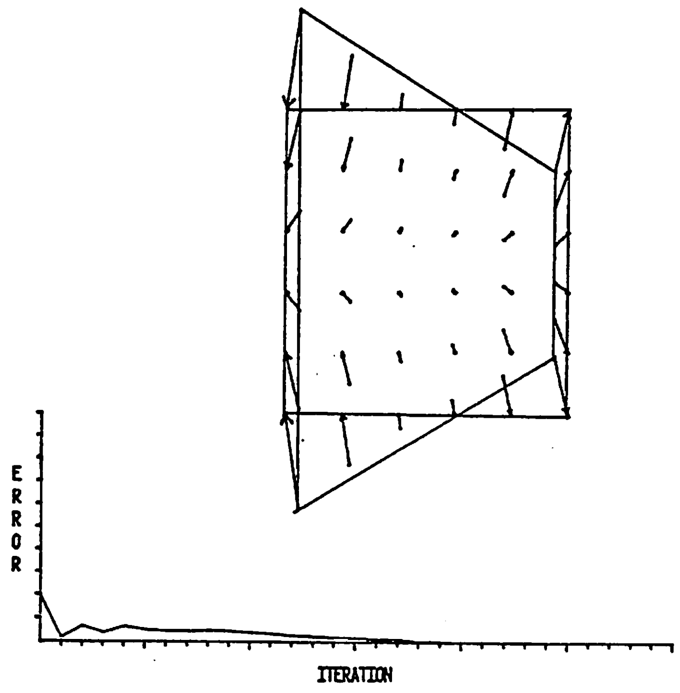


Figure 5.7. Rotation about a Line Parallel to the Line of Sight. A 6 x 6 grid of points was rotated about a point near the bottom-left corner of the figure. Note that for points in the top right-hand corner, the nearest neighbours are not the correct matches.



5.8a

Figure 5.8. Rotation about a Line Perpendicular to the Line of Sight. Figure 5.8a shows the two frames of points superimposed prior to the matching process. Figure 5.8b shows the correspondence formed. Bounding Quadrilaterals of the two sets of points have been drawn in.



5.8b

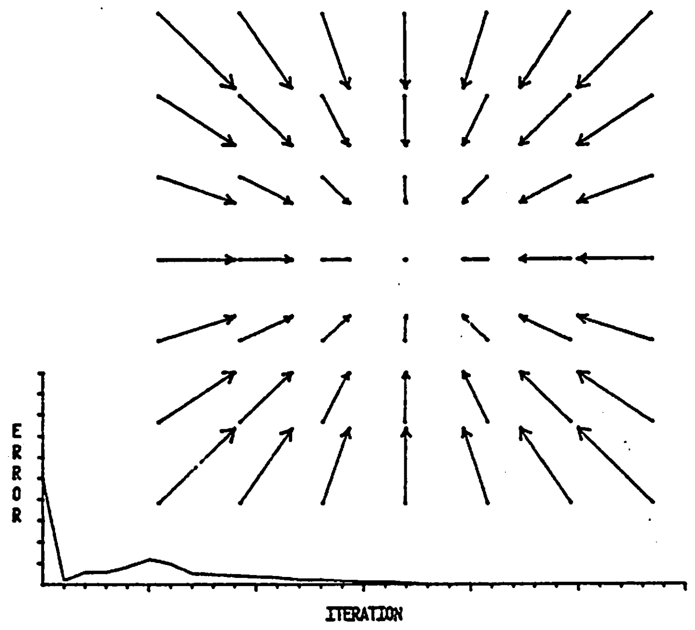


Figure 5.9. Movement along the Line of Sight. A 7 x 7 grid of points was translated away from the observer. Note that around the periphery, nearest neighbours are not the correct matches.

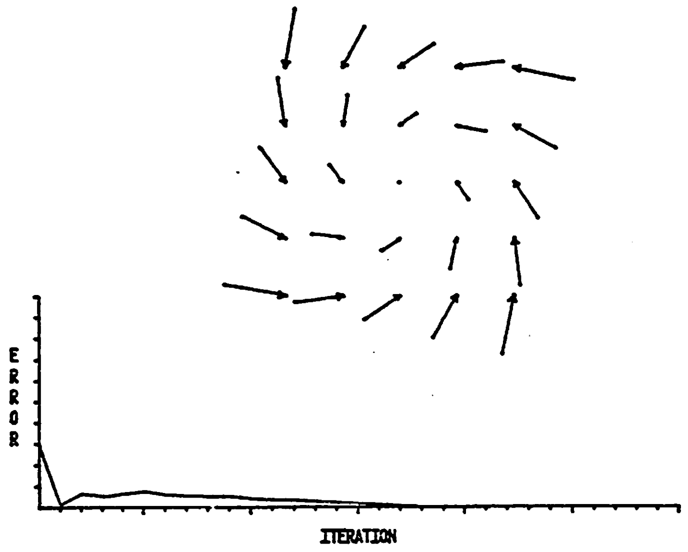


Figure 5.10. Combination of Translation and Rotation. Note again that near the periphery, nearest neighbours are not the correct matches.

Experiment 6. Comparison of Attraction Functions

We discussed two Attraction functions in Section 4: the original, discontinuous one A, and the continuous analogue $\hat{A}$. Experiment 4 was repeated using $\hat{A}$ instead of A to see if either did significantly better than the other. As before, a 4 x 4 grid of points was generated and displaced 1 IED down and to the left. Several runs were made with the same (random) feature-type as used in the corresponding run with the discontinuous function. The results are plotted in Figure 5.11. As with the discontinuous function, the continuous one improves with increasing R. However, for every value of R tried, the continuous function did as well or better than the former one. This effect is probably due to the averaging properties of the continuous Attraction function $\hat{A}(X, S)$, when there are two close candidates S_1 and S_2 in S for 'nearest' to X. By not committing itself too early, the system avoids making wrong decisions.

Figure 5.12 shows the result of a particular run in which the continuous function achieved the correct correspondence, while the discontinuous function did not. The outstanding error in the latter case was due to two points incorrectly matched. The system got 'hung-up' in this case, a possibility noted at the end of Section 5.3.

5.3.2 Experiment 7. Filling-In of Grid Positions not Occupied by Feature-Points

In the previously described experiments, there was a vector in the D-array associated with each point in Q. In the experimental scheme de-

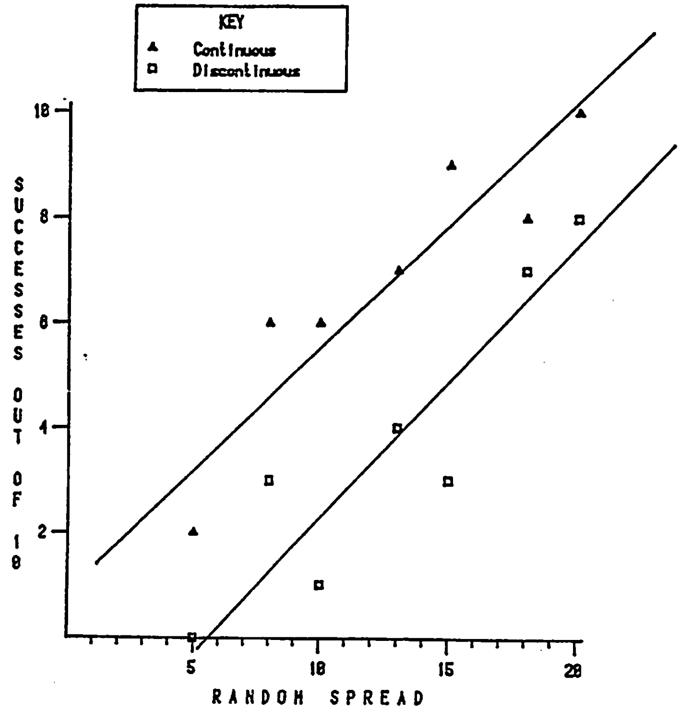


Figure 5.11. Comparison of Attraction Functions. For a variety of values of the random "feature-spread" R, ten runs of Experiment 3 were performed with each Attraction function. In every case, the continuous function converged more often than the discontinuous function. Lines of best fit of successes against R are drawn in.

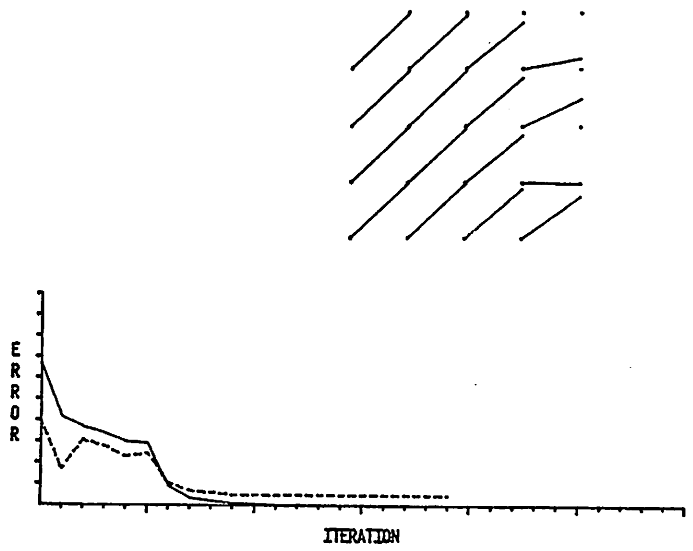


Figure 5.12. A Run Which did not Converge Correctly. Under the discontinuous Attraction function, the system stabilized in the state shown above. At those positions where the flow vectors are incorrect, the updating components are equal but opposite, and so cancel out. In the graph, the dotted line indicates the progress of the system using the discontinuous function. When the continuous function was used, indicated by the solid line, the system converged.

scribed here, there are vectors associated with grid positions in space extending beyond the feature-points in Q. Thus they serve to fill in the vector velocity field at positions where there are no stimuli, and simulate the occurrences of homogeneous regions, over which the feature-extraction process such as the one described in Section 4.3, could not find any feature-points. For example, the rectangle in Figure 5.13 produces feature-points only at its edges, although it is desired to have the flow associated with the entire surface. These vectors are influenced by their neighbours as usual during relaxation, but do not have any direct input from point-matching via the Attraction function. This may be clarified by examining the updating equations.

The relaxation equation for these 'filled-in' positions is

$$DX(i)_{t+1} = DX(i)_t + \lambda \left[\frac{\sum_{j \neq i} w_{ij} DX(j)}{\sum_{j \neq i} w_{ij}} - DX(i) \right]. \qquad (10a)$$

That is, these 'filling-in' positions are updated solely on the basis of maintaining a smooth velocity field. The other velocity field elements, namely those associated with feature-points in Q, will, as in (9), be updated by two components, viz

$$DX(i)_{t+1} = DX(i)_t + \lambda \left[\frac{\sum_{j \neq i} w_{ij} DX(j)}{\sum_{j \neq i} w_{ij}} - DX(i) \right] \qquad (10b)$$
$$+ \mu \left[QX(i) - DX(i) - A[EX(i); P] \right]$$

The definition of the attraction weights in (10) is slightly modified, however, to cope with the different cases that arise. The attrac-

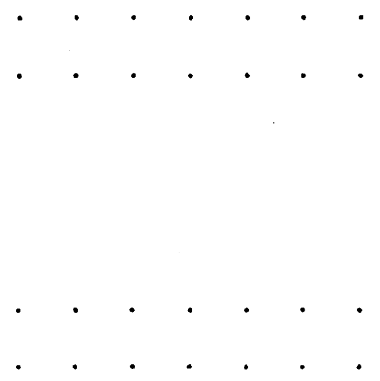


Figure 5.13. Object with Homogeneous Interior Region. The configuration of points depicted here represents a rectangle with feature-points generated solely in narrow regions along its top and bottom boundaries.

tion weight for points X_i and X_j is $w_{ij} = k_j e^{-(x_i - x_j)^2}$, where $k_j = 1$ (as before) if position X_j has direct stimulation, and $k_j = .25$ otherwise. This is to ensure that vectors at non-stimulated positions do not have as great an influence as those which do. The results of running the program in this mode are shown in Figure 5.14. The computed field varies smoothly over the object, as desired.

5.3.3 Experiment 8. Stationary Background

In this mode, different motions of two nearby objects were simulated. The set of feature-points P was generated as before, and was displaced to the positions in Q. A set of points was generated in surrounding grid-positions, however, and these points were not displaced at all. These two sets of points then represent points on two objects, one of which moved while the other stayed still. Since with two adjacent but different motions it is their relative velocities that are important, we may arbitrarily choose one velocity to be zero, without loss of generality. Since the relaxation equation (9) takes no account of whether pairs of points belong to the same object or not (this is not known to the program), points across object-boundaries will influence each other.

A critical test of the relaxation procedure is how well it performs under these conditions; results are shown in Figure 5.15. During the relaxation, the vector velocity field shows a gradient at the boundary instead of a step-discontinuity. However, this discontinuity extends over only two IEDs, and it is still clear where the boundary is. As we shall discuss in Section 5.5, the gradient will suggest a boundary to

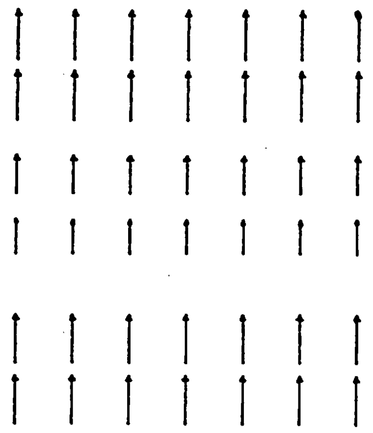


Figure 5.14. After translating the points in Figure 5.13 in the vertical direction, the flow in Figure 5.14a is generated. Note how the flow in the interior is "filled in".

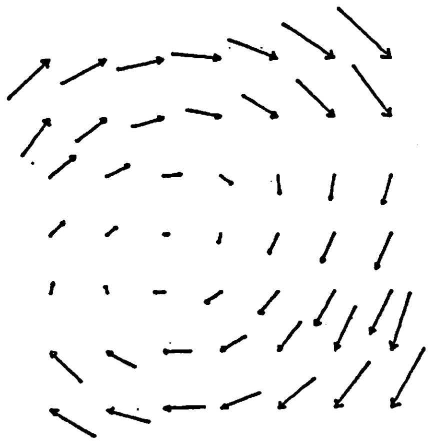
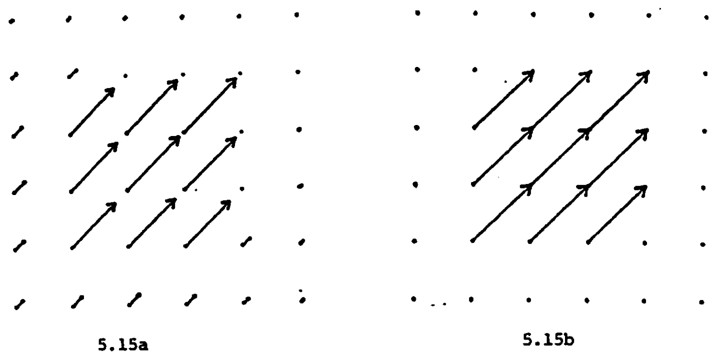


Figure 5.14b. After rotating the points in Figure 5.13 about a point to the left of and just below the middle of the figure, the flow in Figure 5.14b is generated. Note how the flow generated in the interior of the object helps visually to identify the centre of rotation.



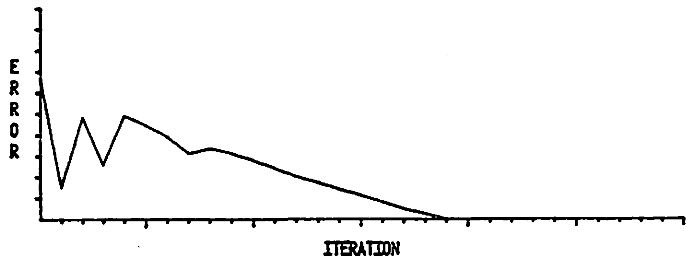


Figure 5.15. Stationary Background. A 3 x 3 grid of points was translated in front of a stationary background grid of dots. The different flows associated with the foreground and background caused the system to take longer than usual to converge, but nonetheless did not prevent convergence. Figure 5.15a shows the state of the flow after 10 iterations, 5.15b after 20.

another process; this may weaken the influence (i.e. the attraction weights) between points across the boundary, thus in turn sharpening the boundary. Again, with only two frames to consider, we can let equation (9b) take over after several iterations, and end up with a sharp boundary, as shown in Figure 5.15b.

5.3.4 Experiment 9. Feature-Points out of Register with Processing Elements (in D-layer)

In MATCH there was an element of the vector velocity field associated with each stimulus-point in the following frame Q. For several reasons it would appear preferable to use an array of vector velocity elements at fixed positions, independent of the instantaneous stimulation pattern. Firstly, instead of representing frames such as P and Q as lists of pairs (location, feature), the representation can be truly retinotopic. That is, we need only consider an array of feature-values: the position is implicit in the array index. The other major reason for the change is that the distances between velocity elements, and hence the attraction weights, are fixed and do not have to be continually recomputed. Moreover, since the attraction weights fall off very quickly with distance, the set of other elements to be considered when updating any vector velocity element can be reduced to just the immediate neighbours, instead of the whole array as before. Thus the weights in (9a) can be represented by the array in Figure 5.16 where $a = e^{-k^2}$ and k is the inter-element distance. For simplicity, the negative logarithms of the weights are given, relative to the centre point *. The neighbourhood of * can be limited by regarding each weight e^{-K} to be zero, for

8	5	4	5	8
5	2	1	2	5
4	1	*	1	4
5	2	1	2	5
8	5	4	5	8

Figure 5.16. Attraction Weights. A number K in a cell indicates that the Attraction weight of that cell with respect to the central cell * is e^{-K}. K = 0 at *.

each K greater than some K_0, thus providing a finite window around *.

We will introduce an extra layer C which will represent the updating component evaluated throught (9b). The CX(i) will be directly associated with QX(i), whereas the DX(j) are fixed in space. The density of elements in the D-layer will not necessarily correspond to the resolution in the P, Q and C layers. Therefore there may be several points in the C layer which fall within the receptive field of a particular element in the D-layer (see Figure 5.1). Thus

$$CX(i) = [\ QX(i) - \hat{A}(QX(i) - DX(i');\ P) - DX(i')\ ] \qquad (11)$$

where i' is that element of the D-array in whose receptive field QX(i) lies.

This is interpreted as follows. Point Q_i is in the receptive field of D_i, and thus uses DX(i') as its 'own' flow-vector. The Attraction function finds the closest match of QX(i) - DX(i') in P. CX(i) is then calculated as the discrepancy between QX(i) - DX(i') and this closest match.

The D-layer is updated by

$$DX(i) = DX(i) + \lambda\,(\underbrace{\frac{\sum_{i\neq k} w_{ik}\,DX(k)}{\sum_{i\neq k} w_{ik}}}_{(12a)} - DX(i)) + \mu\,(\underbrace{\frac{1}{N_i}\sum_{j\in J_i} CX(j)}_{(12b)}) \qquad (12)$$

where J_i is the set of N_i stimulus points in the receptive field of point i in the D-layer. (12b) is thus the average value represented in this area of the C layer. (12b) is taken to be zero if $N_i = 0$, in which case (12) reduces to (10).

Several results of running the MATCH program in this configuration are shown in Figures 5.17a-d. In all of these figures, the optic flow vectors are not in register with the feature-points. The motion of the feature-points is snown by dotted lines, the flow vectors by solid lines. Notice how the flow-field is filled-in in a smooth manner in all positions.

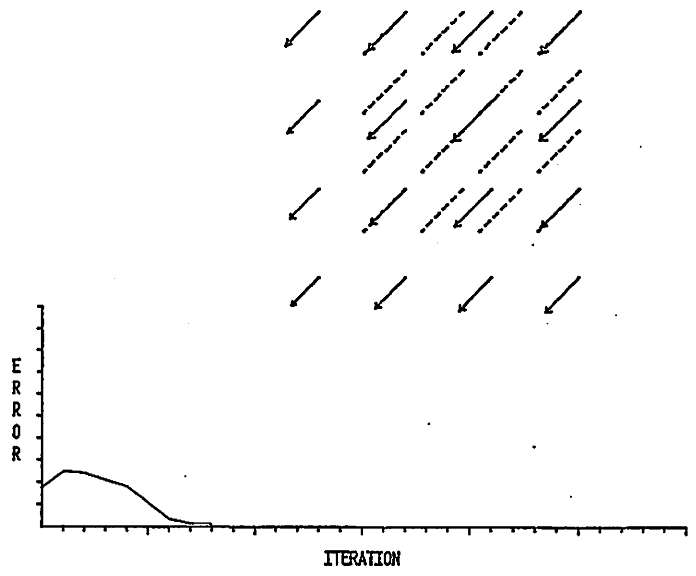
5.4 Sequences of More than Two Frames

Although motion analysis and stereopsis share the correspondence problem, motion has the added feature of a sequence of frames of data, instead of just two. These extra frames contain information that is of use to the analysis processes. We would like to be able to use the current representation of the motion extracted from early frames in the sequence to aid the analysis of subsequent frames. In this manner the computational cost per frame for subsequent frames may be greatly reduced. There are some problems which need to be considered first, in extending the model to deal with such a stream of input data.

(1) Feature-points will frequently appear and disappear as surfaces occlude and disocclude each other, or through noise in the preprocessing stages.

(2) Most of the feature-points will stay from frame to frame, and the vector velocity field will remain applicable to succeeding pairs of frames in the large part. The problem is how to use this information to speed up the relaxation procedure.

(3) If the vector velocity field was calculated to be v at loca-



5.17a

Figure 5.17. Optic Flow not Registered with Feature-Points. Figures 5.17a - d show the flow generated when the flow field is not in register with the stimulation, i.e. there is no assignment of processors to feature-points, the solid lines the flow generated.

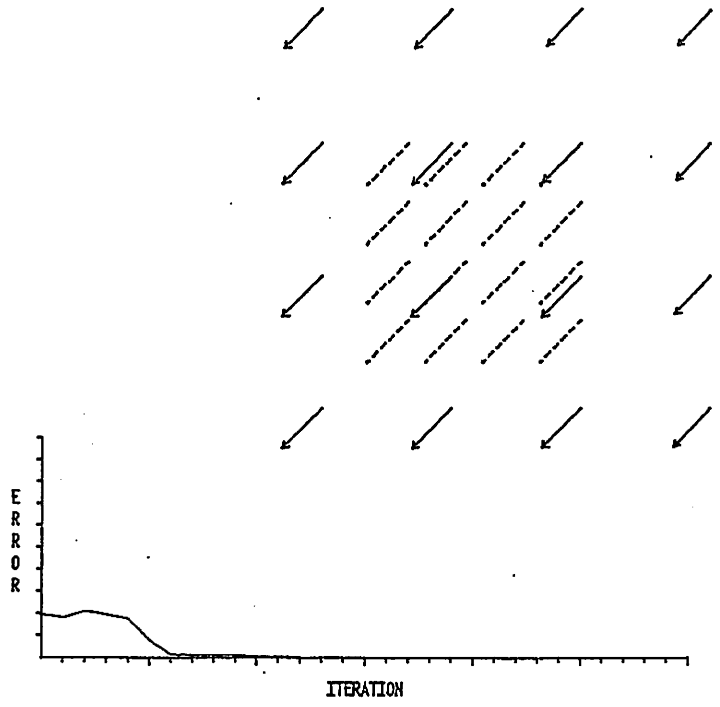


Figure 5.17b

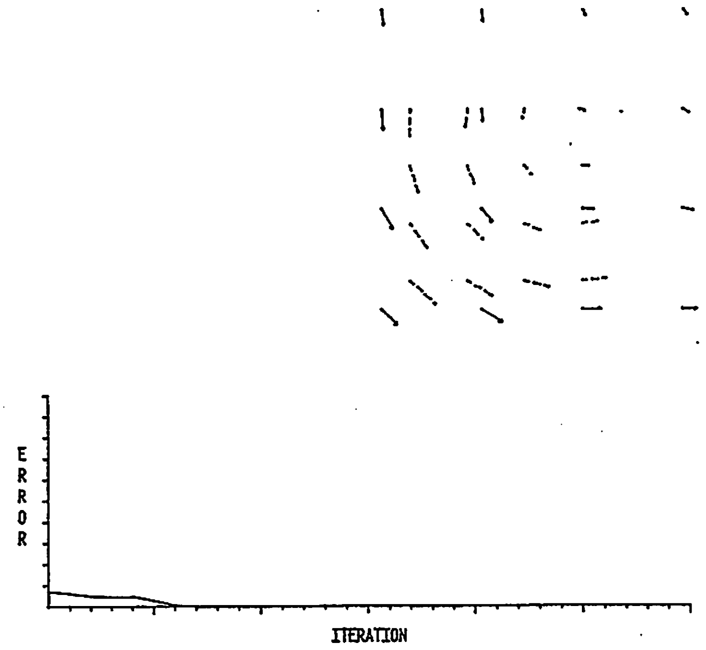


Figure 5.17c

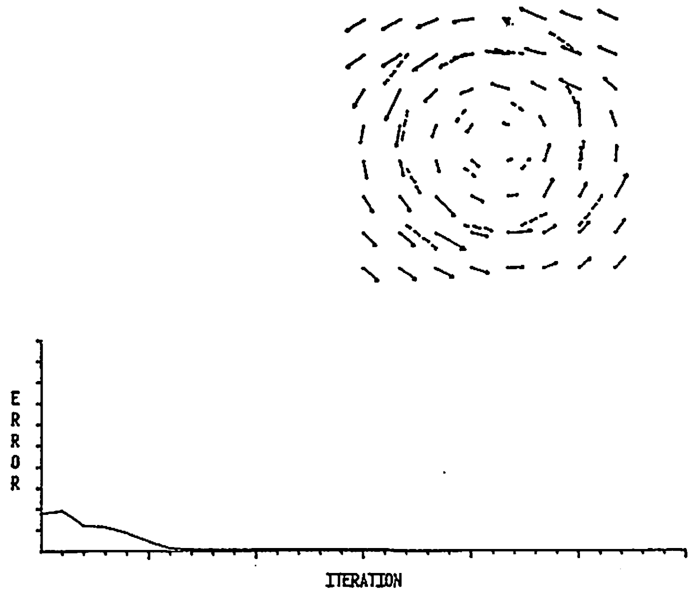


Figure 5.17d

tion x on analysis of frames F1 and F2, say, then on analysing F2 and F3 it should be found to have value v at location $x + v\delta t$ (approximately), where δt is the inter-frame time-step. That is, the vector velocity field should be extrapolated by an amount proportional to its local value.

$$v(x + v(x, t)\delta t, t + \delta t) = v(x, t). \qquad (13)$$

Suppose we have a vector velocity field D which has been computed from two or more frames. Suppose that the most recent frame was P, and that the next to be processed is Q. What we shall do is to use D to predict the position of feature-points in Q from those in P. Having shifted D between frames as given in (13), we can start off the relaxation process for the new frame with the D-layer in this initial state, instead of a quiescent zero state. The following will occur:

(i) In the case of no occlusion or disocclusion, the same feature-points are in the P and Q layers, and D will locally be close to the displacement between corresponding points in the P and Q layers. If the motion is not accelerating, or accelerates very little between frames, D will be almost exactly correct. In either case, the relaxation process should reach convergence in a very short time.

(ii) If there is occlusion, it is likely that a strip of points will appear or disappear, as one surface moves in front of another. How this may be dealt with will be described in Section 5.5.

We will describe below how the vector velocity field gets displaced between one frame and the next. The problem is that of moving the internal representation of an object to keep in step with the actual move-

ment of the object in the real world, and was discussed in the one-dimensional case by Burt (1975). We are going to treat the two-dimensional case; a more complete description of the differences between the approaches is given in Section 5.4.3. Our treatment will be entirely local, that is, each element in the vector velocity field will communicate solely with its immediate neighbours. We are not concerned with movement at the 'object-level'.

Now, occlusion raises a problem here. Suppose that two objects are moving towards each other, and the instantaneous vector velocity field is as shown in Figure 5.18. Both objects would appear to occupy the shaded region in the next time-instant. Presumably, one will occlude the other, but unless the program knows the relative depth of the two objects, it cannot predict which object will remain visible. So how does the program shift the velocity field? It turns out that it is sufficient to ignore the problem and to perform the displacement as usual (to be described below). The resultant vector velocity field in the shaded region will be an average of the two incoming fields. It will be remembered, though, that this value is just the starting value for the relaxation process. This process will now have more work to do to match up the points, but no more than in the start-up case when the velocity field is initially zero. A process which keeps track of the persistence of points over time may find that occlusion has taken place, which in turn would indicate the presence of a boundary. This is discussed in more detail in Section 5.5.

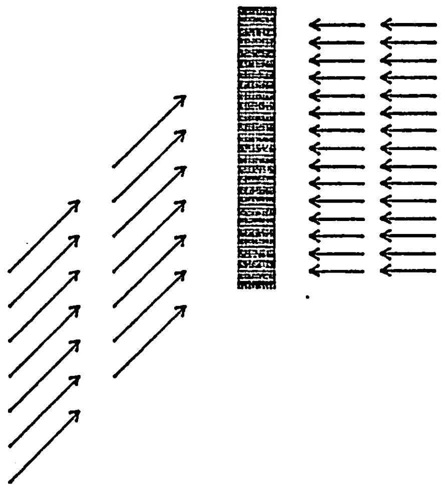


Figure 5.18. Occlusion. In this figure, the movement of two objects approaching each other is depicted. The arrows depict the flow generated by the leading edges of the objects over two frames. In the next frame, both objects will be in line with the shaded region. Which one gets occluded depends upon their relative depth.

5.4.1 Level of Resolution of Velocity Field

We have talked about the introduction of an extra layer (C in Figure 5.1) in order to handle fields of feature-points (P and Q) of different resolution than the flow-field (D). This may be desirable for computational reasons. We will discuss below the relationships between velocity levels in the different layers in this arrangement.

Let us suppose that the reduction in resolution between the C layer and the D-layer is r:1, so that r^2 pixels of P and Q fall within the receptive field of one element in the D-layer. Let us choose a velocity scale so that 1 unit corresponds to 1 IED in the D-layer per inter-frame interval (IFI). This therefore corresponds to r units per IFI at the retinal level. If our input array (which is of the same size as the P and Q arrays) is 256 x 256 pixels, and the field of view of the system is, say, 128 degrees squared, and if the frames are generated at an IFI of 1/30 second, then one velocity unit corresponds to

$$r \times 30 \times 128/256 = 15r \text{ degrees per second.}$$

If r is, say, 4 (not an unreasonable value), then 1 unit = 60 degrees per second. It will be convenient in what follows if 1 velocity unit is the largest velocity the system can keep track of successfully. Almost all reasonable motions seen by the system, whether due to motion of the environment or the camera, will be less than 60 degrees per second. In any case, we will only expect the system to track motions of less than this velocity. We may therefore regard 1 as the upper limit of the velocity in the D-layer. It is a useful strategy to restrict the dis-placement at any time to be less than or equal to 1 IED, since then each element in the D-layer need only 'talk' to its immediate neighbours.

5.4.2 Smoothing the Inferred Motion

There would appear to be a problem with generating the velocity field from the relatively small velocities frequently encountered in the real world. These velocities can best be expressed, not in number of IEDs per unit time, but rather as number of time units per IED. Suppose, for example, that an object's velocity is 1/5 at the pixel level. As a consequence of using digitization-related processes such as UNMIX (described in Section 2.1), a point on the object will be within one pixel for up to 5 time units, then move to a neighbouring pixel and stay there for a further 5 time units and so on. This is related to the effects of using large receptive fields, discussed in Section 3.2. We see then that the velocity of a point as it is tracked over the image will appear to follow the sequence

0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 ......

the average value is 1/5, as required, but it oscillates quite far from this point. We would like to smooth this sequence out somehow, so that the velocity field remains fairly constant for smooth, slow motion.

Recall equations (11) and (12):

$$CX(j) = QX(j) - A\,[\,QX(j');\,P\,] - DX(j') \qquad (11)$$

$$DX(i)_{t+1} = DX(i)_t + \qquad (12)$$

$$\lambda\,[\,\frac{\sum_{i\neq k} w_{ik}\,DX(k)}{\sum_{i\neq k} w_{ik}} - DX(i)\,] + \qquad (12a)$$

$$\mu\,[\,\frac{1}{n_i}\sum_{j\epsilon J_i} CX(j)\,] \qquad (12b)$$

(for an explanation see Section 5.3.4). If we assume that the vector velocity field has been generated and is constant over a large area, then the averaging term (12a) will be zero. The resultant equation can be written in a simplified manner as

$$\begin{aligned} D(t+1) &= D(t) + \mu\delta D(t) \\ &= (1-\mu)\,D(t) + \mu(D(t) + \delta D(t)) \qquad (14) \end{aligned}$$

where D(t) is the average of all corrections to D(t) at a point i within the receptive field J_i of that point in the C-layer.

D(t) may be regarded as the 'old' value for D, and D(t) + δD(t) the 'new estimate' through the Attraction equation. D(t + 1) is thus the weighted mean of the two values. Let us write E(T) for D(t) + δD(t). E(t) will take on sequences of ones and zeroes as discussed earlier. Let T be the operator 'one unit negative time-step' (i.e. (TE)(t) = E(t - 1)), and let μ' = 1 - μ. Then

$$(1 - \mu'T)\,D(t+1) = \mu E(t). \qquad (15)$$

So formally,

$$\begin{aligned} D(t+1) &= (1-\mu'T)^{-1}\,\mu E(t) \qquad (16) \\ &= (1 + \mu'T + \mu'^2T^2 + \ldots\,)\,\mu E(t) \\ &= \mu(E(t) + \mu'E(t-1) + \mu'^2E(t-2) + \ldots\,) \\ &= \mu\sum_{i=0}^{\infty} \mu'^2E(t-i) \end{aligned}$$

Now suppose that the velocity concerned is 1/n, so that E follows the sequence

$$\underbrace{0 \ldots\ldots\ 0}_{n-1 \text{ times}}\ 1\ \underbrace{0 \ldots\ldots\ 0}_{n-1 \text{ times}}\ 1\ \underbrace{0 \ldots\ldots\ 0}_{n-1 \text{ times}}\ 1\ \underline{0 \ldots.}$$

and that E(t) = 1.

Then

$$D(t+1) = \mu(1 + \mu'^{n} + \mu'^{2n} + \mu'^{3n} + \ldots) \qquad (17)$$

$$= \frac{\mu}{1 - \mu'^{n}}$$

$$= \frac{1 - \mu'}{1 - \mu'^{n}}$$

$$= \frac{\mu'}{1 + \mu' + \mu'^{2} + \mu'^{3} + \ldots}$$

$$= \mu' \left(\sum_{i=0}^{\infty} \mu'^{i} \right)^{-1} .$$

$D(t+2)$ will then be $\mu' D(t+1) + (1-\mu')\cdot 0 = \mu' D(t+1)$, so that n values of D in sequence starting at $D(t+1)$ will be

$$D(t+1),\ \mu' D(t+1),\ \mu'^{2} D(t+1),\ \ldots,\ \mu'^{(n-1)} D^{(t+1)} \qquad (18)$$

with $D(t+1) = \left(\sum_{i=0}^{\infty} \mu'^{i} \right)^{-1}$.

The sum of these values is clearly 1, so the mean is 1/n. We therefore see that D follows a sequence of values whose mean is the desired velocity, as does E. The important difference is that the individual values that D assumes are all much closer to 1/n than the values of E. It is clear that the closer μ' is to 1, the closer each value D takes is

to 1/n. The disadvantage with having μ' too close to 1 is that the contribution of E to D is governed by $\mu = 1 - \mu'$. If μ' is close to 1, then μ will be close to 0, so the system will be very slow to respond to changes in the motion.

It should be noted that if μ is close to 1, the μ' will quickly drop off to zero with increasing i. This will not be a problem if n is not large (say 2 or 3). In fact, the process can be iteratively applied to the sequence (18) producing sequences of values with the same mean but smaller variances.

If we are concerned with finding an optimal value for μ, then by regarding (16) as a linear filter we observe some interesting properties. Suppose that

$$E(t) = M(t) + cN(t),$$

where M is a message signal, N is noise and 1/c is the signal/noise ratio, being in the range 0 - 1. We wish to minimize the mean square error of D from the message M, i.e. to minimize

$$\text{EXPECTATION}(\Sigma\, E(t-i)\mu'^{i} - M(t+1))^{2}$$

with respect to μ', for given M, N and c. That is, given the signals M and N, we wish to find which value of μ' minimizes (19) as a function of c.

Several simulations were performed, with M = sin (ωt) and using various waveforms, such as sinusoids of frequency different from ω, sawtooth waveforms and even random noise for N.[†] While the exact values dif-

[†]These experiments were performed in conjunction with Thomas H. Probert.

fered, the qualitative results were the same in each case. Plotting μ against c showed a drastic step-discontinuity every time. For values of c close to 0, the optimal value of μ was close to 1. For values of c close to 1, the optimal value of μ was close to 0. At some value for c between 0 and 1 (depending on the precise waveforms used) the optimal value for μ changed discontinuously (see Figure 5.19). By increasing c beyond 1, further discontinuities were found.

The interpretation of these results in terms of our model for values of c near 0 and 1 is fairly straightforward. For low noise ($c \approx 0$), μ should be close to 1, so that E(t) provides the major contribution to D(t + 1). This is desirable, since with low noise, E(t) will not vary too much. In very noisy situations ($c \approx 1$) μ should be close to 0, so that D(t) provides the major contribution to D(t + 1), i.e. the system should be conservative. What is most interesting is the sudden switch in strategy when the noise increases beyond a certain level.

5.4.3 Tracking the Motion - "Moving Memories"

Burt (1975) presented a neural network which supported the movement of patterns of activity. This network was based upon the Wave Equation, and would propagate the activity in one dimension at constant velocity. He also described other networks, which had requirements that, for example, the patterns be binary, or that the motion be based upon the local slope of the velocity. We would like to remove these restrictions and develop a local process which would have the following properties. Given a layer or array of elements at each of which a two-

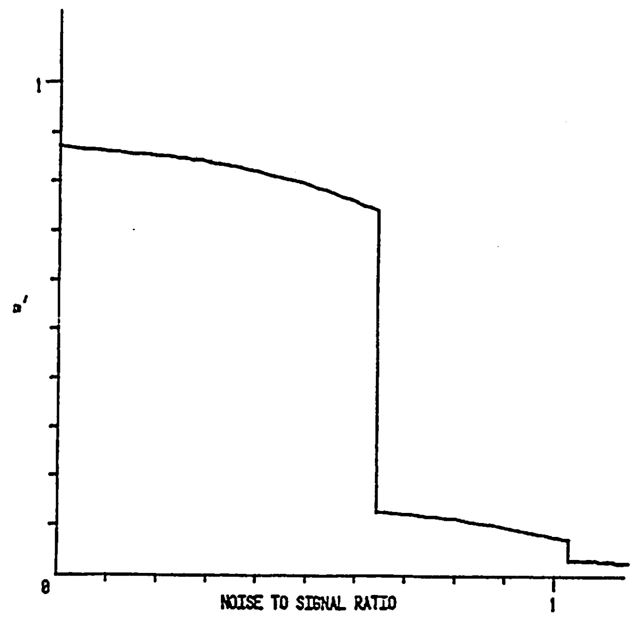


Figure 5.19. Optimization of Filter Parameter. Let D(t) be related to E(t) by the equation $D(t + 1) = (1 - m') \Sigma m'^i E(t - i)$, which is a linear filter. Suppose $E(t) = M(t) + cN(t)$, where M(t) is a message signal and N(t) noise, 1/c being the signal/noise ratio. We wish to minimize the expected value of $[\Sigma m'^i E(t - i) - M(t + 1)]^2$ with respect to m' for given c. Using various waveforms, the value of m' which miminized that value was found as c ranged from 0 to 1 and beyond. The graph of m' against c was found to have the same shape, whether the waveforms were sinusoidal, triangular, sawtooth or even quadratic; a typical such graph is shown in this figure. The sharp discontinuity occurred in every case. Note also the smaller step discontinuities for $c > 1$.

dimensional velocity is represented (our D-layer), then assuming that the velocity changes continuously over most of the layer, this local process should shift the activity by a displacement which is directly proportional to the local vector velocity. The pattern of activity in our case is the velocity field, is not binary, and the motion is to be two-dimensional. Let us assume that our D-layer consists of a rectangular grid of elements, and that each has connections with its eight most immediate neighbours. The activity at each time-step at any location will be a function of the activity at the previous time-step at that location and its neighbours. The process can be considered from two equivalent viewpoints:

(1) Diffusion view. Each cell 'sends' a certain amount of activity to each of its neighbours during each time-step.

(2) Recombination view. Each cell receives a certain amount of activity form each of its neighbours during each time-step.

These views are clearly reciprocal. We will describe the process from the diffusion point of view, although the program will calculate the new states of the system through the recombination view applied to each location.

Consider for the moment a velocity between 0 and 90 degrees to the horizontal at location A (see Figure 5.20). We will denote this velocity by VA = (Ax, Ay), and we will assume, following the earlier discussion that $0 \le Ax, Ay \le 1$. Now, we would like VA to shift entirely to

E if $Ax = 1$ and $Ay = 0$

F if $Ax = 1$ and $Ay = 1$

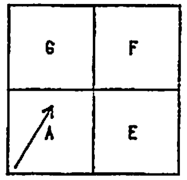


Figure 5.20. Velocity Direction in First Quadrant. A velocity at cell A in direction between 0° and 90° will influence cells E, F and G as the velocity field is propagated.

G if $Ax = 0$ and $Ay = 1$

but not to shift at all if $Ax = Ay = 0$. We would also like intermediate effects for fractional values of the components of the vleocity. We can imagine that VA gets split up and 'sent' or diffused to E, F and G in the folloiwng proportions:

$$\begin{aligned} \text{fraction that stays at A} &= (1 - Ax)(1 - Ay) \\ \text{fraction that goes to E} &= Ax(1 - Ay) \\ \text{fraction that goes to F} &= AxAy \\ \text{fraction that goes to G} &= (1 - Ax)Ay. \end{aligned} \tag{20}$$

The recombination view would say that F (still assuming velocity in the 0 - 90 degrees quadrant) would receive the following contributions:

$$\begin{aligned} &\text{from A} && AxAy \\ &\text{from E} && (1 - Ex)Ey \\ &\text{from F} && (1 - Fx)(1 - Fy) \\ &\text{from G} && Gx(1 - Gy) \end{aligned} \tag{21}$$

so that F is updated in the following way:

$$\begin{aligned} Fx(t + 1) = {} & [(1 - Fx)(1 - Fy)] \cdot Fx + [Gx(1 - Gy)] \cdot Gx \\ & + [AxAy] \cdot Ax + [(1 - EX)Ey] \cdot Ex \end{aligned} \tag{22}$$

and similarly for Fy, where all the terms on the right-hand-side of (22) are evaluated at time t.

Notice that the expressions in (20) sum to 1. It follows that these are the exact fractions of VA that are diffused to the four neighbouring cells, as expressed in (22). Moreover, if the field is constant over A, E, F and G then (22) gives

$$Fx(t + 1) = Fx(t)$$

$$Fy(t + 1) = Fy(t),$$

so that the contributions which were diffused from A, E, F and G recombine to give the same velocity as at the previous time-step. That is, interior to regions of the D-layer over which the velocity is constant, the diffusion/recombination process regenerates the velocity field precisely. At the boundaries, however, degradation occurs, and if the diffusion process were to be repeated many times in succession, the velocity field would tend to smear itself out over the whole layer. It will be remembered, though, that the purpose of shifting the velocity field is to predict the whereabouts of stimulus points in succeeding frames. The shifted velocity field will be the initial state of the D-layer in the relaxation process, which when iterated will correct the erroneous values in the velocity field. For smooth motion, then, the relaxation process will simply rebuild the velocity field at the edges.

It will be noticed that having the velocity less than 1 in the given units is an essential factor in the correct functioning of (20), since the complement of each component with 1 is used to represent the lack of motion in the relevant direction. We saw that this requirement imposed a limit on acceptable real-world velocities (60 degrees per second in the example given earlier). It turns out to be possible to relax this requirement. To double the range of acceptable velocities, for example, we merely scale the velocity down by a factor of 2 between the C and D layers, and then iterate the moving memory process twice instead of once. This will cause greater degradation between rebuildings than before, but probably still within acceptable limits.

It was supposed earlier for explanatory purposes that the velocity made an angle with the horizontal of between 0 and 90 degrees. We show below the amounts that central cell A diffuses to its eight immediate neighbours E, F, G, I, J, I, L and M for arbitrary velocities (Ax, Ay) (see Figure 5.21). We use H(x) to denote the function:

$$H(x) = \begin{cases} x, & x > 0 \\ 0, & x \le 0 . \end{cases}$$

The following are the amounts diffused from A to

E	$[H(Ax)(1 - \lvert Ay\rvert)] \cdot (Ax, Ay)$	# @	(23)
F	$[H(Ax)H(Ay)] \cdot (Ax, Ay)$	#	
G	$[(1 - \lvert Ax\rvert)H(Ay)] \cdot (Ax, Ay)$	#%	
I	$[H(-Ax)H(Ay)] \cdot (Ax, Ay)$	%	
J	$[H(-Ax)(1 - \lvert Ay\rvert)] \cdot (Ax, Ay)$	%&	
K	$[H(-Ax)H(-Ay)] \cdot (Ax, Ay)$	&	
L	$[(1 - \lvert Ax\rvert)H(-Ay)] \cdot (Ax, Ay)$	&@	
M	$[H(Ax)H(-Ay)] \cdot (Ax, Ay)$	@	

and remaining at A

$$[(1 - |Ax|)(1 - |Ax|)] \cdot (Ax, Ay) \qquad \#\%\&@ .$$

Notice that for any direction of velocity, most of these terms will be zero. The non-aero terms for the four ranges of direction are as follows:

0 - 90 degrees	#
90 - 180 degrees	%
180 - 270 degrees	&

I	G	F
J	A	E
K	L	M

Figure 5.21. Neighbourhood of Central Cell A. This figure depicts the labels used in the text to refer to the eight immediate neighbours of cell A.

270 - 260 degrees @.

For any velocity, only four cells, including A itself (indicated by the symbols in the right-hand column), will get input from A. It is easily seen that the sum of these inputs will be (Ax, Ay), that is, the current value at A.

Results with Real Data

We present in this section the results of applying some of the foregoing processes to some real data. The images were taken using a movie camera from the front seat of a car as it travelled at about 35 mph down a country road. The frames were taken at 1/18 sec intervals and digitized into a 128 x 128 grid.[†]

Four consecutive frames were examined. The first of these is depicted in Figure 5.22. Notice tha rather coarse resolution. Due to computational and storage reasons, it was decided to process only a small portion of these frames. Accordingly, a small window around the diagonal road-sign (just to the right of centre) was extracted from each frame. These four 44 x 50 windows are shown in Fibure 5.23. It is patent how ill-defined the sign's boundaries are.

Only a 20 x 20 subwindow in each of these frames was used in the subsequent analysis. The feature-points extracted from these images are shown in Figure 5.24. Here, a digit N indicates a feature-type of 45N degrees. A blank in any position indiates that no feature mask had an

[†]The author wishes to thank Thomas D. Williams for making these images available to him.

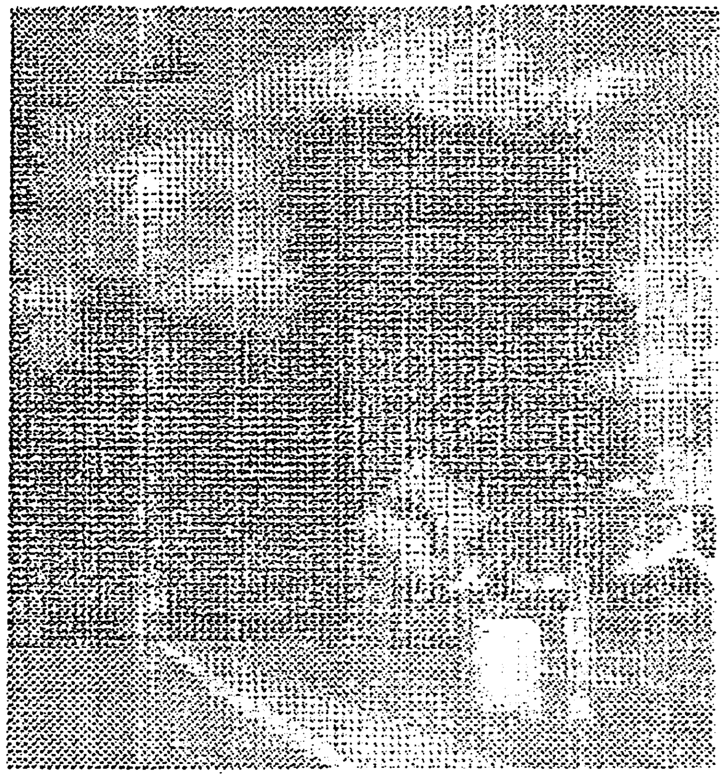
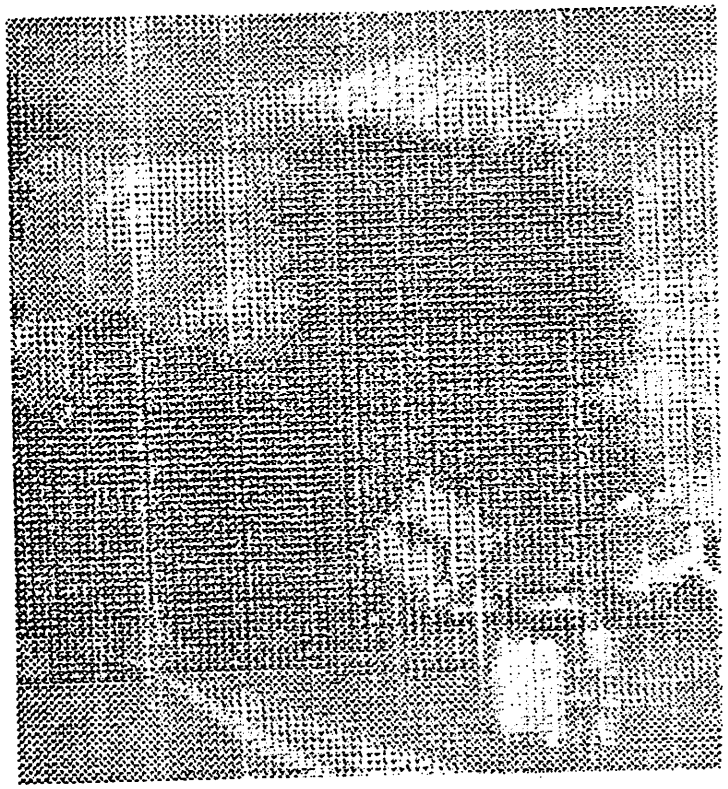
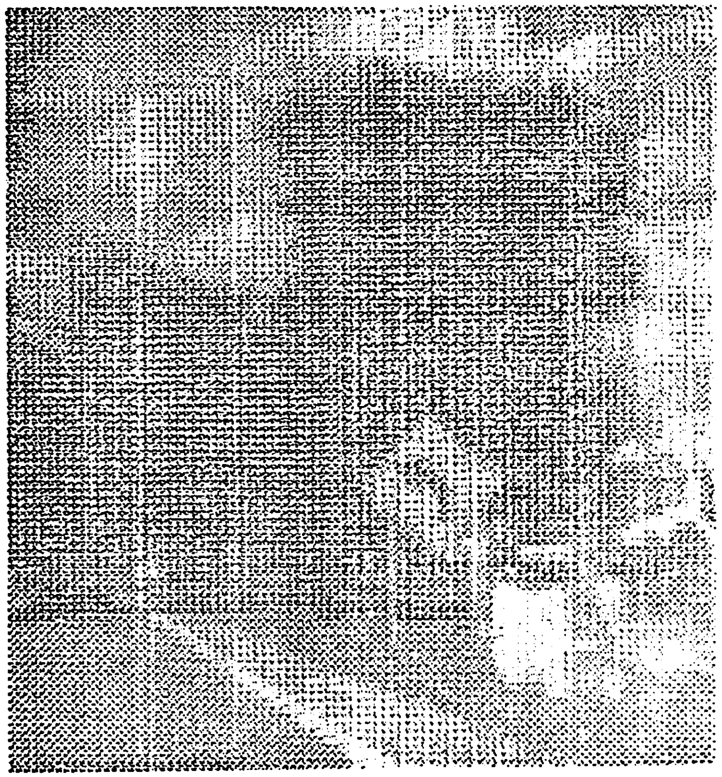


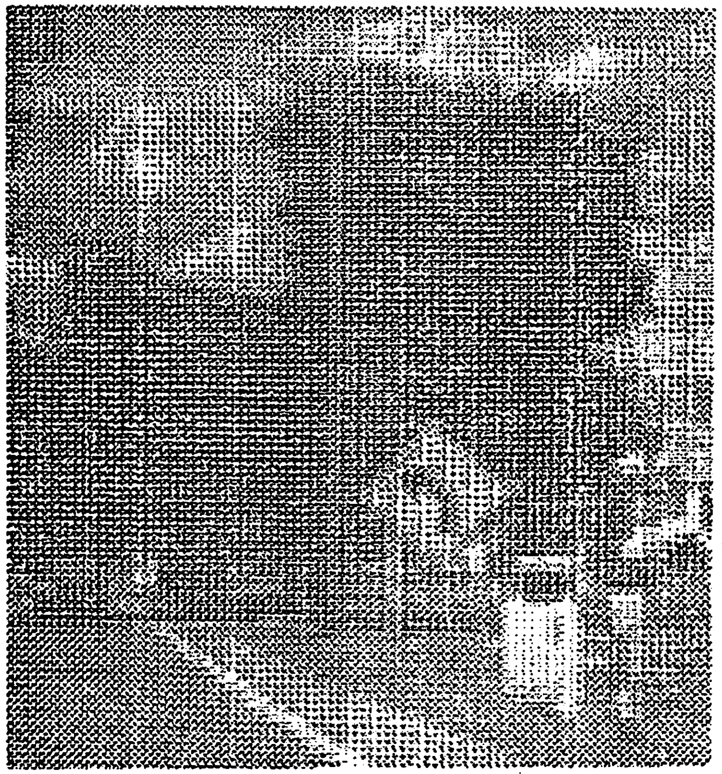
Figure 5.22. Digitized Road Scene. F.gures 5.22a - d show four consecutive frames taken at 1/18 second intervals. The resolution is 6 bits per pixel. Notice the diamond-shpated road sign just to the right of and below centre.



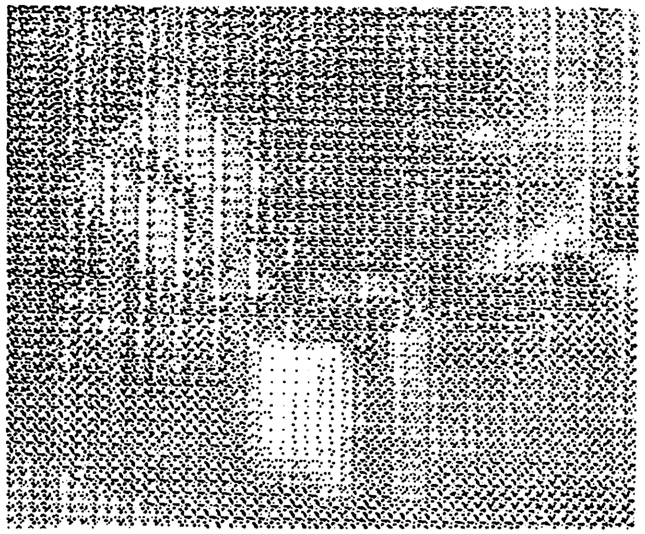
5.22b



5.22c



5.22d



5.23a

Figure 5.23. Segments of a Movie Sequence. Figures 5.23a - d show an expanded view of the road-sign in Figures 5.22a - d. (The windows are of size 44 x 50 pixels.)

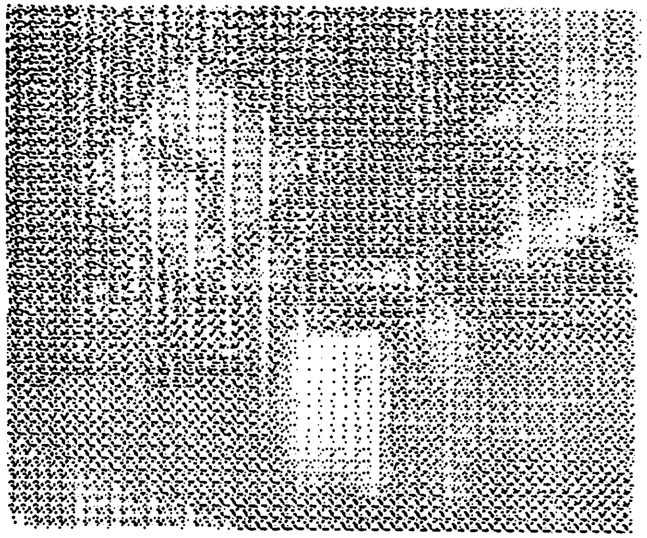


Figure 5.23b

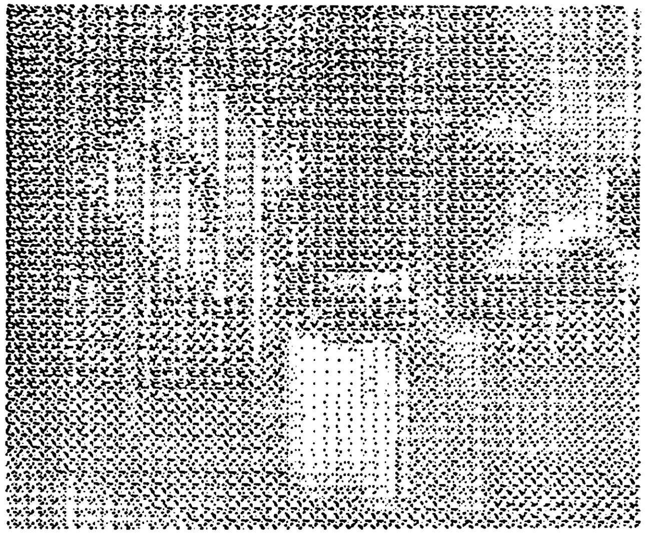


Figure 5.23c

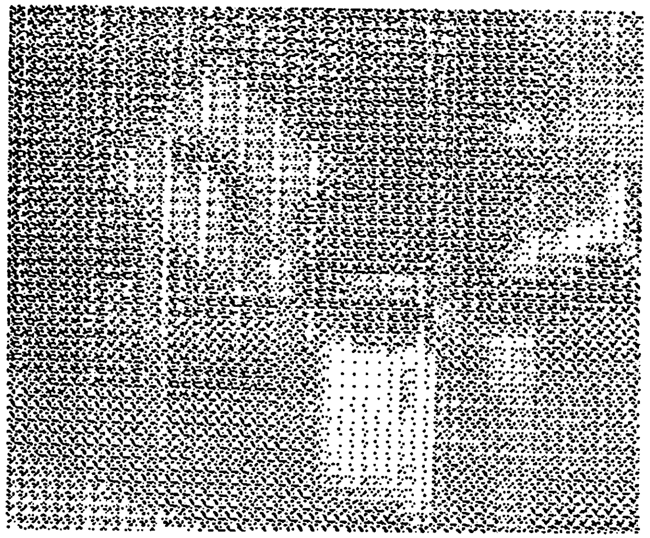
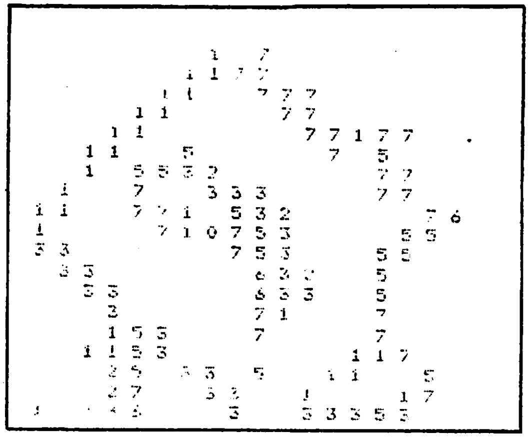
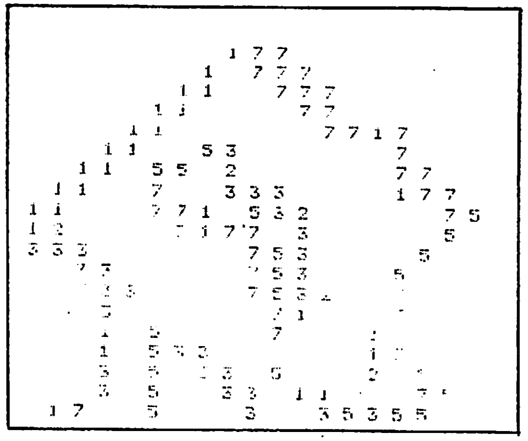


Figure 5.23d

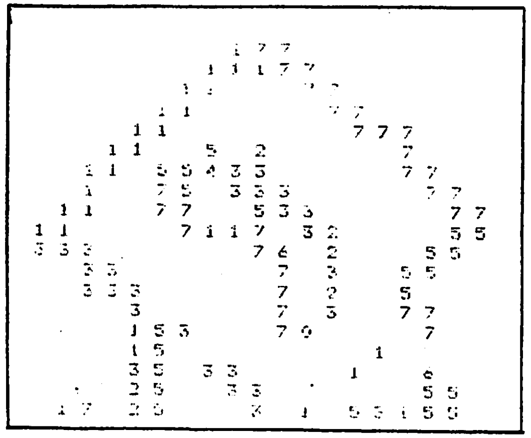


5.24a

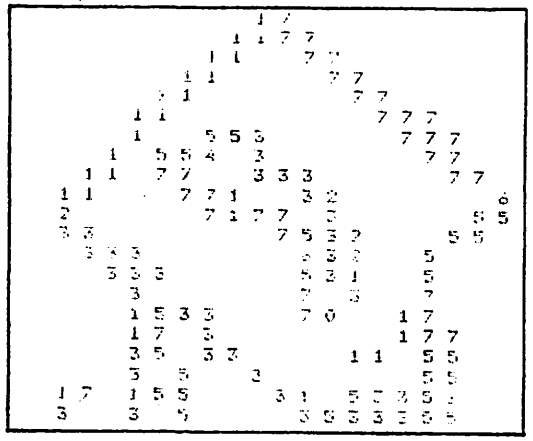


5.24b

Figure 5.24. Feature-Points. Figures 5.24a - d show the feature-points extracted in a 20 x 20 window around the road sign in Figures 5.23 a - d. A number N in the range 0 - 7 indicates a feature-type of 45N degrees. Blanks indicate the absence of feature-points at the corresponding pixels.



5.24c



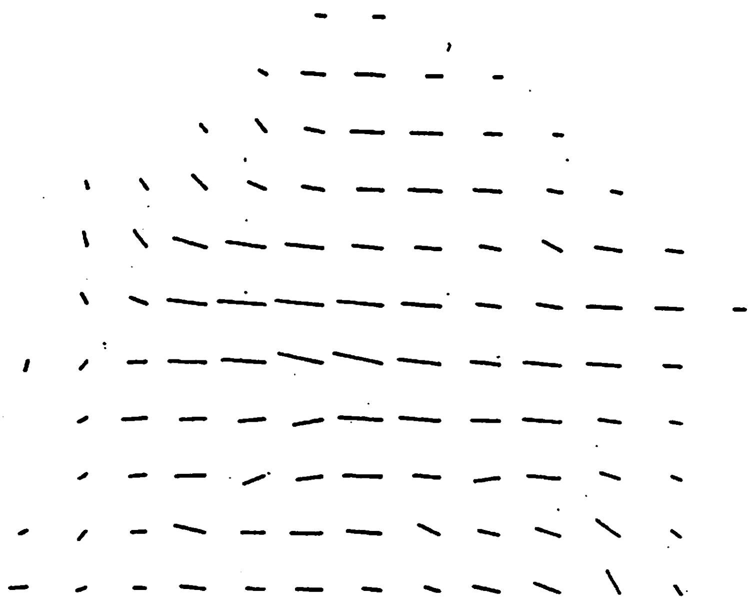
5.24d

output above the threshold, which was set to 1/10 of the dynamic intensity range.

The optic flow generated from these images is shown in Figure 5.25. Figure 5.25a shows the flow generated by attempting to match the feature points from the first two frames. In this display, flow vectors small in magnitude were thresholded out. This flow was then extrapolated and used as a starting point in the relaxation process for computing the flow between the second and third frames (see Figure 5.25b). Finally, this flow was used in the same way to generate the flow between the third and fourth frames (see Figure 5.25c).

Notice how the shape of the sign is apparent right away in Figure 5.25a. The displacement motion between frames 5.24a and b is mainly horizontal (left to right) and this is reflected in the flow vectors in Figure 5.25a. The displacement in-between frames 5.24b and c is minimal however, resulting in a much reduced flow in Figure 5.25b. We discussed earlier that the motion, when looked at locally (in the temporal sense), will appear jerky due to the digitization process; the lack of displacement here is a good example of this phenomenon.

The displacement between frames 5.24 c and d is both upwards and to the right, which brings about the flow shown in Figure 5.25c. This flow being a weighted average of the flow over the last three time-steps, is presumably a better estimate of the actual motion of the sign than the other flows in Figures 5.25a and b. It is expected that a longer sequence of frames would produce increasingly accurate flow by this method. It should be pointed out that the 'extra' material at the bottom of



(a)

Figure 5.25. Optic Flow from Real Data. Figure 5.25a shows the flow generated from the frames of feature-points shown in Figures 5.24a and b. Figure 5.25b shows the flow generated from the points in Figures 5.24b and c, using the flow in 5.25a after extrapolation as a starting-point in the relaxation. Similarly, 5.25c shows the flow generated from the points in Figures 5.24c and d, using the flow in 5.25b as a starting point.

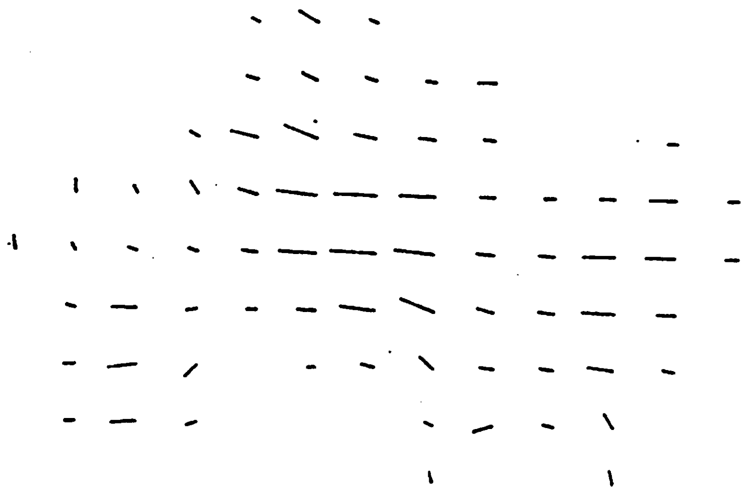


Figure 5.25b

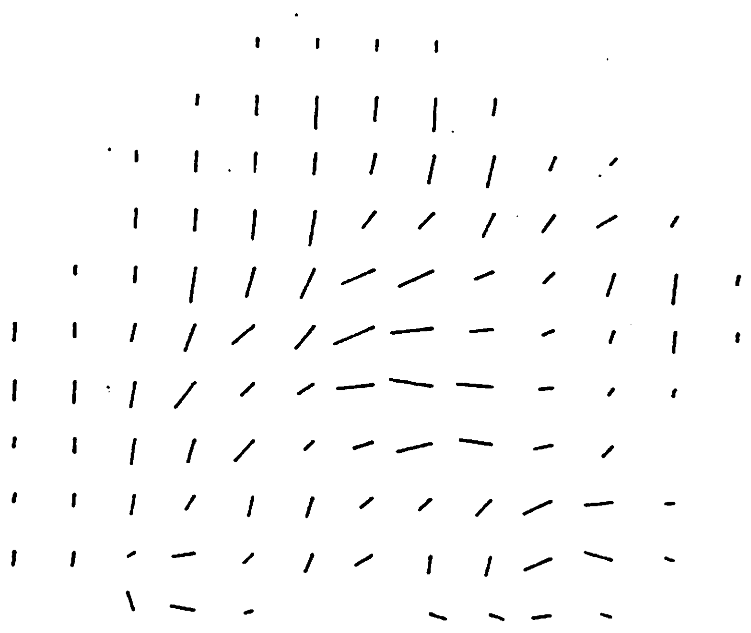


Figure 5.25c

the flow fields is due to the posts which may be seen in Figure 5.22, and whose motion is similar to that of the sign.

What is especially important to us is the computational effort required by the system to perform this analysis. In performing the computation, the system kept track of the total change in the flow field between iterations. This change is graphed in Figure 5.26. The vertical axis represents the change in the flow field, the horizontal axis the iterations of the relaxation process. Introduction of a new frame is indicated by an arrow under the horizontal axis.

Two things are clear. Firstly, the system required a great deal more computational 'effort', as measured by the area under the graph, in generating the optic flow from the first two frames than in maintaining and updating it in the subsequent frames. The second point is that many fewer iterations were required on presentation of the third and fourth frames before the activity reached a 'suitably' low level.

5.5 Occlusion

Occlusion is potentially the most difficult aspect of the correspondence problem since when it occurs, no correspondence exists for the points concerned. In this section, we address two issues. Firstly, we discuss how our MATCH process may handle occlusion, that is, how the disruptive effect of occlusion on its operation may be minimized. secondly, we look at ways in which a machine vision system may make active use of the occurrences of occlusion for purposes of segmentation. Throughout the following discussion, the principles that apply to occlu-

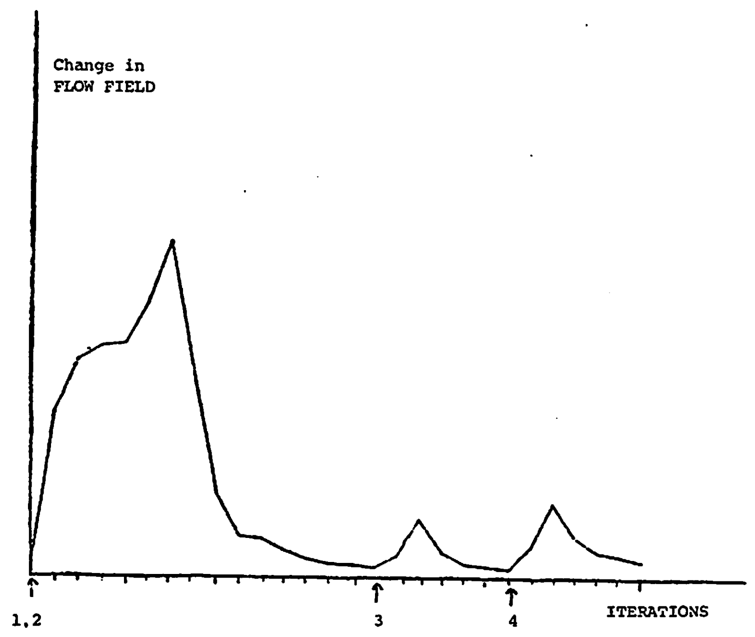


Figure 5.26. Computation Cost. This figure shows the computational cost of processing the four frames of real data. The iterations of the relaxation process are marked along the horizontal axis of the graph. The vertical axis shows the total change in the flow field between iterations. Arrows on the horizontal axis show the points at which new frames were processed. It is clear that (1) less 'effort' is expended per iteration, and (2) fewer iterations are required for processing the third and fourth frames than the first two.

sion apply equally well (but, where appropriate, in reverse) to disocclusion.

5.5.1 Occlusion and the Matching Process

It was stated at the beginning of this chapter that, for the time being at least, the existence of occlusion would be acknowledged but ignored. The rational for this will be repeated here. Consider any point in any frame of the moving-image data. Even in a complex environment, it might be expected that the point would continue to be present for a number of frames, especially at high sampling-rates. If the point gets occluded, then it might also reasonably be expected that the point would stay hidden for a number of frames. Thus over an extended period of time (frames), there will only be one time at which an 'incongruity' appears at that spatial location, i.e. when a point is present in one frame but absent in the next.

Now, in the case of processing real data, noise and digitization produce a 'pseudo-occlusion' effect; points may tend to appear and disappear, although there are no boundaries present. The system we have developed can easily 'absorb' these temporary inconsistencies, for two reasons. Firstly, as we saw in Section 5.4, many fewer iterations of relaxation are needed to process frames 3 and beyond than are needed for the first pair of the sequence. Thus the system will barely dwell long enough on the inconsistency to be disturbed by it. [N.B. We will later discuss the detection of occluding edges by a separate process.] Seccondly, it will be remembered that each element in our D-layer is updated by components from several elements in the C layer, of which the occluded point provides a single input (see Figure 5.1). This local averaging process automatically counters small local variations. The fact that we do not need to associate a processor to each feature-point, and that we can maintain the optic flow representation at interior regions of an object where there are no feature-points, leads to the success of the process.

5.5.2 Segmentation from Occlusion

In Chapter II we performed static segmentation of images by finding sharp gradients in the intensity over the images. In Chapter IV we observed that optic flow was a particularly useful representation of dynamic images because discontinuities in the flow occur at the boundaries of objects. We may now add a third dimension to the segmentation process by using occlusion to indicate the existence of boundaries, too. We should note that segmentation may be performed by region extraction techniques (see, e.g. Nagin 1979, Ohlander 1975). These are static image techniques, and suffer from the same problems as our static segmentation process.

The relationships between the three segmentation processes are depicted in Figure 5.27, which is a partial flow diagram of a dynamic scene-analysis system. We will note that the three kinds of boundaries formed obey certain subset relationships, and that they are all produced by certain kinds of discontinuities.

Static grey-level differentiation produces lines representing not

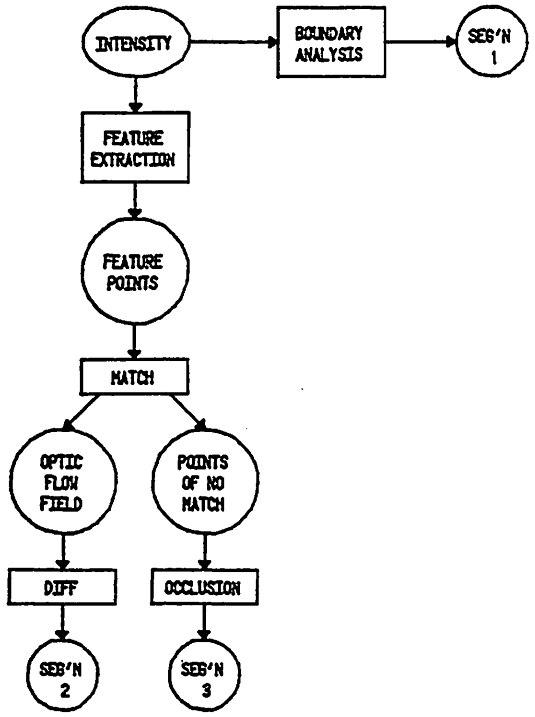


Figure 5.27. Segmentation and Motion. In this flow diagram, three segmentation processes are depicted. (1) Static image segmentation via boundary analysis. (2) Segmentation by differentiating Optic Flow. (3) Segmentation formed by detecting sites of occlusion.

only the boundaries of objects, but also boundaries of texture elements, (artificial) patterns and shadow edges, all on the objects' surfaces. These latter kinds of edge are indistinguishable from object boundaries, failing any semantic knowledge. In fact, if the background of an object is of the same intensity as the object itself, then at least that portion of the boundary of the object that is adjacent to the background in the image will be indiscernible. Thus there are two drawbacks to using static segmentation to find object boundaries: (1) extra, unwanted lines will be found, and (2) some wanted lines will not be found.

The second of these problems can be overcome by using discontinuities in the optic flow to indicate object boundaries. As all markings on an object will move with the same velocity (at least locally), not only will they not produce spurious edges, but will aid the analysis by producing 'feature-points' (see Section 4.1), which are used to generate the optic flow.

In general, this method does not suffer from problem (1) of static image differentiation mentioned above. However, in the (presumably rare) cases where an object at distance d moves with velocity v, and behind it another object at distance kd moves with velocity kv, then they induce the same flow on the retina, and so the boundary of the nearer object will not be found. This possibility can usually be ignored.

The occlusion process produces boundaries also. As one object moves in front of another, feature-points on the further object become hidden. This phenomenon gives rise to discontinuities in the matching process itself, since the continued matching of these points from one

frame to the next gets interrupted. We mentioned earlier that 'pseudo-occlusion', i.e. the disappearence of features due to non-occlusion processes, can be handled well by the system. Pseudo-occlusion, due mainly to noise and the effects of the digitization, appears in an essentially random fashion in the data. Occlusion boundaries occur in a systematic manner, both spatially and temporarily, and it is this property that allows their useful exploitation.

Just as boundaries formed by differentiating the optic flow, i.e. depth boundaries, form a subset of grey-level discontinuities, so the occlusion boundaries form a subset of the depth boundaries. Occlusion boundaries occur at the leading edges of moving objects (and disocclusion boundaries at the trailing edges). Those boundaries of objects which lie parallel to the direction of motion, or nearly so, will not cause occlusion, so will not become evident in this way.

Let us examine the nature of occlusion edges. In one frame, they all appear in the form of (not necessarily linear) sequences of locations where the matching process fails. By using grouping techniques, such as relaxation, these locations may be linked to form extended boundaries. This process will be hampered by the fact that, unlike edges derived from grey-level differentiation, these edges will lack local direction components. This may be remedied by considering the occlusion process over more than one frame.

Although a particular feature-point will, in general, only disappear once per occluding edge, neighbouring feature-points in the direction of the motion will disappear in subsequent frames. By letting information from the two consecutive frames interact, the local orientation of the edge may be deduced. However, this computation is not direct, for the following reason.

Let E be an occluding edge locally of orientation α and moving with velocity v in direction θ. Let P_i and Q_j be feature points occluded by E in two successive frames P and Q. Suppose that the line joining them is of length d and at orientation β. Then

$$v \sin(\theta - \alpha) \geq d \sin(\alpha - \beta)$$

(see Figure 5.28). The closest such Q_j to P_i will satisfy $\alpha \approx \beta + 90°$.

We have then

$$v \sin(\theta - \alpha) \approx d$$

or

$$v \cos(\theta - \beta) \approx d.$$

d and α will be known locally, but v and θ may only be found with respect to each other. v and θ may be determined explicitly by using co-operating processes from adjacent portions of the image. This problem was discussed in detail in Section 3.2.

5.3.3 Merging of Representations

We have discussed three boundary detection techniques, and have seen their relationships. The major problem of integrating the results of these different processes remains. Three possible approaches will be outlined below. We will call the result of the static segmentation SEGMENTATION 1, the result of differentiating the optic flow SEGMENTATION 2, and the output of the occlusion process SEGMENTATION 3. The

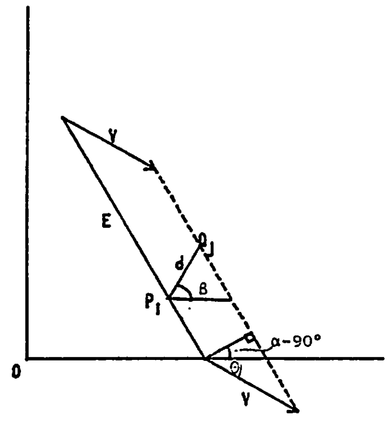


Figure 5.28. The Occlusion of Two Feature-Points. Edge E at orientation α is moving with velocity V in direction θ. Two points P_i and Q_j distance d apart are occluded. For equations, see text.

processes will be called PROCESS 1, PROCESS 2 and PROCESS 3 respectively.

We should note that PROCESS 3 is expected to form a subset of all real object boundaries, that is, every boundary produced by this process is a real boundary, but it might miss some. SEGMENTATION 1 will only be an approximate superset of the desired boundaries, since as well as having superfluous texture-based edges, some object boundaries will be missing. SEGMENTATION 2 will be expected to be close to the desired segmentation.

(1) Feedforward (see Figure 5.29). The boundaries extracted by each process will be fed into an additional process which will attempt to find the 'best' segmentation. This process would probably form the union of SEGMENTATIONs 2 and 3. Any obvious gaps or discrepancies may be filled in by reference to SEGMENTATION 1.

(2) Feedback (see Figure 5.30). The outputs of SEGMENTATIONs 1, 2 and 3 will be fed back into the processes producing SEGMENTATIONS 2 and 3, 1 and 3, 1 and 2. The three processes will then 'try again', using the information from the other processes as guidance. For example, boundaries present in SEGMENTATION 2 and/or 3, but not 1, might cause a local lowering of a threshold in PROCESS 1 allowing boundaries to be formed there that were previously rejected. In addition, the lack of boundaries in SEGMENTATIONs 2 and 3 might cause a raising of thresholds. This procedure might be repeated either until a consensus is reached, or for a certain number of iterations.

(3) Inter-Process Cooperation (see Figure 5.31). This may be regarded as a variant of approach (2) above. In this procedure, the three

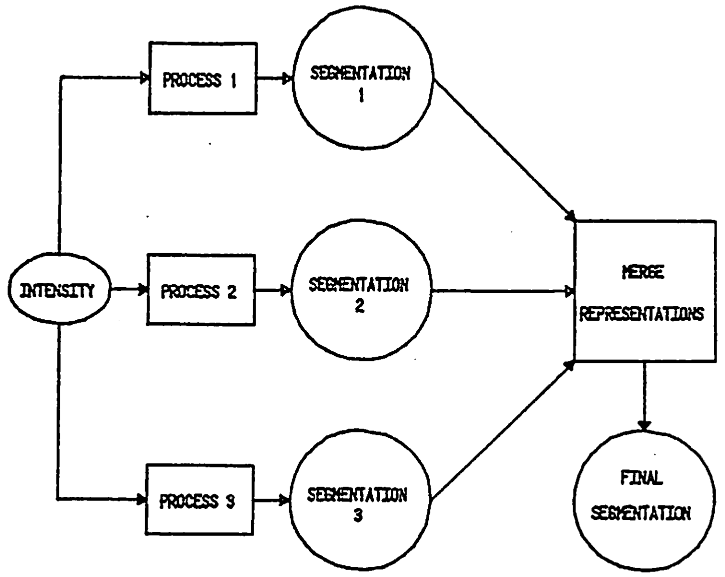


Figure 5.29. Feedforward. Segmentation formed by merging the results of three separate segmentation processes.

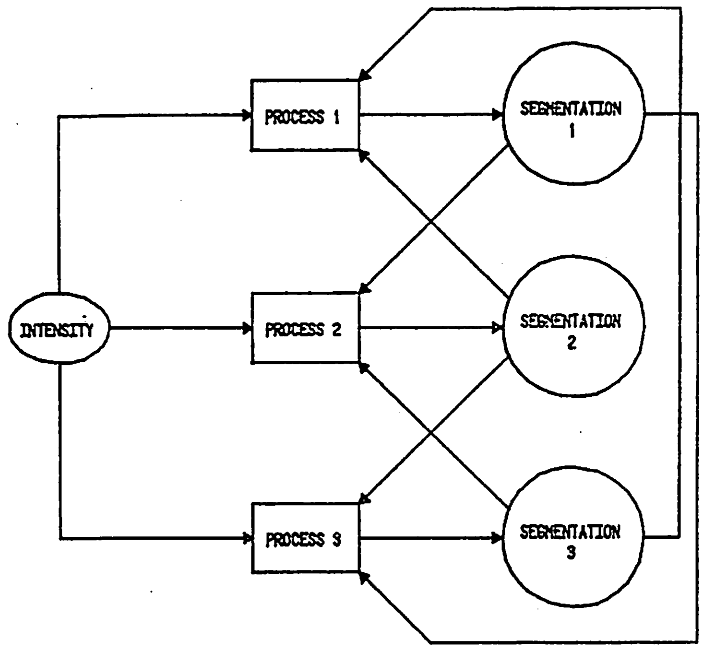


Figure 5.30. Feedback. Three segmentations are formed by feeding back each of three separate segmentation processes to the other two.

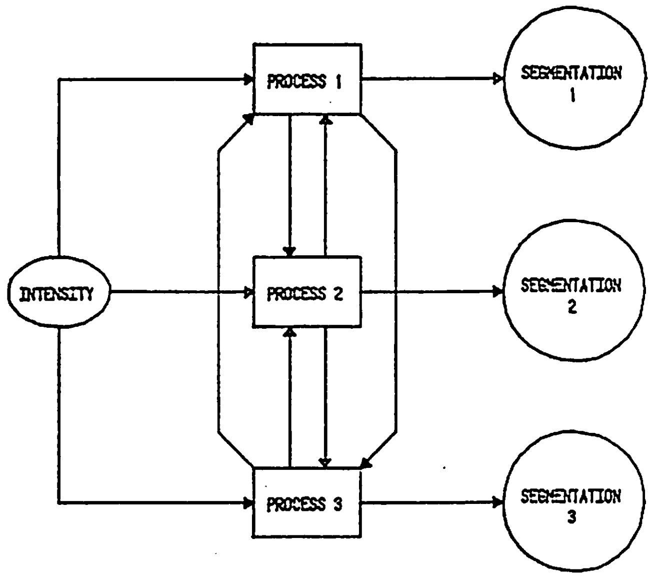


Figure 5.31. Inter-Process Communication. Final segmentations are formed by letting the three individual segmentation processes cooperate while forming the segmentations.

segmentation processes operate concurrently and intercommunicate while doing so. Thus, rather than making committments and then changing them, as in (2) above, the committments will only be made under consensus (or maybe, under a condition of 'no contradiction'). One of these processes, possibly PROCESS 2, may be regarded as the basic segmentation process, which is influenced by the other two, so that the ultimate output of this process will be taken as the output of the entire system.

Whichever approach is taken, but particularly in (1) Feedforward, the problem of 'merging representations' will have to be attacked at some level. Some preliminary work has been reported (Prager et al. 1977) but altogether this problem has so far excaped much serious attention, in the literature at least. This problem consists of two parts. Suppose that two segmentations S1 and S2 are to be merged. The first problem occurs when S1, say, contains a line-segment in a region of the image, while S2 does not. The second problem occurs when they both produce segments, but due to the different ways in which the processes operate, the line-segments may not be precisely aligned.

The second problem may be solved by allowing a match to be made if the line-segments lie within a pixel or two of each other. Intersecting lines of similar but different orientations (say 10° difference) may cause difficulties.

The solution to the first problem depends on many factors, for example whether it is known if either process generates a strict subset of the actual object boundaries. If not, then assigning confidence measures (see Section 2.5) may be helpful. Alternatively, only those

boundaries occurring in just one of the segmentations that are needed to cause continuation of the common boundaries may be selected. Possibly a scheme may be developed which involves relaxation based upon the confidences of entire lines (as opposed to the relaxation described in Section 2.4 which is based upon the local confidences, i.e. the local edge-strengths).

Altogether there is a large amount of work which remains to be done before outputs of occlusion processes can be used. If the feedforward approach is used, then the representation-merging process must be developed. If either of the feedback approaches are used, then the individual segmentation-generating processes need to be redesigned in order to take advantage of incoming information. It is hoped that the above analysis provides a useful basis for further work.

C H A P T E R VI

SUMMARY AND FUTURE DIRECTIONS

6.1 What Have We Achieved?

In Chapter I we introduced the subject of dynamic scene analysis and discussed many of the inherent problems. We looked at a number of recent investigations in this field and the related fields of stereopsis and change detection. We saw that relatively few of the systems developed to this date used real data, and those that did generally had no mechanism for dealing with rotation and/or occlusion. This motivated the evolution of our system (described in detail in Chapter V) which, through the maintenance of a low-level representation of the on-going motion, can accommodate any non-jerky motion of real data.

In Chapter II we described a system for the segmentation of static scenes. This system consisted of a sequence of processing steps, grouped into four stages. Our major contributions may be briefly summarized as: two preprocessing algorithms to correct for errors in digitization, a very simple differentiation operator, relaxation using case-analysis, an elaborate statistical confidence measure and a simple heuristic equivalent of that measure.

We looked at some of the recent neurophysiological data on biological visual systems in Chapter III. We found evidence for three or more distinct visual systems, and suggested that a comparable division of function might be profitable in machine-vision systems. We also explored the use of population encoding as a device for achieving high

resolution in any of several sensory modes.

In Chapter IV we described a concept central to our motion-analysis system: optic flow. We examined in considerable detail the information available in the optic flow generated under linear and rotational motion. We saw how the analysis provides an excellent test for possible collision, and discussed how segmentation may be performed on flow-fields. We looked at a method for resolving the translational and rotational components of the motion of an object from its optic flow, and flow might be filled-in in locations in the visual image where lack of texture and boundaries cause no flow to be induced directly.

Realizing that optic flow may be generated by matching-up points in successive frames, we developed a mechanism for extracting these points. A set of masks was constructed which would act as detectors for edges and corners at various orientations. The output of these detectors was a set of feature-points with associated feature-types, related to the edges' orientations. To aid in the determination of the best match in one frame of a feature-point in another, a new function called the Attraction function was developed. Both continuous and discontinuous versions of the function were defined. The application of the Attraction function to psychophysical phenomena was discussed.

The efforts of the first four chapters culminated in Chapter V. In this chapter we presented a series of programs called MATCH for the matching of feature-points after they had undergone various motions. The correspondence formed by the matching process was represented by an optic flow field. This field was generated by a relaxation process in which were embodied certain consistency conditions which the flow field must obey.

After demonstrating the success of the MATCH process on simulated data, we turned our attention to real data. We firstly showed how the optic flow field may be extrapolated in order to predict the positions of objects in future frames. We then applied the feature-extraction process to a sequence of four frames of moving-image data and used the relaxation process to generate the optic flow. We extrapolated the flow generated from earlier frames to produce a starting-point for the relaxation process applied to subsequent frames. We demonstrated that this assistance from previous analyses greatly cut down the computational effort required to process the later frames. This was taken as a substantiation of our claim that there is more information for segmentation in dynamic scenes than in static scenes.

6.2 What Remains to be Done?

During the course of the work described in this thesis we opened up or extended several areas which deserve further exploration. These areas were felt to be outside or beyond the scope of this thesis, but will be outlined now for completeness.

The static segmentation system described in Chapter II consisted of a series of modules, most of which were designed to be the minimal functional implementation of the embodied ideas. The differentiation operator, being so small, could only capture gradients extending over two or three pixels, although it was shown to work very well on several scenes.

If it is desired to detect very wide gradients, then sets of masks of different sizes could be used, but careful attention should be given to taking care of multiple, equivalent edges. An alternative approach is to extend the relaxation scheme. Presently, there is no influence on the edge to be updated by edges parallel to it. Furthermore, parallel and perpendicular continuations of an edge are taken *a priori* to be equally likely. While these have been shown to be entirely reasonable choices for the current implementation, they are suggested as areas for further investigation in more complex systems (cf. Hanson and Riseman 1978a).

The matter of assessing segmentations was also discussed. At present there are no standards by which segmentations may be judged either absolutely or in comparison with others. This is probably at least partly due to the immense number of boundaries in any natural scene. Possibly only with the aid of machine interpretation systems can the effectiveness of a segmentation system be determined. However it is achieved, such a methodology would be extremely useful to the development of scene analysis systems.

In Chapter III we looked at biological visual systems with a view to modelling machine vision systems around their biological counterparts, in particular their multi-channelled nature. This should be regarded as part of an on-going effort, since new neurophysical data are revealed each week. Such data may also throw more light on the problem of biological velocity detectors. Do they exist explicitly, rather than being a by-product of some other mechanism, and is their use widespread, or do most animals get by without them entirely, and if so, how? Also, do animals extract optic flow from moving images, i.e. are there neural layers containing representations of the optic flow fields, we found so useful?

Our relaxation equations developed in Chapter V contained two components, each providing a contribution to the updating of the local flow field based upon certain consistency conditions. These contributions were weighted by factors λ and μ, which turned out to be equivalents of time-constants in the associated differential equation. It was discovered that the performance of the system (i.e. time to convergence) was improved by varying these factors dynamically. However, an exhaustive determination of the influence of these factors (or the general problem of setting parameters in a relaxation paradigm too complex to be handled mathematically) remains to be accomplished.

Further work should be undertaken on incorporating other of the consistency conditions listed in Chapter IV into the analysis. For this and other reasons the system should be extended to maintain high-level constructs such as 'object'. That is, not only should the system use common motion to generate the flow, but should apply a segmentation process based upon common motion (differentiating the optic flow) in order to derive hypotheses of objects. Then the system should be able to track occluded objects, and anticipate their re-appearance. This would require the invocation of an object-schema to 'own' the space the object occupies, or is thought to occupy (cf. Burt 1976).

We closed Chapter V with a discussion of how results of different

segmentation systems may be merged. This remains a very difficult problem, and one which deserves further attention.

Throughout this thesis we have introduced algorithms for the handling of subproblems in the areas of static and dynamic scene analysis. Indeed, in Appendix IV we present two algorithms which found no place in the body of the thesis, but were felt to be potentially useful. All of these algorithms have been phrased in terms of parallel computations in layered, retinotopic systems. Due to the great complexity of scene analysis, the vast mass of data in a visual scene, and the fact that parallelism exists in biological systems, we believe that this is the most profitable approach. Martin and Aggarwal (1978) in their survey of motion-analysis systems call for more parallelism: we hope that this work provides a good initial response.

BIBLIOGRAPHY

Aggarwal, J.K. and Duda, R.O. 1975. Computer analysis of moving polygonal images. IEEE Trans. Computers Vol. C-24, 966-976.

Anstis, S.M. and Moulden, B.P. 1970. Aftereffect of seen movement; evidence for peripheral and central components. Q. J. Exp. Psychol. 22, 222-229.

Arbib, M.A. 1975. Artificial Intelligence and Brain Theory: unities and diversities. Annals of Biomed. Eng. 3, 238-274.

Bajcsy, R. and Tavakoli, M. 1976. Computer recognition of roads from satellite pictures. IEEE Trans. on Systems, Man and Cybernetics Vol. SMC-6, No. 9, 623-637.

Barlow, H.B. and Brindley, G.S. 1963. Interocular transfer of movement aftereffects during pressure blinding of the stimulated eye. Nature 200, 1347.

Barlow, H.B. and Levick, W.R. 1965. The mechanism of directionally selective units in rabbit's retina. J. Physiol. 178, 477-504.

Borjesson, E. and von Hofsten, C. 1973. Visual perception in depth. Applications of a vector model to 3-dot motion patterns. Perception and Psychophysics 13, 169-179.

Brice, C.R. and Fennema, C.L. 1970. Scene analysis using regions. Artificial Intelligence 1, 205-226.

Bullock, B. 1974. The performance of edge operators on images with texture. Technical Report, Hughes Research Laboratories, Malibu, California.

Burt, P.J. 1975. Computer simulations of a dynamic visual perception model. Int. J. Man-Machine Studies 7, 529-546.

Burt, P.J. 1976. Stimulus organizing processes in stereopsis and motion perception. Ph.D. Thesis and COINS Technical Report 76-15, University of Massachusetts, Amherst, Massachusetts.

Burt, P.J. 1977. A procedure for evaluating cooperative models for stereopsis. Brain Theory Newsletter 3, 31-35.

Campbell, F. and Robson, J. 1968. Applications of Fourier analysis to the visibility of gratings. J. Physiol. 197, 551-566.

Chow, W.K. and Aggarwal, J.K. 1977. Computer analysis of planar curvilinear moving images. IEEE Trans. Comp. Vol. C-26, 179-185.

Clocksin, W. F. 1978. Perception of surface slant and edge labels from optical flow: a computational approach. Department of Artificial Intelligence Working Paper No. 33, University of Edinburgh, Edinburgh, Scotland.

Cornsweet, T.W. 1970. Visual Perception. Academic Press, New York.

Cynader, M. and Berman, N. 1972. Receptive-field organization of monkey superior colliculus. J. Neurophysiol. 35, 187-201.

de Monasterio, F.M. and Gouras, P. 1975. Functional properties of ganglion cells of the rhesus monkey retina. J. Physiol. 251, 167-195.

Dev, P. 1975. Perception of depth surfaces in random-dot stereograms: a neural model. Int. J. Man-Machine Stud. 7, 511-528.

Didday, R.L. 1970. The simulation and modelling of distributed information processing in the frog visual system. Ph.D. Thesis, Stanford University, Stanford, California.

Dreschler, L. and Nagel, H-H. 1978. Using "affinity" for extracting images of moving objects from TV-frame sequences. University of Hamburg Report No. IfI-HH-B-44/78.

Duda, R.O. and Hart, P.E. 1973. Pattern Classification and Scene Analysis. Wiley, New York.

Ellias, S.A. and Grossberg, S. 1975. Pattern formation, contrast control and oscillations in the short term memory of shunting on-center, off-surround networks. Biol. Cybernetics 20, 69-98.

Enroth-Cugell, C. and Robson, J.G. 1966. The contrast sensitivity of retinal ganglion cells of the cat. J. Physiol. 187, 517-522.

Erickson, R.P. 1974. Parallel "population" neural coding in feature extraction. The Neurosciences: Third Study Program, F.O. Schmitt and F.G. Worden, (Eds.), MIT Press, Cambridge, Massachusetts 155-169.

Erman, L.D. and Lesser, V.R. 1975. A multi-level organization for problem-solving using many, diverse, cooperating sources of knowledge. Department of Computer Science Technical Report, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Favreau, O.E. 1976. Motion aftereffects: evidence for parallel processing in motion perception. Vision Res. 16, 181-186.

Feller, W. 1968. An Introduction to Probability Theory and its Applications. Vol. 1, Third Edition, Wiley and Sons, New York.

Freud, S.C. 1964. The physiological locus of the spiral aftereffect. Am. J. Psychol. 77, 422-428.

Frisby, J.P. and Clatworthy, J.L. 1974. Evidence for separate movement and form channels in the human visual system. Perception 3, 87-96.

Fukuda, Y. and Stone, J. 1974. Retinal distribution and central projections of Y-, X- and W-cells in the cat's retina. J. Neurophysiol. 37, 749-772.

Ganapathy, S. 1975. Reconstruction of scenes containing polyhedra from stereo pairs of views. Stanford AI Lab Memo AIM-272.

Gates, L.W. 1934. The aftereffect of visually observed movement. Am. J. Psychol. 46, 34-46.

Gennery, D.B. 1977. A stereo vision system for an autonomous vehicle. International Joint Conference on Artificial Intelligence-5, 576-582, MIT, Cambridge, Massachusetts.

Gibson, J.J. 1966. The Senses Considered as Perceptual Systems. Houghton-Mifflin, Boston.

Goldberg, M.E. and Wurtz, R.H. 1972. Activity of superior colliculus in behaving monkey. Visual fields of single neurons. J. Neurophysiol. 35, 542-559.

Grossberg, S. 1976. On the development of feature detectors in the visual cortex with applications to learning and reaction-diffusion systems. Biol. Cybernetics 21, 145-159.

Hammond, P. and MacKay, D.M. 1975. Differential responses of cat visual cortical cells to textured stimuli. Exp. Brain Res. 22, 427-430.

Hannah, M.J. 1974. Computer matching of areas in stereo images. Stanford University Memo AIM 239, Stanford, California.

Hanson, A. and Riseman, E. 1978a. Segmentation of natural scenes. In "Computer Vision Systems", A. Hanson and E. Riseman (Eds.), Academic Press.

Hanson, A. and Riseman, E. 1978b. VISIONS: a computer system for interpreting scenes. In "Computer Vision Systems", A. Hanson and E. Riseman (Eds.), Academic Press.

Hanson, A. and Riseman, E. 1976. A progress report on VISIONS: representation and control in the construction of visual models. COINS Technical Report 76-9, Dept. of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts.

Haskell, B.G. 1975. Entropy measurements for nonadaptive and adaptive, frame to frame, linear-predictive coding of video-telephone signals. The Bell System Technical Journal 54, No. 6.

Henrikson, P. 1972. Techniques and applications of digital change detection. Computer Image Processing and Recognition Symposium, University of Columbia, Columbia, Missouri.

Hoffman, K-P. 1973. Conduction velocity in pathways from retina to superior colliculus in the cat: a correlation with receptive-field properties. J. Neurophysiol. 36, 404-424.

Hoffman, K-P. and Stone, J. 1971. Conduction velocity of afferents to cat visual cortex: a correlation with cortical receptive field properties. Brain Res. 32, 460-466.

Hubel, D.H. and Wiesel, T.N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. 160, 106-154.

Hubel, D.H. and Wiesel, T.N. 1965. Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat. J. Neurophysiol. 28, 229-289.

Hubel, D.H. and Wiesel, T.N. 1968. Receptive fields and functional architecture of monkey striate cortex. J. Physiol. 195, 215-243.

Hueckel, M. 1971. An operator which locates edges in digitized pictures. Journal of the ACM 18, No. 1, 113-125.

Humphrey, N.K. 1970. What the frog's eye tells the monkey's brain. Brain Behav. Evol. 3, 324-337.

Ikeda, H. and Wright, M.J. 1975. Spatial and temporal properties of 'sustained' and 'transient' neurones in area 17 of the cat's visual cortex. Exp. Brain Res. 22, 363-383.

Ingle, D. 1967. Two visual mechanisms underlying the behavior of fish. Psychol. Forsch. 31, 44-51.

Jain, R., Militzer, D. and Nagel, H-H. 1977. Separating non-stationary from stationary scene components in a sequence of real-world TV images. International Joint Conference on Artificial Intelligence 5, 612-618, MIT, Cambridge, Massachusetts.

Johannson, G. 1950. Configurations in Event Perception. Almquist and Wiksell, Uppsala.

Julesz, B. 1971. Foundations of Cyclopean Perception. University of Chicago Press, Chicago.

Kirsch, R.A. 1971. Computer determination of the constituent structure of biological images. Computers and Biomedical Research 4, 315-318.

Kolers, P.A. 1972. Aspects of Motion Perception. Pergamon Press, Oxford.

Kuffler, S.W. 1953. Discharge patterns and functional organization of mammalian retina. J. Neurophysiol. 16, 37-68.

Kulikowski, J.J. and Tolhurst, D.J. 1973. Psychophysical evidence for sustained and transient detectors in human vision. J. Physiol. 149-152.

Lam, C.F. and Hoyt, R.R. 1972. High speed image correlation for change detection. National Aeronautics and Electronics Conference, Dayton Ohio.

Land, E.H. 1977. The retinex theory of color vision. Scientific American 237, No. 6, 108-128.

Lee, D.N. 1974. Visual information during locomotion. In "Perception: Essays in Honor of J.J. Gibson", R.B. MacLeod and H.L. Pick (Eds.), 250-267, Cornell University Press, Ithaca, New York.

Lesser, V.R. and Erman, L.D. 1977. A retrospective view of the HEARSAY II architecture. International Joint Conference on Artificial Intelligence-5, MIT, Cambridge, Massachusetts, 790-800.

Lettvin, J.Y., Maturana, H.R., McCulloch, W.S. and Pitts, W.H. 1959. What the frog's eye tells the frog's brain. Proceedings of the Institute of Radio Engineers 47, 1940-1951.

Lev, A., Zucker, S.W. and Rosenfeld, A. 1977. Iterative enhancement of noisy images. IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-7, 435-442.

Levine, M.D., O'Handley, D.A. and Yagi, G.M. 1973. Computer determination of depth maps. Computer Graphics and Image Processing 2, 131-150.

Lichtenstein, M. 1963. Spatio-temporal factors in cessation of smooth apparent motion. J. Opt. Soc. Am. 53, 302-306.

Limb, J.O. and Murphy, J.A. 1975. Estimating the velocity of moving images in television signals. Computer Graphics and Image Processing 4, 311-327.

Maffei, L.A. and Fiorentini, A. 1973. The visual cortex as a spatial frequency analyser. Vision Res. 13, 1255-1267.

Marr, D. 1975. Early processing of visual information. MIT A.I. Memo 340, Cambridge, Massachusetts.

Marr, D. and Poggio, T. 1976. Cooperative computation of stereo disparity. Science 194, 283-287.

Martin, W.N. and Aggarwal, J.K. 1978. Dynamic scene analysis. In Computer Graphics and Image Processing 7, 356-374.

McCann, J.J. and Benton, J.L. 1969. Interaction of the long-wave cones and the rods to produce color sensations. J. Opt. Soc. Am. 59, 103-107.

McIlwain, J.T. 1972. Central vision: visual cortex and superior colliculus. Annual Rev. of Physiol. 34, 291-314.

McIlwain, J.T. 1976. Large receptive fields and spatial transformations in the visual system. International Review of Physiology, Neurophysiology II, Vol. 10, 223-248.

Mohler, C.W. and Wurtz, R.H. 1976. Organization of monkey superior colliculus: intermediate layer discharging before eye movements. J. Neurophysiol. 39, 722-744.

Moravec, H.P. 1977. Towards automatic visual obstacle avoidance. International Joint Conference on Artificial Intelligence-5, 584, MIT, Cambridge, Massachusetts.

Nagel, H-H. 1976. Formation of an object concept by analysis of systematic time variation in the optically perceptible environment. University of Hamburg Report No. IfI-HH-B-27/76.

Nagin, P. Segmentation using feature-clustering and relaxation. In preparation.

Nakayama, K. and Loomis, J.M. 1974. Optical velocity patterns, velocity-sensitive neurons and space perception: a hypothesis. Perception 3, 63-80.

Nelson, J.I. 1974. Motion sensitivity in peripheral vision. Perception 3, 153-168.

Nelson, J.I. 1975. Globality and stereoscopic fusion in binocular vision. J. Theor. Biol. 49, 1-88.

Nevatia, R. 1976. Depth measurement by motion stereo. Computer Graphics and Image Processing 5, 203-214.

Ohlander, R.B. 1975. Analysis of natural scenes. Ph.D. Thesis. Computer Science Dept., Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Palmer, L.A. and Rosenquist, A.C. 1975. Single unit studies in the cat. In "Mammalian tectum intrinsic organization, afferent inputs and integrative mechanisms." Neurosci. Res. Bulletin 13, 214-220.

Pantle, A.J. and Sekuler, R.W. 1969. Contrast response of human visual mechanisms sensitive to orientation and direction of motion. Vision Res. 9, 397-406.

Pantle, A.J. and Sekuler, R.W. 1968. Velocity-sensitive elements in human vision: initial psychophysical evidence. Vision Res. 8, 445-450.

Pettigrew, J.D., Nikara, T. and Bishop, P.O. 1968. Responses to moving slits by single units in cat striate cortex. Exp. Brain Res. 6, 373-390.

Potter, J.L. 1977. Scene segmentation using motion information. Computer Graphics and Image Processing 6, 558-581.

Prager, J., Nagin, P., Kohler, R., Hanson, A., and Riseman, E. 1977. Segmentation processes in the VISIONS system. International Joint Conference on Artificial Intelligence-5, 642-643, MIT, Cambridge, Massachusetts.

Price, K.E. 1976. Change detection and analysis in multi-spectral images. Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Price, K.E. and Reddy, R. 1977. Change detection and analysis in multi-spectral images. International Joint Conference on Artificial Intelligence-5, 619-625, MIT, Cambridge, Massachusetts.

Quam, L.H. 1968. Computer comparison of pictures. Stanford AI Lab Memo AIM-144.

Radig, B. 1978. Description of moving objects based on parametrized region extraction. University of Hamburg Report No. IfI-HH-M-61/78.

Richards, W. 1971. Size-distance transformations. Pattern Recognition in Biological and Technical Systems, Springer Verlag, New York.

Riseman, E.M. and Arbib, M.A. 1977. Computational techniques in the visual segmentation of static scenes. Computer Graphics and Image Processing 6, 221-276.

Roberts, L.G. 1965. Machine perception of three-dimensional solids. Optical and Electro-optical Processing of Information. J.T. Tippet

(Ed.), 159-197. MIT, Cambridge, Massachusetts.

Robinson, G.S. 1977. Color edge detection. Optical Engineering 16, No. 5, 479-484.

Rosenfeld, A., Hummel, R.A. and Zucker, S.W. 1976. Scene labelling by relaxation operations. IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-6, 420-433.

Rosenfeld, A. and Thurston, M. 1971. Edge and curve detection for visual scene analysis. IEEE Trans. Computers, Vol. C-20, No. 5, 562-569.

Schiller, P.H., Finlay, B.L. and Volman, S.F. 1976a. Quantitative studies of single-cell properties in monkey striate cortex. I. Spatiotemporal organization of receptive fields. J. Neurophysiol. 39, 1288-1319.

Schiller, P.H., Finlay, B.L. and Volman, S.F. 1976b. Quantitative studies of single-cell properties in monkey striate cortex. V. Multivariate statistical analyses and models. J. Neurophysiol. 39, 1362-1374.

Schiller, P.H. and Koerner, F. 1971. Discharge characteristics of single units in superior colliculus of alert rhesus monkey. J. Neurophysiol. 34, 920-936.

Schiller, P.H. and Malpeli, J.G. 1977. Properties and tectal projections of monkey retinal ganglion cells. J. Neurophysiol. 40, 428-445.

Schneider, G.E. 1967a. Contrasting visuomotor functions of tectum and cortex. Psychol. Forsch. 31, 52-62.

Schneider, G.E. 1967b. Two visual systems: brain mechanisms for localization and discrimination are dissociated by tectal and cortical lesions. Science 163, 895-902.

Southwell, R.V. 1935. Stress calculations in frameworks by the method of 'systematic relaxation of constraints'. Proc. Roy. Soc. A 151.

Sterling, P. and Wickelgren, B.G. 1969. Visual receptive fields in the superior colliculus of the cat. J. Neurophysiol. 32, 1-15.

Stevens, K.A. 1977. Computation of locally parallel structure. AI Memo AIM-392. MIT, Cambridge, Massachusetts.

Stone, J. 1972. Morphology and physiology of the geniculo-cortical synapse in the cat: the questions of parallel input to the striate cortex. Invest. Opth. 11, 338-346.

Stone, J. and Dreher, B. 1973. Projection of X- and Y-cells of the cat's lateral geniculate nucleus - areas 17 and 18 of visual cortex. J. Neurophysiol. 30, 551-567.

Stone, J. and Fukuda, Y. 1974. Properties of cat retinal ganglion cells: A comparison of W-cells with X- and Y-cells. J. Neurophysiol. 37, 722-748.

Tenebaum, J.M. and Barrow, H.G. 1976. Experiments in interpretation-guided segmentation. Technical Note 123, Stanford Research Institute, Menlo Park, California.

Thompson, C. 1975. Depth perception in stereo computer vision. Stanford AI Lab. Memo AIM-268.

Tolhurst, D.J. 1973. Separate channels for the analysis of the shape and the movement of a moving visual stimulus. J. Physiol. 231, 385-402.

Trehub, A. 1978. Neuronal model for stereoscopic vision. J. Theor. Biol. 71, 479-486.

Trevarthen, C.B. 1968. Two mechanisms of vision in primates. Psychol. Forsch. 31, 299-377.

Ullman, S. 1978. The correspondence process in motion perception. ARPA I.U. Workshop.

Ullman, S. 1977. The interpretation of visual motion. Ph.D. Thesis MIT, Cambridge, Massachusetts.

Walls, G.C. 1953. Interocular transfer of afterimages. Am. J. Opt. 30, 57-64.

Waltz, D. 1972. Generating semantic descriptions from drawings. Ph.D. Thesis and AI Lab. Tech. Rep. No. 271, MIT Cambridge, Massachusetts.

Weisstein, N. and Maguire, W. 1978. Computing the next step: psychophysical measures of representation and interpretation. In "Computer Vision Systems", A. Hanson and E. Riseman (Eds.), Academic Press, New York.

Weisstein, N., Maguire, W. and Berbaum, H. 1977. A phantom-motion aftereffect. Science 198, 955-957.

Wiedmann, C. 1975. APLUM reference manual. Second Edition. University of Massachusetts Computing Center, University of Massachusetts, Amherst, Massachusetts.

Wurtz, R.H. and Mohler, C.W. 1976. Organization of monkey superior colliculus: enhanced visual response of superificial layer cells. J. Neurophysiol. 39, 745-768.

Yakimovsky, Y. 1976. Boundary and object detection in real world images. Journal of the ACM 23(4), 599-618.

Zucker, S.W., Hummel, R.A. and Rosenfeld, A. 1977. An application of relaxation labelling to line and curve enhancement. IEEE Trans. Computers, Vol. C-26, 394-403 and 922-924.

Zucker, S.W., Krishnamurthy, E.V. and Haar, R. 1977. Relaxation processes for scene labelling: convergence, speed and stability. Tech. Rep. 77-6, Dept. of Elect. Eng., McGill University, Montreal, Canada.

APPENDIX I

We derive here the maximum likelihood estimate of the confidence of an edge which is used in Section 2.5. We follow Yakimovsky (1976), making appropriate modifications to cope with the use of dynamic mean.

Let S be a set of points $\{ (X_j, Y_j, I_j), j = 1,\dots,n \}$ where I_j is the intensity of pixel (X_j, Y_j). Let $\nu = \frac{n}{2}$. We will suppose that the intensity distribution is 'normal' with variance σ^2 and dynamic mean

$$\mu^*(j) = \mu + aX_j + bY_j \; .$$

S may be considered to be a random sample from this distribution, so will have probability density

$$\prod_{j=1}^{n} \frac{1}{\sqrt{2\Pi\sigma}} e^{-\frac{1}{2\sigma^2}(I_j - \mu - aX_j - bY_j)^2}$$

$$= \left(\frac{1}{2\Pi\sigma^2}\right)^{\nu} e^{-\left(\frac{1}{2\sigma^2}\right) \Sigma (I_j - \mu - aX_j - bY_j)^2}$$

The logarithm of this likelihood function is

$$L = \nu \log 2\Pi - \nu \log \sigma^2 - \frac{1}{2\sigma^2} \Sigma (I - \mu - aX_j - bY_j)^2 \; .$$

We compute

$$\frac{\partial L}{\partial \mu} = \frac{-2}{\sigma^2} \Sigma (I - \mu - aX_j - bY_j)$$

$$\frac{\partial L}{\partial a} = \frac{-2}{\sigma^2} \Sigma (I - \mu - aX_j - bY_j)X_j$$

$$\frac{\partial L}{\partial b} = \frac{-2}{\sigma^2} \Sigma (I - \mu - aX_j - bY_j)Y_j$$

$$\frac{\partial L}{\partial \sigma^2} = \frac{-n}{2\sigma^2} + \frac{1}{2\sigma^4} \Sigma (I - \mu - aX_j - bY_j)^2$$

Putting these derivatives equal to zero gives the estimators $\bar{\mu}$, $\bar{a}$, $\bar{b}$, and $\bar{\sigma}^2$. It is clear that $\bar{\mu}$, $\bar{a}$ and $\bar{b}$ are those values which minimize

$$\sum_{j=1}^{n} (I_j - \mu^*(j))^2.$$

We have

$$\bar{\mu}^* = \frac{1}{n} \Sigma (I_j - \bar{a}X_j - \bar{b}Y_j)$$

$$\bar{\sigma}^2 = \frac{1}{n} \Sigma (I_j - \mu^*(j))^2.$$

The quantity $I_j - \bar{a}X_j - \bar{b}Y_j$ may be considered as a 'normalized' intensity measure I_j^*, that is, I_j with spatial dependencies removed. In any case, if X_j and Y_j are calculated with respect to the centroid of S, then

$$\bar{\mu} = \frac{1}{n} \sum_{j=1}^{n} I_j \; .$$

The probability density that I_j is generated by $N(\bar{\mu}, \bar{\sigma})$ is there-

$$\frac{1}{\sqrt{2\Pi\bar{\sigma}^2}} e^{-(I_j - \mu(j))^2/2\bar{\sigma}^2} .$$

Assuming independence, which is a valid assumption if the variation in I* is due to noise, then the joint probability density for the whole region is the product

$$\prod \frac{1}{\sqrt{2\Pi\bar{\sigma}}} e^{-(I_j - \bar{\mu}^*(j))^2/2\bar{\sigma}^2}$$

$$= \left(\frac{1}{\sqrt{2\Pi\bar{\sigma}}}\right)^n e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(I_j - \bar{\mu}^*(j))^2}$$

$$= \frac{1}{(2\Pi)^\nu} e^{-\frac{1}{2}} \frac{1}{\sigma^n} .$$

Now we consider the sets $S_1 = \{(I_{1j}, X_{1j}, Y_{1j}), j = 1,\ldots,n_1\}$, $S_2 = \{(I_{2j}, X_{2j}, Y_{2j}), j = 1,\ldots,n_2\}$ and $S_0 = S_1 \cup S_2 = \{(I_{0j}, X_{0j}, Y_{0j}), j = 1,\ldots,n_0\}$ where $n_0 = n_1 + n_2$ and the points in S_0 are just the points in S_1 and S_2 relabelled. Let $\nu_i = \frac{n_i}{2}$, for $i = 0, 1, 2$.

We calculate for $i = 1, 2$

μ_i, a_i and b_i to minimize $\sum_{j=1}^{n_i}(I_{ij} - a_iX_{ij} - b_iY_{ij})^2$ so that

$\mu_i^*(j) = I_{ij} - a_iX_{ij} - b_iY_{ij}$, also

$$\mu_i = \frac{1}{n_i}\sum_{i=1}^{n_i} I_{ij}$$

$$\sigma_i^2 = \frac{1}{n_i}\sum_{j=1}^{n_i}(I_{ij} - \mu_i^*(j))^2$$

$$P_i = \left(\frac{1}{2\Pi}\right)^{\nu_i} e^{-\nu_i} \frac{1}{(\sigma_i^2)^{\nu_i}}$$

Hence

$$\mu_0^* = (n_1\mu_1^* + n_2\mu_2^*)/n_0$$

$$\sigma_0^2 = [n_1\sigma_1^{*2} + n_2\sigma_2^{*2} + n_1(\mu_0^* - \mu_1^*)^2 + n_2(\mu_0^* - \mu_2^*)^2]/n_0$$

$$P_0 = \frac{1}{(2\Pi)^{\nu_0}} e^{-\nu_0} \frac{1}{(\sigma_0^2)^{\nu_0}} .$$

We wish to test the hypothesis H_1 that the points in S_1 and S_3 belong to different regions, against H_0 that they belong to the same region. Assuming independence, in case H_1 the joint probability density of P_1 and P_2 is their product, viz

$$P_1P_2 = \frac{1}{(2\Pi)^{\nu_0}} e^{-\nu_0} \frac{1}{(\sigma_1^2)^{\nu_1}(\sigma_2^2)^{\nu_2}} .$$

The maximum likelihood ratio, which we use to estimate the confidence of the line between S_1 and S_2, is

$$\frac{P_1P_2}{P_0} = \frac{(\sigma_0^2)^{\nu_0}}{(\sigma_1^2)^{\nu_1}(\sigma_2^2)^{\nu_2}} .$$

APPENDIX II

JASON: THE ROBOT IN THE FOREST

In this appendix we provide a description of a computer system simulating the visual experience of movement through a forest. The system is written in APL on the CYBER-74 computer with graphic output on a Tektronix 4013 scope. No knowledge of APL is required to run the system although some knowledge will allow the full power of the system to be used.

A2.1 The Simulation System

Our robot Jason lives in a simulated forest. This forest is populated by (a variable number of) simulated trees. These trees are generally stationary, although one of them can be made to move in order to demonstrate the optic effects that motion produces.

In general, Jason may be made to move anywhere in the forest, although the system is set up so that the user may easily specify any linear motion. Jason may look in any direction parallel to the ground, either in a fixed direction in space, or to fixate some stationary or moving object (tree).

Three forms of graphic output are available:

(1) a 'bird's-eye-view' of the forest,

(2) a static view of the forest as seen by Jason, and

(3) the optic flow produced as Jason takes a step.

The two frames concerned in case (3) are the static views before and after the step. What Jason sees will depend on the nature of his enviment, which will now be described.

Only the trunks of the trees in the forest are generated. These are straight, and are all of height 5 units, compared with Jason's height of 1.0. Their radii are randomly determined. In order to provide some semblance of reality, texture markings are generated on the tree-trunks. These take the form of dashed lines regularly spaced around the circumferences of the trunks. Since the texture markings represent fixed points in space, while the apparent edges of the trees depend on the orientation from which they are viewed, the user is given the ability to specify whether the optic flow should be generated from the tree boundaries, the texture, or both. The flow is generated from the end-points and mid-points of these lines. All of these points are given unique labels, so that the correspondence between frames may be directly determined.

A2.2 APL Syntax

We provide below the minimal understanding of APL syntax needed to use the system effectively.

Variables. The basic data-types used in the system are scalars and vectors. A scalar is a single quantity with an associated name, e.g. HEIGHT. A vector is a list of scalars, and it too has a name. Individual items are referenced by indexing: thus X[3] is the third component of vector X. Some of the scalars used are real-valued, others are used as logical variables which can only (meaningfully) take on the values 1 and 0 (representing TRUE and FALSE, respectively).

Assignment. Values may be assigned to variables in the following way. The name of the variable is typed, followed by a left-arrow, followed by the value the variable is to assume, which may be the result of evaluating an expression. Thus

```
A ← C + D
B ← 1 2 3 4 .
```

The type of a variable is the type of the last value it is assigned. So, in the example above, B becomes a vector, irrespective of its previous data-type.

Printing a Variable. To examine the value of a variable, the name of the variable is simply typed. The system responds with its value. Thus

```
          B
      1 2 3 4.
```

Functions. There are six kinds of APL functions. They may be niladic, monadic or diadic, depending on the number of arguments they take, and they may or may not return a result. If a result is returned, it may be assigned to a variable, used in an expression or simply printed. The forms of the niladic, monadic and diadic function calls are:

```
FUNCTION
FUNCTION P
Q FUNCTION R
```

where P, Q and R are the arguments. Typing the function name (with arguments if appropriate) causes it to be evaluated.

When operating the system, it should be remembered that when APL is ready for input, it will indent by six spaces. For more details, see Wiedmann (1975).

A2.3 System Usage

Coordinates. In this system, the following convention is followed. The Y-direction is vertical, and the X-Z plane is parallel to the ground. The origin of coordinates is at the robot's eye.

System Variables and Functions. Provided below is a list of some of the more important system variables and functions. The user may wish to examine or change some of these variables. In the list that follows, those variables that are scalars and real-valued are denoted by (R), and those that are logical by (L). Vectors of variable length will be denoted by (V), while those of fixed length N are denoted by (N).

The Forest. The forest consists of a number of trees randomly distributed on a 20 x 30 grid.

NUMTREE (R)	the number of trees in the forest.
X (V)	the X-coordinates of the trees.
Z (V)	the Z-coordinates of the trees.
RADIUS (V)	the radii of the trees.
HEIGHT (R)	the height of the trees (they are all of the same height).
NUMDASH (R)	the number of dashes per line of texture-markings on the trees.
MTREE (R)	the index of the moving tree (in the range 1-NUMTREE). MTREE = 0 if no tree is to move.
TREEV (3)	the velocity of the moving tree.
TARGET (R)	the tree Jason is fixating (in the range 1-NUMTREE). TARGET = 0 if Jason is to look in a constant direction (OV).

The Robot.

PR (3)	the position coordinates of the robot.
OV (3)	the orientation vector (direction of gaze) of the robot. N.B. OV[2] is always 0.
TV (3)	the translation vector (velocity) of the robot. N.B. TV[2] is always 0.
THETA (R)	the angle Jason will turn through after taking each step.

The Display.

SHOWTREE (L)	whether Jason's static view is shown after taking a step.
TEXONLY (L)	whether optic flow generation from tree boundaries should be suppressed.
TREEONLY (L)	whether optic flow generation from texture markings should be suppressed.
NOPLOT (L)	whether display of optic flow should be suppressed.
NOERASE (L)	whether the screen should maintain the display from the previous frame while plotting the next (used in generating extended flow-line segments).

Useful Functions.

SEEFOREST	will provide a bird's-eye-view of the forest, indicating Jason's position and approximate orientation with an arrow.
SEETREE N	as SEEFOREST, but will mark tree N with an 'X'.
GENFOR	will generate and display a new (random) forest.
SIMPLOT	will display the forest as Jason sees it. [N.B. This is only meaningful if Jason has taken a step via MASTER (see below). Just

	changing PR and/or OV alone are not sufficient for seeing the new view from this new position/orientation.]
PARMS	will print the current values of a number of system variables.
RESET	initializes the robot and blanks the screen.

A2.4 Running the System

(1) To access this system, after signing on to the CYBER, type

```
SETTL, 1000
APL, WS = JASON
```

to enter APL, then

```
)COPY * APL50   TEKTRON .
```

(2) To set Jason in motion, type

```
MASTER N
```

where N is an integer. Jason will then take N steps. The system will then respond with ⎕:, requesting input. By entering

an integer M, where 1 ≤ M ≤ 9,	Jason will take M further steps, and the system will prompt again.
0	the function will terminate.
'any APL expression in quotes'	the system will execute that expression and then prompt again.

It is most usual to let Jason take one step at a time.

APPENDIX III

USING MATCH ON THE GT-44

Introduction

In this Appendix we describe the use of the program MATCH on the GT-44 computer. Two variants of MATCH have been written for running the algorithm in certain special modes. A full users' guide to MATCH will firstly be given, followed by a description of how the variants differ.

MATCH is the basic stimulus-matching program. The user may specify configurations of dots, and the motions which these dots are to undergo. There will be one 'processor' associated with each dot, and in some conditions (described later) associated with other (non-stimulated) points too.

MATCH2 is the collective name given to a set of three programs used to examine the effectiveness of two different matching algorithms (and different parameter values) for the same motion of a given configuration of dots.

MATCH3 is the name of a program used to simulate conditions where the processes are not assigned to, or even in register with the configurations of dots.

A full description of the use of MATCH2 and MATCH3 will be given following the users'-guide to MATCH. In the final sections of this appendix details of logging-on, compiling the programs, and setting-up the plotter are given.

A3.1 User's Manual for MATCH

MATCH consists of six stages for specifying the input to the model and running it. Each stage is described in a separate section. In each section the user interactions are covered in full, and the displays are explained.

The sections that follow are:

Input

1) Specifying conditions of the run.
2) Choosing the size and location of the moving object.
3) Determining the motion.

Run

4) Checking the input prior to running.
5) Running - generating the 'Optic Flow'.
6) Running - resolving the flow.

These sections are divided into four subsections: description, display, user action and exit.

General Comments

- To start running the model, after the usual starting-up procedure (see Section A3.1.5) type

  R MATCH.

- The term Message Area refers to a small portion of the screen down in the bottom left-hand corner.
- At any stage, the run may be terminated by hitting CONTROL/C twice.

- Every individual stage of the program may be exited by hitting carriage return (CR), which will cause the next stage to be immediately entered. Thus during the running of stage S ($1 \leq S \leq 6$) hitting CR 7 - S times will restart the program.
- A summary of the significance of all keys throughout the running of the model is given in Table A.1.
- The more points that are used in each run, the longer the computations will take. (The relationship is quadratic.)

Key or Key Combination	Stage	Effect
		Move object along:
1	3b	TX: X axis in L-R direction.
SHIFT/1	3b	-TX: - - - - - R-L - - - - -
2	3b	TY: Y - - - - D-U - - - - -
SHIFT/2	3b	-TY: - - - - - U-D - - - - -
3	3b	TZ: Z - - - - F-B - - - - -
SHIFT/3	3b	-TZ: - - - - - B-F - - - - -
		Rotate object about:
4	3b	RX: X axis clockwise.
SHIFT/4	3b	-RX: - - - anticlockwise.
5	3b	RY: Y - - clockwise.
SHIFT/5	3b	-RY: - - - anticlockwise.
6	3b	RZ: Z - - clockwise.
SHIFT/6	3b	-RZ: - - - anticlockwise.
P	5	causes the program to complete the iteration then PAUSE prior to plotting the display
T	6	cause TRANSLATION component to be displayed
R	6	- - - ROTATION - - - - - - - - - - -

Table A.1. Keyboard Keys and Their Significance in MATCH.

Any Integer N and Carriage Return	end of 5	N further iterations are preformed
Carriage Return (CR)	1,2,3a,3b } 4,5 }	Pass to next stage of program
	6	Restart
CONTROL/E	3b	Cause name of motion to be output (or not) to keyboard
CONTROL/C (hit twice)	all stages	Exit from Program

Table A.1. Keyboard Keys and Their Significance in MATCH. (cont.)

A3.1.1 Stage 1: Specifying the Conditions of the Run

Description

The user is to make three selections in this stage:

(a) Mode of Motion Description - The alternatives are 'Implicit', using light-pen, and 'Explicit', using the keyboard. This determines how in Stage 3 the object specified in Stage is to be moved.

(b) Background Condition - The alternatives are 'None', 'Fill-in', and 'Stationary'. The meaning of these is not described here.

(c) Size of Display - The number of points in the display may be chosen, from 4 x 4 to 10 x 10. This is the size of the 'visual world' of the model.

Display (Chinese menu)

Three columns are displayed, with the various choices under the appropriate header. The currently selected choice is made to blink. When the program is entered, certain defaults are already blinking. Choices may be made and changed as often as desired during this stage. The word 'next' is displayed at the bottom of the screen and may be selected by the light-pen for passing onto Stage 2.

User Action

Choose which alternatives are wanted and hit them with the light-pen until all desired choices are blinking. Due to occasional light-pen 'flakiness' it may be necessary to re-make old choices. If due to such flakiness the program passes on to one of the real stages before Stage 1 choices are complete, hit CONTROL/C twice to exit the program and

R MATCH to re-enter, or CR repeatedly until Stage 1 is re-entered.

Exit

To exit this stage and enter Stage 2, either hit 'next' with the light-pen or hit CR (once). The screen will be blanked and the Stage 2 display will appear.

A3.1.2 Stage 2: Choosing the Size and Location of the Moving Object

Description

The user may now decide which subset of the grid points in the model's visual field is to belong to the moving object. (The motion is specified in the next section.) At the moment the shape is limited to a rectangle oriented to the areas of the display, but the wide variety of possible motions should offset this limitation. In the 'Fill-in' background condition, the dots formed by reflecting the rectangle in the centre of the figure are also selected, in order to generate a figure with an 'outside', and a 'middle' to be filled in.

Display

A grid of n x n points is displayed, where n was chosen in Stage 1. Superimposed upon this is a rectangle with 2 small boxes in diagonally opposite corners. By moving this rectangle a set of points (those interior to it) may be defined.

User Action

The rectangle may be 'picked up' with the light-pen at either of the corners indicated by the small boxes and moved around until in the desired position. The size of the included set of points may be anything from n x n to 1 x 1 to n x 1 to 1 x n. However, the results will be meaningless if the rectangle is turned 'inside out' by passing opposite sides over each other. This condition is indicated by the small boxes being <u>outside</u> the rectangle. If this condition occurs, the rectangle may be returned to its proper state, but if left 'inside out', a

null set of points will be selected.

Exit

This stage may be exited either by hitting 'next' with the light-pen or CR on the keyboard. The appropriate Stage 3 display will then appear.

A3.1.3 Determining the Motion

Stage 3(a): Implicitly using the Light-Pen

Description

As a surface moves in space, so its projection on a watching retina varies. By indicating how the projection changes, so the motion can be described implicitly. In this stage, the surface selected in Stage 2 can be distorted to represent a motion.

Since the model attempts to deal with simulations of two successive frames or snapshots, small distortions only are recommended.

Display

The display is the same as in Stage 2, except that

(i) all the dots outside the rectangle disappear

(ii) a small box appears at the top left and bottom right corner of the rectangle, so there is now a box in each corner.

User Action

The rectangle can now be 'picked up' at any of its corners with the light-pen and can be distorted to an arbitrary quadrilateral. There are two caveats:

(i) do not turn the quadrilateral 'inside out'.

(ii) use small distortions only.

Exit

Hit 'next' with the light-pen or CR on the keyboard to pass to the RUN phase.

Stage 3(b): Explicitly, Using the Keyboard

Description

The object represented on the screen may be made to undergo any combinations of 6 basic motions: translation in the X(L → R), Y(D → U) and Z(F → B) axes, or rotations about these axes. These motions are denoted by TX, TY, TZ, RX, RY, RZ respectively.

For convenience, it is assumed internal to the program that the surface displayed is always approximately in the X - Y plane. Thus the display will only be roughly accurate for small motions in these areas.

Display

The display will be the same as in Stage 2 except that all dots outside the rectangle and both small boxes will disappear. The rectangle and remaining dots will move according to keyboard selections. The most recent motion executed is displayed in the Message Area. A complete history may be recorded on the keyboard typewriter by pressing CONTROL/E. This combination is idempotent.

User Action

A small motion of TX, TY, TZ, RX, RY or RZ may be accomplished by hitting key 1, 2, 3, 4, 5 or 6 respectively. Multiple hits will produce compounded motions. Hitting the SHIFT key simultaneously with 1, 2, 3, 4, 5 or 6 will produce the inverse motions, denoted by -TX, -TY, -TZ, -RX, -RY, -RZ respectively. Thus, hitting '2' will move the object 1 unit up, 'SHIFT/2' will move it 1 unit down. All other keys except CR are disregarded.

Exit

Hit CR to pass to the RUN phase.

A3.1.4 Stage 4: Checking the Input Prior to Running

Description

This stage allows the user to view the input to the model just prior to running, and to decide if it is suitable.

Display

The set of dots selected in Stage 2 is displayed with the dots in their original positions. Also displayed, blinking, are the dots after they have undergone the motion specified during Stage 3.

'PAUSE' is displayed in the Message Area.

User Action/Exit

Examine the displacements which the dots have undergone. If this is acceptable, hit CR. Else hit CONTROL/C (twice) to exit the program and type R MATCH to restart (or CR three times).

A3.1.5 Stage 5: Running: Generating the Optic Flow

Description

The model attempts to determine which flashing dot 'came-from' which stationary dot. Twenty iterations are used (at first).

Display

This depends upon which background condition was specified in Stage 1 (and so does the computation of course).

(i) None. The dots in Stage 4 are displayed, along with a displacement vector at each flashing dot 'reaching back' to its estimated source dot.

(ii) Fill-in. As (i), but with displacement vectors at each background position not covered by the object that has moved.

(iii) Stationary. A static background is represented by superimposed flashing and non-flashing dots at each background and position not covered by the moving object. Displacement vectors are associated with these as well.

The iteration number is displayed in the Message Area.

At the bottom of the screen a graph of 'Error' against Iteration is plotted. The 'error' is calculated as the sum of the differences between the coordinates of the optic-flow arrowheads and their correct matches (target points). This correspondence is known to a monitor process but not the matching process itself. N.B. During each iteration; the error pertinent to the previous iteration is calculated. Thus after N iterations, the graph only covers the first N-1 iterations, although

the Message Area indicates the iteration in progress, i.e. No. N+1.

User Action

Watch!

If at any point the display is to be plotted, hit P. At the end of the iteration the program will PAUSE. When the plotter is ready, hit CR.

Exit

At the end of 20 iterations, the program will display 'FURTHER ITERATIONS?' in the Message Area. Typing a positive integer N and carriage return will set the program on N further iterations.

Hit CR (or Ø, CR) to pass to Stage 6.

A3.1.6 Stage 6: Running: Resolving the Flow

Description

In this, the final stage of the program, the optic flow generated in the previous stage may be resolved into translation and rotation components. This process is not meaningful (at present) if any motion in depth has occurred. The program will run for 200 iterations, then PAUSE.

Display

The particular component selected at any instant is displayed on the screen, with TRANSLATION COMPONENT or ROTATION COMPONENT in the Message Area, as appropriate. The display is updated every iteration. At the end of the run, when the program PAUSEs, both components will be displayed together.

User Action

Press T to have the Translation component displayed, R for the Rotation component. These keys may be hit any number of times.

Exit

Press CR when the program PAUSEs to restart. Alternatively, hitting CR while the program is running will also cause the program to restart.

A3.2 MATCH2

MATCH2 is the name given to a set of three programs MATCH1, CATCH2, and DATCH2 used for the experimentation with different algorithms and parameter setting.

A3.2.1 MATCH1

In order that experiments may be performed with exactly the same motions of dots, the MATCH program was split in two. MATCH1 consists of the first three stages of MATCH (Input) and terminates by writing all relevant information about the specified configuration of dots and their motions to a file (MDATA.DAT).

A3.2.2 CATCH2 and DATCH2

The other two programs, CATCH2 and DATCH2, consist of the RUN portion of MATCH (slightly modified, see below). Upon entry they read the file MDATA.DAT produced by MATCH1. CATCH2 and DATCH2 are identical except that they use different forms of the Attraction function (see Section 4.3). <u>C</u>ATCH2 employs the <u>c</u>ontinuous form, <u>D</u>ATCH2 the <u>d</u>iscontinuous form. Also, in the error graph, CATCH2 uses a continuous line, DATCH2 a dashed line. These are the only differences between the two.

CATCH2 and DATCH2 consist of an initialization and three other stages:

(1) Random Spread input. The system prompts the user for a value of the random spread to be input.

(2) Pause to examine the input prior to running.

(3) Running: generating the 'Optic Flow'.

Stages (2) and (3) are identical to stages (4) and (5) of MATCH. Stages (1), (2) and (3) are linked cyclically, so that hitting carriage return will cause the system to pass from one stage to the next.

The purpose of CATCH2/DATCH2 is thus seen to be to examine the effectiveness of the different Attraction functions, on a given dot pattern but with variable feature-spread.

The RUN NUMBER

The system keeps track of how many times the program is run in the variable RUN NUMBER. This is displayed in stage (1) and incremented each time stage (1) is re-entered. Since the system used a pseudo-random number generator, the same feature types will be used for a given run number on a given number of dots. Thus runs performed by CATCH2 and DATCH2 on the same data with the same run number will start with identical conditions.

In the Initialization phase, after the data-file MDATA.DAT is read in, the system asks for an initial value M for the run number. It then exercises the random number generator (M - 1)*N times, where N is the number of dots in the pattern, in order to achieve the same state as if M-1 runs had already taken place.

Plotting

At any time during the run phase (stage (5)) the display may be plotted. If during this stage the 'P' key is hit, the system will complete the current iteration then PAUSE, to allow the user to ready the plotter. When ready, hit CR. The system will plot the flow pattern.

It will then PAUSE again. When CR is hit a second time, the system will plot the dots and the error graph. The purpose of the second PAUSE is to enable the pen to be changed, if desired.

If 'Q' is hit instead of 'P', the system will jump straight to the second PAUSE mentioned above at the end of the iteration. When CR is hit it will plot the error graph. This is useful if the error graphs of two runs are to be superimposed.

A3.3 MATCH3

MATCH3 is a program very similar in use to MATCH; the major difference is that the processing elements (PE's) are not in register with the dot patterns. This situation is described in detail in Section 5.3. Most of the stages of MATCH3 are the same as those of MATCH. The differences will now be described.

(i) Between stages (3) and (4), the system prompts for two variables: RANDOM SPREAD and SIZE of FIELD. The RANDOM SPREAD is the size of the range of feature-types used in the runs. The SIZE of FIELD S is used to calculate the distance between the PEs.. Suppose in stage (1) a SIZE of DISPLAY (dots) of N x N was chosen. Then the distance between processing elements will be 6S/N. There will be N x N such elements, although the number of dots, as selected in stage (2) may be less than this.

(ii) In stage (4), the system displays, along with the dot patterns prior to matching, a grid representing the receptive fields of the PEs. In stage (4) each dot will 'report' to the PE in whose field it lies (see equation 12 in Chapter V). The PEs are in the centres of the squares in the grid.

(iii) Plotting is performed exactly as described in Section A3.2 on CATCH2 and DATCH2.

A3.4 Getting on the System

(1) Turn machine on via key.

(2) Insert disc 3 "JP" and press RUN on disc drive.

(3) Set switch registers to 173100.

(4) Press HALT, then LOAD ADDRESS.

(5) Set switch registers to 177406.

(6) Raise ENABLE switch.

(7) Press START.

An RT-11 version 2 message should appear on the console.

(8) Type
```
GT ON
R ASSIGN
DAT dd-mmm-yyy
```

where dd-mmm-yy is the day's date, e.g. 17-APR-79.

A program is run by issuing the command R <program name>.

A3.5 Compiling and Linking MATCH Programs

Since the linking procedure for most of the programs is rather complicated, BATCH files have been generated to ease this process. To set up the BATCH handler, issue the commands:

```
ASS TT LOG
LOA TT, BA
```

Then for each compilation/linking:

```
R BATCH
<proc file>
```

where <proc file> is PROC1 for MATCH1

PROCC2 for CATCH2

PROCD2 for DATCH2

PROC3 for MATCH3.

The corresponding source file will be compiled and linked.

The four BATCH files are known to the system as <proc file> · BAT.

A3.6 Using the Plotter

Setting up

(1) Turn the plotter on (rocker switch at left).

(2) Depress LOAD button.

(3) Adjust paper on bed of plotter.

(4) Press LOAD button again so that it releases. The paper should be held to the plotter bed electrostatically. Smooth out air bubbles.

(5) Insert pen in holder.

(6) Using joystick, move pen to bottom left of desired plotting area and press SET LOWER LEFT until there is a 'beep'.

(7) Repeat (6) for upper right.

(8) Make sure LOCAL button is down

The plotter is now ready for use.

After Plotting

(9) Depress the LOAD switch. The pen will move off the paper.

(10) Remove paper and get new sheet.

(11) Perform steps (3) and (4) above.

The plotter is ready for use again. The dimensions and position of the plotting area may be changed at any time via steps (6) and (7) above.

At the end of the session:

(12) Remove pen.

(13) Turn off plotter.

APPENDIX IV

PARALLEL PROCESSING IN MOTION ANALYSIS

In this appendix we use two approaches to solving the problems of motion analysis within a parallel computational paradigm. We will in both cases consider arrays of computational elements whose task it is to compute via interactive processes certain properties or functions of the visual input. We will suppose that each element can 'see' a small portion of the visual field, and can communicate with its immediate neighbours.

A4.1 Properties of an Averaging Network

Suppose that a region in the visual field has been extracted (perhaps by differentiating the optic flow), and that it is desired to find its centroid in a parallel manner. This is a possible application of the following process.

Let $G = \{V, E\}$ be a network of N nodes, where the links from each node v_i include a link with v_i itself. Let there be a value $x_i(t)$ associated with node v_i, and suppose that at each iteration x_i is replaced by the average of itself and the values at neighbouring nodes, i.e.

$$x_i(t+1) = \frac{1}{k_i} \sum_{(v_i,v_j)\in E} x_j(t) \qquad (*)$$

where k_i is the order of node v_i. We will show that the x_i converge to a common value

$$\frac{\Sigma\, k_i x_i(0)}{\Sigma\, k_i} .$$

Choosing an arbitrary but fixed ordering for the nodes, if we average the x_i as a column vector X, we can express the transformation at each iteration by the equation

$$X(t+1) = PX(t)$$

where P is an N x N matrix with the following property. For each row i of P, k_i elements take the value $1/k_i$, and the rest the value 0. For example, the following is the transition matric P for the network of Figure A4.1:

$$\begin{pmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}$$

Clearly, for any such P,

$$\sum_j P_{ij} = 1 .$$

Put $K = \sum_{s=1}^{N} k_s$. Define the matrix M to have components $m_{ij} = \frac{k_j}{K}$.

Then M is of the following form:

$$M = \frac{1}{K} \begin{pmatrix} k_1, & k_2, \ldots\ldots & k_n \\ k_1 & \ldots\ldots\ldots & \vdots \\ \vdots & & \vdots \\ k_1 & \ldots\ldots\ldots & k_n \end{pmatrix}$$

Consider the product PM. The i - jth component of this product is

$$(PM)_{ij} = \sum_s P_{is} M_{sj}$$

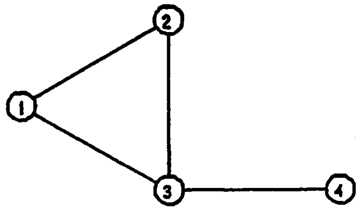


Figure A4.1. Example Network. The transition matrix P for this network is given in the text.

$$= \left(\sum_s P_{is} \right) \frac{k_j}{K}$$

$$= \frac{k_j}{K}$$

so it follows that PM = M.

Now, assuming that the network G is connected, P may be regarded as the transition matrix of an irreducible Markov chain with ergodic elements (see, e.g. Feller 1968 ch. XV · 7). This guarantees that $\lim_{n\to\infty} P^n$ exists, and must equal M. Thus, in the limit,

$$X = MX(0).$$

But as all of the rows of M are identical, the limit of the x_i is independent of i, and is equal to

$$\sum_{j=1}^{n} m_{ij} x_j(0) = \frac{\sum k_j x_j(0)}{K}$$

as required.

Interestingly, a kind of 'conservation of mass' law may be shown to apply. Multiplying either side of (*) by k_i and summing over i, we get

$$\sum_i k_i x_i(t+1) = \sum_i \sum_{(v_i, v_j)\in E} x_j(t)$$

$$= \sum_j k_j x_j(t)$$

since x_j (for some j) appears as many times in the inner summation on the R.H.S. as there are x_i to which it is connected, i.e. k_j times.

Suppose that G represents the pixels within an extracted region, as discussed above, with connectivity signalling adjacency. Suppose that $x_i(0)$ at each pixel is the cartesian coordinate pair of that pixel.

Then after several iterations, each pixel will contain a value (coordinate pair) close to that of the centroid of the region.

A4.2 Velocity Computation Via Intensity Analysis

Suppose we have a rectangular grid, whose rows and columns are indexed by i and j respectively. Suppose a light pattern falls upon some portion of this array, and in the next instant is moved to some other location, but preserving its orientation, shape and internal light distribution. Our aim is to compute the pattern's velocity. We will assume that at each cell there is a detector capable of recording the incident light intensity, and the change in intensity when the pattern is moved.

We will initially look at the case where the pattern is moved in the direction of the rows by a displacement of v units per unit-time.

Take any row i. We will denote the incident light intensity at (i, j) at time t by $f_i(j, t)$. We may for convenience drop the subscript i.

We will assume nothing else about the pattern except that its maximum extent is less than n units from the origin of coordinates, in either direction of motion. We will suppose that the displacement does not put it further than N units from the origin. Thus

$$N - n \geq v.$$

We will take the intensity of the background to be zero. The situation for any row i is depicted in Figure A4.2.

We know that

$$f(j, t) = 0 \quad \text{for} \quad j < 0 \quad \text{and} \quad j > n.$$

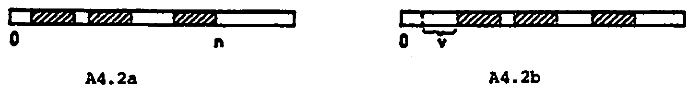


Figure A4.2. Example of Movement. Figure A4.2a shows a representation of a distribution of light intensity. In Figure A4.2b this pattern has moved v units to the right.

$$f(j,\ t+1) = f(j-v,\ t), \text{ for } j = v, \ldots n+v.$$

Define

$$d(j,\ t+1) = f(j,\ t+1) - f(j,\ t).$$

Consider

$$\sum_{-(n+v)}^{n+v} j d(j,\ t+1) = \sum_{-(n+v)}^{n+v} j\,[\,f(j,\ t+1) - f(j,\ t)]$$

$$= \sum_{-(n+v)}^{n+v} j\,[\,f(j-v,\ t) - f(j,\ t)]$$

$$= \sum_{-n-2v}^{n} (j+v)\ f(j,\ t) - \sum_{-(n+v)}^{n+v} j\ f(j,\ t)$$

$$= \sum_{-n-2v}^{n} v\,f(j,\ t) + \sum_{-n-2v}^{1-n-v} j\,f(j,\ t) - \sum_{n+1}^{n+v} j\,f(j,t)$$

where all summations are over j.

But $f(j,\ t) = 0$ for $j < -n-v$ and $j > n$. Therefore

$$\sum_{-(n+v)}^{n+v} j\,d(j,\ t+1) = \sum_{-n-2v}^{n} v\,f(j,\ t)\ .$$

Hence, by adding and subtracting extra terms which are all zero,

$$v = \frac{\sum_{-N}^{N} j\,d(j,\ t+1)}{\sum_{-N}^{N} f(j,\ t)}\ .$$

Our 'detector' had a sufficiently large window to see across the whole pattern. Let us examine what happens if it can only sample a por-

tion of it.

Suppose that the window is of width m at position k, so that the new m is from j = k to j = m + k - 1. Then

$$\sum_{k}^{m+k-1} j\ d(j,\ t+1) = \sum_{k}^{m+k-1} j[f(j-v,\ t) - f(j,\ t)]$$

$$= \sum_{k-v}^{m+k-1-v} (j+v)\ f(j,t) - \sum_{k}^{m+k-1} j\,f(j,t)$$

$$= v \sum_{k-v}^{m+k-1-v} f(j,t) + \sum_{k-v}^{m+k-1-v} [j\,f(j,\ t) - (j+v)\ f(j+v,t)\,]$$

so

$$\sum_{k=-K}^{K}\ \sum_{j=k}^{m+k-1} j d(j,t+1) = v \sum_{k=-K}^{K}\ \sum_{j=k-v}^{m+k-v-1} f(j,t)\ +$$

$$\sum_{k=-K}^{K}\ \sum_{j=k-v}^{m+k-1-v} [j\,f(j,t) - (j+v)\ f(j+v,t)]$$

Consider the first term of the right-hand-side. The double summation has the effect of putting a window of size m across the row and shifting it by one unit each time. Therefore each term will be included m times, except for those towards either end of the row. But for sufficiently large K, $f(j,\ t) = 0$ for j in the neighbourhood of K.

Consider the second term of the right-hand-side. Here there will be cross-cancellation except for terms with j near ±K, but again, these are zero for sufficiently large K. Assuming this is the case, then, we have

$$\sum_{k=-K}^{K}\sum_{j=k}^{m+k-1} j\,d(j,\ t+1) = vm\sum_{k=-K}^{K} f(j,\ t)$$

or

$$v = \frac{\sum_{k=-K}^{K}\left[\frac{\sum_{j=k}^{m+k-1} jd(j,\ t+1)}{m}\right]}{\sum_{k=-K}^{K} f(j,\ t)} .$$

Now, the term in square brackets is the mean of jd(j, t+1) evaluated over a window of size m located with its left hand and at j = k. The outer summation adds these contributions for windows spanning the field of view. The denominator is the total light intensity over the row.

Suppose now the displacement is v units in the j-direction, as before, and w units in the i-direction. That similar analysis holds is shown briefly below.

Using an obvious extension of notation, we can modify (1) to get

$$\sum_{i=-N}^{N}\sum_{j=-(n+v)}^{n+v} jd(i,\ j,\ t+1) = \sum_{i=-N}^{N}\sum_{j=-(n+v)}^{n+v} j[f(i-w,j-v,t)-f(i,j,t)]$$

$$= \sum_{j=N}^{N}\{\sum_{j=-n-2v}^{n} j[f(i-w,j,t)-f(i,j,t)] + v\sum_{j=-n-2v}^{n} v\ f(-w,j,t)\}$$

$$= \sum_{j=-n-2v}^{n} j\sum_{i=-N}^{N}[f(i-w,j,t)-f(i,j,t)] + v\sum_{i=-N}^{N}\sum_{j=-n-2v}^{n} f(i-w,j,t)$$

$$= \sum_{j=-N}^{N} j\{\sum_{i=-N-w}^{N-w} f(i,j,t) - \sum_{i=-N}^{N} f(i,j,t)\} + v\sum_{j=-N}^{N}\sum_{i=N}^{N} f(i,j,t)$$

$$= v\sum_{j=-N}^{N}\sum_{i=-N}^{N} f(i,j,t) .$$

Hence

$$v = \frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N} jd(i,j,t+1)}{\sum_{i=-N}^{N}\sum_{j=-N}^{N} f(i,j,t)}$$

and similarly,

$$w = \frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N} id(i,j,\ t+1)}{\sum_{i=-N}^{N}\sum_{j=-N}^{N} f(i,j,t)} .$$