Edge Detection for Semantically Based Early Visual Processing

Robert J. Beattie

Ph. D.

University of Edinburgh

1984

Abstract

This thesis describes the design and implementation of an edge detection system for use in semantically-based early visual processing. An overall vision system architecture is first selected, based on the need to clearly specify the performance of each module. The role of early processing within this system is then discussed. To specify the edge detection module, both its input and the required output are examined in detail. On the input side, the scene layout and scene-to-image transformation are investigated in both signal processing and information processing terms. On the output side, a particular example of constraint analysis (determining if an image edge could be the result of a reflectance boundary in the scene) is formally analysed in photometric terms. From these considerations, a set of goals of edge detection for constraint analysis are listed.

Since image edges occur over a wide range of scales, the resulting structure of the edge detector is multi-level, i.e. it attempts to find edges of a single scale separately before integrating all of the image edges into a single representation. This is achieved by applying the same "single-level system" to several different-resolution versions of the image, before attempting to construct the final edge representation.

The single-level system finds and tracks step-like edges in its input, and also labels areas of extended intensity gradient and image texture. To avoid making assumptions about the nature of the underlying intensity changes and to ensure that edges are localised accurately, first differencing followed by thresholding is the initial processing step. Thresholding in the presence of noise is modelled by a communication channel, and the best threshold chosen to be that which maximises the information transmitted over the channel. The edge tracker design is based on an analysis of how step-like edges digitise. Image areas of extended gradient and texture are located using a region-finding technique, and identified by examining the smoothness of intensity variations within each region.

The multi-level system consists of two parts, a set of adjacent-level comparators, and a multi-level integrator. By analysing the constraints on the output of the single-level system and the relationship between the outputs of two nearby resolutions, techniques for combining the resulting two sets of edge data in an adjacent-level comparator are developed. One important goal here is to find and represent fuzzy edges. This is achieved by tracking along step edges at the lower resolution, at the same time examining corresponding areas of the higher resolution output. The role of the multi-level integrator is to combine the outputs of the adjacent-level comparators into a single coherent representation of all the edges present in the image.

The algorithms used throughout the system are described in detail, and examples of the system output for a range of images are given.

Acknowledgements

I would like to thank all those who helped me complete this thesis. In particular, I would like to thank my supervisor, Robin Popplestone, and Pat Ambler, who both made many useful comments on drafts of this thesis. I would also like to thank Bob Fisher, who has been a constant source of encouragement, comment, criticism, and discussion.

I acknowledge the financial assistance of the Science and Engineering Research Council.

Declaration

This thesis has been composed and written by myself and the work described in it is my own.

R.J. Beattie

Contents

1 Introduction

The field of computer vision is concerned with methods for automatically extracting scene information from images. Most computer vision systems can be usefully split into two parts; that concerned with what is termed "low-level" vision, and that concerned with "high-level" vision. Broadly speaking, low-level vision is image-based, while high-level vision is model-based.

The role of low-level vision, also known as early visual processing, is to locate and represent particular types of image intensity variations which contain useful information about interesting events in the corresponding scene. For example, it may be desirable in a particular system to find the image curves which are the projection of shadow boundaries in the scene.

This type of early processing normally takes place in two stages; first, places of interest in the image are located, then, they are analysed to extract the desired information about the scene. The former stage is conventionally known as edge detection, although the term covers a wide range of techniques. Some programs known as edge detectors simply mark image locations where the local intensities satisfy a particular property. Others measure the edge strength and direction at each point, while yet others form lines and curves representing the changes in image intensity.

1.1 Area of Investigation

The subject of this thesis is edge detection. The two main problems tackled involve the identification of a suitable methodology for designing edge detectors, and the design and implementation of an edge detection system for a specific task.

The methodology proposed is an overall framework in which to approach edge detector design, rather than a set of design rules or an algorithm for producing variants of a particular type of edge detector. I consider how the design of edge detectors relates to the complete system design, and investigate the role of edge detection within a total system. I believe that arriving at a suitable specification for edge detection within a given system is of crucial importance.

Since it is very difficult to discuss in detail issues such as design methodology in general terms, the design and implementation of an edge detection system for a specific task is also presented in this thesis. The modern conception of general purpose vision systems in which, relatively autonomous low-level processing interfaces to the high-level system, at the level of surface information, is used as the basic system architecture. The specific task examined is that of constructing an edge detector whose output will be used to "label image edges as particular types of scene boundaries", (based on examining the intensities in the neighbourhood of the edge). In other words, we decide on an appropriate set of goals for an edge detector whose output is used as input $\overset{\text{to}}{\text{by}}$ an edge labelling process, and try to design a system to satisfy them.

1.2 Overview

There has recently been considerable interest in the methodology of computer vision system design. To some extent this stems from the work of David Marr, who suggested some general principles based on treating vision explicitly as a complex information processing task. In this thesis, I investigate some of these issues from the point of view of their use in designing edge detectors. I suggest that understanding the scene to image transformation in detail is the fundamental step in designing early visual processors. The thesis is entitled "Edge Detection for Semantically-Based Early Visual Processing", where the term "semantically-based" is used to emphasise what I see as the crucial role of the scene to image relationship, both in forming an overall methodology for designing low-level vision system modules, and in the specific design of particular components.

There are several ways to go about designing and implementing edge detectors. Commonly, a particular mathematical model is chosen to represent the intensities in a small neighbourhood of the image. The model is fitted to intensities in an image neighbourhood, then parameters of the best-fit model are used as a suitable description of the intensities on which to base decisions about the presence of edges. The important first step in this approach is the choice of model. Unless it represents the image intensities in a way which allows the important relationships to be expressed, no amount of ingenuity in model-fitting or parameter extraction will produce the desired result.

By closely examining the scene to image relationship in this work, I have tried to determine the important properties of image intensities. I believe the key to this lies in recognising the veridicality of the image. In other words, image intensities are not probabilistically generated (apart from any electrical noise present), but faithfully reflect the layout of the scene under projection. A direct result of this is a desire to use a simple technique to perform the first stage of edge detection. Complex methods usually enforce the selection of a particular model for the intensity function in a neighbourhood of the picture. The process of fitting the intensities to the model can be thought of as the invocation of a set of constraints about the underlying intensity function, which may not be justified.

Attempting to describe the properties of the image intensity function is effectively a means of defining the edge detector's input. To fully specify an edge detector, at least at the theoretical level, the required output must also be defined in some way. To simplify the specification, I have chosen an overall system architecture which assumes that early processing is autonomous, i.e. not closely directed by the high-level system. There are many ways of performing early processing within this paradigm, for example, stereopsis and the use of optical flow are two popular methods.

As one might expect, no single edge detector can be optimally suitable for all types of early processing. Different methods of using the edge information require it to be found and represented in different ways. So before attempting to design and implement an edge detector, it is advisable to decide what its output is to be used for, and to use this knowledge in specifying and designing the detector.

The edge detector described in this thesis is designed for use with intensity-based edge labelling. Considerable interest has been shown in this type of early processing recently. The basic idea is to examine the patterns of image intensities produced by different types of entity in the scene, e.g. shadow boundaries, reflectance boundaries, etc., and to use this information to label image edges with their scene "meaning". This problem is interesting from the point of view of edge detection because it involves being able to find edges accurately and being able to select intensities from the neighbourhood of the edge in a known way for analysis. Thus it makes quite stringent demands on the performance of the edge detector. In fact, most of the work reported on edge labelling in the literature has used edges selected manually for their input, indicating the need for work in this area.

By selecting a particular use for the edge detector's output, the demands the edge labeller puts on the performance of the edge detector, and on the representation of the edges detected, can be investigated in detail. This enables me to suggest some quite specific performance goals for the edge detector, which can be used as a basis for design.

One particular goal is that the edge detector should find and represent all the significant intensity changes in the image. These typically occur over a wide range of scales. For example, the edge resulting from one object obscuring part of another is normally very sharp, while edges generated by shadows in the scene are often quite blurred. It is very difficult to find both blurred and sharp edges within the same neighbourhood simultaneously, so the normal solution to this problem is to make several passes over the image, each time looking for edges of a different width. Systems with this structure are sometimes referred to as multi-channel or multi-level edge detectors. Unfortunately, the difficult problem with such systems is that of integrating the multiple sets of edge data produced into a single representation. This is hard because some edges occur in more than one data set, and because edges can interact in complex ways in different channels.

I attack this problem by finding edges accurately, and by carefully specifying the output of each channel. By representing edges accurately, it is easy to compare edges in different channels to see if they are in the same place. The point of carefully specifying the single channel output is to make it easier to work out the relationships between channels. Based on this work, I suggest three features which, I believe, it is important to incorporate into an edge detector if it is to be multi-channel.

Like most edge detectors, the one described here involves thresholding. Choice of threshold is usually a difficult issue, especially if the quantity being thresholded is subject to noise. In this thesis I present a new method for threshold selection in the presence of noise. It treats the effect of noise in a similar way to its treatment in a communication channel. In terms of the analogy, the threshold selected is that, which maximises the information transferred over the channel in an information-theoretic sense, by using the noise statistics and the image intensity distribution.

1.3 A Note on Terminology

The terminology of edge detection tends to be confusing for two reasons. Related events occur in the scene, the image, the digitised image, and the edge detector output, and it is not always clear which is being referred to by a particular term. Also, some terms are used both with their general meaning and with a technical meaning. Unfortunately, there does not appear to be any simple way round this problem except by introducing yet more terms for various entities, which should be kept to a minimum.

In this thesis, I try to use appropriate terminology to distinguish scene entities from image entities. The scene layout refers to the position and orientation of objects in the world. The term "boundary" refers to entities in the scene, such as reflectance boundaries where the photometric surface properties change, or surface orientation boundaries, where the local orientation of a surface changes. The terminology is slightly complicated by the fact that the position of the camera in the scene has important implications, and generates another type of boundary where one object passes in front of another. In this thesis, these are called obscuring boundaries. Because obscuring boundaries are particularly important, but depend on the camera position as well as the scene layout, I use the term "scene/imaging" to refer to such entities.

The general term used to refer to entities in the image is "intensity change". The term "edge" is used both in its general sense to refer to a large subclass of intensity changes and, prefixed by an adjective such as "step", to refer to a defined type of intensity change. The term "significant" is initially used to refer, in a general way, to edges which are interesting for some reason. However, significant is formally defined in chapter four and is used according to that definition thereafter. The location or value of the intensity change between two adjacent 4-connected pixels is referred to as a "boundary segment", and the term "region boundary" has its normal meaning. These are the only two exceptions to the use of "boundary" to refer to scene/imaging entities.

1.4 Readers Guide

Chapter two contains the discussion of general methodological issues, and the derivation of the goals of edge detection. It begins by investigating the process of design of vision systems, followed by a discussion of the choice of an appropriate system architecture. This leads to an investigation of the role of edge detection. Then, by considering the input and output requirements for the task chosen, a set of goals for edge detection are selected.

Chapter three is a survey of related work in edge detection and its use in edge labelling. It concludes by suggesting, from considering the results of other research, that the system required should be multi-channel.

The design of the single-level system is begun in chapter four. After a discussion of the issue of representation, the first processing step is selected. Since this involves thresholding, a method of selecting a threshold is required. A new technique for automatic threshold selection in the presence of noise is presented.

The design of the remainder of the single-level system is described in chapter five. This includes pixel labelling, edge tracking, and finding regions of extended gradient and texture. The way in which image texture arises is discussed and a computational definition of texture given.

Chapter six contains the design of the other components needed to construct a multi-channel system. A technique for comparing the outputs of two single-level systems using different resolution images as input is described in detail. From consideration of the issues involved in designing the multi-level system, three main features are suggested as being important in the construction of multi-channel systems.

Finally, the contributions of the thesis are summarised in chapter seven, and some suggestions for further work are made.

2. Towards a Theory of Edge Detection

2.1 Introduction

In this chapter an attempt will be made to specify the goals of edge detection for analysing monochrome images of three-dimensional scenes. It will be argued that it is only by first choosing an architecture for the complete vision system, then specifying the role of early processing, and within that the role of edge detection, that the desired output of the edge detector can be defined.

It seems likely that any high performance computer vision system will be an entity of great complexity. To understand and construct such systems, it is essential that they consist of a number of communicating modules. The overall system structure and the interfaces between the modules must be clearly specified. One of the greatest problems in building large A.I. programs has been the difficulty in imposing a coherent overall structure and in defining the interfaces. This is due to two related problems: lack of underlying theory and lack of rigour in system specification. The result of this is that while these programs may do something, perhaps even something useful, they have had no lasting value because they cannot be related to other work in the field, and though a problem may have been solved, it's often not quite clear which problem has been solved.

These criticisms are particularly pertinent to work in computer vision, since it is usually clear what the desired output of the overall system should be for any given input. Specification of the role of individual modules is not, however, so straightforward, since it is often not obvious how the system should be decomposed. Additionally, the decomposition will depend on the information being used by the system, e.g. the structure of a system based on exploiting optical flow may well be different from one using intensity-based edge classification. One breakdown of a vision system has been proposed by the M.I.T. vision group (p.4 Grimson 1981). The differences between that structure and the one described in this thesis will be discussed in chapter three.

The goal of a 3-d vision system is normally to describe the imaged scene in some way, often in terms of surface layout or known objects, so the appropriate method of analysis must be strongly influenced by the three-dimensional nature of the scene. Since the scene is subject to physical laws and produces the corresponding image via a deterministic imaging transform, one might expect scene analysis also to be supported by a firm theory.

David Marr had much to say on these issues. He suggested that any complex information processing task has to be understood on three different levels (Marr 1982). The top level being that of computational theory. This specifies what problem is being solved, what the basic constraints affording a solution are, and what the strategy of the solution is. The middle level is that of the algorithm. This specifies in a formal way how the solution can proceed: there can be many algorithms for any given computational theory. The bottom level is that of the implementation. It is at this level that the theory is ultimately used to solve problems and produce practical results. There can be many implementations of any given algorithm. In terms of edge detection, the computational theory should first of all specify what the role of edge detection is in terms of its overall objectives and its input/output behaviour. Then by analysing the form of the input available to the edge detector, it should outline the properties of and constraints on the input which lead to an overall strategy for achieving the desired performance. This leads to the design of algorithms to meet the goals specified by the theory, utilising the constraints provided by the analysis. The specification of the algorithms should be sufficiently detailed to allow straightforward implementation of them in any particular programming language - the implementation level.

It is useful to bear this tripartite division in mind, although the three levels are usually not independent. In the case of early processing, the characteristics of the imaging device, nominally an implementation-level consideration, certainly affect the development of the computational theory. Many special-purpose imaging devices and techniques, such as triangulation rangefinders (Popplestone et al 1975) or the use of photometric stereo (Woodham 1978), have been used to simplify the accompanying computational theory of scene analysis. Similarly, the type of computer on which the system is to be implemented has a strong effect on the language used for describing algorithms and, inevitably, on the algorithms themselves. This issue is particularly relevant to the case of early processing where there may be good opportunities to use parallel processing techniques. Anyone who has programmed regularly on a serial machine has probably adopted serial thinking habits unconciously. This is another good reason for emphasising the importance of also working at the level of computational theory, not just at the level of algorithms.

Elsewhere, Marr (1977) specified in a related way that a result in A.I. should consist of the following:

1. isolation of a particular information processing problem
2. formulation of a computational theory for it.
3. construction of a set of algorithms that implement it

4. a practical demonstration that the algorithms are successful.

In practice these four categories are rarely as well defined as one might hope. In particular, isolation of a particular problem and the formulation of a computational theory of it are sometimes closely linked. In a large and complex system it may be necessary to have an overall computational theory in mind before the individual modules can be identified, their theory developed, and algorithms constructed. This is certainly true in the case of vision. Different vision system architectures can require quite different performance from the early processing module. In this chapter an attempt is made to isolate and specify the problem of edge detection for monochrome images of 3-d scenes. In view of the number of edge detectors that have been presented in the literature, it may seem unnecessary to design yet another, but it can be argued that many of the edge detectors so far constructed have been designed on a fairly ad hoc basis. It's often not clear what the goals of a particular edge detector are, nor how the output is related to the input, i.e. the image. It is interesting to note that many edge detectors have been proposed with no reference to the vision system of which they are to be a component. Indeed, a separate discipline of edge detection seems to have arisen with no obvious means of comparing the performance of different edge detectors. This issue will be further discussed in chapter three, where the field of edge detection will be surveyed.

The construction of a general purpose vision system, i.e. one able to deal with a wide variety of scenes, has only recently become a feasible project due to increased understanding of what the early processing can, and should, deliver. Previously, various attempts to get round this problem were tried. Some programs working on the blocks world (e.g. Waltz 1975) used a perfect line drawing as input, assuming that such a drawing could be obtained from an image. This was a dangerous assumption because it was rarely the case in practice that such a perfect line drawing could be easily obtained. Later, systems working on more general images used edges derived by hand. Again, this is dangerous because it's not clear that a subjectively "reasonable" hand segmentation could be achieved automatically in practice. Also, hand segmentation by the system designer could unconciously incorporate high-level, scene-specific or program-specific knowledge in reaching the final segmentation.

This problem of defining the input doesn't arise in the same way for edge detection, the input is fixed once a particular class of imaging devices has been chosen. Thus, edge detection is a good place to start to try and develop computational theories of vision because the imaging transformation is known. By trying to specify an allowed range of scenes or equivalently by listing a set of constraints on the types of scenes to be dealt with, a reasonable attempt can be made to produce a specification of the image.

2.2 Choosing a Vision System Architecture

Before trying to specify in detail the goals of early processing and then edge detection, it is first necessary to have some idea of the overall role of early processing within the vision system. It is generally accepted that a useful division can be made between low-level vision and high-level vision. Low-level vision is taken to encompass such functions as edge detection and region finding, whereas high-level vision normally refers to tasks such as object recognition. The reason for this split is that low-level vision bears a close relationship to the scene being investigated, and is always data driven to some extent, whereas high-level vision is mainly concerned with performing a specific task in a goal-directed way. In research in computer vision, two quite different types of relationship have been proposed between the low- and high-level vision systems, crucially affecting the role of early processing.

In the first paradigm, the vision system is seen as being roughly hierarchical with the early processing being autonomous and entirely data-driven. This is based on the observation that the image is veridical and on the belief that generally useful constraints can be found, which, operating at the early processing level, allow an appropriate segmentation of the image to be made. The veridicality of the image arises from the causal relation of the image to the scene. If the scene changes in some way then the image will also change in an entirely predictable way. Founded on this relationship, and on the physically describable nature of objects in the world is the hope of formulating a set of constraints providing the basis of a segmentation strategy. However, for this to be successful, the scene has to be physically describable in a certain way. What is required is a set of laws governing the behaviour and interactions of light sources, objects, and imaging devices at the level of surfaces. This has not yet been satisfactorily achieved although there are encouraging signs (Barrow and Tenenbaum 1978, Binford 1981, Horn 1977).

In the second paradigm, low-level vision is seen as a more goal-directed process, controlled to a greater or lesser extent by the high-level vision system. This, it is claimed, is necessary to deal with the severe difficulty of early processing caused by electrical noise and the complexity of the scene. Every available piece of knowledge and data should be brought to bear on each part of the analysis task. The principle of operation is that the system begins by doing the easier bits of the early processing, then does as much high-level processing as can be done with this initial segmentation. These intermediate results of high-level processing are used to guide the low-level system to a more refined segmentation, which in turn leads the high-level system to reevaluate its output, and so on in this cyclic fashion until a satisfactory overall result is achieved.

This idea has been further generalised in the case of what are known as "heterarchic" systems. These are well described by the phrase "community of experts", and usually consist of a set of knowledge sources, a global interface/data structure for communication between knowledge sources, and an executive for deciding issues such as what should be done next. The best known example of such a system is the Hearsay speech understanding system (Lesser and Erman 1977) which apparently performed well, but it should be noted that the task of speech understanding is fundamentally different from that of scene analysis. In vision, as mentioned above, the input is an image which is veridical, i.e. faithfully informs about the scene. Also, the imaging transformation is always the same for a particular imaging device. In the case of speech understanding the transformation from an intended utterance to pressure waves varies widely, not only from speaker to speaker, but also in separate utterances by the same speaker. This is most easily seen in the case of people with different accents speaking the same language: the same words have a very different sound. Thus, the problem of speech understanding is quite different from that of vision since the transformation from intended utterance to pressure waves cannot be described as veridical in the same way as the transformation from scene to image. The fact that the Hearsay system has been a successful one as far as speech understanding is concerned does not mean that the same type of structure is necessary to analyse a scene successfully.

Chronologically speaking, the first computer vision systems used autonomous early processing, then, later, goal-directed vision became popular. Recently, autonomous early processing has staged a strong comeback and is again the most popular.

The first influential piece of work in scene analysis was by Roberts (1965). A photograph of some white polyhedral blocks on a dark background was digitised to provide the input. The desired output was a set of 3-d positions for certain prototypical object models whose projection "explained" the lines found in the image. The structure of the system was simple; first, a line drawing was derived from the image, then object models were matched to the line drawing. Roberts found that obtaining a good line drawing was the most difficult part of the process. He had three criteria for his edge detection process

- the edges produced should be as sharp as possible
- the background should produce as little noise as possible
- the intensity of the lines produced should correspond closely to a human's ability to perceive the edge in the original picture.

The first two of these goals are strongly dependent on the domain of the program. One can hope to find sharp edges corresponding to the edges of blocks, with relatively uniform, noise-free regions between them corresponding to the planar faces only in simple blocks world scenes. Even in this specially chosen domain, shadows and surface markings can lead to problems. One of the main criticisms of Roberts's work is that in his early processing stage he used an over-simplified scene/image model. If his early processor failed to produce a perfect line drawing, the model matching routines would also fail since they discarded lines not attached to junctions at both ends. The main point to note is that after a line drawing had been produced, no further examination of the image or reevaluation of the result of early processing (i.e. the line drawing) took place.

It was discovered that even with such scenes as white blocks on a dark background, it was sometimes not possible to achieve the perfect segmentation required with a single pass of, for example, a Roberts-type edge detector. This led to the development of heterarchical systems, the best known of which in vision is Shirai's (1973). Again this dealt with light coloured blocks on a dark background. Shirai classified the edges into three types:

contour - between object and background

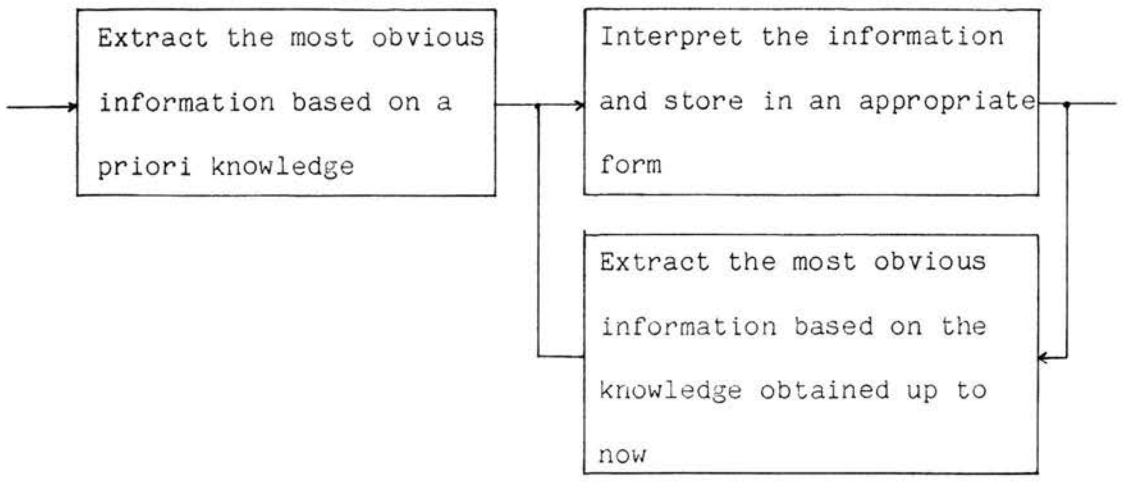
boundary - between two objects

internal - between faces of the same object

He assumed that contour edges could easily be found. Then, based on two guidelines:

- find boundary lines before internal lines
- first look for lines which have smaller search areas

he developed a set of ten heuristic rules for guiding the search for new lines based on the current interpretation of the lines previously found. The overall structure of the system is described by the block diagram below:


Extract the most obvious information based on a priori knowledge
Interpret the information and store in an appropriate form
Extract the most obvious information based on the knowledge obtained up to now

In designing his particular edge detection algorithms, Shirai took into account the various types of edges that arise in polyhedral scenes (Herskovits and Binford 1970). Also, because his system always looked for a specific edge in a specific place, he could use sensitive detectors without incurring prohibitive processing times or too many spurious edges.

Although this was one of the most successful programs in dealing with the blocks world, it did have some problems. In certain circumstances it could miss objects altogether, and the heuristics could fail with concave polyhedra.

It is interesting to consider why Roberts, the first person in the field, took a hierarchical approach, whereas Shirai, who came to the same problem ten years later, decided to use the more complex heterarchical system. This can be related to the different problems they faced. Roberts, meeting a new problem, had to understand the basic structure of the solution and identify the available constraints. By the time Shirai was working, much more had been done on blocks world vision and many of the constraints were well known. His program can be thought of as a reconfiguration of the knowledge in a more complex way using a variety of special tools to solve particular

problems. In other words, his main contribution with this piece of **work was** in the field of controlling or structuring vision algorithms.

This pattern of first understanding the problem, finding the basic constraints and implementing them in a system with a relatively simple structure, then later reconfiguring the system in a more complex way for better performance, can be expected in other problems. Although there may be some deep equivalence between hierarchical and heterarchical systems operating on the same basic system of constraints, it's much easier to develop theories and find constraints within the more comprehensible structure of the hierarchical system. The "community of experts" idea is a seductive one, but leads to difficulties in specifying the role of each knowledge source and the interfaces between them, especially if the basic theory of the system is not fully worked out. The problem of edge detection for complex scenes is not yet sufficiently well understood in terms of the basic principles and constraints to allow a heterarchical approach.

Recent advances in vision have come about as a result of adopting a more rigorous methodology as detailed in the previous section. This attempt to modularise vision, and to specify and design individual modules has naturally led back to the idea of autonomous early processing. The emphasis must be on understanding the problem and providing an acceptable theory; restructuring the system for better performance can come later. Using this more rigourous methodology, several pieces of work have been described in the literature which seem to be leading towards a solution to the problem of early visual processing. Probably the best known of these is Marr's work (see Marr (1982) in the first instance), his theory of edge detection will be discussed in detail in chapter three. Barrow and Tenenbaum (1978) and Binford (1981) have also made important contributions, their work will also be described later. The real value of this type of work is that the constraints identified are of lasting value, and are not dependent on any particular algorithm or set of hardware.

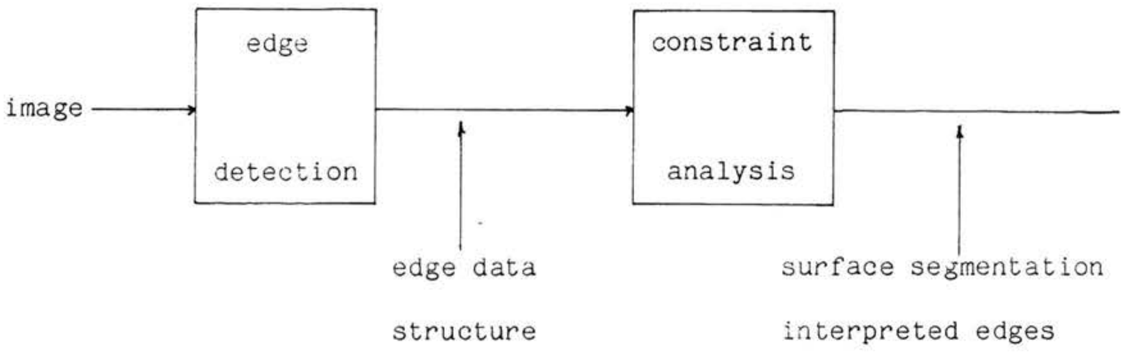
However, there is one final caveat. Since the problem of early processing has not yet been "solved" there is always the possibility that autonomous early processing won't work and the extra complexity of heterarchic systems will be needed to solve the problem at all. This seems unlikely in vision since to date heterarchical systems have simply given improved performance. They have not been shown to work on problems where hierarchical systems have failed completely. Since the object of this thesis is to develop a theory of edge detection for monochrome images, it is more appropriate to do it within the framework of autonomous early processing. The basic issues are more likely to come to the fore within the simpler hierarchical structure than in the more complex heterarchical type of system.

2.3 The Role of Edge Detection in Early Processing

Having decided on an autonomous early processing system working on a single monochrome image as input, it is now necessary to clarify the desired output of early processing and the role of edge detection. Based on the realisation that the image is veridical and only makes sense as a projection of a 3-d scene, the objective of early processing is considered here to be the interpretation of the image data in terms of 3-d scene/imaging events. The term "scene/imaging" is used because the generation of, for example, obscuring edges depends on both the layout of the scene and the position and orientation of the camera within the scene. This has been shown to be feasible, at least for a limited domain by Barrow and Tenenbaum (1978), who constructed a world model consisting of a certain limited class of objects under restricted lighting conditions. By then analysing the types of edges which were generated in images of such scenes in terms of relationships on image intensities across and along the edges, they were able to identify a number of tests which could be used to identify some of the edges as reflectance edges, etc. Their ultimate objective was to produce "intrinsic images" of reflectance, incident light, range, and surface orientation, evaluating these quantities at each point in the image. Unfortunately, it does not seem likely that this can be achieved with a single monochrome image for any general class of scenes, because it requires quantitative photometric analysis of every aspect of the scene. Thus, the output of early processing for this particular approach is likely to be a compromise between the desirable and the achievable. A more realistic goal is to segment the image into surfaces, with every image "edge" labelled with its scene interpretation. Possible interpretations for such edges would be obscuring edge, shadow edge, reflectance edge, surface orientation edge, and highlight. The problem of texture will also have to be investigated if the system is to be applied to a wide class of scenes. There is some indication that this is a realistic goal, although the problem has not yet been solved (Ullman 1976, Binford 1981, Fischler et al 1982, Witkin 1982).

Achieving this segmentation would seem to fall naturally into two stages. First, find all the places in the image corresponding to significant scene/imaging events. Given the veridicality of the image, any intensity variation not attributable to noise must have been caused by some variation in incident light, surface reflectance, surface orientation, or surface range. In this sense, all non-noise intensity variations are significant. However, the majority of image intensity changes are due to smooth variations in incident light or surface shape and these are not particularly informative, unless one is interested in recovering local surface shape. Discontinuities or steep gradients in image intensity are more interesting because they are the result of important changes in the scene - it is these which are referred to by the term "significant". This definition will be extended and formalised in section 4.2.

Second, given this set of intensity changes, interpret each one and label it with its scene/imaging meaning. Thus, early processing is seen as edge detection followed by constraint analysis as shown below:



This thesis is concerned with specifying, designing, and implementing the edge detection block in such a system. The input to the edge detector is a monochrome image, the output remains to be specified, but, broadly speaking, should be the complete set of image edges in a representation suitable for constraint analysis. Clearly, the exact role of edge detection depends critically on what is a useful input to constraint analysis. (The importance of taking into account the requirement of the succeeding stage of processing in designing a particular system component has previously been pointed out by Hildreth (1980).) In general terms, it should be possible to relate edges to the underlying image intensities in their vicinity, because one aspect of constraint analysis involves various tests on the pattern of image intensities across and along edges. Attention must also be paid to finding and representing edges as accurately as possible, and to maintaining connectivity relations between edges.

However, before an attempt can be made to design an edge detection system, the goals of edge detection have to be stated precisely, and a better understanding of the relationship between scene and image is necessary. For example, what is meant by the term "edge" and how does it relate to the scene? To achieve this both the scene and the process of imaging have to be investigated at the appropriate level. In the next section the imaging process will be analysed rather formally, and this clarifies the meaning of a number of terms which will be used later. Then, in the following two sections, a further analysis of the scene and imaging will be carried out in terms of surfaces, light intensities, and projections. This will enable a more specific set of goals for edge detection to be delineated which can act as a basis for the construction of algorithms.

2.4 The Imaging Transformation

Three separate entities are identifiable in the imaging transformation; these are

- the scene
- the analogue image
- the digitised image.

The geometrical relationship between points in the scene and corresponding points in the analogue image is, for a reasonable camera, adequately modelled by perspective projection. It is this relationship which forms the basis of the claim for image veridicality. However, a computer vision system doesn't have access to the analogue image, only to the digitised version. The process of digitisation, sampling followed by quantisation, severely affects the relationship between scene and image, and will often be mentioned in this thesis.

2.4.1 Imaging

Light from the scene is focussed through a lens system onto the target of the imaging device. It is assumed in this thesis that imaging is ideal and that the target is a bounded rectangular plane. The image irradiance, E, which is the incident flux density (in W/m^2) on the target, is taken to be a function of the position on the target, and the wavelength of light.

$$E=E(x,y,\lambda)$$

where x,y are independent and bounded, with

$$x_1 \leq x < x_m$$

$$y_1 \leq y < y_m$$

2.4.2 Noise

Various sources contribute to what can be regarded as random noise in the image intensity. These include defects in the imaging device (Brain 1979) and noise in the electronics. For many systems, and for the one used here in particular (Beattie 1980), the noise probability density function (pdf) can be described by the Gaussian curve

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}}$$

where σ is the noise standard deviation, and μ is the mean. This is conventionally written $z=N(\mu,\sigma^2)$. We will later use the fact that the sum or difference of two variables with Gaussian pdfs is another variable with a Gaussian pdf (p.740 Kreyszig 1972). If

$$z_1 = N(\mu_1,\sigma_1^2) \qquad z_2 = N(\mu_2,\sigma_2^2)$$

then

$$z_1 - z_2 = N(\mu_1 - \mu_2, \sigma_1^2+\sigma_2^2)$$

2.4.3 Sampling

The first stage in the conversion from analogue image intensity to an array of quantised values is sampling. For typical sensors this is not adequately modelled by the Dirac delta function, but involves integrating the product of image intensity and some response function over time at a large number of sampling points on the image plane. The response function defines the performance of an individual receptor, e.g. a single photodiode in a photodiode array, and has three arguments, two for the spatial dimensions and one for the wavelength of the incident light, $S(x,y,\lambda)$. It is assumed that in spatial terms S simply integrates the incident light equally over a rectangular area $\Delta x,\Delta y$,

i.e.

$$S(x,y,\lambda) = \begin{cases} k(\lambda) & |x| < \frac{\Delta x}{2}, \ |y| < \frac{\Delta y}{2} \\ 0 & \text{otherwise} \end{cases}$$

This is a reasonable approximation to the performance of many practical imaging devices. It is also assumed that the spectral response of S is normalised to some central frequency λ_0, as shown in fig.2.1.

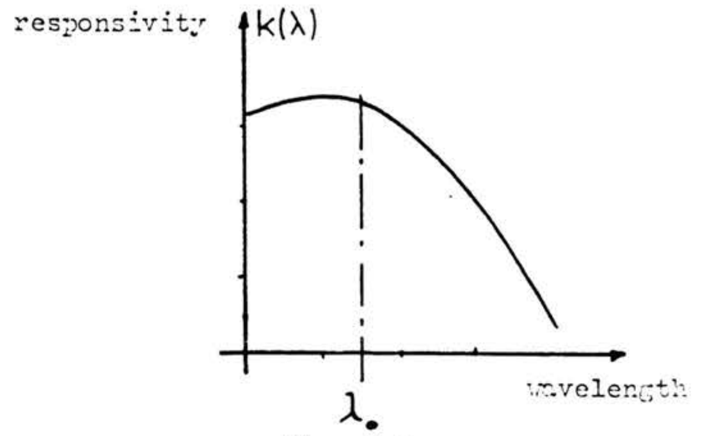


Fig. 2.1

Typical receptor spectral response.

Thus the response of a single receptor cell centred on (x_0,y_0) is

$$C(x_0,y_0) = \int_{\infty}^{\infty}\int_{\infty}^{\infty}\int_{\infty}^{\infty} E(x,y,\lambda).S(x_0-x,y_0-y,\lambda_0-\lambda)\ dx\ dy\ d\lambda$$

In fact, the output of typical imagers is the time integral of C over some period. It is assumed that both S and E are constant over time so:

$$D(x_0,y_0) = T\ C(x_0,y_0)$$

where T is the period of integration. Most real imaging devices consist of a rectangularly spaced array of receptors, which will be denoted by R. To produce an m*n array by sampling over the area of the image plane already defined by x_1, x_m, y_1, and y_m, then:

$$\delta y = \frac{y_m - y_1}{m}$$

$$\delta x = \frac{x_m - x_1}{n}$$

and the i,j th. component of R will be given by

$$R(i,j) = D(\ x_1 + (j - \tfrac{1}{2})\delta x,\ y_1 + (i - \tfrac{1}{2})\delta y)$$

where i refers to the row coordinate and j to the column coordinate. For the sampling areas to cover the image without gaps between individual receptors:

$$\delta x = \Delta x$$

$$\delta y = \Delta y$$

produces the situation shown in fig.2.2. Some cameras have gaps between the receptors in either or both of the x and y directions. The model decribed here can easily be extended to deal with these cases.

2.4.4 Quantisation

The final step in producing the pixel array is that of quantisation. Each R(i,j) is compared to a set of decision levels. Each adjacent pair of decision levels has associated with it a unique code. For convenience, the pairs of decision levels are often num-

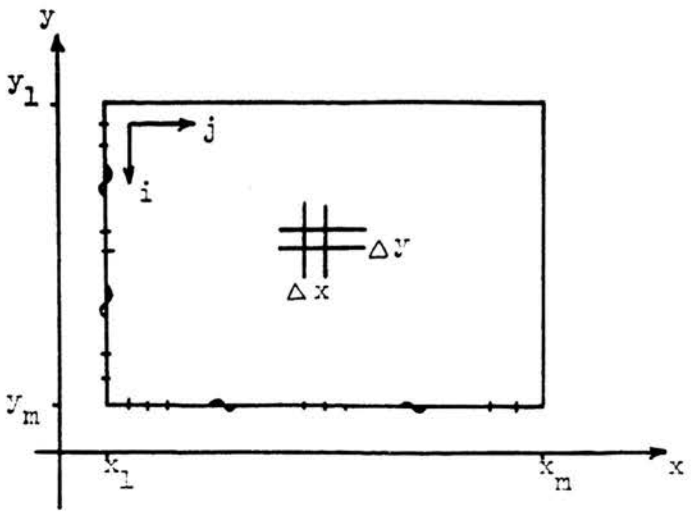


Fig. 2.2
Image plane layout.

bered with integers as shown in fig.2.3. The only case considered here is that of regularly spaced decision levels, thus, given a particular $R(i,j)$, the quantised value, $q(i,j)$, is

$$q(i,j)=n$$

where

$$n\Delta q \leq R(i,j) < (n+1)\Delta q$$

where Δq is the spacing between quantisation thresholds, which completes the transformation from image irradiance to greyscale pixel value.

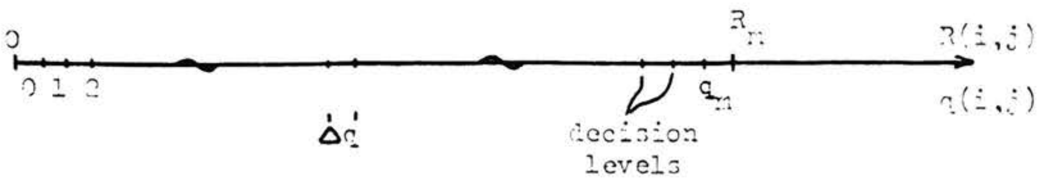


Fig. 2.3
Intensity quantisation levels.

2.5 The Relationship Between Scene and Image

While the above mathematical analysis of the process of forming a digitised image is useful, it is even more important to perform an analysis in terms of information processing. After all, the object of a vision system is to infer information about the scene, given the digitised image. Within the constraints imposed by the imaging system, the image is veridical. That is, it faithfully informs about the scene producing it. Any relationship among intensity values is not a random occurrence (apart from a noise component), but is the causal result of the way the scene is. Examining the possible range of scenes in very general terms should be useful in helping to specify the goals of edge detection more closely.

The world contains objects, light sources, and a transparent medium. Assuming that the imaging device is a monochrome T.V. camera, then we are interested in how light is reflected from objects to the camera. The imaging equation (Horn and Sjoberg 1978) shows that image irradiance is directly proportional to surface radiance. So if we examine the behaviour of surfaces in terms of surface radiance, our observations will be easily expressible in terms of image irradiance. The radiance of a Lambertian surface varies continuously across the surface except under two conditions:

1. a step change in surface reflectance
2. a step change in surface irradiance.

The step change in surface irradiance would normally be at a shadow boundary caused by another object obscuring a light source. For a step change, the light would have to be from a point source. The step change could also be caused by indirect illumination, but this is very unlikely. Indirect illumination usually produces smooth gradients in surface irradiance. We only consider Lambertian surfaces in this section because our main purpose is the derivation of the edge detection process, not the construction of a full world model. This does not mean that our early processing system will be limited to scenes containing only Lambertian surfaces, since at this stage analysis of the world only serves to produce goals for edge detection.

Consider the illumination of two adjacent elementary areas on the image plane. Two cases exist as shown in fig.2.4:

1. they can be illuminated by adjacent areas of the same surface
2. they can be illuminated by unrelated areas of different surfaces.

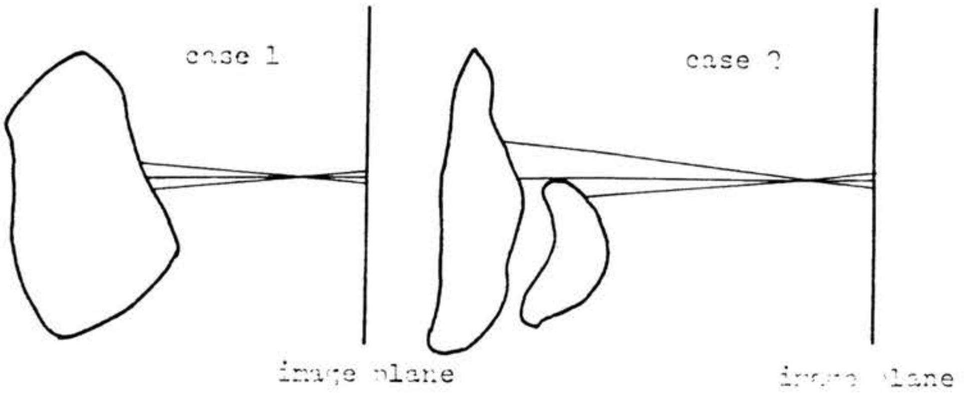


Fig. 2.4
The two ways of illuminating adjacent image plane areas.

In the first case what we know about surface radiance also holds for image irradiance, since there is a direct mapping from adjacent pieces of surface to adjacent pieces of image. In the second case it is likely that the adjacent image areas will be subject to quite different amounts of irradiance, producing a discontinuity known as an obscuring edge. In practice, due to the defects of practical devices these discontinuities will be blurred (see Brain (1979) for optics, Herskovits and Binford (1970) for empirical results). Steep gradients in image irradiance can also be caused by shadows from extended sources and from variations in object shape, such as the edges of polyhedral blocks. The analogue image, then, contains a wide range of intensity variations (assuming perceived intensity is proportional to image irradiance), ranging from steep gradients to uniform regions.

In processing the image we should be particularly interested in places where the intensity is changing quickly, since we might be able to recognise these as due to object boundaries, shape changes, shadows, or reflectance changes. Of course, a computer vision system only has access to a digitised version of the image. So, while it may be relatively easy to formally define notions of continuity and discontinuity for analogue signals, it is not so easy for digitised ones. Edge detection, which can be viewed as a search for significant gradients in the digitised image, has reflected this difficulty. As shall be seen in the next chapter, some methods treat the digitised image as if the underlying signal is continuous everywhere, whereas others assume that the underlying signal contains ideal step edges. As we have noted above, the analogue image does, in fact, contain intensity variations over a wide range of scales, limited at the upper end by the optical and electronic signal processing capabilities of the imaging device.

The process of digitisation raises significant issues itself. Some important image effects are largely dependent on the sampling and quantisation resolutions of the imager. Consider an image of a chessboard pattern. If each black or white square covers many pixels, then it will be easy to identify each square as a separate region. At the other extreme, if many squares are subtended by each pixel, the image will be a uniform grey. Only in some intermediate range where the number of squares subtended by the image is of the same order as the number of pixels will the image take on a textured appearance. The concept of texture is largely to do with relative sizes and resolutions. Related to this is the question of object shape variations which occur over a wide range of scales both in the scene and in the image. In images of natural scenes these variations will cover a range from step edges to regions of uniform intensity. What constitutes a step edge is highly dependent on the relative size and distance of the phenomenon, the optical imaging system, and the resolution of the device. The corner of a building may look like a step edge from a distance, but appears as a gradual shape change close up. One cannot get rid of this problem simply by using higher resolution imaging devices. In terms of the above example, that just means you have to retreat a little further before the corner of the building looks like a step edge again.

The point of this discussion is to illustrate that the concept of "edge" is a complex one, especially in digitised images. Since the scene events which are most informative (e.g. reflectance boundaries, shadows, etc.), produce discontinuities and steep gradients in analogue image intensity, we are interested in places in the digitised image where the intensity values are changing quickly. The difficulty is in being any more specific at this level. If we design a mask which will react to some specific pattern, say, ideal step edges, we will miss the gradients which are also important. Additionally, the characteristics of an edge may change along the edge, so any simple approach seems certain to fail. In other words we must be careful not to over-specify what we are willing to accept as a significant entity in the digitised image because that will find only some of the legitimate edges.

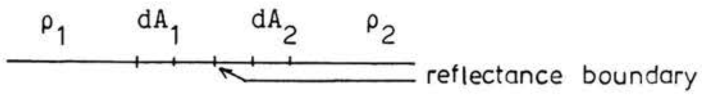
2.6 The Goals of Edge Detection

Having found the places in the digitised image where significant intensity changes take place, it is necessary to represent those changes in a way useful for later processing. By investigating one specific case in detail, we should gain a much better idea of what is a suitable representation. Hence, to identify the specific goals of edge detection we consider the particular case of a reflectance boundary. By analysing its transformation from scene to image to digitised image, it is possible to gain a better understanding of how image edges are formed and relate to their counterparts in the scene. More importantly, we can also take note of any assumptions made in the derivation of constraints on image edges and determine how these should affect edge detection.

First, consider a smooth, not necessarily planar, surface on which lies a sharp boundary in reflectance. In a recent standardisation of photometric terms (Nicodemus et al 1977) the reflectance of a small surface patch is defined as the dimensionless ratio of reflected flux to incident flux and is denoted by ρ, i.e.

$$\rho = \frac{\phi_r}{\phi_i}$$

Consider a small thin strip of the surface <u>approximately perpendicular</u> to the reflectance boundary, and imagine looking at this strip "side on":



To simplify the analysis assume that the surface is Lambertian, and consider two small areas dA_1 and dA_2 near the reflectance boundary. The reflectance of surface patch dA_1 is ρ_1, and that of dA_2 is ρ_2. The irradiance of a small surface patch, dA, is the incident flux density

$$\text{irradiance } (E) = \frac{d\phi_i}{dA} \quad \frac{W}{m^2}$$

The radiant exitance is the exitant flux density

$$\text{radiant exitance } (M) = \frac{d\phi_r}{dA} \quad \frac{W}{m^2}$$

Thus, the reflectance of a surface can also be defined as

$$\rho = \frac{M}{E}$$

The radiant exitance, being the total exitant flux, can also be described in terms of the radiance (L_r) which is the flux emitted per unit surface area per unit projected solid angle. By integrating over the projected solid angle (Ω) we get

$$M = \int_{\Omega_r} L_r \, d\Omega_r$$

For a Lambertian surface L_r is the same in all directions, so

$$M = L_r \int_{\Omega_r} d\Omega_r$$

$$M = \pi L_r$$

So we have

$$\rho = \frac{M}{E} = \frac{\pi L_r}{E}$$

for our two small areas dA_1 and dA_2, we have

$$\rho_1 = \frac{\pi L_{r1}}{E_1} \qquad \rho_2 = \frac{\pi L_{r2}}{E_2}$$

Assuming that the distance between the areas is small so that $E_1 \approx E_2 = E$ then

$$L_{r1} = \frac{\rho_1 E}{\pi} \qquad L_{r2} = \frac{\rho_2 E}{\pi}$$

Now the imaging equation as expressed by Horn and Sjoberg (1978) is

$$I = L_r \left(\frac{\pi}{4}\right)\left(\frac{d}{f_p}\right)^2 \cos^4\alpha$$

where I is the image irradiance, d, f_p, and α are as shown in

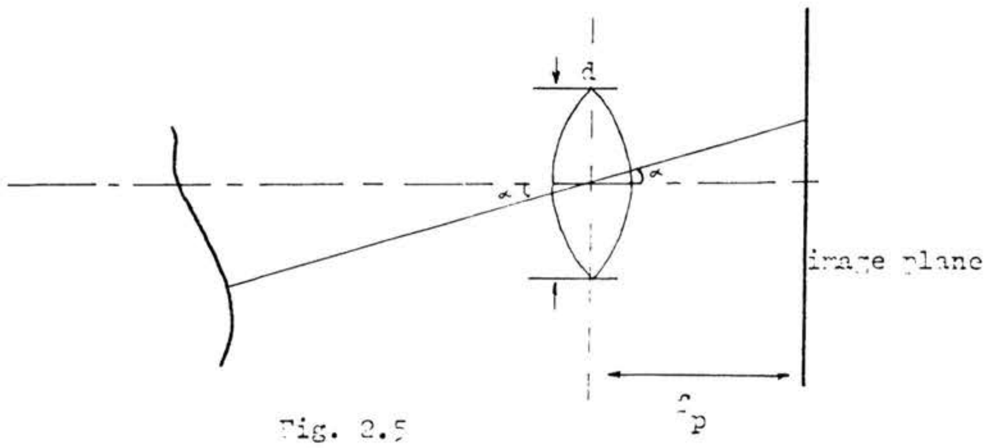


Fig. 2.5
The imaging model.

fig.2.5. So

$$I \propto L_r \cos^4\alpha$$

For our two small surface patches, the corresponding image areas will have

$$I_1 \propto L_{r1} \cos^4\alpha_1$$

$$\propto \frac{\rho_1 E}{\pi} \cos^4\alpha_1$$

$$I_2 \propto \frac{\rho_2 E}{\pi} \cos^4\alpha_2$$

and the ratio of these will be

$$\frac{I_1}{I_2} = \frac{\rho_1}{\rho_2}\frac{\cos^4\alpha_1}{\cos^4\alpha_2}$$

Since we have already assumed that dA_1 is near dA_2 it is reasonable to also assume that $\cos^4\alpha_1 \approx \cos^4\alpha_2$ (or the difference could be corrected for) giving

$$\frac{I_1}{I_2} = \frac{\rho_1}{\rho_2}$$

So the ratio of image irradiance is equal to the ratio of surface reflectance. Assuming that image intensity is proportional to image irradiance then $\frac{P_1}{P_2} = \frac{\rho_1}{\rho_2}$ where P_1, P_2 are the intensities of the appropriate pixels.

Looking back over this derivation, the main assumptions were:

1. the thin strip of surface is approximately perpendicular to the reflected boundary.
2. the surface is Lambertian
3. the distance between the areas is small with respect to the variations in incident light
4. image intensity is proportional to image irradiance.

It is convenient to deal with these in reverse order, starting with the requirement that image intensity is proportional to image irradiance. If digitisation was accurately modelled by applying the Dirac delta function at each position in a grid of sampling points, then we wouldn't have to worry about this assumption. Unfortunately, as noted in section 2.4, this is not the case and normal imaging devices effectively integrate the image irradiance over some small image area. If we suppose the image of a sharp reflectance boundary to be a step edge in image irradiance and imagine superimposing the edge on a digitisation grid, in the general case we get something like the effect shown in fig.2.6 where image irradiance is integrated over each rectangular area, then quantised, to give the corresponding

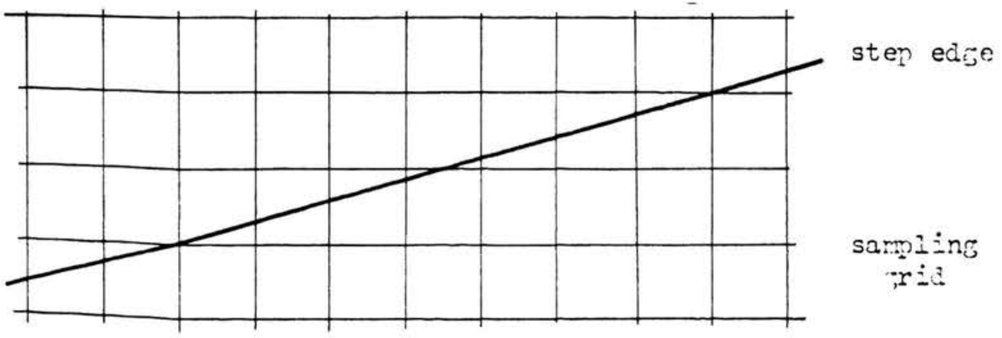


Fig. 2.6
Digitising a step edge.

pixel intensity. Clearly, those pixels through which the edge runs (known as mixed pixels) will have an image intensity which depends on the position of the edge within the pixel as well as the intensities on either side of the edge. These pixels obviously violate assumption 4 and should be avoided when testing ratios. Of course, since point sampling is not the case, no pixel is ideal, but mixed pixels should certainly be avoided.

Mixed pixels arise because of both aliasing and the non point-like nature of the sampling function. One technique that has been used to eliminate this problem is matched Gaussian filtering of the image to remove high frequency components and enhance edges in the selected frequency band. While effective, this method reduces the accuracy of edge localisation because the higher frequency components in the edges are removed. To achieve maximum accuracy in edge localisation the edge detector described later in this thesis does not prefilter the image before searching for edges. This means that steps will have to be taken to deal with mixed pixels explicitly.

To satisfy assumption 3, we must choose pixels as close to the edge as possible in order to minimise differences due to intensity gradients. This is somewhat at odds with assumption 4 which indicated that pixels actually on the edge, i.e. mixed pixels, were of no use. In other words, assumptions 3 and 4 taken together imply that for constraint analysis we want access to pixels as close to the edge as possible as long as they are not mixed pixels. To make sure we find pure (i.e. not mixed) pixels as close to the edge as possible we must find and represent the edge accurately. In terms of actually performing the ratio test, one can attempt to measure the intensity gradients on either side of the edge and factor out the variation (Ullman 1976).

For a non-Lambertian surface, L_r is a function of the amount and direction of the incident flux and the exitant direction, so the relationship between M and L_r becomes, in the general case, much more complicated. While this clearly has important ramifications for constraint derivation and may produce extra edges, e.g. highlights, it seems likely that any edge detector suitable for dealing with edges from scenes containing only Lambertian-surfaced objects will also do quite well for more general scenes, since constraint analysis will still consist of examining intensities in the neighbourhood of edges.

The first constraint was that the pixels investigated should be roughly perpendicular to the edge. This again means that our edge detector should find the edge, including its local direction, accurately, and from the edge representation it should be possible to compute the direction perpendicular to the edge.

Summarising the above, the edge detector should find and represent image edges as accurately as possible. From the edge representation it should be possible to tell which pixels are mixed and which are pure. It should also be possible to select pixels on

either side of the edge roughly perpendicular to the direction of the **edge.**

Although the discussion above has been couched in terms of reflectance edges, the constraints will also apply when finding and interpreting other types of image edges. Based on both the general and specific analyses of edge formation and detection, and the needs of constraint analysis, we can now write down a set of goals for edge detection upon which the effective design of algorithms may be based.

These are:

<u>General</u>

1. Find all the significant intensity variations - they are all caused by something in the scene.
2. Relate edges to their scene meaning, this is, after all, the stated goal of early processing and it might be possible to get some way toward this end in edge detection.

<u>Step-like Edges</u>

Step-like edges are important for constraint analysis so:

1. Find and represent step edges as accurately as possible.
2. Try to maintain a clear relationship between edges and their underlying image intensities. In particular, know which pixels are mixed and which are pure.

In the next chapter we will examine existing methods of edge detection to see if they satisfy these criteria.

3. A Survey of Edge Detection and Its Use in Edge Labelling

3.1 Introduction

The key step in the development of an edge detection system is the identification of the correspondence between the elements of interest in the scene and their characteristic appearance in the image. This is (at least conceptually) simple in the blocks world domain, where it means recognising the image intensity patterns produced by polyhedral surface boundaries. For general scenes, the situation, as discussed in chapter two, is much more complicated. In the blocks world one might naively expect all the edges to be of the same width, whereas in the general case, edges of a range of widths are produced. In the polyhedral blocks world, one can also expect the edges to be straight, with areas of relatively smooth gradient between them. For general scenes, no such simplifying assumptions may be made.

A further important difference between the desired performance of blocks world and general purpose edge detectors arises due to the different system structures. In blocks world vision the output of the edge detector is normally expected to be a set of lines suitable for use in modelling or model-matching. This leads to two points of interest:

1. Surface marks, shadows, etc. are not normally considered useful to the high-level system, in fact, they will confuse it, so they shouldn't be found by the edge detector. Alternatively, if shadow edges are detected and recognised as such before the edge data is passed on to higher levels of processing, they can be used or discarded as appropriate. Waltz (1975) found shadow boundaries to be

an effective constraint on line labelling, although his system used perfect line drawings rather than image edges as its input data.

2. In most blocks world scenes the number of desired edges is often small. Terms such as "data reduction" are sometimes listed as desirable properties of edge detectors.

In general purpose vision systems, edge detection must normally be followed by a process for extracting surface information. For example, in the case of stereopsis, surface markings, etc. are important for producing a less sparse array of disparities, so the edge detector should produce as many meaningful edges as possible to improve the system's performance. In general, all early processing systems which attempt to extract 3-d information from the image, produce better results from more detailed input data. In the case of edge labelling, the edge detector will not know whether particular image edges are important or not, so it should output as many edges as possible. Thus we see that edge detection for general purpose vision systems has a completely different set of requirements than that for blocks world vision. This point is worth stressing because the different requirements have not been explicitly recognised, which results in some confusion about what edge detectors should be doing.

In the remainder of this chapter the field of edge detection will be selectively surveyed. In section 3.2 ways of extracting edges of a single width from images are examined. In section 3.3 systems for finding edges of several widths, either using a family of different-sized edge detectors, or using several different-resolution versions of the image, are considered. This is followed by a discussion of methods of assigning a scene meaning to image edges.

Finally, in section 3.5, general conclusions of the survey are noted.

3.2 Finding Edges of a Limited Range of Widths

Interesting scene events such as surface boundaries normally give rise to large rates of change in image intensity, so the basic idea of edge detection is to search the image for high gradients. Two main techniques have been widely used. In the first the pixel values are assumed to be ideal samples of some smooth underlying function, and the presence or absence of an edge at a particular point is decided by examining the first or second derivatives of the function. In the second method, an ideal edge model is fitted to intensity values in the neighbourhood of a pixel. Parameters of the fitted model and a measure of the goodness of fit are used to determine the presence or absence of an edge.

Some idea of the advantages and disadvantages of the former method can be obtained from consideration of the ideal edges and their first and second derivatives shown in fig. 3.1. In 3.1(a) the ideal step produces a peak in the first derivative and a zero-crossing in the second, both in the same place as the edge. In the slightly more realistic rounded step edge shown in 3.1(b) the peak and zero-crossing are still in the correct position. However, if the first derivative peak is found, as is normally the case, by thresholding, several points will be above threshold. Non-maxima suppression or thinning are sometimes used to obtain a single response, although in many cases these will produce inaccurate results. The zero-crossing of the second derivative corresponds to the peak of the first derivative (and the point of inflection on the original edge), so it doesn't need thinning. The main problem with it is that other, non-edge, image features also produce zero second derivatives. For

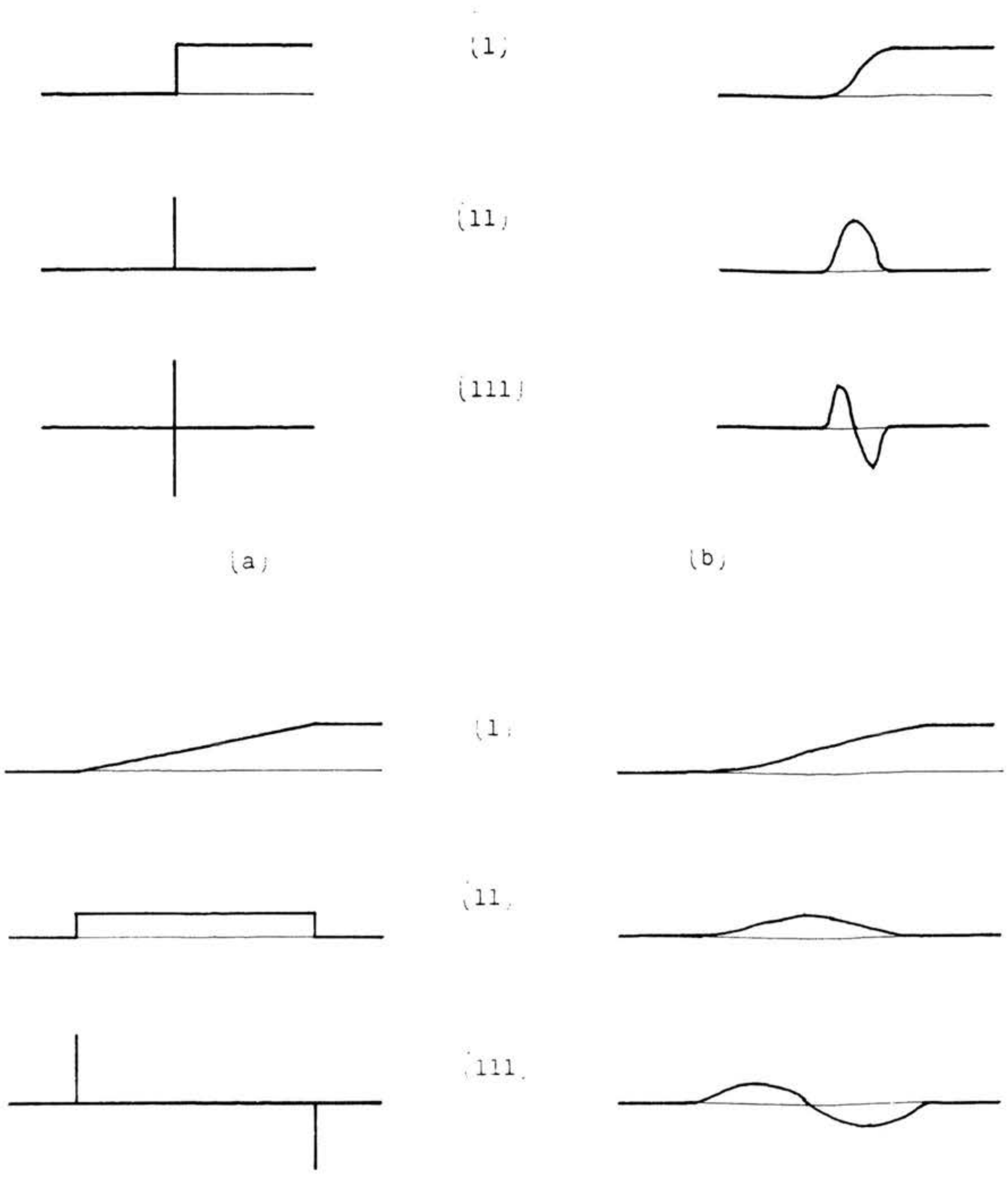


Fig. 3.1

This shows the first and second derivatives of four (one-dimensional) idealised edge models. In each case (i) is the original intensity, (ii) is the first derivative, and (iii) is the second derivative.

example, the uniform areas on either side of the edge do so. Any noise on the intensities will produce spurious zero-crossings. These problems are exacerbated with wider edges as shown in fig. 3.1(c) and (d). One solution to this is to threshold the slope of the zero-crossings. In terms of performance in the presence of noise, first difference operators are more reliable than second difference operators. To summarise: it is easier to achieve higher accuracy in localisation with second difference operators due to the broad response of first difference operators. However, detection reliability is harder to achieve for second difference operators due to their higher sensitivity to noise and the fact that they respond to image entities other than edges.

Roberts (1965), whose goals of edge detection were listed in section 2.2, based his edge detector on a simple gradient operator, as shown in fig. 3.2. He recognised that an important additional constraint was that edges are continuous in one direction, and used this to find feature points, i.e. image points where the gradient is above a threshold. Perhaps the most widely used operator in this class has been the Sobel (Duda & Hart 1973), shown in fig. 3.3. It is usually thought of as a set of two 3*3 masks applied to the neighbourhood of the point in question. The computationally expensive square root operation is normally replaced by taking the sum of the magnitudes of z_x and z_y, which is much quicker but less effective. It may appear slightly surprising that the Sobel operator has been so popular in view of the small size of the masks used and the current widespread interest in using very large neighbourhoods to combat noise. But, because of its small size, the Sobel operator tends to locate edges accurately in the sense that the maximum response to an intensity change will be within a pixel distance of the maximum

$y_{i,j}$	$y_{i,j+1}$
$y_{i+1,j}$	$y_{i+1,j+1}$

$$Z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2}$$

Fig. 3.2

Robert's cross operator. The Z values were thresholded and correlated with short line segments in four directions to decide on the presence or absence of edge points.

-1	0	+1
-2	0	+2
-1	0	+1

z_x

+1	+2	+1
0	0	0
-1	-2	-1

z_y

$$Z_{i,j} = \sqrt{z_x^2 + z_y^2}$$

$$\theta = \tan^{-1} \frac{z_y}{z_x}$$

Fig. 3.3

The Sobel operator. z_x and z_y are weighted sums of intensities in the neighbourhoods of the pixel under examination. $Z_{i,j}$ represents the edge magnitude at pixel (i,j) and θ gives the edge direction.

gradient in the underlying intensity function. However, because it is a thresholded first difference operator it will often respond over a wide band of pixels near the location of the maximum gradient, leading to a need for non-maxima suppression. A second reason for the popularity of the Sobel operator is that it is very easy to compute, requiring only additions and subtractions.

The second main class of edge detectors are based on using a set of ideal edge templates. These consist of matching a series of such templates (of ideal step edges over a range of orientations and step sizes) to the greyscale picture values in a local window, then thresholding some measure of the goodness of fit between the best-match template and the window to find the edge points. The set of templates may be represented by a truncated orthogonal series with parametrised coefficients. Truncation is useful for noise rejection and can also lead to an analytic solution to finding the best fit template. One of the first edge detectors of this type was introduced by Hueckel (1971) and was based on the Fourier series. This suffered from problems due to the mismatch between the 2-d series, which are polar in nature, and the rectangular digitisation grid.

O'Gorman (1978) designed a detector on the same principles as Hueckel's, but used the more computationally appropriate Walsh functions. The basic idea is that an ideal step edge template is represented by a truncated series of Walsh functions. To allow for edges of varying intensity, step size, and orientation, the template is parametrised and thus so are the Walsh function coefficients. The equivalent series of coefficients is computed for a small window of the picture function. The distance between the two sets of coefficients is then minimised to find the parameters describing the best-fit template. Finally, to decide whether or not an edge segment is present in the window, a measure of the closeness of fit between the window coefficients and the best-fit template coefficients is thresholded. Note that the Sobel, Hueckel, and O'Gorman operators simply mark edge points, and give some measure of edge magnitude and orientation. Line finding must still be carried out on their outputs.

A complete edge detection and line finding system based on template matching has been described by Nevatia and Babu (1980). Rather than use a parametrised template method, they prefer six 5*5 masks representing ideal step edges spaced 30° apart in orientation. The magnitude of the highest output and the direction of the mask producing it are noted for each pixel. Then, by thresholding and non-maxima suppression, edge points are located. Interestingly, they use a very low threshold, arguing that it is the job of the high-level system to select and use the edges as desired. Line linking is carried out by examining neighbours of each point for edges in the same direction, and using the information for tracking. Finally, each line is represented by a set of linear segments.

One reason for the existence of such a large body of work as that on edge detection may be the difficulty in evaluating edge detector performance. A figure of merit for edge detector performance has been suggested by Pratt (1978). It is defined as:

$$F = \frac{1}{\max[I_I, I_A]} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha\, d^2(i)}$$

where I_I and I_A are the number of ideal and actual edge points repectively. $d(i)$ is the offset, i.e. positional error, between the ith. edge point found and its true position, and α is a weighting factor which can be used to adjust the relative penalties for errors in position vs. errors in detection. To use this measure, an ideal narrow ramp edge is subject to Gaussian noise to provide the same known input to each edge detector.

The main problem with this method of evaluation, in common with the design procedure of some edge detectors, is the oversimplified edge model. Additionally, edge detector performance and utility is task dependent. For example, an edge detector which is optimal for use in a system for inspecting printed circuit boards may be of little use in a system for analysing chest x-rays. As we saw in chapter two, although step edges are important, they are by no means the only type of edge needing consideration. Even in the blocks world, several different types of edge profiles arise (Herskovits and Binford 1970, Horn 1977). Thus, template matching is not suitable for a general purpose vision system, where edges of many different types, widths, and spacings must be expected. Similarly, the result of applying the above test to an edge detection system measures only one aspect of required performance, and cannot be used as the sole measure of good design.

Edge detectors based on first or second derivatives do not suffer from the use of an oversimplified edge model in the same way as those based on template matching. However, they still must be able to deal with multiple edge widths and image texture to be considered general purpose. In the next section we examine edge detector systems designed to deal with these problems.

3.3 Representing Edges of a Range of Widths

In principle, detecting edges of a range of widths is simple. The image can be filtered in some way such that only edge components of a certain spatial frequency remain. These can then be detected.

The major problem is representing the output. Should all edges of all widths be stored? Should only the "best" edges in a particular image region be stored? How can the notion of "best" edge be formalised? How does texture relate to variable-width edge detection? These are some of the questions that must be answered if edge operators of several sizes are to be successfully applied to an image.

It has been recognised for some time that solving the issue of detecting edges of different widths and relating them in a coherent way is central if progress is to be made in early visual processing. Insight into this problem has been provided by Witkin (1983) with his "scale-space" filtering ideas. He solved the problem of matching edges at different widths by:

1. using a continuously-varying scale parameter
2. using a second derivative convolution function with particular properties which guarantees that certain constraints will hold on the zero-crossing contours at different scales.

His basic system, which is currently applicable to one-dimensional signals only, convolves the signal, $f(x)$, with (in theory) a continuum of Gaussian second derivatives with varying variance (σ^2). This produces a two-dimensional output since the convolution is a function of both the position on the signal and the variance of the Gaussian. This output is called a scale-space image. A line of constant σ in the image represents the convolution of the original signal with the second derivative of a Gaussian. A line of constant position (x) in the image represents the convolution output at that point as σ is varied. Witkin chose to use the Gaussian because as σ is decreased additional zero-crossings may appear but existing ones never disappear (although they may move in the x direction). As a result, if a "line drawing" of the scale-space image is formed from the zero-crossing contours, it consists of a set of arches (assuming the horizontal variable is x and σ decreases from top to bottom).

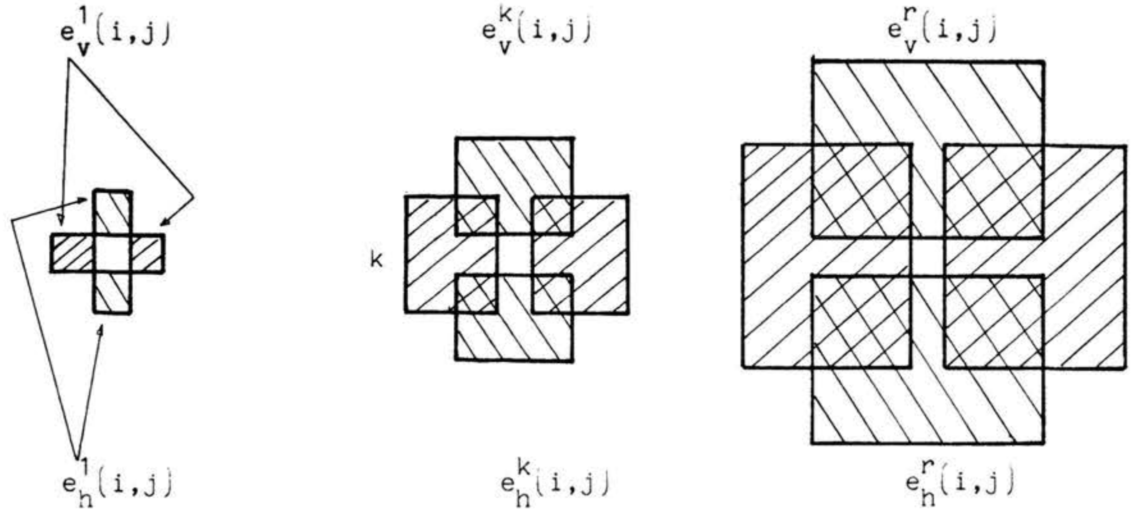
Witkin introduced two assumptions on which he based his claims for the utility of the representation.

1. Zero-crossings on the same zero-crossing contour (across variations in σ) arise from a single underlying event.
2. Tne localisation accuracy of the zero-crossing contours increases as σ decreases.

The first of these is clearly important in any system attempting to label image entities with their scene meaning since it provides a way of grouping the edges produced by a single scene phenomenon at different scales. Taken together the two assumptions mean that the more reliable (in terms of noise performance) but less accurate zero-crossings found with large σ values can be easily related to the corresponding (and more accurate) zero-crossings produced with small σ values, thus holding out the prospect of an edge detector which is both robust and accurate. It is interesting to note that some of the zero-crossing contours illustrated in Witkins scale-space images exhibit considerable variation in the x-direction. This shows why attempts to use only a few mask sizes (corresponding to a few values of σ in Witkin's system) have run into difficulties in matching edges found with different masks.

Finally, Witkin investigated the use of a ternary tree representation of the scale-space image and produced criteria for pruning the tree to effectively filter the image to produce versions of the original signal which emphasised those features which were found to be particularly noticeable to human observers.

An earlier attempt to investigate these issues was undertaken by Rosenfeld and his collaborators (Rosenfeld & Thurston 1971; Rosenfeld, Thurston & Lee 1971) as detailed in fig. 3.4. At each point in an image the average intensities of a pair of neighbourhoods on each side of the point were differenced to give an edge value. By varying the size and position of the neighbourhoods, edges of different orientation and size could be found. The idea of the "best" mask size was introduced and defined to be the size of the largest operator whose output was significantly greater than that of larger masks at the same point, but not significantly smaller than the next smallest mask. The idea of this being to use the largest mask which doesn't suffer from interference from other edges, hence reducing the effect of noise as much as possible. Non-maxima suppression was used for edge thinning, taking into account the best mask size and orientation at each point. The output of the system was a single array of the same dimensions as the original picture, containing a measure of the edge magnitude at each point. It was not possible to estimate edge width in this system because the neighbourhood size chosen as "best" for each edge depended on the image distance to adjacent interfering edges, not on the parameters of the edge in question. The system was also applied to textured images using the output of a Roberts cross operator applied to the original image as input. The idea was that pixel values then represented the "edgeness" of the local neighbourhood rather than its intensity. However, no attempt was made to integrate the output of the latter system with that of the normal edge detector.



1*1 neighbourhoods k*k neighbourhoods r*r neighbourhoods

1. Take horizontal and vertical differences of neighbourhoods at a range of sizes (e_h represents vertical difference magnitude, e_v represents horizontal difference magnitude).
2. Pick best orientation at each size, given by

$$\text{direction of } \max(e_h^k, e_v^k)$$

3. Pick best size, which is the largest k such that

$$e^k > e^{k+1} > \dots > e^r$$

and

$$e^K \not< e^{k-1}$$

4. Non-maxima suppression, set e^k to zero if there is a larger value within k/2 in the direction perpendicular to the best orientation.

Fig. 3.4

This summarises the algorithm of Rosenfeld, Thurston & Lee (1971) for variable-width edge detection. In their system four orientations were used: horizontal, vertical, and the two diagonals.

Some experiments were also carried out using non-overlapping neighbourhoods of a range of sizes in a top-down way. The edge points found using large neighbourhoods were used to decide where to look for edges of smaller neighbourhoods. However, no attempt was made to integrate the output of different-sized neighbourhoods.

This idea of using the edges in a reduced resolution version of the image to guide the search for edges at higher resolution was applied to finding human profiles in images by Kelly (1971), and later generalised by Tanimoto & Pavlidis (1975) and Tanimoto (1976). The object of this work was not to find edges of different widths but to increase the computational efficiency and robustness of the vision process. Starting with the original image, a reduced resolution version is produced by averaging the intensities over a 2*2 neighbourhood to produce the new intensity value. This process may be carried out until as small an image as desired is obtained, as shown in fig. 3.5. The complete data structure thus formed is normally referred to as a pyramid.

Although these experiments had been carried out with multiple-size edge detectors, the goal of the systems concerned was mainly to achieve improvements in speed with some added noise rejection. It had not been explicitly stated that many images contain edges of a wide range of widths, and that representing these edges in a coherent way was an important problem. The first person to address these issues in some detail was Marr, whose work in this area culminated in a comprehensive theory of edge detection (Marr & Hildreth 1980). Generating the final edge representation took place in three steps (in practice the first two are combined):

1. Pass the image through several smoothing filters centred at dif-

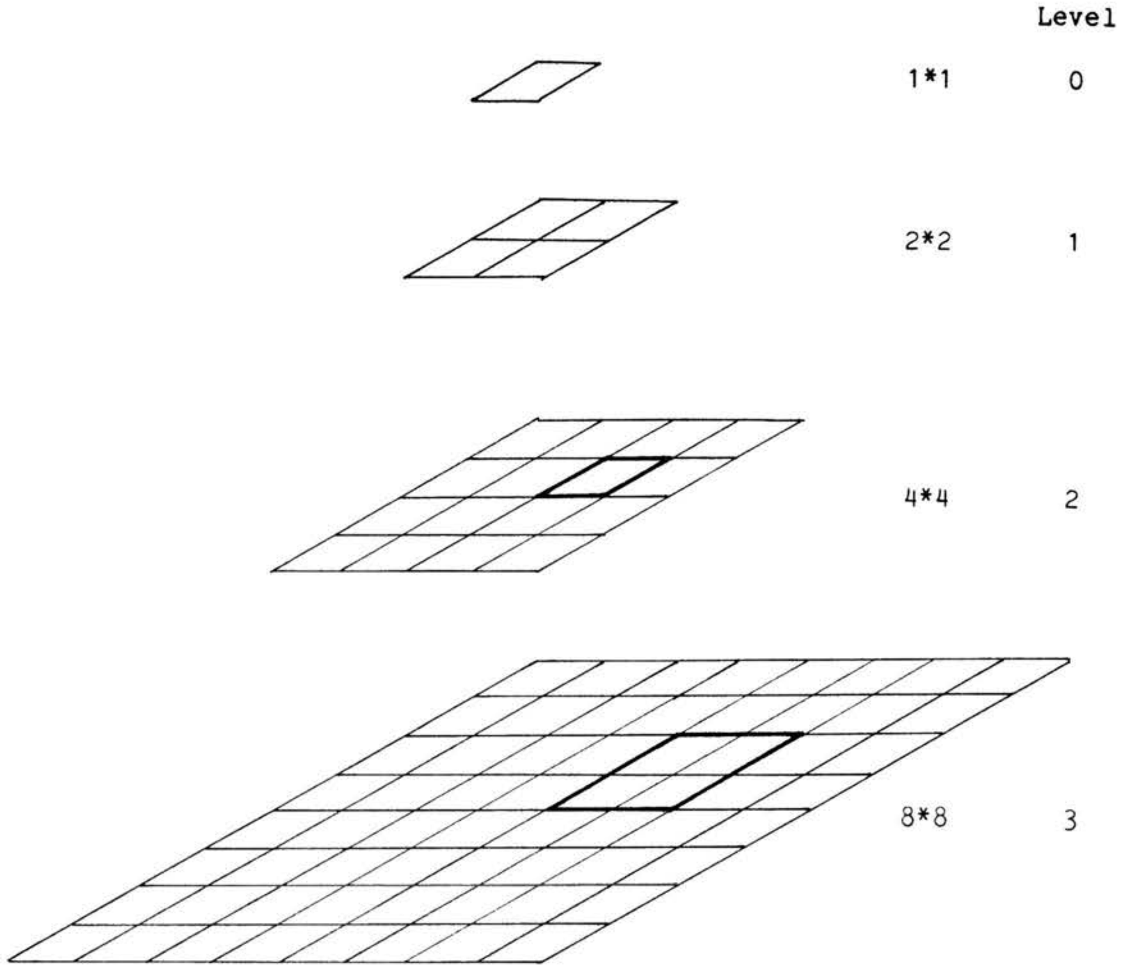


Fig. 3.5

The pyramid image data structure. Starting with the original image, an averaging operation is used to produce a reduced resolution image. Most commonly, 2*2 neighbourhoods at level n are averaged to produce a single pixel intensity at level n-1, as shown above. Various properties of these data structures have been discussed by Tanimoto (1976).

ferent spatial frequencies.

2. Find the edges in each filtered image.
3. Consider the edge evidence from each channel to produce the final representation.

This is illustrated in fig. 3.6 below:

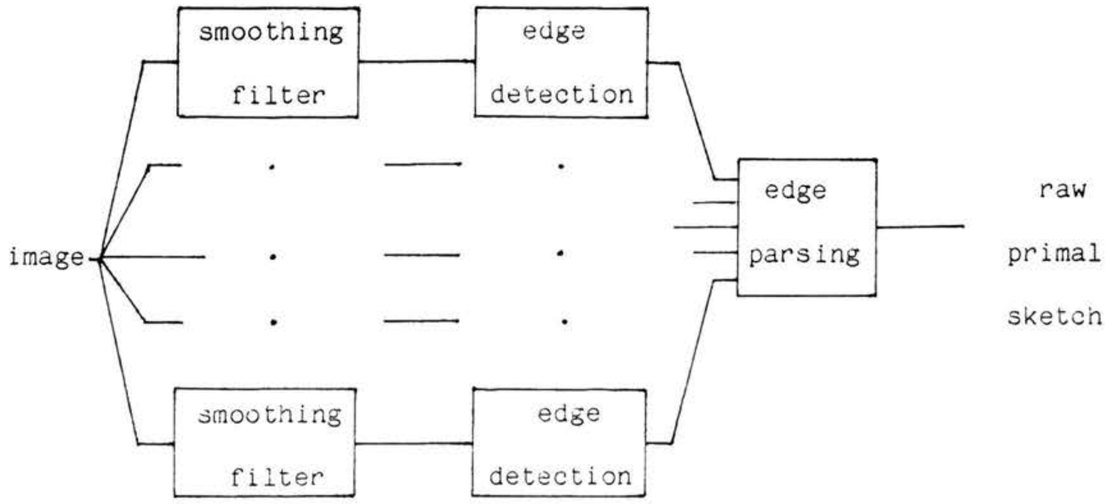


Fig. 3.6

The overall structure of the Marr/Hildreth system.

Their choice of smoothing filter was based on two constraints. First, since the motivation for filtering was to reduce the range of intensity changes, the filter should have an approximately band-pass response. Second, they claim that since the entities in the scene which give rise to intensity changes, such as obscuring boundaries, are spatially localised then the filter also needs to be localised in the spatial domain. Marr and Hildreth call this the "constraint of spatial localisation". These two requirements are conflicting, but the frequency response which best satisfies them has been shown to be a Gaussian curve.

It is interesting to note that it is not sufficient for the filter simply to extract edges within a certain narrow frequency band, removing all others. An ideal bandpass filter produces echoes of strong edges because of its extent in the frequency domain. Considering a single line of the image as a 1-d signal, and taking this idea to the limit, if we could extract the (1-d) component of the line at a single frequency we would get a constant amplitude sine wave, which would not be very informative about the position of edges. It seems that it may be more fruitful to examine desirable properties of the smoothing filter in the spatial domain rather than the frequency domain. This explains Marr and Hildreth's need for their constraint of spatial localisation.

To detect edges Marr and Hildreth use the second derivative. In particular, they chose to use an orientation-independent second derivative operator for computational efficiency, namely the Laplacian. In other words, their theory suggests that the image is first convolved with a 2-d Gaussian filter, then the Laplacian is applied. In fact, these two steps can be combined so that the image can be convolved with the Laplacian of a Gaussian, $\nabla^2 G$. The edges being the zero crossings of $\nabla^2 G * I$, where * is the convolution operator. To find the underlying edge direction, the direction of the contour of the zero crossings may be used provided certain restrictions hold on the variation of intensity in the vicinity of the edge. The zero crossings are represented by short linear segments together with the slope of the convolution output, taken normal to the zero crossing direction.

Finally, the zero crossings of several channels are combined using a set of parsing rules, based on what Marr and Hildreth call their "spatial coincidence assumption", which is:

If a zero-crossing segment is present in a set of independent

$\nabla^2 G$ channels over a contiguous range of sizes and the segment has the same position and orientation in each channel, then the set of such zero-crossing segments may be taken to indicate the presence of an intensity change in the image that is due to a single physical phenomenon (a change in reflectance, illumination, depth or surface orientation).

This assumption is particularly important for the Marr/Hildreth system because it provides the key link between scene/imaging events and image edges on which the integration of the outputs of several channels are based. In the theory, Marr and Hildreth provide parsing rules for dealing with isolated edges, bars, and blobs. For each of these three classes a set of properties, {position, orientation, contrast, length, width} is calculated. This is easy for an isolated edge using the smallest channel to which the edge appears to be a step (as opposed to a ramp). For bars, small channels must also be used to avoid interference. Blobs are small closed contours whose properties are computed from fitted rectangles.

The final representation, the "raw primal sketch", thus consists of a large number of lists, each containing the parameters listed above for an isolated edge, bar, or blob.

By considering the Marr/Hildreth theory in the light of the discussions in chapter two, we can see that it made several contributions in two areas:

1. The explicit realisation:
   a) that the input to an edge detector needs to be well defined,
   b) that edges in images of natural scenes occur in a range of widths.
2. The outline of an appropriate overall structure, showing that the important problems are:
   a) the design of smoothing filters,
   b) designing systems to find the significant edges of a

particular width,

c) finding a way of combining the channel outputs into a useful representation.

Note that in a particular system it may be appropriate to combine 2(a) and (b) by designing a family of related masks which each detect edges of a certain width, rather than by prefiltering the image.

The detailed aspects of the Marr/Hildreth theory need to be improved. Haralick (1984) has shown that the single channel Marr/Hildreth edge detector performs significantly worse than other detectors of comparable size. The parsing rules for multi-channel integration were also unsatisfactory. Typical images don't just contain isolated edges, (isolated) bars, and blobs. They also contain corners, texture, highly non-linear, but nevertheless significant, intensity variations, etc. The Marr/Hildreth system was not capable of dealing with these. Hildreth (1980) suggests that more must be known about the requirements of the succeeding processes before the outputs of different channels can be combined successfully into a useful representation.

Another edge detection system with a similar overall structure has been devised by Canny (1983). Derivation of the edge detector masks is undertaken by specifying a set of three mathematically-formulated performance criteria and finding the optimum mask. The three criteria are:

1. There should be a low probability of error.
2. The edge points should be found as accurately as possible.
3. There should only be one response to a single edge.

When all three criteria are used the best operator turns out to be closely approximated by the derivative of a Gaussian. In fact, the result of applying the criteria is a family of operators of the same shape but different size, depending on the desired error rate and accuracy of localisation. Hence, Canny's method is useful for generating a series of masks of different sizes. Notice that the three criteria do not actually contain a definition of an edge. For instance, the first criterion specifies that there should be a low probability of marking non-edges and of failing to mark true edges without specifying what an edge is. In fact, the criteria were applied to an ideal step edge, so the resulting operator is optimal only for that input. Canny also developed optimal detectors for ridge and bar edge types, however, he found difficulty in integrating the ridge detector output with the step-edge detector output and did not attempt to integrate the output of the bar detector. He suggests that this problem of integrating different edge type representations into a single description of the intensity changes in the image is one which still needs a lot of work.

To integrate the edges found at different scales, Canny uses what he calls a "feature synthesis" approach. The smallest operator being used is first applied to the image. Then, the edges produced are used to synthesise the output of a larger mask which would result if these were the only edges in the image. These edges are compared with the actual edges generated by the larger mask. Any additional edges in the actual output are added to those produced by the smaller mask to produce a complete set of the edges found by the two masks. By concentrating mainly on step edges the multi-scale integration is simplified. Additionally, the third performance criterion guarantees that only those edges which are a significant distance apart are found. This effectively removes the problem of microtexture, i.e. image areas producing many edges close together - these are filtered out in Canny's system.

Interestingly, in the light of the design presented later in this thesis, if only the first two performance criteria are used, a difference of boxes operator performs best. In fact, it appears to give maximum signal to noise ratio with arbitrarily good localisation. However, because it has a high bandwidth it tends to produce other peaks near the true one generated by the edge. This is not in conflict with the claim that it maximises the signal to noise ratio, since that only applies at the step. By adding his third criterion, Canny effectively adds a smoothing requirement. Thus the trade off is localisation accuracy and high bandwidth versus possible multiple responses to a single edge. Canny chose to eliminate the multiple response problem by adding his third criterion.

A potential problem with all edge detection systems is aliasing. If the analogue image focussed on the target contains frequency components greater than twice the sampling frequency then aliasing will occur, the high frequency components being reflected down about the sampling frequency. Because it involves some low-pass filtering, Canny's system reduces the effect of aliasing both in the original image and in applying several operators at different scales. The problem is more severe with difference of boxes operators and is discussed in detail in chapter six.

Canny's operator for step edges is similar to the one used by Witkin (1983) in his work on scale-space filtering. In terms of the scale-space image (in the 1-d signal case) Canny's system involves using a finite set of masks rather than a continuum as Witkin did. This means that the multi-scale integration is more difficult, since a correspondence problem is introduced, i.e. which edges at one scale should be paired with which edges at an adjacent scale? Recall that in Witkin's system additional edges (as represented by zero-crossings in the second derivative of a Gaussian convolution) appear as the mask size is reduced but that existing edges cannot disappear. Thus, we might expect the edge set found by the smallest of Canny's masks (as represented by peaks in his approximately first derivative of a Gaussian convolution) to contain the complete set of edges for the image. If this is so then why bother with the larger masks? One reason is that the convolution output in Canny's system is thresholded, so that only peaks greater than a certain height are retained. Faint edges may not be strong enough, relative to the noise, to be detected by the small masks, but may be found with larger masks which can extract edges with a smaller signal to noise ratio. Canny provides a partial solution to this problem by using an adaptive thresholding technique. If any part of an edge contour is above a high threshold than all connected parts of the same contour above a second, lower threshold are also marked.

It is not clear whether Canny's system notes the width of each edge in the final representation. It is also not clear how it deals with complex cases, such as the situation where a sharp intensity change, e.g. a reflectance edge, is superimposed on an extended change, e.g. a blurred shadow. However, Canny's system is probably the most successful to date.

3.4 The Use of Edge Detectors in Edge Labelling

Since an image is a 2-d projection of a 3-d scene, there is insufficient data in a single image to reconstruct the scene which produced it. Even with two or more images, it is not possible to perform a unique reconstruction unless assumptions are made about the nature of the scene, i.e. the range of possible scenes is constrained in some way. In this thesis we are concerned with designing an edge detector for intensity-based edge classification, which is one particular type of constraint-based early processing. The goal of edge labelling is to classify image edges as particular types of scene boundaries, and possibly also to gain some idea of surface shape. For a simplified domain this was shown to be theoretically possible by Barrow & Tenenbaum (1978). The overall strategy has been outlined in a more practical context by Fischler et al (1982).

An early attempt by Horn (1974) to determine true reflectances from image intensities was later generalised by Ullman (1976) to include light source detection. However, these methods only applied to "achromatic Mondrians", which are flat, evenly lit surfaces of a collection of rectangles of different reflectances (but not colours). There was no need for edge detection since the method of processing could be applied along scan lines of the image and be guaranteed to cross edges normally. It is interesting to note that Ullman's "S" operator basically compared intensities and intensity gradients on either side of the edge. The same operator was later applied by Forbus (1977) to the task of identifying highlights in more general scenes, but in this case it was applied along straight lines manually selected from the image.

Witkin (1982) has presented a method for finding edges in images generated by obscuring and shadow boundaries. The basic idea is that because the surfaces on either side of an obscuring boundary are separated in space, image curves on either side of the resulting edge should correlate badly. On the other hand, curves on either side of a shadow edge should correlate well (under a linear intensity transformation to account for the shadow), because they probably lie on the same surface. Clearly, finding edges accurately is important in getting the correct correlations. Although he mentions using an edge detector, Witkin used edges traced by hand for the examples given in his paper, which seems to imply that the edge detector was not entirely satisfactory.

The spatial arrangement of edges can also help to decide their type, as has been shown by Binford (1981). The ancestry of this work can be traced to early blocks world edge-labelling programs. An example is shown in fig. 3.7, where, if the edges labelled "s" have previously been identified as shadow edges, then the other edges can be identified as obscuring, surface orientation, and reflectance edges respectively.

3.5 Conclusion

In view of the significant body of work noted above, deriving scene boundary interpretations from image edges is an important goal of early processing. To achieve this goal, edge detection must provide a suitable input to the edge labelling process, but edge detectors do not appear to be satisfactory in this respect as yet. From the complexity of image edges and the results of previous work in this area, the appropriate overall structure suitable for such a system seems clear. It should be multi-channel; significant intensity changes must be found separately in each channel, then integrated into a single coherent representation. In the next chapter, we begin the synthesis of a system based on this structure designed to satisfy the goals of edge detection listed at the conclusion of chapter two.

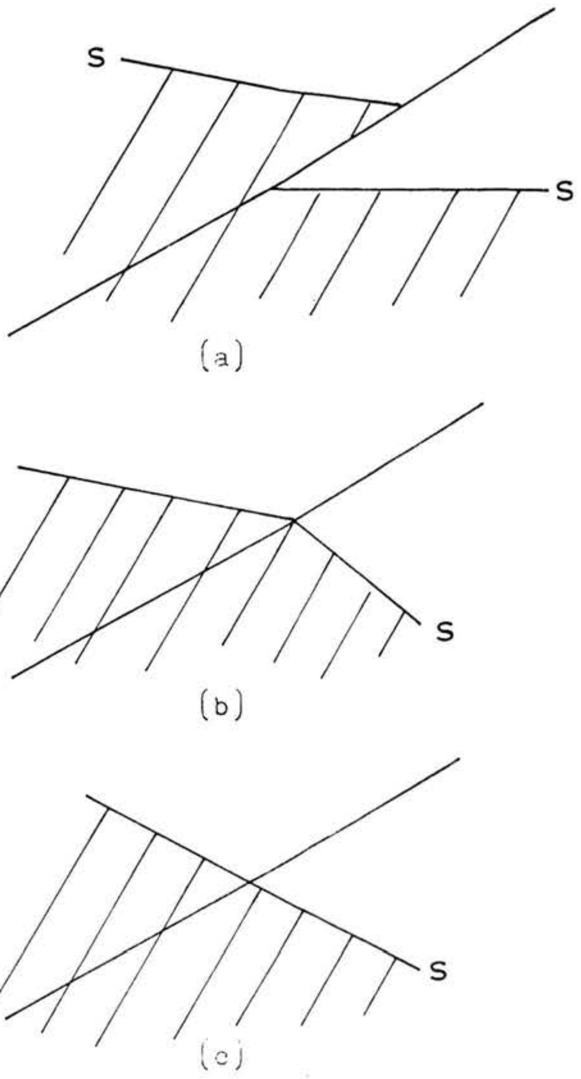


Fig. 3.7

If the edges labelled "s" in each of the above have previously been identified as shadow edges, then the other edges are most likely to be:

(a) obscuring edge

(b) surface orientation edge

(c) reflectance edge

(After Binford (1981)).

4. The Initial Image Representation

4.1 Implications of the General Goals

In the preceding two chapters, suitable goals for edge detection for constraint analysis have been derived, based on analysing the needs of the succeeding process, and an appropriate structure for the overall system has been chosen, based on previous work in the field. The need for an improved edge detection system to enable constraint analysis to be automated was also demonstrated. In this chapter, the synthesis of a system which is specifically designed to satisfy the goals of edge detection for edge labelling, which are restated below, is begun.

Goals of Edge Detection

General

1. Find all the significant intensity variations - they are all caused by something in the scene.
2. Relate significant intensity variations to their scene meaning. Although this is really the job of the constraint analysis system, it may be possible to achieve some preliminary semantic labelling during edge detection.

Step-like Edges

3. Find and represent step edges as accurately as possible.
4. Try to maintain a clear relationship between an edge description and its underlying image intensities. In particular, know which pixels are mixed and which are pure.

The greyscale value of a pixel in a digitised image of a typical scene depends on many factors. Some of these, such as the relationship between image intensity and greyscale value, can be assumed to be constant and known. Others, such as the angle between elements of surface and the imaging device, are not constant and are unknown. The object of constraint analysis is to solve for some of these unknowns.

Due to the complexity of the scene/image relationship and the reduction in dimensionality, it is always necessary to make assumptions about the scene and the scene-to-image transformation. These assumptions are manifested in a search for expected relationships among pixel intensities. Image analysis can be considered to consist of searching for particular intensity patterns and deducing scene properties from their parameters and relationships. In all but the simplest systems this complete process takes place over several stages. Each stage can be thought of as a process-representation pair. Given the image as input, some process is used to transform the image to produce an initial image representation. This first level representation forms the input to another process which transforms it into a second-level representation, and so on. To begin with, it is often useful to think only of the representations, the appropriate processes can be designed later, once their specifications, i.e. input representation, output representation, and the relationship between them, have been determined. Of course, representations have to be chosen for which it is possible to find processes able to generate them, so compromise is usually necessary, and the overall system design may require several iterations.

The role of a representation is to explicate certain properties of the data set concerned, usually at the expense of suppressing others if the size of the representation is to be kept within bounds.

Typical early processing systems use two representations. The first makes explicit certain pixel positions and properties in the image which are in some way interesting. The second usually explicates certain relationships among entities in the first representation, for example, representing certain sets of data points which satisfy some geometric property, such as connectedness, as a single entity.

The choice of system architecture described in chapter two and the resulting goals of edge detection rested on the veridicality of the image. In other words, pixel intensities and relationships among them are a faithful representation of the scene concerned under projection. Since the image is veridical, and only makes sense as a 2-d projection of a 3-d scene, the best way to analyse it is to use the veridicality and attempt to label image entities with their scene meaning, even at the early processing stage. This principle and the goals of edge detection should direct our choice of initial image representation. However, since relatively few simplifying assumptions were made in deriving the goals (for example, the occurrence of texture has not been ruled out), it is to be expected that achieving them will be a complex process using several intermediate representations. At this stage we must try to decide what is appropriate for the initial representation of the image.

The first goal requires that all significant intensity variations should be found. This implies that every possible intensity change location should be examined and the magnitude of the change should be classified, at least as significant or not. In a digitised image, every boundary segment between two adjacent pixels is such a location, where the difference in pixel intensities gives the magnitude of the change.

Alternatively, some measure of the intensity gradient at each pixel could be found using a neighbourhood centred on the pixel. This approach suffers from the disadvantage that it requires certain assumptions to be made about permissable intensity patterns, for example by fitting a plane to the intensities in the neighbourhood. Since the first goal of edge detection requires the identification of all intensity variations, this approach will not be used here.

Further support for the simple representation suggested above comes from the third and fourth goals of edge detection, which require step edges to be found as accurately as possible while maintaining a simple relationship between the edge representation and pixel intensities. Most representations based on local neighbourhoods effectively low-pass filter the image, resulting in a distortion of phenomena with significant high-frequency components, such as step edges. Consequently, the accuracy of edge localisation is reduced (Witkin 1983). It has been shown, for certain types of edge detector, that there is an uncertainty principle relating sensitivity to localisation (Canny 1983). In other words, the better an edge detector performs at extracting edges from noise, the worse will be its localisation ability and vice versa.

Since the ultimate goal of the edge detection system described here is to provide a suitable input for constraint analysis, some further light may be shed on what is a "significant" intensity variation by considering the semantics of intensity changes. As we saw in chapter two, intensity changes occur over a wide range of scales in the image. Parts of the image where the intensity is changing quickly are particularly interesting because they normally correspond to interesting events in the scene. So we begin by dividing the range of intensity changes into three classes: uniform, fuzzy, and step, basing the classification on scene/image semantics.

A region of the image containing only small intensity changes - known as a uniform region - normally corresponds to a single surface in the scene whose shape, incident light, and reflectance properties are changing slowly.

An image region containing intensity changes of an intermediate magnitude - a fuzzy region - could, for example, be caused by a shadow from an extended source, indirect illumination, a change in surface orientation, or an extended change in surface reflectance properties.

A step region in the image corresponds to what would be an ideal step discontinuity in intensity in a perfect imaging system. These are most commonly caused by obscuring boundaries where one object passes in front of another. Step-like edges may also be caused by shadows from point sources, sharp changes in surface reflectance properties, or parts of a surface whose orientation is changing very quickly.

In signal processing terms this classification is relatively arbitrary, but it is justifiable on a semantic basis since one of our general goals of edge detection is to relate image intensity variations to their scene counterparts. For any particular image understanding system the boundaries between the classes will depend both on the overall goals of the system and on the detailed performance of its image acquisition hardware.

The spatial resolution of the imaging device is particuarly crucial, since it limits, via the sampling theorem, the highest spatial frequency of change the system will "see". For any practical imaging device a step edge will be blurred to some extent, resulting in it not being possible to distinguish step edges from fuzzy regions simply by thresholding. This is not only due to the sampling frequency being too low, but is also caused by the non point-like nature of the sampling function. That is, the receptive field of each pixel covers a finite area and has a particular spatial response. Taking this into account, the formal definition of a step edge given later does not depend only on the magnitude of the intensity step. This difficulty in distinguishing steps and fuzzy regions further justifies choosing an initial representation which separates intensity changes into one of just two classes; significant or not. Later, significant intensity changes can be subdivided into fuzzy regions or steps by considering the spatial distribution of the intensity change.

The simplest way of carrying out the above suggestions, i.e. flag each boundary segment between adjacent pixels if the corresponding intensity change is significant, is to use first differences, followed by thresholding the magnitude of the difference.

To summarise this section:

1. We want to identify every "significant" intensity change in the image.
2. Ideally, we would like to separate intensity changes into three classes: uniform, fuzzy, step.
3. (2) is not possible only on the basis of intensity change, so we begin by labelling intensity changes as significant or not.
4. To avoid making assumptions about the underlying intensity function, we should use the simplest method possible.
5. In practice, the first processing step is first differencing

followed by thresholding.

4.2 Thresholding in Noise

The conclusion of the preceding section was that the initial image representation should contain a flag for every boundary segment between adjacent pixels where the value of the flag indicates whether or not the corresponding pixel boundary is significant. If we had access to a noiseless digitised image we might begin by labelling the boundary between pixels P_1 and P_2 with intensities I_1 and I_2 as "primary" (significant) if

$$|I_1-I_2|>1$$

In other words any difference in the greyscale values of the two pixels would indicate the presence of a significant boundary.

However, we may only be interested in intensity changes which are greater than a certain magnitude, in which case we could increase the significance threshold to some larger value. Given a particular imaging device it would be relatively easy to work out the limit this puts on variations in image irradiance. If desired, one could then compute, for example, how much a particular surface, under known scene layout and illumination conditions, would have to be changing in shape to surpass the threshold. Alternatively, one could turn this computation around and decide what was significant in scene terms, then calculate the resulting minimum variation in image intensity for use as the threshold setting.

This approach is complicated by the presence of noise, which means that the value and method of application of any threshold chosen for the kind of reasons given above may have to be modified. All image acquisition systems suffer from noise, which arises both in the imaging device and the associated electronics. The first problem is to characterise the noise. If a solid state camera is being used, preprocessing may be necessary to correct for the variable response of the individual detectors, which can be large relative to the random noise (Binford 1981).

When using a vidicon it's normally assumed that adjacent pixels have similar responses, although there may be significant variation over the whole target area. For random noise we follow the conventional and reasonable assumption that the noise probability density function is Gaussian or normal. The parameters of the distribution can be found by several methods. One, used by Herskovits and Binford (1970), is to capture an image of a uniform scene and average pixel intensities over local neighbourhoods, comparing the average with the pixel intensity in the centre of the neighbourhood to find the variance. A more accurate, but also more computationally expensive, method is to capture several images of the same scene and use corresponding pixels in each image to calculate the noise parameters. One problem with this method is the potential existence of periodicity in the scene lighting.

The first step in the edge detection system described here is first differencing, followed by thresholding the magnitude to obtain the significant intensity changes. Since the image intensities are affected by noise, so are the first differences. This makes the choice of threshold important. If it is too low, many spurious edges will be generated by noise. On the other hand, if it is too high, the sensitivity of the system will be needlessly reduced. Different images have different edge distributions, so the selection of a best threshold may be an image-dependent task. Ideally, it is desirable to automatically find the threshold for any given image.

To find an automatic threshold selection mechanism, we must be able to measure the utility of the output as the threshold is varied. The best threshold can then be found. We describe the effect of noise on intensities using an information-theoretic communication channel model (Gallager 1968). This provides access to a body of theory which is ideally suited to describing the problem noted above.

The image itself and estimates of the noise statistics are available. To fully specify the noise we must know both its distribution of intensity values and the spatial layout of those values. In this thesis, we assume that the noise statistics do not vary across the image plane. We also assume that the noise on each pixel is stationary and independent.

Assume, for the moment, that the ideal distribution of first differences in the image concerned is also available, i.e. the noise-free distribution of first differences. If this distribution is I, we have

$$\sum_{i=-imax}^{imax} I(i)=1$$

where imax is the maximum first difference amplitude. If the first differencing threshold is denoted by s, the probability of no-boundary at a particular location in the noise-free image is given by

$$P(0)=\sum_{i=-s}^{s} I(i)$$

and the probability of a boundary at a particular location is

$$P(1)=\sum_{i=-imax}^{-s-1} I(i) + \sum_{i=s+1}^{imax} I(i)$$

These are the a priori probabilities of no-boundary and boundary respectively.

We next estimate the effect of noise on the ideal first difference distribution to generate the corresponding noisy distribution, from which we can find the probabilities of no-boundary and boundary in the noisy image. Since the noise distribution on pixel intensities is known, the corresponding distribution on first differences is easily found. The effect of noise on any particular element in the ideal distribution can then be computed. This can be represented by a set of conditional probabilities of altering the first difference amplitude, and can be illustrated graphically in a transition diagram, a simple example of which is shown in fig.4.1. Each transition is labelled with a conditional probability, namely, the probability of that transition occurring given that an edge of the magnitude indicated on the left has already occurred. The set of transitions is represented by $N(j)$:

$$\sum_{j=-jmax}^{jmax} N(j)=1$$

To obtain the complete noisy distribution we apply the set of noise transitions to every element in the noise-free first difference distribution and sum over all impinging noise transitions for each output element. In other words, we convolve the noise distribution with the noise-free distribution of first differences.

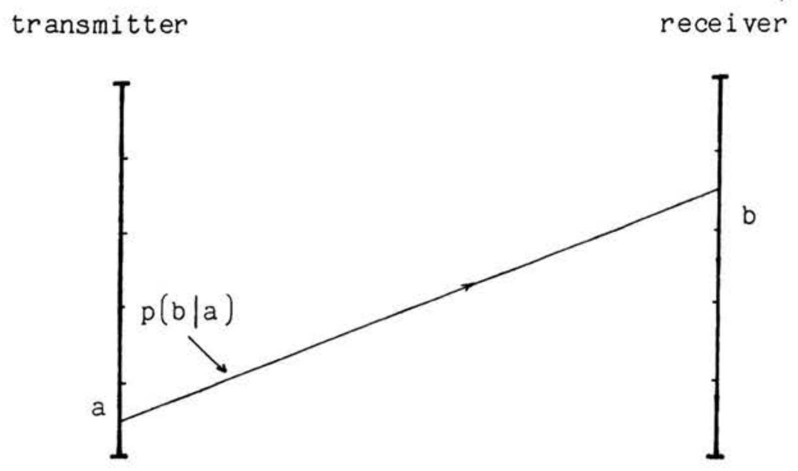


Fig. 4.1

Each line in a transition diagram indicates a conditional probability of the input symbol at the left of the line appearing as the output symbol at the right end of the line.

Call the output (noisy) first difference distribution O:

$$\sum_{k=-imax}^{imax} O(.)=1$$

i.e. the range of the output signal is limited to that of the input signal. In practice this has no effect, since edges of such large amplitudes don't usually occur. In terms of the input and noise distributions, O is given by:

$$O(k)= \sum_{i=-imax}^{imax} \sum_{j=-jmax}^{jmax} N(j)*I(i) \qquad : k=i+j$$

and the output probabilities of no-boundary and boundary are given by

$$Q(0)= \sum_{k=-s}^{s} O(k)$$

$$Q(1) = \sum_{k=-imax}^{-s-1} O(k) + \sum_{k=s+1}^{imax} O(k)$$

Having calculated the a priori and a posteriori probabilities of no-boundary and boundary, we can begin looking for a way of measuring the utility of a particular threshold choice. One appropriate set of measures is provided by information theory. Although information theory has been applied to threshold selection before [Pun (1980), Johannsen & Bille (1982)] the effect of noise has not previously been included.

For instance, we might initially consider maximising the information content (entropy) of the output (noisy) distribution. The appropriate formula for the entropy is

$$H = -Q(0)\log_2 Q(0) - Q(1)\log_2 Q(1) \quad \text{bits/boundary}$$

This is a maximum when $Q(0)=Q(1)=0.5$, i.e. it would always recommend a threshold which produced an equal number of boundaries and no-boundaries. This is undesirable because in typical images there are many fewer real boundaries than no-boundaries, so equalising their probabilities will generate many spurious boundaries due to noise.

What is really required is a measure of the similarity between the noise-free and noisy edge maps. In other words, given an edge map for the noise-free image resulting from a particular threshold choice, we want to know how much the edge map produced by using the corresponding threshold on the noisy image will be corrupted. If we think of the noise generation process as a communication channel, with the noise-free distribution representing the transmitter alphabet probability distribution, the effect of noise on the edges representing the effect of errors in transmission, and the noisy distribution representing the receiver alphabet distribution, what we require is a measure of the information transmitted over the channel. With conventional communication channels, the channel behaviour is fixed by its physical properties, and maximising channel capacity becomes a process of finding the transmitter alphabet distribution which best matches the channel (Gallager 1968). In our case the situation is different because both the transmitter probabilities and channel transition probabilities depend on the threshold choice. Basically, we want to find the value of threshold which gives the transmitter and transition probabilities which maximise the information transmitted over the channel.

In conventional information theory, the measure of the amount of information transmitted over a noisy channel is given by the mutual information of the source and receiver. This can be justified as follows. First, consider a noiseless channel as shown in fig. 4.2(a). The entropy of the source, $H(S_1)$, is given by

$$H(S_1) = -\sum_{k=1}^{K} P(a_k) \log_2 P(a_k) \quad \text{bits/symbol}$$

for a source alphabet $[a_1 \ldots a_k \ldots a_K]$. Since the channel is noiseless, no errors can occur, so the entropy of the receiver, $H(S_2)$, will be the same as that of the tranmitter, i.e.

$$H(S_1) = H(S_2)$$

Also, in the noiseless channel, the conditional probabilities of the transmitter symbols given the receiver symbols will be

$$p(a_k | b_j) = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

and so the conditional entropy of the transmitter given the receiver, given by

$$H(S_1|S_2)=\sum_{k=1}^{K}\sum_{j=1}^{K} p(a_k \text{and } b_j) \log\frac{1}{p(a_k|b_j)}$$

will be zero. This follows from the fact that the receiver specifies the transmitter (and vice versa) in the absence of errors.

Now consider the case of the noisy channel, as shown in fig. 4.2(b). In this case, the receiver entropy will not be the same as the transmitter's in general, and the transition probabilites will be nonzero for j=k. The conditional entropy $H(S_1|S_2)$ is now nonzero and, in fact, measures the freedom of the transmitter with respect to the receiver. In other words, it measures the information lost in the channel. Thus if we take the difference of the information originally available, as measured by the source entropy, and the information lost in the channel, as measured by the conditional entropy, we get a measure of the information transmitted over the channel. This measure is known as the mutual information of the channel, i.e.

$$I(S_1,S_2) = H(S_1)-H(S_1|S_2)$$

$$= \sum_{k=1}^{K} p(a_k)\log\frac{1}{p(a_k)} - \sum_{k=1}^{K}\sum_{k=1}^{K} p(a_k \text{and } b_j)\log\frac{1}{p(a_k|b_j)}$$

$$= \sum_{k=1}^{K}\sum_{j=1}^{K} p(a_k \text{and } b_j)\log\frac{p(a_k|b_j)}{p(a_k)}$$

or, since we can also think of the freedom of the receiver with respect to the transmitter,

$$I(S_1,S_2)= \sum_{k=1}^{K}\sum_{j=1}^{K} p(a_k \text{and } b_j)\log\frac{p(b_j|a_k)}{p(b_j)}$$

$$I(S_1,S_2)= \sum_{k=1}^{K}\sum_{j=1}^{K} p(b_j|a_k)\ p(a_k)\log\frac{p(b_j|a_k)}{p(b_j)} \quad \text{bits/symbol}$$

In the application described here, the thresholding operation reduces a multi-symbol channel to a binary channel. Ideally, we

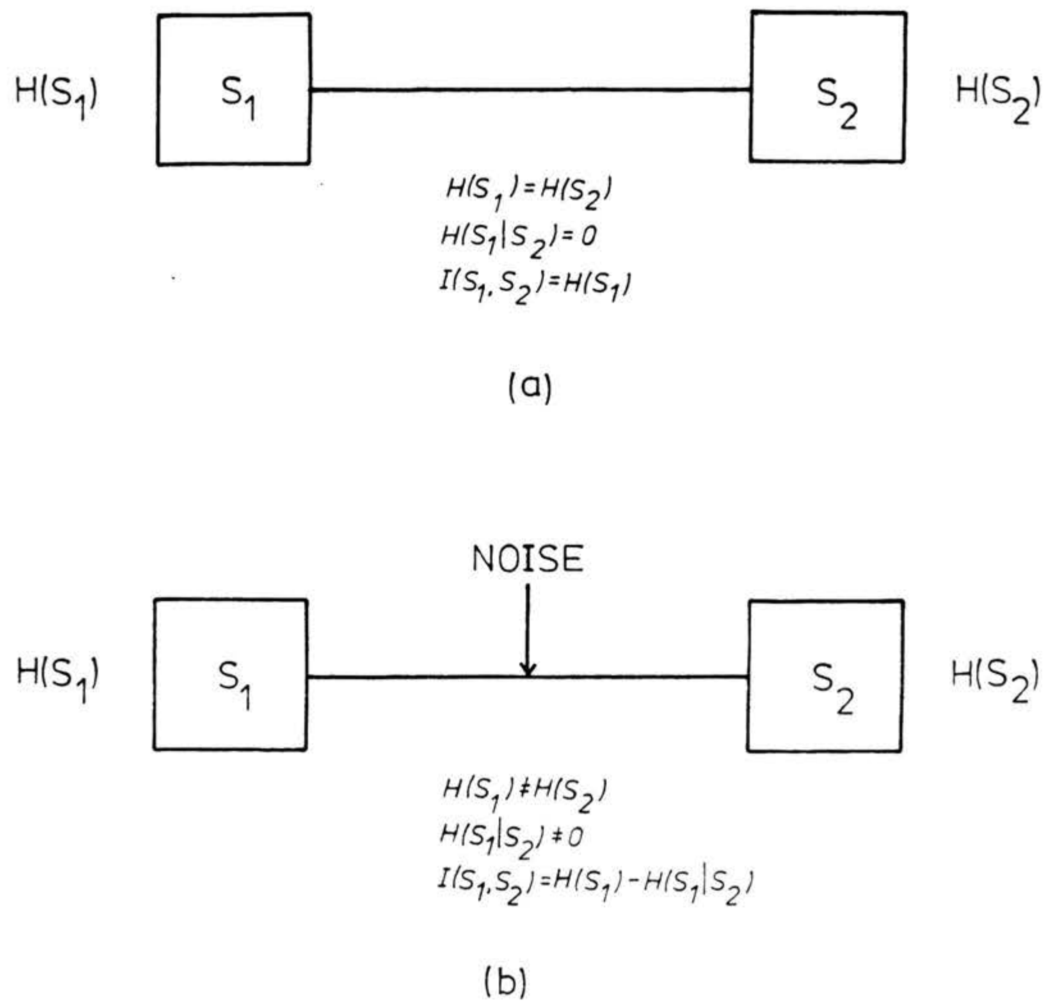


Fig. 4.2
In the noiseless channel, shown in (a), the source entropy is the same as the receiver entropy and because the channel has no freedom of choice the conditional entropy of the transmitter with respect to the receiver is zero. In the noisy channel, shown in (b), the source and receiver entropies are no longer the same and the conditional entropy becomes a measure of the information lost in the channel. Taking the difference of the source entropy and the conditional entropy produces the mutual information, which is a measure of the information transmitted over the channel.

would like to select a threshold on the noise-free edge map from semantic considerations, then find the equivalent threshold on the noisy edge map producing the maximum information transfer over the binary channel. However, because of the difficulty of choosing a threshold on semantic grounds we begin with a method for finding the threshold which simply maximises information transfer over the binary channel unconditionally. In terms of first differencing we find the threshold which, when applied to the noisy first differences, produces the edge map most similar to that which would be obtained from the noise-free first differences using the same threshold.

Remembering that the mutual information is the difference between the source entropy, which is a measure of the information available, and the conditional entropy, which is a measure of the information lost in the channel, we can interpret the maximisation as follows. Beginning with a very high threshold value, we have small source entropy because of the imbalance in the probabilities of edge and no-edge (few edges generated), and low conditional entropy because the high threshold produces a small probability of error (few spurious edges). As the threshold is decreased, the source entropy increases (more edges generated) as does the conditional entropy (more spurious edges). For a typical first difference distribution (unimodal, peak around zero), the conditional entropy initially rises more slowly than the source entropy. The threshold selected is the one where reducing it any further increases the conditional entropy more than the source entropy. This means that we should obtain the threshold producing the largest number of significant edges consistent with keeping the number of erroneous edges small.

Since we are only interested in whether an edge is significant or not, the system is represented by a binary channel as shown in fig. 4.3.

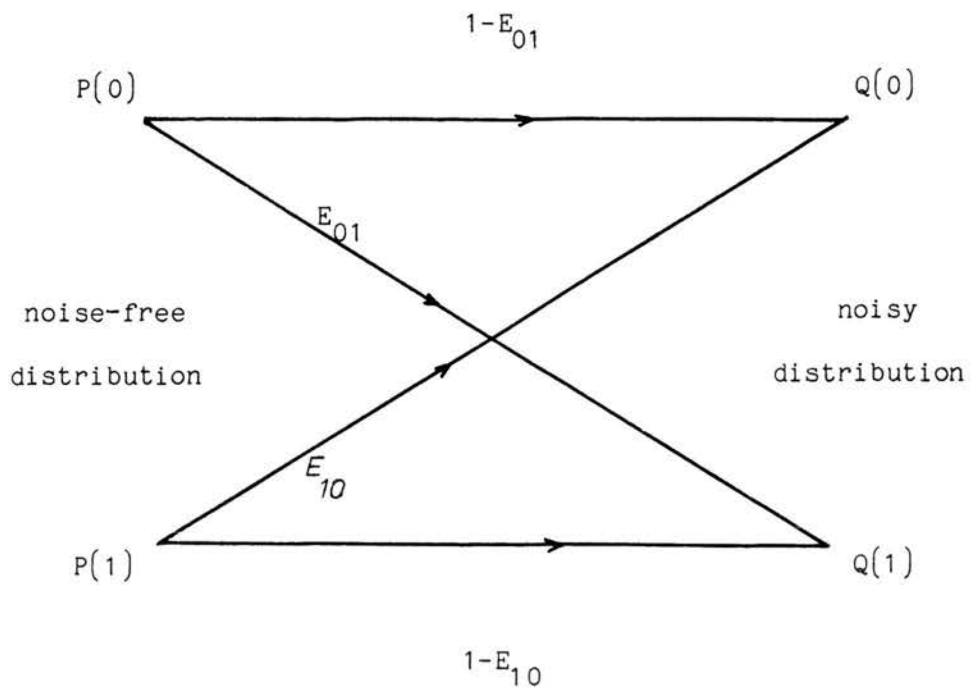


Fig. 4.3

The transition diagram for a binary channel representing the effect of noise on the thresholding process.

P(0), P(1), Q(0), and Q(1) are already known, additionally it is necessary to find E_{01}, the probability of a spurious boundary being generated due to noise, and E_{10}, the probability of a real boundary being lost due to noise. This information can be obtained from the much larger transition diagram representing the greyscale system. Consider an individual transition, whose conditional probability is given by N(j), from I(i) to O(k), i.e. k=i+j. Four different types of such transitions occur as shown in fig. 4.4.

To find the probability of the transition labelled E_{01} in fig. 4.3, we sum I(i)*N(j) for every transition such that

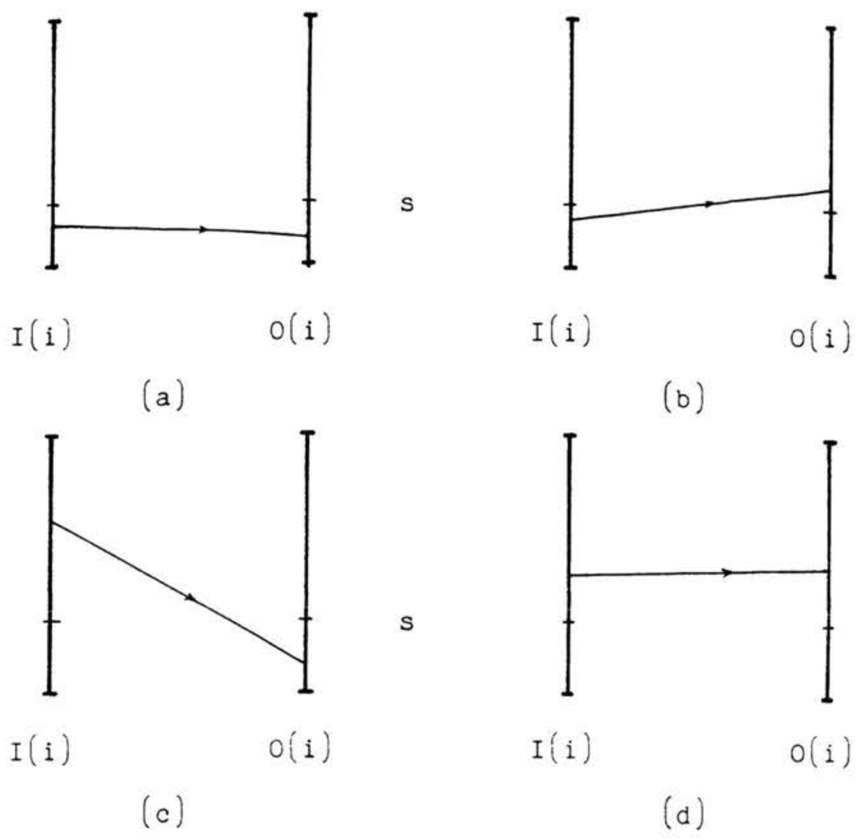


Fig 4.4

This shows the four types of transition on the full system transition diagram, enabling it to be reduced to a binary channel. The four types are:

a) The magnitude of this boundary both before and after noise is applied is not significant.

b) In this case, a boundary which is not significant becomes so due to noise.

c) Here, a significant boundary loses its significance due to noise.

d) Finally, in this case a significant boundary is not sufficiently affected by noise to lose its significance.

$$|i| \leq s$$

$$\text{and} \quad |i+j| > s$$

Similarly, to find the overall probability of the transition E_{10}, we sum $I(i)*N(j)$ for every transition for which

$$|i| > s$$

$$|i+j| \leq s$$

Now E_{01} and E_{10} are conventionally given as conditional probabilities, and since $p(b|a)=p(a \text{ and } b)/p(a)$ we must divide the overall probability by the appropriate source probability

$$E_{01} = \frac{\sum_{i=-imax}^{imax} \sum_{j=-jmax}^{jmax} I(i)*N(j)}{P(0)}$$

$$E_{10} = \frac{\sum_{i=-imax}^{imax} \sum_{j=-jmax}^{jmax} I(i)*N(j)}{P(1)}$$

The general formula for mutual information is

$$I(S_1,S_2) = \sum_{i=imin}^{imax} \sum_{j=jmin}^{jmax} p(b_j|a_i).p(a_i)\log_2\frac{p(b_j|a_i)}{p(b_j)}$$

where the a_i refer to transmitter probabilities and the b_j to receiver probabilities. In the particular case of the binary channel of fig. 4.3, we get

$$I(S_1,S_2) = (1-E_{01})P(0)\log_2 \frac{(1-E_{01})}{Q(0)} + E_{01}P(0)\log_2 \frac{E_{01}}{Q(1)}$$

$$+ E_{10}P(1)\log_2 \frac{E_{10}}{Q(0)} + (1-E_{10})P(1)\log_2 \frac{(1-E_{10})}{Q(1)} \quad \text{bits/edge.}$$

In applying this formula, the noise-free distribution of first differences is not available. However, the noisy distribution of first differences is available and can initially be used as a noise-free distribution. Applying the method produces the mutual information as a function of the threshold, and also produces an output distribution, which is the input distribution blurred by noise. This output can be used as another input distribution to obtain a second mutual information function and another blurred distribution. This process can be repeated as often as desired, then back extrapolation can be used to obtain the best threshold for the original image.

Justification for the extrapolation comes from the fact that the sum of two variables with Gaussian distributions also has a Gaussian distribution. If the original first difference distribution approximates a Gaussian (which it often does in practice) then adding noise to the image will just produce another Gaussian edge distribution with a slightly larger variance. This follows from the fact that the convolution of a Gaussian of variance σ_1^2 with a Gaussian of variance σ_2^2 is another Gaussian with variance $\sigma_1^2 + \sigma_2^2$ as shown in Appendix 1. Hence, if the first difference distribution of the original image is Gaussian, then successively blurring it with Gaussian noise produces a series of Gaussian distributions with increasing variances.

Since the shape of the image distribution is fixed (i.e. assumed Gaussian) and the shape of the noise distribution is also fixed (assumed Gaussian) the threshold producing the maximum mutual information must only be a function of the ratio of the image variance to the noise variance. Unfortunately, it is not easy to derive an analytical expression for this function since the integral of an error function is encountered in the analysis. However, the relationship between the variance ratio and the best threshold can be investigated numerically. By generating a series of Gaussian distributions and using them to represent the image distribution of first differences, the threshold value giving maximum mutual information can be plotted as a function of the ratio of the first difference variance to the noise variance as shown in fig. 4.5. It turns out that this relationship is very nearly linear for variance ratios between 1 and 9. In practice, the series of Gaussians obtained by successively blurring an image distribution will not have linearly increasing variances, but will instead have variances given by

$$\sigma_p^2 = \sigma_{p-1}^2 + \sigma_n^2$$

where σ_n^2 is the noise variance, σ_{p-1}^2 is the variance of the original first difference distribution blurred p-1 times, and σ_p^2 is the variance of the pth. blurred first difference distribution. Rearranging this,

$$\frac{\sigma_p^2}{\sigma_n^2} = 1 + \frac{\sigma_{p-1}^2}{\sigma_n^2}$$

Now, in the linear portion of fig. 4.5,

$$T_p = C_1 \frac{\sigma_p}{\sigma_n} + C_2$$

where T_p is the threshold producing the maximum mutual information on the pth. iteration and C_1 and C_2 are constants. So

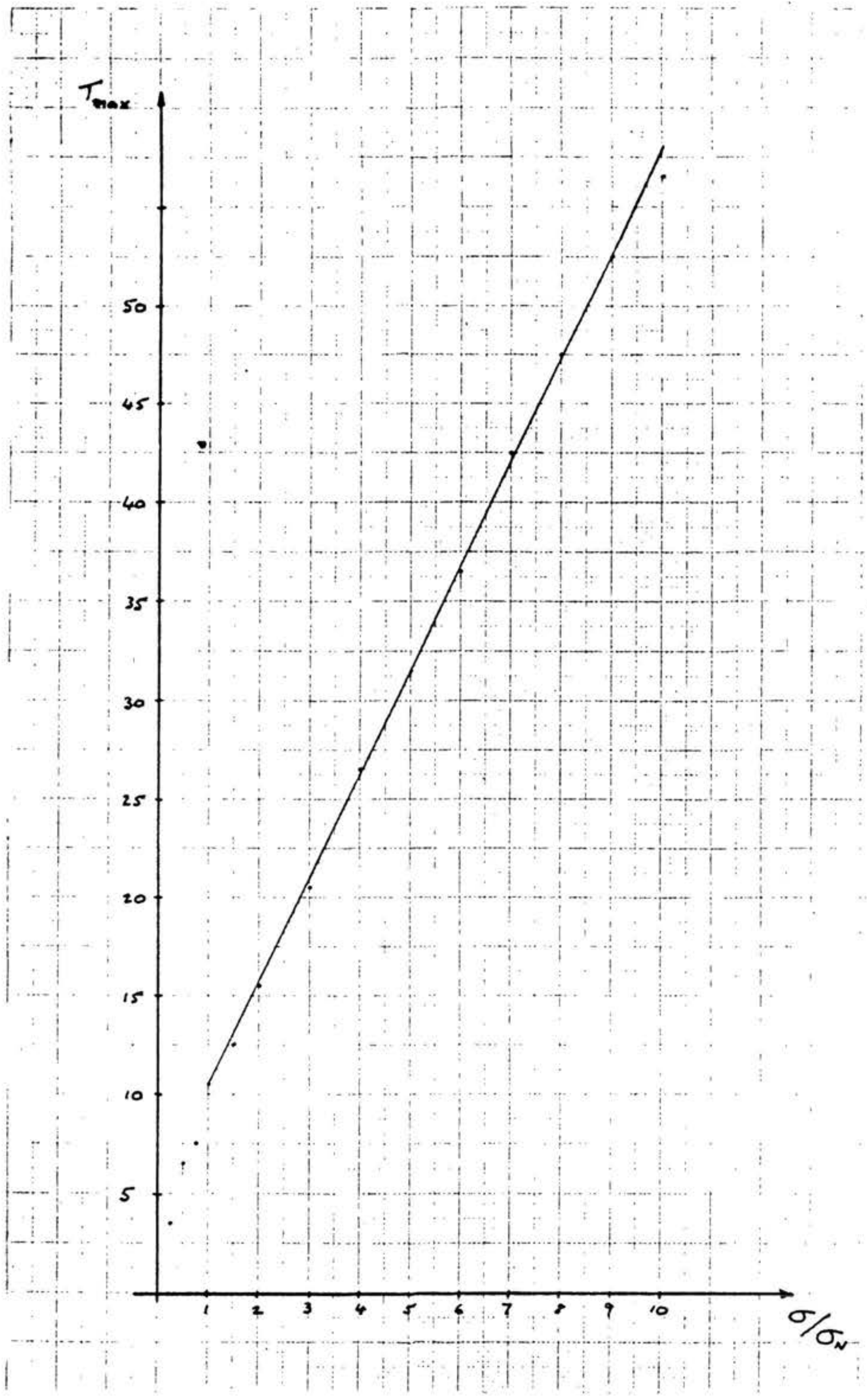


Fig. 4.5

Threshold versus variance ratio.

$$\frac{(T_p - C_2)^2}{C_1^2} = 1 + \frac{(T_{p-1} - C_2)^2}{C_1^2}$$

$$(T_p - C_2)^2 = C_1^2 + (T_{p-1} - C_2)^2$$

The constants, C_1 and C_2, can be determined from fig. 4.5. So, given the threshold producing the maximum mutual information on any iteration, it is straightforward to use the recursive equation given above to find the corresponding threshold on any other iteration. In particular, given the maximum mutual information threshold on the first iteration, i.e. the threshold resulting from applying the method to the first difference distribution obtained from the actual (noisy) image, the appropriate threshold on the ideal (noise-free) image can be obtained. The best threshold on the noise-free image is the one, according to the method outlined here, which will give the edge map, resulting from first differencing and thresholding, most similar to that which would have been obtained from the noise-free image.

If the distribution of first differences is only approximately Gaussian or the variance ratio is not in the range 1-9, then the relationships defined above will not hold exactly, but a maximum mutual information threshold successively increasing with each iteration can still be expected and the set of such thresholds can be used in back extrapolation to estimate the maximum mutual information threshold on the noise-free image.

The algorithm used is as follows:

Threshold-finding Algorithm

1. Construct first difference distribution for given image.
2. For every threshold value in range compute mutual informa-

tion I(S1,S2).

3. Store threshold value which gives max{I(S1,S2)}.
4. Convolve noise with first difference distribution to generate new first difference distribution.
5. Repeat 2-4, n-1 more times.
6. Using the n values of best threshold, extrapolate back to get best threshold for original image.

For practical purposes, blurring the input distribution is carried out separately from the mutual information calculation. The algorithm is summarised graphically in fig. 4.6.

The variation in the value of the mutual information as the threshold is varied can be investigated with the aid of test distributions. A flat distribution is shown in fig. 4.7(a). As the threshold is increased the information transmitted will increase until the source probabilities are equal (threshold in centre of range). The information lost in the channel will be constant above a certain low value of threshold. Hence, we expect the mutual information to be a maximum at a threshold value of about 127, dropping off to zero symmetrically on each side. The resulting graph of mutual information vs. threshold is shown in fig. 4.7(b).

A second test distribution consisting of two equal-sized Gaussian peaks is shown in fig. 4.8(a), with the resulting graph of mutual information vs. threshold in fig.4.8(b). In this case, both the tendency to equalise the source probabilities, and the tendency to reduce errors in the channel combine to produce a threshold (maximum mutual information) midway between the two peaks.

To investigate the relative effect of equalising source probabilities versus minimising channel errors, the third test

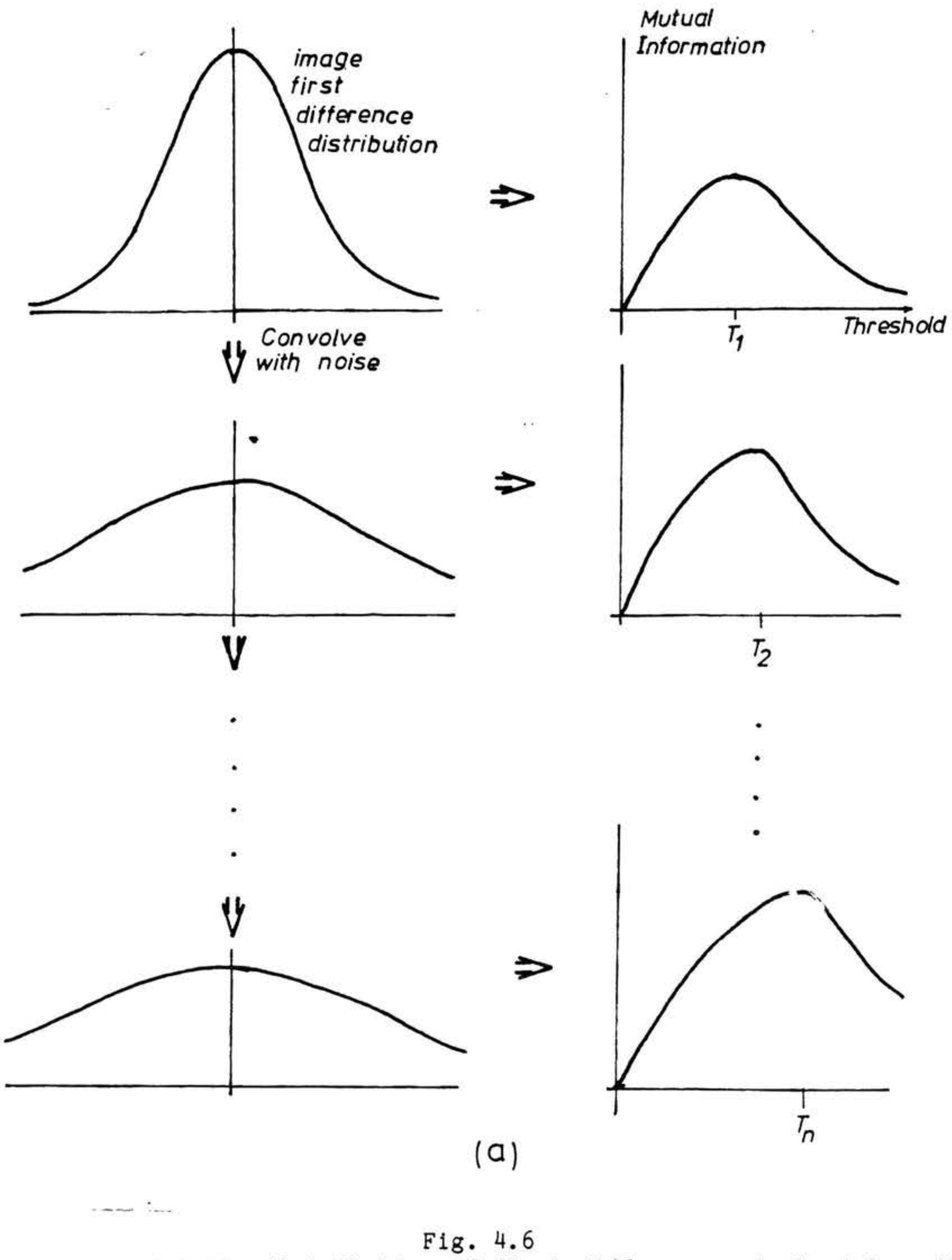
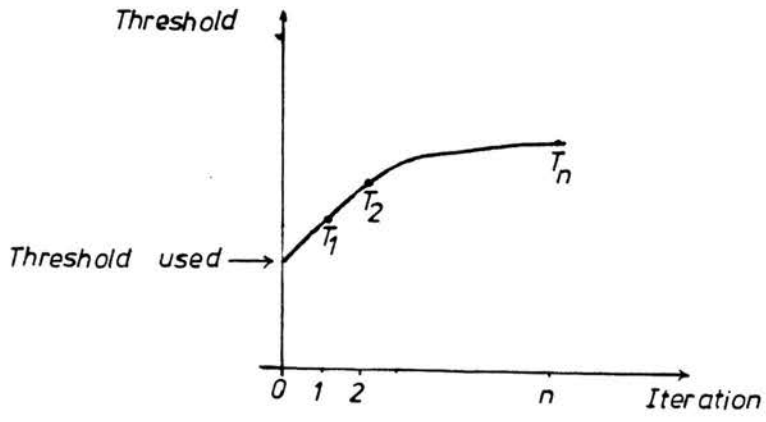


Fig. 4.6
Beginning with the distribution of first differences derived from the image, the mutual information is plotted against threshold using the known noise distribution. The image distribution is then blurred with the noise and the resulting distribution of first differences obtained. Again, the mutual information is plotted against threshold for this distribution. This process of blurring followed by calculating the mutual information as a function of threshold is repeated several times, as shown in fig. 4.6(a) above. Finally, as shown in fig. 4.6(b) overleaf, the thresholds producing the maximum mutual information are plotted and used to estimate the best threshold for the zeroth iteration. This is taken to be an estimate of the best threshold for the original image.



(b)

distribution shown in fig. 4.9(a) is appropriate. Again this consists of two Gaussian peaks, the one on the left of area 0.75 and the one on the right of area 0.25. The resulting graph of mutual information vs. threshold is shown in fig. 4.9(b), with its maximum again roughly midway between the peaks. This ability to select a threshold in the valley between two peaks of dissimilar size is often listed as a desirable property of threshold selection techniques.

The distribution of first differences from most real images tends to be unimodal with a peak around zero. In this case, very low threshold values produce a high probability of error, keeping the information transmitted low. As the threshold increases away from the peak the effect of errors in the channel becomes negigible but the imbalance in the source probabilities increases, again producing a low information transfer. The maximum information transmitted occurs with a threshold value between these extremes.

As an example we will use the sports shoe picture (shown in fig. 5.18). The initial distribution of first difference magnitudes is shown in fig. 4.10. Since we are only interested in edge magnitude, the histogram is one-sided. The resulting mutual information as a function of threshold is shown in fig. 4.11. Iterating the method twice more produces the mutual information functions shown in fig. 4.12. Taking the maxima of these three curves and extrapolating backwards suggests a threshold value of 18 greylevels. The edge maps resulting from thresholds of 10-30 greylevels are shown in fig. 4.13.

Satisfactory performance of this algorithm depends on several factors. Basically, it finds the threshold which generates the largest number of edges subject to a low probability of error. This means that in a low-noise situation the algorithm would recommend a very low threshold, which may not be suitable for some early processing systems. In the limit, if we had a completely noise free image, the algorithm would always recommend using the threshold which produces an equal probability of edge and no-edge. This would probably not be desirable for most systems. The behaviour of the algorithm has already been described for several different types of first difference distributions. For the most common type of distribution (unimodal, peak around zero) the algorithm recommends a threshold value similar to one a human user might choose if he or she had access to the distributions. For atypical distributions, which could be produced, for example, by texture field images, or by close-ups of surfaces with strong illumination gradients, it is probably desirable to base the choice of threshold on semantic considerations as well as statistical ones. In these circumstances the algorithm described above may not be ideal.

An alternative way of applying the same basic method is to first choose a particular threshold on the noise-free distribution. This choice could be based on knowledge of the image contents, or on other semantic considerations. This threshold then remains fixed on the input distribution, while the output threshold is varied and the mutual information calculated. Selecting the threshold producing the maximum mutual information would generate the edge map most similar to the desired edge map resulting from the initial threshold choice. The algorithm for this technique is given below.

Alternative Threshold-Finding Algorithm

1. Choose desired threshold from semantic considerations.
2. Construct distribution of first differences.
3. Deconvolve noise distribution with first-difference distribution

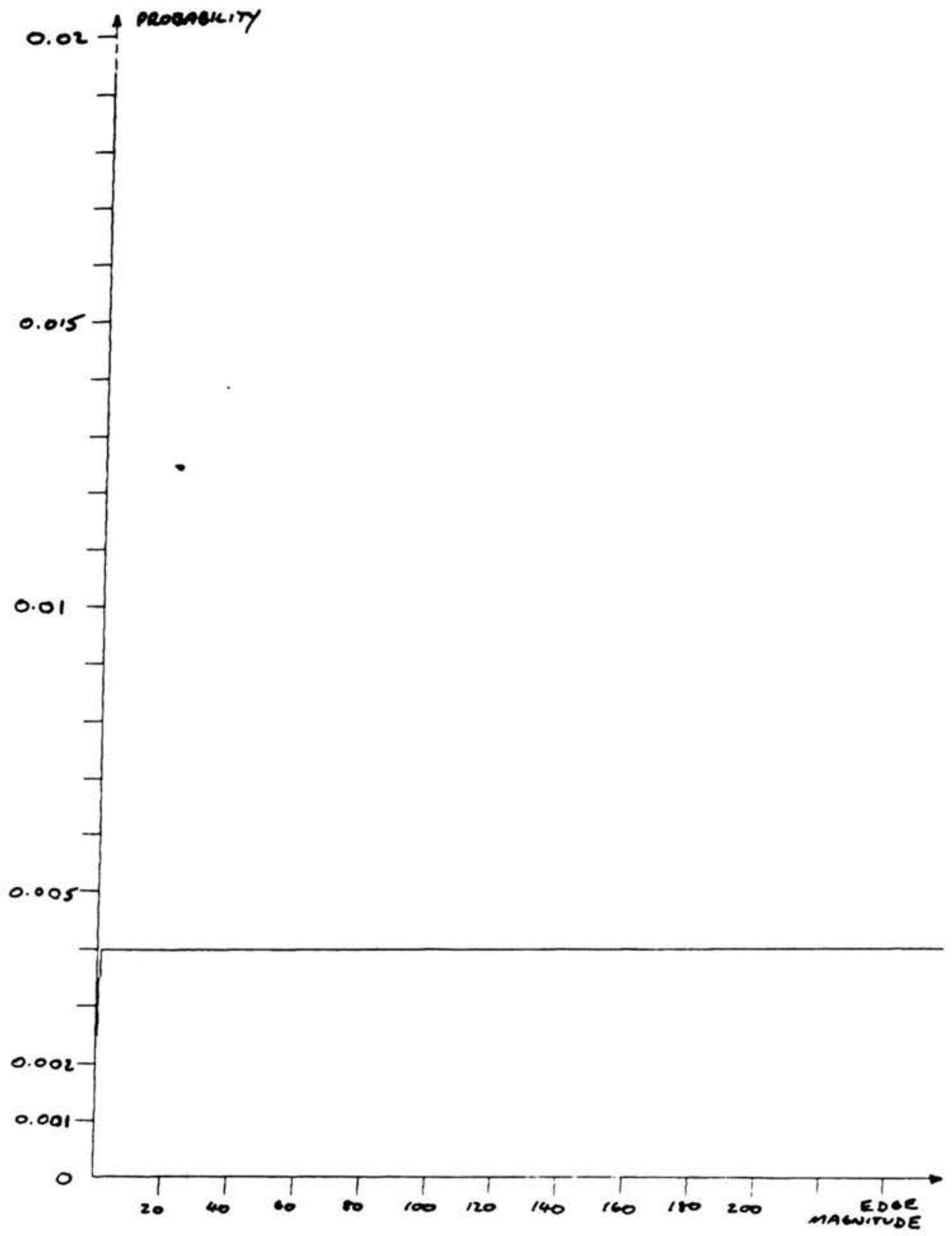
to produce noise-free distribution of first differences.

4. For fixed threshold in noise-free distribution (chosen in #1) and for every threshold value in range for noisy distribution (constructed in #2) compute I(S1,S2).
5. Use threshold value giving max{I(S1,S2)}.

Difficulties in applying this algorithm arise in steps 1 and 3. The main difficulty lies in choosing a desired threshold value on semantic grounds. For a given restricted set of images, the most useful threshold could be estimated by empirical methods, but for a more general scene analysis system it would be better to have a theoretically well-founded way of choosing the ideal threshold. However, this is a problem which is, as yet, difficult to formulate, let alone solve, so it has not been investigated in this thesis.

Deconvolving the noise distribution with the first difference distribution is not difficult if both distributions are simply treated as deterministic signals. It is then identical to deblurring a (line of a) blurred image - a standard problem in image restoration (Kimia and Zucker 1983). However, in the application considered here, we are dealing with finite samples of statistical distributions, so the problem may be ill-conditioned.

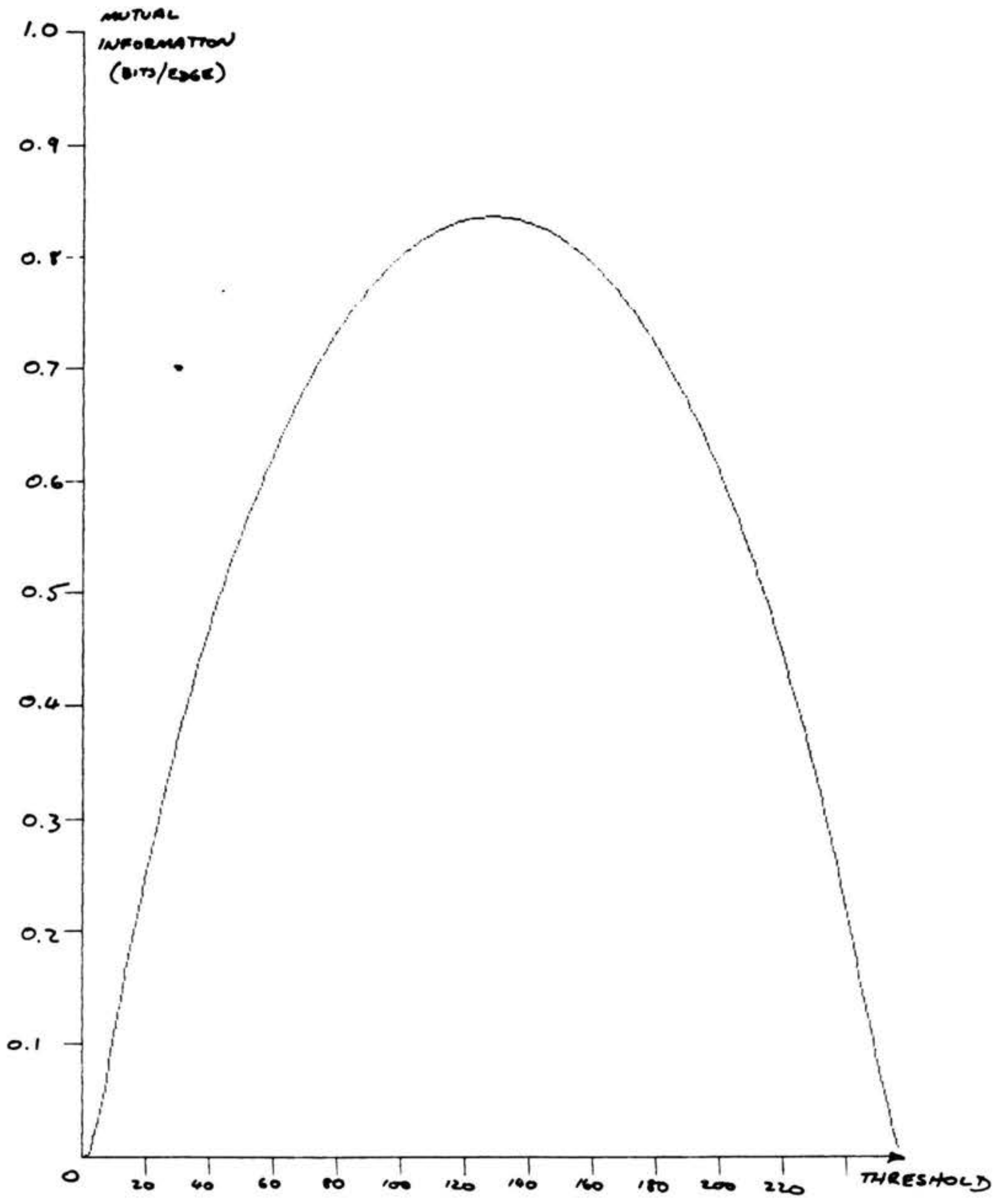
To summarise this chapter: For various reasons, first differencing followed by thresholding is chosen as the first processing step. The effect of noise on the image is modelled by a communication channel, and the threshold which maximises channel capacity is applied to the first differences.



(a)

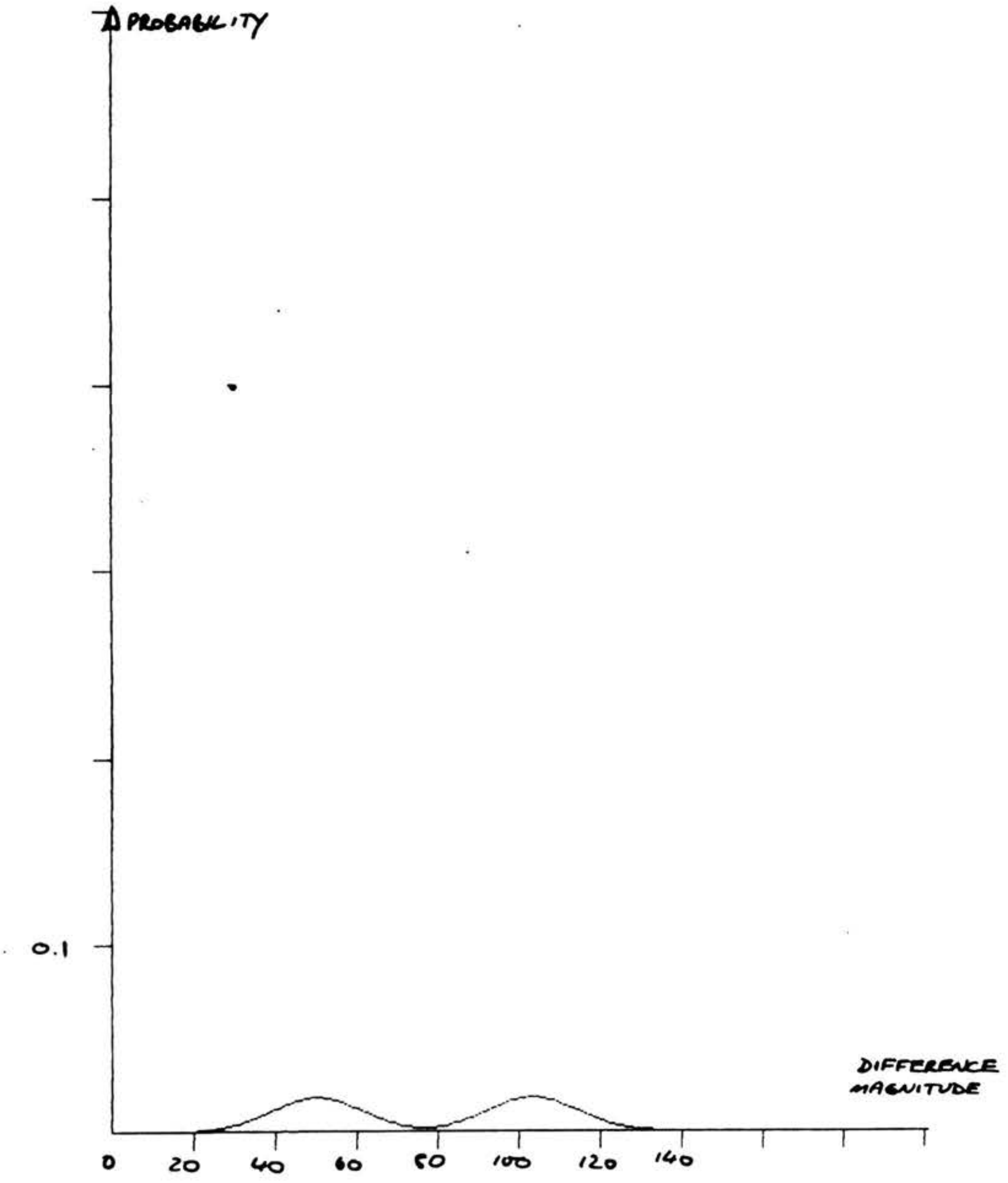
Fig. 4.7

The probability distribution, shown in fig. 4.7(a), is flat, producing the graph of mutual information shown in fig. 4.7(b) overleaf. The maximum mutual information occurs with a threshold in the centre of the intensity range.



MUTUAL INFORMATION (BITS/EDGE)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140
160
180
200
220
THRESHOLD

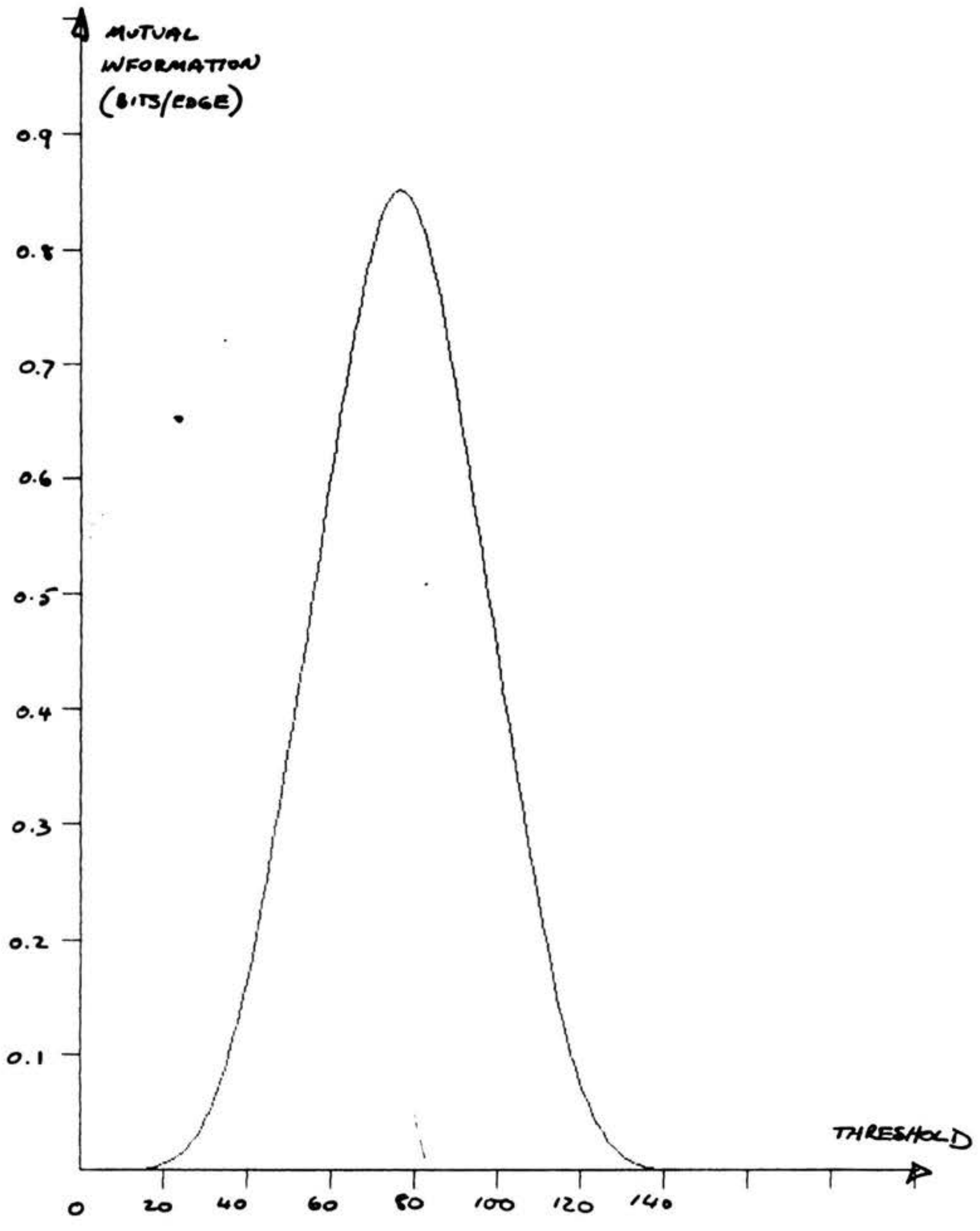
(b)



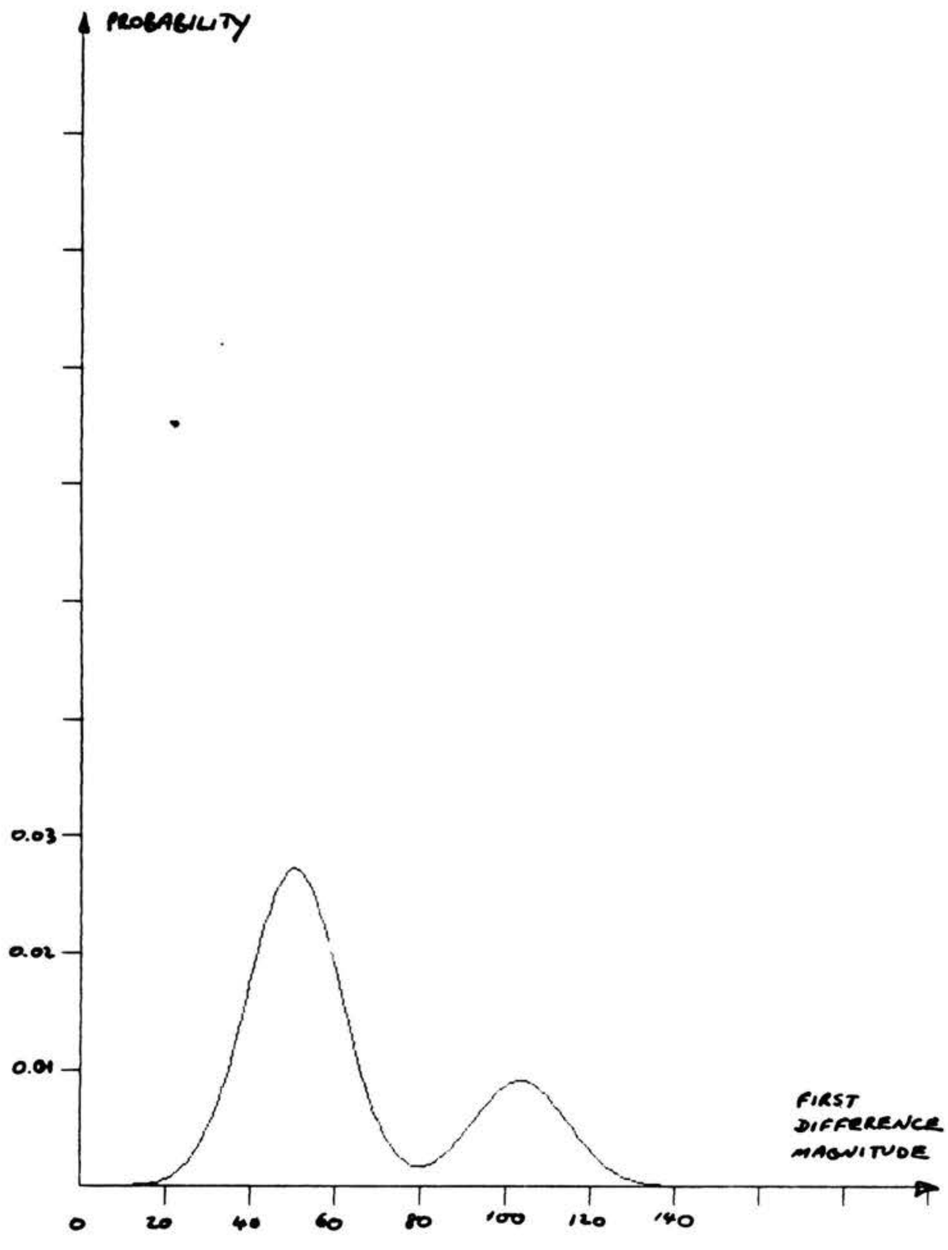
(a)

Fig. 4.8

The test probability distribution, shown in fig. 4.8(a) above, consists of two Gaussians of equal area. As with all the examples used in this chapter, the noise distribution used was Gaussian with a standard deviation of 7.75 greylevels. The resulting mutual information, shown overleaf in fig. 4.8(b), peaks in the centre of the trough in the input distribution.


MUTUAL INFORMATION (BITS/EDGE)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
THRESHOLD
0
20
40
60
80
100
120
140

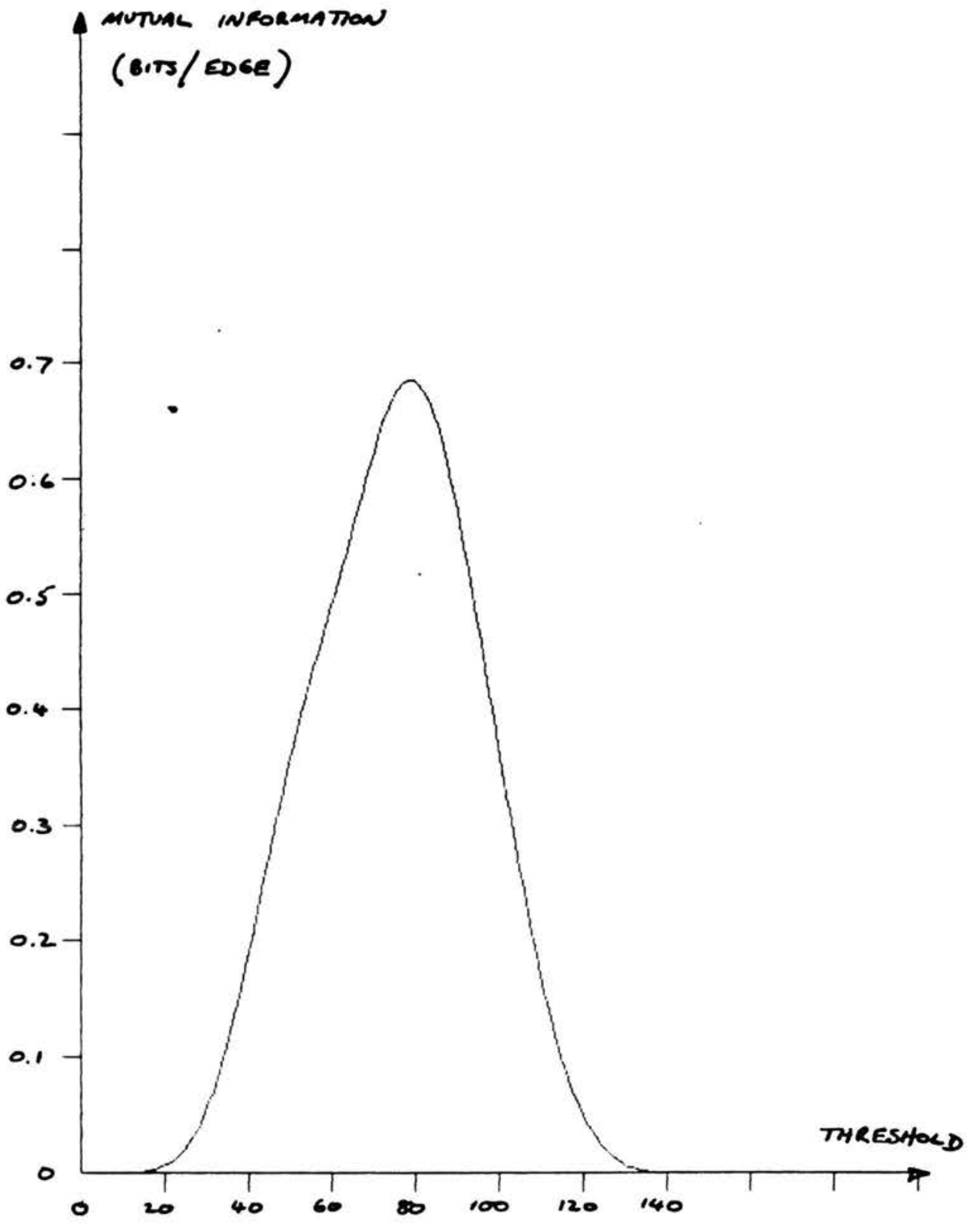
(b)



(a)

Fig. 4.9

This test distribution, above, consists of two Gaussians of unequal area. The resulting graph of mutual information, overleaf, again peaks in the trough of the distribution.


MUTUAL INFORMATION
(BITS/EDGE)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
THRESHOLD
0
20
40
60
80
100
120
140

(b)

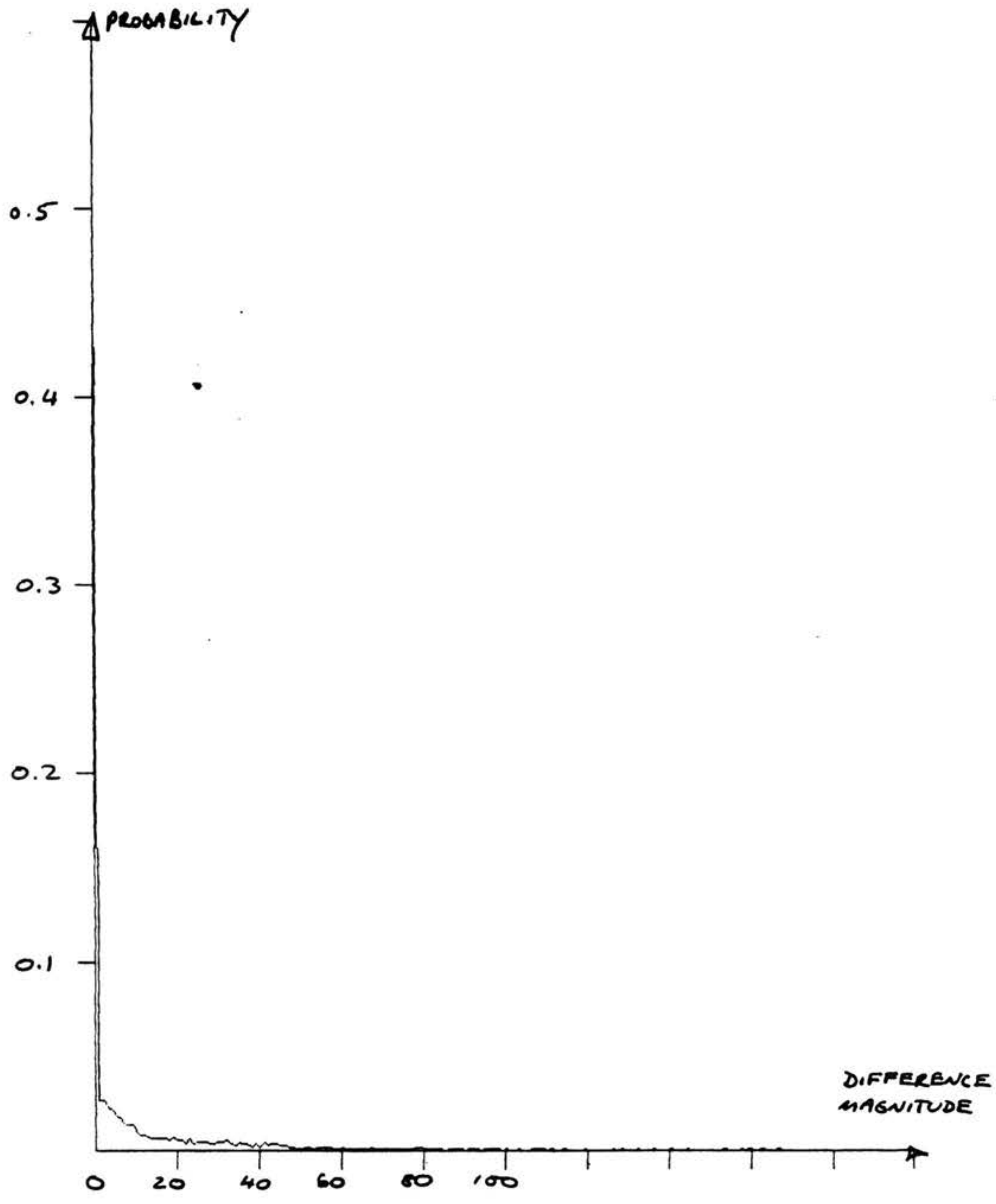


Fig. 4.10

This shows the distribution of first differences in the shoe picture.

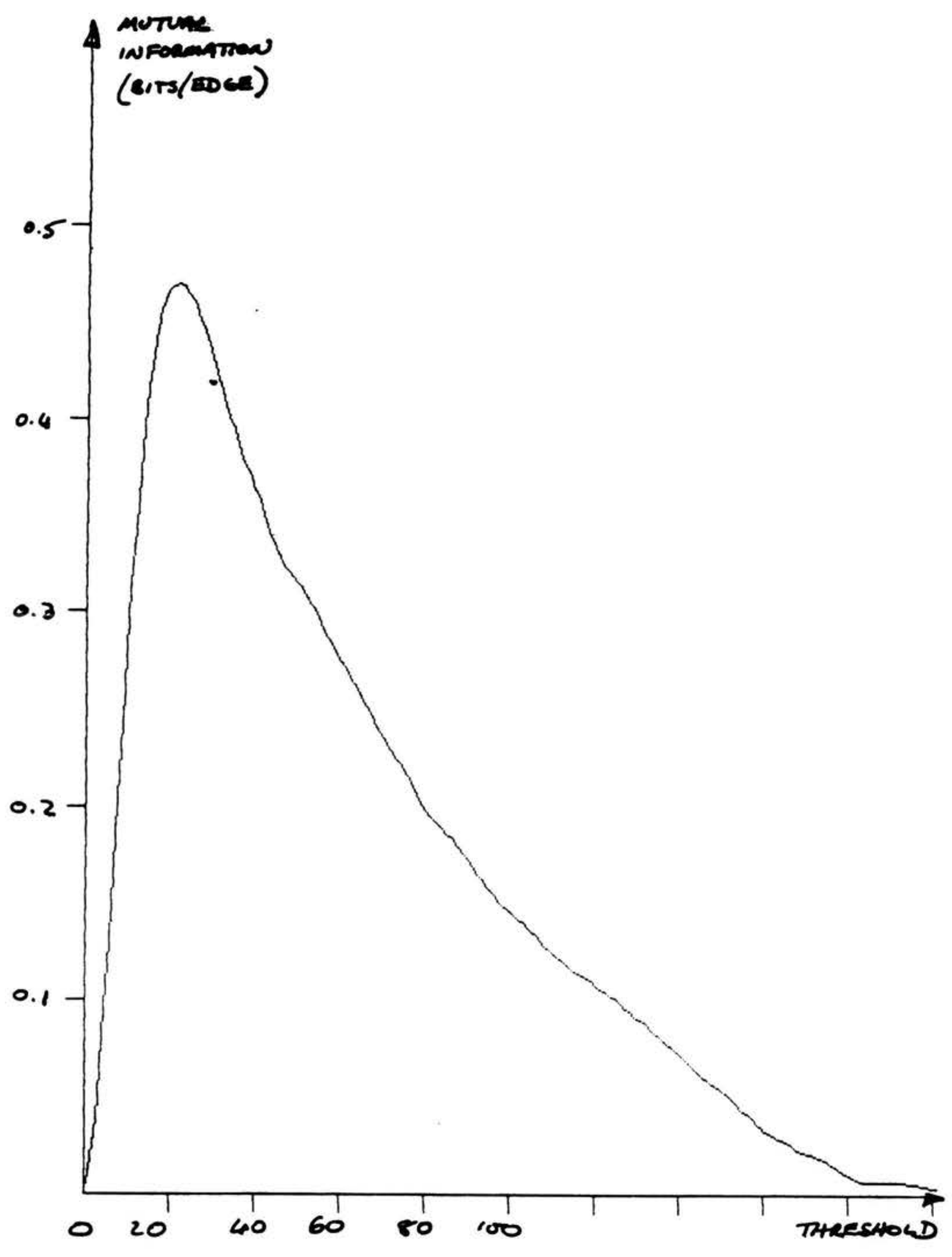
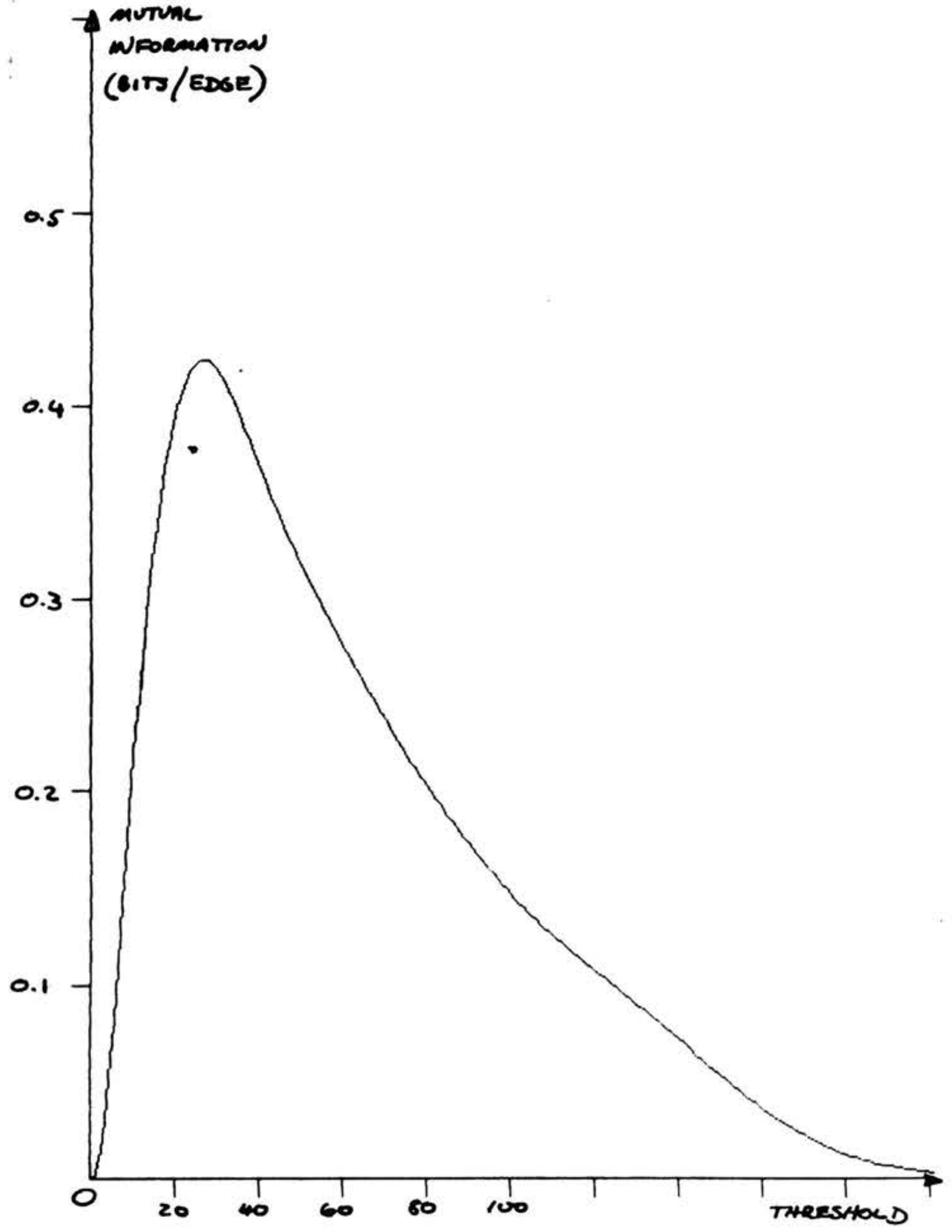


Fig. 4.11

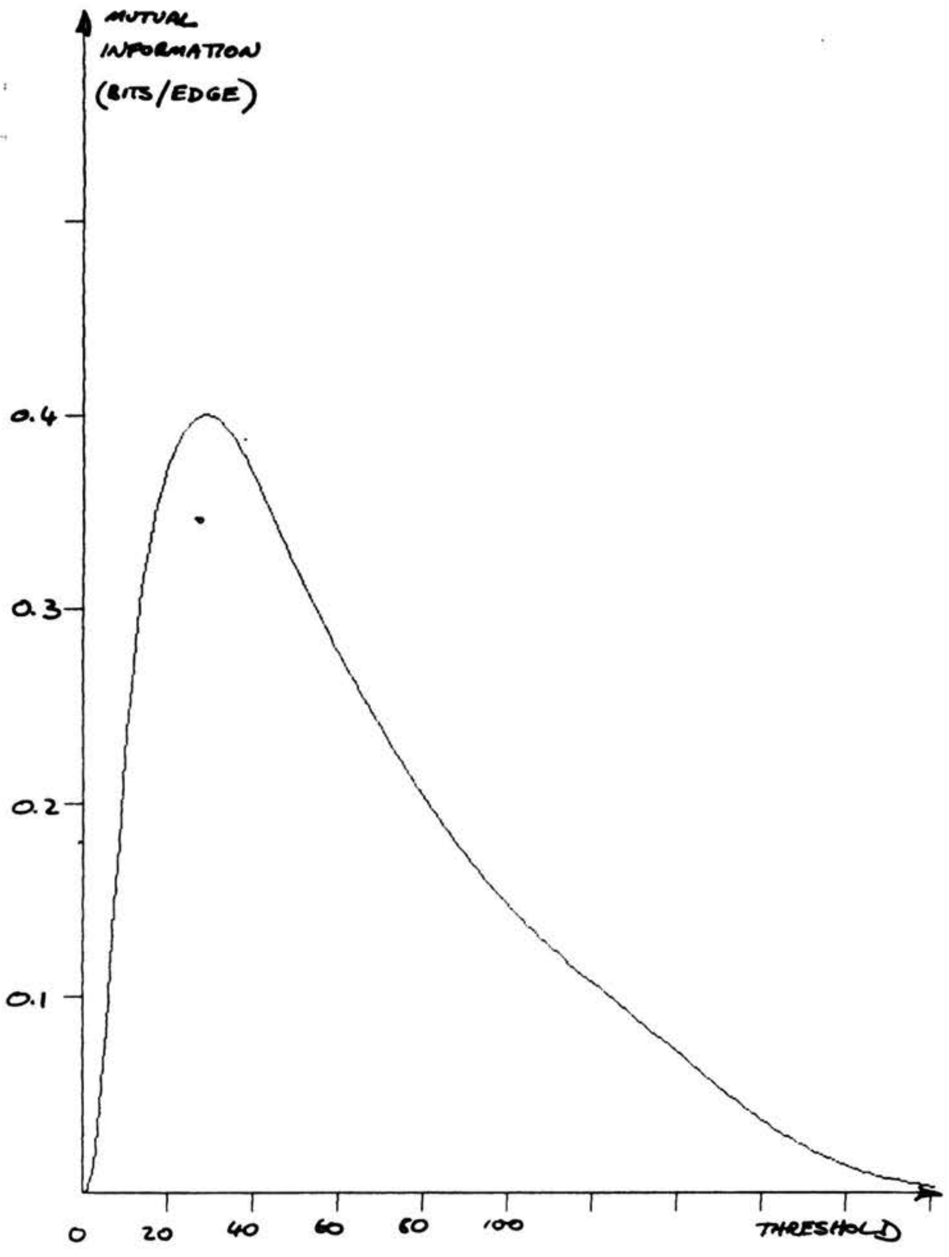
This shows the mutual information as a function of threshold using the distribution of fig. 4.10 as input.



(a)

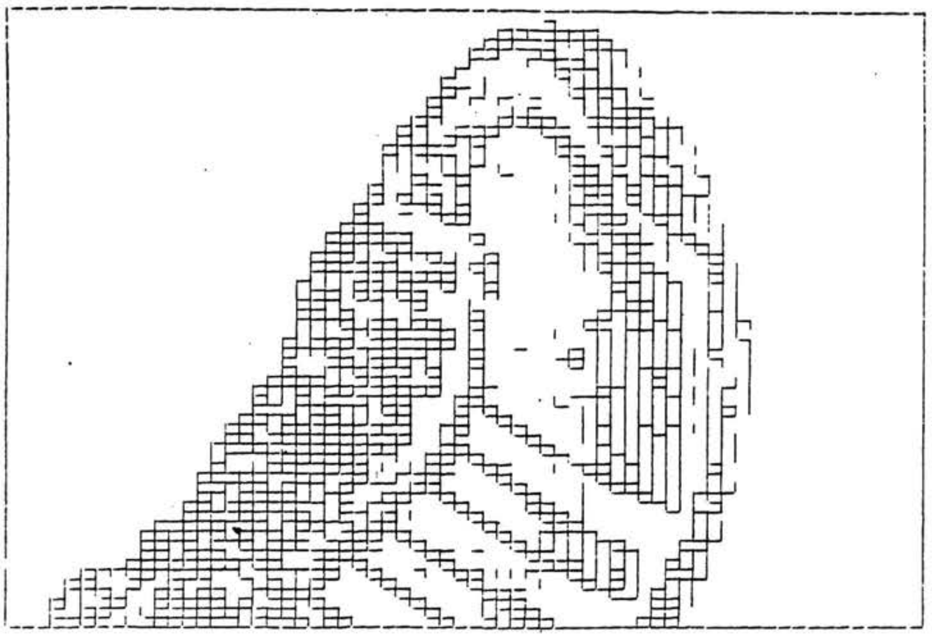
Fig. 4.12

The graph of mutual information vs. threshold, above, is the result of using the blurred version of the distribution shown overleaf as input. The graph overleaf is the result of using the same distribution blurred twice.



MUTUAL INFORMATION (BITS/EDGE)
0.4
0.3
0.2
0.1
0
20
40
60
80
100
THRESHOLD

(b)


(a)

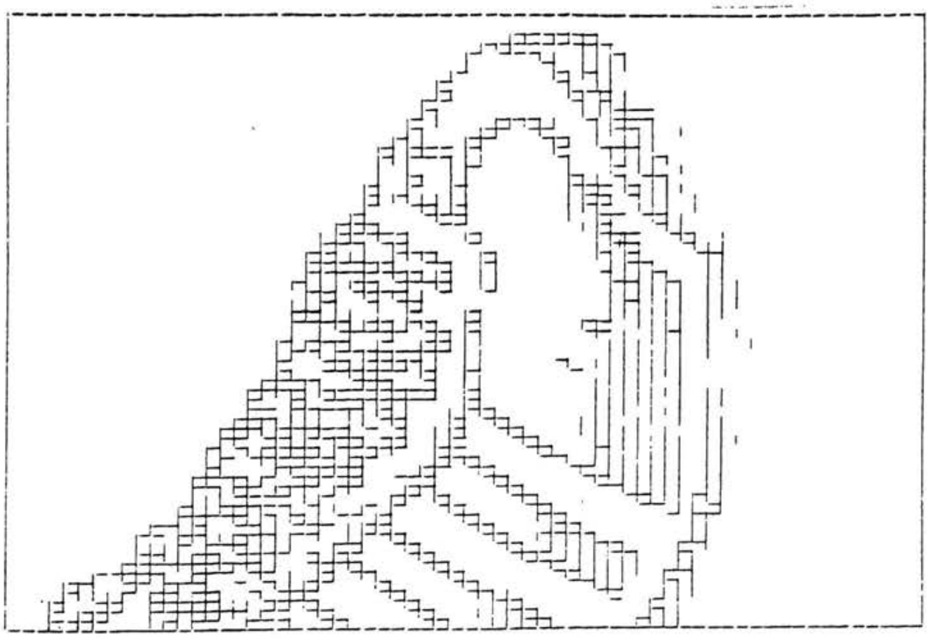

(b)

Fig. 4.13

These edge maps show the primary boundary segments produced with thresholds of:

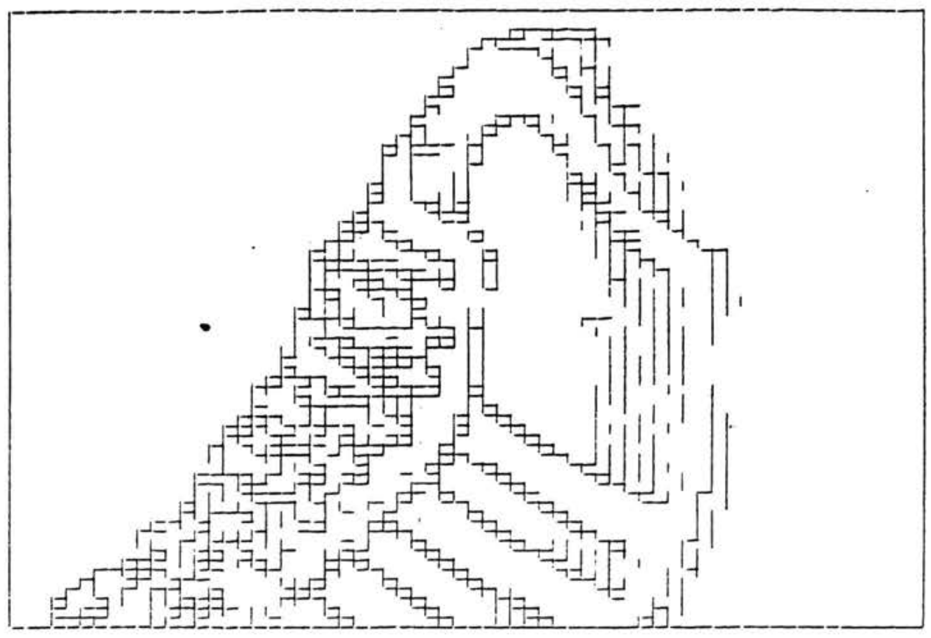
a) 10

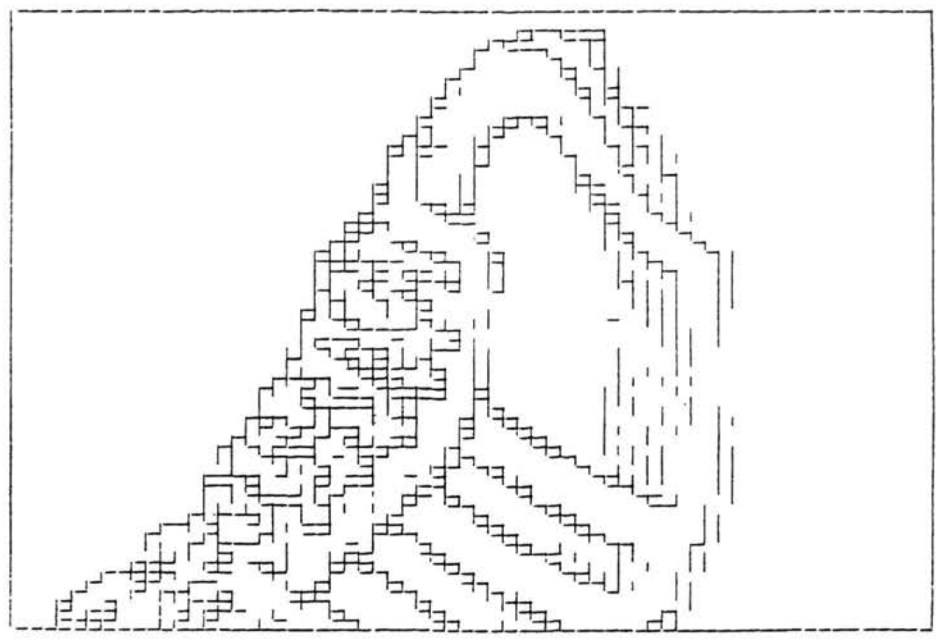
b) 18

c) 25

d) 30

The edge map shown in (b) is the one chosen by the thresholding method.


(c)


(d)

5 The Single Level System

Although we ultimately want to find and represent edges of a range of widths, it is difficult to do so in a single pass over the image. Hence, we begin by finding sharp, step-like edges and representing them accurately. At the same time we mark other interesting image entities (as gradient regions or texture regions), so that when it comes to constructing the multi-level system, integration of the outputs of the various single-level systems will be facilitated.

In fig. 4.10, we saw the effect of labelling boundary segments as significant or not. It is noticeable that there seems to be two distinct types of grouping of segments. One type consists of large regions of the image containing many significant boundary segments, while the other type is thin and extended. This is quite reasonable, since fuzzy regions, for example as produced by blurred shadows, have significant width even along the direction of steepest intensity gradient, while step-like edges are always thin in one direction, namely, perpendicular to the edge. In view of our previous desire to classify image intensity changes into three types (uniform, fuzzy, and step) and since the boundary segments now labelled primary (significant) should include both fuzzy regions and steps, the next problem is to distinguish between these two classes and find and represent them in a useful way. Because of their different image characteristics, it is necessary to utilise a range of techniques to analyse the image further in terms of these components.

5.1 Finding Step Edges

In view of the importance of step-like edges in the goals of edge detection derived earlier, and since the patterns of primary boundary segments produced by step-like edges are more constrained, it is advantageous to deal with step-like edges first. Recall that the goals of edge detection which specifically apply to step-like edges were:

- find and represent the edges as accurately as possible
- try to maintain a clear relationship between edges and their underlying image intensities. In particular, know which pixels are mixed and which are pure.

Since these goals refer explicitly to the mixed pixel problem and express the need to find step edges accurately, while at the same time making as few limiting assumptions about the nature of the intensities in the vicinity of the edge as possible (this follows from the general goal of dealing with all significant intensity variations), the method of finding step-like edges is based on analysing how they transform under digitisation. The basic idea is to identify constraints among pixel intensities which must be satisfied by all step-like edges. These constraints can then be used as a test. Any image neighbourhood whose pixel intensities violate the constraints can be immediately excluded from further consideration. The constraints can then be further utilised to represent the edges in a useful way. Finally, if desired, the set of constraints produced can also be used to generate an explicit formal definition of a step edge.

5.1.1 Pixel Classification

First, we note the mixed pixel problem. Because of optical imperfections and the non-pointlike nature of the sampling, when a step edge is digitised, the intensity transition may lie, not between pixels, but within a pixel. This mixed pixel will have a greyscale value somewhere between those of its neighbours across the edge, as shown in fig. 5.1. The effect of this is that a step edge, even an ideal step edge, will not generate a single line of primary boundaries but pixels with one, two, three, or four primary boundaries depending on the local geometry and intensity variations of the edge relative to the quantisation grid. This phenomenon could be observed in fig. 4.10.

As a first pass at identifying mixed pixels, we could classify all pixels with more than one primary boundary segment as mixed, but this leads to errors. An edge that runs diagonally across the sampling grid as shown in fig. 5.2 can produce pixels such as those labelled "p" in the figure, which have more than one primary boundary segment, but are unmixed. To combat this problem only pixels which have two parallel primary boundary segments are labelled as mixed. This is reasonable since all step edges of sufficient amplitude, regardless of their attitude to the quantisation grid, will produce parallel pairs of primary boundary segments as shown in fig. 5.3.

In our attempt to identify the primary boundary segments belonging to step edges, the first task is to label all pixels with two parallel primary boundary segments as mixed. However, by no means all such pixels will be due to step edges, some of them will be caused by gradients and textured areas of the image. To find those really resulting from step edges, we can use a distinguishing property of such edges, which is their very limited physical extent perpendicular to the direction of the edge. Consider a pixel through which a step edge is passing, which is locally straight, as shown in fig. 5.4.

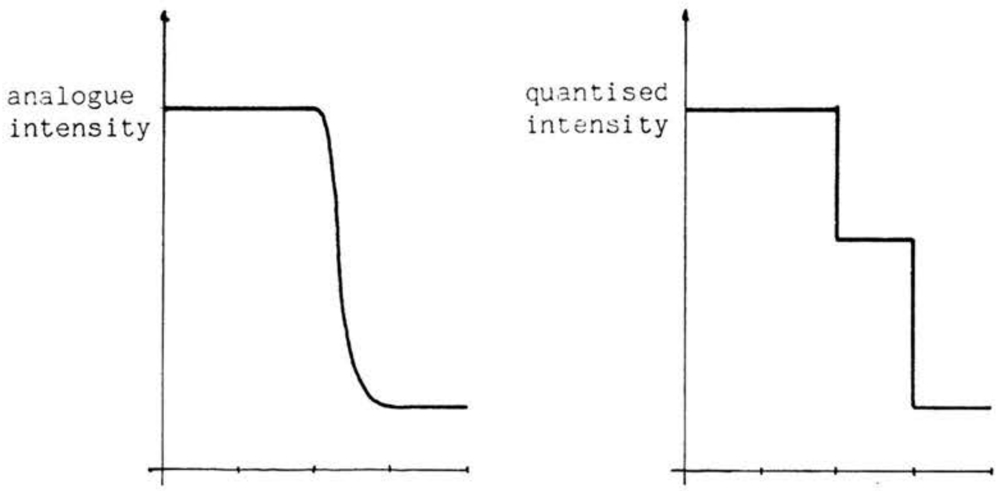


Fig. 5.1
This illustrates how mixed pixels arise when an edge is digitised.

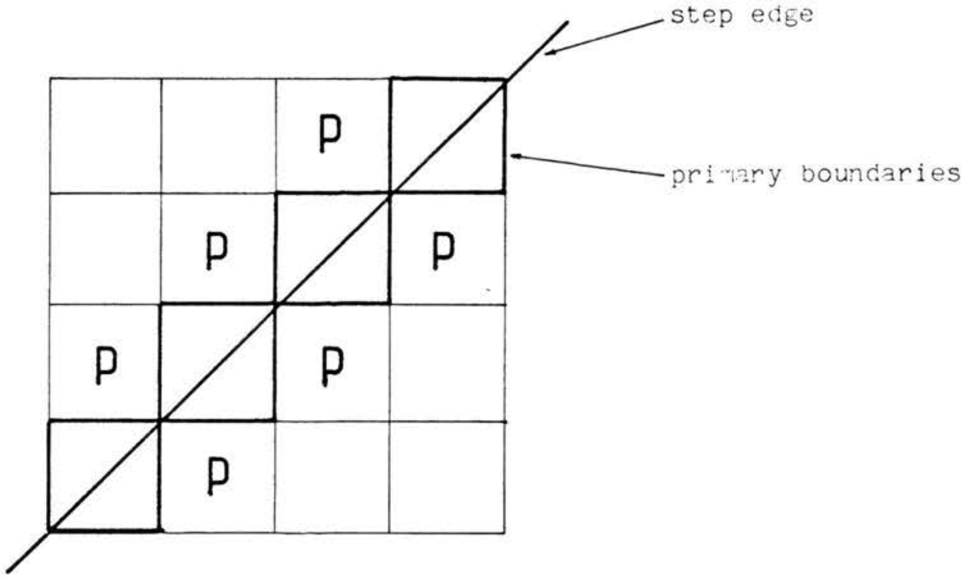


Fig. 5.2
If each mixed pixel with more than one primary boundary segment is labelled "mixed", then errors arise with diagonal edges, the pixels marked "p" above being wrongly labelled.

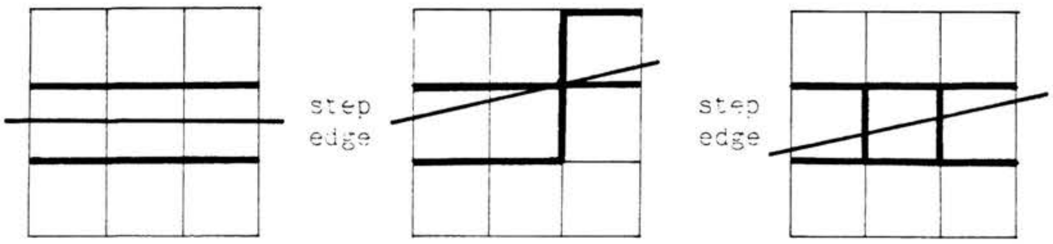


Fig. 5.3
This shows how step edges usually produce parallel pairs of primary boundary segments at mixed pixels (primary boundary segments shown emboldened).

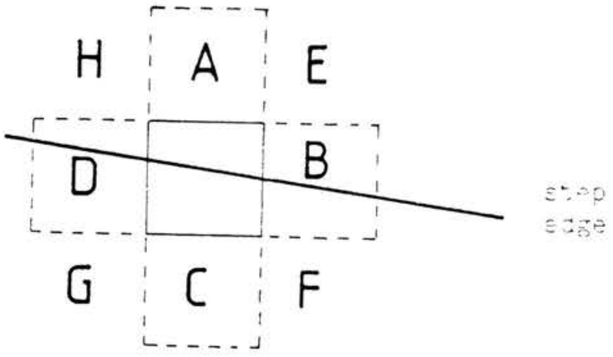


Fig. 5.4
This shows a step edge going through a pixel. {A,B,C,D} are the pixel's 4-connected neighbours, {E,F,G,H} are the pixels diagonally-connected neighbours.

The edge cannot pass through any more than two of the adjacent 4-connected pixels, [A,B,C,D]. Neither can it pass through more than two of the four diagonally-connected pixels, [E,F,G,H]. Notice that the edge can either pass through two adjacent 4-connected pixels, such as A and B, or through two opposite 4-connected pixels, such as B and D. However, it <u>cannot</u> pass through two adjacent diagonal pixels, such as E and F, but only through two opposite diagonal pixels, such as E and G. This results from the fact that edges are thin perpendicular to their direction and can be incorporated into a test as follows:

<u>The Cross Test</u>

> If a mixed pixel has a pair of opposite diagonally-connected pixels, neither of which are mixed, then label it "legal".

A step edge will always produce mixed pixels which are legal as defined above. Extended gradients and regions of texture are not guaranteed to do so. In fact, gradients, in general, won't produce legal mixed pixels because they usually produce several adjacent rows of mixed pixels, which will then fail the cross test. It is possible to imagine a textured region that would produce mixed pixels which all pass the test, but it is highly unlikely to occur.

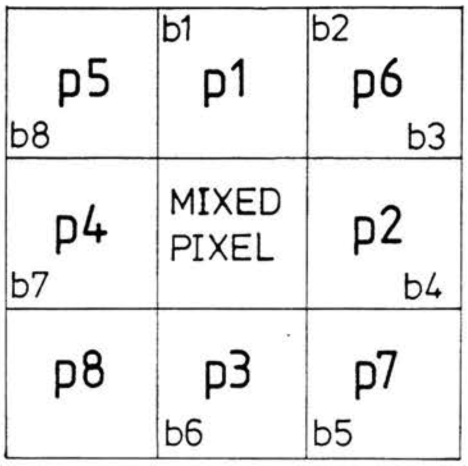
Although the analysis above has concentrated on mixed pixel production by step edges, if the edges pass between the receptive fields of adjacent pixels, then single primary boundary segments will result. Hence, in general, the digitised image of a step edge will contain legal mixed pixels (i.e. those which pass the cross test), and primary boundary segments. The latter being horizontal (between two pixels in the same column) or vertical (between two pixels in the same row). The existence of a primary boundary segment automatically implies that the pixels on either side of it are unmixed (since mixed pixels "swallow" the four adjacent boundary segments). Note that, although a straight edge was assumed in the derivation of the cross test, the method also works for curved edges. The amount of curvature allowed depends on both the pixel and greyscale resolution.

An analysis of the performance of the cross test with curved edges is given in appendix 2. It shows that for a relatively high significance threshold, as might be used in a noisy image, curved edges of high curvature are successfully dealt with. In such images, problems with noise will affect the tracker much more than problems with edge curvature. The use of a low significance threshold, as would be appropriate in a low-noise image, does lead to the cross test failing with edges of high curvature. Basically, the higher the amplitude of the edge the lower the curvature which causes problems. The highest curvature which can be handled successfully is tabulated against step amplitude for a particular case in appendix 2. If desired, mixed pixels which fail the cross test could be investigated in more detail, and those resulting from high curvature marked as points of interest.

5.1.2 Step Edge Tracking

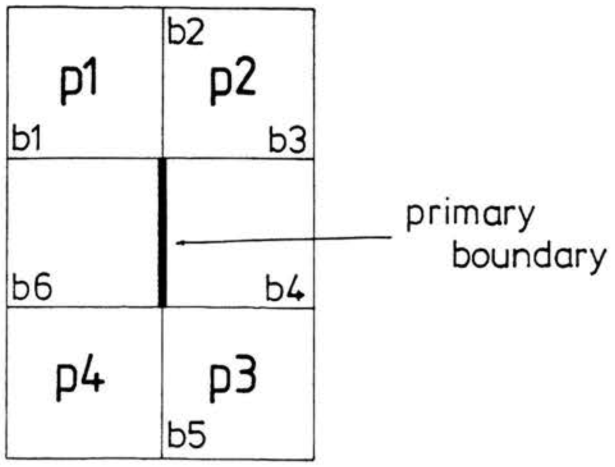
Having found the constituents of step edges, we next need to represent them in a useful manner. The ultimate goal of this system is to produce an output suitable for use by a constraint analysis program, which wants access to pixel intensities across the edge, along the length of the edge. The first requirement, therefore, is for each complete edge to be ultimately represented as a single entity. That is, merely marking the pixels and boundary segments belonging to step edges in the image array isn't enough - each step edge must be represented separately, enabling it to be handled as a single item. Since the constraint analysis program will also find it necessary to investigate intensity variations along edges, the position of each edge point within the edge must be known. This can easily be taken care of if each edge point is linked in some way to adjacent edge points. To achieve this, we return to the analysis of step edge digitisation and examine the constraints produced by the fact that the edge is continuous.

The basic approach is to classify the possible connectivity relationships for each of the three primitive edge segments, i.e. legal mixed pixels, horizontal and vertical primary boundary segments. This is done by taking each of the primitives in turn and working out what patterns of legal mixed pixels and single primary boundary segments could result in its immediate neighbourhood if that primitive was in a step edge. For example, the appropriate neighbourhood of a legal mixed pixel is shown in fig. 5.5. It consists of the eight adjacent pixels and eight adjacent boundary segments. Under the assumption that the edge is locally straight, step edges are drawn through the central pixel in all significantly different angles and positions. The resulting pattern of mixed pixels and primary boundary segments is analysed and reduced to a set of if-then rules as shown in fig. 5.5. mixed(p) is a predicate returning true if its argument is a legal mixed pixel, ses(b) returns true if b is a single primary boundary segment. lnp(x) indicates that pixel or boundary segment x is a legal next tracking position on the edge. A similar set of rules for primary boundary segments is shown in fig. 5.6.



```
1                 mixed(p1) -> lnp(p1)
2                 mixed(p2) -> lnp(p2)
3                 mixed(p3) -> lnp(p3)
4                 mixed(p4) -> lnp(p4)
5        mixed(p5) and not(mixed(p1) or mixed(p4)) -> lnp(p5)
6        mixed(p6) and not(mixed(p1) or mixed(p2)) -> lnp(p6)
7        mixed(p7) and not(mixed(p2) or mixed(p3)) -> lnp(p7)
8        mixed(p8) and not(mixed(p3) or mixed(p4)) -> lnp(p8)
9                 ses(b1) -> lnp(b1)
10                ses(b2) -> lnp(b2)
11                ses(b3) -> lnp(b3)
12                ses(b4) -> lnp(b4)
13                ses(b5) -> lnp(b5)
14                ses(b6) -> lnp(b6)
15                ses(b7) -> lnp(b7)
16                ses(b8) -> lnp(b8)
```

Fig. 5.5
This shows the 3*3 neighbourhood of a mixed pixel and the set of rules used for tracking.



```
17    mixed(p1) -> lnp(p1)
18    mixed(p2) -> lnp(p2)
19    ses(b1) -> lnp(b1)
20    ses(b2) -> lnp(b2)
21    ses(b3) -> lnp(b3)
22    mixed(p3) -> lnp(p3)
23    mixed(p4) -> lnp(p4)
24    ses(b4) -> lnp(b4)
25    ses(b5) -> lnp(b5)
26    ses(b6) -> lnp(b6)
```

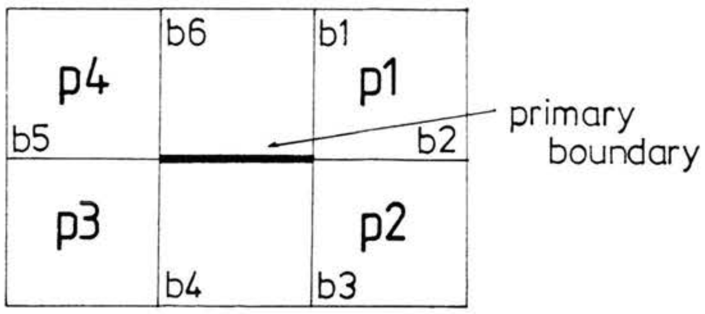
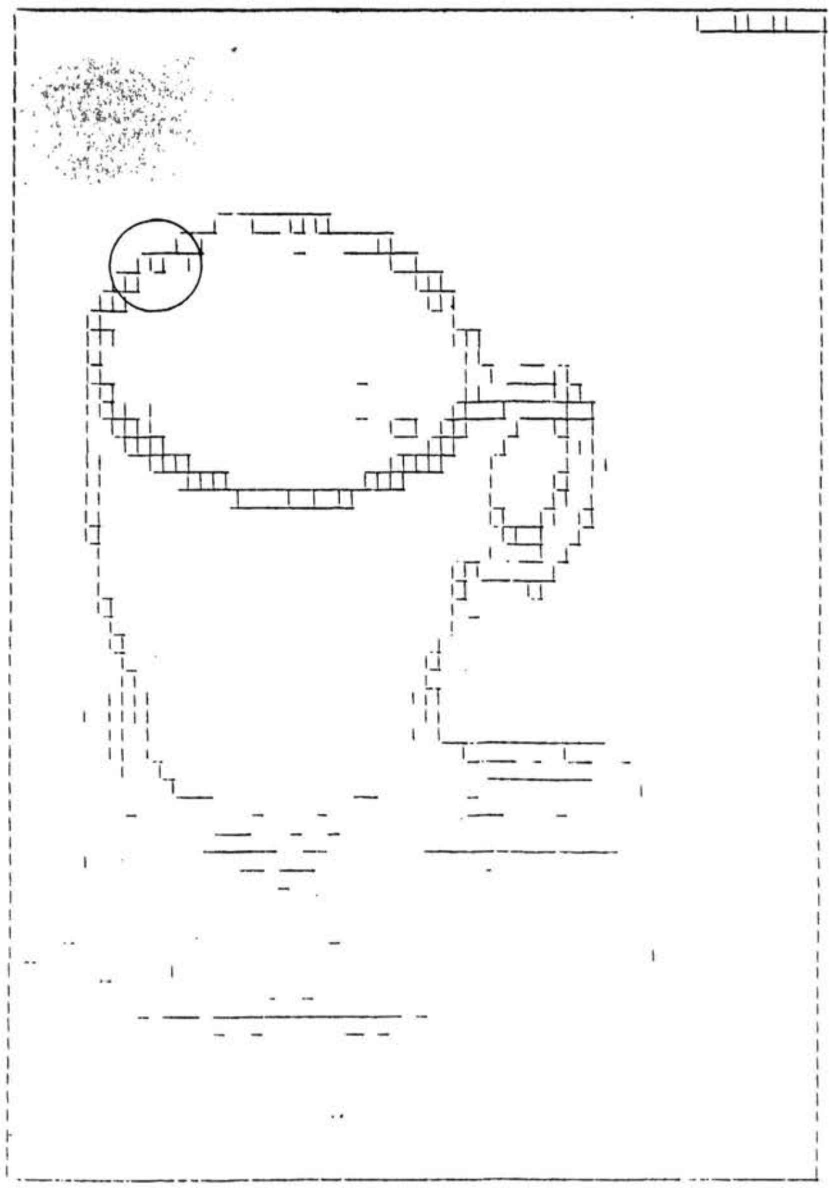


Fig. 5.6
This shows the 3*2 neighbourhood of a vertical single boundary segment, the 2*3 neighbourhood of a horizontal single boundary segment, and the corresponding set of rules for edge tracking.

Connected edges are found by tracking using these rule sets. Consider moving a pointer along the elements of a step edge. In operation, each of the rules of the appropriate set is applied at the current edge element. If the LHS of any rule succeeds then the RHS defines one adjacent pointer position on the edge currently being tracked. So the rule set in fig. 5.5 is applied whenever the pointer is at a legal mixed pixel. For example, the first rule indicates that if the pixel directly above the current one is also mixed, then it is an adjacent pointer position. When all of the rules have been applied the number of "successes" (i.e. the number of rules that fired) is examined. If the edge primitive has only one neighbour, i.e. only one rule fired, then it must be at the end of the edge, such points are called terminators. If it has two neighbours then it is a mid-line point, whereas, if it has three of four neighbours it is classified as a junction.

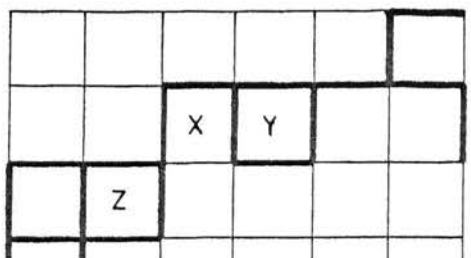
To illustrate the use of these rule sets, consider fig. 5.7(b) which shows the primary boundary segments of part of an edge in a picture of a cup (fig. 5.7(a)). Let's say that the pointer is currently at the mixed pixel marked x. Applying the rule set of fig. 5.5, rule 2 fires because of the mixed pixel on the right of x (marked y), rule 8 also fires because of the mixed pixel marked z. None of the ses predicates return true because boundary segments of mixed pixels are not considered. Now imagine moving the pointer to pixel y and applying the same rule set. In this case rule 4 fires because of pixel x and rule 11 fires because of the single boundary segment attached to the top right hand corner of pixel y. This boundary segment is considered because neither of the pixels above it or below it is mixed (not having two parallel primary boundary segments).



(a)

Fig. 5.7
This shows (above) the edge map of a picture of a cup, and (below) the primary boundary segments (emboldened) in part of the edge.

(b)



Since the rules of figs. 5.5 and 5.6 give the neighbours of any primitive edge point they can easily be used for edge tracking. The tracking algorithm scans an enlarged image array (with both pixels and boundary segments explicitly represented) in a raster fashion until it finds an interesting point, i.e. a single primary boundary segment or a legal mixed pixel. It then finds how many neighbours the point has by applying the appropriate rule set. If it has only one, then it must be at the end of an edge, i.e. a terminator. If it has three or four then it must be a junction point. If the point is a junction or a terminator then it and its adjacent edge points are put on a stack. The top of the stack is taken and used as a starting point to track an edge. As tracking proceeds used points are marked in a Boolean array overlaid on the picture array. When the end of the edge is reached (i.e. when another junction or terminator is found), if it is a junction, any unused adjacent edge points are put on top of the stack, and the edge coordinate list of the edge just tracked is stored. Now, unless the stack is empty the top starting point is taken and tracking proceeds again. When the stack is empty, the raster scan proceeds to find other unused starting points. When the raster scan is finally complete all the edges which have terminators or junction points at their ends have been found and are represented by lists of coordinates. However, any simple closed loops will not have been found since they consist entirely of mid-line points, so an extra pass through the picture has to be made to find these edges (which are very rare). In this pass as soon as an unused edge point is found it is used as a starting point and its two neighbours noted. Tracking proceeds from one of its neighbours and terminates when the other neighbour is reached. The core of the algorithm is shown below in POP2 (Burstall, Collins and Popplestone

1977).

```
until i>imax do
   1->j;
   until j>jmax do
      if interesting(picture,i,j) and unused(picture,i,j) then
         neighbours(picture,i,j)->number->start_list;
         if number=1 or number>2 then  !junction or terminator
            false->unused(i,j);   ! mark pixel used
            start_list<>stack->stack;
            until null(stack) do
               track(stack,edge_list)->stack->edge_list;
            enddo;
         close;
      close;
      j+1->j;
   enddo;
   i+1->i;
enddo;
```

The function track takes the top starting point off the stack and tracks along the edge which starts there until it reaches a terminator or junction. If it finds a junction, it puts the unused starting points on the stack.

5.1.3 Finding Step Edges - Discussion

A schematic of the complete edge finding process, as described so far, is shown in fig. 5.8. First, every boundary between adjacent pixels is examined and classified as primary if it is greater than the classification threshold. Otherwise, it is classified as secondary. Next, pixels are classified as mixed if they have two parallel primary boundary segments. Then the diagonals of mixed pixels are tested to see if they are legal or not. Both of these can be done in one pass using a raster scan. Finally, edge tracking is based on the resulting classification. The output at this stage consists of an enlarged array in which both pixels and boundaries are classified, and a list of lists, with each sublist containing a set of image coordinates representing one edge.

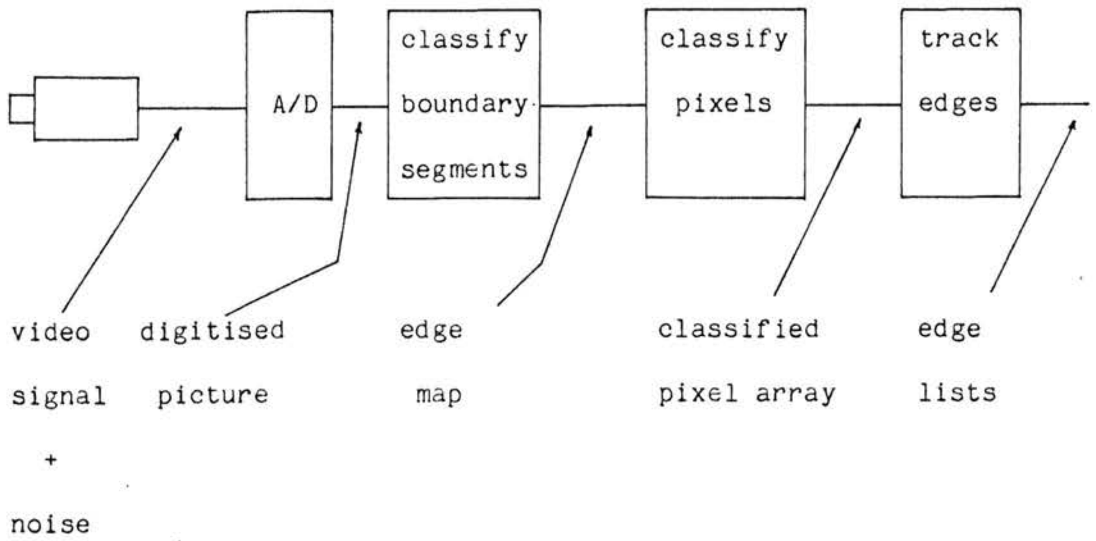


Fig. 5.8
Block diagram of the edge detection system

The initial pixel classification, the cross test, and the set of tracking rules implicitly define a step edge. Less formally, we can say that a step edge is a steep gradient in image intensity which extends for less than one pixel width, and which has intensity gradients on either side of the edge which are not significant. Clearly, this definition is rather more restrictive than is conventionally understood by the term "edge" in computer vision. This is intentional as the strict definition provides the constraints on which the method of edge detection and tracking is based. Furthermore, such strict definitions are essential if scene/image semantics are to be fully exploited. The strict edge definition is not contradictory to the earlier assertion that we need to make as few limiting assumptions about the nature of the intensities in the vicinity of the edge as possible. The definition must be based on the constraints provided by the nature of image edges, which ultimately result from the physics of the scene and the properties of the imaging transform and digitisation. It implies that there exists a set of scene/imaging entities such as obscuring edges, shadows from point sources, etc., whose resulting image characteristics are distinguishable from the characteristics of scene/imaging entities which are not in the set. The definition should encompass all edges resulting from entities within the set, regardless, for instance, of the general lighting conditions, but strictly exclude other image entities. This is only possible if the scene/image semantics have been correctly taken into account. Thus the apparent contradiction above is resolved. We need a strict definition to ensure a clean semantic relationship between the edges found by the edge detection system and their corresponding scene causes, but must make no extra assumptions which would lead to some legitimate edges being missed in certain circumstances. The discussion of the semantics of edge formation in chapter two, and the analysis of step edge digitisation provide the necessary precision for the edge detector described here. If edge detectors are not strictly defined, nor based on any rigorous examination of the relationship between scene and image, the result of applying them is an output which is not clearly related to the scene and is thus difficult to use as a basis for further analysis.

Our initial concentration on step-like edges is not to imply that blurred edges are considered unimportant. That would be contradictory to the goals of edge detection. Blurred edges will be found by applying the same methods as described in this chapter to reduced resolution versions of the image, as described in chapter 6.

5.1.4 Edge Tracking Examples

To verify that the methods worked as expected, a test image of a white figure on a dark background was used. Photographs of a monitor screen showing the original picture and of a line drawing display showing the tracked edges are shown in fig. 5.9. The input was digitised to 64*64 pixels with 256 greylevels. In this section, we have chosen particular thresholds to illustrate various points, rather than use the automatic threshold selection method described in chapter four. Threshold selection is not critical in this first example. The resulting primary boundaries are shown in fig. 5.10. The pixel classification is shown in fig. 5.11. Both pixels and boundary segments exist separately in the data structure, but, for convenience, only pixels are used to show the classification. The boundary segment information is included in the figure using the following legend:

N		single primary boundary on the north, south,
S		east, or west side of the pixel.
E		
W		
M		legal mixed pixel
X		illegal mixed pixel
1	┐	
2	┘	pixel with two primary boundary segments
3	┌	adjacent in the positions shown (where the
4	└	lines indicate primary boundary segments).

If a pixel is classified as mixed, then none of its four surrounding boundary segments are considered in the tracking process. This is implicit in the tracking rule sets, based on the observation that an edge cannot both go through a pixel, and go between it and an

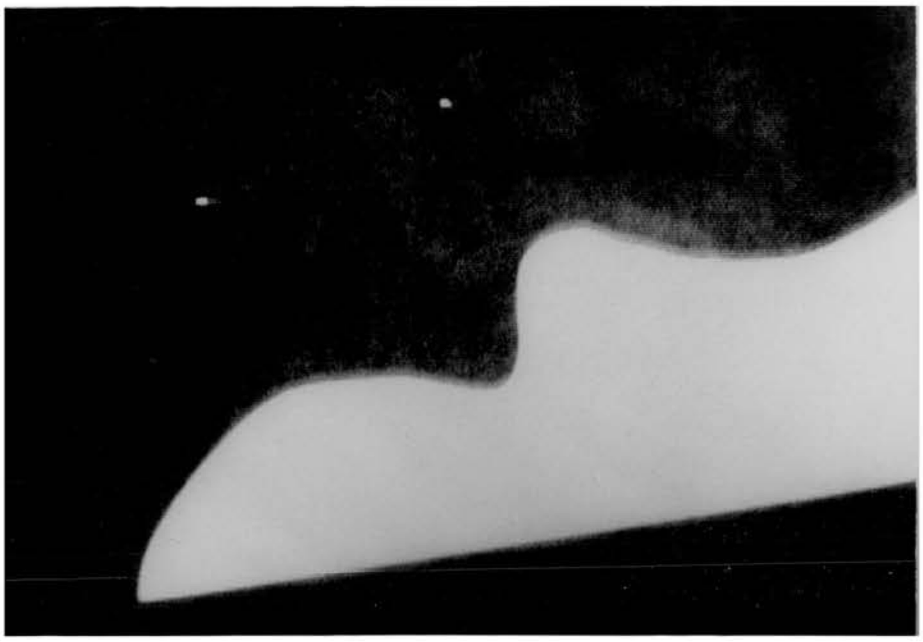
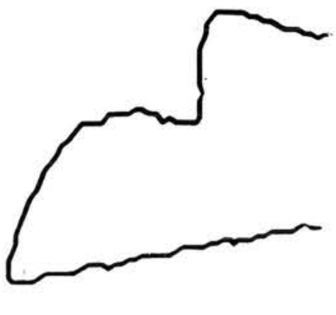


Fig. 5.9
The top photograph shows the test image of a white figure on a black background. The lower photograph shows the result of edge tracking (note that the scale and aspect ratio of the line-drawing output device are different from those of the monitor).



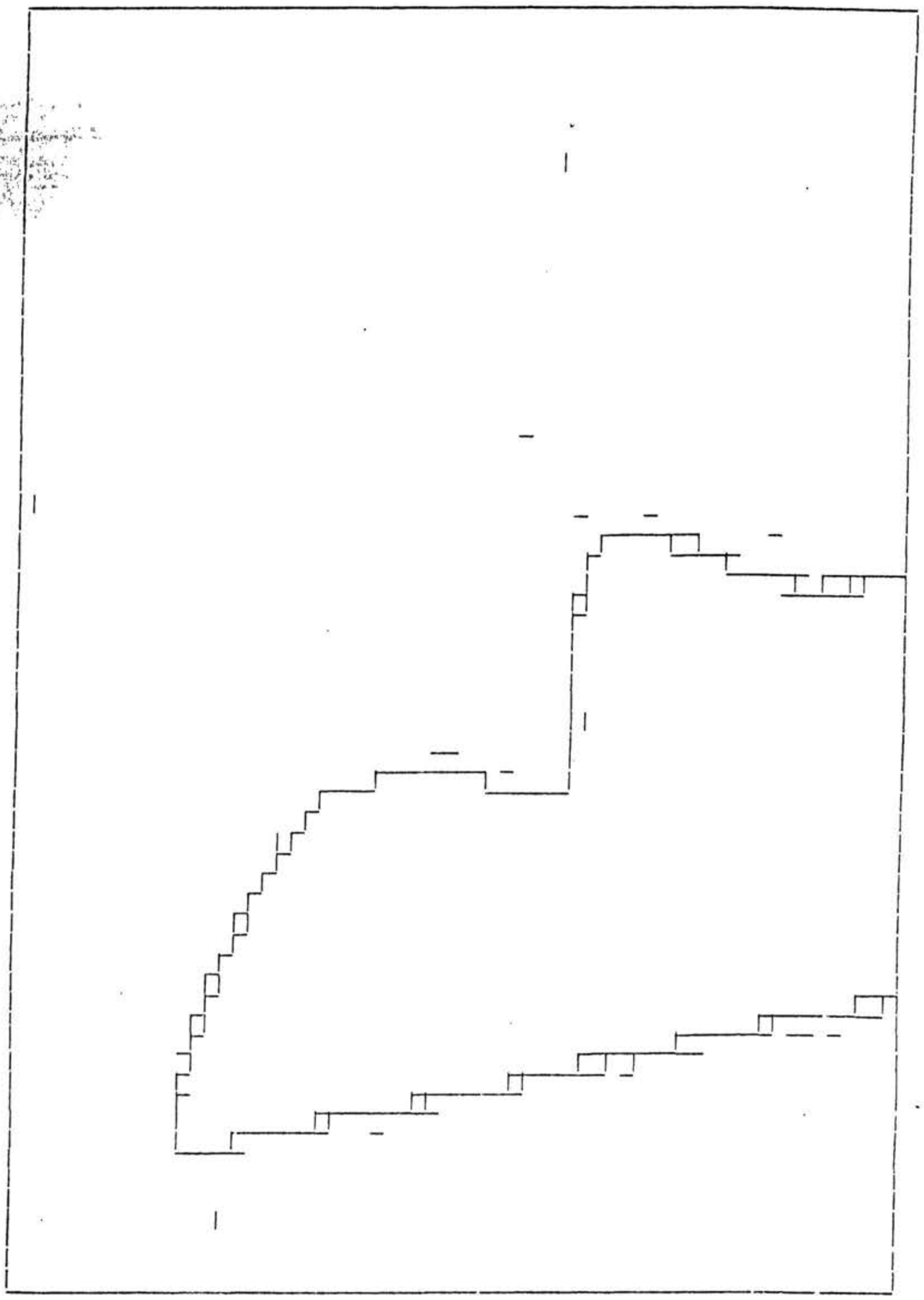


Fig. 5.10
This shows the primary boundary segments resulting from the test image. In some cases the boundary passes between pixels, in others it goes through them.

```
                                      EW









                                   S
                                   N

W                                      S
                                       N SSSSS      S
                                        23NNN1MM4S  N
                                       E3       N1MSSS     SS
                                       2W          NNNMM2MMM3N
                                       M                N
                                      E3
                                      EW
                                      EW
                                      EW
                                      EW
                                       M
                                      EW
                         SSSSMMSS     EW
                     SSS23NNN  N14MSSS2W
                    23NNN        N NNNN
                   23
                  M3
                 23
                23
               23
               M
              23
             23
             M
            23
           M
          E3
          2W
          M
          E3                                  S  S 2MM
                                                 SSS2M3MMNMN
                                          SSSS2MMNNNN
                                    SSS2MM313NN
                              SSSS2M3NNN
                      SSS S2MMNNNNN
          EW    SSSS2M3NNMNN
          E4SS2MNNNNN
           NNNN
```

Fig. 5.11
This shows the pixel and pixel boundary classification for the test image. M indicates a mixed pixel. N,S,E,W indicate pixels with a single primary boundary segment on the north, south, east, or west side respectively. 1,2,3,4 are pixels with two adjacent primary boundaries, as follows:

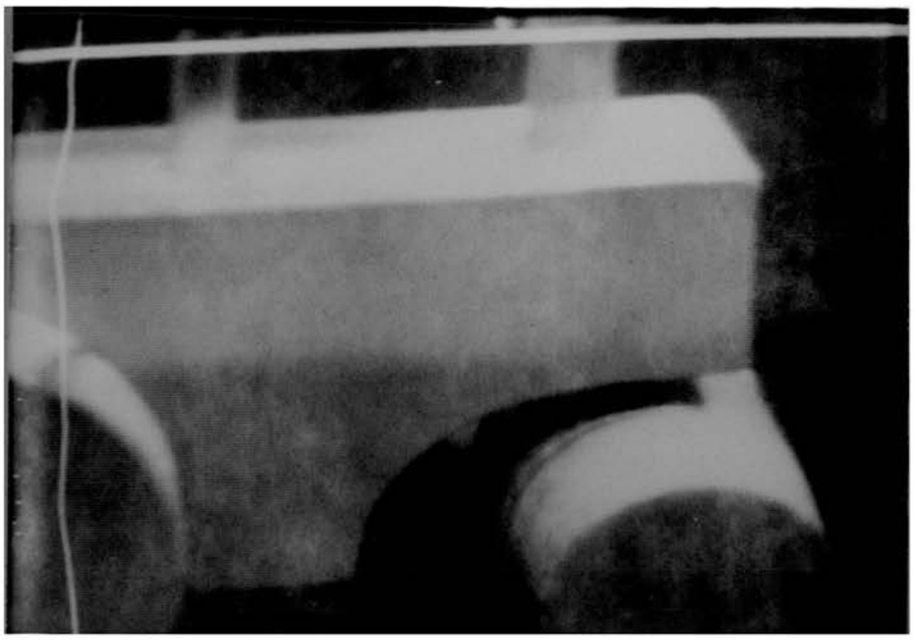
1 - ┐   2 - ┘   3 - ┌   4 - └

adjacent pixel. Since illegal mixed pixels are typically generated by extended gradients or by texture it is not surprising that there are none in this test image.

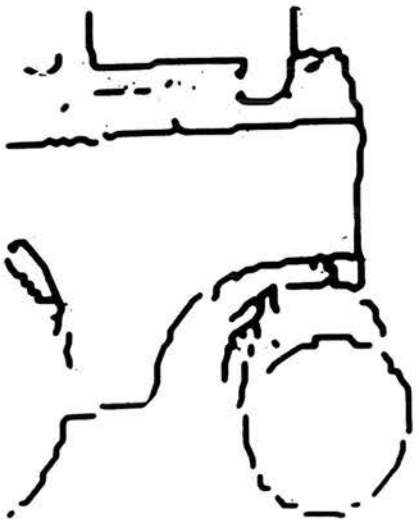
Finally, the edges are tracked based on the pixel/boundary classification. The tracker produces a list of edges, each edge being represented by a list of coordinate pairs. Each pair denotes either a pixel or a boundary segment. The results of tracking for the test image produce the expected edges as shown in the lower photograph of fig. 5.9.

A more complicated image produced by a toy steamroller and the resulting tracked edges are shown in fig. 5.12. The classification, which includes some illegal mixed pixels, is shown in fig. 5.13. Notice how several illegal mixed pixels are caused by the combination of the shadow edge and the variation in radiance due to the shape of the front roller. Also interesting is the break on the curved front edge of the body in the output image. Close examination of the input picture shows this is due to a part of the front roller of about the same intensity as the body which gradually merges with the background because of its curvature. In cases such as this, the result of our acceptance of the preeminence of image veridicality is that the output of the system should accurately reflect the structure of the input image, even if, as in this case, it leads to shorter tracked edges.

The effect of varying the threshold on this image is shown in fig. 5.14. With a very high threshold of 30 greylevels, it can be seen that several lengthy edges are still found. These edges are mainly obscuring edges (the top, front, and bottom of the body, and the inner edge of the rear wheel) and sharp surface orientation edges



(a)

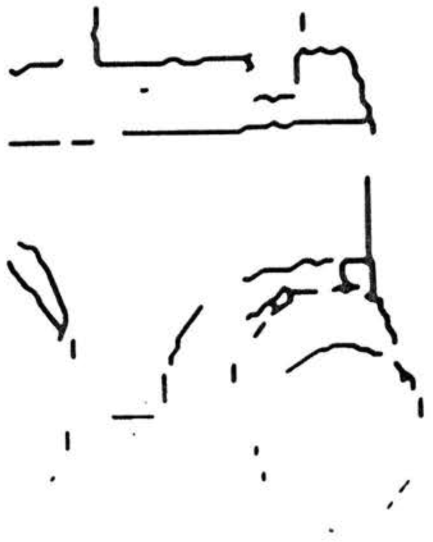


(b)

Fig. 5.12
(a) shows the input, which is a picture of a toy steam engine (64*64 pixels).
(b) shows the edges tracked with a significance threshold of 20 greylevels (d=20).

```
               M                      NEW     EW
               M                       EW     EW
               M                              EW
               M                       EW     EW
               M                              EW
          EW   EW                             24SM42MM4
      S   2W M E4SSSSSSSSMMMMMMMMMMMMMMM      M3N NN1MM
  MMMNMMMN      NNNNNNNNN             2M      M EW   1M
                           S          1W      M       M4
  S S  S S      SSSSS SSS E3                 EW       1M
  N N  N N      NNNNNN NNN      EW S 24     2M4 S      M
           2W  EW                  N NMMMMMN N N      M4
           N              EW                          1M
                           E4                 SSSSSSSS2M
                 S SSSSSSSSSMSSSSSSS2MMMMMMMMM3NNNNNNNNM
  SSSSS2M2M4MMSSNMNNNNNNNNNN NNNNNNNN                   1W
  NNNNNN N N   NN                                       EW
                                                        M
                                                        M
                                                        M
                                                        M
                                                       EW
                                                       EW
                                                       EW
                                                       EW
                                                       EW
                                                       EW
 SMM4                                                  EW
 13 M4                                                 EW
 4  1M4                                          S SSS24
 M4  1M                                 SSS2MMMMMM1M3NNMM
 1MM  14                            S2MMNNNN    M MM  1M
  1MM  M                         2M13          2X 2M4  M4
   1MM414                       E3       2MXMMMMMM3MMM2MX
    1MMMM                    2M 2W    2M3MXX3         1XX
     1X4EW                  23  N    2M3MXM3           MM4
      1MM                  23      2M32M3EW             1M
       1M                 23      E3 2M3                 M4
       E3EW              E3        2M3EW                 1M
         EW              2W       E3EWEW S      MMMMMM4   M
         M              23        MM EWEW1W 2MMM      MMMXX4
         EW             M         M  EW   2M3           1XMX
         EW           E3         EW EW   M3              1XM4
                      EW         EW EWEWM3                1MM
                      EW         EW EW  S                   E4
                      M            M  N                      1W
              SSSSSSS2W            M EW                      EW
         SSSSSNNNNNNNN            EW                         EW
        E3NNNN                     EW                        EW
        EW                         EW                        EW
        2W                         EW                        EW
      E3                           E4                      EW
      2W                            M                      EW
     23                             1W                     M
    E3                              EW                    2W
    2W                               EW M               23
   23                                E4S               23
 S23                                  NNS             23
SNN                                     14SS       SS N
3                                        NNNSSSSSSSNN
                                             NNNNNNN
```

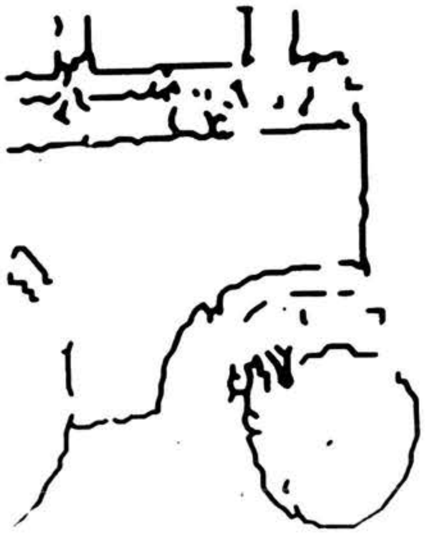
Fig. 5.13
This shows the pixel classification of the steam engine picture with a threshold of 20 greylevels. Notice the illegal (for tracking) mixed pixels, marked X, which result from a shadow on the front roller producing an extended gradient.



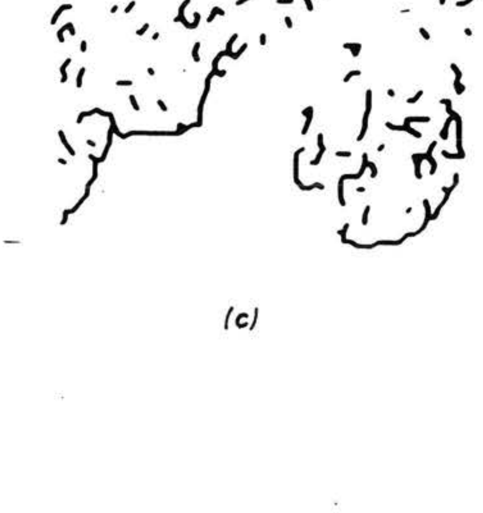
(a)

Fig. 5.14
These show the effect of varying the threshold on edge tracking. The pixel noise was estimated at 5 greylevels standard deviation. The thresholds used were:

a) d=30
b) d=12
c) d=5



(b)



(c)

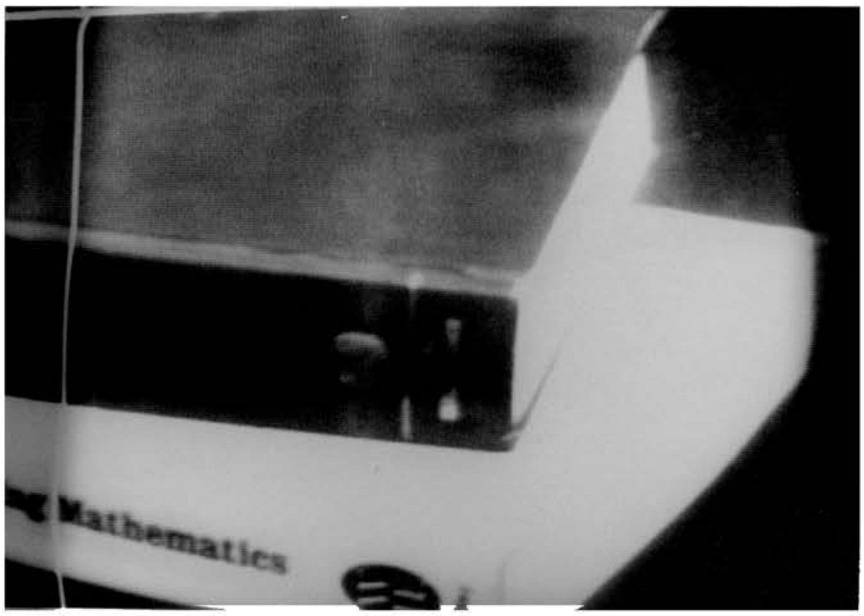
(on the top of the body and on the front wheel). It may be useful to first use a high threshold when interpreting an image, then when some attempt has been made to classify the edges, go on to using lower thresholds to get more detail. The precise representation of the edges is useful in this respect: if an edge segment is found at a high threshold, it is also going to be found in the same place at a lower threshold. Suppose an extended shadow was coincident with a sharp reflectance edge, then the reflectance edge could be found with a high threshold, and the shadow at a low threshold. If only a low threshold was used the region of illegal mixed pixels due to the shadow could lead to the reflectance edge being missed.

The effect of reducing the threshold too much is shown in fig. 5.14(c), and its accompanying classification in fig. 5.15. Most of the edges are too short to be of any use for constraint analysis. Such a classification could be useful for defining areas of gradient, i.e. areas of the image where surface shape or incident light variations extend over a substantial 2-d area. By taking the difference of two classifications made using different thresholds, all the gradients of a particular magnitude could be found.

The image and edges of fig. 5.16 and the accompanying classification in fig. 5.17 illustrate that the edge finder also finds thin bars successfully. Both the bottom cover of the top book and a thin line of print on the face of the bottom book (faint in the photograph) are successfully found. This is not surprising since bars of a certain width also satisfy the two basic constraints of thinness and continuity used in the derivation of the edge detector. It would be a trivial matter to modify the edge detector to distinguish between edges and bars using a modified cross test. If the intensity

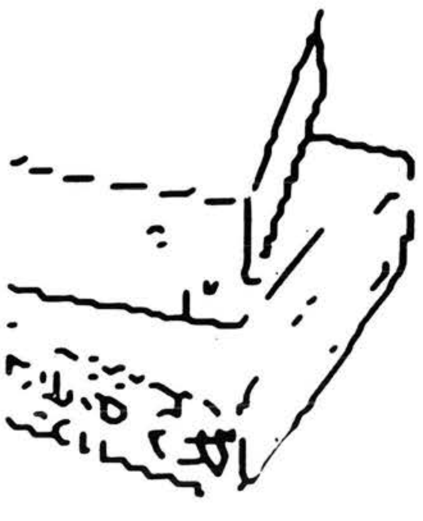
```
  S      XXXMXM       N     E4          1M31MM    M        NN          S
  N S    XX    M      S          N S   M   S M4 MX MM3EW          M   M   E3
 S   N   MXM S MX   E3SEW       MM3   M   N MXMM3 2X
 3 2M      X N2M   M   N      S          1M     MX     2XMXM                 EW
   M3     MX4 XM4   M EW      N                X 2WE3 XXM4                        S M
         M4MXXX3     EW M   242X          2XX3S2XXX4XXMXMX4  EW  S   1W
         23XXXX4S  S S2MMXMXXXXMXMXXXXMX41XXXXXXXX1XMXMM      14
MMMMMMMMXMXXMMNM3M3N1X1X3XXXX3XXXXXX4M  2XXX32XX  1XXX2W       1W
X 1X3    X1X413M2X2X  EWXX42XXX  2M3XXXX3  MXXXXXXXX   XXX1W          M
M4 X   2XXXXM4MXMXMMMMXMXMXXXX4XM   1XX  23XXXXXXXX    1XXX
MMXMXXX1XXX41M1X3  XX3  XXX3X  NXX4  2XXXX  XXXXX2XX2W  XXX
 1WXX MXM32XM3E3S       2XXX  14  MMMXXXXXXXXX3XMXM  NS  XXXX
MMM3M231M2X4X  24N    S231XX    1414XXXXXXXXXX    X3  EW1XXXX4E4M  E4
X      13   M3  X3  1M  2M3NEWXXX42XXXXXXXXXXX242M42M42XXXMXE3       N
1MXM   X4X   MXMX42MX4242MXMMMMXXXXXXXXXXXMXMM3NMMNN1X  X32W
SX2XXMMXMMMX2XMMMM1MM3N                 XXXX    XX              E3  M  N    M
N1XMX            X                   EW     M3                         2M4
   X                                                                    XM3
       M       E4                    M               S          2W          1M
                  N                        S               E41WE4        MEW        MXM
    EW       S                  2W  N          S           M4    M                   2M    2W
SSS         N S       S          N              NS        2M3                 EW     XM    N
NNN         EWN      1W                         N        N       M          S          2MX  S  24
  E4                M                      S                  E4             NM          1M     N  NN
4 E3    S          M                   S  N                      1M                          M      2W
X4       1W     E3                 E3             M4         EW      S                        M       N
XXM          2M4    S                                1W                N       S  S  EW   EWEW   EW   S   2W
XXM4         M  N  E3          M              S              S                N  NSS       S  M  EW  N  N
X3MMMM4  M     2WEW                           N              N          SSE4       N14      N24EW  S     S  E
X    1X41W2W   N                              MM     2W             NN2XX4        XMS2MX24E3   MN
X4    1M4   N                       EW               N       M4MS   2XXXXXXXXX31XXX3
XMM42XM       M             EW          S       S  EW   2MXM3MNXXXXXXXXX     XXX      M        2W
31XXX1X          S                 M           N  2M3   2XMMM314    XXXXXXXXX4   XXXX   M   MMN
 MMXMXX42W23             M3S     M4  N     2X3XX3     MXXXXXXXXXXXXMXXXX                  M
M    1XXXX3   N            SM  1W2WN     2XX2X3E4   2XXXXXXXXXX        1XXX4  M
W2M  1XXX            M       N1MM  N  EW2XXXX3    MX3XXXXXX3E3              1XMX         E4E
E3EW  1XX4        M  2W                    2M3XXM      2XXXXXXXXX4         S          1XMXX       N
 M    MXXXXM           NS          2W     2M3231M   2XXXXXXXXXMMM          N            1XXXX
     MXXXXX            N          M     2MM23           XXXXXXXXX      EW             X         XXX         S
  M     1XXXX          EW      M  1W   M3  N             XXXXXXX41X42XXXXXXXXXX1X4E4   N
M3  M  1XXX        E4                 2M2W              XXXXXX41X4XMMXXXXXXXXXMXXX    1W
2WEW    XX4SS  SNS                 M3N                 XXXXXX1XXXMX3           X          XXXXXM4
N  EWS  1X3NN  N  N                M                   XXXXXMXXXXX     EW  14       MXXX4E3
     N  XM     M       S          EW23                XXX41XXXX3        EW        N MM3  XMM
       S  X     E4      14      S  M                  XXXXSMX3            2W           2X          1M4
    MM1MM        N        N      1XM                  XXX31M        S       M      S     MMMM     M1W   S
  EWX      24  2X4          M2MM                      XXX423  EWN  EWM      1MMMS         23EW   N
M     X3     MMSMMMMMMMM3                             1XXXM  EW           23           S  1M  S1WEW  S
E4    M4  M3N                                          XXXS  E4         N     2W  N         23S  EW  N
W  14   SN23                                           1XMNXE3  MMM  M4N  S      MMM  1MMM
 MSN  1MM                                              1MXMM              MM      N      EW  M     X3
  N     2MX                                             M              SS232WS  M  E4EW      MX
 EW     XM3                                             M    S     E3NN  N  N           N    S2M
M     M  MX                                             1MM1M  EW  MM                       1M3
    2M3                                                   XX  X4E4        EW              E4  23
   2M3                                                    XMXMX4M     2W             2W  M23
  MM3                                                     1X4XXX         M                N     MM
 2M3                                                        1XXXX2W   M                  23     S  S
2M3                                                          1XXMM                   2MM3     E3   N
3                                                                 1MM4SSMMMM3
                                                                        NNN  X                EW   SS
                                                                              M                   NN
```

Fig. 5.15
This shows the classification resulting from a low threshold. Shadows, intensity gradients, and noise, produce a large number of illegal mixed pixels.



(a)

Fig. 5.16
The input picture, in (a), consists of two books (64*64 pixels). The tracked edges (d=20) are shown in (b). The bottom cover of the top book, which is effectively a bar in the image, is tracked successfully.



(b)

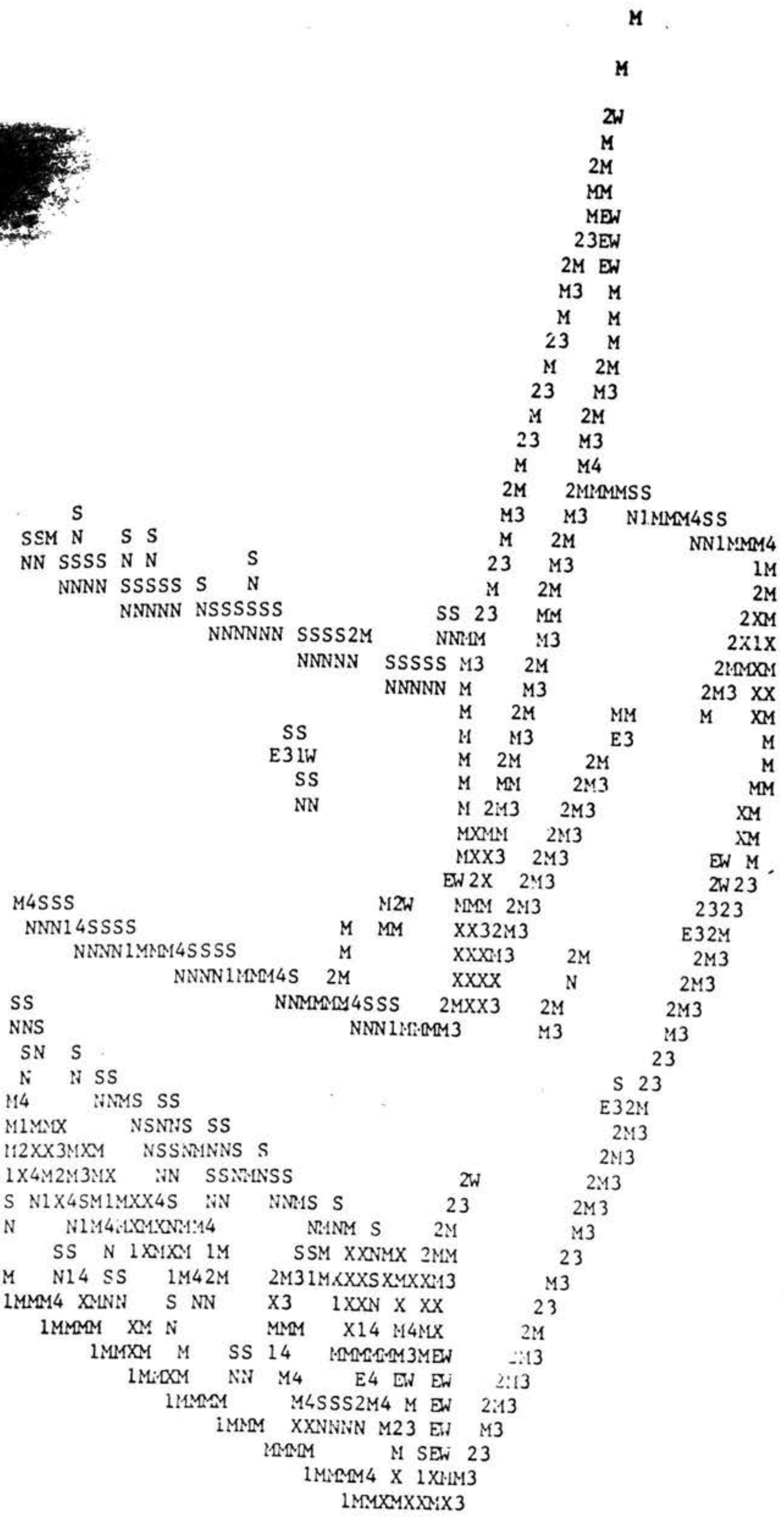
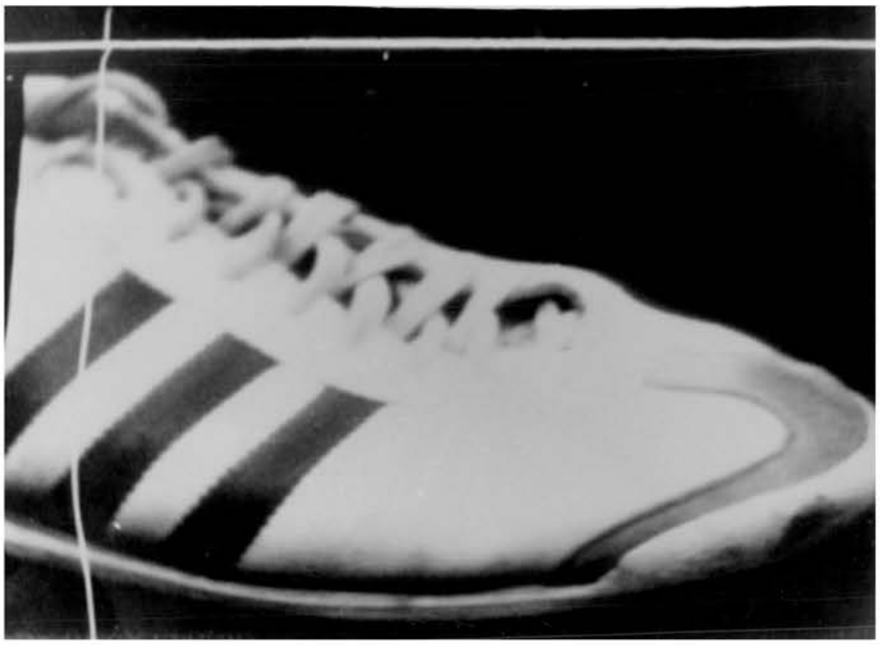


Fig. 5.17
This shows the classification of the books picture. The lettering on the spine of the lower book produces illegal mixed pixels because the edges are too close together.

of the unmixed diagonal pixels were both greater than or less than the intensity of the central mixed pixel then a bar would be indicated. For a normal step-like edge the mixed pixel intensity is between the intensities of those on either side. Notice how the lettering on the spine of the bottom book gives rise to illegal mixed pixels. Ideally, we would like that region to be ultimately labelled as texture.

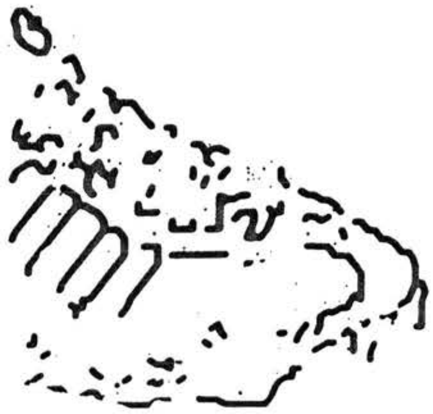
Finally, another example illustrating how too much detail and shading give rise to large areas of illegal mixed pixels is shown in fig. 5.19 which is the classification of the sports shoe picture. Figs. 5.18(a) and (b) show the original picture of the shoe and the resulting tracked edges. At this resolution the shoelaces are just a jumble of edges and so produce a large region of mainly illegal mixed pixels. Similarly, the shading towards the sole of the shoe produces further areas containing many illegal mixed pixels.

The process detailed above satisfies the specific goals relating to step edges. To some extent we have also satisfied our first goal of edge detection which was to find all the significant intensity variations. However, it is desirable to do rather more than just identify those places in the image where the intensities are changing quickly. This is reflected in the second goal which exhorts us to relate image events to their scene causes, if possible. From fig. 5.19, we see that in a complex image, many of the mixed pixels are not in step-like edges and so are labelled illegal. In the next section we investigate the causes of these illegal mixed pixels and try to identify methods for analysing the corresponding image regions further.



(a)

Fig. 5.18
The input picture of a sports shoe (64*64 pixels) is shown in (a). The tracked edges shown in (b) were obtained with a threshold of 20 greylevels.



(b)

```
MM4
4 MM
M MX4
MXM 1W
1XM 24
  1MMX4 S
S  1XXXM14
1XX4MXXX M
XXXX32XX 24
XXX14XM1MXXX4S
XXXXXXX4MXXMX1M4
XXX31XXXM 1XX2MMMM4
XX   X32X 2X3X2313MX4
MX EWMXXXXXXXXX  2XMX4
X 2W2XXXXXXXXXM 2X XXXXM4
MX32MMXXXXXXX414XXXXXX MX4
XMXMS 1XX X2MM XXXXXX  XXXMMM4S
 XXXN   1XM3 XXXXXXXXXXXX3 1MN1MX X4
2MXM42M 2MM2M1XXXXXXXX3XX  2M42XXXXXXX4
M3 1MXX N13X141XXX2X31XM3 MXX3XMXXXXXXXM
3    XXX4 E3 MXXXXXX42XX 2MXMXMX  X1XX1MX4
    2MXMMME4 1XXXXX3MXXXXXX231X  2XXMX2XMX4
    M3  1M41W   1X3E3XXXXXXX4 M42MMXX3N2X31MM4S
  2M3   2MM4    MM 2XXXXXXXXXM3NX32X42M3  M NNMX4
 2M3   2M3 1M4EW 1MMMMXX3 XX 1M 2MXM3MMM   M4 SXMX4
2M3   2M3   1MX4     1M  XX4 MXMM1M M3   2M1M31XXMM4
M3   2M3    MMM4      MMMMMMMM  2X M   2XM    M3 SMM4
3   2M3   2M3 1MXXX       1XX3  MMX3   XXX   2M  N  MM4
   2M3   2M3   MXXMMM      XX          MXMMMMX        14S
  2M3   2M3   2M    MXMMMMMMMMM      M      1MMM4     N14
X M3   2M3   2M3   2M3          MMMM           1M4      M
XXM   2M3   2M3   2M3                           1M      M4
XX3  2M3   2M3   2M3                            M       MX
X3   M3   2M3   2M3                            2M      2XM
3   2X4X4MM3    M3                            2MM     2M M
    2X4XXXM3   23        2W            S    S2MM     2M3 M
    X4XXXX3   23        E3             NMXXM3N     2M3M 2M
   2XXXXX3   2X4  XX       2M         X XXXX   S 2XXX4XXXMM
  2XXXXX3    X4XXXMXMXXX X  1M4  24XXMMXMXX3  23MXXXXXXXXX3
  MXXXXX    2XXXXXX XXXXXXXMS MXXXXXXXX31W  2MMX4XXXXX  XX3
    XXX3   2XXXXXXX  XXXXXXXN XXXXXXXXM  2M3 XXXMXX3
4          XXXXXXX M XXXXXXX XXXXXXX3  2XXX2XMXXX2M
XX4        XXXXXXXS X X XXXXXXXXXXX3  2MMXMX3X3  N
XSMMX4   E4MXXXXXMNSMSMSMXXXXXXXXX3  2MM  X S   S
 NMX2MMM4 XM       N N N            2M3 SS  N   N
   N  2MMX3X4       S    SSSS      2M   NN
      N  1XMMMMMMMMNMM   NNNNMMXMMMMMS
                 X         S  XXX    N
                          SN
                          N
```

Fig. 5.19
In the classification of the sports shoe picture, the laces produce interfering edges and hence illegal mixed pixels. The intensity gradients near the sole of the shoe also produce illegal mixed pixels.

5.2 Dealing with Gradients and Texture

The edge detection system described in section 5.1 finds and represents in a useful way those intensity variations which satisfy the two constraints of thinness and continuity. As an intentional side effect, it also marks those pixels in the image where there are other significant intensity variations. These illegal (for edge tracking) mixed pixels have failed the cross test and so must result from significant intensity variations which extend in more than one direction. In this section, we first consider the origin of these illegal mixed pixels, then try to delimit their extent and classify them further.

5.2.1 The Generation of Extended Gradients and Texture

By considering the scene and image-forming process, we see that illegal mixed pixels can be formed in two ways. Firstly, gradual changes in surface orientation, surface illumination, surface reflectance, or some combination of these at the appropriate scale, relative to the spatial and greyscale resolution of the imaging device, can produce extensive intensity variations of sufficient amplitude to be significant. Normally, the region of significant intensity variation will correspond to all or part of a single surface in the scene. The key point about variations arising in this way is that they are smooth.

Secondly, extensive image intensity variations can be caused by "texture". However, texture is difficult to define for scene analysis. According to the Concise Oxford Dictionary texture is

> "arrangement of threads etc. in textile fabric, characteristic feel due to this; arrangement of small constituent parts, perceived structure, (of skin, rock, soil, organic tissue, literary work, etc.); representation of

structure and detail of objects in art; (Mus.) quality of sound formed by combining parts."

For computer vision, a rather more precise definition of texture is needed (one will be given later), but we can begin from the above by considering texture as "perceived structure". It is important to note that texture is not simply structure but <u>perceived</u> structure. It is clear that texture cannot be defined solely in terms of a world model. The optical system, image resolution, relative size and distance of objects in the scene all affect whether an object is seen as an individual item or as an element in a texture field. However, some world considerations may be useful in coming to an understanding of texture. Various scene arrangements can constitute texture for a human observer. Sometimes it is a large number of similar objects close together at a relatively large distance, e.g. the leaves of a tree. Alternatively, the perception of texture can be caused by multiple variations in surface reflectance properties, e.g. wood grain. Also, a textured appearance may be caused by multiple surface orientation variations viewed at an appropriate distance, e.g. the bark of a tree. There are two main factors here - surface orientation (and range) variations, and surface reflectance variations. When there are many of these close together such that we cannot clearly perceive the individual variations, we call that texture.

What is being suggested is that the perception of texture occurs when the perceiver is unable to fully "process" that part of the visual field to a stage where the individual surface orientation/reflectance changes can be unambiguously interpreted as scene events because of insufficient information. For a computer vision system, this implies that a definition of texture must rest, not only on the image effects, but also on a model of the early pro-

In fact, most of the work done with texture in computer vision has been concerned with the problem of discriminating between different textures (Bajcsy (1973), Haralick et al (1973), Maleson et al (1977), Schatz (1977), Laws (1980)). According to an authoritative survey of this work (Haralick (1979))

> "...a formal approach or precise definition of texture does not exist. The texture discrimination techniques are, for the most part, ad hoc."

Usually textures are classified as either micro-textures or macro-textures. Briefly, micro-textures are image fields containing many intensity changes with no obvious geometrical relationship to them. We only consider micro-textures in this thesis, and only they satisfy the definition given above. Macro-textures, such as the pattern of paving stones that precedes you as you walk along a pavement, are not textures in the same sense.

A summary of the argument so far is:

1. we previously devised a way of finding and tracking step edges which also labelled as illegal (for tracking) other significant intensity variations.
2. these illegal intensity variations can only be due to two types of scene events:
    - a. smooth changes of reflectance, surface orientation, illumination on a single surface producing a smoothly varying region of image intensity.
    - b. sharp changes (relative to the sampling frequency) of reflectance, range, or illumination on single or multiple surfaces such that a textured region results.

cessing. When there is not enough information, i.e. not high enough resolution, for individual intensity changes to be distinguished and processed normally, then the region containing these changes can be called "texture" and other methods applied.

The early processing system described in chapter two is based on finding step edges and interpreting them by analysing the intensity variations across and along the edges. This requires that each edge is distinguishable by the edge finding and tracking process. If the edges are too close together they will interact and fail the cross test. As mixed pixels which fail the cross test are marked illegal, regions of texture in images will consist of many illegal mixed pixels. An attempt can now be made at a definition of texture, as follows:

> When a number of scene/imaging events, such as obscuring boundaries, surface orientation boundaries, or reflectance boundaries, are close enough together to produce a set of intensity changes in the image which cannot be distinguished by the edge detection and tracking mechanism being used by the perceiver concerned, then the image region enclosing that set of intensity changes is said to be textured.

To achieve the necessary precision, this computational definition of texture is closely coupled to the rest of the early processing system. However, the general idea of defining texture to be those regions of the image where the intensity changes are too close together to be separably investigated is more general. Thus, a comparable definition of texture could be devised for any similar early processing system.

We now consider how we can find these regions and label them appropriately as texture or gradient.

5.2.2 Finding Gradient and Texture Regions

The process of classifying illegal mixed pixels as gradient or texture takes place in two stages:

1. group the illegal mixed pixels into illegal regions.
2. classify each region as gradient or texture.

The definition of an illegal mixed pixel as being one in a part of the image where there is extensive intensity variation over a substantial area of the image makes it unlikely that they occur in isolation, so it is a reasonable first step to attempt to group nearby illegal mixed pixels together. Then, the assumption that each illegal region (a connected group of illegal mixed pixels) arises either because of gradient or texture can be made. Of course, it can happen that the two can be superimposed, in which case it is desirable that the region should be labelled as texture because texture-based analytic methods will need to be applied to recover further information about the scene. Also, the situation can arise where adjacent regions of gradient or texture give rise to a single illegal region, but no attempt is made to consider that eventuality at present.

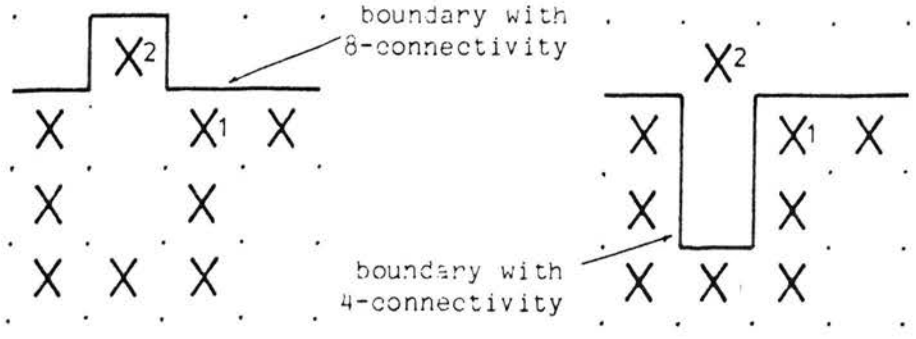
So, illegal mixed pixels are usually found in groups because of the way they arise. However, these groups are not always completely homogeneous, occasionally containing legal mixed pixels or even pixels without a significant boundary. This can be seen in the initial classification of the sports shoe picture (fig. 5.19) which is reproduced overleaf as fig. 5.20. The illegal mixed pixels on the top of the shoe result from the large number of step-like edges produced by

```
[illegible]
[illegible]XM
[illegible]X4
MXM 1W
1XM 24
  1MMX4 S
S  1XXXM14
1XX4MXXX M
XXXX32XX 24
XXX14XM1MXXX4S
XXXXXXX4MXXMX1M4
XXX31XXXM 1XX2MMMM4
XX   X32X 2X3X2313MX4
MX EWMXXXXXXXXX  2XMX4
X 2W2XXXXXXXXXM 2X XXXXM4
MX32MMXXXXXXX414XXXXXX MX4
XMXMS 1XX X2MM XXXXXX  XXXMMM4S
 XXXN   1XM3 XXXXXXXXXXXX3 1MN1MX X4
2MXM42M 2MM2M1XXXXXXXXX3XX 2M42XXXXXXX4
M3 1MXX N13X141XXX2X31XM3 MXX3XMXXXXXXXM
3    XXX4 E3 MXXXXXX42XX 2MXMXMX  X1XX1MX4
    2MXMMME4 1XXXXX3MXXXXXX231X  2XXMX2XMX4
    M3  1M41W   1X3E3XXXXXXX4 M42MMXX3N2X31MM4S
  2M3   2MM4     MM 2XXXXXXXXXM3NX32X42M3  M NNMX4
 2M3   2M3 1M4EW 1MMMMXX3 XX 1M 2MXM3MMM   M4 SXMX4
2M3   2M3   1MX4      1M  XX4 MXMM1M M3   2M1M31XXMM4
M3   2M3     MMM4      MMMMMMMM   2X M   2XM    M3 SMM4
3   2M3    2M3 1MXXX       1XX3   MMX3   XXX   2M  N  MM4
   2M3    2M3   MXXMMM      XX           MXMMMMX        14S
  2M3    2M3   2M    MXMMMMMMMMM       M      1MMM4      N14
X M3    2M3   2M3   2M3           MMMM           1M4       M
XXM    2M3   2M3   2M3                            1M       M4
XX3   2M3   2M3   2M3                              M       MX
X3    M3   2M3   2M3                              2M      2XM
3    2X4X4MM3    M3                              2MM     2M M
    2X4XXXM3    23        2W             S    S2MM     2M3  M
    X4XXXX3    23        E3              NMXXM3N     2M3M  2M
   2XXXXX3    2X4  XX        2M         X XXXX   S 2XXX4XXXMM
  2XXXXX3    X4XXXMXMXXX X   1M4   24XXMMXMXX3  23MXXXXXXXXX3
  MXXXXX     2XXXXXX XXXXXXXMS MXXXXXXXX31W  2MMX4XXXXX  XX3
    XXX3    2XXXXXXX  XXXXXXXN XXXXXXXXM   2M3 XXXMXX3
4           XXXXXXX M XXXXXXX XXXXXXX3   2XXX2XMXXX2M
XX4         XXXXXXXS X X XXXXXXXXXXX3   2MMXMX3X3  N
XSMMX4    E4MXXXXXMNSMSMSMXXXXXXXXX3   2MM  X S   S
 NMX2MMM4  XM       N N N             2M3 SS  N   N
    N  2MMX3X4       S     SSSS      2M   NN
       N  1XMMMMMMMMMNMM   NNNNMMXMMMMMS
                  X          S  XXX    N
                            SN
                            N
```

Fig. 5.20
The initial classification of the sports shoe picture.

the shoelaces. Because the edges are very close together (in terms of the digitised image) they interact and fail the cross test. The illegal pixels near the bottom of the shoe are the result of intensity gradients caused by changes in the surface orientation and incident light. Considering the definitions given above for texture and gradient, and taking image veridicality into account, it is appropriate to group illegal mixed pixels into "illegal regions" on the basis of connectivity. So, for any two illegal mixed pixels in an image, they are in the same illegal region if a path between them can be found which only passes through illegal mixed pixels. For this purpose each pixel is considered adjacent to its eight neighbours. This definition is easily formalisable and consistent with the earlier assumptions concerning the formation of illegal mixed pixels.

The algorithm for finding illegal regions is based on a couple of well-known techniques for boundary tracking and connectivity analysis (Rosenfeld and Kak 1982). One distinguishing characteristic of each illegal region arising from its definition is that it has an exterior boundary. By classifying pixels into two sets: illegal mixed pixels and others, this boundary can be found by tracking between pixels of the two classes, and hence the illegal regions can be defined. In fact, we actually track along the border of the illegal region, which is the set of illegal mixed pixels just inside the region boundary. Operating a raster scan, each pixel is examined in turn until an unused illegal mixed pixel is found. The coordinates of this pixel are stored and tracking proceeds by "keeping your right hand on the boundary" (from inside the region), i.e. tracking anticlockwise. The region boundary is imagined to pass between pixels, since this avoids problems with unit width regions and makes it easier to tell whether a pixel is inside a particular region. Tracking proceeds, based on 8-connectivity, until the starting point is reached again. 8-connectivity was chosen because the illegal regions are sometimes not homogeneous. This is especially true of textured regions. Using 8-connectivity makes the illegal region definition slightly more flexible in overcoming these variations. For example, if we are tracking the boundary and reach pixel 1 in fig. 5.21 then, since tracking is based only on illegal mixed pixels, pixel 2 would not be encompassed by the region if only 4-connectivity was used. Because of the inhomogeneous nature of illegal regions resulting from texture, holes in the regions are filled in and assumed to be part of the region. Normally these holes are very small, consisting of only one or two pixels. If desired, holes could also be tracked and their areas computed. Large holes could then be dealt with separately since they probably arise from a scene source different from the surrounding region.



X indicates an illegal mixed pixel.

Fig. 5.21
This shows the advantage of 8-connectivity for tracking illegal region boundaries.

During tracking, the position of each vertical boundary segment is noted and the bounding rectangle of the region is kept. Using these, the region is "filled in", i.e. every pixel inside the region boundary is marked. This is done by moving a pointer from left to right along every row in the bounding rectangle. Each time the pointer is moved the two pixels indicated before and after the movement are checked to see if a legal-illegal (or illegal-legal) transition occurred. If such a transition did occur then a further check is made to see if the boundary between the two pixels was stored as one of the vertical boundary segments of the illegal region. If it was part of the region boundary, then a switch, initially set at off, is flipped. The switch indicates whether or not the pixel currently being pointed to is inside the region. This is an implementation of the well known technique for finding whether or not a point is inside a closed curve. You simply draw a line from the current position to a point known to be outside the curve. Then count the number of times the line intersects the curve. If the number of intersections is odd, then the current point is inside the curve; if the number is even, then it is outside. By repeating the above process for every row of the bounding rectangle, the illegal region is marked. It's necessary to store the vertical boundary segments on the region boundary to ensure that holes are filled in. The illegal regions found by applying this method to the shoe picture are shown in fig. 5.22.

5.2.3 Distinguishing between Gradient and Texture

Illegal regions are produced either by gradients or by texture. We need a test which, given an illegal region as an argument, indicates whether it's a gradient region or a texture region. A gradient region is produced when the image intensity variations in a region,

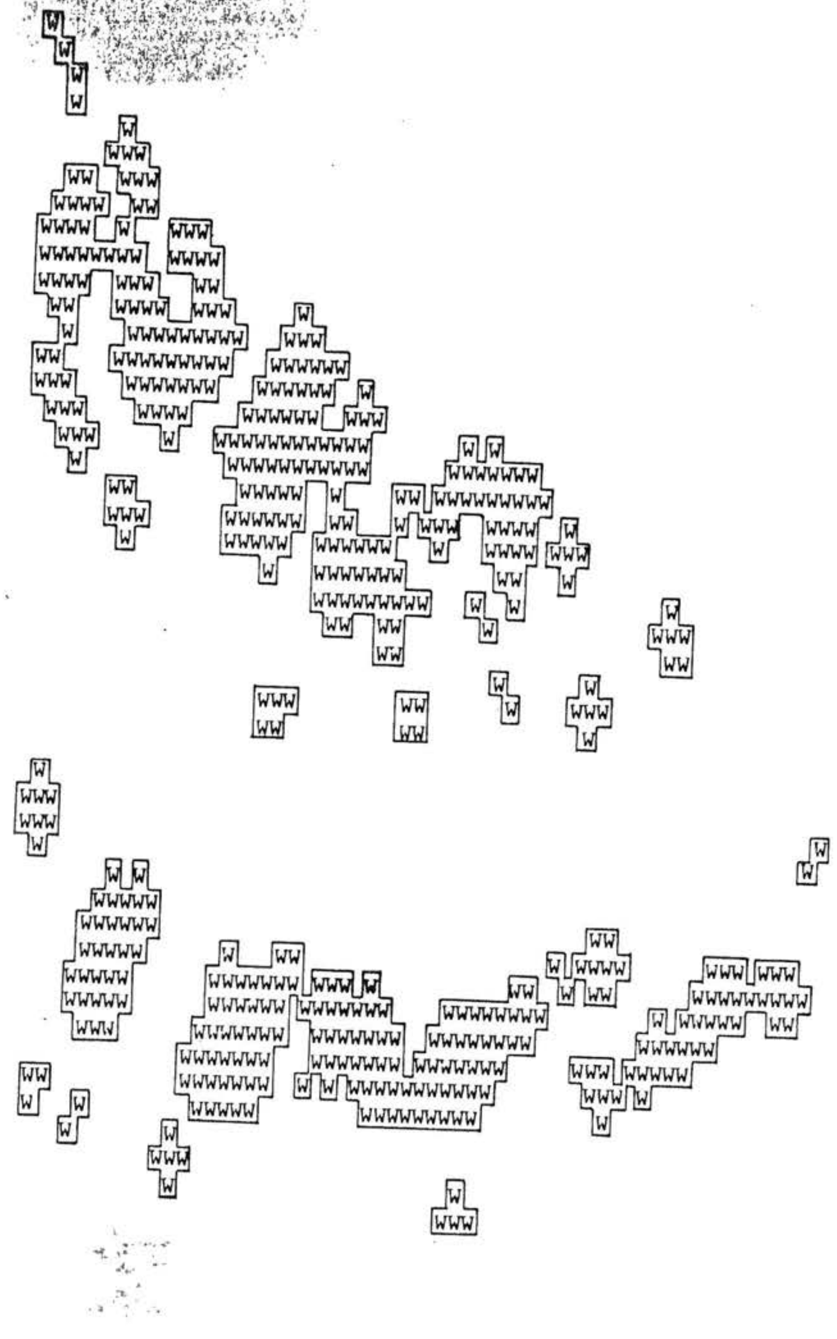


Fig. 5.22
The illegal regions found in the sports shoe picture.

which are the result of smoothly changing incident light and/or surface orientation and/or reflectance variations on a single surface in the scene, are significant. A texture region is produced when many sharp intensity changes are so close together in the image that they cannot be distinguished separately. These edges may be of any kind. Because textured regions can be produced in so many different ways it is difficult to model them. However, in general, the intensity variations within a textured region will not have an obvious relationship to neighbouring intensity variations, unlike gradient regions where smoothness can be expected. Thus, we can see a broad division between gradient regions consisting of smoothly changing intensities, and texture regions consisting of relatively unrelated intensity changes. Since it's not obvious how to model texture for our purposes a reasonable approach is to model gradient regions and to make a decision based on how well a particular region fits the model.

One possible method is to constrain the allowed surface shapes in the world, and to classify the light source types. The expected class of gradient regions could then be derived, as a set of intensity surface types. Classification would involve finding the best fit match between the region under consideration and one of the allowed set. The goodness of match would be thresholded to provide a decision - gradients should fit well, textures badly. This approach is, however, unsatisfactory, from several points of view:

1. it puts constraints on world shapes and light sources, which limits the applicability of the system and hence violates our first goal of edge detection.
2. deriving an effective set of allowed image intensity surfaces may be difficult.

3. this method would incur a heavy computational load.

An alternative approach would be to first find the intensity gradient (magnitude and direction) everywhere within the illegal region and then use these to produce a histogram of amplitude vs. direction. This can be represented in a polar coordinate system with the angle given by the direction and the radius given by the amplitude (fig. 5.23). Since a gradient surface should be smoothly varying both in amplitude and direction, it should produce a single large cluster in the histogram. Texture, on the other hand, should produce a scattered histogram with many small clumps (fig. 5.24). However, it is not obvious what decision criteria could be used to distinguish between gradient and texture and how these could be arrived at. Another problem with this approach, in common with many other histogram-based methods, is that it doesn't take into account the local image intensity relationships in the region above the level of first differences. Since we're interested in examining image intensity smoothness we need to examine relations between nearby first differences. As a result of this flaw, it is conceivable that a region which is not smoothly varying could still produce a single cluster in the histogram. For example, a large texture region, assuming it had randomly distributed intensities, might well produce a single large connected region in the histogram.

An alternative approach, which attempts to make use of local smoothness, is to base the decision on the second differences within the region. In a gradient region, because the intensity is smoothly varying in amplitude and direction, the second differences should be small. In a texture region, where these constraints don't hold, the second differences should be large.

In the case of an ideal plane, which we can take as the limiting case of a gradient region, the second differences will be zero. At the other extreme, where the sign of adjacent first differences is different, the magnitude of the 2nd. differences will be the sum of the magnitudes of the first differences as shown in fig. 5.25. So we base our method of distinguishing between gradient and texture on whether the magnitude of the 2nd. difference tends towards zero (gradient) or towards the sum of the magnitudes of the first differences (texture).

For each illegal region, the region-finding process outlined above marks all the pixels inside the region in an auxiliary array in registration with the picture array, and returns the bounding rectangle within which the illegal region is enclosed. Each row of the bounding rectangle is tracked along, and at any point where there are three consecutive pixels inside the region the second difference and the average first difference are evaluated. If the pixels are p_1, p_2, and p_3, then

$$\text{2nd.difference} \qquad d_2 = |p_1 - 2p_2 + p_3|$$

$$\text{ave.1st.difference} \qquad \bar{d}_1 = \frac{|p_1 - p_2| + |p_2 - p_3|}{2}$$

For a gradient region we expect d_2 to tend to zero and for a texture region we expect it to tend to $|d_{11} + d_{12}|$. Since the 2nd. differences are subject to symmetrical Gaussian noise (standard deviation 2σ, where σ is the picture noise standard deviation) the local decision is made as follows:

if $d_2 > \bar{d}_1$ then texture else gradient

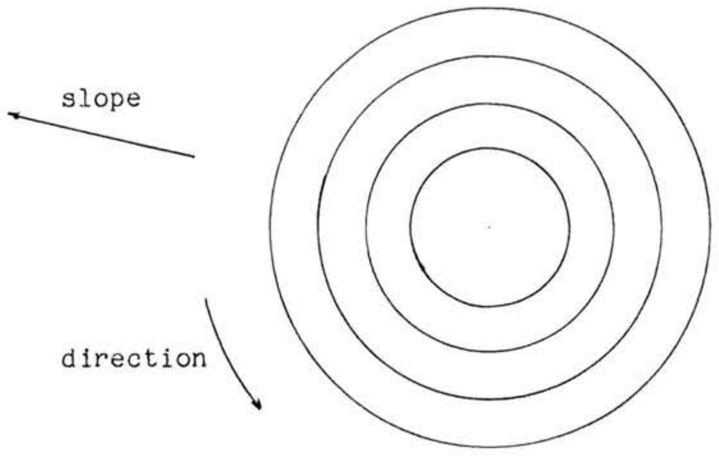


Fig. 5.23
A polar histogram of intensity gradients could be used to discriminate between gradient and texture regions. The intensity gradient is computed at every point in the region, then plotted on the histogram, with slope as radius and direction as angle.

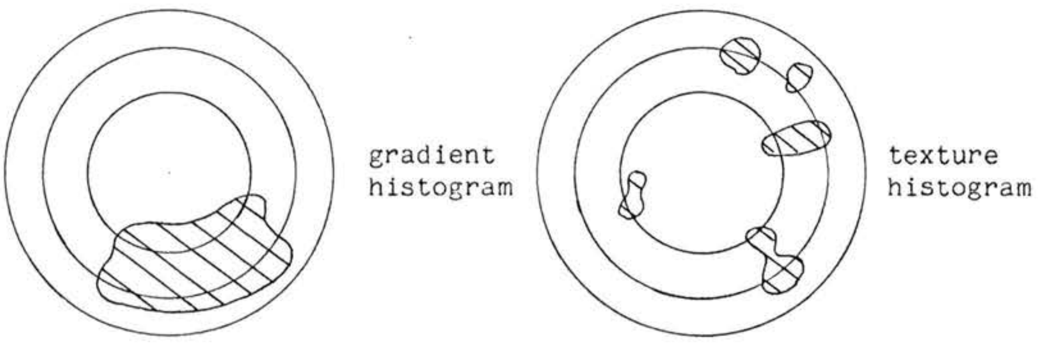


Fig. 5.24
Since a gradient region arises from one surface with smoothly-varying properties, the corresponding histogram should have a single connected cluster of entries. On the other hand, since intensities within the texture region will often be locally uncorrelated, a histogram with scattered entries should result.

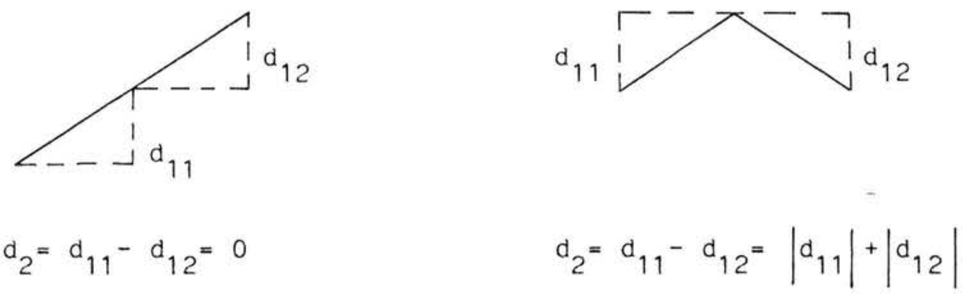

d_{12}
d_{11}
$d_2 = d_{11} - d_{12} = 0$
d_{11}
d_{12}
$d_2 = d_{11} - d_{12} = |d_{11}| + |d_{12}|$

Fig. 5.25
2nd. differences for idealised gradient and texture.

That is, since zero and the sum of the first difference magnitudes are the bounds on d_2 and since the noise is symmetrical, the threshold is chosen in the middle of the range $(\bar{d}_1)$.

A pair of tallies is kept representing the number of times the decision was made in favour of texture and the number of times in favour of gradient. The same process is repeated for the columns of the illegal region producing another pair of tallies based on the vertical 2nd. differences in the illegal region. The results of applying this technique to the shoe picture are shown in fig. 5.26.

All that remains is to make a decision for each region depending on its four tallies. We must take into account the possibility of unidirectional textures (i.e. textures which are oriented in one direction only, e.g. looking at a zebra from a distance). So if either the horizontal or vertical texture tally is significantly greater than the corresponding gradient tally, then the region should be labelled "texture". Since there is obviously no point at which gradient suddenly becomes texture (or vice-versa) there should be a "don't know" classification category which is chosen if the gradient and texture tallies are about the same. Thus the decision test is as follows:

```
if h_tex_tally - h_grad_tally > hthreshold
or v_tex_tally - v_grad_tally > vthreshold then TEXTURE

elseif h_grad_tally - h_tex_tally > hthreshold
    or v_grad_tally - v_tex_tally > vthreshold then GRADIENT

else DON'T KNOW
```

The thresholds are chosen to be some constant factor times the

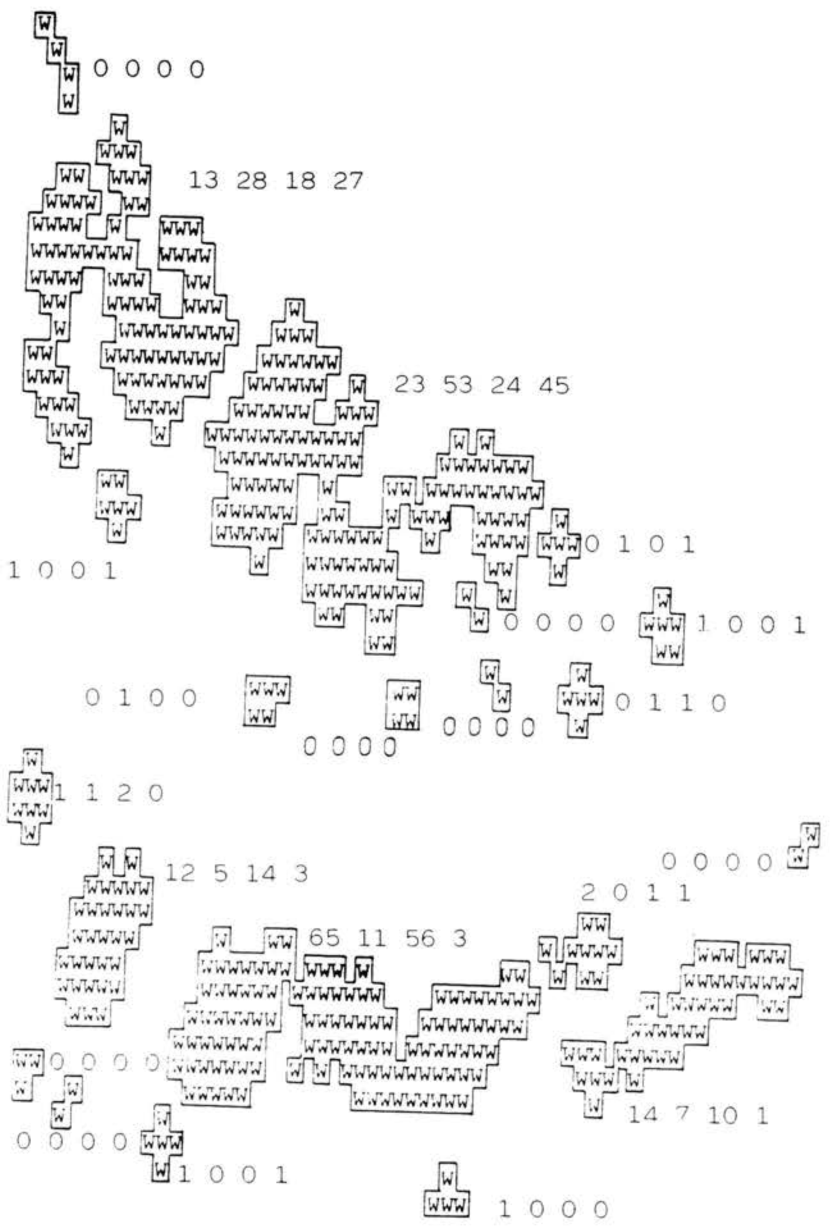


Fig. 5.28
The four numbers beside each region are their gradient and texture tallies in the order:
h_grad_tally h_tex_tally v_grad_tally v_tex_tally

equivalent horizontal or vertical region sizes (i.e. the number of times the appropriate 2nd. difference operator was applied). The equivalent horizontal region size is then simply the sum of h_grad_tally and h_tex_tally; similarly for the vertical size. The final classification obtained by using a multiplying factor of 0.25 on the shoe picture is shown in fig. 5.27.

We originally performed gradient/texture classification by finding the average first and second differences for an entire region and then basing the classification on these (Beattie 1982).

if $d_2-\overline{d}_1>t$ then the region was labelled as texture,

if $d_2-\overline{d}_1<-t$ then the region was labelled as gradient.

$-t<d_2-\overline{d}_1<t$ led to the region being unclassified.

The threshold depended on the picture noise and the size of the illegal region. This method suffered from the same disadvantage as the histogram technique mentioned above, in that averaging the magnitude of the first and second differences over an entire region destroyed the clear relationship between local smoothness of intensity and the region texture measure. The method described in this section is better in the sense that it usually produces the classification expected by the user from her/his knowledge of the scene which produced the image, as it did in the sports shoe picture.

In the previous discussion of texture it was suggested that the perception of texture occurs when the perceiving system is unable to fully interpret each intensity change in terms of its scene meaning. This results in a field of intensity changes being dealt with as a single unit. Implementation of this idea was performed in two stages:

```
 MMM4
 X4 MM
 MX MX4
 MMXM 1W
 W1XM 24
 W  1MMT4 S
 4S  1TTTM14
 M1TT4MTTT M
 MTTTT32TT 24
 TTTT14TM1MTTT4S
 TTTTTTTT4MTTTT1M4
 TTTT31TTTM 1TT2MMMM4
 WTT   TTTT 2TTT2313MT4
 4MT EWMTTTTTTTTT  2TTT4
 TT 2W2TTTTTTTTTM 2TTTTTTM4
 TTT32MMTTTTTTT414TTTTTT MT4
 MTTTMS 1TTTT2MM TTTTTT  TTTMMM4S
3 TTTN   1TM3 TTTTTTTTTTTT3 1MN1MT T4
W2MTM42M 2MM2M1TTTTTTTTTTT  2M42TTTTTTT4
MM3 1MTT N13X141TTTTT31TM3 MTT3TTTTTTTTTM
M3    TTT4 E3 MTTTTTT42TT 2MTMTTT  TTTT1MT4
     2MTMMME4 1TTTTT3MTTTTTT231T  2TTTT2TTT4
     M3  1M41W   1T3E3TTTTTTT4 M42MMTT3N2T31MM4S
   2M3   2MM4     MM 2TTTTTTTTTM3NX32T42M3  M NNMT4
  2M3    2M3 1M4EW 1MMMMTT3 TT 1M 2MXM3MMM   M4 STTT4
 2M3   2M3   1MX4      1M  TT4 MXMM1M M3   2M1M31TTMM4
2M3   2M3     MMM4      MMMMMMMM   2X M   2TM    M3 SMM4
M3   2M3    2M3 1MTTT       1XX3   MMX3   TTT   2M  N  MM4
3  2M3    2M3   MTTMMM      XX            MTMMMMX         14S
W  2M3    2M3   2M   MXMMMMMMMMM       M       1MMM4      N14
4G M3    2M3   2M3   2M3            MMMM           1M4      M
GGGM    2M3   2M3   2M3                            1M       M4
GGG3   2M3   2M3   2M3                             M        MX
MG3    M3   2M3   2M3                              2M       2XM
M3    2G4G4MM3    M3                              2MM      2M M
    2GGGGGM3    23         2W              S    S2MM     2M3  M
    GGGGGG3    23         E3               NMGGM3N      2M3M  2M
   2GGGGG3    2G4  GG          2M          G GGGG   S 2GGG4GGGMM
  2GGGGG3     GGGGGGGMGGG G  1M4   24GGMMGMGG3  23MGGGGGGGGG3
  MGGGGG     2GGGGGG GGGGGGGMS MGGGGGGGG31W  2MMG4GGGGG  GG3
    GGG3     2GGGGGGG  GGGGGGGN GGGGGGGGM   2M3 GGGGGG3
M4           GGGGGGG M GGGGGGG GGGGGGG3   2GGG2GGGG2M
MXX4         GGGGGGGS G G GGGGGGGGGGG3   2MMGGG3G3  N
MXSMMX4    E4MGGGGGMNSMSMSMGGGGGGGGG3   2MM  G S   S
 NMX2MMM4  TM        N N N                2M3 SS   N   N
    N  2MMTTT4       S      SSSS        2M   NN
       N  1TMMMMMMMMMMNMM   NNNNMMGMMMMMS
                  X          S  GGG    N
                            SN
                            N
```

Fig. 5.27
The final classification:
gradient regions are marked G
texture regions are marked T.

1. First identify those intensity changes to which the system would not be expected to assign a scene meaning because of their proximity to other intensity changes.
2. Separate these into those caused by smooth extended gradients and those caused by step-like intensity changes.

The algorithms for these two steps comprise an algorithmic definition of texture. If the algorithms have been well chosen, the texture regions produced by applying them to a given image should correspond closely to what we would expect given our earlier computational definition of texture. Notice that, as Marr (1982) has suggested, the algorithmic definition is only one particular interpretation of the computational definition. Many other algorithms could have been used, each of which might produce slightly different results, indeed, several have been suggested in this section. Tne one selected was chosen because it seemed closest to the computational definition. However, if one of the others was used, the results should not differ too drastically since they should all be constrained by being different implementations of the same computational definition.

5.3 The Structure of the Single Level System

Consider again the final shoe picture classification given in fig. 5.27. The illegal regions have been correctly labelled as gradient or texture, and by applying the edge tracker described in section 5.1 it is possible to identify the step-like edges. The output of the system would be a pixel/boundary segment classification (in an array in which each illegal region is identified) and a list of step edges, each list represented by a sub-list of coordinate pairs. This representation is a good one if it forms an appropriate input to the next level of the system. We have assumed that edges will next be subject to constraint analysis, for which our representation is suitable. However, consider fig. 5.28 which is meant to represent the image of a cylinder with a strong intensity gradient on its cylindrical face. Running the programs of sections 5.1 and 5.2 would produce the results shown in fig. 5.29 There is an obvious problem here in that the gradient region boundaries on the left and right of the region are really part of the same scene element as the left and right edges of the face as found by the edge tracker. This represents a case where a uniform region and a texture region should be blended. This can be detected by checking the smoothness of intensity change across gradient region boundaries and marking those where the intensity change is smooth, since they are the result of a smoothly increasing gradient going above threshold and are not particularly significant in scene terms. For example, in fig. 5.29 the top boundary of the gradient region is caused in this way. Removing gradient regions which have such boundaries is simple and would effectively blend the gradient region with the adjacent uniform region. In the current implementation of the system the resulting region would be considered to be uniform. Thus, although gradient region boundaries are often significant in scene terms, there are exceptions which could be detected if desired. This phenomenon can also be seen in the shoe picture classification where the sides of some of the gradient regions at the base of the shoe are really continuations of the step edges produced by the stripes on the shoe (i.e. they are reflectance edges). Similarly, texture region boundaries are usually highly significant in scene terms (often they are obscuring edges). To summarise the above: the boundaries of gradient and texture regions are often significant in scene terms in that they correspond to particular scene/imaging events, such as reflectance

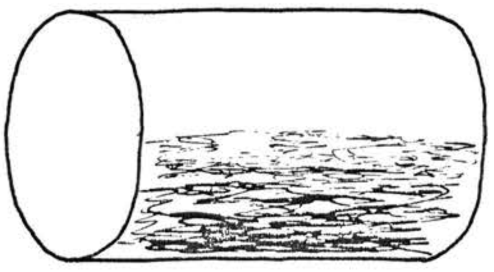


Fig. 5.28
This shows a cylinder with the lighting arranged so as to produce a strong intensity gradient on the cylindrical face.

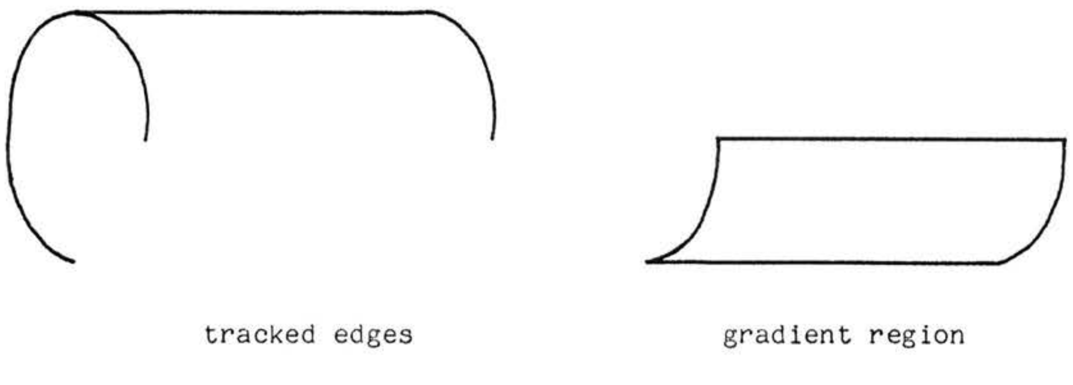


Fig. 5.29
As the system is currently organised, the input image shown in fig. 5.28 would produce the set of tracked edges shown on the left, and the gradient region (within the classified pixel array) on the right. The left, right, and bottom sides of the gradient region are significant scene/imaging entities, so it is desirable to represent them explicitly.

edges or obscuring edges, so they should be explicitly represented in a useful way.

This can be accomplished quite elegantly by reorganising the system as shown in fig. 5.30. After the initial pixel classification, as described in the first part of section 5.1, the illegal regions are found and classified, as described in section 5.2. At this point the boundary of each illegal region is explicitly inserted into the data structure using horizontal and vertical primary boundary segments. Some care has to be taken to ensure that the constraints on which the tracking rules of figs. 5.5 and 5.6 are based are not violated. Now the edge tracker as described in the latter part of section 5.1 can be run unaltered and will automatically find the illegal region boundaries as well as the step-like edges.

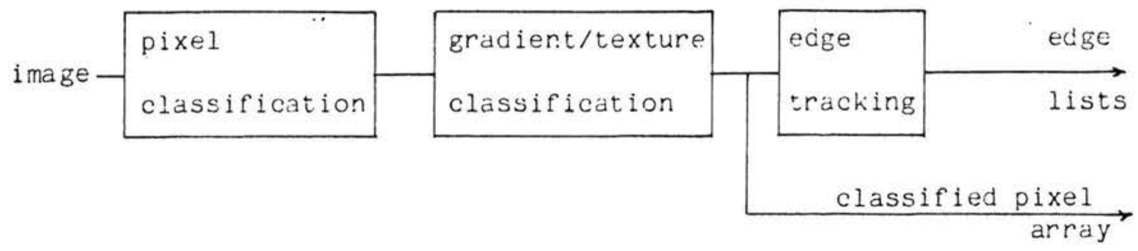


Fig. 5.30
The final structure of the single-level system.

The effect of reorganising the system in this way is shown in figs. 5.31 and 5.32 overleaf. Edge tracking on the pixel/boundary classification before inserting illegal region boundaries produces

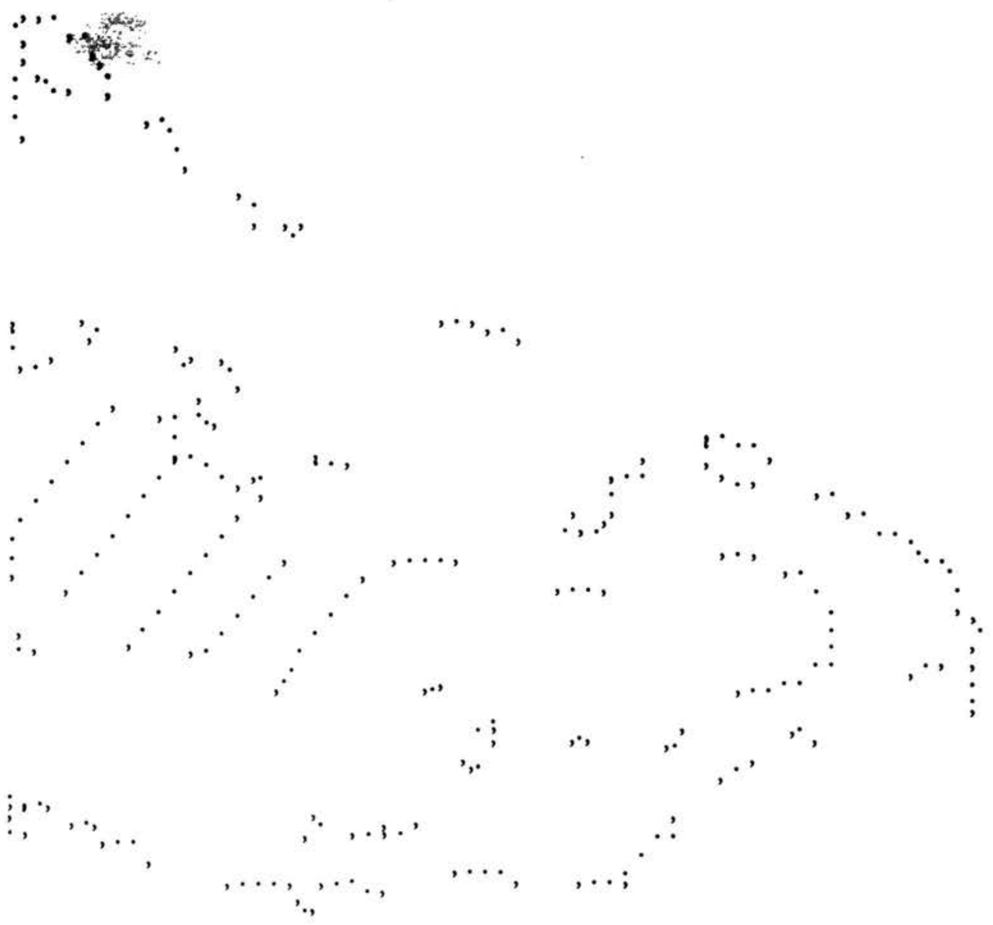


Fig. 5.31
This shows only the step-like edges which were tracked in the shoe picture. The commas are used to indicate terminators or junction points, while dots indicate mid-line points.

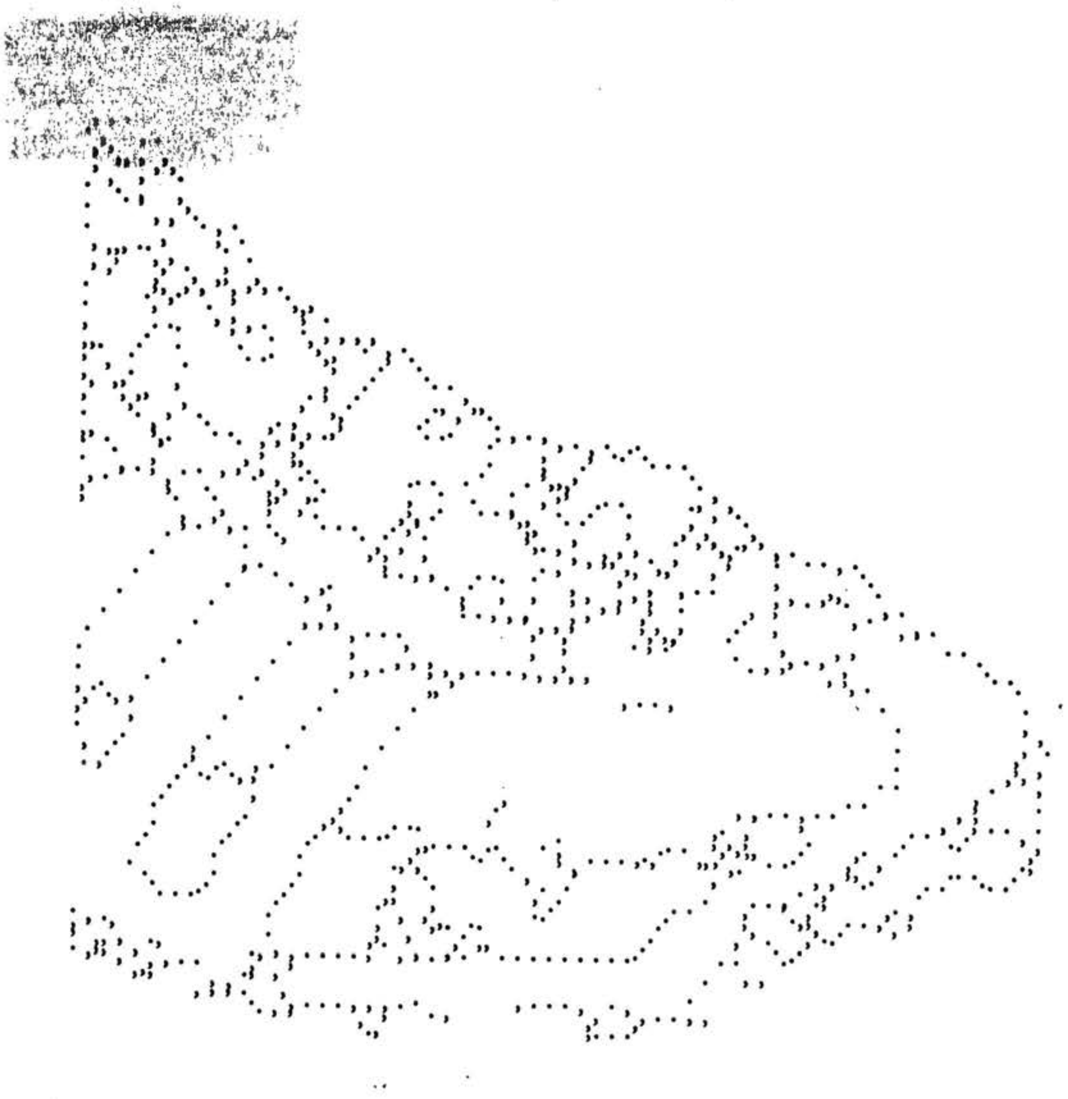


Fig. 5.32
This shows the step-like edges and the illegal region boundaries tracked by delaying the edge tracking until after gradient/texture classification.

the edges shown in fig. 5.31. If the illegal region boundaries are inserted then edge tracking produces the enlarged set of edges shown in fig. 5.32. Notice that this has the desired effect of finding the edges of the stripes on the side of the shoe. To facilitate further analysis, the edge list produced is divided into three sub-lists. One contains the step-like edges tracked, one contains the boundaries of texture regions, and one contains the boundaries of gradient regions. If desired, some further analysis could be carried out on the gradient region edges, since these are easily discriminable into two classes. Consider the gradient region found in fig. 5.29. On the top edge of the region, the boundary is produced because the intensity gradient resulting from the combination of surface orientation change and variation in illumination goes above threshold. On the sides of the region, the boundary is due to a sharp change in surface orientation which is unrelated to the factors causing the gradient region. Boundaries of the former type occur quite commonly and can be identified by examining the smoothness of the image intensity across the boundary.

This completes the design of the single-level system. The goals of edge detection for step edges have been satisfied, as has the first general goal of finding all the significant intensity variations. Some progress has been made on the second general goal by classifying illegal regions as gradient or texture, although further constraint analysis is not the subject of this thesis. However, as previously mentioned, intensity variations occur over a wide range of scales in the image. In particular, blurred edges whose width is two or three pixels wide commonly occur. At present these are represented as gradient regions, but it would be advantageous to also represent these as edges in some way. In the next chapter we investigate the problem of finding and representing edges and regions at different scales.

6 The Multi-Level System

In this chapter, we investigate the requirements and behaviour of multi-level systems. The major area of interest is in integrating the data provided by several single-level systems, operating on different-resolution versions of the image, to generate a useful representation of all the significant intensity changes in the image.

6.1 Need for the Multi-Level System

As we have seen, images of most types of scenes contain edges of a range of widths.. Edges containing the highest frequency components, i.e. step edges, are the most important type in scene analysis, but blurred edges are also important since they also result from significant scene/imaging events. To be general purpose, an edge detection system should be capable of explicitly representing blurred edges. Several image analysis algorithms, particularly in the area of stereopsis, use results obtained with low-pass filtered images to guide the vision processes working with full images. Ideally, there should be explicit links between input representations to facilitate the propagation of disparities between channels. More specifically, the edge detection system being designed here is intended for intensity-based edge labelling, part of which should involve identifying shadow edges which are often blurred.

A second reason for using low-pass filtered versions of images is to increase system robustness. Filtering out the higher frequencies reduces the noise present in the output image. The penalty for removing the higher frequencies is that the accuracy of edge localisation is generally reduced. Hence the need for multi-channel systems: high bandwidth channels can be used to find the step edges accurately, but low-bandwidth channels are also needed to find the blurred edges and improve the overall system performance in noise. To make full use of the multi-channel system, however, the outputs of the various channels must be successfully integrated into a single coherent representation of all the edges present in the image. This is currently the major problem in successfully designing and implementing such systems.

As implied above, the basic idea of multi-channel edge detection is to use a set of low-pass or band-pass filters to produce a series of filtered versions of the original image in which the rate of intensity change is limited, then to find the edges present in each filtered image. There are two ways of doing this. The first is to combine the filtering and edge detection tasks in a single operation. This results in a family of edge detector masks which have the same basic shape, but variable spatial extent. The second method is to prefilter the image before applying the edge detectors. One variant of this second method is to reduce the resolution of the image as well as, or instead of, filtering. Reducing the resolution is basically a low-pass operation itself, as we shall see. Apart from the computational advantages of such an approach, some justification for reducing the resolution comes from the reduced information content of the filtered image, which should require a smaller data set to fully describe it.

In terms of the scene-image relationship, reducing the resolution is very similar to increasing the scene-camera distance. Averaging an n*n block of pixels at a high resolution to produce a single pixel value at a lower resolution is similar to increasing the scene-camera distance until the portion of scene that previously subtended an n*n block of pixels, subtends a single pixel.

Since the single-level system (SLS) described in chapters three to five is general purpose and, therefore, not dependent on the scene-image distance, it should be able to be successfully applied to an image at any reasonable resolution. Also, since resolution reduction is a low-pass operation, the SLS will react to different freqency components in different resolution versions of the image. So, one way to find edges of a range of widths is to apply the SLS to several different-resolution versions of the same image, then combine the multiple sets of SLS outputs.

The need to combine outputs is the major factor in determining how much resolution reduction is suitable. For computational efficiency, it would be better to use few resolutions as widely spaced as possible. At the other extreme, using a continuum of mask sizes (Witkin 1983), corresponding to the use of many resolutions in our case, has been shown to have advantages. Since computational efficiency is not the main goal of this thesis for the reasons given in chapter two, a small reduction in resolution will be used here to simplify the combination process. As we shall see in the next section, for our system there are advantages in averaging the intensities in a 2*2 window to produce the intensity of a pixel in the next lower resolution (as was shown in fig. 3.5). Therefore, we begin the multi-level system by successively reducing the original image by a factor of four, applying the single-level system to each version of the original image produced.

To design a method for combining the outputs of two or more single-level systems, we must consider how the image entities produced by significant scene events change as the resolution is reduced. These changes can be examined in either the frequency or

spatial domains.

First, consider the frequency domain. Tanimoto & Pavlidis (1975) and Tanimoto (1976) have investigated this issue. They show that the transfer function of the resolution reduction is basically low-pass, but that aliasing occurs as shown in fig. 6.1. Although the effect on the frequency spectrum gives some idea of the overall effect of reducing the resolution, namely that high frequencies are attenuated more than low frequencies, and that frequency components above half the new sampling frequency are aliased down about it, it is very difficult to estimate the results in the spatial domain. Yet our method of combining the outputs of SLSs run on different-resolution versions of the image must rest squarely in the spatial domain, because it is the spatial arrangement of image intensities which reflects the spatial layout of the scene. This is generally true of any vision system based on trying to extract scene properties from images.

To investigate the effect of reducing image resolution in the spatial domain, we must find how the image appearance of each type of scene/imaging entity is affected. For instance, how does a blurred shadow edge change under a reduction of resolution? Fortunately, because we have carefully specified the single-level system, and because the design of the SLS rested on image veridicality, we can consider the effect of reducing resolution on the output of the SLS rather than the input. This would not be the case if, for example, we had some relaxation operation built into the SLS. Then, it would be difficult to predict the exact relationship between the input and output at each level, making it much harder to combine the outputs.

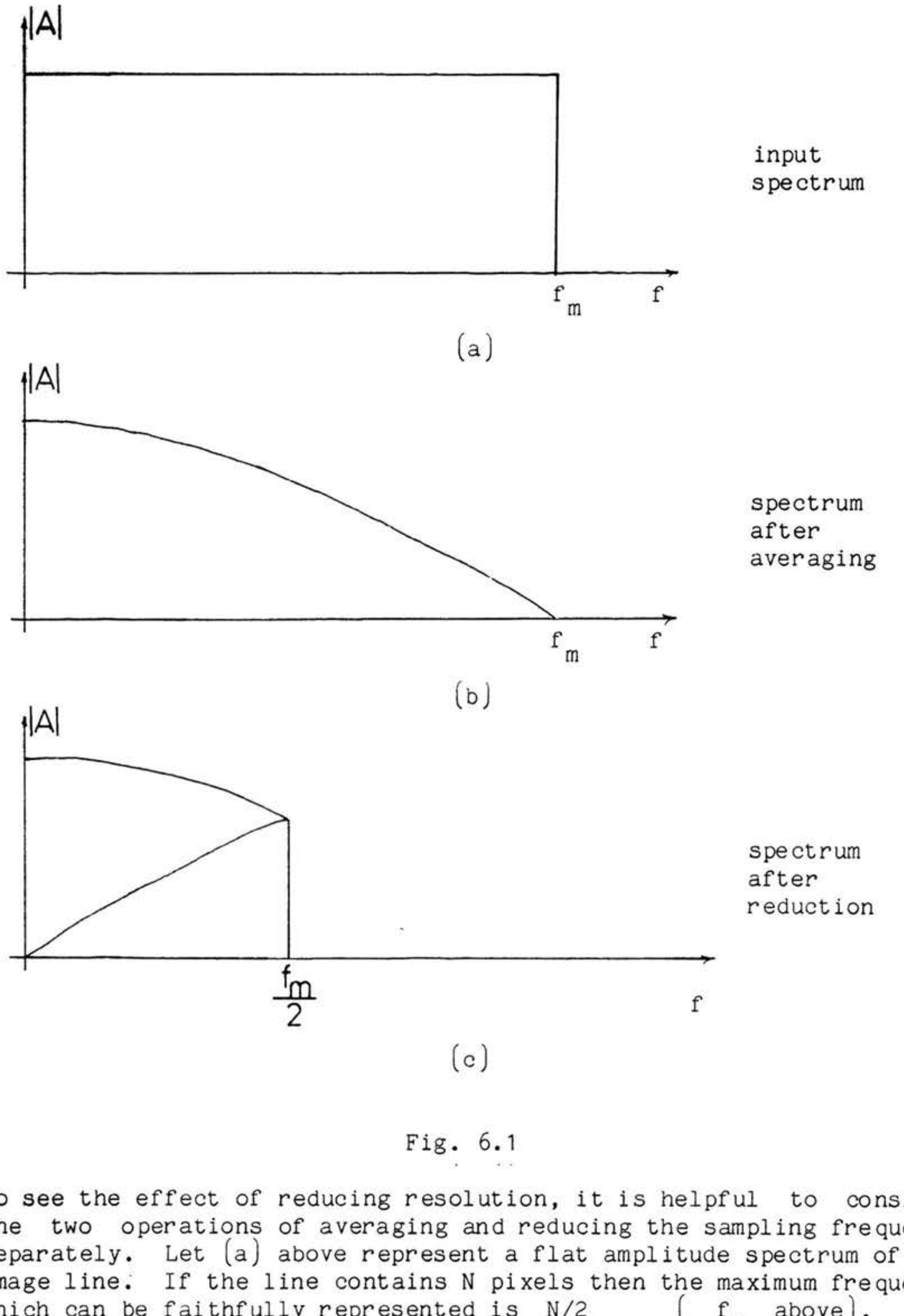


Fig. 6.1

To see the effect of reducing resolution, it is helpful to consider the two operations of averaging and reducing the sampling frequency separately. Let (a) above represent a flat amplitude spectrum of an image line. If the line contains N pixels then the maximum frequency which can be faithfully represented is N/2 (f_m above). The effect of averaging, i.e. replacing each pixel intensity by the average of its own and the succeeding pixel's intensity, is a low-pass filtering operation producing the spectrum shown in (b). Reducing the resolution, by deleting every other pixel, is equivalent to reducing the sampling frequency to half its original value, producing the aliasing effect shown in (c), where the frequency components previously above f_m /2 are folded down about that frequency.

There are four identifiable entities in the single-level system output, as follows:

- step edges
- gradient regions
- texture regions
- uniform regions

In fact, gradient and texture region boundaries are also represented explicitly, but it is not necessary to consider these here since the regions themselves are used. Each of the four types of entity satisfies certain constraints, as follows:

step edge	- thinness, connectivity, gradient.
gradient region	- thickness, connectivity, gradient, smoothness.
texture region	- thickness, connectivity, gradient, roughness.
uniform region	- gradient (lack of).

By examining the effect of reducing image resolution on these constraints we can see how the entities in the SLS output change with resolution.

For a step edge, both thinness and connectivity are preserved. Since the low resolution SLS should be able to use a lower significance threshold for the same error rate, gradient significance should also be preserved. In other words, a step edge should map to a step edge under a reduction in resolution (barring interference).

Next, consider the spatial aspects of gradient and texture regions, i.e. their properties of thickness and connectivity. Both of these may be violated by reducing the resolution. The connectivity of a convex gradient or texture region would not be affected, but in extreme cases, the connectivity of a concave region could

change, as shown in fig. 6.2. Since the spatial extent of a gradient **or texture** region will be reduced, it will often be the case that an **elongated** gradient, e.g. as produced by a blurred edge, or texture region, will lead to the thickness constraint being violated; and hence the thinness constraint being satisfied. So under a reduction in resolution a gradient or texture region may become a step edge. The smoothness or roughness of gradient or texture regions may or may not be preserved depending on the periodicity of the intensity changes within the region relative to the new sampling frequency.

Finally, consider uniform regions. Their only constraint is a lack of gradient amplitude. Since the sensitivity of the system should be increased as the resolution is decreased, faint intensity changes which may not have been detectable at high resolution may be found under reduction.

We can summarise the effects of reducing resolution on the four types of image entities as follows:

steps	->	steps; illegal regions (I)
gradients	->	gradients; steps
textures	->	textures; gradients; steps
uniform	->	uniform; others

where the I indicates interference.

Now the reason for reducing the resolution is mainly to find and represent blurred edges explicitly, although we are also interested in increasing sensitivity. From the above we see that blurred edges will produce low-resolution step edges (as will what were originally step and faint edges). Hence, if we take high and low resolution versions of the same image and compare the two lists of step edges produced (not including gradient and texture region boundaries), we

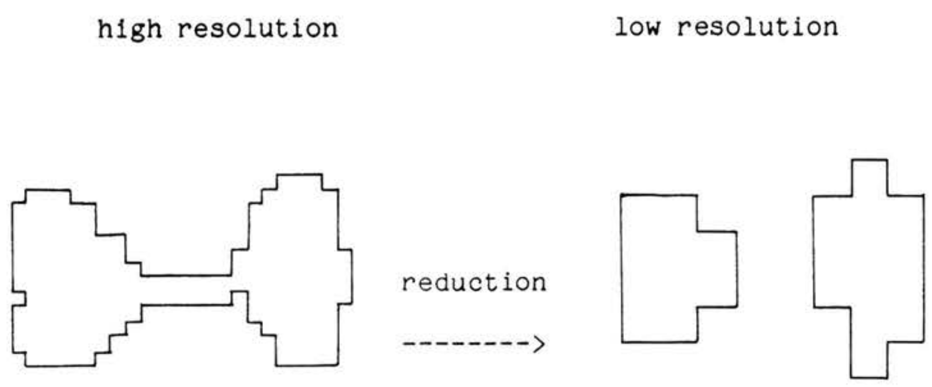


Fig. 6.2
A strongly concave region, such as the one shown above, may undergo a change in connectivity as the image resolution is reduced.

should find extra entries in the low resolution edge list corresponding to blurred and faint edges. Some edges will also have disappeared due to interference. So one way to improve the edge detector's performance on blurred and faint edges is to run the SLS on two resolutions, take each step edge found at the low resolution, and check the corresponding part of the high resolution classified array to determine the type of the edge. Four edge types are possible depending on the high resolution output. They are:

Low resolution	High resolution	edge type
step	step	step
step	gradient	fuzzy
step	texture	complex
step	uniform	faint

The complex type covers the situation where a single edge at low resolution is a result of the interaction of several image entities at the higher resolution, for example, a bar of the appropriate width could produce this type of edge.

In the next section, the algorithms and implementation details of this method of combining the edges from two different resolutions will be described.

6.2 The Adjacent-Level Comparator

We can now proceed to investigate in detail techniques for combining the outputs of two SLSs run on different-resolution versions of an image, based on the general principles described in the previous section. The process for combining the two SLS outputs is called an "adjacent-level comparator", since it essentially compares the step edges found in one resolution with the coresponding intensity changes in the other.

The input to the adjacent-level comparator(ALC) is provided by running two instantiations of the single-level system on different-resolution versions of the same image, as shown in fig. 6.3. The lower resolution version of the image is produced from the higher resolution version by averaging 2*2 blocks of pixels to produce a single intensity value. This is the smallest reasonable reduction in resolution for discrete systems. It has the advantage that examining the high-resolution neighbourhood of, for instance, a low-resolution mixed pixel only involves looking at a few pixels and boundary segments, and that part of either a gradient region <u>or</u> a texture region, but not both, can exist in a particular neighbourhood. (Any two pixels in the 2*2 neighbourhood are 8-connected, and so must be in the same region, if any.)

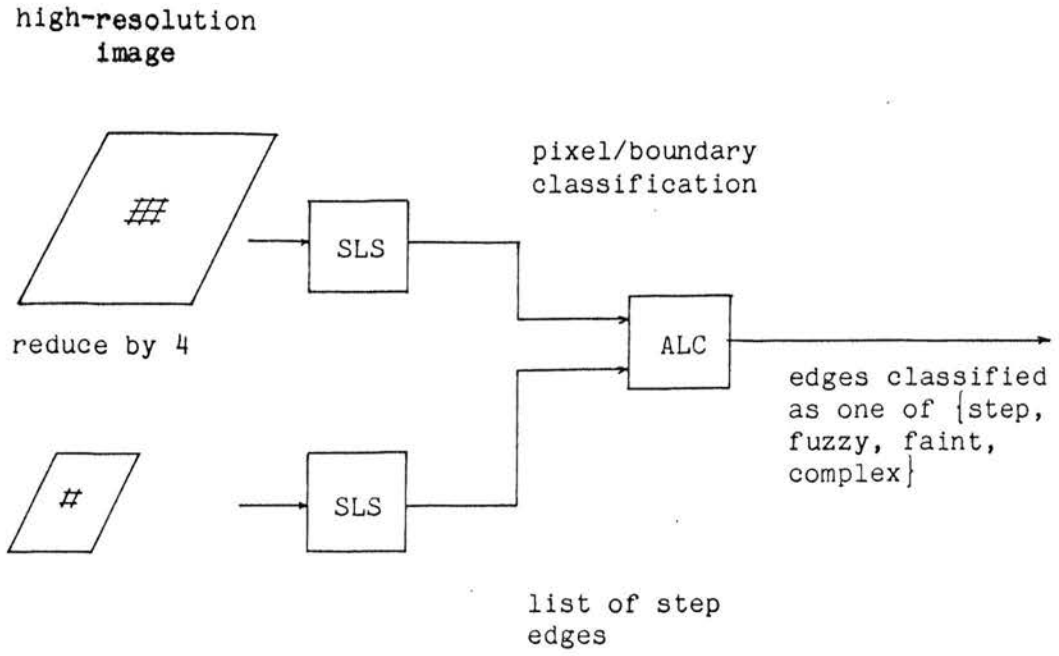

high-resolution image
pixel/boundary classification
SLS
reduce by 4
ALC
edges classified as one of {step, fuzzy, faint, complex}
SLS
list of step edges

Fig. 6.3
The input to the adjacent-level comparator is provided by the pixel/boundary classification of the higher resolution image and the list of step edges produced by the lower resolution image. The lower resolution image is obtained from the higher resolution by averaging non-overlapping 2*2 neighbourhoods to produce a single low-resolution pixel intensity.

Recall that when tracking an edge, three types of edge elements occur, mixed pixels and horizontal and vertical primary boundary segments. The high-resolution neighbourhoods corresponding to each of these are shown in fig. 6.4.

The choice of significance threshold for each resolution is important, since the two thresholds should be related to each other for best results. Various arrangements may be useful. The method described in chapter four can be used to find the threshold value for the high resolution, then choosing a threshold of half this value at the low resolution should give roughly the same probability of error (since averaging by a factor of four should reduce the noise standard

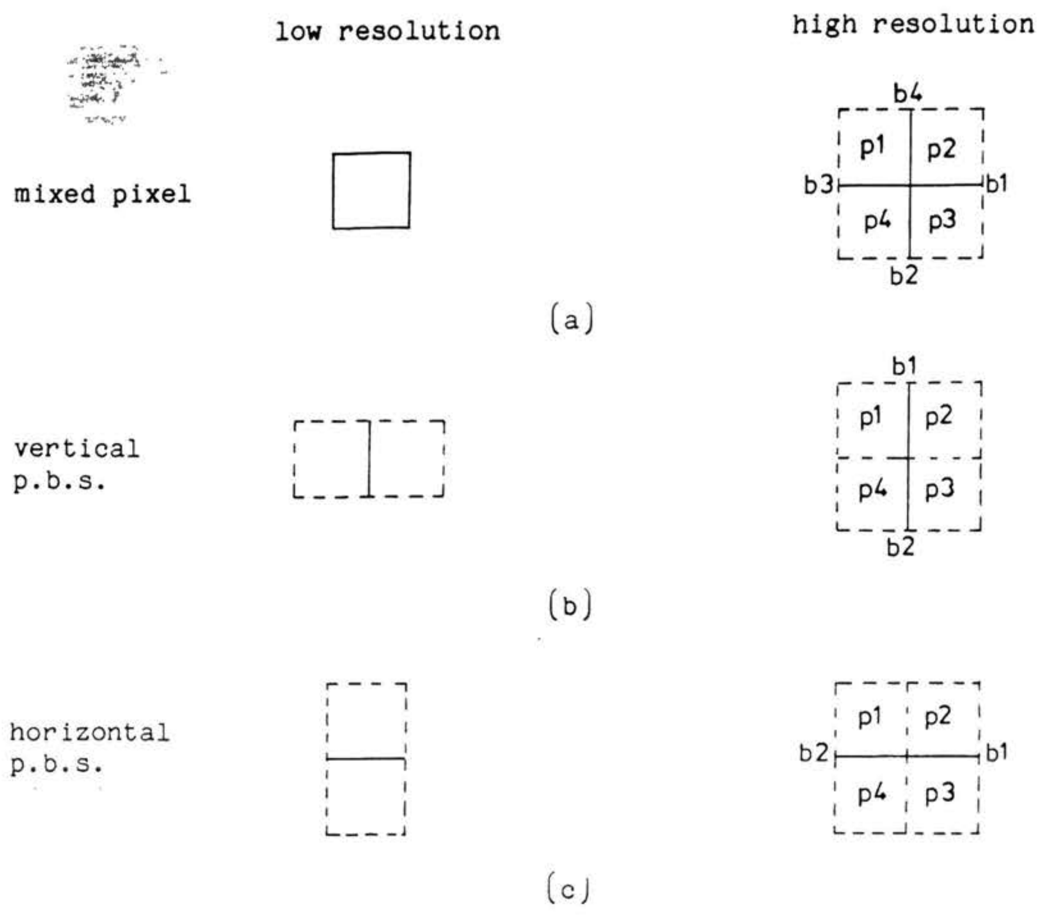


Fig. 6.4
From investigation of how the three types of edge element occur and change with resolution, the high-resolution neighbourhoods which must be investigated for each type are shown above. For a low-resolution mixed pixel, the appropriate high-level neighbourhood consists of the corresponding four high-resolution pixels (p1-p4) and four boundary segments (b1-b4). For the low-resolution primary boundary segments, the high-resolution neighbourhood to be investigated consists of two boundary segments (b1-b2) and four pixels (p1-p4).

deviation by a factor of two). Alternatively, the entropy method can be applied to the low resolution, and the high-resolution threshold chosen to maintain a constant probability of error. Or the entropy method can be applied separately to both resolutions to produce both thresholds.

Possibly the simplest approach is to use the same threshold with both resolutions, leading to a considerably smaller probability of error at the lower resolution (i.e. fewer spurious edges generated by noise or true edges removed by noise). The effect of this on a test image is shown in figs. 6.5 and 6.6. Fig. 6.5 shows the high resolution pixel classification in which the background, using the threshold chosen, is fairly noisy. By using the same threshold with the low resolution, as shown in fig. 6.6(a), the noise is effectively removed.

The input to the adjacent-level comparator in this case is provided by the high-resolution classification of fig. 6.5, stored as an array, and the list of step edges produced by running the SLS on the low-resolution image as shown in fig. 6.6(b), stored as a list of lists of coordinate pairs. In fact, with this simple image only a single edge was produced.

To classify the edge as one of {step, fuzzy, faint, complex} the adjacent-level comparator (ALC) retraces its path along the step edge produced at the lower resolution. As the edge is tracked, the ALC maintains four tallies, one for each of the four edge types listed above. At each location it examines the neighbourhood in the high-resolution classified array, as shown in fig. 6.4, corresponding to the edge element under consideration. Based on the contents of the neighbourhood, two points are awarded at each location. This is because up to two of the four types of entity in the SLS output can occur in any one neighbourhood. When the complete edge has been tracked, the edge is labelled as one of the four types based on the final values of the four tallies.

```
3NN NNN NN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNMNNNNNNNNNNNNNNNNN N1
W                                                         S      E
                                       2W                 1W     E
W                                      N                         E
S                                                                E
3                         S       EW  S                          E
W      EW        E4       N           NS                         E
W                 N                    1W                        E
W      EW             EW    S                                    E
W              S           N                                  E4 E
W   S          N              S              EW              1WE
W   N                         N      S    EW                     E
W                      2W            1W        2W                E
W S       S            N                       N                 E
W N       N         S  2W                                        E
W M                 N  N           EW                            E
W                                                                E
W                  EW                                    S       E
W                                     S  SS   EW         N     EWE
W      S  EW         EW                N  NN                     E
W      N                             S    S      2MM    S   EW   E
W                   S                N    N      N      N        E
W                   N                                            E
W                                                             S  E
M                       EW               2W             S    S N E
W                                        N 2MSMM        14   N   E
W                                         23 N 1MM4S    N        2
W                                        E3      N1MMSS24      24M
W                 S     EW               2W            N1MMMMMM3N1
W                 N                EW    M                       E
W                              EW        M                       E
W       EW         EW        EW        EW                        E
W                                      EW                        E
W          E4                          EW                        E
W           N  M          M              M                       E
W                                        M                       E
W                              M        E3                       E
W                         SSMSMM4       EW                       E
W       S  S  S       SSSM3N N   1MMMMSS2W                       E
W       1W N  N      23NN             NNN                        E
W                   23                                           E
W                  M3                                            E
W                 23                                             E
W    EW          23                                              E
W     S         23                                               E
W     N       S M                                                E
W             1M3                                                E
W             23                                                 E
W           2MM                                                  2
W           N23                                             S 2MMM
W            M                                     S  S2MMMMNMN  E
W    EW  S 23                                SS2MMNM13     E3    E
W         1MM                         SSS2MMMM3NN       S        E
W            M                  SSSSMM3NNN             E3     S  E
W        EWE3           S S S2MMNNNN                   S     SN  E
W    EW   EW     SSS2M3MNMNM         EW                NS    N   E
W   S     ME4 2MMMNNNN               EW                E3      EWE
W   N     1WNMN                      S             EW            E
W                       S            N                           E
W                    S  N                 S                      E
W             M4     1W                   N                      E
W              N                   EW                            E
W                                                                E
4SSSSMSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSMSSSSSSSSSSSSS2
```

Fig. 6.5
This shows the high-resolution pixel (and boundary segment) classification of the test picture using a significance threshold which produces some noise in the background.

(a)

```
3NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN1
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                              E
W                    SSS       E
W                   M3N1MM4M SS2
W                   M     N1M3N1
W                  E3          E
W                  EW          E
W                  EW          E
W           SSSS  2W          E
W         2MM3NN1MMM           E
W        2M3                   E
W       2M3                    E
W      2M3                     E
W      M3                      2
W     23                    SS2M
W     M               S2MMMMNNN1
W     M         S2MMMMNN       E
W     M SS2MMMMMNN             E
W     MM3NN                    E
W                              E
W                              E
4SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS2
```

(b)

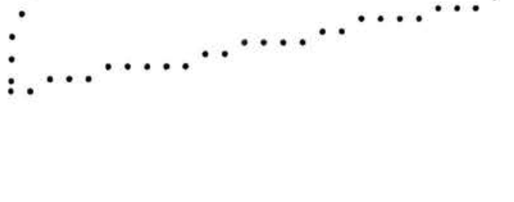


Fig. 6.6
The low-resolution classification produced by using the same threshold as was used to obtain the high-resolution classification shown in fig. 6.5 is shown in (a). Because the threshold is now relatively larger with respect to the noise standard deviation, the spurious boundary segments have been removed. Edge tracking on this classification produces the single step edge plotted in (b).

The details of the algorithm for awarding the points are somewhat complicated, but can be summarised as follows. They are derived from an examination of what happens at the places in the image where different entities meet. The basic idea is to look for evidence of step, fuzzy, and complex edges in the high resolution image. Lack of evidence is taken to indicate that the appropriate edge type is "faint".

First, consider the high-resolution classification neighbourhood corresponding to a low-resolution mixed pixel, using the notation of fig. 6.4. If any two or more of the pixels, p1-p4, are of the same type, i.e. one of {mixed, gradient, texture}, then both points are awarded to the corresponding edge tally, {step, fuzzy, complex}. If two pixels are of different types, one point is awarded to each of the corresponding edge tallies. If none of the above holds, the four boundary segments, b1-b4, are examined. If one or more of these is significant then the step edge tally is increased appropriately. Finally, if the total number of points awarded at this stage is less than two, the remaining points are awarded to the faint edge tally. So at each edge element location exactly two points are awarded.

The procedure for vertical primary boundary segments is similar. Three scores are used for step, fuzzy, and complex edge types. The two boundary segments, b1 and b2, are examined first. The step edge score is set equal to the number which are significant (0, 1 or 2). Next, the four pixels, p1-p4, are examined and the three scores updated according to the type of the pixels. Now if any score is two or more, both points are awarded to the corresponding tally, otherwise points are awarded according to the scores (if no score is two or more then the sum of the scores cannot be greater than two). However, if the sum of the scores is less than two, the tally for the faint edge type is updated accordingly to bring the total number of points awarded to two.

The procedure for horizontal primary boundary segments is identical to that just described if the high-resolution neighbourhood is relabelled as shown in fig. 6.4(c).

When the end of the edge is reached, the four tallies are used to classify it as one of {step, fuzzy, faint, complex}. The classification procedure is simple. If the step tally is greater than 3/4 of the total number of tallies, then the edge is labelled "step", otherwise the edge is labelled as being the type of the largest of the other three tallies. In fact, it may be better not to classify the edge at this stage but to pass each edge list to the next process with its full set of tallies. This will be discussed in the next section. For a more detailed description of ALC operation, see appendix 3 where a simplified POP2-like implementation of the algorithm is listed.

To gain some idea of how the two resolutions are related in practice, the two classifications can be superimposed. The test picture is shown in this way in fig. 6.7. Each box shows four high-resolution pixels and one low-resolution pixel, the central one. As can be seen, the step edge produced by the object boundary produces edge elements at both resolutions in the same place, while the noise only produces edge elements at the high resolution.

To see the effect of using a different threshold, fig. 6.8 shows the high-resolution classification produced by using a relatively high threshold on the cup picture. By using a low-resolution threshold of half the high-resolution value, to maintain approximately

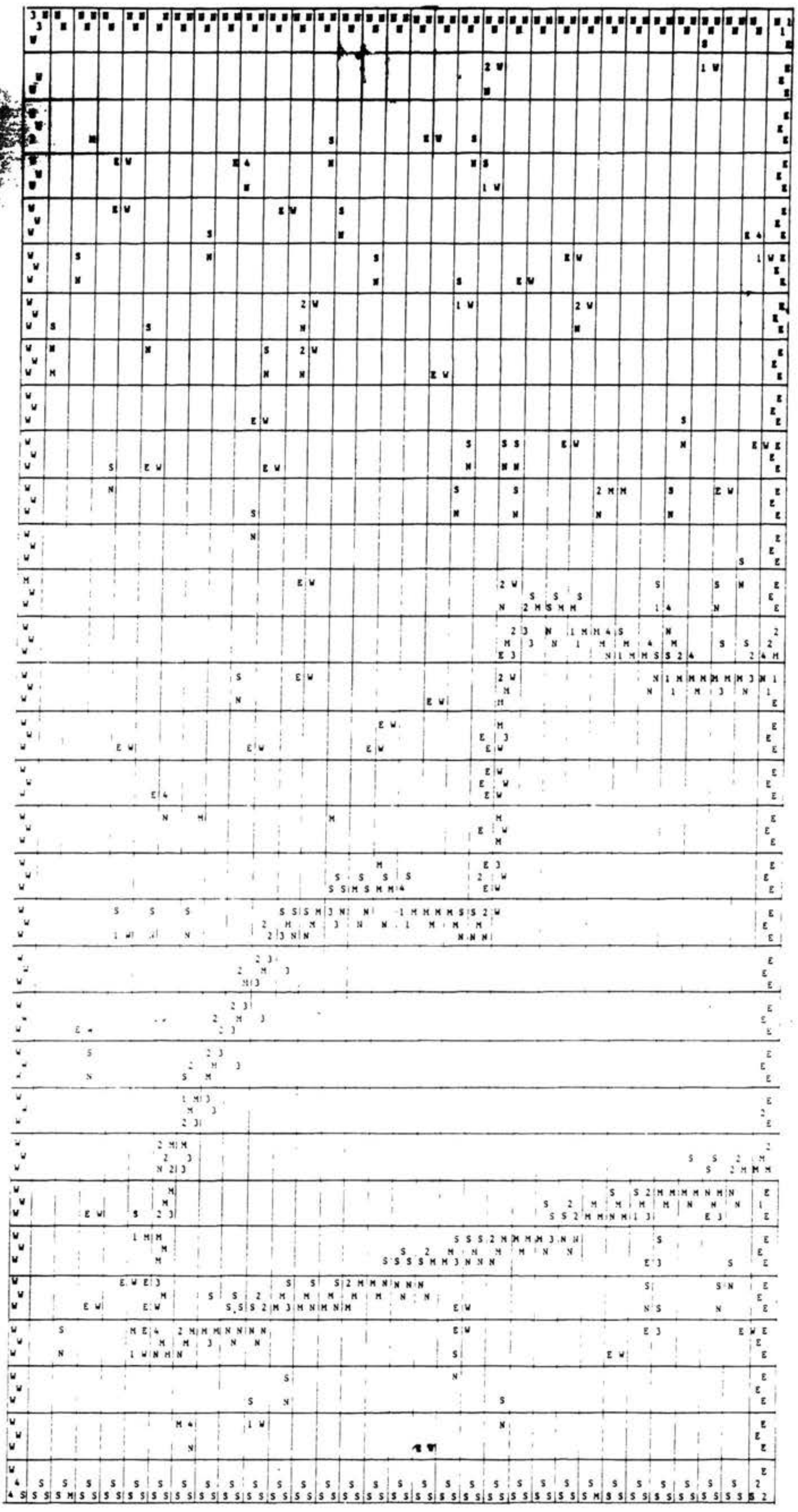


Fig. 6.7
This shows both the high- and low-resolution classifications of the test picture superimposed. Each box contains four high-resolution pixels, in the corners, and the corresponding low-resolution pixel, in the centre.

```
3NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN1MM14SS
W                                                                 NN1
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W               SSSSSSSS                                            E
W            SS23NNNNNN1WSSSSS                                      E
W          S23NN         NNNNNSS                                    E
W        SE3N                 N14                                   E
W       23                      14S                                 E
W      23                        N1W                                E
W     E3                          E4                                E
W     M                            1W                               E
W     EW                           EW                               E
W     M                            EW       S                       E
W     M4                           EW  S X SNS                      E
W    EW14                          2W MNMMMN 1W                     E
W    EW MM4                      2M3   23 14 EW                     E
W    EW  1M4                   M2MM   23   1WEW                     E
W    EW   1MM4                2MX3   E3    EWEW                     E
W    EW      NMMM4S     S S2MMM      EW    2WEW                     E
W    EW          NNMMMMMNMNN         EW   23 2W                     E
W    EW                               M4 23 E3                      E
W     EW                               N N  2W                      E
W     EW                                   23                       E
W     EW                            M4SSSS23                        E
W     EW                          EW NNNNNN                         E
W      2W                         EW                                E
W      1W                         2W                                E
W      E4                        E3                                 E
W       1W                                                          E
W                               EW                                  E
W                                                                   E
W                                                                   E
W                                     SSSSSS                        E
W                                     NNNNNN                        E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
W                                                                   E
4SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS2
```

Fig. 6.8
This figure shows the high resolution classification of the cup picture obtained using a fairly high threshold value.

constant error probability, the low-resolution classification and step edges shown in fig. 6.9(a) and (b) respectively, are generated. Notice that the edges produced by the cup handle which were separate in the high-resolution classification have interfered to produce a texture region at the low resolution. Since the low-resolution threshold is much more sensitive, it finds substantially more of the cup outline and also finds some shadows which were too faint to be picked up at the higher resolution. These are labelled as faint edges by the ALC. By examining the superposition of the two classifications, as shown in fig. 6.10, it can be seen how the edges produced by the cup handle interfere at the low resolution to produce a texture region.

The output of the ALC consists of an edge list for each of the four types, each individual edge being represented by its low-resolution list of coordinate pairs. In the cup example, the edges found at the bottom of the cup are labelled "faint" as shown in fig. 6.9(b), while the others are labelled "step".

As a final example of the use of the ALC, we consider its response to a blurred edge. A suitable test image was generated by placing some light-coloured shapes on a dark background and defocussing the camera slightly to blur the edges. The pixel classification of a 64*64 neighbourhood of the (256*256) test image is shown in fig. 6.11. The blurred edge produces gradient regions as expected. With the threshold value chosen, some noise can be seen.

The effect of reducing the resolution of the image section used above to produce a 32*32 neighbourhood is shown in fig. 6.12. The classification of fig. 6.12(a) was obtained by using a threshold half the value of that used to obtain fig. 6.11. It was suggested above

(a)

```
3NNNNNNNNNNNNNNNNNNNNNNNNNM1MMM
W                             1
W                             E
W                             E
W                             E
W    SS2MMMM4SSS              E
W   2313    NNN14             E
W  E3            14           E
W  EW             1W          E
W  2W              M4         E
W  MX4            2TTTTM      E
W  MMM4        2XMMMTTTT      E
W EW MMMM4   2MMX3E3 TT       E
W  M     MMMMN     MTTT       E
W  M              2MTTT       E
W  M             MMM2T3       E
W  14            M3 N         E
W   M           23            E
W   M           M             E
W  XX           M424  SS      E
W  MMM4       S  NNMMM3N      E
W  EW M4     SN               E
W      MMS SMN  SSSSSSSSS     E
W        NMN  S NNNNNNNNN     E
W            N                E
W                             E
W      S                      E
W      NMMMMMMSS              E
W           NN                E
W                             E
W                             E
4SSSSSSSSSSSSSSSSSSSSSSSSSSSSS2
```

(b)

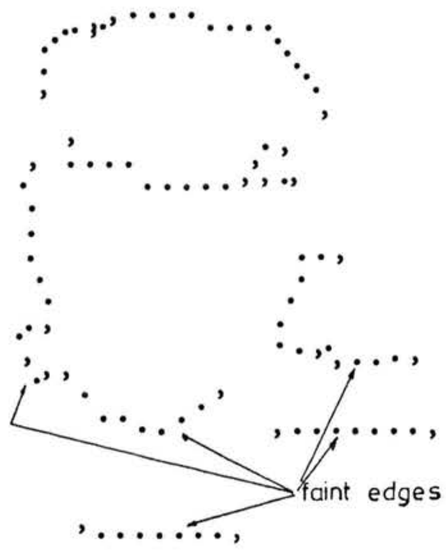


Fig. 6.9
The low-resolution classification generated by using a threshold chosen to maintain the same probability of error as that in fig. 6.8 is shown in (a) above. Because the noise is reduced by a factor of two by averaging, the threshold value is halved and the system is more sensitive. Applying the edge tracker to this classification produces the edges shown in (b).

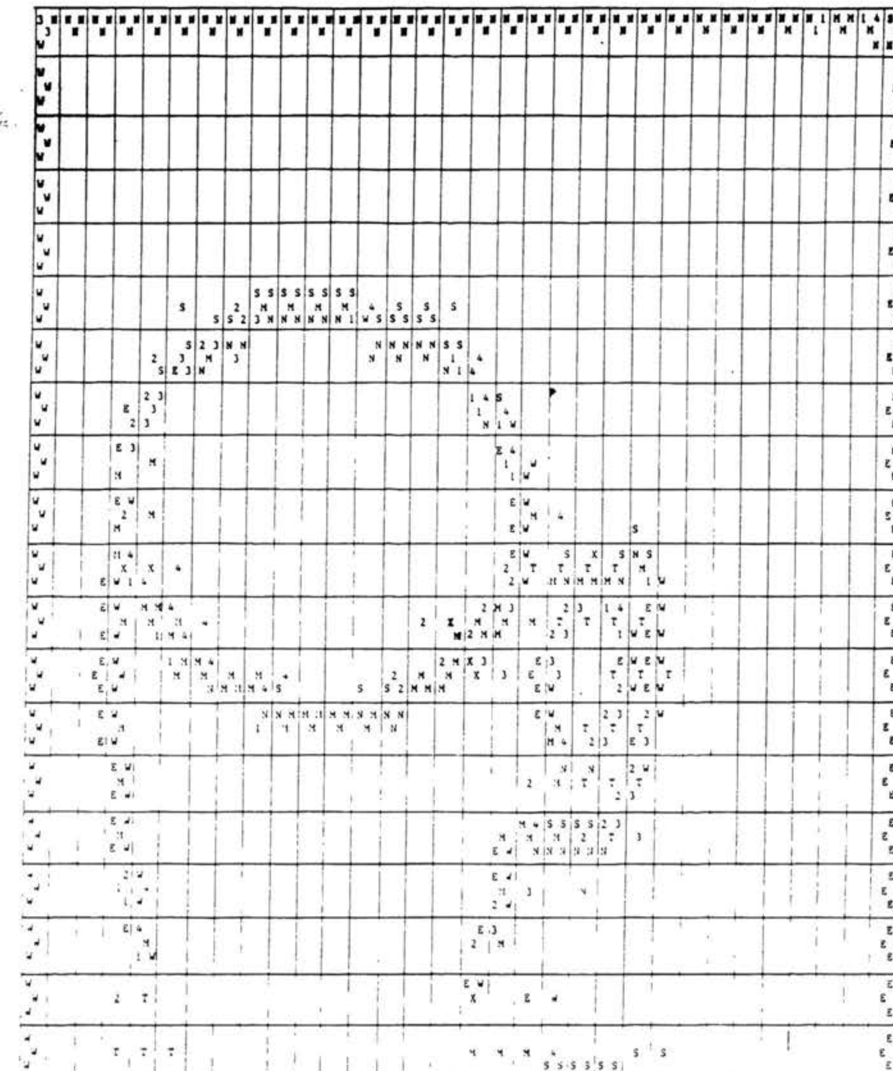


Fig. 6.10
This shows the high- and low-resolution classifications of the cup picture superimposed. Of particular interest is the region containing the handle of the cup, just above the centre right of the picture. The edges, generated by the obscuring boundaries of the handle, are just sufficiently far apart to be found separately at the high resolution, but at the low resolution they interfere to produce a texture region.

```
3N1M3NNNNNNNNNNNNNNNNNN1MGGM3NNNNNNNNMN13NNNNNNNNNNNNNNNMMMMMMN1
W     S          EW     2GGGG                EW          1XX31MM
W     N                 GGGG      M                  S     1W241
W                       1GGGG   EW S   S             N   S E41XM
W                        GGGM      14  N                 N  1MMM
W                      EW G EW  S   M     EW   S  EW         EW1
W                        GG EW  N   M          1W      M     S E
W                        GGGG          S                     N M
W   S                    GGGGM         N     EW    M       2W  M
W   1WS                   GG                M       S      N   E
W     N                EW  G           EW      EW   N        S E
W         EW           EW GGG                      E4        N E
W                 S       GGG M   M         S   EW  N          E
W                E3      GGGG            S  N                  E
W   EW                 EW GG E4          N  EW     S           E
W                         GG  N      S             N           E
W          EW      S      GGG        1W   EW                   E
W                  1W     GGGG             E4                  E
WEW                       GGG EW         24 N                  E
W                         GGGG EW    M  2M3              SS    E
W                    S    GGG        1M N              2W13    E
W                    N    MGM   EW  M                  NE4     E
W         M              G4EW S     1W     EW      M    SN     E
W  S   EW               EWGM E3       M       M 2W      N      E
W  N                     GGG   S            S   N      S       E
W S                       GG   N   M        N          N EW    E
W N                      GGGG        E4          EW       S    E
W                        GGGGG        N     EW            N    M
W                        GGGG  M   EW       EW   S             E
W                         GGG  S                 N    M     MMMM
W                      EW MGM  N                            1W E
W             S        EW G  M     M                           E
W             N           MG                          EW       E
W                         G EW                        S EW     E
W                         GGM       S    S            N       SE
W            EW  S       GGG EW     N    N                    NE
W                N       GGGG               S  EW          EW  E
W                         GGMM EW          E3                  E
W2W                       G  MM4            E4        EW       E
WN   EW                   MM4EWN     EW EW   M EW     2W       E
W                        X  G    S               2W   N        E
W                         MGG   M3               N 2W          E
W                         GGGG            EW       N           E
W                         GGG  EW        EW      S             E
W            EW  EW       GGG               S    N EW   EW  S  E
W            EW           GGGG              N S             N  E
W                          GGG        M   EW  N              EWE
W              S          GGG3         S                S      E
W              N          GGG  S       N                N      E
W                       EWGGGG N   EW                          E
W                         GGG S         EW        EW EW        E
W                        GGGG N               S                E
W                      S  GGG    2W   EW     M3      EW        E
W                 S    1W GGG    N           EW                E
W            EWS  N       GGGG  EW             S               M
W              N          GGG                  1W              E
W                         MG EW   M                      S S   E
W               2W       X  G                  EW        1W1W  2
W               N         MGG     EW                 S         1
W                       EW GGG               S       N         E
W                       EW GG             S  N                 E
W                         MGM          E4 N       M            E
W   EW                   M  M           N                      E
4SSSSSSSSSSSSSSSSSSSSS242M42M4SSSSSSSSSSSSSSSSSM4SSSSSSSSSSSSSS2
```

Fig. 6.11
This shows the pixel classification of a 64*64 picture neighbourhood containing a vertical blurred edge.

```
3NNNNNN13N1MGM3NNNN13NNNNN1MMGM
W          GGG  S         1GGG
W          GGG 23 S  S     1GM
W          GGG M  N  1W   S 1M
W          GGG           23  M
W          GGG          MN   E
W          GGG    S          E
W          GGG    N          E
W   EW     GGG               E
W          GGG               E
W EW       GGG            M  E
W      EW  GGG               E
WS         GGG        M      E
WN         GGG  EW  S E4 2W  E
W          GGG      N  N  N  E
W     S    GGG EW            E
W     N    GGG               E
W    EWS   GGG               E
W     E3EW GGGG              M
W          GGG           S   1
W           GG        24E3   E
W    EW  EW GG        M3     E
W     EW    GG               E
W     S    GGG       2W      E
W     N    GGG  M    N     EWE
W          1GG          S    E
W     S    GGG   EW     1W   E
W     1W EW GG               E
W          GGG EW          EWE
WEW   EW   GGG      SSEW     E
W           GG     E3N       E
4SSSSSSSSS2MMM4SS24SSSSSSSSSS2
```

(a)

```
3NNNNNNNNNN1MG3NNNNNNNNNNNNN1MMM
W           GGG              1XM
W           GGG               1M
W           GGG                M
W           GGG                1
W           GG                 E
W           GGG                E
W           GGG                E
W            GG                E
W            GG                E
W            GG                E
W           GGG                E
W            G                 E
W           GGG                E
W           GGG                E
W           GGG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
W            GG                E
4SSSSSSSSSSS2MM4SSSSSSSSSSSSSSS2
```

(b)

Fig. 6.12
The classification shown in (a) above, resulted from reducing the resolution of the neighbourhood used to generate fig. 6.11 and applying a threshold of half the value used in fig. 6.11. This maintains an approximately constant error probability. The classification in (b) was obtained using the same threshold as that used in fig. 6.11. This removes the noise, but also reduces the sensitivity.

that this should maintain a roughly constant error probability, and this does seem to be the case here. The classification of fig. 6.12(b) was obtained using the same threshold as that used in fig. 6.11. Although this reduces the sensitivity of the system, as can be seen from the smaller average width of the gradient region, it also decreases the error probability, in this case removing all of the noise. Comparing fig. 6.11 with fig. 6.12(a), we see that the average edge width is smaller due to the reduction in resolution.

Further reducing the resolution has the effect shown in fig. 6.13, where the edge width is less than one pixel, resulting in the edge being found as a step. Hence, if this resolution and the next higher one were used to provide the inputs to an ALC, the edge would be found and tracked as a step at the lower resolution. When the higher resolution neighbourhoods corresponding to the step edge were examined, the edge would be labelled as type "fuzzy" due to the gradient region.

```
3NNNN1M3NNNNN1MM
W     M       1M
W     M        1
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
W     M        E
4SSSS2MM4SSSSSS2
```

Fig. 6.13
Further reducing the resolution causes the edge width to become less than one pixel. Since the thickess constraint is now violated, but the thinness constraint is now satisfied, the edge is found and tracked as a step.

Many blurred edges are not of uniform width. Because they are normally generated by shadow or surface orientation boundaries, the resulting edge width can vary considerably. In fact, if the 256*256 test image used above is reduced to 64*64 pixels, some of the edges are not of uniform width as can be seen from the classification shown in fig. 6.14. However, if we further reduce the resolution to 32*32 pixels, then all edges in this image become steps, as shown in fig. 6.15. This illustrates the general principle of the multi-level system. All blurred edges will eventually become steps at some resolution, if they are not subject to interference.

The fact that blurred edges have variable width results in different parts of the edge becoming steps at different resolutions, as happened in the example just discussed. In the final output representation of the system, this behaviour should be taken into account, and it should be possible to relate the different parts of the edge across resolutions. The form of a suitable representation to do this will be discussed in the next section.

6.3 The Multi-Level Integrator

By taking a particular image, reducing the resolution as described in section 6.2, inputting both images to single-level systems, and feeding the outputs into an adjacent-level comparator, edges of 1-2 pixels width can be found and represented. The structure of this process was shown in fig. 6.3. If the low-resolution image produced above was further reduced in the same way, and the two lower-resolution images fed into a second ALC, then edges of 1-4 pixels width could be found. The problem is that they would be in two different representations, i.e. the outputs of the two ALCs. So to find edges of a range of widths, it is necessary to run several

```
3NNNNN13NNNNNNNNNNNNNNNNNNNNNNN13NNNNNNNNNNNNNNN13NNNNNNNNNNNNN1
W     EW                       EW              EW              E
W     EW            EW         EW              EW              E
W    EW                        EW                              E
W    EW                        EW                              E
W    EW            EW          EW                              E
W    EW                        EW                    E4        E
W    EW                        EW                    E3T4      E
W    EW                        EW                     TTT4     E
W    EW                        EW                     MTMM4    E
W    M                         EW                     M 1MM4   E
W    EW                        EW                     M  1MM   E
W    EW                        EW                     M   1M4  E
W    EW                        EW                     M    1M4 E
W                              EW                     M     1M4E
W                              EW                     M      MMM
4                              EW                     M      1XM
M4          S                  EW                     M       1M
MX4         N                  EW                     M        1
MMM                            EW                     M        E
W1MM                           EW                     M        E
W 1M4                          EW                     M        E
W  1M4                         EW                     M        E
W   1M4                        EW                     M        E
W    1M4                       EW                     M        E
W     MM4                      EW                     M        E
W      1MM                     EW                     M        E
W       1M4                    EW                     M        E
W        1M4                   EW                     M        E
W         1M4                  EW                     M        E
W          1M4                 EW                     M        E
W           1M4                2W                     MG       E
W            1M4              2M                      GG       E
W             1M4             XM                      GG       E
W              1M4           2XX                      GG       E
W               1M4         2MMM                      GG       E
W                1M4        MM3EW                     GG       E
W                 1MM      2M3 EW                     GG       E
W                  1MM4    MM  EW                     GG       E
W                   1MM4  MM3  EW                     GG       E
W                    1MM42M3   EW                     GG       E
W                     1MMMM    EW                     GG       E
W                      1XX3    EW                     GG       E
W                              EW                     GG       E
W                              EW                     GG       E
W                              EW                     GG       E
W                              EW                     GG       E
W                              EW                     GM       E
W                              EW                      M       E
W                              E4                     2M       E
W                               MGGGGGGMMMMMMMMMMMMMMMM3       E
W                               MGGGGGGG  X                    E
W                              E3                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              EW                              E
W                              E4                              2
4SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS2MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
```

Fig. 6.14
This shows the classification of the 64*64 version of the test image. Note that both the right vertical edge and the horizontal edge are not of constant width.

```
3N1M3NNNNN13NNN13NMNNNNM3NNNNNN1
W EW           EW M  SSM       E
W EW     EW    EWM   NN        E
W EW     EW    EW          M   E
W EW     EW    EW          MT4 E
W EW     EW    EW          TTT E
W EW    SEW    EW          MTM4E
W E4   SN      EW          M 1MM
4  14MSN       EW          M  1M
M4  N N        EW          M   1
MM4            EW          M   E
W1M4           EW          M   E
W 1M4          EW          M   E
W  1M4         EW          M   E
W   1M4        EW          M   E
W    1M4       2W          M   E
W     1M4      M           M   E
W      1M4    2M           M   E
W       1M4  2MM           M   E
W        1M42M3EW          M   E
W         1MMM EW          M   E
W           X3 EW          M   E
W              EW          M   E
W              EW          M   E
W              E4         2M   E
W               MMMMMMMMMMMM   E
W              E3              E
W              EW              E
W              EW              E
W              EW              E
W              E4              E
4SSSSSSSSSSSSSS2MMMMMSSMSSSSSSS2
```

Fig. 6.15
Further reducing the resolution of the test image to 32*32 pixels produces the classification shown above. At this level, all of the blurred edges have become steps.

versions of the ALC, as shown in fig. 6.16. To find broad edges the resolution has to be reduced, but to make sure all the sharp edges are found the highest resolution must also be used.

Hence, it is necessary to propose another process which takes the multiple ALC outputs, and forms a single description of all the edges in the image. In other words, the overall system structure should be as shown in fig. 6.17. The new block is called a multi-level integrator (MLI) because its job is to take the multiple

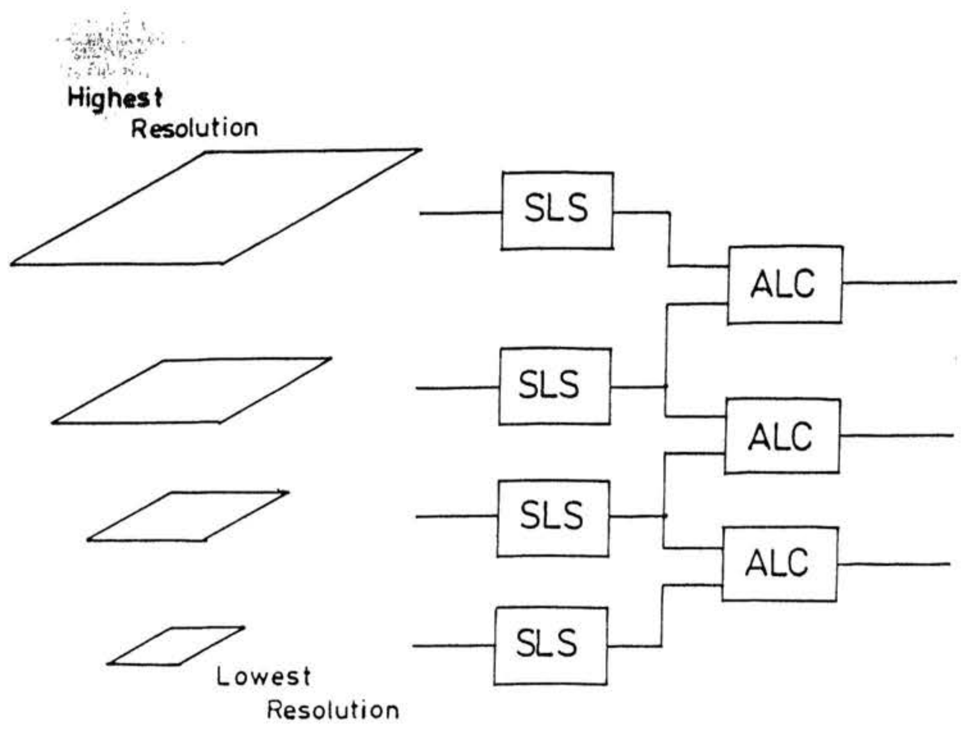


Fig. 6.16
To find edges of a range of widths, multiple applications of the SLS are required. Each pair of adjacent SLSs feed an ALC which compares the output at the two resolutions.

outputs produced by ALCs and form a single representation.

In view of the need for this additional block, one might question the need for the ALC. However, if we simply fed the outputs of several SLSs into the MLI we would still need some way of relating the outputs of different SLSs. I believe it is best to use a uniform approach, and form links between adjacent resolution SLS outputs before attempting the integration. This is the role of the ALC - to provide an explicit link between the outputs of two SLSs, where the SLSs have been fed with images related in the way specified in section 6.2. Once these links have been formed, the job of the MLI is greatly simplified. Currently, the ALC classifies each low

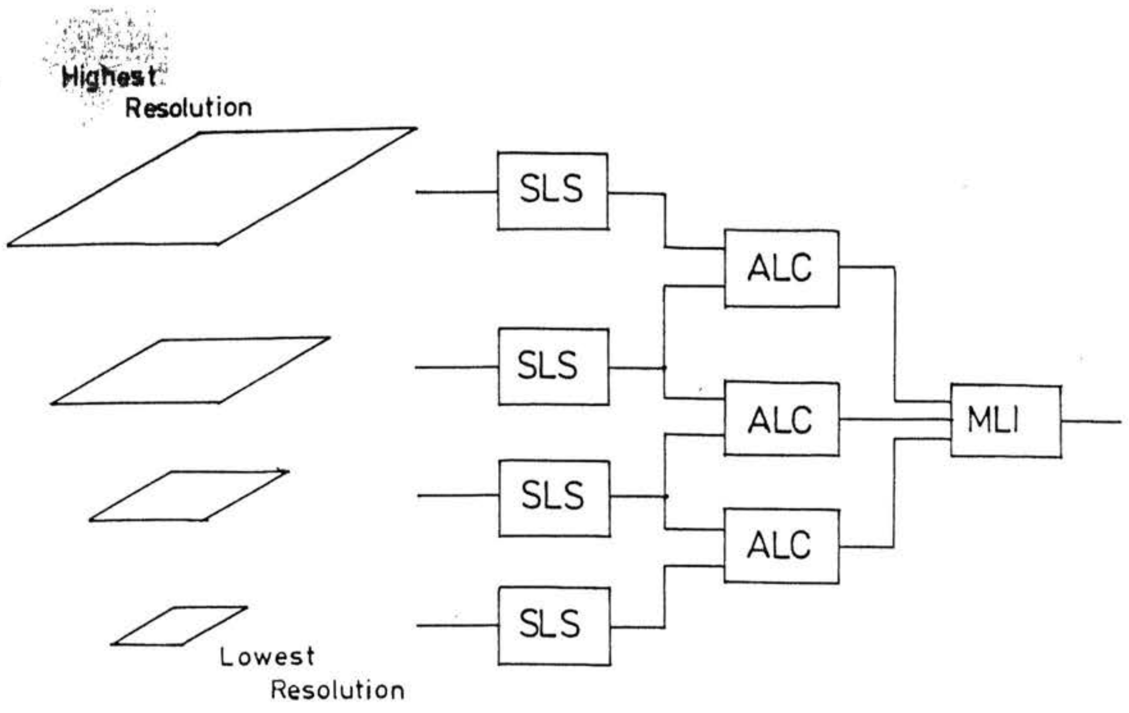


Fig. 6.17
The final system architecture contains three layers. Using the links between resolutions provided by the ALC, the multi-level integrator constructs the final data structure.

resolution edge as one of four types. It may be better to pass the four tallies to the MLI with each edge, rather than a single interpretation, which would give the MLI more scope in making decisions over several resolutions in difficult cases.

This use of the ALC is an example of a useful modularisation technique. Where several entities, in this case SLS outputs, have to be compared with each other, it greatly simplifies the comparison process if the entities can be compared in pairs, rather than in total. Additionally, since the relationship between the outputs of adjacent SLSs is the same for each pair, the same comparison process should be applied in each case.

In summary, although it is necessary to use several SLSs if a wide range of edge widths is to be dealt with, it is better, due to the relationship between the SLSs, to compare them in pairs. This simplifies the process of comparison, increases the modularity of the overall system, and constructs the explicit links necessary for successful multi-channel operation.

In general terms, the role of the multi-level integrator is to take the multiple sets of edges produced by the adjacent-level comparators used and relate them in a single representation. If no single representation was available, the constraint analysis system would not be able to identify the set of edges (i.e. one at each resolution) resulting from a particular scene entity.

The final output representation should make constraint analysis easy. Therefore, to obtain greatest accuracy, step edges should be represented at the highest resolution. Blurred edges, to be tracked and manipulated easily, should be represented as step edges at the resolution at which this occurs, but, for more detailed investigation, the simple transforms between levels make it easy to find the corresponding neighbourhoods of higher-resolution images. An edge of non-uniform width should be held as a step at the appropriate resolution with pointers to pieces of the same edge which are steps at higher resolutions.

This representation, used in conjunction with the original image, would provide the input to a set of processes whose purpose is to generate a full semantic labelling of image edges. Reflectance edges could be labelled using the test described in section 2.6. Something similar to Ullman's "S" operator (Ullman 1976) could be used to identify shadow and highlight edges, while a variant of

Witkin's technique (Witkin 1982) would provide a method for labelling obscuring edges. Notice that the multi-level edge representation can play a useful part in this edge labelling process. For example, because of the way in which they are generated, obscuring edges should exist at the highest resolution, whereas shadow and surface orientation edges will often be slightly blurred thereby causing gradient regions at the highest resolution. Hence, the test to find obscuring edges doesn't need to be applied to edges labelled "fuzzy" in the multi-level representation. This also helps improve the efficiency of the edge-labelling process. The test for obscuring edges, for example, should be applied to edges labelled "step" first, while the test for shadow edges should start with edges labelled "fuzzy". Notice that, strictly speaking, the edge labelling processes cannot assert that an edge is definitely of a specific type. All they can do is assert that an edge satisfies the constraints which should be met by an edge of that type. The multi-level integrator has not been implemented because, apart from time constraints, it is difficult to know in detail what the next stage in the process, in this case constraint analysis, needs in the form of a representation. This claim is not inconsistent with the earlier use of a hypothetical constraint analysis system to derive the goals of edge detection, because like vision processes, representations can also be thought of at different levels. We can use Marr's (1982) three levels: In this case the theory level might involve considering what is being represented and why, and investigating the structure of the representation and its properties in a mathematical sense. The algorithm level would obviously be concerned with setting up the appropriate structures and accessing data items. The implementation level would contain the implementation of the algorithms in the appropriate software or hardware.

An example serves to illustrate this point. Witkin(1982) has described a method for distinguishing obscuring and shadow edges. A set of image curves parallel to the edge in question are formed at various distances from the edge. By correlating the intensities on these curves between curves, the type of the edge can be estimated from the variation in the correlation parameters as the edge is crossed. Ullman (1976) and Forbus (1977) have described a method for identifying shadow and highlight edges. In this case, a set of paths perpendicular to the edge are required along which a particular operator is applied. Hence, we see two techniques of edge labelling, both based on investigating intensities in the neighbourhood of the edge, which require substantially different edge representations. In one case, a set of curves (with intensities) parallel to the edge in question are required, in the other, a set of curves (with intensities) perpendicular to the edge are needed.

An additional problem in deciding on a suitable representation is that to date constraint analysis systems have used only single-width edges. However, we are justified in making some suggestions about the final representation, since it is clear what the next stage will require at the theory level, if not at the algorithm and implementation levels. Edges should be represented as steps at the highest resolution possible. If it is desirable to maintain a complete set of edges at each resolution, pointers can be used to link the appropriate edges, producing the tree-like structure shown in fig. 6.18, where nodes represent edges. Each edge at a particular resolution is linked to the corresponding edges in the next highest resolution. Note that not all edges will have such links, because of

interference. This structure can be generated by starting with the **lowest resolution** edge list, tracking along each edge checking the next higher resolution classified array and edge list to form the edge correspondences. In fact, the ALC could be extended to perform this task at the same time as it classifies edge types. The task of the MLI would then be to set up the final data structure.

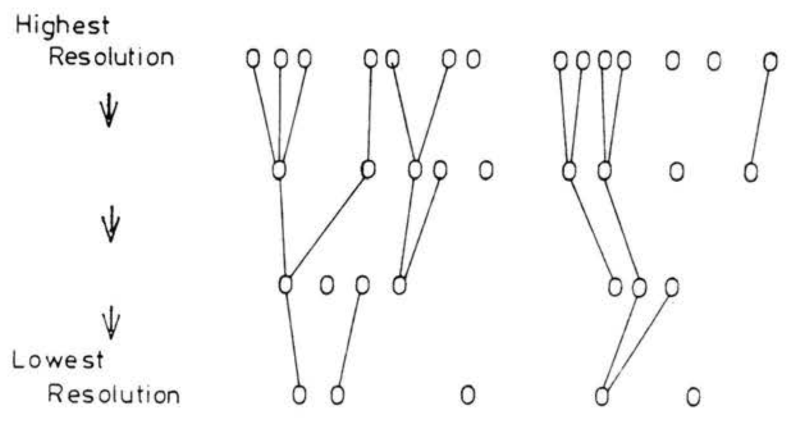


Fig. 6.18
A **tree-like** structure like the one shown above may be an appropriate final representation for the multi-level system. Each node **represents** an edge, probably stored as a list of coordinates and an associated edge type. The links are connections between the same edge at different resolutions. Hence, due to interference, not every node will link to another level.

6.4 Features Needed In Multi-Channel Systems

In this chapter an attempt has been made to identify the important features of multi-channel edge detector systems. There are three interrelated points to note:

1. The classes of entities produced by the single-channel system should be precisely defined.
2. There should be a clear relationship between the outputs of adjacent single-channel systems.
3. There should be a simple (or at least calculable) relationship between the output of the single-channel system and the underlying intensity changes.

The first two of these are needed so that the way the various entities in the channel output change as the channel parameters change can be worked out. It is necessary, but not sufficient, to define the output of the single-channel system well. It must also be defined in such a way that the effect of changing channel parameters produces a useful relationship between the channel outputs. In the system described here, this was done by describing each of the classes of channel output in terms of a few basic constraints, then examining the effect of reducing resolution on the constraints. Because the entities in the single-channel output were easy to describe in terms of these basic constraints, and because the constraints were simple image properties, the effect of reducing resolution was easy to specify.

The third point is important for two reasons. Firstly, it must be the aim of any edge detector to provide an output which is in some way clearly related to the image. Secondly, it has been by maintaining simple, well-defined relationships to the image throughout the derivation of the edge detector described here that has allowed the properties of multi-channel systems to be investigated. In fact, points 1 and 2 can be interpreted as extensions to point 3. Tying the edge detector output to the image is the key step in enabling the design of the edge detector itself, and in specifying the performance of the succeeding module in the system.

7 Conclusion

7.1 Contributions of this Thesis

The main problems in designing and implementing edge detectors are deciding what exactly is supposed to be detected, doing the detection, and reducing the effect of noise during detection. In this thesis all of these areas have been investigated.

The methodology of producing vision systems has been discussed in recent years by Marr and others. In this thesis I have tried to present a deeper investigation of some of the issues involved by explicitly considering the methodology being used while designing a specific vision system component. In particular, I have shown how it is preferable to choose a simple system architecture when one is investigating vision issues at Marr's theory level. I have also discussed the role of the succeeding process in a system when designing a particular component. In chapter two, we saw how investigating a particular edge labelling task in detail enabled several features of the required edge detector to be listed, from which suitable goals of edge detection could be derived.

Since the earliest days of computer vision, it has been recognised that image noise is a problem. In this thesis, I have described a technique for automatically selecting a threshold in the presence of noise. It takes both the noise statistics and the distribution of intensities in the image into account. However, I avoided the use of the word "optimal" to describe the threshold selected. While a particular quantity may be optimal with respect to a particular criterion, the real issue is the choice of criterion. The threshold selected by the method described in chapter four is based on modelling the addition of noise to the image by a communication channel. Finding the threshold which maximises the mutual information of the channel should give the output edge map most similar to the ideal edge map in an information-theoretic sense.

There was some difficulty in choosing appropriate terminology to describe the "edge detector" described in this thesis, because it does rather more than most systems so named. As well as finding and tracking step edges, the single-level system also locates and identifies image regions containing extended smooth gradients or texture. In this context, a computational definition of texture was given, which, although closely related to the design of the rest of the system described here, is, I believe, usefully generalisable to other systems. The edge detector was designed so that the various classes of entities present in its output were closely related to the input image. I believe this is a very desirable property of any edge detection system, particularly if it is to be multi-channel. This relationship also enabled the entities in the single-level system output to be described in terms of a few basic constraints, which proved to be very useful in constructing the adjacent-level comparator.

One advantage of the single-level system over other edge detectors is that the output contains four classes of entity (step edges, gradient regions, texture regions, and uniform regions) rather than the usual two (edge or no-edge). The way this can be exploited by any user of the system obviously depends on the application, but one obvious advantage is that step-like edges are separated from other types of intensity change.

Another advantage of the single-level system is that the four classes of entity in its output are well defined. Hence, if a step edge is represented by a given list of coordinate pairs, it will be the case that there exists an intensity change, of at least a certain amplitude, which is thin in one direction, on the image curve indicated. The fact that I could rely on knowing what kind of intensity change produced a particular entity in the single-level system output was one of the main advantages in the design of the multi-level system.

Although the multi-level system was not fully implemented, enough was done to enable three important features necessary for the design of multi-level systems to be identified. Namely;

1. The classes of entities produced by the single-level system should be precisely defined.
2. There should be a clear relationship between the outputs of adjacent single-level systems.
3. There should be a simple (or at least calculable) relationship between the output of the single-level system and the underlying intensity changes.

The straightforward nature of the design of the adjacent-level comparator would seem to reflect a successful single-level system design and support the claim for the importance of the three features listed above.

Of course, the edge detector does not perform perfectly. Like most detectors based on tracking, it is susceptible to noise generating a break in an edge, and hence causing tracking to terminate. No measures were taken to combat this because I felt that, for example, "jumping" short breaks in edges, violated the principle of veridicality in that the final tracked edge coordinate list would not faithfully reflect the image structure. On the other hand, it is clear that some measures have to be taken to combat noise. Just how far one should go is, to some extent, an open question.

To make as few assumptions as possible about the nature of the image, the simplest available method of finding intensity changes was used, i.e. first differences. In effect, this was to avoid prejudging the image. Again, this makes the system somewhat susceptible to noise. However, this is partially alleviated by reducing the resolutiion and linking edges across resolutions.

7.2 Suggestions for Further Work

Although the field of edge detection has been the subject of many research programmes over the past two decades, I feel that there is still a need for further work and I will try and suggest ways in which the research described in this thesis could be carried on.

I believe that the most important area requiring attention is in the specification and evaluation of edge detectors. In this thesis I have made a start at improving the appropriate design methodology. This could be refined and improved by trying to use the same basic ideas to design edge detectors for different tasks. For instance, it would be interesting to produce an edge detector for stereopsis using the methodology described here. This should illustrate how different requirements in the succeeding process affect edge detector design.

If further work could be expended on designing an edge labelling system, which would allow the implementation of a suitable multilevel integrator, use of the total system would undoubtedly highlight shortcomings in the current edge detector design as described here. It may then be possible to relate these to the design methodology as well as to specific problems in the edge detector.

Edge detector evaluation is closely related to specification **because we** have to know what an edge detector is supposed to be doing before we can say how well it is doing it. I discuss evaluation in this further work section because of its close links to specification which has been a topic in this thesis, and because the somewhat unconventional nature of the edge detector designed here illustrates the failings of current evaluation techniques.

Two approaches to evaluation are extant. One involves the use of a figure of merit, such as that discussed in chapter three, with very simple, artificially-generated images, such as those containing only a single ramp edge. Usually, Gaussian noise is added in varying amounts. Since the image is so simple, an ideal response can be suggested and the actual output of a particular edge detector measured via the figure of merit. The problem with this approach is that the test images are nothing like real images, so it is not at all clear that detectors with a high figure of merit will perform well with real scenes. The alternative approach, which attempts to get round this problem, involves using a set of real images. Now the problem is in measuring performance. Currently, this seems to be done on a manual basis, which is also unsatisfactory, since it is purely subjective.

The way round these problems would seem to be to merge these two techniques using modern computer graphics methods. Complex, realistic scenes could be created and used to generate test images. The big advantage is that the user could specify the desired output of the edge detector in scene/imaging terms, for instance, giving the minimum change in surface radiance across a reflectance boundary that was to be detected. These specifications could then be automatically transformed to generate a set of image edges to be matched with those found by the detector under test.

The threshold selection techniques described in chapter four could also be the subject of some further work. If a method of deconvolving the (known) noise distribution from the image distribution could be found, the technique would be a very useful one. The problem is that, while the image is a fairly large sample, it is not an infinite one, and so the problem may be ill-conditioned. However, I believe that such techniques are available in the field of image restoration, so this problem should be tractable.

In general, I believe the approach taken in this thesis has led to both the elucidation of some fruitful general principles, and the development of some useful specific techniques. Carrying on the work along the lines suggested above could further extend our understanding of edge detection and vision.

8 References

Bajcsy, R. (1973) "Computer Description of Textured Surfaces", Proc. IJCAI-3, 572-579, Stanford, U.S.A.

Barrow, H.G. and Tenenbaum, J.M. (1978) "Recovering Intrinsic Scene Characteristics From Images", p.3-26 in Computer Vision Systems, eds. Hanson and Riseman.

Beattie, R.J. (1980) "Initial Image Representation in Computer Vision", W.P.80, Dept. of A.I., Univ. of Edinburgh.

Beattie, R.J. (1982) "Edge Detection for Semantically Based Early Visual Processing", Proc. ECAI-82, 190-196, Orsay, France.

Binford, T.O. (1981) "Inferring Surfaces from Images", A.I., 17, 205-244.

Brain, A.E. (1979) "Lenses For Industrial Automation, Part One: A Brief Review of Basic Optics", S.R.I. tech. note 201.

Burstall, R.M., Collins, J.S. and Popplestone, R.J. (1977) "Programming in POP-2", Revised Edition, Dept. of A.I., Univ. of Edinburgh.

Canny, J.F. (1983) "Finding Edges and Lines in Images", M.I.T. A.I. Lab. Report No. AI-TR-720.

Duda, R.O. and Hart, P.E. (1973) "Pattern Classification and Scene Analysis", Wiley, New York.

Fischler, M.A., Barnard, S.T., Bolles, R.C., Lowry, M., Quam, L., Smith, G. & Witkin, A. (1982) "Modelling and Using Physical Constraints in Scene Analysis", Proc. AAAI-82, 30-35.

Forbus, K. (1977) "Light Source Effects", M.I.T. A.I. Memo No. 422.

Gallager, R.G. (1968) "Information Theory and Reliable Communication", John Wiley & Sons, Inc, New York.

Grimson, W.E.L. (1981) "From Images to Surfaces", MIT Press, Cambridge USA.

Hanson, A.R. and Riseman, E.M. (1978) "Computer Vision Systems", Academic Press, London.

Haralick, R.M., Shanmugam, K. and Dinstein, I. (1973) "Texture features for image classification", IEEE Trans. Systems, Man & Cybernetics, SMC-3, 610-621.

Haralick, R.M.(1979) "Statistical and Structural Approaches to Texture", Proc. IEEE, 67, 786-804.

Haralick R.M. (1984) "Digital Step Edges from Zero Crossing of Second Directional Derivatives", IEEE trans. Pattern Analysis & Machine Intelligence, PAMI-6 (1), 58-68.

Herskovits, A. and Binford, T.O. (1970) "On Boundary Detection", M.I.T. A.I. Lab. Memo 183.

Hildreth, E.C. (1980) "Implementation of a Theory of Edge Detection", M.I.T. A.I. Lab. Memo 579.

Horn, B.K.P. (1974) "Determining lightness from an image", Computer Graphics and Image Processing, 3 (4), 277-299.

Horn, B.K.P. (1977) "Image Intensity Understanding", Artificial Intelligence, 8, 201-231.

Horn, B.K.P. and Sjoberg, R.W. (1978) "Calculating the Reflectance Map", M.I.T. A.I. Lab. Memo 498.

Hueckel, M. (1971) "An Operator Which Locates Edges In Digitised Pictures", Journal A.C.M. 18 (1), 113-125.

Johannsen, G. & Bille, J. (1982) "A Threshold Selection Method using Information Measures", Proc. IJCPR-6, 140-142, Munich.

Kelly, M. (1971) "Edge Detection in Pictures by Computer Using Planning", p.397-409 in Machine Intelligence 6, eds. Meltzer and Michie, Edinburgh Univ. Press.

Kreyszig, E. (1972) "Advanced Engineering Mathematics", (3rd. edition) John Wiley and Sons, Inc. New York.

Laws, K.I. (1980) "Textured Image Segmentation", Ph.D. thesis, Univ. of Southern California.

Lesser, V.R. & Erman, L.D. (1977) "A Retrospective View of the HEARSAY-II Architecture", Proc. IJCAI-5, 790-800.

Maleson, J.T., Brown, C.M. and Feldman, J.A. (1977) "Understanding Natural Texture", Proc. DARPA Image Understanding Workshop, Science Applications Inc.

Marr, D. (1977) "Artificial Intelligence - A Personal View", Artificial Intelligence, 9, 37-48.

Marr D. and Hildreth E. (1980) "Theory of Edge Detection", Proc. Roy. Soc. Lond. B, 207, 187-217.

Marr, D. (1982) "Vision", W.H. Freeman Co., San Francisco.

Nevatia, R. & Babu, K.R. (1980) "Linear Feature Extraction and Description", Computer Graphics and Image Processing, 13, 256-269.

Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., and Lamperis, T. (1977) "Geometrical Considerations and Nomenclature for Reflectance", N.B.S. monograph 160, National Bureau of Standards, U.S. dept. of Commerce, Washington D.C.

O'Gorman, F. (1978) "Edge Detection Using Walsh Functions", Artificial Intelligence, 10, 215-223.

Popplestone, R.J., Brown, C.M., Ambler, A.P. and Crawford, G.F. (1975) "Forming models of plane- and cylinder-faceted bodies from light stripes", Proc. 4th. IJCAI, 664-668.

Pratt, W.K. (1978) "Digital Image Processing", John Wiley & Sons, New York.

Pun, T. (1980) "A new method for grey-level picture thresholding using the entropy of the histogram", Signal Processing, 2, 223-237.

Roberts, L.G. (1965) "Machine Perception of Three-Dimensional **Solids"**, p.159-197 in Optical and Electro-Optical Information **Processing**, eds. J. Tippet et al, M.I.T. Press, Cambridge MA.

Rosenfeld, A. and Thurston, M. (1971) "Edge and Curve Detection for Visual Scene Analysis", I.E.E.E. trans Computers, C-20 (5), 562-569.

Rosenfeld, A., Thurston, M. & Lee, Y. (1971) "Edge and Curve Detection: Further Experiments", IEEE trans. on Computers, C-21, 677-715.

Rosenfeld, A. and Kak, A.C. (1982) "Digital Picture Processing", Academic Press, Inc. London.

Schatz, B.R. (1977) "The Computation of Immediate Texture Discrimination", AI memo 426, MIT AI Lab.

Shirai, Y. (1973) "A context sensitive line finder for recognition of polyhedra", Artificial Intelligence, 4 (2), 95-120.

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

Tanimoto, Steven L. (1976) "Pictorial Feature Distortion in a Pyramid", Computer Graphics and Image Processing, 5, 333-352.

Ullman, S. (1976) "Visual Detection of Light Sources", Biological Cybernetics, 21, 205-212.

Waltz, D. (1975) "Understanding Line Drawings of Scenes with Shadows", p. 19-91 in The Psychology of Computer Vision, P.H. Winston (ed.), McGraw-Hill.

Witkin, A.P. (1982) "Intensity-Based Edge Classification", Proc. AAAI-82, 36-41.

Witkin, A.P. (1983) "Scale-Space Filtering", Proc. IJCAI-83, 1019-

1022, Karlsruhe, West Germany.

Woodham, R.J. (1978) "Photometric Stereo: A reflectance map technique for determining surface orientation from image intensity", Proc. 22nd. Int. Symp., 136-143, Society of Photo-optical Instrumentation Engineers, San Diego USA.

Appendix 1

Blurring an image which has a Gaussian probability distibution with Gaussian noise is equivalent, in terms of the distributions, to convolving one Gaussian with another.

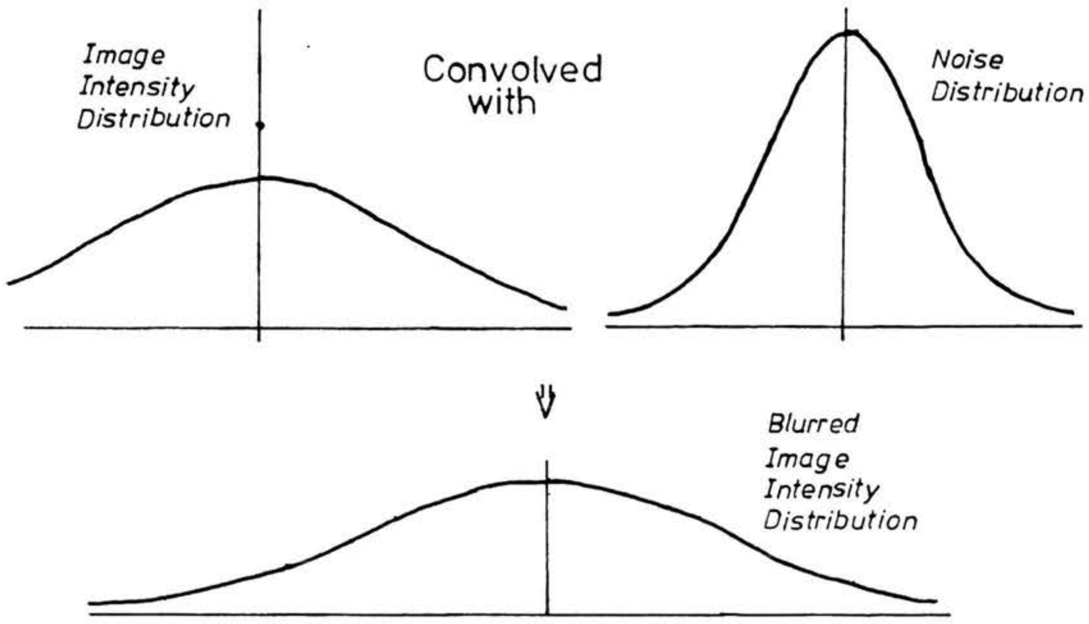


Image Intensity Distribution
Convolved with
Noise Distribution
Blurred Image Intensity Distribution

Let the original image distribution be

$$I(t) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma_1^2}}$$

and let the noise distribution be

$$N(t) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma_2^2}}$$

then the blurred image distribution is

$$y(t) = I(t) * N(t)$$

where * is the convolution operator.

$$y(t) = \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{T^2}{2\sigma_1^2}} e^{-\frac{(t-T)^2}{2\sigma_2^2}} dT$$

let $K = \dfrac{1}{2\pi\sigma_1\sigma_2}$

$$y(t) = K \int_{-\infty}^{\infty} e^{-[\frac{T^2}{2\sigma_1^2} + \frac{(t-T)^2}{2\sigma_2^2}]} dT$$

consider just the power of e

$$-[\frac{T^2}{2\sigma_1^2} + \frac{(t-T)^2}{2\sigma_2^2}] = -\frac{1}{2}\,\frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}[T^2 - \frac{2t\sigma_1^2}{\sigma_1^2 + \sigma_2^2}T + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}t^2]$$

completing the square and rearranging

$$= -\frac{t^2}{2\sigma_2^2}[1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}] - \frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1^2\sigma_2^2}[T - \frac{t\sigma_1^2}{\sigma_1^2 + \sigma_2^2}]^2$$

$$y(t) = K\, e^{-\frac{t^2}{2\sigma_2^2}[1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}]} \int_{-\infty}^{\infty} e^{-\frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1^2\sigma_2^2}[T - \frac{t\sigma_1^2}{\sigma_1^2 + \sigma_2^2}]^2}$$

letting

$$K^1 = e^{-\frac{t^2}{2\sigma_2^2}[1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}]}$$

and utilising the change of variable

$$u = \sqrt{\frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2\sigma_2^2}}[T - \frac{t\sigma_1^2}{\sigma_1^2 + \sigma_2^2}]$$

we get

$$y(t) = K\,K^1 \frac{\sigma_1^2\sigma_2^2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}u^2}\, du$$

now utilising the error function integral (Kreyszig 1972, p.692)

$$y(t) = K\, K^1 \frac{\sigma_1^2 \sigma_2^2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \sqrt{2\pi}$$

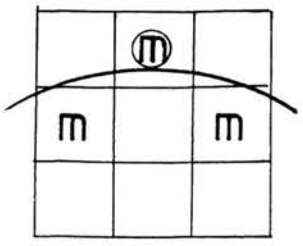
restoring K and K^1 and simplifying, we get

$$y(t) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma_1^2 + \sigma_2^2}}\, e^{-\frac{t^2}{2(\sigma_1^2 + \sigma_2^2)}}$$

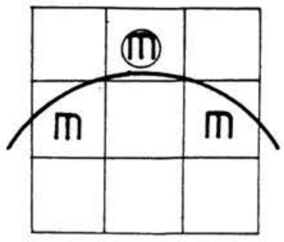
which is Gaussian with mean zero and variance $\sigma_1^2 + \sigma_2^2$.

Appendix 2

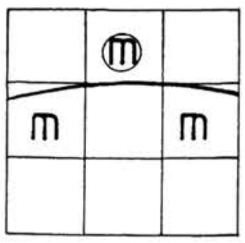
The cross test labels a mixed pixel as illegal (for tracking) if neither diagonal has both adjacent pixels unmixed. If a step-like edge is straight, it should generate mixed pixels which all pass the cross test, but if an edge is curved the cross test may fail as shown below.



A



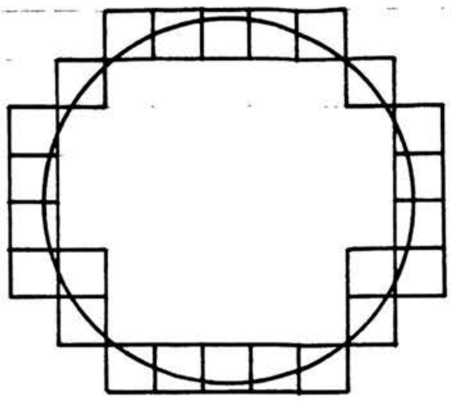
C



B

In each case the pixels labelled ⓜ fail. To quantify the amount of curvature which causes failure we can examine how circular edges digitise, circles having constant curvature. Assume we have an image consisting of a light-coloured circle on a dark background, with circle image intensity s and background intensity 0. Also assume the image is digitised by uniformly integrating over square pixel areas of length of side one unit. For a typical case, even with small circles, there is no problem, as shown overleaf.

The three examples A, B, and C above, where failure does occur represent worst cases. We will examine only case C since it is the easiest to analyse, but because of their similarity cases A and B should produce similar results.



The basic idea will be to choose a significance threshold t and find the smallest circle which can be successfully digitised as a function of both t and the edge amplitude s. An appropriate part of the circle and quantisation grid is shown overleaf.

For the cross test to fail both p2 and p5 must be mixed, so

$$|p1 - p2| > t$$

$$|p2 - p3| > t$$

$$|p4 - p5| > t$$

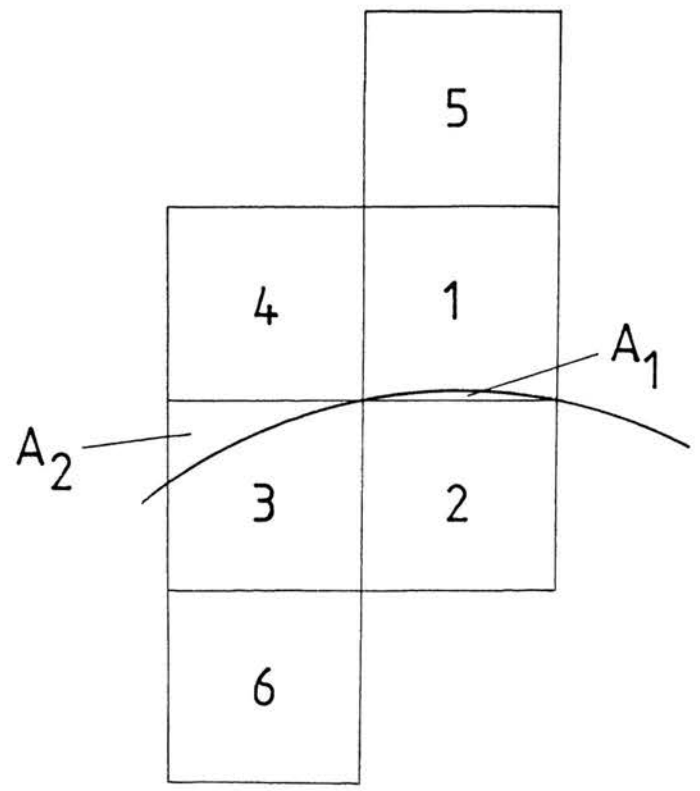
$$|p5 - p6| > t$$

but, in fact, since we are trying to minimise circle radius, we need only check

$$|p1 - p2| > t$$

$$|p5 - p6| > t$$

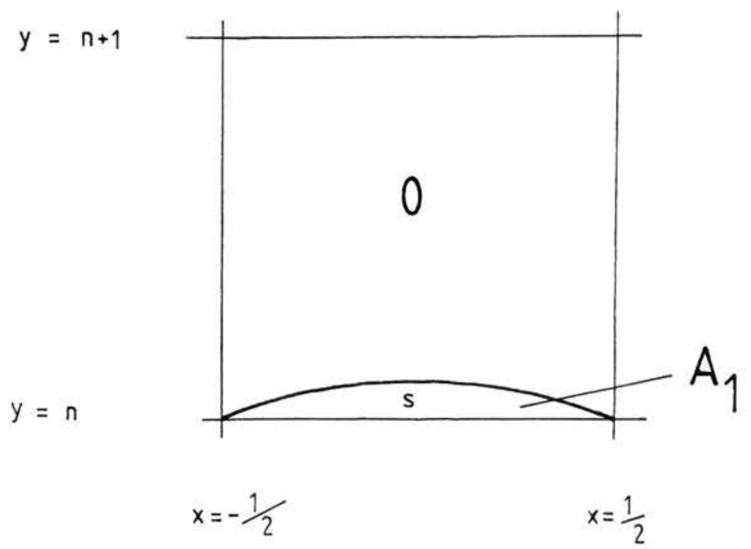
with a background (analogue) intensity of 0 and an intensity inside the circle of s, this means we need



$$A_1 s > t \quad \text{and} \quad s-(1-A_2)s > t$$

$$A_1 > \frac{t}{s} \quad \text{and} \quad A_2 > \frac{t}{s}$$

To find A_1



$$y = \sqrt{r^2 - x^2}$$

$$r^2 = n^2 + \frac{1}{4}$$

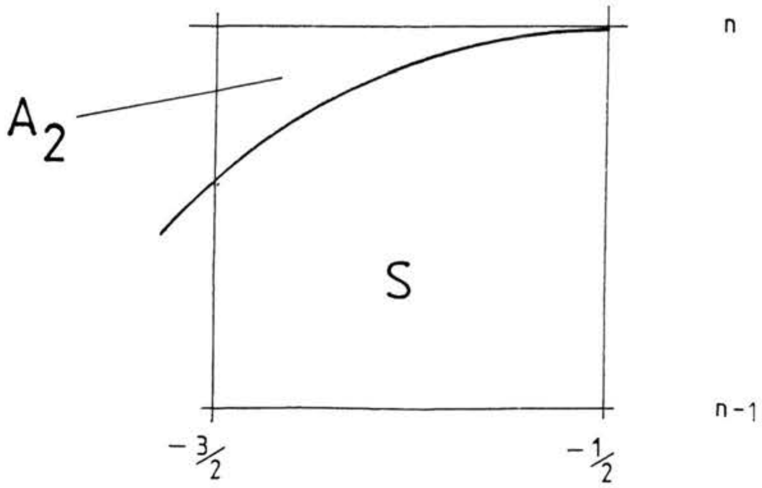
$$A_1 = \int_{\frac{-1}{2}}^{\frac{1}{2}} y - n \quad dx$$

$$A_1 = \int_{\frac{-1}{2}}^{\frac{1}{2}} \sqrt{r^2 - x^2} - n \quad dx$$

integrating and simplifying, we get the following expression for A_1 in terms of n

$$A_1 = \left(n^2 + \frac{1}{4}\right) \quad \sin^{-1}\left[\frac{1}{2\sqrt{n^2 + \frac{1}{4}}}\right] - \frac{n}{2}$$

Similarly, for A_2



$$A_2 = \int_{\frac{-3}{2}}^{\frac{-1}{2}} n - y \quad dx$$

$$A_2 = \int_{\frac{-3}{2}}^{\frac{-1}{2}} n - \sqrt{r^2 - x^2} \quad dx$$

$$A_2 = \frac{5n}{4} - \frac{3}{4}\sqrt{n^2-2} - \frac{1}{2}(n^2-\frac{1}{4})\left[\sin^{-1}\frac{3}{2\sqrt{n^2+\frac{1}{4}}} - \sin^{-1}\frac{1}{2\sqrt{n^2+\frac{1}{4}}}\right]$$

Tabulating both A_1 and A_2 against n and r, we get

n	r	A_1	A_2
2	2.06	0.0412	0.2284
3	3.04	0.0276	0.1441
4	4.03	0.0208	0.1063
5	5.02	0.0166	0.0844
6	6.02	0.0139	0.0701
7	7.02	0.0119	0.0599
8	8.01	0.0104	0.0523
9	9.01	0.0093	0.0465
10	10.01	0.0083	0.0418

From this we see that $A_2 > A_1$ so we can concentrate on

$$A_1 > \frac{t}{s}$$

A_1 increases as the radius decreases so we can find the smallest A_1 which doesn't satisfy the inequality. This will be the smallest circle which can be successfully tracked. Assuming we have a significance threshold of 18 greylevels then we can tabulate the smallest circle radius correctly tracked, r_c, as a function of the edge amplitude

s	20	50	100	150	200	250
$\frac{t}{s}$	0.9	0.36	0.18	0.12	0.09	0.072
r_c	2	2	2	2	2	2

Using a very much smaller significance threshold, t=2, gives

s	20	50	100	150	200	250
t/s	0.1	0.04	0.02	0.013	0.010	0.008
r_c	2	3	5	7	9	11

Hence, in a low-noise image where a relatively large threshold is being used, curved edges are not a problem. However, in a low-noise image using a low threshold will limit the edge curvature that will be successfully tracked.

Appendix 3

The ALC operates by tracking low-resolution step edges, at each point examining the corresponding area of the high resolution pixel/boundary segment classification. Based on the type of pixels and boundary segments found in the classification, the edge is labelled as one of {step, fuzzy, complex, faint}.

In the following listing, the function classify_edge controls the edge tracking and makes the decision on edge type. At each edge point it calls one of the three functions, mixed_rules, vertical_rules, or horizontal_rules, depending on whether the edge point is a mixed pixel, a vertical significant boundary segment, or a horizontal significant boundary segment. These functions perform the examination of the high resolution classification, allocating points among four tallies (one tally for each edge type). When the end of the edge is reached, it is classified according to the tally totals.

```
function classify_edge edge_list => edge_type;
!Given the list of coordinate pairs of a low-resolution
!tracked edge, this function retracks the edge examining
!the corresponding part of the high-resolution edge/pixel
!boundary segment classification. Based on the type of
!pixels in the appropriate high-resolution neighbourhood
!it assigns points to tallies representing the four edge
!types, then classifies each edge as one of {step, fuzzy,
!complex, faint} based on the total tallies for the edge.
find_length(edge_list)->edge_length;
while in_edge do
   get_edge_point(edge_list)->edge_point;
   if is_mixed(edge_point) then                  !mixed pixel
      mixed_rules(edge_point)->point_tallies
   elseif is_vertical(edge_point) then           !vertical b.s.
      vertical_rules(edge_point)->point_tallies
   else                                          !horizontal b.s.
      horizontal_rules(edge_point)->point_tallies;
   update_tallies(point_tallies,tally_set)->tally_set;
enddo;
!Now, given the set of tally totals and the length of the
!edge, decide which type it is.
decompose(tally_set)->step_tally->fuzzy_tally->complex_tally
   ->faint_tally;
edge_length*1.5->threshold;
if step_tally > threshold then
   "step"->edge_type
elseif fuzzy_tally>complex_tally and fuzzy_tally>faint_tally then
   "fuzzy"->edge_type
elseif complex_tally>fuzzy_tally and complex_tally>faint_tally then
   "complex"->edge_type
elseif faint_tally>fuzzy_tally and faint_tally>complex_tally then
   "faint"->edge_type
else
   print('edge not classified')
endfunction;
```

```
function mixed_rules edge_point => point_tallies;
!Given the coordinates of a mixed pixel, this function
!examines the appropriate neighbourhood of the high-
!resolution pixel/boundary segment classification and,
!depending on the pixel and boundary segment types,
!allocates two points among the four edge-type tallies.
!                             b4
!                        p1        p2
!                    b3                b1
!                        p4        p3
!                             b2
for p=p1 to p4 do
   if is_mixed(p) then step_score+1->step_score
   elseif is_gradient(p) then fuzzy_score+1->fuzzy_score
   elseif is_texture(p) then complex_score+1->complex_score
enddo
for b=b1 to b4 do
   if is_significant(b) then bounds+1->bounds
enddo
make_zero(tallies);
if step_score>1 and not(fuzzy_score>1 or complex_score>1) then
   2->step_tally;make_set(tallies)->point_tallies;exit;
if fuzzy_score>1 and not(step_score>1 or complex_score>1) then
   2->fuzzy_tally;make_set(tallies)->point_tallies;exit;
if complex_score>1 and not(step_score>1 or fuzzy_score>1) then
   2->complex_tally;make_set(tallies)->point_tallies;exit;
step_score->step_tally;
fuzzy_score->fuzzy_tally;
complex_score->complex_tally;
step_score+fuzzy_score+complex_score->total_score;
if total_score=1 then
   if bounds>0 then step_tally+1->step_tally
      else 1->faint_tally;
if total_score=0 then
   if bounds>1 then 2->step_tally
   elseif bounds=1 then 1->step_tally;1->faint_tally
   else 2->faint_tally;
make_set(tallies)->point_tallies;
endfunction;
```

```
function vertical_rules edge_point => point_tallies;
!Given the coordinates of a significant vertical
!boundary segment, this function examines the
!appropriate neighbourhood of the high-resolution
!pixel/boundary segment classification and allocates
!two points among the four edge-type tallies
!depending on the type of the pixels and the boundary
!segments in the neighbourhood.
!                          b1
!                     p1      p2
!                     p4      p3
!                          b2
for b=b1 to b2 do
   if is_significant(b) then step_score+1->step_score
enddo;
for p=p1 to p4 do
   if is_mixed(p) then step_score+1->step_score
   elseif is_gradient(p) then fuzzy_score+1->fuzzy_score
   elseif is_texture(p) then complex_score+1->complex_score
enddo;
if step_score>1 and not(fuzzy_score>1 or complex_score>1) then
   2->step_tally;make_set(tallies)->point_tallies;exit;
if fuzzy_score>1 and not(step_score>1 or complex_score>1) then
   2->fuzzy_tally;make_set(tallies)->point_tallies;exit;
if complex_score>1 and not(step_score>1 or fuzzy_score>1) then
   2->complex_score;make_set(tallies)->point_tallies;exit;
step_score->step_tally;
fuzzy_score->fuzzy_tally;
complex_score->complex_tally;
step_score+fuzzy_score+complex_score->total_score;
if total_score=1 then
   1->faint_tally
elseif total_score=0 then
   2->faint_tally;
make_set(tallies)->point_tallies;
endfunction;
```

```
function horizontal_rules edge_point => point_tallies;
!Given the coordinates of a significant horizontal
!boundary segment, this function examines the
!appropriate neighbourhood of the high-resolution
!pixel/boundary segment classification and
!allocates two points among the four edge_type
!tallies depending on the type of the pixels and
!the boundary segments in the neighbourhood.
!                   p4    p1
!              b2               b1
!                   p3    p2
get_neighbours(edge_point)->neighbourhood;
rotate_by_90(neighbourhood);
overlay(neighbourhood,classification);
vertical_rules(edge_point)->point_tallies;
reset(neighbourhood,classification);
!This function uses the symmetry of the horizontal
!and vertical primary boundary segment neighbourhoods
!to make the vertical_rules function apply to the
!horizontal case.
endfunction;
```

Paper published in proc. ECAI-82, Orsay, France, July 1982, p.190-196.

EDGE DETECTION FOR SEMANTICALLY BASED EARLY VISUAL PROCESSING

R J Beattie
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland EH1 2QL
Current address: Department of Electrical
and Electronic Engineering
Napier College
Colinton Road, Edinburgh EH10 5DT

Keywords: low-level vision, early visual processing, edge detection, line finding, region finding, texture.

Abstract: This paper is in two parts. In the first, we argue that edge detectors cannot be designed in isolation but only as components whose function is defined in terms of system goals. The second section describes the current state of an edge detector intended as part of a semantically based early visual processing system. The role of this system is to label image entities with their scene "meaning". As a first step in the process, the edge detector finds and tracks step-like edges and labels areas of texture and gradient in the image.

1.0 Introduction

Edge detection is the first step in most computer vision systems. Hundreds of edge detectors have been developed, (see eg Davis {1975}) but only a few are useful as part of a vision system. We suggest that this is due to two interrelated failings. Firstly, it is not usually clear how the output of an edge detector can be used for the next stage of processing and, secondly, it is not clear how the edge detector output relates to the input, ie the image. Both of these problems arise from considering edge detection as an end in itself, not as a component which has to fit into a system. One result of this is that techniques for measuring edge detector performance (eg p.495 Pratt {1978}) are based on relatively arbitrary signal processing measures, not on information processing. Only by clearly defining the role of edge detection in terms of the relationship between the scene, image, and the rest of the early processing system can an edge detector be designed as an effective component. This paper describes the current state of an edge detector which has been designed as part of an early visual processing system which itself embodies an opinion about how to do 3-d vision.

There is a significant and growing body of opinion in A.I. that early processing can be performed autonomously, ie without direction from higher levels of the vision system, (eg Tenenbaum, Barrow, and Fischler {1980}). The basis for this view is the belief that entities in the world (surfaces and light sources) can be adequately described without specific knowledge of scene content. In this approach, which arises from the veridicality of the image, image entities are interpreted as scene entities even at the early processing stage. An appropriate goal for early visual processing is to interpret significant image entities as shadow edges, regions of texture, etc. This type of early processing takes place in two stages:

1. find the edges in the image
2. attempt to label these edges with their scene 'meaning' based on constraint analysis.

The role of edge detection is to find image entities on which constraint analysis can take place. To make a sensible choice, the edge detector must incorporate some notion of how image entities relate to scene events. So before starting to design an edge detector, it is necessary to consider what is meant by the term "edge", how it relates to scenes, and how edges should be represented for constraint analysis.

To establish how image intensity variations arise, we begin by examining the behaviour of surfaces and light sources in the scene. The world contains objects, light sources, and a transparent medium. Assuming that our imaging device is a monochrome TV camera, we are interested in how light is reflected from objects to our imaging device. The imaging equation (Horn and Sjoberg 1978) shows that image irradiance is directly proportional to surface radiance. So, if we examine the behaviour of surfaces in terms of surface radiance, our observations will be easily expressible in terms of image irradiance. The radiance of a Lambertian surface varies continuously except under two conditions:

1. a step change in surface reflectance
2. a step change in surface irradiance.

The step change in surface irradiance would normally be a shadow caused by another object obscuring a light source. For a step change, the light would have to be from a point-like source. The step change could also be caused by indirect illumination, but this is very unlikely. Indirect illumination usually produces smooth gradients in surface irradiance. We only consider Lambertian surfaces in this paper because our main goal is the analysis of the edge detection process, not the construction of a full world model for constraint derivation. This does not mean that our early processing system will be limited to scenes containing only Lambertian surfaces, since at this stage, analysis of the world only serves to produce general goals for edge detection.

Having looked at the scene, we can now examine the transition from scene to image. Consider the illumination of adjacent elementary areas on the image plane. Two cases exist, as shown in fig.1:

1. they can be illuminated by adjacent areas of the same surface
2. they can be illuminated by unrelated areas of different surfaces.

In the first case, what we know about surface radiance also holds for image irradiance since there is a direct mapping from adjacent pieces of surface to adjacent pieces of image. This means that discontinuities in surface radiance will cause discontinuities in image irradiance, due to shadows and to changes in surface reflectance. In the second case, it is likely that the adjacent image areas will be subject to quite different amounts of irradiance, producing a step discontinuity. In practice, due to the defects of practical devices, these discontinuities will be blurred (see Brain {1979} for optics, Herskovits and Binford

Acknowledgements

This paper has benefited from discussions within the Edinburgh Vision Group. Several members of the group made useful comments on a draft of this paper. The work was financially supported by an SERC research studentship.

{1970} for empirical results). Steep gradients in image irradiance can also be caused by shadows from extended sources and from variations in object shape, such as the corners of polyhedral blocks. The analogue image, then, contains a wide range of intensity variations (assuming perceived intensity is proportional to image irradiance), ranging from steep gradients to uniform regions.

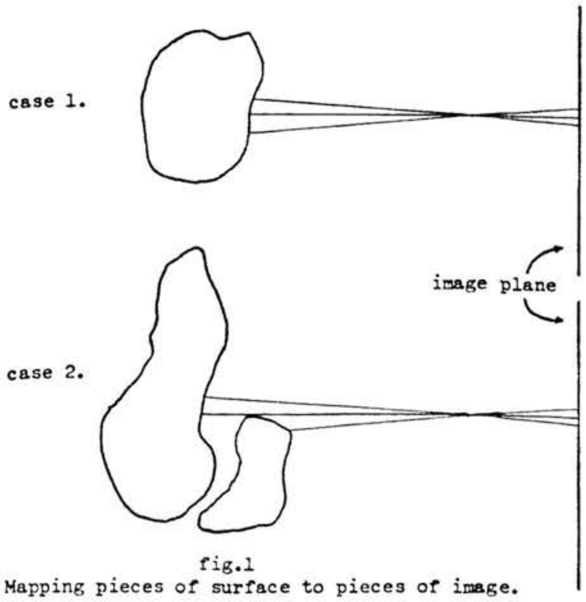


fig.1
Mapping pieces of surface to pieces of image.

In processing the image we are particularly interested in places where the intensity is changing quickly, since we might be able to recognise these as due to object boundaries, shape changes, shadows, or reflectance changes. Of course, a computer vision system only has access to a digitised image. So, while it may be relatively easy to formally define notions of continuity and discontinuity for ideal analogue signals, it is not easy for noisy digitised ones. Edge detection, which can be viewed as a search for discontinuities and significant gradients in the digitised image, has reflected this difficulty. Some methods treat the digitised image as if the underlying signal is continuous everywhere, but others assume that the underlying signal contains ideal step edges (see Brooks {1978}). As noted above, the analogue image does, in fact, contain intensity variations over a wide range of scales, limited at the upper end by the optical and signal processing capabilities of the imaging device.

The process of digitisation raises some significant issues. Some important image effects are largely dependent on the sampling and quantisation resolutions of the imaging device. Consider an image of a chessboard pattern. If each black or white square covers many pixels, then it will be easy to identify them as separate regions. At the other extreme, if many squares are subtended by each pixel, the image will be a uniform grey. Only in some intermediate range, where the number of squares subtended by the image is of the same order as the number of pixels, will the image take on a textured appearance. The concept of texture is largely to do with relative sizes and resolutions. Similarly, object shape variations occur over a wide range of scales, both in the scene and in the image. In images of natural scenes, these variations will cover a range from step edges to uniform regions. What constitutes a step edge is highly dependent on the relative size and distance of the phenomenon, the optical imaging system, and the resolution of the device. The corner of a building may look like a step edge from a distance, but appears as a gradual shape change close up. One cannot get rid of this problem simply by using higher resolution imaging devices. In terms of the above example, that just means you have to retreat a little further before the corner of the building again looks like a step edge.

The point of this discussion is to illustrate that the concept of "edge" is a very complex one, especially in digitised images. Since the scene events we are interested in (obscuring edges, shadows, etc) produce discontinuities and steep gradients in analogue image intensity, we are interested in places in the digitised image where the intensity values are changing quickly. The difficulty is in being any more specific at this level. If we design a mask which will react to some particular pattern, say, ideal step edges, we may miss other gradients which are important. In other words, we must be careful not to over specify what we are willing to accept as a significant entity in the digitised image.

Having found the places in the digitised image where significant intensity changes take place, we need to represent those changes in a way useful for later processing. The stage following edge detection in our system is constraint analysis, which involves examining image intensities in the vicinity of the edge. This means finding sets of pixels on either side of the edge, as close to the edge as possible. For example, as Ullman {1976} has noted, the intensity across a reflectance edge has a constant ratio, which is the ratio of surface reflectances. But this assumes that the incident light on both sides of the edge is the same. To test for reflectance edges then, we need access to intensity values as close to the edge as possible to ensure that the lighting constraint holds. However, we must beware of mixed pixels, since these are no use for such a test. (A mixed pixel is one through which a step edge passes. Its intensity depends on the position of the step within the receptive field of the pixel, as well as the intensities on either side of the edge.) It seems that we need a method of finding and representing edges which satisfies the following criteria:

1. the edge should be represented as accurately as possible
2. pixels should be classified as mixed or pure
3. it should be possible to tell from the edge detector output what the underlying image intensity variations were.

2.0 Finding Significant Intensity Changes

As noted above, intensity changes occur over a wide range of scales in the image, but the places where intensity changes quickly are of particular interest, since they correspond to informative scene/imaging events. We begin by dividing this range into 3 classes: uniform, gradient, and step. A uniform region of the image corresponds to a surface in the scene whose shape, incident light, and reflectance properties are changing slowly. A gradient region is one where the intensity values are changing more quickly and could be caused by a shadow from an extended source, indirect illumination, a piece of surface whose shape is changing, or an extended change in the surface reflectance properties. A step region in the image corresponds to what would usually be an ideal step discontinuity in a perfect imaging system. These can be caused by obscuring edges where one object passes in front of another, shadows from point sources, sharp changes in surface reflectance properties, or parts of a surface whose shape is changing very quickly. It is important to note that, for a particular imaging device, the boundaries of this classification depend crucially on both its spatial and greyscale resolution. Also, for a real imaging device, step edges will be blurred to some extent and noise will be present, so it will be impossible to separate gradients from steps by simply thresholding the intensity change.

Since both the gradients and steps are informative about scene events, they can be considered to be significant. We begin by attempting to label each boundary between pixels (assuming 4-connectedness) as primary or secondary. Primary boundaries are considered to be significant. In terms of the above classification they should be part of steps or gradients. In an ideal noiseless image, uniform regions could be defined as groups of pixels connected by secondary boundaries, a secondary boundary being defined as one between two pixels which have identical or adjacent greyscale values, ie

$$|/_1-/_2|\leq 1$$

However, the situation is complicated by the presence of noise arising in the electronics of the imaging device and associated systems. In line with the earlier statement about image veridicality, small surface marks and defects are not considered to be noise. It is assumed that the noise is Gaussian. This holds for many systems, and for ours in particular (Beattie 1980). The intensity difference (b) between two pixels with intensity values $/_1$ and $/_2$, subject to Gaussian noise of standard deviation σ, is

$$b = N(/_1-/_2, 2\sigma^2)$$

So, the noise standard deviation on the difference between the pixels is greater by a factor of $\sqrt{2}$ than the noise on the pixels themselves. It is this observation which has led to the plethora of mask-based edge detectors, since these reduce the effect of noise in proportion to the square root of the size of the mask. However, we want to find all of the significant boundaries between pixels so a mask will not be used because it tends to limit both the type of edge points found and the accuracy with which they can be located. Even with a small mask, eg 3*3, the same output can be generated by an edge, a bar, or a blob (p.566 Rosenfeld and Thurston (1971)). To help combat noise, we want to choose a threshold d such that if $|/_1-/_2|>d$ the boundary is labelled primary (significant); otherwise, it is labelled secondary.

For a particular choice of d, given the picture noise, we can find an expression for the probability of a legitimate secondary boundary being wrongly labelled as primary. In the case where $/_1=/_2$ the probability of the boundary being wrongly classified as primary can be shown to be

$$P_1 = 2(1 - \Phi(\tfrac{d}{\sigma}))$$

where

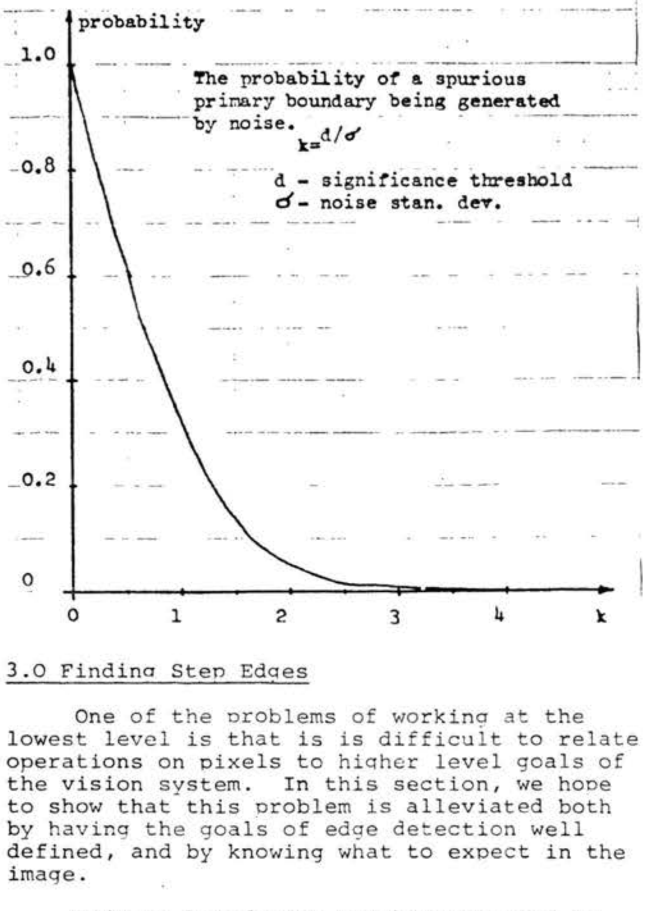
$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{u^2}{2}}du$$

P_1 is the probability of a spurious primary boundary being generated by noise. By taking the ratio $k = \frac{d}{\sigma}$, ie expressing the threshold in terms of the noise, this probability can be graphed as shown in Fig.2. For a low probability of wrongly classifying a boundary in a uniform region a value of k of 2·5 is appropriate. Thus the classification threshold d should be set about

$$d = 2\cdot 5\times\sqrt{2}\sigma$$

where σ is the picture noise standard deviation. As well as enabling a reasonable choice of threshold to be made, knowing the probability of generating spurious primary boundaries could be useful in finding and interpreting edges. A more extensive investigation of the probabilities involved in choosing a threshold is given in Beattie(1981).

fig.2



3.0 Finding Step Edges

One of the problems of working at the lowest level is that is is difficult to relate operations on pixels to higher level goals of the vision system. In this section, we hope to show that this problem is alleviated both by having the goals of edge detection well defined, and by knowing what to expect in the image.

Primary boundaries can be generated by image gradients or by step-like edges. Step edges are particularly interesting because they correspond to informative scene/imaging events. We first identify those primary boundaries which are part of step edges, then use the results to track the curves in the image.

3.1 Classifying Pixels

First, we consider the mixed pixel problem. When a step edge is digitised, the intensity transition may lie within a pixel. This pixel will have a greyscale value somewhere between those of its neighbours across the edge. The effect of this is that a step edge, even an ideal step edge, will not generate a single line of primary boundaries but pixels bordered by one, two, three, or even four primary boundaries depending on the local geometry and intensity variations of the edge relative to the sampling grid. Since the intensity of the mixed pixel lies between the intensities of those on either side of it across the edge, all pixels which have two parallel primary boundaries are labelled "mixed". Some of them will be due to gradients and textured areas. To find those mixed pixels which are really part of step edges, we can use their thinness as a distinguishing property. Consider a pixel through which a step edge is passing. Assume that the edge is locally straight as shown in Fig.3. The edge cannot pass through any more than two of the 4-connected pixels (p1-p4). Neither can it pass through any more than two of the diagonally-connected pixels (p5-p8). Notice that the edge can either pass through two adjacent 4-connected pixels, such as p1 and p2, or through two opposite 4-connected pixels, such as p2 and p4 as shown in Fig.3. However, it cannot pass through two adjacent diagonal pixels, such as p6 and p7, but only through two opposite diagonal pixels, such as p6 and p8. This can be incorporated into a test as follows (the cross test):

If a mixed pixel has a pair of opposite diagonally-connected pixels, neither of which are mixed, then label it "legal".

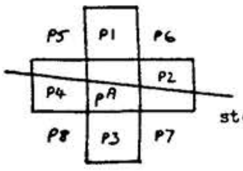


fig.3

A step edge passing through pA.

In other words, if you imagine drawing a little cross on every pixel labelled "mixed", then label as legal those which have at least one diagonal which doesn't enter mixed pixels, otherwise label it illegal. The labels "legal" and "illegal" are simply used to indicate whether or not a mixed pixel could legally be part of a step edge. A step edge will always produce mixed pixels which are legal as defined here. Gradients and regions of texture are not guaranteed to do so. In fact, gradients in general won't produce legal mixed pixels because they usually result in multiple adjacent rows of mixed pixels all of which fail the cross test. Similarly, textured regions usually produce a large proportion of illegal mixed pixels. It is possible to imagine a textured region which would produce only legal mixed pixels, but it is exceedingly unlikely to occur in a real scene.

At this stage boundaries and pixels have been classified: boundaries are primary or secondary; pixels are unmixed, legal mixed, or illegal mixed. A mixed pixel is understood to "swallow" the primary boundaries surrounding it. This is reasonable since the boundaries don't indicate an edge between two pixels. An "unswallowed" primary boundary is called an unattached primary boundary (upb). The pixel and boundary classification for the 64*64 picture of a toy steam engine shown in Fig.4 is shown in Fig.5. Although both pixels and boundaries exist, the whole classification can be given, using only pixels and one character per pixel, with the following legend:

M		legal mixed pixel
X		illegal mixed pixel
N S E W		unattached primary boundary on the top, bottom, right, or left side of the pixel
1 ┐	NE	pixel with two adjacent
2 ┘	SE	unattached primary boundaries
3 ┌	NW	as indicated
4 └	SW	

3.2 Tracking

Following boundary and pixel classification, the next stage is tracking. The two primitives are legal mixed pixels and unattached primary boundaries (horizontal and vertical). Note that the existence of an unattached primary boundary automatically implies that the pixels on either side of it are unmixed. Note also that the method works for curved edges. The amount of curvature allowed depends on the greyscale resolution. The basic approach taken for tracking is to examine the patterns of mixed pixels and unattached primary boundaries produced by a step edge. Imagine moving a pointer along the edge. The current position of the pointer can either be a mixed pixel or an unattached primary boundary. Legal next positions for the pointer can be expressed as a pair of sets of if-then rules which express the possible connectivity relationships of the primitives in step edges.

The two sets of rules are shown in Fig.6. Mixed (p) is true if the relevant pixel p is a legal mixed pixel; upb (b) is true if boundary b is an unattached primary boundary; lnp (n) indicates that n, which can be a legal mixed pixel or an unattached primary boundary, is an adjacent pointer position. In operation, each of the rules of the appropriate set is applied in turn at the current pointer position. If the LHS of any rule succeeds then its RHS defines one legal next position for the pointer. If the current position has only one neighbour then it must be at the end of an edge and is called a terminator. If it has two neighbours then it must be a mid-line point. If it has three or four neighbours it is classified as a junction. Tracking is based on a raster scan with edges being marked as used when they are found. Further details of the tracking algorithm are given in Beattie (1981). The results of tracking on the steam engine are shown in Fig.7.

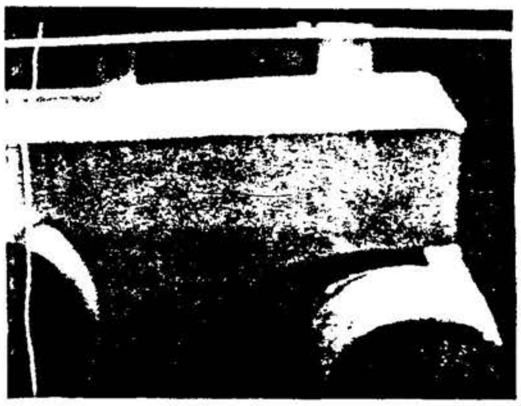


fig.4

A photograph of a T.V. monitor screen showing the picture of the toy steam engine before digitisation.

fig.5
The results of pixel and boundary classification on the steam engine picture (64*64 pixels).

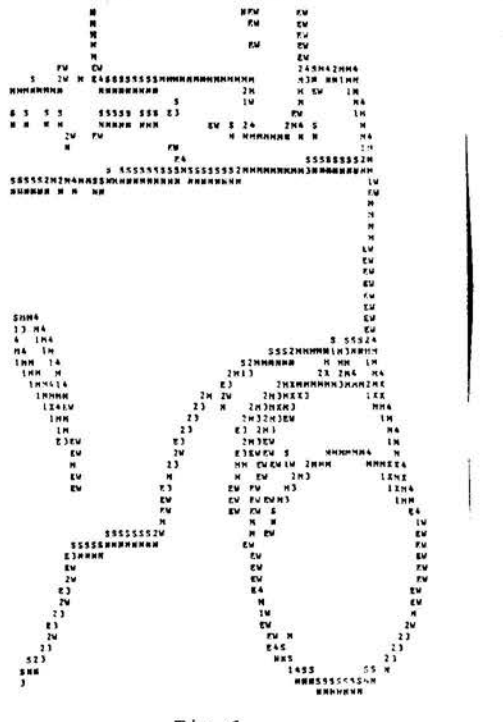
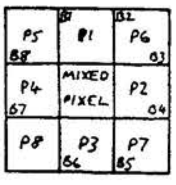
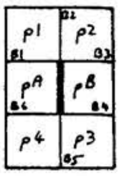


Fig.6

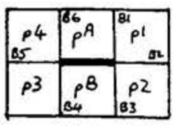
mixed(p1) → lnp(p1)
mixed(p2) → lnp(p2)
mixed(p3) → lnp(p3)
mixed(p4) → lnp(p4)
mixed(p5) and not (mixed(p1) or mixed(p4)) → lnp(p5)
mixed(p6) and not (mixed(p1) or mixed(p2)) → lnp(p6)
mixed(p7) and not (mixed(p2) or mixed(p3)) → lnp(p7)
mixed(p8) and not (mixed(p3) or mixed(p4)) → lnp(p8)
upb(b1) → lnp(b1)
upb(b2) → lnp(b2)
upb(b3) → lnp(b3)
upb(b4) → lnp(b4)
upb(b5) → lnp(b5)
upb(b6) → lnp(b6)
upb(b7) → lnp(b7)
upb(b8) → lnp(b8)





mixed(p1) → lnp(p1)
mixed(p2) → lnp(p2)
upb(b1) → lnp(b1)
upb(b2) → lnp(b2)
upb(b3) → lnp(b3)
mixed(p3) → lnp(p3)
mixed(p4) → lnp(p4)
upb(b4) → lnp(b4)
upb(b5) → lnp(b5)
upb(b6) → lnp(b6)

neither pA nor pB can be mixed.



unattached primary boundary

These two sets of tracking rules are used to find the neighbours of mixed pixels or unattached primary boundaries on step edges.

4.0 Dealing with Gradients and Texture

As a by-product of the method of finding step edges, certain pixels in an image are labelled "illegal". These pixels are in regions where significant intensity variations are taking place, but which fail the cross test. Recall that pixels which pass the cross test are part of intensity variations which are very thin in one direction. Pixels which fail the cross test must, therefore, be part of image intensity variations which are fat, ie occur over significant distances in two orthogonal directions. Because of this,

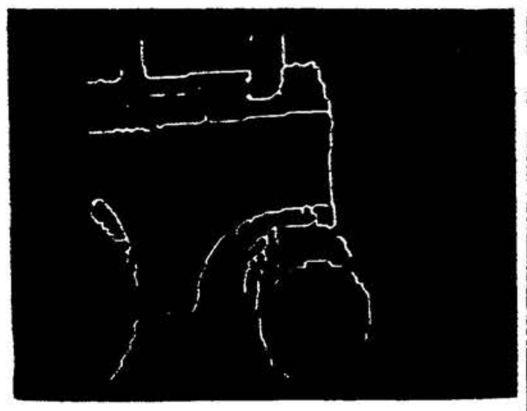


fig.7
The results of edge tracking on the pixel classification of fig.5.

illegal mixed pixels don't usually occur in isolation, but in clusters. These illegal regions can arise in two ways. As already described they can result from extended gradients in image intensity which can be caused by a number of scene events. This type of illegal region corresponds to a single surface, one or more of whose parameters is varying significantly but smoothly. Alternatively, illegal regions can be caused by many step-like edges sufficiently close together such that they cannot be resolved by the cross test. In other words, they can result from textured areas of the image.

4.1 Texture

So far texture has been mentioned a couple of times without being defined. The concept of texture is a complex one for early visual processing, it is easy to be misled by its everyday use. It is important to realise that texture is not simply structure but perceived structure. Clearly, texture cannot be defined in terms of a world model alone. The optical system, image resolution, and the relative size and distance of objects in the scene all affect whether an object is seen as an individual item or as an element in a texture field. However, some world considerations may be useful in coming to an understanding of texture. Various scene arrangements can constitute texture for a human observer. Sometimes it is a large number of similar objects close together at a relatively large distance, eg the leaves of a tree. Alternatively, the perception of texture can be caused by multiple variations in surface reflectance properties, eg wood grain. Also, a textured appearance may be caused by surface shape variations viewed at an appropriate distance, eg the bark of a tree. There are two main factors here - surface shape (and range) variations and surface reflectance variations. When there are many of these close together such that we cannot clearly perceive the individual variations, we call that texture. (This is a more restricted description of texture than many in common usage.)

What is being suggested is that the perception of texture occurs when the perceiver is unable to fully "process" that part of the visual field to a stage where the individual shape/reflectance changes can be interpreted as 3-d events because of insufficient information. For a computer vision system, this implies that a definition of texture must rest, not only on the image effects, but also on a model of the early processing. When there is not enough information, ie not high enough sampling frequency or greyscale resolution for individual intensity changes to be distinguished and processed normally, then the region containing these changes can be called "texture" and other methods applied.

The early processing system described in preceding sections is based on finding step edges and interpreting them by analysing the intensity variations across and along the edges. This requires that each edge is distinguishable by the edge finding and tracking process. If the edges are too close together they will interact and fail the cross test. With this in mind an attempt can now be made at a definition of texture, but note that it will only hold for the particular early processing method being used.

When a number of scene/imaging events, such as obscuring edges, shape edges, or reflectance edges, are close enough together to produce a set of image edges which cannot be distinguished by the edge detection and tracking mechanism, then the image region enclosing that set of edges is said to be textured.

Although this definition is customised for our early processing system, there must be an equivalent one for any similar system.

4.2 Distinguishing Between Texture and Gradient

Fig.8 shows a 64*64 picture of a sports shoe after pixel classification. We want to decide which illegal mixed pixels are due to image gradients and which are due to image texture. This decision is made in two steps:

1. group the illegal mixed pixels into illegal regions
2. classify each region as gradient or texture.

For grouping purposes every pixel is considered to belong to one of the following two classes:

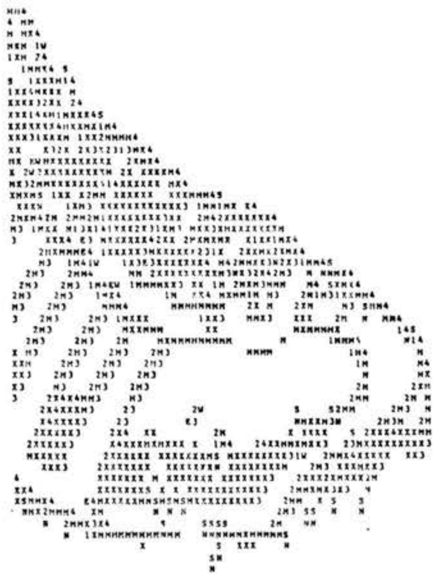
1. illegal mixed pixels
2. others.

Because illegal regions, especially those due to texture, are not always homogeneous, ie they don't contain only illegal mixed pixels, boundary tracking is used to find the illegal regions. A boundary passes between pixels of the two classes mentioned above. After tracking the external boundary of a region, the entire interior is considered to be part of the illegal region. The actual tracking process is of the "keeping your left hand on the wall" type, and is based on 8-connectivity. Again, this provides for slightly more flexibility in dealing with inhomogeneous regions. The algorithm used is described in detail in Beattie (1982).

The assumption is made that each illegal region arises either because of gradient or texture. Of course, it can happen that the two can be superimposed, in which case it is desirable that the region should be labelled texture, because texture-based analytic methods will be needed to recover further information. Also, the situation can arise where adjacent regions of gradient or texture give rise to a single illegal region. At this stage in the development of the system no attempt is made to divide these regions.

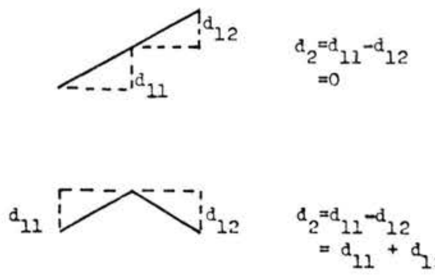
To distinguish between gradient and

fig.8
The initial classification of a 64* 64 pixel picture of a sports shoe.



texture we want to use the fact that the underlying intensity variations are smooth in a gradient region but not in a texture region. There are several ways to do this (see Beattie (1982)). As a first pass we use a method based on first and second differences of intensity within the regions. In the case of an ideal plane, which we can take as the limiting case of a gradient region, the 2nd difference will be zero. In the case of some idealised piece of texture, the 2nd difference will be the sum of the first differences as shown in Fig.9. We base our method of distinguishing between gradient and texture on whether the 2nd difference tends towards zero (gradient) or towards the sum of the first differences (texture). The 2nd difference operator (both horizontal and vertical) is applied wherever possible in an illegal region and the average 1st and 2nd differences computed (d_1, d_2).

fig.9



Limiting cases of the simple gradient and texture models.

Taking:

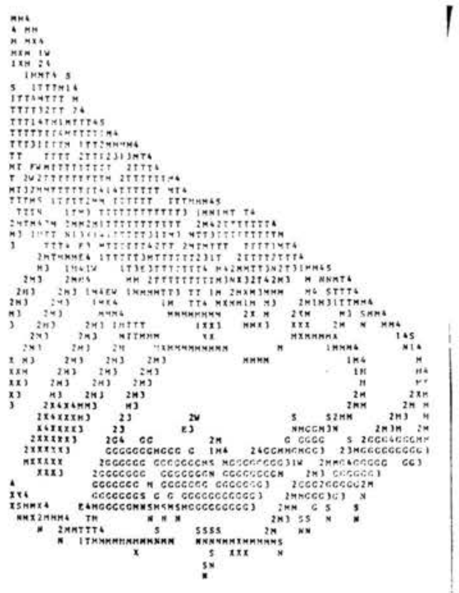
$$d_2-d_1$$

results in a number which should be positive for texture and negative for gradient. Rather than simply setting the threshold at 0, because noise is emphasised by the differencing operations and also because the distinction between gradient and texture is fuzzy, we use two thresholds at +d and -d. If $d_2 - d_1 > d$ then the region is labelled textured, if $d_2 - d_1 < -d$ it is labelled gradient, otherwise it is labelled "don't know". The threshold is based on the picture noise and the size of the illegal region. Using a similar investigation of probabilities as that used to decide on the significance threshold in section 2.0, a threshold of $d = (\sqrt{5}\sigma)/\sqrt{n_2}$ was chosen, where n_2 is the number of second difference operator applications. Again, details of the threshold derivation and the classification algorithm are given in Beattie (1982). The resulting updated pixel classification for the shoe picture is shown in Fig.10, with the following amendment to the legend.

G	gradient
T	texture
X	don't know (or region too small)

This approach has proven to be satisfactory in operation in that it usually produces the classification expected from knowledge of the scene which produced the image. However, it can be criticised for being somewhat ad hoc due to the crudeness of the models of gradient and texture and the rather arbitrary threshold selection.

fig.10
The final classification of the sports shoe picture. One of the illegal regions (marked with X's) has not been classified as gradient or texture.



5.0 Summary

The structure of the current system is shown in Fig.11. It produces an edge list in which each step edge found is represented as a list of co-ordinate pairs, and a pixel classification with some regions marked as gradient or texture. By analysing how intensity variations in the image arise, it has been possible to devise an edge detector whose output should provide a basis for interpreting image entities as scene entities. Designing the edge detector as a component, with specific goals, in an early processing system, enables a series of rational choices to be made, even at the pixel level, where the danger of basing algorithms on ad hoc heuristics is greatest. We claim that the added complexity of this scheme over some others is needed to deal with images of real scenes in a useful way. The system is still being refined.

To help combat noise and to find edges of a variety of sizes it would be interesting to apply our method to several versions of the same picture at different resolutions. Because the position of each edge is known exactly there should be no great problem in inventing a set of rules for combining the information. This has been a difficulty in the Marr and Hildreth system (Marr and Hildreth (1980), Hildreth (1980)). In accordance with our general philosophy these rules should be worked out by considering how the edges have arisen as well as their image appearance.

fig.11
The structure of the system

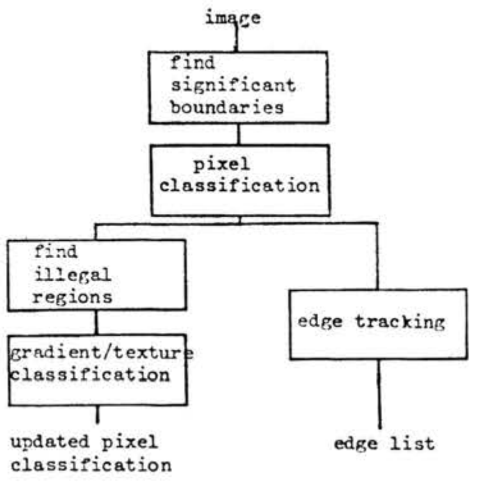


image
find significant boundaries
pixel classification
find illegal regions
gradient/texture classification
edge tracking
updated pixel classification
edge list

References

Beattie, R.J. (1980) Initial Image Representation in Computer Vision, W.P.80 Dept. of A.I., Univ. of Edin.

Beattie, R.J. (1981) Edge Detection For Semantically Based Early Visual Processing, W.P.95 Dept. of A.I., Univ. of Edinburgh.

Beattie, R.J. (1982) Edge Detection For Semantically Based Early Visual Processing, forthcoming W.P., Dept. of A.I., Univ. of Edinburgh.

Brain, A.E. (1979) Lenses For Industrial Automation, Tech. Note 201, Stanford Research Institute.

Brooks, M. (1978) Rationalising Edge Detectors, Comp. Graphics & Image Proc. 8, 277-285.

Davis, L.S. (1975) A Survey of Edge Detection Techniques, Comp. Graphics & Image Proc. 4, 248-270.

Herskovits, A. & Binford, T.O. (1970) On Boundary Detection, A.I. Memo 183, .. M.I.T.

Hildreth, E.C. (1980) Implementation of a Theory of Edge Detection, A.I. Memo 579, A.I. Lab. M.I.T.

Horn B.K.P. & Sjoberg, R.W. (1978) Calculating the Reflectance Map, A.I. Memo 498, A.I. Lab. M.I.T.

Marr, D. & Hildreth, E. (1980) Theory of Edge Detection Proc. Roy. Soc. Lond. B, 207, 187-217.

Pratt, W.K. (1978) Digital Image Processing, New York, John Wiley & Sons Inc.

Rosenfeld, A. & Thurston, M. (1971) Edge and Curve Detection for Visual Scene Analysis, I.E.E.E. Trans. Computers C-20(5), 562-569.

Tenenbaum, J.M., Fischler, M.A. & Barrow, H.G. (1980) Scene Modelling: A Structural Basis For Image Description, Comp. Graphics & Image Proc. 12, 407-425.

Ullman, S. (1976) Visual Detection of Light Sources, Biological Cybernetics 21, 205-212.

Paper published in proc. ICPR-7, Montreal, Canada, August 1984, p.653-655 (Vol.1)

A SEMANTICALLY-BASED MULTI-LEVEL EDGE DETECTION SYSTEM

R J BEATTIE

DEPT. OF ELECTRICAL AND ELECTRONIC ENGINEERING
NAPIER COLLEGE, EDINBURGH, EH10 5DT
SCOTLAND, U.K.

Abstract: It is well known that intensity changes in images of natural scenes take place over a wide range of scales. To find such edges a family of different-sized edge detection masks may be used, or the same mask may be applied to different-resolution versions of the image. The outstanding problem with such systems lies in integrating the multiple sets of edge data produced into a single coherent representation. The system described in this paper is an attempt to solve this problem based on accurate edge localisation and the use of scene-image semantics.

Images of natural scenes contain edges of many widths, from sharp step-like edges to blurred edges many pixels wide. Since it is difficult to design a single edge detection mask which will perform well over a range of edge widths, the normal solution is to search the image several times, each time looking for edges of a particular width.[1,2,3] This can be accomplished either by using a family of different-sized masks, or by applying the same mask to multiple different-resolution versions of the image. The major difficulty in this approach lies in constructing a single coherent description of all the edges in the image given the multiple sets of image data. In this paper we describe a multi-level system which attempts to use accurate edge localisation and scene-image semantics to simplify this problem.

Before attempting to design an edge detector its role must be specified by defining an overall vision system architecture and hence determining the needs of the component succeeding the edge detector. It is assumed here that the images to be used are of natural 3-d scenes produced via a conventional monochrome imaging device (e.g. vidicon). The overall vision system architecture is assumed to be hierarchical, since this simplifies modularisation and helps in determining the specifications of individual components. It is widely accepted that vision systems can be usefully split into high-level and low-level parts, with the interface being a description of the scene at the level of surfaces. The basic mechanism of the low-level vision system can be thought of as edge detection followed by constraint analysis, possibly over a set of images. Constraint analysis may consist of labelling each edge in a single image with its scene interpretation based on examining intensities in the neighbourhood of the edge [4,5] or of stereopsis or the use of motion.

THE GOALS OF EDGE DETECTION

The input to the edge detection system is provided by the image, which, apart from random noise and other minor degradations, is veridical. In other words, the image is a faithful representation of the scene under perspective projection. The output of the edge detection system must be a description of the intensity changes in the image in a representation suitable as an input to constraint analysis. Since on the one hand we have the veridical image and on the other hand the need to interpret image edges with their scene meaning, there is a clear implication that the edge detection process could benefit from both an analysis of how scene events produce image edges and the kind of information that should be explicit in the final edge representation to facilitate constraint analysis. Furthermore, to exploit image veridicality, edges should be related to their corresponding scene events throughout the derivation of the edge detector, even at the earliest processing stages. One consequence of this approach is that certain scene events which in some systems are considered to be noise to be eliminated, e.g. small surface marks, now assume parity with all other scene/imaging events, such as obscuring boundaries.

To investigate the imaging process and the needs of constraint analysis, one example was studied in detail, that of a sharp reflectance boundary on a surface. The constraints on the resulting image edge are well known [6,7]. By examining the assumptions that had to be made to label the image edge with its scene meaning, this study led to the formation of a set of goals for edge detection, as follows:

GENERAL

1. Find all the significant intensity variations - they are all caused by something in the scene.

2. If possible, relate significant intensity variations to their scene meaning. Although this is really the job of the constraint analysis system, it may be possible to achieve some preliminary semantic labelling during edge detection.

STEP-LIKE EDGES

3. Find and represent step edges as accurately as possible.

4. Try to maintain a clear relationship between an edge representation and its underlying image intensities. In particular, know which pixels are mixed and which are pure.

Since edge detection is the first operation carried out on the image, it is impossible to know at this stage which edges correspond to important scene events. In particular, the importance of an edge may not be directly related to its amplitude. Thus it is essential that the low-level vision system make as few assumptions about the image as possible, because these may then limit the performance of the high-level system.

It is well-known that step-like edges are particularly important because they often correspond to interesting scene events. When using step-edge representations for constraint analysis, we are usually interested in the location of the edge and the pattern of intensities in its neighbourhood. Hence the accuracy with which the edge is found and represented is crucial. A limiting factor in determining edge position is provided by the mixed pixel problem. If a sharp edge passes through the receptive field of a pixel, its resulting intensity depends on the position of the edge as well as the local image irradiance. Unless some assumptions are made about the shape of the edge, which is undesirable for the reason given above, the intensity of a mixed pixel should not be used in constraint analysis.

Structure of the System

Designing an edge detector based on the considerations given above led to a system with the structure shown in Fig. 1. Exactly the same edge detection process, the single-level system (SLS), is applied to several different-resolution versions of the image.

The outputs of each pair of adjacent resolutions are then input to an adjacent level comparator (ALC) which integrates the information contained in the two representations. Finally, the ALC outputs are combined into a single representation of the intensity changes in the image in a multi-level integrator (MLI). The current status of the system is that the single-level systems and adjacent-level comparators have been implemented.

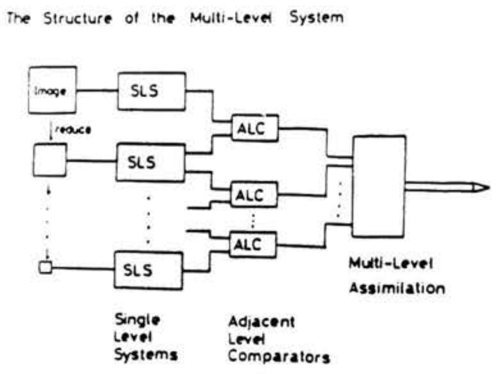


Fig. 1

The Single-level System

The SLS will only be summarised here, since it has already been described in detail elsewhere[8]; its internal structure is shown in Fig. 2.

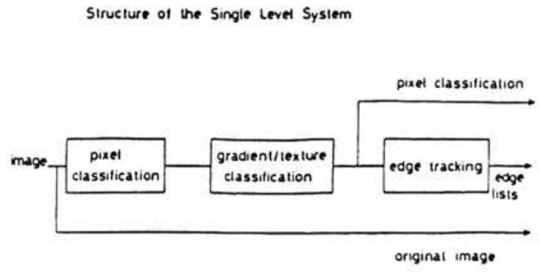
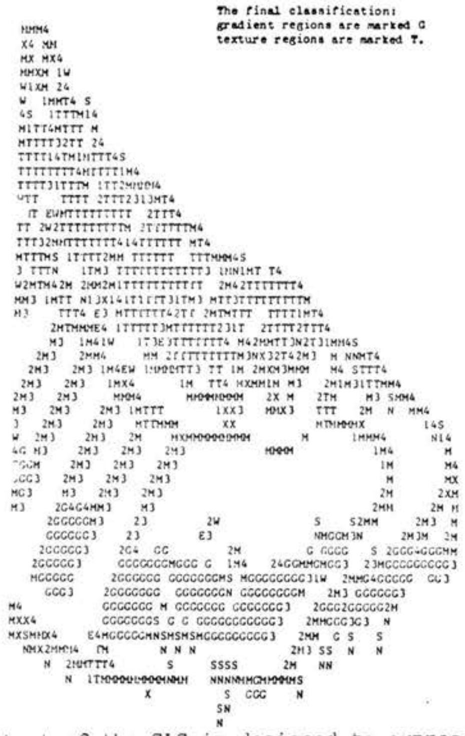


Fig. 2

Based on first differencing, pixel classification, finding image areas of high gradient, smoothness analysis, and edge tracking, its output has two main parts: an edge list and a classification array. The classification array overlays the image and marks areas where extensive significant intensity changes are taking place as "gradient" or "texture". The edge list contains sublists of sets of co-ordinate pairs for each step edge and gradient and texture boundary. The pixel classification resulting from a picture of a sports shoe is shown in Fig. 3. Mixed pixels in step edges are marked M.

Fig. 3


The final classification:
gradient regions are marked G
texture regions are marked T.

The output of the SLS is designed to express four types of image entity, each type satisfying certain constraints as follows:

Step edge	- thinness, connectivity, gradient.
Gradient region	- thickness, connectivity, gradient, smoothness.
Uniform region	- gradient (lack of).

Texture region - thickness, connectivity, gradient, roughness.

The Adjacent-level Comparator

In the SLS output, blurred edges occur implicitly as gradient regions in the classification array, but since they are often important in scene terms, it is desirable to have them represented explicitly in the final data structure. By reducing the resolution of the image, say by averaging intensities in an n x n window to produce a new single pixel value. Blurred edges of a certain width will become low-resolution step edges. The basic idea of the ALC is to use the outputs of two SLSs, whose inputs are an image and its reduced resolution version produced by averaging 2 x 2 windows, to find and represent blurred edges explicitly and increase system sensitivity.

The effect of reducing resolution on the constraints of the four SLS output entities provides the basis for ALC operation. If no interference occurs, step edges should produce step edges at lower resolutions. The connectivity of a gradient or texture region is normally preserved under reduction, but the thickness constraint will eventually be violated. Finally, since the low resolution SLS should be more sensitive than its high resolution counterpart, gradient significance should be preserved, and additional faint edges may be found. The effects of reducing resolution can be summarised as follows:

Steps	→ steps.
Gradients	→ gradients, steps.
Textures	→ textures, gradients, steps.
Uniform	→ uniform, others (t.g.s.).

By comparing the two edge lists produced from two SLSs, extra edges should be found in the low resolution edge list corresponding to blurred and faint edges. Some edges will also be missing because of interference. The ALC traverses the high resolution classified array examining the pixels corresponding to the low resolution step edges. Four edge types are labelled depending on the array contents, as follows:

Low Resolution	High Resolution	Edge Type
step	step	step
step	gradient	blurred
step	texture	complex
step	uniform	faint

The complex type covers the situation where a single edge at low resolution is a result of the interaction of several image entities at the higher resolution. Fig. 4 shows part of the classified array output of two adjacent SLSs superimposed. Each rectangle encloses four high-resolution pixels and one low-resolution pixel (the central one). The part of the image concerned clearly has a step edge running through it.

Conclusion

This paper has described the current status of an edge detection system, in which an attempt is being made to find a way of producing a single coherent representation of all the edges in the image.

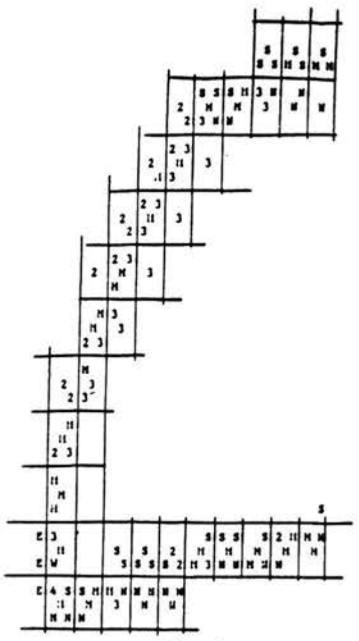


Fig.4

Assuming that the basic structure is to first find edges of different widths separately, then integrate the information into one description, three features seem to be necessary to the system's operation:

1. Precise definition of the classes of entities produced at each resolution (or by each mask of a certain width).
2. A simple (or at least calculable) relationship between the output of the system at one resolution (or mask size) and the underlying image intensities.
3. A clear relationship between the outputs of the system at adjacent resolutions (or mask sizes).

Using the first of the above, the way in which each type of entity changes as the resolution (or mask size) is changed can be worked out. The latter two features are needed to ensure that the corresponding output primitives in different resolution (mask size) outputs can be correctly matched.

References

1. MARR, D & HILDRETH, E. Theory of Edge Detection. Proc. R.Soc.Lond.B 207, 187-217, 1980.
2. ROSENFELD, A & THURSTON, M. Edge and Curve Detection for Visual Scene Analysis. Trans.IEEE Computers C-20 (5), 562-569, 1971.
3. CANNY, J F. Finding Edges and Lines in Images, M.I.T. A.I. Lab. Tech.Report 720, 1983.
4. BARROW, H G & TENENBAUM, J M. Recovery of Intrinsic Scene Characteristics from Images, p 3-26 in Computer Vision Systems, Eds. Hanson, A & Riseman, E, Academic Press, Now York 1978.
5. WITKIN, A P. Intensity-based Edge Classification, Proc. AAAI, 36-41, 1982.
6. ULLMAN, S. On Visual Detection of Light Sources, Biol.Cybernetics 21, 205-212, 1976.
7. HORN, B K P. Determining Lightness from an Image, CGIP, 3, 277-299, 1974.
8. BEATTIE, R J. Edge Detection for Semantically-based Early Visual Processing. Proc.ECAI-82, 1982.