Machine Interpretation of Boundaries in Textured Images

Paul Mackerras

January 1988

A thesis submitted for the degree of Doctor of Philosophy
of The Australian National University.

Computer Sciences Laboratory,
Research School of Physical Sciences,
Australian National University,
Canberra, A.C.T. Australia.

I declare that, except where acknowledgement is given, this thesis is my own original work.

Paul Mackerras 28/1/88

(Paul Mackerras)

Preface

This thesis presents the results of a detailed study of the problem of automatically locating boundaries between regions of textured imagery. This problem is part of the wider problem of segmentation of images, which falls within the area of machine vision. In addition to current research in machine vision, this study draws on the fields of computer science, perceptual psychology, and visual neurophysiology.

The course of this study ran from early 1984 to early 1988 in the Computer Sciences Laboratory within the Department of Engineering Physics, Research School of Physical Sciences, at the Australian National University. The work reported herein was supervised by Prof. Richard Brent, and was largely under the guidance of Dr. Iain Macleod. Financial support was provided by a Commonwealth Postgraduate Research Award.

The chapters are organized as follows: Chapter 1 introduces the area of segmentation in machine vision, and the particular problem of accurately locating the boundaries between textured regions in an image. Chapter 2 reviews various segmentation strategies with particular reference to the accuracy of the location information they provide, and identifies their shortcomings. Chapter 3 discusses three criteria for a good description of a boundary, and develops an algorithm for finding a description satisfying these criteria. Chapter 4 presents results obtained with this algorithm on a number of images, and compares and contrasts these results with those obtained with two convolution edge detectors. Chapter 5 discusses various interesting aspects of the new algorithm. Finally, Chapter 6 summarizes the main conclusions of this study.

Preliminary results from this study were presented at the Australian Joint Conference on Artificial Intelligence, Sydney, November 1987, and were published in the conference proceedings.

Notable contributions of this study include the following:

- Fundamental limitations of convolution edge detection approaches as applied to textured imagery have been identified.
- Three criteria—accuracy, simplicity, and consistency—have been argued as essential for a good boundary description, and formulated in mathematical terms.
- An algorithm for locating and describing boundaries on the basis of these criteria has been developed.
- A consistency test has been devised, and has been used in elaborating boundary descriptions, and in obtaining confidence intervals on the positions of boundaries.
- The consistency test has also been used to test the accuracy of conventional techniques, without knowledge of the true boundary position.
- Extensions of the technique have been suggested, for handling multiple regions, and for modelling spatially-varying regions.

Acknowledgements

I would like to thank Dr. Iain Macleod and Prof. Richard Brent for valuable guidance and encouragement, and for their constructive criticisms of this thesis. In particular, I would like to thank Dr. Macleod for giving his time so generously for many fruitful discussions throughout the course of this study. I would also like to thank Dr. Terry Bossomaier, Dr. Michael Cook, and Prof. Allan Snyder for many interesting discussions on aspects of biological visual systems and perceptual psychology.

Thanks are also due to Mr. Darrell Martens and Mr. Joe Elso for their work in developing the software and hardware of the image display system, which formed a central part of the environment in which this study was conducted. I would also like to thank all the members of the Computer Sciences Laboratory for their friendship during the last four years.

Special thanks go to my wife Alison, for her love, encouragment, and support, and for her assistance in preparing and proofreading this thesis.

Abstract

Existing approaches to the problem of image segmentation do not give accurate and concise descriptions of boundaries in texture images. This thesis presents a new approach for locating and describing such boundaries, which gives substantially improved results. The new approach seeks the simplest description which is consistent with the image data for the boundary between two specified regions.

The Marr-Hildreth and Canny edge detectors have been implemented for the purpose of comparison with the approach developed in this study. These edge detectors gave boundary contours which were irregular and inaccurate when applied to test images containing smooth boundaries between regions of texture. Several reasons for their shortcomings have been identified; the principal reason is that the textured regions contain intensity variations which are irrelevant to the boundary.

The new approach is motivated by three criteria for a good boundary description: accuracy, simplicity, and consistency. These criteria are formulated in mathematical and statistical terms, enabling the development of an algorithm for finding a boundary description which satisfies these criteria.

The accuracy criterion requires that the boundary should be the most likely at the given level of complexity, that is, that it should have the highest likelihood according to a statistical model which allows for a distribution of pixel intensities in each region. The simplicity criterion requires that the simplest description be used which is still consistent with the image data, and is formulated in terms of the number of parameters in a parametric description of the boundary. The consistency criterion requires that there be no areas adjacent to the boundary which are demonstrably on the wrong side of the boundary; this is formulated in terms of a likelihood ratio test.

The algorithm for finding a description satisfying these criteria has four main elements: maximizing the likelihood of the boundary description, testing whether the

description is consistent with the image data, elaborating the description as necessary (using its inconsistencies as a guide), and simplifying the description where possible. Implementation of each of these four steps is discussed in detail.

The results obtained with this algorithm on a number of test images demonstrate the appropriateness and effectiveness of the algorithm, and therefore of the three criteria and their mathematical formulations. The test images include both real-world images and images that have been artificially constructed from natural textures. The results are accurate and concise, displaying no more complexity than the observable level of detail in the image warrants, and are generally in excellent agreement with human perception. These results represent a substantial improvement over conventional techniques.

In contrast to many existing techniques, the algorithm developed here has no critical parameter values that must be set accurately for individual images or classes of images to obtain correct results. The algorithm has two main parameters, both of which were set at the same values for all images tested, namely the significance level in the consistency test, and the maximum scale of intensity variation which will be considered as texture.

The new approach has been applied in this study to the problem of locating a single boundary between two homogeneous textured regions. Extension of the approach to handle multiple regions and boundaries is straightforward. The same approach could also be used to identify distinct regions and to obtain descriptions of the spatial variations within non-homogeneous regions. Thus, the new approach yields a consistent framework for the overall problem of image segmentation.

Contents

List of Figures

List of Tables

List of symbols and abbreviations

BLM	boundary likelihood maximization
GRF	Gibbs random field
LLR	log likelihood ratio
MAP	maximum *a posteriori*
ML	maximum likelihood
MRF	Markov random field
PWL	piece-wise linear
x, y	Cartesian coordinates
N_G	number of grey-levels in image
N_R, N_C	number of rows and columns in image
$G(x,y)$	two-dimensional Gaussian
∇^2	Laplacian operator
σ	scale (standard deviation) of Gaussian
w	half-width at half-height of Gaussian
$I(x,y)$	image intensity function
H_0	null hypothesis
H_A	alternative hypothesis
$h_i(x)$	histogram of grey-levels in region i
$p_i(x)$	estimated pixel grey-level distribution in region i
L	log likelihood
$\lambda(x)$	log likelihood ratio function
λ_{ij}	image of $\lambda(x)$ values
S_{ij}	image of row-sums of λ_{ij}

$\mathbf{z}_k$	position of knot k
$\mathbf{z}_k(t)$	vector function defining segment k
$\mathbf{u}_k$	initial velocity of (cubic) segment k
$\mathbf{v}_k$	final velocity of (cubic) segment k
$r(i,j)$	rotation number of pixel (i,j)
$\mathbf{z}_d$	direction of movement of knot
α	distance that knot is moved
h	height of protrusion
l	log likelihood ratio statistic
μ, σ^2	mean and variance of l under H_0
ν	normalized log likelihood ratio
μ_i	mean of λ-image in region i
$R_i(j,k)$	autocovariance function of λ-image in region i
`autodist`	maximum span for calculating autocovariances
`siglev`	threshold controlling significance level of test
$d_{\max}$	maximum distance to move knot
$h_{\max}$	maximum height of protrusion

Chapter 1

Introduction

Visual perception is a fascinating field of study. Although it is automatic and effortless for most humans, it involves very complex processes, about which little is known in detail, despite years of research. Perception involves interpreting sense data, such as the patterns of light on the retinas, in terms of the real world of three-dimensional objects. In fact, we are usually not directly aware of the sense data.

Vision is our most useful sense for determining the presence, identity, position, and characteristics of the objects around us. Since it is essential for many tasks, it would be an extremely useful capability for a computer to have, and this has been the aim of much research. For example, two important industrial applications are control of robot behaviour and automated inspection of products for quality assurance. What is required is a computational method for interpreting two-dimensional images (patterns of light) in terms of three-dimensional objects in the real world.

The universal conclusion is that mechanizing visual perception is an extremely difficult and complex problem. Consequently, most computer vision research concentrates on limited goals, either on interpretation in a limited domain, or on a single aspect of the process of perception. One important aspect involves image segmentation, that is, finding the boundaries between visually distinct areas of an image. This thesis describes a new method for accurately locating such boundaries in images containing textured regions.

This chapter considers the role of segmentation in an overall strategy for machine vision, the two dominant approaches to segmentation, and finally, the prevalence of texture and the problems that it creates for segmentation algorithms.

1.1 Segmentation in machine vision

Why is visual perception so difficult? A central problem is that it involves relating two quite different things: an image, and some sort of internal model of the object world. The image has little structure; it is simply a two-dimensional array of numbers (or vectors). In contrast, the model must be highly structured, so that it can adequately describe the wide variety of possible objects and their relationships with each other. A second significant problem is that the same object can appear in different ways in the image, depending on its position and orientation in relation to the camera, other objects, and light sources. A third problem stems from the ambiguity which results from representing a three-dimensional world in a two-dimensional image; in principle there are an infinite number of three-dimensional object-world arrangements which could give rise to a given two-dimensional image. In addition, practical problems arise because the amount of data to be handled is substantial.

Most researchers accept that the task of visual perception has to be broken down into sub-problems, if it is to be computationally tractable (Ballard and Brown (1982), Marr (1982)). Breaking the problem down involves creating a series of representations that are intermediate between the image and the object-world description. Each sub-problem involves interpreting some aspects of the current representation, and possibly creating a new representation which is different in form. As the processing proceeds, therefore, more and more 'meaning' is attached to parts of the image, and the representations become more abstract in form.

Before any interpretive steps are performed, a variety of image-processing operations can be used to enhance the appearance of the image or make certain features clearer. These operations transform the original image into one or more additional images. For example, texture filters can be used to enhance the differences between textures. Another example is the determination of a depth image from stereoscopic information.

Segmentation is an important early stage in the interpretation process. It is the step in which pixels begin to be associated with each other on the basis of having similar characteristics, rather than on the basis of propinquity in the image. Its purpose is to identify parts of the image which are likely to correspond to significant features of the arrangement of the object-world, and which are thus likely to be important in forming a satisfactory interpretation of the image. These important parts of the image include

the following:

- Regions which are homogeneous with respect to intensity or texture, and are thus likely to correspond to surfaces of objects
- Sharp changes in intensity or texture, corresponding to contours where one object occludes or abuts another
- Changes in intensity or texture corresponding to
  - the junction of two surfaces of an object
  - a surface receding from the camera
  - surface markings
- Slow variations in intensity, corresponding to changes in distance and aspect of a surface with respect to the light source(s) and the camera
- Changes in intensity corresponding to shadows (occlusion of the light source(s) by another object).

Thus, *edges* and *regions* are important parts of the image, and there are two main approaches to segmentation: edge-based and region-based. Edge-based approaches attempt to identify edges, which are contours across which the characteristics of the image change abruptly. These are likely to correspond to the edges of surfaces, and are therefore likely to be important in determining the shapes of the visible objects. On the other hand, region-based segmentation algorithms concentrate on identifying regions of the image which are homogeneous in some sense, and can therefore be expected to correspond to the surfaces themselves.

Segmentation involves deciding that certain parts of the image are similar, and therefore belong together, and that other parts are different. At this early stage, it is not possible to avoid making some wrong decisions, because correct decisions may depend on interpretations put upon other parts of the image, or on use of object-world knowledge. For example, if analysis of some parts of an image suggests that it portrays a cube, then a small intensity difference between two areas of the image might be important in determining the location of the edge where two faces meet. This difference could well be dismissed as insignificant if the global interpretation was not available.

It is therefore highly desirable for the segmentation step to be able to interact with higher-level processes, so that incorrect decisions can be subsequently corrected.

In the example of the cube, the segmentation process could then report the intensity difference which had previously been dismissed as insignificant. It is necessary for the segmentation step to make some initial decisions, so that higher-level processes can begin to operate, but it is also important that the initial decisions be subject to revision once more information has been accumulated. In ambiguous situations, it may also be possible for the segmentation process to produce a variety of possible interpretations.

Once the image is segmented, there still remain many significant problems in the interpretation process. The image is organized to an extent, but still in two-dimensional terms. Some of the remaining tasks are to group regions of the image into objects, determine their three-dimensional shapes, positions, and relations to other objects, and to identify the objects. Segmentation is a useful step because it creates a representation in which most of the important features of the image are represented explicitly, and also reduces the quantity of data which has to be handled.

There is an important distinction between determining that two adjacent parts of the image are different, and determining precisely where the boundary between them lies. The first is defined here as the *detection* problem, and the second, the *location* problem. A complete segmentation strategy requires a good solution to both, although the specific difficulties in determining an accurate boundary location have often been glossed over. This thesis addresses itself particularly to the location problem.

1.2 Edge-based approaches

Edge-based segmentation approaches attempt to find the boundaries between regions by determining the places in the image where there are significant changes in intensity. Many different variations have been tried, but the overall strategy is as follows. First, local edge elements are detected. These are places where a local discontinuity in image intensity appears to exist. These edge elements may have an associated strength, orientation, and/or length, and are usually represented in the form of an image (*edge map*). They are then grouped together into edge contours, which may then be approximated by a parametric representation of the contour (such as a piece-wise linear or cubic-spline representation). These edge contours could then correspond to the edges of objects, or to any of the other causes of intensity variations listed above. They do not, in general, divide the image into disjoint regions.

The edge elements are obtained by application of one or more local operators at points covering the whole image. These operators, or edge detectors, compare the intensities in adjacent parts of the image, and signal the degree of difference between them. In many cases, two or more operators sensitive to different orientations of edge are used. The results are usually combined in some simple way, for example, by taking the operator giving the largest output at each point.

Further processing at this stage may include a thresholding operation to reject weak edge elements. Such an operation is useful because it reduces the quantity of data, and also tends to eliminate most of the responses which are due to noise or texture in the image, while retaining the most obvious edges. However, it has the danger that a weak, but vital, edge will be also eliminated; it is not possible to discriminate between desirable and unwanted edge elements purely on the basis of their strengths.

An important parameter of an edge detector is the size of its support, that is, the area of the image which is used in applying the edge detector at a given point. Small-support edge detectors have the advantage that they are simple and efficient to use, but they have only limited ability to discriminate between variations due to noise or texture and true edges such as object boundaries, particularly in images containing coarse textures. Consequently, they have the problem of tending to report many edge elements within a uniformly textured region. Relaxation techniques can be used to improve the quality of the edge map by iteratively adjusting the edge element strengths according to the local evidence for an edge contour. Such techniques can help with this problem to a limited extent, by reducing individual responses due to noise or fine textures, but they cannot cope effectively with coarse textures.

Another solution to this problem is to use an edge detector with larger support, allowing more information to be taken into account at each point. Such large-support edge detectors usually involve smoothing the image by taking a weighted average of the neighbouring pixels at each point, before using a derivative or derivative-like operator. The smoothing operation reduces the magnitude of the variations due to noise or texture without reducing the magnitude of edges between regions of significant size. This method can cope with arbitrarily coarse textures, provided that a large enough support is used. However, the smoothing blurs the edges, resulting in the edge detector output being high over a thick band or ridge around the edge position. Usually this is overcome either by a 'non-maximum suppression' step, in which all pixels not at the top of a ridge

are set to zero, or by a 'thinning' step, in which broad bands of edge pixels are reduced to their central axes. The blurring makes it more difficult to determine the precise location of an edge, and can cause problems at sharp corners in an edge contour and at the junction of three or more edges. In some cases, several different sizes of edge detector are used; there is then the non-trivial problem of combining the results.

Edge-based approaches to segmentation generally do not differentiate between the location and detection problems. The assumption is made that the edge detector will respond most strongly at the correct location of an edge. This assumption is often incorrect, especially when large-support edge detectors are used on textured images, as the following chapters show.

1.3 Region-based approaches

In contrast to edge-based approaches, region-based segmentation approaches attempt to locate the regions themselves rather than their boundaries. Regions are usually defined to be areas of an image which are homogeneous in some sense. These approaches are complementary to edge-based approaches in the sense that a set of regions and their boundaries are duals: each can be determined from the other. However, the edge contours obtained using an edge detector need not divide the image into disjoint regions, whereas region-based approaches generally do. If the image is divided into disjoint regions, the segmentation can be expressed in the form of a 'segmented image', in which each pixel contains the region number to which it belongs.

Having the image divided into disjoint regions has advantages for higher-level processes if they wish to identify regions with surfaces. The incomplete boundaries obtained with edge-based approaches will generally have to be completed, since surfaces are bounded on all sides. In contrast, regions have complete boundaries by definition, and are perhaps closer to the representation used at higher levels.

There are, however, some dangers in attempting to divide the image into disjoint regions at an early stage, particularly if the image contains regions with gradual variations in brightness or texture, or if there are smooth transitions between regions. Most studies assume that the regions obtained must be uniform in brightness or texture, and therefore have difficulty coping with brightness or texture gradients. Regions with slow variations in their characteristics therefore tend to be broken up into arbitrary

pieces in which the extent of variation is small enough for the region to be considered homogeneous. This may well involve placing boundaries where none are visible in the image.

Later processing stages therefore must be prepared to merge some regions and split others. As with edge-based schemes, faint but vital boundaries can be missed. Thus, a region may not be split when it should be, because decisions as to whether a region is homogeneous are most often made on the basis of the magnitude of intensity differences within the region. Dividing the image into disjoint regions at this stage thus tends to involve making firm decisions on the basis of insufficient information.

These approaches are concerned with the properties of regions, such as the range of grey levels present, and whether or not a region can be considered homogeneous, according to an appropriate test of homogeneity. Therefore, the basic operations used include the following:

- Testing whether a region (or the union of two or more regions) is homogeneous
- Splitting a region into two or more regions
- Merging two or more regions into one
- Classifying a region as one of a number of classes

The simplest technique for splitting an image into disjoint regions is thresholding, which assigns each pixel to one of two regions if it is above a given threshold, or to the other if it is below. In essence, this technique involves independently classifying each pixel as belonging to one of two classes. The technique can be generalized to handle vector images (in which each pixel has an associated vector, rather than a simple scalar value), and to handle several classes. In some cases, small windows are classified rather than individual pixels. The classes may be determined by some form of cluster analysis, e.g., by searching for a valley in a histogram. Because the pixels or windows are classified independently, these techniques generally give poor results; the regions tend to be disconnected, and the boundary locations tend to be inaccurate.

These deficiencies can be remedied to a certain extent by using statistical models for the shapes of the regions. Statistical estimation techniques can then be used to determine a segmentation of the image by estimating the shapes of the regions, taking into account both the image data and the preference for connected regions with smooth boundaries. Such techniques effectively make the classification of each pixel (or window)

dependent on the classifications of its neighbours, and therefore produce better-formed regions than those which classify pixels or windows independently. However, there are many problems with this approach; perhaps the principal one is the difficulty of obtaining adequate statistical models for the region shapes.

Instead of classifying individual windows, another strategy is to split and/or merge regions until they are homogeneous. Region-growing techniques start with each pixel as a region, and then merge regions together as far as possible, whereas region-splitting techniques recursively split regions until each subregion is homogeneous. The split-and-merge technique combines these two ideas to obtain better results than either can give alone. It involves splitting initial large regions until each region so obtained satisfies the uniformity test, and then merging adjacent regions as far as possible. The merge phase produces regions that are not broken up to the same extent as with region-splitting algorithms.

A major problem encountered by most region-based segmentation approaches is the 'small region problem'—as the regions become smaller, the statistical tests used to determine whether the region is homogeneous, or which class it belongs to, become less reliable. This problem results in inaccurate boundary location information, and it can also result in many small regions being reported along the boundary between two regions.

Region-based approaches generally do not explicitly attempt to obtain a good solution to the location problem. The boundaries of the regions are usually represented implicitly in the segmented image, and no particular effort is expended to ensure their accuracy. Consequently, the regions are often rather rough and 'blocky' in outline.

1.4 Texture

The segmentation approaches described above have mostly been developed with the assumption that images consist of regions of approximately uniform brightness. However, images of real-world scenes usually contain textured regions; the problem of texture can be completely avoided only in highly artificial and constrained situations. Outdoor images (or any images of objects of natural origin) will almost certainly have textured regions. Brodatz (1966) gives many examples of textured surfaces, including, for example, grass, wood, cloth, and leather. Other examples include concrete, carpet,

and the leaves of a tree (observed from a distance).

Texture can arise in a number of ways, for example:

- as a result of microstructure in the surface of an object, e.g., the grain in wood or leather
- as a result of considering many similar small objects as forming a larger surface, for example, the individual blades of grass in a lawn
- as a result of many small markings covering a surface.

The distinctions between these sources of texture are not sharp. Texture can even be hierarchically arranged, as in a texture made up of objects which are themselves textured.

There is a continuum in the degree of regularity of textures: they range from being quite random to completely deterministic. Most textures have a degree of randomness while still retaining a deterministic component. An example of this is the grain in wood, in which the individual streaks are somewhat randomly placed, but are all oriented in approximately the same direction. An example of a completely random texture is noise introduced in the camera or during subsequent storage, transmission, or processing of the image. Although such noise does not correspond to any characteristic of a surface, we can include it as a special case of texture, because the techniques developed in subsequent chapters for dealing with texture can deal equally well with such noise.

There are two main approaches to describing and analyzing texture: statistical and structural. Statistical approaches generally describe the texture in terms of the distributions of pixel intensities and the dependence between pixels, or in terms of other similar statistics. Structural approaches, in contrast, attempt to obtain placement rules describing the structure of the texture, and are really suitable only for textures which are largely deterministic. Statistical approaches are applicable both to random and deterministic textures, although they will generally not capture as much of the structure of a deterministic texture as a structural approach.

A texture can be thought of as being composed of 'texture elements', which are the components which are repeated to make up the texture. These elements can be defined with respect to either the image or the object-world surface; in this study, we are concerned primarily with image texture, so the term 'texture' generally refers to texture in a two-dimensional image. The elements of an image texture may or may

not correspond to identifiable objects or structures. If the texture is composed of identifiable objects, or of structures which can be interpreted in three dimensions, then these objects or structures can be regarded as the texture elements. If, however, there are not distinguishable objects (or the image resolution is too coarse to resolve them), the texture elements can be regarded as being the patches of similar grey level which make up the texture. These texture elements can be many pixels across, or they can be as small as single pixels.

Textures vary along a continuum from coarse to fine. The coarseness of a texture is a function of the size and spacing of the texture elements. This relates also to the degree of correlation between adjacent pixels, and the distance over which substantial correlation exists between pixels. The coarseness of a texture is described by the 'scale' of the texture, which is informally defined here as the distance over which it is necessary to take into account the correlation between pixels, that is, the distance beyond which pixels can be regarded, to a sufficiently good approximation, as uncorrelated.

In describing a textured surface, there are at least two levels of description: a higher level, at which the surface is described in terms of its overall shape and other gross properties (e.g., illumination), and a lower level, at which the individual texture elements are described. A textured surface is thus not just an arbitrary collection of small objects: there must be a sense in which the surface is homogeneous over a considerably larger distance than the size of the texture elements. For example, if the texture is due to roughness of a surface, the scale of the relief should be considerably smaller than the width of the surface. One of the characteristics of a textured region is therefore a degree of homogeneity in the texture.

It is generally the higher level of description which is most useful to later stages of processing in interpreting the image. Thus, the term 'texture' is informally defined here as referring to intensity variations which are merely part of the appearance of a surface. That is, the variations are not individually significant in forming an interpretation of the image (although the overall characteristics of the texture may be). Thus, when the image is segmented, a textured region should be regarded as a unit. This frees later processing stages to concentrate on the higher-level aspects of the shape and orientation of the surface, and its relation to other surfaces of the same object.

Usually, the microstructure of the surface is not of immediate interest; if it becomes of interest, the image then needs to be re-examined at a finer scale, so that the individual

texture elements are segmented from each other. An important parameter for a texture segmentation procedure is therefore the scale at which the image is to be examined, i.e., an indication of how coarse a texture can be and still be regarded as a texture. Initially, this would be set at a large value, to extract the broad outlines of the arrangement of the scene. Later, individual regions could be further segmented at a finer scale.

However, texture usually causes significant problems to segmentation algorithms which are designed to segment images on the basis of differences in intensity. Edge detection algorithms generally give poor performance, tending to give many unwanted responses to intensity variations within the texture, and also reporting inaccurately the positions of boundaries between textured regions. Small-support edge detectors, and in particular those limited to a support of 3×3 pixels, are virtually useless for dealing with textures—they cannot cope with any except very fine textures, and there is no provision for specifying the scale of texture. Edge detectors of sufficiently large support do rather better at discriminating between textural intensity variations and region boundaries, but the broad smoothing required causes inaccuracy in reporting the positions of region boundaries, and also causes problems where three or more edges meet. Region-based techniques can adequately discriminate region boundaries from textural variations if they are constrained to consider windows of a sufficient size, but once again, this causes considerable inaccuracy in the locations of boundaries.

The main problems in segmenting textured images are therefore discriminating between textural intensity variations and region boundaries, and accurately determining the locations of these boundaries. These are the detection and location problems defined above. A major contribution to the difficulty of these problems is that the pixels in each region can have a range of intensities, and can be correlated over significant distances; the ranges of intensity in adjacent regions can overlap. Current techniques can give adequate solutions to the detection problem, provided that inaccuracy in the boundary location can be tolerated, but there are no existing techniques which can give accurate location information. This thesis therefore considers the problem of accurately determining the location of a boundary between textured regions.

Humans generally have the ability to discriminate between textures that have the same distribution of pixel intensities, and therefore the same average brightness. Such textures cannot be discriminated by standard edge detection techniques, and consequently much previous research effort has been focussed on obtaining techniques for

discriminating them. The usual strategy involves two steps. The image is first processed with some kind of texture analyzer, which examines the texture in local patches of the image, and produces statistics for each patch describing the texture in the patch. These can be thought of as forming another image, possibly of lower resolution and/or with vector-valued pixels. This image is then segmented by strategies very similar to those used on grey-scale images.

The texture analyzer makes the difference between the regions obvious as a difference in the average value of some statistic on the texture, that is, some textural measure or 'feature', and thus enables the regions to be discriminated. Many suitable features have been proposed. However, in most cases the statistical variations in a textured region will cause the value of the feature to fluctuate within the area occupied by the region. Consequently, the corresponding region in the feature image will also be textured—the texture analyzer does not eliminate the textured nature of the region. It may, perhaps, reduce the extent of variation within the texture, but it will not entirely remove it.

Therefore, the problem of segmenting a textured image remains. It has not been solved by the application of the texture analyzers. The analyzers have made it easier, or possible, to discriminate differently textured regions, but they will not eliminate the problems due to texture. The problem of segmenting regions with *different* intensity distributions is therefore the central problem in texture segmentation; the problem of segmenting regions with the *same* intensity distribution is a subproblem, solved in general by the prior application of a suitable texture filter or analyzer. In fact, casual inspection of a range of real-world images indicates that adjacent regions with similar intensity distributions but different textures are relatively rare.

Chapter 2

Review of previous studies

The area of segmentation in machine vision is one which has been extensively studied over more than two decades, and the volume of published studies in the field is substantial. This chapter therefore concentrates on the problem of segmenting textured images. Some standard grey-level segmentation algorithms are useful in this context, both as applied directly to images where the regions differ in their grey-level distributions, and as applied to the output of a texture analyzer. Some appropriate edge-based and region-based approaches are considered below in sections 2.1 and 2.2. Texture analysis methods are necessary where the regions do not differ in their grey-level distributions; section 2.3.1 discusses methods which are suitable for use in segmentation. Section 2.3.2 then discusses several studies where these two elements have been put together in a texture segmentation algorithm. Finally, section 2.4 discusses a relatively new approach to segmentation which views segmentation as a statistical estimation problem.

Because of the extent of the literature, this chapter concentrates on the better approaches which have been developed in each area so far, and on those studies which are representative of different approaches to the problem of segmentation. The various techniques discussed below are considered particularly in terms of their ability to cope with texture, and the accuracy of the boundary location information they provide.

2.1 Edge detection

One major approach to segmentation involves finding edge contours in the image by detecting edge elements and linking them together. The problem of detecting the edge

elements is not at all trivial. It has long been a subject of interest, and many different edge detectors have been proposed; for reviews see Davis (1975) and Levialdi (1981). Because of the huge variety of different edge detectors which have been proposed, a comprehensive review of the field has not been attempted here. Instead, several different approaches are mentioned briefly. Two large-support convolution edge detectors are considered in more detail, as they have been implemented in this study for the purpose of comparison with the methods described in following chapters. The problem of linking edge elements together has received rather less attention and is not considered in detail here.

An obvious first approach to the problem of detecting edge elements is to search for places where adjacent pixels differ by more than a suitable threshold. A generalization of this approach is to use one or more convolution masks of small size, e.g., 3×3 or 5×5 pixels, and combine the outputs in some way. Many different edge detectors of this type have been proposed; some well-known examples are given in Frei and Chen (1977) and Roberts (1965). However, as discussed previously, the small support of these edge detectors means that they cannot give correct results on other than fine textures; there is no provision for specifying the scale of the texture. Therefore, small-support edge detectors are not considered further.

One way to approach the problems caused by noise and texture is to smooth the image, e.g., by low-pass filtering, before applying an edge detector. This will reduce the contrast of the textural intensity variations, but not the contrast of the edges, provided that the adjacent regions are sufficiently broad. However, it will blur the edge; consequently, the edges are usually taken to be at the points of local maximum of gradient in the smoothed image. The operations of linear smoothing and differentiating (taking the gradient) can be combined into one convolution mask, since both are linear. Such detectors have large support and can operate at a range of scales, and are therefore suitable for use on textured images. They are referred to here as convolution edge detectors. Two examples, due to Marr and Hildreth (1980) and Canny (1986), are discussed in detail below.

Several non-linear approaches have also been suggested, many of which were originally described with a small support, but which could be generalized to large support. Notable amongst these are methods which attempt to fit an ideal edge profile to a window of the image (Hueckel, 1971, 1973; Nalwa and Binford, 1986), methods based

on use of moments (Machuca and Gilbert, 1981; Tabatabai and Mitchell, 1984), and methods based on the use of statistics other than the mean, such as medians or linear rank sums (Bovik and Munson, 1986; Bovik *et al.*, 1986). These are not considered further, because published results indicate that none of these methods give dramatically improved results over the convolution edge detectors mentioned above (although they may give minor improvements), as illustrated by the images given in the papers listed above and the quantitative comparisons in Suciu and Reeves (1982).

In addition, all of these methods (including the convolution edge detectors) share the disadvantage of assuming that *only one edge is present in the window.* Most methods also assume that the edge is a straight step edge between regions of constant grey-level (possibly with some added noise). The problems demonstrated in subsequent chapters with the convolution edge detectors can therefore be expected to occur with all of these methods.

Several methods have been proposed which use multiple scales or sizes of edge detector at each position, and combine the results in some fashion; for example, Davis and Rosenfeld (1974), Marr and Hildreth (1980), and Canny (1986). However, the results shown in figures 2.1 to 2.3 indicate that both the Marr-Hildreth and Canny edge detectors provide poor location accuracy in textured images over a wide range of scales. Consequently, the use of multiple scales cannot be expected to provide significant improvements in location accuracy.

Once the edge elements have been obtained, they need to be linked together to form continuous edge contours. The difficulty of this problem depends on the quality of the information obtained from the edge detector. Where small-support edge detectors are used, relaxation-based schemes, such as those of Zucker *et al.* (1977) or Hanson and Riseman (1978) can be used to advantage. Other approaches include using heuristic search or dynamic programming (see Ballard and Brown (1982, chapter 4) for a discussion of these approaches). However, one advantage of large-support convolution edge detectors is that they tend to give continuous strings of edge pixels without clutter, and so the edge linking problem is straightforward. Linking algorithms are therefore not considered further here.

Mansouri *et al.* (1987) describe an approach in which the edge elements that are detected are straight lines of various lengths. This approach to some extent circumvents the need to link edge elements, if the boundaries of interest are straight. They use a

hypothesis prediction/verification paradigm in which straight lines of a given length are predicted to occur at various positions, and these hypotheses are then tested. Initially, the image is processed with a small-support edge detector, which gives an gradient magnitude and direction at each pixel. After thresholding on the gradient magnitude and thinning the resulting image, a straight line is hypothesized to occur at the location of each surviving edge pixel. These lines are of a given length (typically 55 pixels) and are oriented perpendicular to the gradient direction at the edge pixel. These hypothesized lines are then tested by assessing the uniformity of the gradient direction along the length of the lines; if the directions are sufficiently uniform, the hypothesis is accepted. This process is repeated for various lengths of lines, and the results are combined by deleting shorter segments which form part of longer segments. The results indicate that this approach can detect long, straight edges in an image without texture, but misses short and/or curved edges.

2.1.1 The Marr-Hildreth edge detector

Two edge detectors have been implemented in this study, for the purpose of comparison with the method presented in chapter 3. The first of these is the Marr-Hildreth edge detector. It was chosen because of its widespread use, and the claims for near-optimality made on its behalf (e.g., Lunscher, 1983).

The Marr-Hildreth theory of edge detection (Marr and Hildreth, 1980) proposes that the image should be smoothed with Gaussian masks at a range of scales. Edges are marked at the zero-crossings of the Laplacian of these images; the Laplacian operator, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, is a non-oriented second derivative operator, thus its zero-crossings represent maxima and minima of the gradient. The Laplacian and Gaussian smoothing operations can be combined into one operation, namely, convolution with

$$\nabla^2 G(x,y) = \frac{-1}{\pi\sigma^4}(1 - \frac{x^2+y^2}{2\sigma^2}) \exp(-\frac{x^2+y^2}{2\sigma^2}) \qquad (2.1)$$

where σ represents the scale of the smoothing.

The zero-crossings can be shown to form closed, non-intersecting contours, only one pixel wide, and are unchanged by additive or multiplicative changes to the image. The edges obtained with this detector therefore have the desirable property of being line-like. In practice, it turns out to be necessary to threshold the zero-crossings, because a uniform field with some noise will tend to have a certain density of zero-crossings

(depending on the scale of the detector), independent of the amplitude of the noise. The slope of the $\nabla^2 G * I$ surface across the zero-crossing depends on the magnitude of the edge, so this is a suitable parameter to use in the thresholding. (Here I denotes the original image, and the symbol '$*$' denotes two-dimensional convolution.)

Marr and Hildreth proposed that the operator should be applied at several scales, and that important edges in the image should give zero-crossings at the same place across a range of scales. In practice, this is not so, because neighbouring edges interfere (Shah *et al.*, 1986). At large scales, edges of opposite polarity repel each other; that is, the reported edges for a narrow bar are too far apart. Edges of similar polarity (as in a staircase function) attract each other and merge into one at large scales, and produce a spurious edge between them at small scales. Methods for combining the results from several scales are still a subject of research, of which Hildreth (1985) gives a summary.

Although the Marr-Hildreth edge detector has the advantage of good noise resistance when a large support is used (large σ), it has some disadvantages. In particular, because the zero-crossings form closed contours, this edge detector can never report one boundary segment crossing or meeting another; instead, one of the segments will veer away to avoid meeting the other. In order to close the contour, this edge detector will sometimes also report zero-crossings where there is very little visual evidence of a boundary.

This edge detector also gives inaccurate boundary location information in textured images. Figure 2.1 shows an image with two textured regions, which was obtained from an image of a uniform grass texture by halving the intensity of the pixels to the left of the boundary, producing an effect very similar to a shadow. The shape of the boundary is two straight lines meeting at a corner, which is the shape correctly reported by human observers.

Figure 2.2 shows the results obtained with the Marr-Hildreth edge detector at four different scales, corresponding to $\sigma = 2$, 4, 8, and 16 pixels. These results have been thresholded, as described above, so as to eliminate some of the unwanted responses to textural intensity variations, without eliminating the desired response to the boundary between the regions. Note that the reported edge position deviates from the true boundary position at all scales. The edge contour also tends to break up. These problems are mainly due to the intensity variations within the texture. At the smaller scales, it is primarily the local intensity fluctuations which cause the edge position to

deviate, and also to break up at intervals. At the larger scales, similar problems are caused by the broader intensity fluctuations, and in particular by the dark patch just above the corner. Note also that the proximity of the edge of the picture causes the ends of the desired edge contour to curl over at the larger scales.

The deviations of the desired response from the true boundary do perhaps have the smallest amplitude at the smallest scale. However, the changes in angle of the desired edge contour are just as severe at the smallest scale as at any other, and the irregularity of the edge contour is greatest. In addition, it would be almost impossible to select the desired edge contour without knowledge of the true boundary position. There is an apparent difference between the regions at the smallest scale; the edges are stronger in the right-hand region because the contrast is higher. However, the response to the true boundary is not distinguishable from the textural responses in the right-hand region.

Hildreth (1985) and Canny (1986), among others, have suggested that important edges should be detected using large-scale detectors, and localized using small-scale detectors. Certainly, the desired response in figure 2.2 is most obvious at the largest scale. However, given the inaccuracy of the reported location at the larger scales, and the irregularity and fragmentary nature of the edge contour at smaller scales, any scheme for combining the results from multiple scales is unlikely to give substantially improved results.

Kass *et al.* (1987) describe a non-traditional approach to the problem of locating contours in an image. Their approach involves finding the curve with minimum energy, where the energy of a curve has three components: internal energy, external constraint energy, and image energy. The internal energy term seeks to keep the curve as smooth and straight as possible, except at points of discontinuous slope, which can be specified manually. The image energy term can be defined so as to attract the curve towards edges in the image, light or dark features, or the terminations of linear features. Many forms are possible for the image energy term; results are given using the Marr-Hildreth edge detector to provide a term which attracts the curve towards the zero-crossings detected at a given scale. The external constraint energy term provides a mechanism for manual control of the curve (or control by higher-level interpretive processes).

Minimization of the total energy proceeds continually, so the curve can track dynamically changing images. The curve can therefore track the zero-crossings in the output of the Marr-Hildreth edge detector as the scale is varied from a large value to

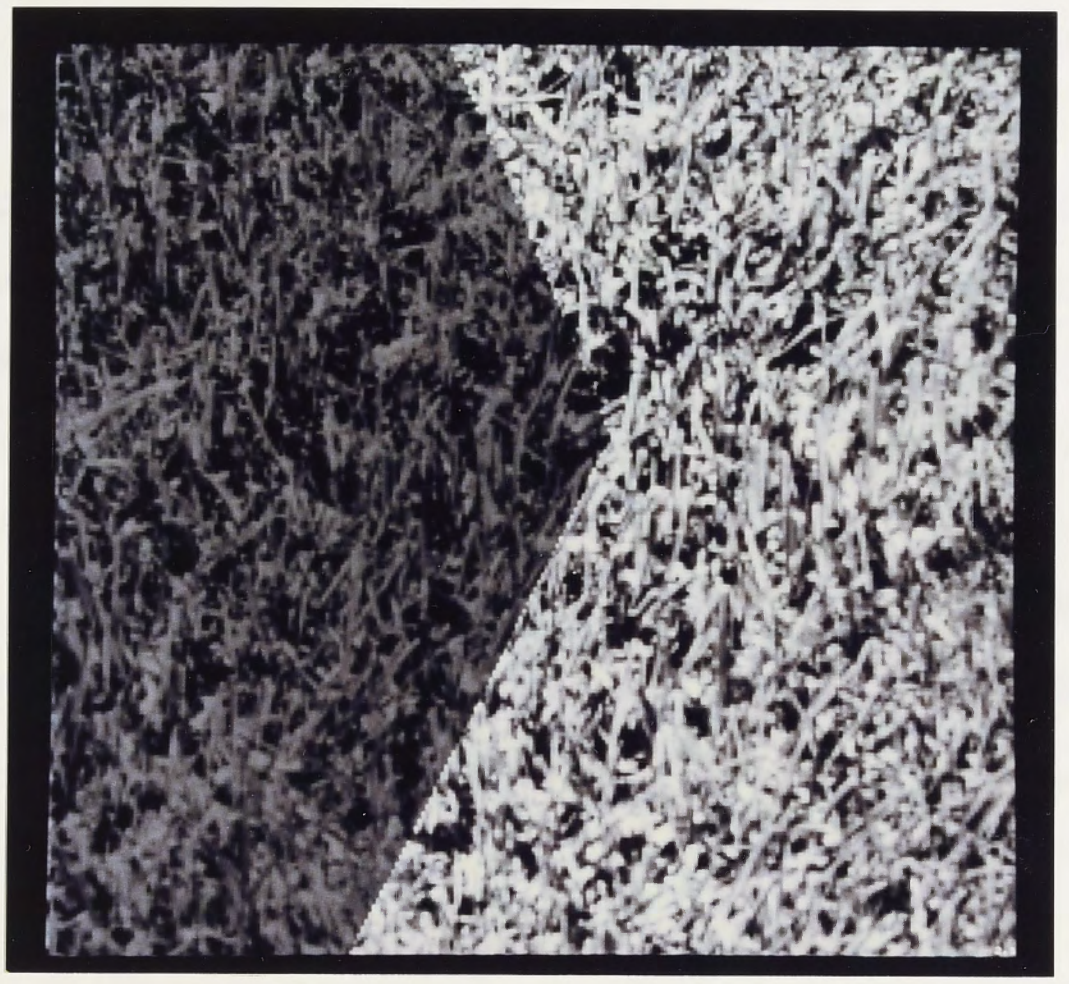


Figure 2.1: Image with two textured regions

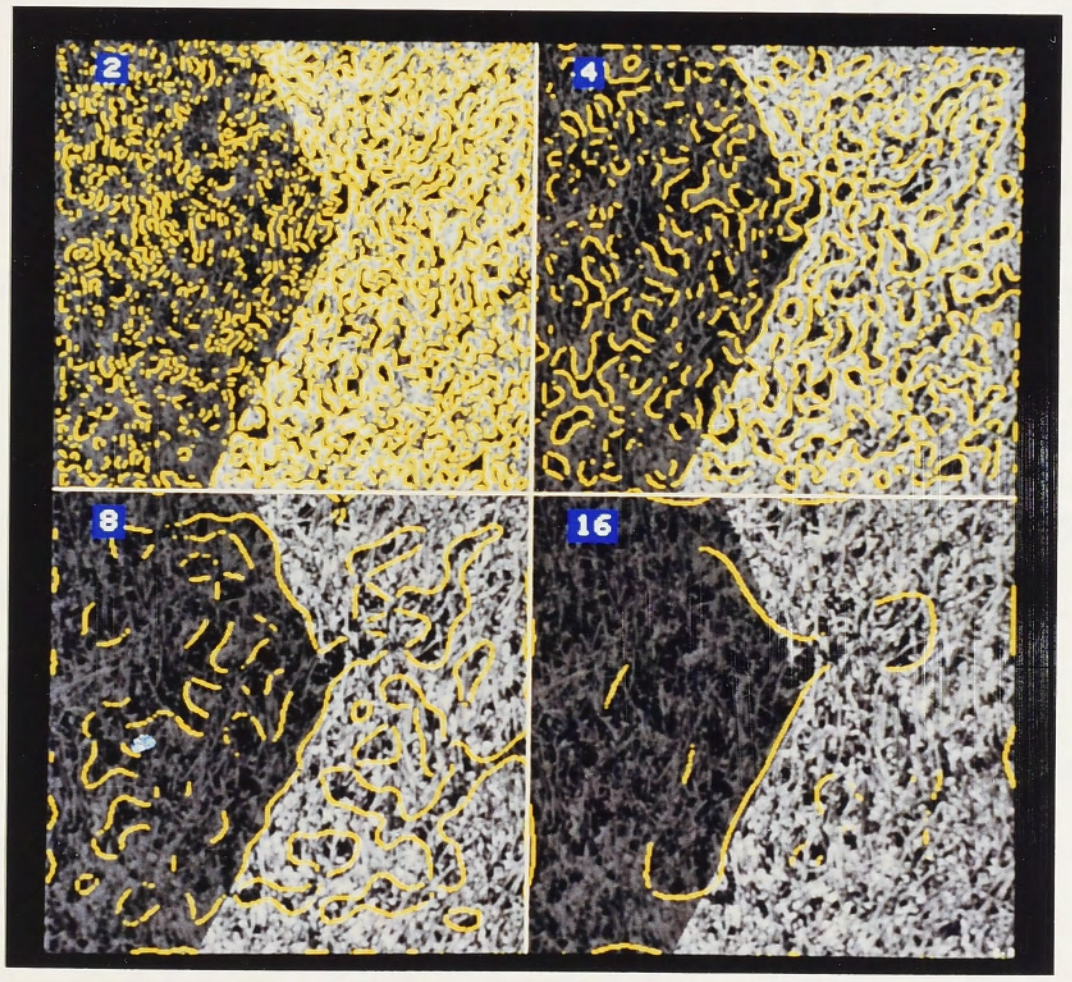


Figure 2.2: Results from figure 2.1 with Marr-Hildreth edge detector

a smaller value. Kass *et al.* (1987, figure 4) show results where this technique has been used. At the broadest scale, the curve is attracted towards the correct boundary from a considerable distance away, but the blurring results in an inaccurate curve. As the scale is reduced, the curve moves towards the correct boundary. At the finest scale, the zero-crossings are somewhat irregular, but the tendency for the curve to stay near a string of zero-crossings, combined with its preference for smoothness, lead to a result which is substantially correct (the true boundary of the object is fairly smooth). This approach therefore represents one way to combine the results from multiple scales; however, it will probably only work well for objects with smooth boundaries. In addition, the results in figure 2.2 show that this approach may well lock on to an incorrect position near the vicinity of the corner in this case, because the zero-crossing contours at the coarser scales deviate around the dark patch just above the corner.

Examples of the results of the Marr-Hildreth edge detector on other images, and a quantitative evaluation of its accuracy, are given in chapter 4.

2.1.2 Canny's approach to edge detection

The edge detector suggested by Canny (1986) has also been implemented for comparison purposes. Canny approached the problem of edge detection by formulating three criteria for a good edge detector, and determined the optimal one-dimensional convolution edge detector according to these criteria. The optimal edge detector for step edges can be approximated very closely by the first derivative of a Gaussian.

For two dimensions, the impulse response is the product of the one-dimensional impulse response with a projection function in the other dimension. This edge detector is orientation-specific; several orientations at each scale are used. Canny suggests two variations. The more complex variation, which involves an extended projection function and several orientations, is not discussed here. The simpler one uses a Gaussian as the projection function, with the same value of σ as the Gaussian in the edge detector function. This profile is very similar in form to one previously suggested by Macleod (1972). Two orientations are used, horizontal and vertical, with profiles given by

$$G_x(x,y) = \frac{-x}{2\pi\sigma^4}\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{2.2a}$$

$$G_y(x,y) = \frac{-y}{2\pi\sigma^4}\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{2.2b}$$

Note that G_x is most sensitive to vertical edges, and G_y to horizontal edges.

Convolution of the image I with these masks gives images I_x and I_y, which can be considered as giving the gradient in x and y directions of a smoothed version of the original image. At each point, these give a vector giving the direction and magnitude of the gradient of the smoothed image. An edge is then defined to exist if the gradient magnitude is a maximum along a line in the gradient direction.

Canny's proposed criteria for a good edge detector are of interest. They are:

1. Good detection: a low probability of failing to detect edge points or reporting false edge points.
2. Good localization: the positions of the reported edge points should be accurate.
3. There should be only one response to a single edge.

Thus, Canny distinguished between the detection and location problems, and recognized the importance of accuracy of location information. He showed that, in the presence of noise, there is a trade-off between criteria 1 and 2; a large detector will have good detection performance but poor localization, whereas a small detector will give good localization, but poor detection.

Figure 2.3 shows the results obtained with the Canny edge detector when applied to the image in figure 2.1. Four different scales were used, with the half-width at half-height w of the underlying Gaussian being 2, 4, 8, and 16 pixels. This is related to σ in equations (2.2) by $\sigma = w/\sqrt{\ln 4} \approx w/1.177$. The results shown have been thresholded on the magnitude of the gradient to give close to optimum results. Although the results are better than those for the Marr-Hildreth edge detector, they display similar kinds of problems, and similar comments apply. Chapter 4 gives results obtained on other images and a quantitative evaluation of accuracy.

2.2 Region-based segmentation

In contrast to the edge-based approaches discussed above, region-based segmentation techniques seek to obtain areas of the image with certain characteristics. Haralick and Shapiro (1985) provide a survey and comparison of these techniques. They divide them into measurement-space guided clustering schemes (i.e., pixel classification and related schemes), region growing schemes, spatial clustering schemes (a combination of pixel classification and region growing), and split-and-merge schemes. Of these, the

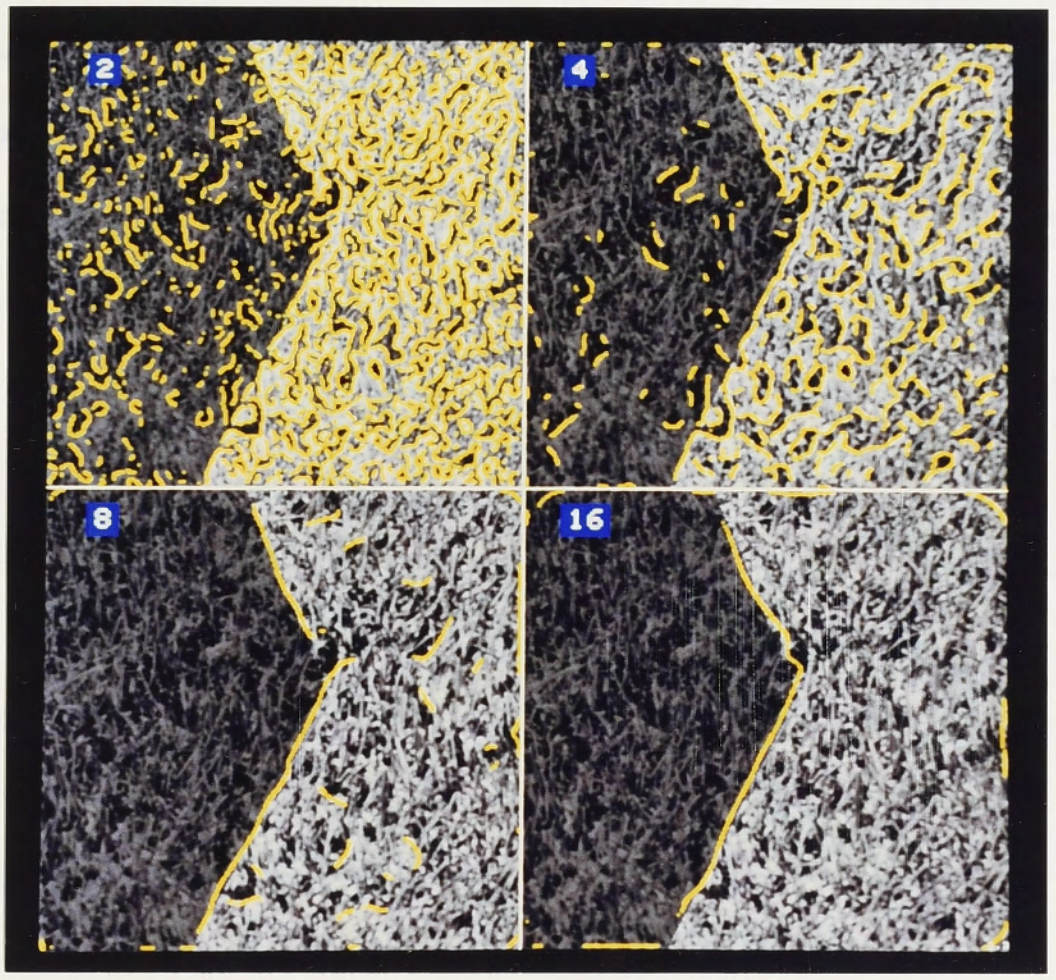


Figure 2.3: Results from figure 2.1 with Canny edge detector

window classification and split-and-merge schemes have been used as a component of texture segmentation algorithms in the studies cited here. These two approaches are examined in some detail below. Section 2.4 discusses statistical segmentation methods involving the use of Markov random fields.

2.2.1 Pixel classification

One simple way to segment an image is to classify each pixel as belonging to one of several regions. In its simplest form, the classification becomes simply a thresholding operation. This approach can be used on vector data (i.e., where each pixel has several components); the classification then becomes a partitioning of multi-dimensional space. In some cases, windows are classified, rather than pixels, which gives a more reliable classification at the expense of a less accurate boundary position. Pixel classification is not suitable for use on textured images, since it will tend to fragment the regions. Window classification is more suitable, since the size of the windows can be varied according to the scale of the texture. Although window classification is not often used for segmenting an image on the basis of intensity, it is often used on the output of texture analyzers (e.g., Kashyap and Khotanzad, 1984).

Because the classification step requires prior knowledge of the regions, it is therefore a boundary *locator*, rather than a boundary *detector*. That is, the number of regions, and their characteristics, have to be determined beforehand. This information is generally obtained by some form of cluster analysis. In the usual case of scalar pixels, this amounts to examining a histogram and, for example, inferring the presence of two distinct regions if there are two peaks. Such procedures are error prone, particularly in the case of textured images, where distinct regions may contain overlapping ranges of grey levels.

Haralick and Shapiro (1985) show examples of the performance of several segmentation schemes of this type. The results display several faults typical of the approach, such as fragmented and disconnected regions and irregular boundary contours.

2.2.2 Split-and-merge schemes

Split-and-merge schemes (Horowitz and Pavlidis, 1976; Chen and Pavlidis, 1979, 1980) attempt to obtain homogeneous regions by an iterative process which splits regions which are not homogeneous, and merges adjacent regions if their union would be homogeneous. The process may start with the whole image being one block, or with the image split already into a number of square blocks. The splitting phase then recursively tests each square block for uniformity; if it is not uniform, the block is split into four square blocks of half the size, and each sub-block is then tested and split if necessary, until blocks of a given minimum size are reached. It is most convenient if the size of the blocks is always a power of 2. This technique is often implemented using a quadtree data structure for the image. After the splitting phase, the regions are generally broken up by the artificially imposed square grid. The merging phase attempts to overcome this problem by merging adjacent blocks which are sufficiently similar that the union of the blocks passes the homogeneity test. (In the studies cited above, the term 'merging' is used when merging the four sub-blocks of a larger block, and the term 'grouping' is used when merging adjacent blocks that have different parents in the quadtree. In both cases the operation is essentially the same.)

Chen and Pavlidis (1980) use a slightly different approach within the same framework. Their procedure starts with an initial segmentation, and splits blocks which are not homogeneous, and merges groups of four sub-blocks where the parent block is homogeneous. At this point, a cluster analysis is performed on the characteristics

of the blocks so obtained, and regions are identified on the basis of the clusters. The 'grouping' phase then consists of classifying each block according to which region it belongs to.

Two main problems occur with split-and-merge schemes, both related to the problem of accurately locating the boundary. First, the boundaries between regions tend to be rather 'blocky', with segments that follow the outlines of large patches which were not split; for example, if the boundary were to just cut off the corner of a block, then the block could well still pass the uniformity test. The corner of the block would then intrude into the shape of the boundary. Figure 3.3 on page 48 illustrates this deficiency.

Second, there is the 'small region problem' (Chen and Pavlidis, 1980), which is that the hypothesis tests used become less reliable as the blocks get smaller. This causes uncertainty about the boundary location, and can result in many small regions near the boundary which cannot be merged. Some studies have used ad-hoc techniques to eliminate these small regions, typically by forcing the small regions to merge with one or other of the major regions adjacent to the boundary.

The split-and-merge technique is suitable for use on textures, if an appropriate homogeneity test is used. Because the tests for homogeneity are performed on relatively large blocks of the image, a homogeneous textured region can be recognized as such. On coarse textures, it is necessary to place a lower limit on the sizes of the blocks.

2.3 Texture segmentation

The previous sections have discussed segmentation techniques which require at least a difference in grey-level distribution between regions, if not a difference in mean grey-level. However, adjacent textured regions may have the same pixel grey-level distributions, but nevertheless appear quite distinct to a human observer. Discriminating such regions generally requires the application of some kind of texture analyzer or filter, which takes into account the spatial arrangement of the pixels. Section 2.3.1 discusses a range of methods of textural analysis. The textured image can then be segmented on the basis of differences in the output of one or more textural analyzers. Section 2.3.2 discusses a number of studies in which this has been done.

2.3.1 Texture analysis methods

For use in a texture segmentation algorithm, a texture analyzer must must be applicable to a wide variety of textures, and it must be able to be applied to small windows of the image without prior identification of areas of homogeneous texture. For both of these reasons, structural approaches to texture analysis are not suitable, and are not discussed here. Statistical methods are generally quite suitable, and several representative methods are discussed below. For comprehensive reviews of both statistical and structural approaches, see Haralick (1979), Wechsler (1980), and Van Gool *et al.* (1985).

Most statistical methods which have been used in segmentation fall into one of the following classes:

1. Second-order statistics (i.e., statistics of pairs of pixels).
2. Filter methods.
3. Methods based on statistical texture models.

2.3.1.1 Second-order statistics

Second-order methods take account of the spatial dependence between pixels by considering the joint properties of pairs of pixels separated by small distances. Such methods are sometimes referred to under the general heading of the 'Spatial Grey-Level Dependence Method'. They involve calculating the second-order statistics of a region, which are based on the joint probability distribution of pairs of pixels separated by a displacement vector **d**, for a range of values of **d**. The joint probability distribution of a pair of pixels is simply a function which gives the probability that the first pixel will have grey-level g_1 and that the second will have grey-level g_2, for any given pair of grey-levels g_1 and g_2. The image is quantized to a limited number of grey levels N_G. In practice, relative frequencies are calculated and used as estimates of the underlying probabilities (which are unobservable). The second-order statistics are usually expressed as a set of co-occurrence matrices, one for each different displacement **d**. Each co-occurrence matrix is of size N_G^2, and expresses the relative frequencies for all values of g_1 and g_2.

Even when N_G is small (usually no more than 8), and only small displacements are used, the second-order statistics still represent a very large number of parameters. For example, with 8 grey-levels, each co-occurrence matrix contains 64 numbers. For

this reason, most methods based on second-order statistics reduce this to manageable proportions by one means or another. Usually, a number of measures are defined on each matrix, and these measures are used for classification. These measures are functions of the entries in the co-occurrence matrix. Measures used include energy, entropy, correlation, inertia, contrast, cluster shade, cluster prominence, and local homogeneity (see Haralick (1979) and Conners *et al.* (1984) for definitions). Generally only a small set of displacements is used. Some researchers combine the matrices for all orientations, producing an orientation-independent matrix (e.g., Chen and Pavlidis (1979)). In contrast, Gagalowicz and Ma (1985) used all displacements within a 25×25 window centred on each pixel in analyzing and synthesizing textures.

2.3.1.2 Filter methods

These methods involve the convolution of the texture image with a set of convolution masks, and the extraction of a set of statistics from the filtered images.

As described in Pietikäinen *et al.* (1982), Laws used a set of 3×3 or 5×5 masks, and calculated statistical measures over large windows. The most useful statistics were the sums of the squared or absolute values of the pixels of the filtered images. Laws uses the term 'texture energy measures' for these statistics. The masks are separable (i.e., the outer product of two one-dimensional masks), therefore the method can be implemented economically. The one-dimensional masks are derived from elementary edge and spot detecting masks. The four most important 5×5 masks for Laws' data set were those sensitive to a horizontal edge, a high frequency spot, a V-shape, and a vertical line. Pietikäinen *et al.* found that the power of the method depends on the general form of the masks rather than the specific numeric values employed, and that the local maxima of the filtered images were most important.

Wermser and Liedtke (1982) used a crude model of the human visual system to implement a texture analysis system. Their method derives 14 features from the image. The first two are the mean and variance of the grey levels in the image after it has passed through a logarithmic function and a bandpass filter. The other 12 features are the mean squared outputs of 12 directional filters—four directions (horizontal, vertical, 45° and 135° diagonals) at three different scales. The inputs of the filters are three binary images, one for each size of directional filter, derived from the original image using a process that Wermser and Liedtke call 'median clipping,' in which each pixel is

compared to the median of the pixels in a square neighbourhood. The corresponding pixel in the output image is set to 1 if the input pixel is greater than the median, otherwise to 0. This process is intended to model the contour formation which is said to occur in early visual processing. In spite of its shortcomings, their method represents an interesting attempt to emulate human texture discrimination capability.

Clark and Bovik (1986) also use a model of the human visual system to analyse textures. They use filters described by Gabor functions, which are Gaussian-modulated sinusoids. These functions provide a good model of the spatial response characteristics of the 'simple' cells in the mammalian visual cortex, which perform the first steps in the processing of the images relayed from the retinas (Marčelja, 1980). Turner (1986) also used a range of Gabor function filters to perform texture discrimination.

The 'texton' theory of human perception of textures proposed by Julesz and Bergen (1983) has much in common with these analysis methods. Their theory proposes that textures are discriminated on the basis of differences in density of textons. Textons are patterns of specific shapes, such as elongated blobs of various sizes, orientations, and colours, and the terminations of line segments. Caelli (1985) showed that textures similar to those used by Julesz could be discriminated using the edge detectors of Frei and Chen (1977) as texture filters.

2.3.1.3 Texture model-based methods

Model-based methods assume some generating process for the texture, and estimate the parameters of the assumed process for a given texture. These parameters are then used to describe the texture, and can also be used in synthesizing a new texture, which is hopefully equivalent in the sense that it is not discriminable from the original. These methods generally seem to produce results as good as or better than methods based on second-order statistics.

One approach is to use a linear model, such as an autoregressive or autoregressive/ moving average model. Kashyap and Khotanzad (1984) use an autoregressive model. Such models express a pixel's intensity as a linear function of the neighbouring pixels' intensities, plus a random component. Problems arise with representing textures containing large elements, since practical considerations limit the order of the model. Some authors have used a one-dimensional model rather than a two-dimensional model, which

increases the order which is practicable, but is obviously not as powerful at expressing spatial constraints.

A more general approach is to model the texture as a Markov random field, which allows an arbitrary dependence between nearby pixels. Markov random fields are discussed in detail in section 2.4.1.

2.3.2 Examples of texture segmentation algorithms

Several examples of texture segmentation algorithms are discussed below. These combine one of the textural analysis methods described above with a segmentation strategy, such as edge detection, window classification, or split-and-merge.

Wermser (1984) and Thompson (1977) have used edge-based methods to segment textures. The textural analyzer used by Wermser is that of Wermser and Liedtke (1982), which is discussed above. It produces a 14-component vector image (i.e., each pixel is a 14-dimensional vector). These components include local mean and variance, and filtered images at three scales and four orientations. The gradient at each point is determined in four directions (0°, 45°, 90°, and 135°), by evaluating the 'dissimilarity' of adjacent patches. The dissimilarity measure is either a ratio of inter- to intra-class distance, or the Euclidean distance between the average vectors for the two patches. At each point, the largest gradient for the four directions is taken, giving a gradient image. Thresholding and line-thinning are used to obtain boundary segments. Examination of the published results indicates that the boundary is detected reasonably well (presumably the threshold is set manually, since no automatic procedure is described), but the reported location is not accurate.

Thompson (1977) used a 'textural distance function,' described in Zobrist and Thompson (1975), which quantifies the degree of dissimilarity between textured image regions. The image is divided into square blocks, and a Roberts-type gradient and a local edge direction are obtained by evaluating the dissimilarity between diagonally-adjacent blocks. Edge points are then located by looking for 'ridge points' in the gradient image.

Edge-based methods such as these have the advantage of not requiring any preliminary information about the characteristics of the textures in the image. They still have the problem of deciding whether two textured regions are in fact different, to avoid

reporting false edges between two patches which are slightly different samples of the one texture. Wermser (1984) uses a threshold to make this discrimination, but does not indicate how the threshold is chosen.

Window-classification techniques have been used by Kashyap and Khotanzad (1984) and Conners *et al.* (1984). Kashyap and Khotanzad analyzed the texture in small windows by estimating parameters for two random field models, namely, a simultaneous autoregressive and a circular autoregressive model. These models express the value of each pixel as a weighted sum of a given set of neighbours plus a random term. A cluster analysis is performed to identify the distinct textures, and each window is then independently classified. A clean-up stage is used to remove some of the misclassification errors, but even so, the results shown have quite 'blocky', inaccurate boundaries.

Conners *et al.* (1984) used co-occurrence matrices to classify small windows of a high-resolution aerial image, according to whether they contain one of a number of pre-specified classes of texture, or a mixture. They propose a split-type algorithm for subdividing the image until each block contains only one texture.

The split-and-merge algorithm has been employed by Chen and Pavlidis (1979) using co-occurrence matrices to analyze the textures. A block was defined to be homogeneous if the difference between the co-occurrence matrices of its four sub-blocks was less than an arbitrary threshold. For each sub-block, the co-occurrence matrices for four orientations were combined into one by adding them together, producing an orientation-independent matrix.

Caelli (1985) used a relaxation-type algorithm called 'impletion' in segmenting textures on the basis of differences in the outputs from several texture filters. The relaxation algorithm removes the textured nature of the filter outputs by an iterative process which smooths the image and uses a non-linear transformation to increase high values and decrease low values. The filters used involved convolution of the image with the 3×3 masks suggested by Frei and Chen (1977) in the context of edge detection.

Pietikäinen and Rosenfeld (1981) used pyramid node linking to segment textures. This is a method which is related to split-and-merge algorithms, in that it uses a pyramid of successively reduced-resolution images. However, the blocks at each level overlap. Links are established between levels with an iterative procedure. These links associate the blocks at the finest resolution level into regions. They used the 'contrast'

statistic obtained from co-occurrence matrices to analyze the textures.

Davis and Mitiche (1982) present a relaxation-type algorithm for texture segmentation, called MITES, which iteratively updates the value of each pixel in an image with the average of a certain set of neighbours. The set of neighbours used depends on which neighbours already have similar values to the pixel, and on the current interpretation of the pixel as being an edge or interior point. The MITES algorithm can either use pre-specified statistical descriptions of the regions, or obtain this information by cluster analysis, and update the clusters as processing proceeds. Statistics used for texture analysis were a 'contrast' statistic (defined in terms of the histogram of differences between pixels for a given set of displacements), and a set of statistics defined on a polar plot of the number of edges as a function of orientation.

Yachida *et al.* (1979) present an algorithm for locating the boundaries of textured regions, which optimizes a figure of merit for a piece-wise linear boundary. The figure of merit has two components: a term which penalizes corners, which depends on the angle between adjacent segments; and a term which is large when patches adjacent to each segment have different textural properties. The textural properties used were grey-level distribution, edges per unit area, and edge direction. The boundary is located by a sequential search procedure within a plan obtained from an initial segmentation, performed by a recursive thresholding procedure (Tomita and Tsuji, 1977). The study of Yachida *et al.* is interesting, because it has several similarities with the method presented in this thesis: it addresses the detection and location problems separately, it uses an optimization procedure to find the best boundary, and the boundary is represented parametrically.

2.4 Statistical segmentation

Whereas most traditional segmentation techniques view the sizes and shapes of the regions as being determined by the sizes and shapes of the objects being viewed, a class of segmentation methods which has been studied recently views the region shapes as stochastic. The problem of segmentation then becomes a problem of statistical estimation. Techniques such as maximum likelihood estimation are therefore appropriate within this model. Given that the objects portrayed in the image are not known *a priori*, it is perhaps reasonable to attempt to obtain a statistical model for the sizes

and shapes of regions likely to be encountered.

Within this model, the image is seen as having been generated by an underlying two-part random process. The first part, called the 'region process', is responsible for dividing the image into disjoint regions in a stochastic manner, thus generating the unobservable correct segmented image. The second part, called the 'texture process', fills in each region of the image with a different texture.

The task of a segmentation algorithm is, then, to estimate the segmented image, and possibly also the parameters of the texture process, that is, the characteristics of the texture in each region. Let X be the segmented image, and Y the observed image. Apart from Y, there are two other pieces of information essential for estimating X:

(a) knowledge (discovered or assumed) about the region process, in the form of an *a priori* probability distribution $P(X)$ for the segmented image,

(b) knowledge about the texture process, in terms of the probability distribution $P(Y|X)$ for the observed image, given the segmented image.

The segmented image which is chosen is that which is most likely, given the observed image; that is, the one which has the maximum *a posteriori* probability. These are therefore called MAP (Maximum *A Posteriori*) estimation schemes. The *a posteriori* probability (i.e., the probability after the image has been seen) is, by Bayes' rule,

$$P(X|Y) = P(Y|X)P(X)/P(Y)$$

The factor $P(Y)$ does not depend on X, and therefore does not enter the optimization. Segmenting the image therefore becomes a matter of finding the X which maximizes $P(Y|X) \cdot P(X)$.

Clearly, then, the choices of $P(X)$ and $P(Y|X)$ are of critical importance in the performance of MAP segmentation schemes. The factor $P(Y|X)$ is the likelihood of the image given a proposed segmentation X. It expresses the degree to which the area of the image assigned to each region resembles the texture which is supposed to fill that region. Thus, a statistical characterization of the textures in the regions is required for computing this factor. All the studies cited below assume that the required characterization is specified *a priori*, which simplifies greatly the calculation of $P(Y|X)$; these schemes therefore address the location problem, not the detection problem.

Two alternatives exist in characterizing the textures. The simpler is to regard the pixels as independent. Under this assumption, each texture process is specified by the distribution of pixel intensities in the texture. The assumption of independence is possibly close to the truth if the intensity fluctuations in the regions are due to noise in the digitization or transmission of the image. If the fluctuations are due to texture or fine detail, the assumption is generally not valid, especially at high resolutions. If the pixels are regarded as independent, the segmentation algorithm will only be able to discriminate regions which differ in their grey-level distributions. The other alternative is to characterize the texture using a Markov random field (MRF). MRFs form a class of statistical models which explicitly take into account the interdependence of pixels.

Whereas the factor $P(Y|X)$ represents the influence that the observed image has on obtaining the segmented image, $P(X)$ represents *a priori* knowledge about the types of region shapes which are desirable. It assigns a higher probability to segmentations in which the region shapes have such properties as connectedness, smoothness, etc. As $P(X)$ represents information about the correct segmentations of the class of images to be considered, it is practically impossible to measure, and is usually specified arbitrarily or by experiment. Another difficulty lies in finding a statistical model for X (or a class of models) which is simple enough to be practical, yet powerful enough to express a preference for simple boundary shapes. MRF models form one possible class which has been extensively investigated in recent years.

The formulation of boundary finding as a MAP estimation problem appears to have been first suggested by Cooper and Elliott (1978). They also made the point that the original image is more useful for estimating the locations of boundaries than is the output of an edge detector. This study and a series of following studies used an assumption of independent pixels, and characterized the boundary as a causal Markov chain of edge elements (Cooper *et al.*, 1980; Elliott *et al.*, 1982). The transition probabilities for each edge element were a function of the previous 7 edge elements, and were set so as to favour straight chains. Two methods were developed to maximize the *a posteriori* probability. The first was the 'ripple filter' (Cooper *et al.*, 1980), which started off with a closed initial boundary estimate, and iteratively made local adjustments to increase the likelihood until a (local) maximum was reached. The main problem with this approach is that the decision about whether to move a particular segment of the boundary is made only on the basis of a few pixels nearby, so the boundary can tend

to get stuck in situations where isolated atypical pixels stop it from moving to a global maximum. The second method was a sequential boundary finder (Elliott *et al.*, 1982), using a heuristic search algorithm to track around the boundary of the region, guided by information in a swath adjacent to the boundary. (This does not guarantee a closed boundary.) The main failing with these approaches was that the boundary model was not powerful enough. These approaches are mentioned here because they bear some similarity to the approach developed in chapter 3.

2.4.1 Introduction to Markov random fields

Markov random field theory (Besag, 1974) is a general way to model a set of random variables which are mutually interdependent, such as the pixels in a textured image. Each random variable in an MRF has a set of neighbours, which are defined simply as those other random variables upon which it depends. In its full generality, the theory allows each random variable to have any other random variables as neighbours, but we consider here only a restricted class of MRFs, where

- the random variables are pixels, laid out on a square grid;
- the neighbours of a pixel are those within a given distance, called the 'interaction distance', and
- the field is homogeneous—the conditional distribution of each pixel, given its neighbours, is identical for each pixel.

Such homogeneous MRFs should be sufficient to describe most textures, provided that the interaction distance is large enough. In practice, the neighbours are usually limited to the nearest 4 or 8 pixels, by considerations of computational complexity.

MRF theory is formulated in terms of the conditional probability distribution for each pixel; that is, the distribution given the values of all neighbouring pixels. For a given realization x of a random field X, one cannot obtain the joint probability for the whole field by multiplying together the conditional probabilities for each pixel. The Hammersley-Clifford theorem (Besag, 1974) states that under the 'positivity' condition[1], an MRF is equivalent to a Gibbs random field (GRF), which is described in terms of the total joint probability of a realization. The MAP segmentation rule given

[1] i.e., that every realization of the field has a non-zero probability

above requires the maximization of the joint probability, which is expressed explicitly in the GRF formulation but not in the MRF formulation. The joint probability is

$$P(X = x) = \frac{1}{Z} \exp(-\sum_{c \in C} V_c(x)) \tag{2.3}$$

where c is a clique—a set of pixels which are all neighbours of each other, C is the set of all possible cliques in the image, $V_c(x)$ is independent of the pixels outside c, and Z is the 'partition function':

$$Z = \sum_{x} \exp(-\sum_{c \in C} V_c(x)) \tag{2.4}$$

That is, Z is a normalizing factor so that the probabilities sum to 1. See Derin and Cole (1986) for a fuller exposition.

The partition function is virtually impossible to calculate in general, except for very small fields, because it involves a sum over all possible realizations of the field. For example, even for a field of 64×64 pixels, quantized to 16 levels, there are $2^{16384} \approx 10^{5000}$ realizations—an astronomical number. However, since Z does not depend on x, it is possible to compare the likelihood of two realizations without knowing Z.

Besag (1974) introduced the class of 'auto-models', which is a subclass of MRFs, for which the functions $V_c(x)$ are non-zero only for those cliques containing one or two pixels; for those containing two pixels, $V_c(x)$ is of the form $\beta_{ij} x_i x_j$. Auto-models are considerably simpler to use than general MRFs, since the number of 2-pixel cliques rises only as the square of the interaction distance, whereas the total number of cliques rises exponentially with the interaction distance. It is possible to write Z for an auto-model in terms of the determinant of a matrix—see Cohen and Cooper (1987). Examples of auto-models which have been used in image segmentation include the auto-logistic, auto-normal, and auto-binomial models.

2.4.2 Use of MRF models in image segmentation

Markov random fields have been used to model both the shape of the regions and the textures they contain. When used to model the shape of the regions, the parameters are set to assign higher probabilities to large connected regions than to small fragmented ones, to encourage larger regions where possible. Here, it is not necessary to calculate the partition function, since relative likelihoods are being compared over the same size and shape of field. Geman and Geman (1984) employ an interesting variant: they use

an MRF to model the interdependence of the pixels of the segmented image with each other and with the presence or absence of edge elements between pairs of pixels.

A few studies have employed MRF models to analyze and describe textures (e.g., Hassner and Sklansky, 1980; Cross and Jain, 1983). However, the practical impossibility of calculating the partition function limits their usefulness in describing texture within segmentation algorithms. Using MRFs to model the textures inside different regions involves the problem of calculating the partition function, since different sizes and shapes for each region must be compared. For this reason, only one study to date (Cohen and Cooper, 1987) uses an MRF characterization to calculate the joint likelihood of all the pixels in a region. In this case, the methods used are restricted to auto-models, and some approximations are used in the case of non-square regions. Derin and Elliott (1987) modelled the texture processes as MRFs, but then used these to calculate the likelihoods of 3×3 regions belonging to each of the texture classes. These likelihoods were assigned to the centre pixel, and multiplied together for all pixels inside one region. (This does not give the true joint likelihood for the region; only an approximation.)

Searching for the MAP segmentation of an image is not a trivial task; an exhaustive comparison of all possible segmentations is not practical, and some method which will go more directly to the desired solution is necessary. Two methods which have been employed are dynamic programming and stochastic relaxation.

Dynamic programming has been used to maximize the *a posteriori* probability with an MRF region model by Derin and Elliott (1987). Dynamic programming reduces the computation required by taking advantage of the limited interaction between pixels. Even so, maximizing the likelihood over the whole image is still computationally intractable, and these studies therefore only maximize the likelihood over narrow overlapping strips of the image. This yields a solution which is sub-optimal, because the maximization cannot combine information over the whole image.

Stochastic relaxation (Geman and Geman, 1984) is a scheme which starts with an initial segmentation and updates it over a large number of iterations so that the segmentation approaches the MAP segmentation. Each iteration involves updating pixels stochastically, according to their local conditional probability distribution. Stochastic relaxation is also known as simulated annealing, since it involves a 'temperature' which is slowly decreased. The temperature controls how probable it is for a pixel to change to

a less probable state; at zero temperature, this probability becomes zero. This scheme never computes the total joint probability; rather, it relies on a theorem which states that, provided the temperature is decreased slowly enough, the algorithm is guaranteed to converge to the global maximum of the *a posteriori* probability in a stochastic sense (i.e., the probability that the solution is the global optimum approaches 1 as the number of iterations approaches infinity). Unfortunately, the number of iterations required by the theorem is impractically large (e.g., e^{40000} pixel updates (Geman and Geman, 1984)). Instead, the number of iterations is generally determined by inspection of the results. Typically 25–1000 iterations are used. The major disadvantage of this scheme is that it requires a large amount of computation.

All of the studies cited above assume that the parameters of the texture and region processes are known *a priori*. These algorithms therefore rely on a previous detection phase having estimated the characteristics of the texture processes in the different regions. In general, therefore, an initial coarse segmentation must be performed using some other method. Derin and Cole (1986) give an example of a segmentation where the parameters for one of the texture processes were inadvertently wrongly specified; the segmentation obtained was quite poor. Adaptive algorithms (those not requiring an *a priori* estimate of the texture processes) are an active area of research.

The parameters of the region process are usually set to values encouraging large connected regions. Derin and Cole (1986) note that there is a trade-off in setting the parameters, between discouraging single-pixel regions and being sensitive to fine details at the boundary.

2.4.3 Evaluation of MRF approaches

MRF theory is appealing, as it is a general way to model the statistical dependence between pixels in an image. However, the computational burden involved in using MRFs is large, and could be expected to increase exponentially with the interaction distance (since the number of clique classes does so), unless attention is restricted to auto-models.

Most studies which have employed MRFs have taken only the nearest 4 or 8 pixels as neighbours, for practical reasons. It is doubtful whether this is sufficient to model effectively the sort of region shapes that are simple on a global scale (even granted that a limited interaction distance can give rise to longer-range order). Those studies

that show results where the region shapes are simple (e.g., ellipses or rectangles), such as Geman and Geman (1984) and Derin and Cole (1986), generally show boundary contours that are quite noisy. That is, a first- or second-order MRF region model does not appear to be able to favour boundaries that are straight or smooth over a distance of tens of pixels. In addition, as the regions become harder to distinguish (signal to noise ratio decreases), the boundary becomes more noisy and therefore more complex in shape, whereas in fact less detail is really perceivable.

It is also doubtful whether practical neighbourhood sizes are sufficient to model natural textures over the range of scales encountered in real-world images. Cross and Jain (1983) give examples of textures synthesized from the parameters extracted from natural textures, for a third-order auto-binomial MRF model; the results are adequate for fine textures, but not for coarse textures. In any case, it appears to be impossible to use MRFs to model the texture in the regions, due to the impossibility of calculating the partition function, unless auto-models are used exclusively.

2.5 Conclusions

Of the approaches listed in this chapter, it is perhaps the edge-based approaches which hold the most promise of being able to determine accurate information about the locations of boundaries in textured images. Region-based approaches generally cannot provide accurate location information, because they must use windows of a reasonable size to be able to categorize the texture in the windows reliably. The size of these windows then limits the accuracy with which the boundary location can be determined. The statistical approaches using MRFs can potentially perform somewhat better, because individual pixels are classified, and the classification of a pixel is dependent on the classifications of its neighbours. However, the region models used in these schemes usually only incorporate dependence of a pixel on its nearest four or eight neighbours, for computational reasons, and this is insufficient to encourage boundaries which are globally simple.

Edge-based approaches are potentially capable of determining the location of a boundary to within half a pixel spacing. Of the edge-based approaches, it is those which can easily be used at a range of scales which are suitable for use on textures. Most edge detection schemes have been developed with relatively small support; of

those which could be used with large support, convolution edge detectors are most widely used, easiest to implement, and scale in an obvious manner. For these reasons, convolution edge detectors were chosen to be implemented in this study for evaluation (and subsequently comparison with the approach developed in chapter 3). However, as the results in figures 2.1 to 2.3 show, even convolution edge detectors do not give accurate boundary location information in textured images. Clearly, a new approach is necessary.

Chapter 3

A new approach to boundary location

In this chapter, attention is focussed on the problem of locating the boundary between textured regions which differ in their pixel grey-level distributions. Sections 3.1 and 3.2 first discuss in more detail the difficulties of the location problem and the deficiencies of conventional techniques, and then propose three criteria which capture essential characteristics of a good boundary description. These three criteria, **simplicity**, **consistency**, and **accuracy**, are formulated in mathematical terms, which enables the specification of an algorithm for finding a boundary description satisfying these criteria. This algorithm, called the 'BLM algorithm' (for Boundary Likelihood Maximization), is outlined in section 3.3, and its implementation is described in detail in section 3.4. The BLM algorithm solves the location problem, but not the detection problem, since it requires knowledge about the regions adjacent to a boundary. Therefore, a split-and-merge algorithm has been implemented to provide the initial statistical information required. (See section 5.3.2 for a discussion of the possibility of using the approach developed here to solve the detection problem as well.)

3.1 The boundary location problem

Ideally, the segmentation phase in a computer vision system should provide accurate, concise, and reliable descriptions of the boundaries between regions in an image, even when those regions are textured. However, all of the schemes reviewed in the previous chapter tend to produce rather inaccurate boundary descriptions on textured images; certainly, the descriptions fall short of the accuracy and precision of human perception of the boundaries. In other words, conventional algorithms may provide

reasonable solutions to the detection problem, but they do not adequately solve the location problem.

The previous chapter referred to a range of published results which demonstrate the inaccuracy of conventional algorithms, and showed some example results from the implementation of two convolution edge detectors on a textured image. These edge detectors were selected as being the most promising candidates for providing accurate boundary descriptions, as discussed in section 2.5. However, the results shown in figures 2.2 and 2.3 have serious deficiencies as descriptions of the boundary in figure 2.1. Assuming that the string of edge pixels which corresponds to the true boundary can be identified (which is not easy at the smaller scales), it is evident that at all scales the string deviates from the true boundary position, and has unnecessary and incorrect undulations. In contrast, the boundary perceived by human observers in figure 2.1 has a simple shape, and is accurately positioned with respect to the true boundary position. (The true position is known, as this is an artificially generated image.)

The main reason for these deficiencies is that the textured regions contain intensity fluctuations which are irrelevant to the boundary. These fluctuations can be locally just as large as the intensity change at the boundary, and indeed, there can be intervals along the boundary where the intensity change across it is locally very small or even of reversed polarity. Thus, it is not necessarily the pixels where the gradient of a smoothed version of the image is highest that give accurate information about the boundary location. The positions of these maxima are easily perturbed by intensity variations within the regions.

3.2 Criteria

The segmentation schemes discussed in chapter 2 have been evaluated informally with respect to their ability to give good descriptions of boundaries in textured images. Before we consider an algorithm for finding a good description for a boundary, we must first consider more precisely what is meant by a 'good' description. This is expressed below in the three criteria of accuracy, simplicity, and consistency. These criteria are expressed first in intuitive terms, and then formalized mathematically.

3.2.1 Accuracy

The first of the three criteria for a good boundary description is accuracy. Clearly, the results from convolution edge detectors are inaccurate, whereas human observers can accurately locate the boundary position. Inspection of figure 2.1 shows that the boundary position is perceptually constrained by the bright pixels which only appear in the right-hand region; they have a disproportionately powerful effect in determining where the boundary is perceived. Rather than points of maximum gradient in a smoothed image, it is therefore those pixels which are unlikely to belong to one of the regions which effectively constrain the boundary position.

Thus, accurate estimation of the location of the boundary requires some statistical knowledge about the regions. Using such statistical knowledge means that some pixels can be given much greater influence in positioning the boundary than others, according to the likelihood of them belonging to one region or the other. Since we are considering the location problem, we assume that an initial segmentation has been performed, and that the necessary statistical information can be collected.

The accuracy criterion obviously cannot be formalized in terms of the distance between the reported boundary and the true boundary. Instead, we require that all the relevant statistical information should be taken into account in determining the boundary location. In other words, the best available estimate of the boundary location should be used. The accuracy criterion is therefore formalized in terms of maximum-likelihood estimation:

> The boundary position should be that which maximizes the likelihood of the image, given the position of the boundary.

(This likelihood is referred to informally as the likelihood of the boundary.) Maximum-likelihood estimators are known to have generally good properties (Freund, 1972, p267), and have been shown to give accurate results in locating the edge point in one-dimensional data (Mazumdar *et al.*, 1985).

Likelihoods are computed in the BLM algorithm using the assumption that pixels are independent. This is a crucial simplifying assumption, which makes it possible to compute the log likelihood of a given boundary by summing, for each pixel, a function of its intensity. The independence assumption also means that regions must differ in their grey-level distributions to be discriminable. The alternative to the independence

assumption is to characterize the regions using Markov random fields. However, calculating the likelihoods would then require computing the partition function, which is virtually impossible. The results in chapter 4 show that good results can be obtained using the independence assumption, even when the pixels are correlated.

Maximum-likelihood estimation must be constrained in some manner if it is not to produce unnecessarily complex results. Under the independence assumption, unconstrained maximum-likelihood estimation becomes simply a pixel classification operation. With the constraint that there are only two regions, which are both connected, maximum-likelihood estimation of the boundary in figure 2.1 will produce something like that shown in figure 3.1. (This boundary was found by a procedure which is not guaranteed to find the global maximum of the likelihood, but has evidently come very close.)

3.2.2 Simplicity

The true boundary in figure 2.1 is actually a very simple shape, just two straight lines meeting at a corner. The reported boundary in figure 3.1 is far more complex than it needs to be (as are the results from the convolution edge detectors). The simple shape is sufficient to adequately represent the image detail, and is clearly preferable to the more complex shape shown in figure 3.1. Therefore, the second criterion is simplicity:

> The boundary description should be no more complex than is necessary.

A simple boundary description has many advantages over an unnecessarily complex one. It has practical advantages in reducing the computational burden of processes which use the boundary description. There is also a more fundamental advantage: reporting an unnecessarily complex boundary description can imply that more information has been extracted from the image than is statistically justifiable.

Quantifying this criterion requires a measure of complexity. An obvious method is to measure the complexity of a boundary by the number of bits needed to specify its shape. This will depend on the representation used. Parametric representations, such as piece-wise linear or cubic spline representations, will generally require fewer bits to specify a given shape than either chain-coded or image-based representations. Boundaries are therefore represented in the BLM algorithm in parametric form as a

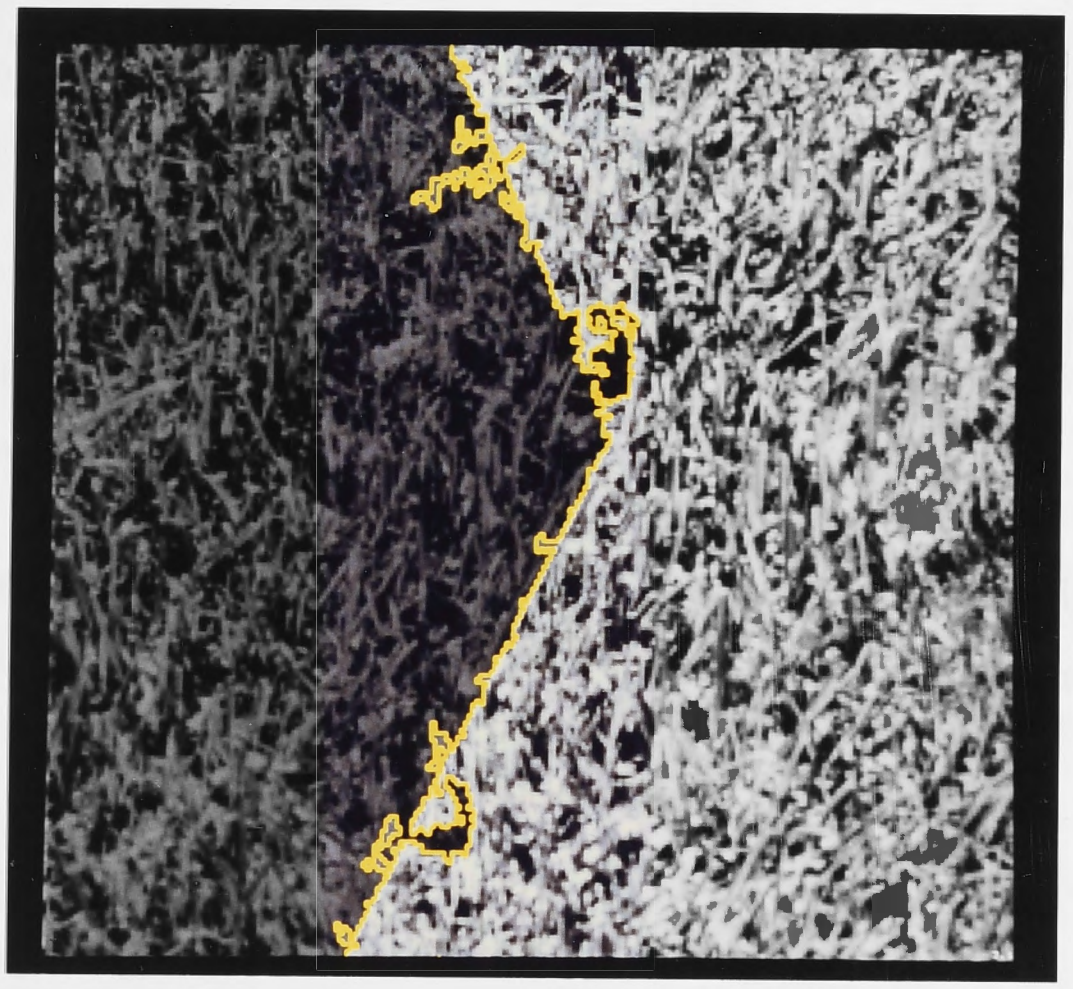


Figure 3.1: Maximum-likelihood boundary

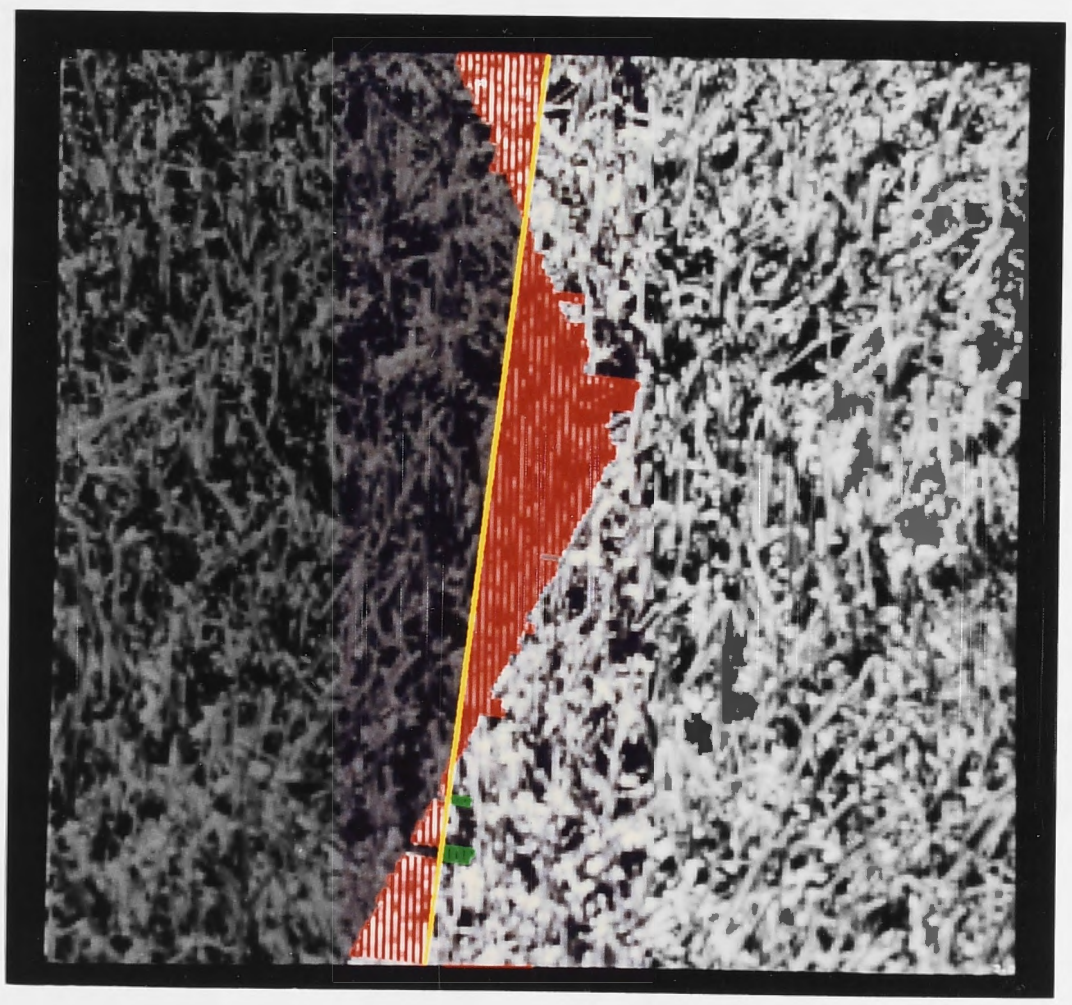


Figure 3.2: Inconsistent boundary

set of linked segments, each either a straight line or a cubic polynomial curve. The endpoints of the segments are called knots. A representation such as this has a number of important advantages:

1. The elements of such a description—straight lines, smooth curves, and corners—correspond directly to the kinds of boundary shapes which appear perceptually simple.

2. A parametric form should be more suitable for subsequent shape analysis than either a chain-coded representation or a segmented image.

3. A parametric representation can easily be converted into other forms, such as a chain-coded string or a segmented image, but the reverse transformations are difficult.

4. A preference for smooth or straight boundary shapes is maintained without excluding the possibility of corners.

5. There is a preference for shapes which are globally straight or smooth, since the possible length of a segment is limited only by the image size.

The complexity of a particular parametric representation is taken as being proportional to the number of parameters (real numbers) used. Strictly speaking, each parameter should be represented to whatever accuracy is necessary, and require a corresponding number of bits. In the current implementation, however, the same floating-point representation is used for all parameters, so the number of bits used is simply proportional to the number of parameters required.

3.2.3 Consistency

The simplicity and accuracy criteria capture certain desirable aspects of a boundary description, but they are not sufficient to ensure that the description corresponds closely to what is actually in the image. If the description is too simple, it will not have sufficient degrees of freedom to follow all the detail perceivable in the image. The simplicity criterion is therefore subject to a 'consistency' criterion:

> There must be no patches of the image adjacent to the boundary which are clearly on the wrong side of the boundary.

Figure 3.2 (page 43) gives an example of a boundary description which is not consistent with the image data. It is inconsistent because the light patches to the left of the boundary at the top and bottom should quite obviously be on the other side, as should the dark patch to the right of the middle section of the boundary. These inconsistent areas are shown in red.

Testing whether a boundary is consistent with the image data or not requires some way to find candidate patches to test, and an appropriate test to determine whether or not they can belong in the region to which they have been assigned. If it can be said, with confidence, that some particular patch cannot belong, then the boundary fails the consistency test. On the other hand, there will usually be patches which are more likely to belong on the other side of the boundary, but which can belong in the region to which they have been assigned. Such patches do not cause the boundary to fail the consistency test.

Suppose that a patch P has been selected for testing, and is currently assigned to region A. What is needed is a statistical test for testing whether the patch can reasonably belong to region A, that is, whether the pixels in P can reasonably be expected to have been generated according to the statistical model for the texture in region A. Thus, we frame a null hypothesis H_0 that the patch P belongs to region A, with the alternative hypothesis H_A that the patch belongs to region B on the other side of the boundary. A likelihood-ratio test (Freund, 1972, p304) of H_0 versus H_A involves comparing the ratio of their likelihoods to an appropriate threshold; H_0 is rejected if

$$\frac{L_0}{L_A} < K$$

where L_0 and L_A are the likelihoods of H_0 and H_A respectively.

Such a test may sometimes make an incorrect decision. These errors are divided into two classes: type I errors, or 'false alarms', when H_0 is true, but is rejected, and type II errors, when H_0 is not rejected although it is false. The probabilities of both types of errors are determined by K. Within the context of the consistency test, a type I error means that the boundary will be rejected, even though it may be correct. This type of error is rather serious, because it will result in a boundary that is more complex than necessary, with the patch incorrectly assigned to region B. Therefore, it is necessary to set K low enough that the probability of false alarms is virtually zero. This probability is sometimes referred to as the 'level of significance' of the test.

Type II errors, on the other hand, correspond to situations where the boundary is incorrect, but the test is not sensitive enough to detect the fact. Increasing K will increase the sensitivity of the test, but will also increase the false-alarm probability. Clearly, we want the most sensitive test with an acceptably low false-alarm probability. The chosen value of K is thus ideally that which will give the highest acceptable false-alarm probability. Section 3.4.5 gives details of the method used for choosing K.

The consistency criterion requires that the boundary description be a good approximation to the true boundary, which is in some ways a consideration of accuracy. However, it is different from the accuracy criterion, because the accuracy criterion is subject to the simplicity criterion. The accuracy criterion requires that the best possible estimate *at the given level of complexity* be used. In contrast, the consistency criterion requires that the boundary be complex enough to represent accurately the observable detail in the boundary. It is thus concerned with how much detail in the boundary description is statistically justifiable. It is not possible to determine, from the likelihood of a description, whether it is complex enough to be consistent, or whether it is unnecessarily complex. The accuracy and consistency criteria therefore have complementary roles.

3.3 Overview of boundary locator algorithm

Given the three criteria discussed above, an automatic procedure is required for finding a boundary description satisfying them. The consistency criterion requires that any proposed boundary description be tested for consistency, and the accuracy criterion specifies that only the most likely description at any given level of complexity needs to be tested. Given the preference for simple descriptions embodied in the simplicity criterion, it seems reasonable to start with a very simple description and elaborate it as necessary until it is consistent, while maximizing its likelihood at each stage. The boundary description at each stage is referred to as a 'boundary hypothesis', because it represents a tentative statement about the boundary, which is tested and then either accepted or rejected. This approach suggests the following procedure, which is an outline of the BLM algorithm:

1. **Initialize:** Obtain statistics of regions and form a simple initial hypothesis.

2. **Optimize:** Adjust the current boundary hypothesis so as to maximize its likelihood.

3. **Test:** Check whether the boundary is consistent with the image.

4. **Elaborate:** If the boundary is not consistent, elaborate it (using the inconsistencies as a guide) and return to step 2.

5. **Simplify:** Once a consistent boundary has been achieved, check whether the boundary can be simplified while still maintaining consistency.

Figures 3.3–3.7 illustrate the operation of the BLM algorithm on the image shown in figure 2.1. A coarse initial segmentation is obtained using the split-and-merge procedure described in section 3.4.7, giving the result shown in yellow in figure 3.3. From this initial segmentation, two areas are defined for evaluating the statistics of the two regions. These two areas are shown outlined in white in that figure. The initial boundary hypothesis is the rectangle in figure 3.4. This is simply the bounding rectangle of the left-hand area for evaluating statistics.

Optimizing the initial boundary hypothesis yields the result shown in figure 3.5. This boundary is not consistent with the image data; the patches shown with red cross-hatching are patches which fail the consistency test. (The procedure which finds these patches only goes out a limited distance from the segment, for efficiency reasons.) The hypothesis must be made more complex; extra knots are added at the midpoints of the inconsistent segments. Returning to the optimization phase yields the result shown in figure 3.6. This result is consistent, but has two more segments than are necessary. The simplification step removes the two superfluous knots, yielding the correct result shown in figure 3.7.

3.4 Implementation of BLM algorithm

The essential problem of locating boundaries can be studied using the simplest possible configuration: two regions, separated by a single boundary. The BLM algorithm has been implemented for this configuration, with the assumptions that the regions are connected and homogeneously textured, and differ in their pixel intensity distribu-

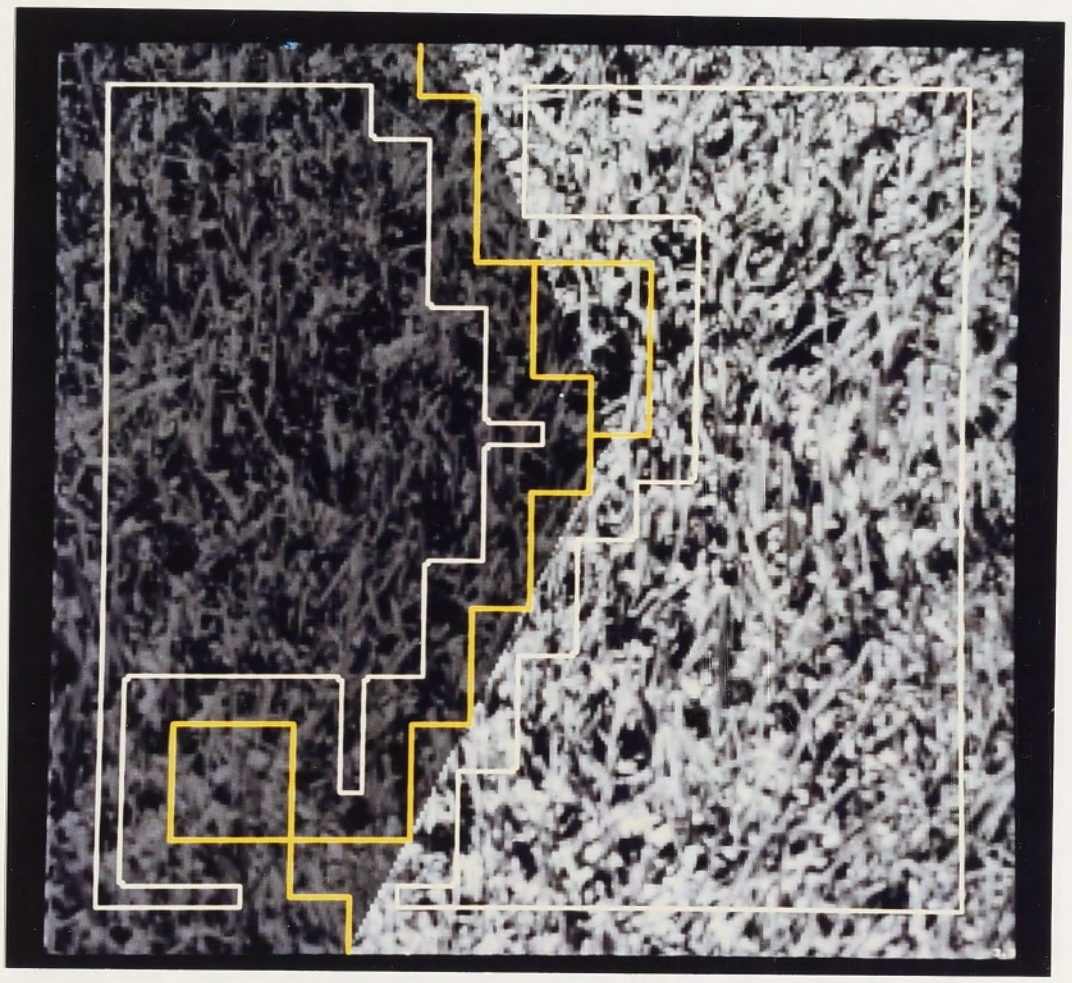


Figure 3.3: Results from split-and-merge procedure

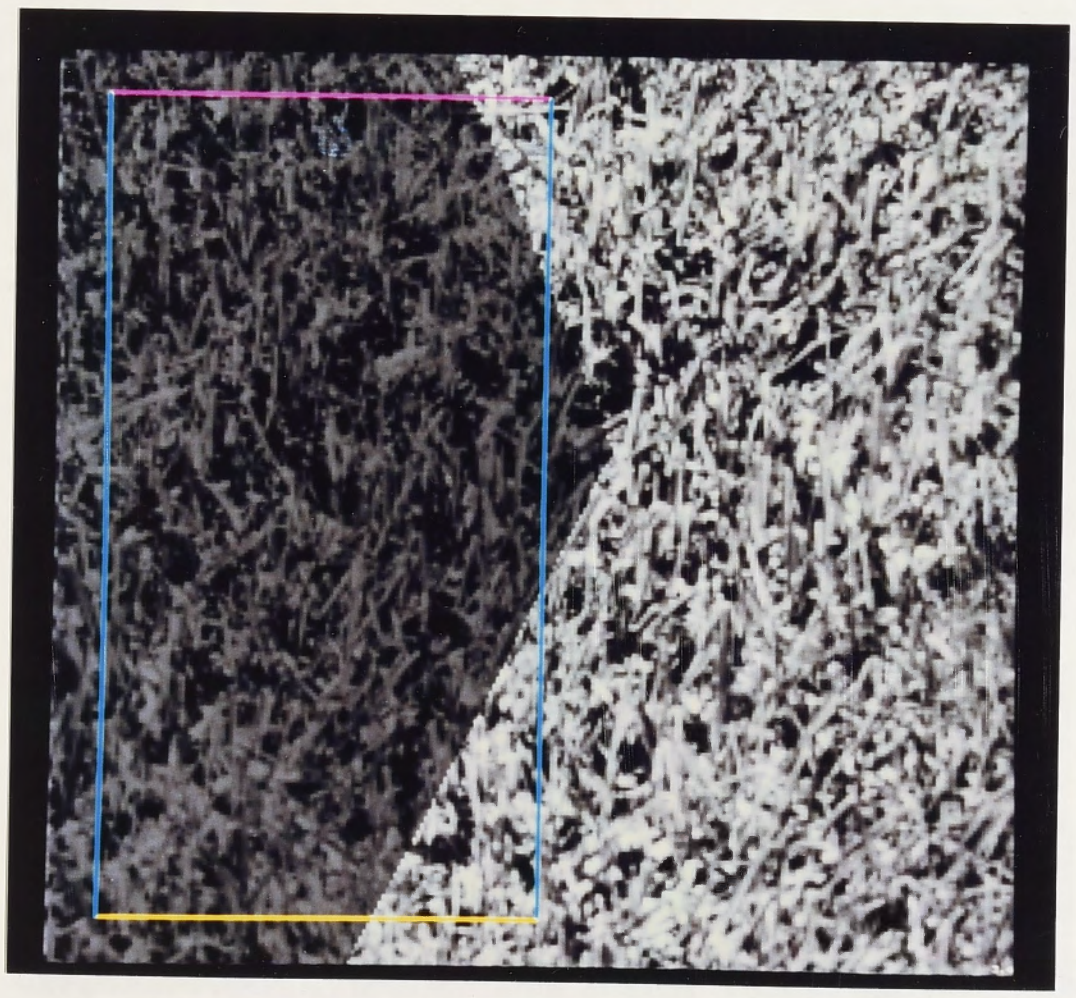


Figure 3.4: Initial boundary hypothesis

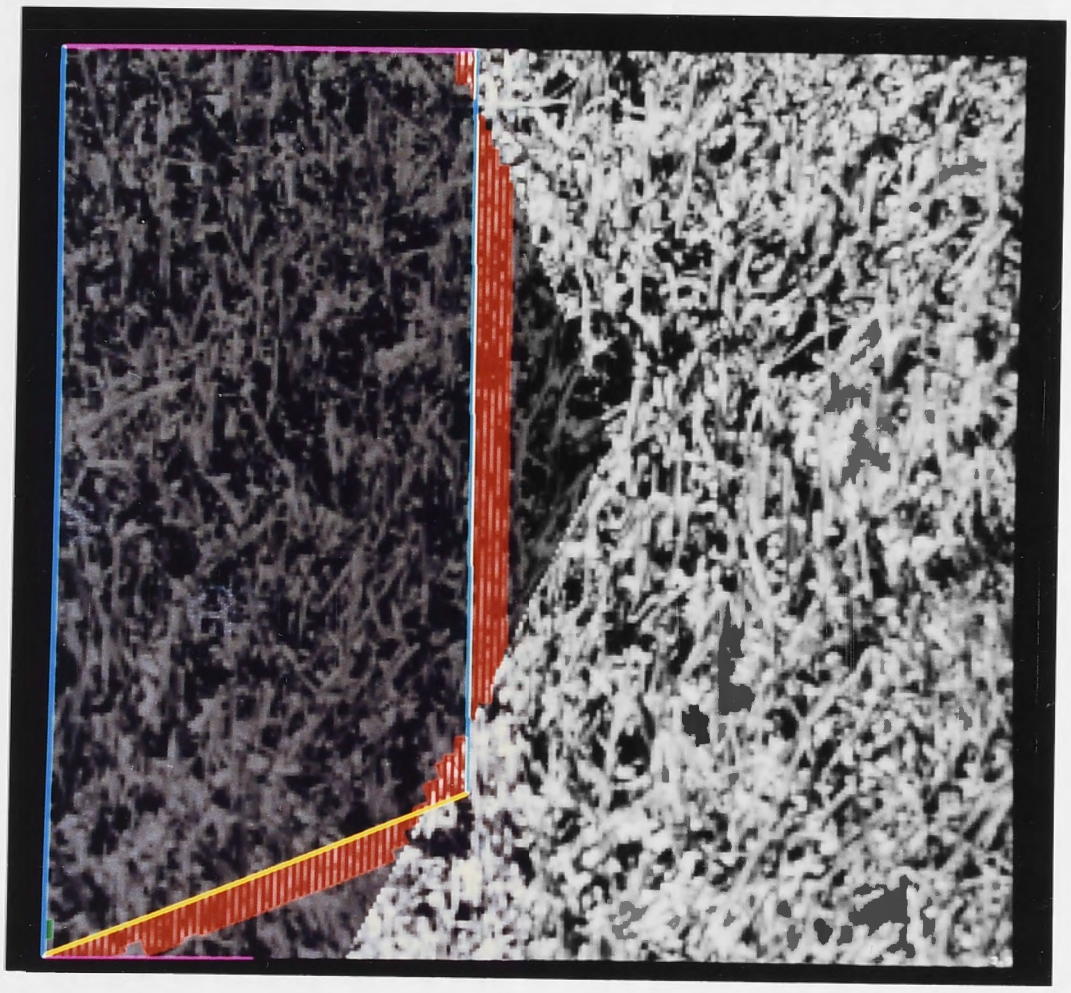


Figure 3.5: Results from optimizing initial boundary hypothesis

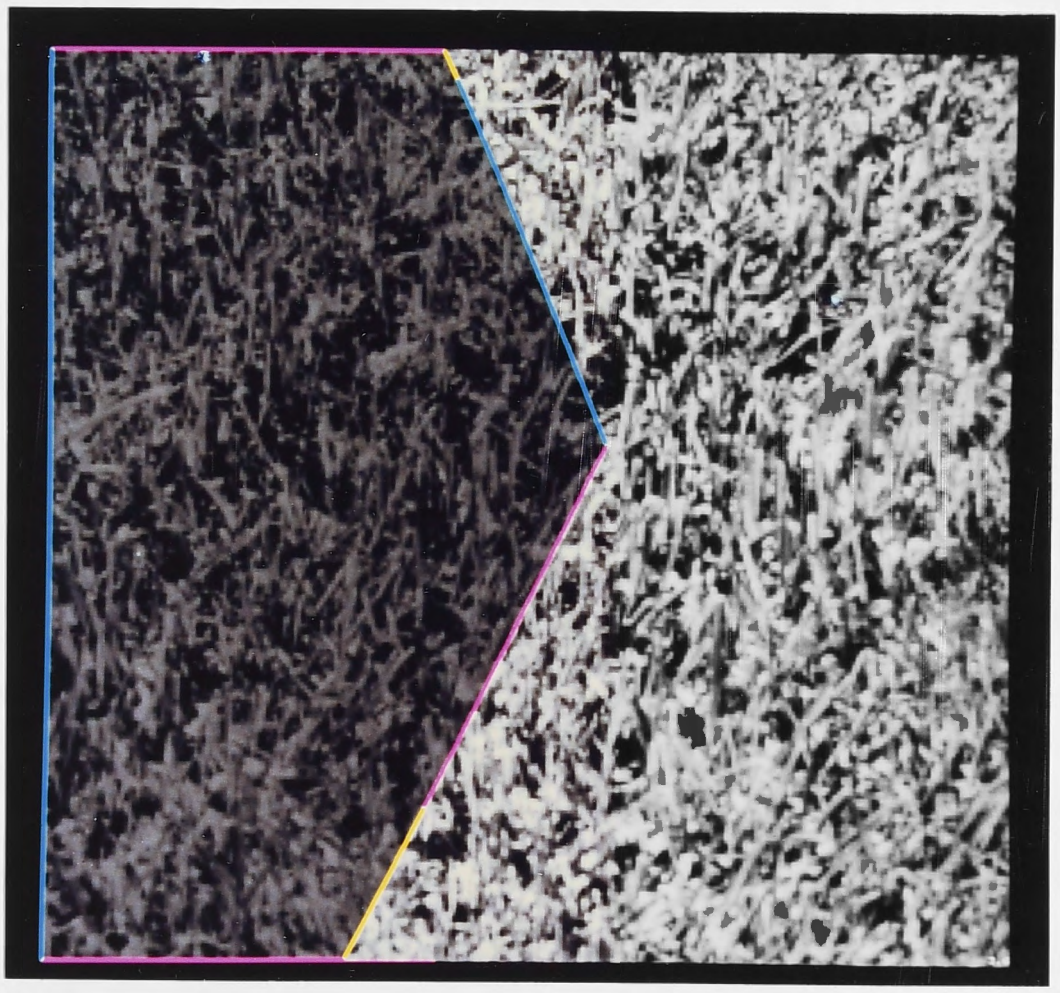


Figure 3.6: Seven-segment boundary

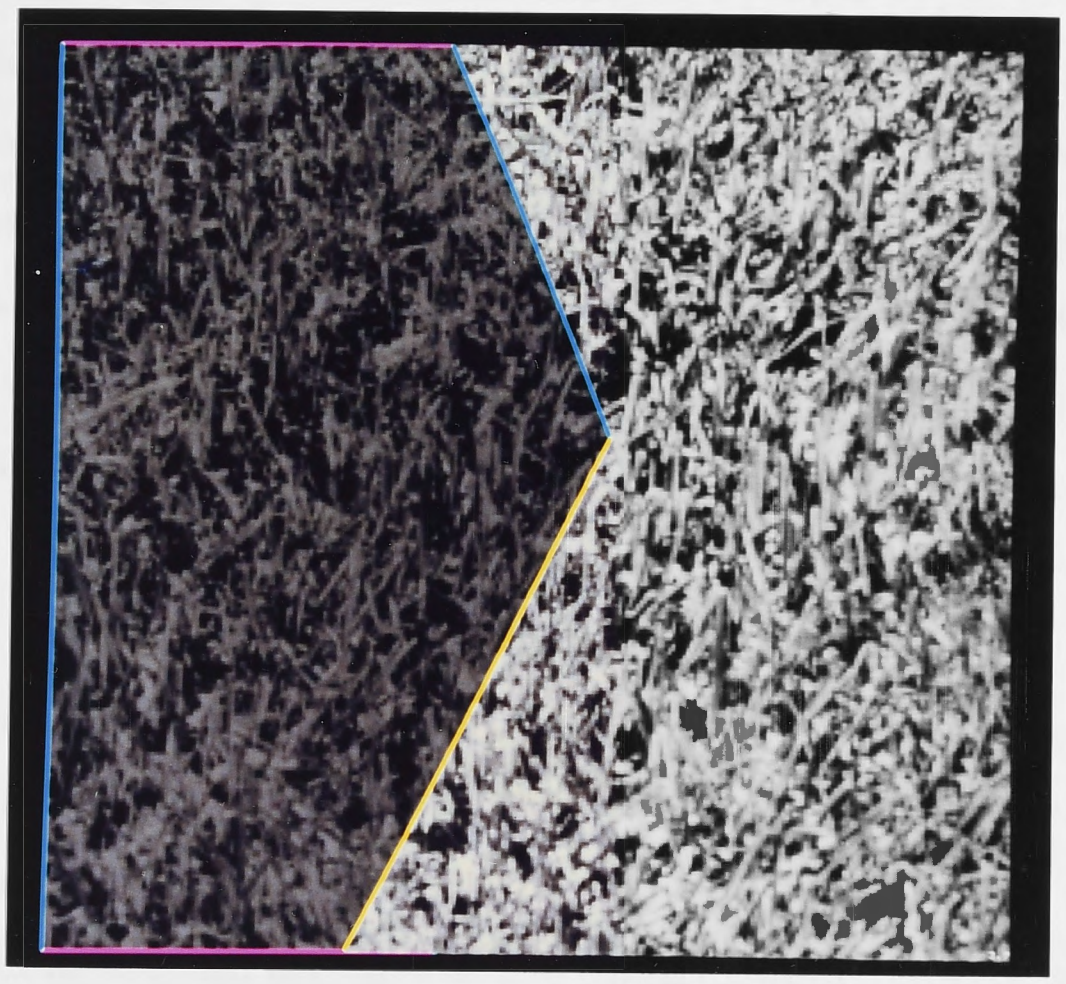


Figure 3.7: Final results from BLM algorithm for figure 2.1

tions. This is also the configuration which has been used in testing the convolution edge detectors in the following chapter.

3.4.1 Boundary representation

Boundaries are represented as a set of knots joined by either linear or cubic polynomial segments. In general, there could be more than two segments connected to each knot. However, since we assume that there are only two connected regions, each knot will have two segments attached. Where the boundary extends from one side of the image to the other, it should be completed along the borders of the image so as to enclose one of the regions. Each pixel is therefore either inside or outside the boundary, and is assigned to region 1 or 2 respectively.

Therefore, the boundary is represented as a circular list of knots and segments, enclosing region 1 and linked together in anti-clockwise order around it. Since there are as many segments as knots, the information for each knot and the following segment are combined in one node, which contains the position of the knot, various details about the following segment, and pointers to the next and previous nodes. Section 5.3.1 discusses a more elaborate data structure, suitable for multiple regions and boundaries.

Each boundary segment is expressed as a vector function $\mathbf{z}_k(t) = (x_k(t), y_k(t))$ for segment k, with $0 \leq t \leq 1$. (Vectors are denoted by bold type; the components of a vector $\mathbf{u}$ are $\mathbf{u}^x$ and $\mathbf{u}^y$. As a special case, the components of $\mathbf{z}$ are x and y.) The boundary runs anti-clockwise with increasing k and t. The knot positions are $\mathbf{z}_k$, for $0 \leq k \leq N$, where N is the number of knots, with $\mathbf{z}_0 = \mathbf{z}_N$. (These should not be confused: $\mathbf{z}_k$ is a knot position, and $\mathbf{z}_k(t)$ is a function defining a segment.) Both linear and cubic segments have the properties that $\mathbf{z}_k(0) = \mathbf{z}_k$ and $\mathbf{z}_k(1) = \mathbf{z}_{k+1}$, for $0 \leq k < N$. Linear segments are therefore defined by the equation

$$\mathbf{z}_k(t) = \mathbf{z}_k + t\,(\mathbf{z}_{k+1} - \mathbf{z}_k) \tag{3.1}$$

Cubic segments express $\mathbf{z}_k(t)$ as a cubic vector polynomial in t, that is, $x_k(t)$ and $y_k(t)$ are each cubic polynomials. Eight parameters are required, of which four are supplied by the knot positions. Four more parameters per segment are required. The parameters used here are the initial velocity $\mathbf{u}_k$ and final velocity $\mathbf{v}_k$ of the segment, that is, $\dot{\mathbf{z}}_k(0) = \mathbf{u}_k$ and $\dot{\mathbf{z}}_k(1) = \mathbf{v}_k$, where $\dot{\mathbf{z}}(t) = d\mathbf{z}/dt$. It can easily be verified that the cubic vector polynomial satisfying these conditions is

$$\mathbf{z}_k(t) = \mathbf{z}_k b_0(t) + \mathbf{u}_k b_1(t) + \mathbf{v}_k b_2(t) + \mathbf{z}_{k+1} b_3(t) \tag{3.2}$$

where the blending functions $b_i(t)$ are defined as

$$b_0(t) = 1 - 3t^2 + 2t^3 \tag{3.3a}$$

$$b_1(t) = t - 2t^2 + t^3 \tag{3.3b}$$

$$b_2(t) = -t^2 + t^3 \tag{3.3c}$$

$$b_3(t) = 3t^2 - 2t^3 \tag{3.3d}$$

These blending functions are shown in figure 3.8. This formulation of the cubic spline is equivalent to a Bézier spline with the control points $\mathbf{z}_k$, $\mathbf{z}_k + \mathbf{u}_k/3$, $\mathbf{z}_{k+1} - \mathbf{v}_k/3$, and $\mathbf{z}_{k+1}$. Continuity of slope may be enforced at any knot between two cubic segments; this requires that $\mathbf{v}_{k-1} = \mathbf{u}_k$ at knot k, if $1 \leq k < N$, or $\mathbf{v}_{N-1} = \mathbf{u}_0$ at knot 0.

The complexity of a piece-wise linear boundary is simply $2N$ parameters, that is, two parameters per segment. If there are cubic segments, the complexity will depend on whether slope continuity is enforced at any knots. Each cubic segment will contribute six parameters without slope continuity at either end, five if the slope is continuous at one end, or four if the slope is continuous at both ends.

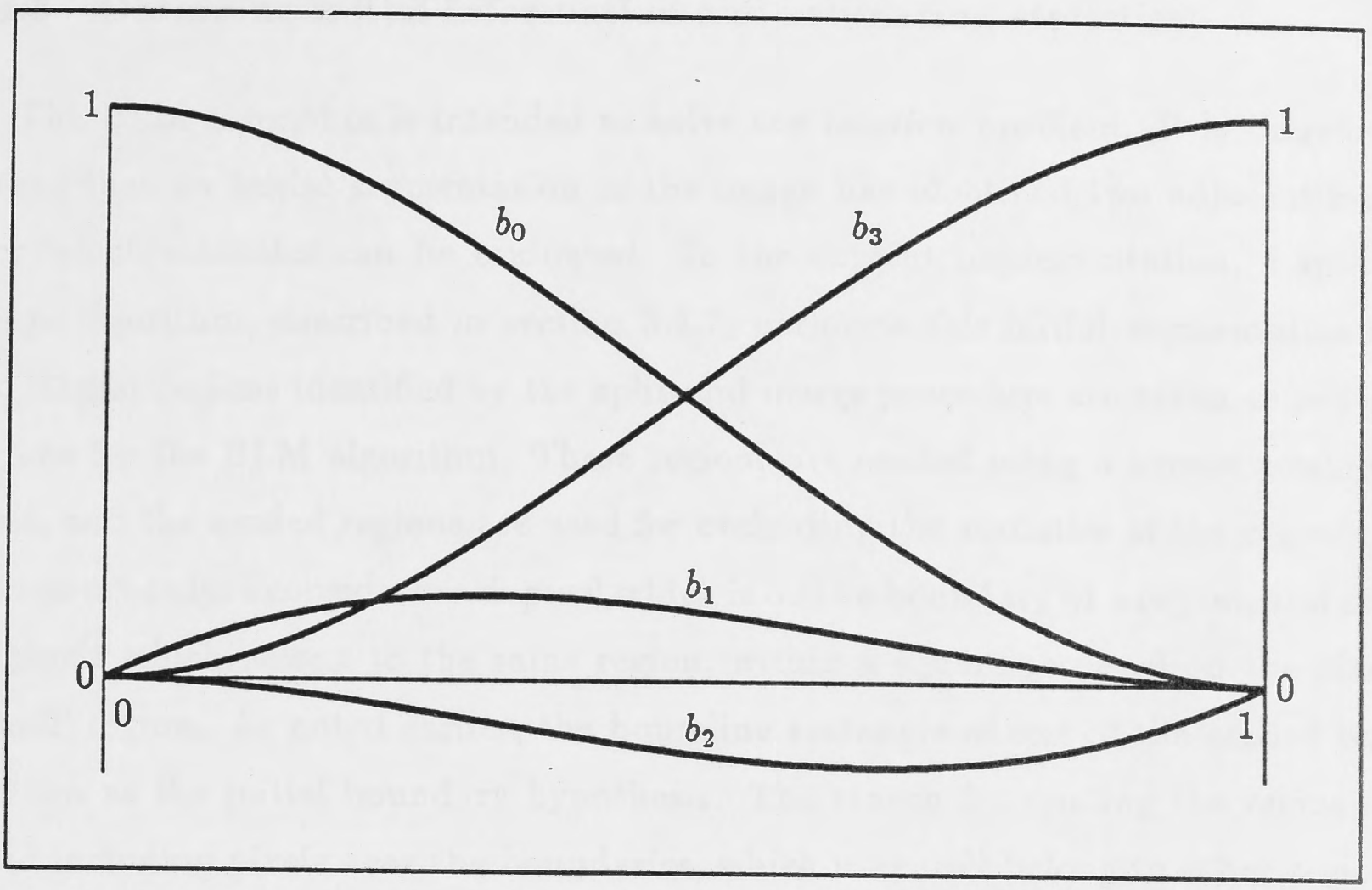


Figure 3.8: Blending functions for cubic segments

3.4.2 Region representation

It is often necessary to deal explicitly with the set of pixels that are contained within a boundary. For example, it is necessary at certain points in the BLM algorithm to determine whether a given pixel is inside or outside the boundary, or to take the intersection of one region with another. These operations could be performed using a representation of the regions in the form of an image, but that would be quite inefficient in terms of both the time and the memory space required. A run-length encoded representation is far more efficient for these purposes.

This representation stores one linked list for each scan-line occupied by the region. Each list consists of a series of nodes which represent the x-coordinates at which one enters or leaves the region on a left-to-right traversal of the scan-line. It is easiest to think of this representation as being derived from an image representation of the region, although it can be determined from the parametric boundary description in a simple manner. Each node then stores an x-coordinate and the corresponding difference between pixels at that point on the scan-line. The nodes are linked together in order of their x-coordinates.

3.4.3 Obtaining initial information and estimating statistics

The BLM algorithm is intended to solve the *location* problem. It is therefore assumed that an initial segmentation of the image has identified two adjacent regions, over which statistics can be evaluated. In the current implementation, a split-and-merge algorithm, described in section 3.4.7, performs this initial segmentation. The two largest regions identified by the split-and-merge procedure are taken as prototype regions for the BLM algorithm. These regions are eroded using a square eroding element, and the eroded regions are used for evaluating the statistics of the regions. The erosion procedure considers each pixel which is on the boundary of a region, and assigns all pixels which belong to the same region, within a square centered on the pixel, to a 'null' region. As noted earlier, the bounding rectangle of one of the eroded regions is taken as the initial boundary hypothesis. The reason for eroding the regions is to avoid including pixels near the boundaries, which may well belong to other regions as the boundaries obtained with the split-and-merge procedure are somewhat inaccurate.

These two eroded regions are then used to characterize the regions in terms of their intensity distributions. Intensities are quantized to a pre-specified number of levels, N_G, usually 8. The intensity distributions are estimated by calculating intensity histograms for the two regions, with one bin for each intensity level. For region i ($i = 1$ or 2), let the histogram be $h_i(x)$ where x is the quantized intensity level, let the total number of pixels be N_i, and let the probability distribution of quantized intensity be $p_i(x)$. The simplest estimate (also the maximum-likelihood estimate) of $p_i(x)$ is $h_i(x)/N_i$. However, this estimate causes severe difficulties when $h_i(x) = 0$, because the log likelihoods calculated in the following steps become infinite. Also, estimating $p_i(x)$ as 0 amounts to saying that there are definitely no pixels of intensity x in region i; it does not seem reasonable to make such a dogmatic statement on the basis of just one part of a region. An alternative estimate which avoids these difficulties is used:

$$p_i(x) = \frac{h_i(x) + 1}{N_i + N_G} \tag{3.4}$$

When $h_i(x)$ is large, this estimate is practically the same as the previous estimate; when $h_i(x)$ is zero, this estimate is small but non-zero. The estimated probabilities add to 1.

3.4.4 Likelihood maximization

Given a boundary hypothesis with a certain number of segments, the task of the optimization step is to adjust it so as to maximize its likelihood, by moving the knots, and by modifying the initial and final velocities of any cubic segments. This step thus produces a maximum likelihood estimate of the boundary position, constrained by the number and type of segments in the current hypothesis.

The likelihood to be maximized is the probability of the event that the image was generated by a random process that assigns intensities to the pixels inside the boundary according to distribution $p_1(x)$, and assigns intensities to those outside according to distribution $p_2(x)$. Here $p_1(x)$ and $p_2(x)$ are the estimates of the pixel intensity distributions on either side of the boundary.

The logarithm of the likelihood is used, rather than the likelihood itself, because of the enormous range of likelihoods encountered; the log likelihood can easily be tens of thousands. Maximizing the log likelihood is equivalent to maximizing the likelihood, since the logarithm is a monotonic function. The log likelihood can be calculated as a sum over the pixels inside the boundary:

$$L = \sum_{(i,j)\in R} \ln(p_1(I_{ij})) + \sum_{(i,j)\in \overline{R}} \ln(p_2(I_{ij})) \tag{3.5a}$$

$$= \sum_{(i,j)\in R} \lambda(I_{ij}) + \sum_{\text{all }(i,j)} \ln(p_2(I_{ij})) \tag{3.5b}$$

where L is the log likelihood for the boundary, R is the set of pixels enclosed by the boundary, I_{ij} is the quantized intensity of pixel (i,j), and $\lambda(x)$ is a log likelihood ratio function defined as

$$\lambda(x) = \ln\left(\frac{p_1(x)}{p_2(x)}\right) \tag{3.6}$$

Here p_1 and p_2 are the estimates of the pixel intensity distributions for the two regions adjacent to the boundary. The second term in equation (3.5b) does not depend on the boundary, and therefore does not enter the optimization.

Given the function $\lambda(x)$, we define an image λ_{ij} (called the λ-image), where $\lambda_{ij} = \lambda(I_{ij})$. Thus, λ_{ij} is an image with predominantly positive pixels in region 1, and predominantly negative pixels in region 2. The problem of estimating the boundary then becomes one of finding the boundary which gives the greatest possible sum of the pixels of λ_{ij} within the boundary.

We assume that each pixel is on one side of the boundary or the other; pixels are never split across the boundary and assigned partly to each region. That is, pixels are assumed to be points. One consequence of this assumption is that the log likelihood will change in discrete jumps with continuous alterations in the boundary shape. For example, as a knot is moved, the log likelihood will change every time the boundary crosses a pixel. That is, the log likelihood as a function of the position of the knot is a staircase function (i.e., a sum of positive and negative step functions).

3.4.4.1 Calculating log likelihoods

The log likelihood for the boundary can be calculated in a time dependent on the length of the boundary, rather than the area enclosed, using a 'scan-row sum' image S, obtained by summing λ_{ij} along scan lines. That is,

$$S_{ij} = \sum_{k=1}^{j} \lambda_{ik} \tag{3.7}$$

The log likelihood is calculated by tracing anti-clockwise around the boundary, and for each intersection with a scan-line, adding or subtracting the corresponding scan-line sum. It is added if the boundary direction is upwards, and subtracted if the direction is downwards. Generally, the scan lines run horizontally; vertical scans are also used in optimizing cubic segments (the sum is then added if the direction is leftwards, and subtracted if the direction is rightwards).

This procedure allows an important simplification: the log likelihood for the whole boundary can be expressed as the sum of the log likelihoods for each segment. That is, the procedure described above can be applied to individual segments, and the results added together to obtain the total log likelihood. The log likelihood sum obtained with the procedure described above is defined, whether the boundary contour is closed or not; however, it can only be interpreted as a genuine log likelihood if it is closed, because only then does it enclose a region of the image.

To show that this procedure for calculating the log likelihood is correct when the boundary is closed, we first define the *rotation number* of a point with respect to the boundary. This is simply the total number of anti-clockwise rotations that the line from the point to a point on the boundary makes, as the point on the boundary makes one circuit of the boundary. For a simple (i.e., not self-intersecting) boundary which runs anti-clockwise, the rotation number will be zero for points outside the boundary,

and 1 for points inside. If the boundary runs clockwise, the rotation number will be -1 for points inside. If the boundary intersects itself, it is possible for some points to have rotation numbers greater than 1 or less than -1.

Each pixel in the image makes a contribution proportional to its rotation number; the procedure described above calculates

$$L = \sum_{i,j} r(i,j)\lambda_{ij} \tag{3.8}$$

where $r(i,j)$ is the rotation number of point (i,j) with respect to the boundary. To see this, consider the intersections of the boundary with a line extending horizontally from point (i,j) to the right-hand border of the image. The rotation number will be equal to the difference between the number of intersections where the boundary direction is upwards, and the number of intersections where the direction is downwards. In the above procedure, each intersection where the direction is upwards will contribute λ_{ij} to the sum; each intersection where the direction is downwards will contribute $-\lambda_{ij}$, so the total contribution will be proportional to $r(i,j)$.

At this point, there are two coordinate systems: the (i,j) indices of pixels in the image, and the (x,y) cartesian coordinate space for describing the boundary. Here i and j are integers, with $1 \leq i \leq N_R$ and $1 \leq j \leq N_C$, where N_R and N_C are the number of rows and columns in the image, whereas x and y are reals. We use the convention that pixel (i,j) is at position $(j - \frac{1}{2}, i - \frac{1}{2})$. Since row 1 is the top row of the images shown in this thesis, the y-axis points downwards on these images.

Calculating the log likelihood requires determination of which side of the boundary each pixel is on; where the boundary intersects a point at coordinates $(j-\frac{1}{2}, i-\frac{1}{2})$, some convention is required to assign it to one side or the other. In terms of the calculation procedure above, there are three ambiguous situations:

(a) where an intersection of the boundary with a scan-line is at an x-coordinate of $j - \frac{1}{2}$,

(b) where the boundary is actually horizontal where it intersects a scan-line, and

(c) where a knot is on a scan line (i.e., its y-coordinate is $i - \frac{1}{2}$).

Situation (a) causes ambiguity about whether the pixel in column j is to the left or right of the boundary; situations (b) and (c) cause ambiguity about whether the boundary direction is upwards or downwards.

The rule which has been adopted for resolving these ambiguities involves conceptually moving the boundary by an arbitrarily small amount δ to the right, and then (if necessary) an even smaller amount ϵ upwards, where ϵ/δ is arbitrarily small. These movements are sufficient to move the boundary off the intersections with pixel points, but are small enough that no other pixels cross the boundary. This is possible because there are only finitely many pixels, of which one will be the closest to the boundary (excluding those that are actually on the boundary); if ϵ and δ are made much smaller than the distance from this pixel to the boundary, then moving the boundary will not cause any pixel to cross the boundary. This rule is implemented by rounding coordinates appropriately, but the above statement of the rule makes it evident that the rule will give the same results no matter in which direction the boundary is traversed.

In case (a) above, this rule can be implemented by rounding upwards when converting a real x-coordinate to an integer column number in the row-sum image s. Thus, if the boundary intersects row i at coordinates $(x, i - \frac{1}{2})$, then s_{ik} is added or subtracted, where $k = \lfloor x + \frac{1}{2} \rfloor$. (Note that $s_{i0} = 0$ for all i.) According to this rule, the pixel in column j will be included in the sum if $x \geq j - \frac{1}{2}$.

Where the boundary is composed of linear segments, situations (b) and (c) can also be resolved simply. Situation (b) only arises if a whole segment is horizontal. In this case, moving the boundary up slightly will cause the segment not to intersect the scan-line at all; therefore horizontal segments make no contribution to the log likelihood sum. Situation (c), where a knot is on a scan-line, is resolved according to the rule given above by moving the boundary up slightly, which is equivalent to having the scan-line intersect the boundary slightly below the knot. Therefore, if a boundary segment comes to the knot from below, the intersection at the knot should be included; if it comes from above, it should not be. That is, the intersection should be included for a segment if the knot is the higher of the two endpoints of the segment. This provides a convenient rule which can be applied to each segment individually: if an endpoint is on a scan-line, include its intersection if (and only if) the endpoint is the higher of the two endpoints. The scan-line sum is added or subtracted according to the slope of the segment.

The procedure for calculating the log likelihood of a linear segment can now be stated in detail. Suppose that the segment runs from $\mathbf{z}_0 = (x_0, y_0)$ to $\mathbf{z}_1 = (x_1, y_1)$. The procedure is:

(a) If $y_0 > y_1$, swap $\mathbf{z}_0$ and $\mathbf{z}_1$.

(b) Set $L = 0$.

(c) Perform steps (d) and (e) for all i such that $y_0 < i - \frac{1}{2} \leq y_1$.

(d) Let $y = i - \frac{1}{2}$ and $x = (y - y_0)(x_1 - x_0)/(y_1 - y_0)$.

(e) Add S_{ij} to L, where $j = \lfloor x + \frac{1}{2} \rfloor$.

(f) If step (a) swapped $\mathbf{z}_0$ and $\mathbf{z}_1$, negate L.

For a cubic segment, $y_k(t)$ is divided into monotonic sections. There will be at most three sections, since a cubic polynomial can have at most one minimum and one maximum. The maximum and minimum, if present, are located by solving $\dot{y}_k(t) = 0$, which is quite straightforward as $\dot{y}_k(t)$ is quadratic. Each monotonic section is then treated in much the same manner as a linear segment. If one or both endpoints of a monotonic section are on a scan line, only the higher endpoint is included, as before. The likelihood for a cubic segment is then calculated by applying a procedure, very similar to that shown above, to each monotonic section. The difference lies in how x is calculated in step (d); now $x = x_k(t)$, where t is obtained by solving $y_k(t) = y$. Solving the cubic equation for y is performed numerically by the Newton-Raphson method, starting from the values of t and y from the previous scan-line. This is quite efficient, as the previous value of t is usually close to the correct answer. For the first scan-line, y and t for the appropriate endpoint of the monotonic section are used.

A simple adaptation of the procedure described above for calculating the log likelihood of a boundary gives a procedure for generating a run-length encoded representation of the region. In fact, it generates a representation of the rotation number function $r(i, j)$, which is 1 for points inside the boundary and 0 for points outside, assuming that the boundary is closed and non-intersecting, and travels anti-clockwise. As before, the procedure considers the intersections of the boundary with the scan-lines. Suppose that such an intersection occurs on scan-line i at x-coordinate x. If the direction of the intersection is upwards, then the rotation number decreases at that point, that is, $r(i, j+1) = r(i, j) - 1$ where $j = \lfloor x + \frac{1}{2} \rfloor$. Consequently, a new node should be inserted in list i of the run-length encoded representation, at x-coordinate j with a change of -1. Similarly, if the direction is downwards, a node is inserted with a change of $+1$. If there is already a node at x-coordinate j, its change is updated, rather than inserting another node at the same x-coordinate. Calculating the total log likelihood from this

representation is very straightforward. Each node is considered in turn, and $-c \cdot S_{ij}$ is added to the total likelihood, for a node at x-coordinate j on row i with change c.

3.4.4.2 Optimization procedure

The optimization procedure used here is a hill-climbing optimization which moves each knot in turn so as to increase the likelihood, until the boundary as a whole converges to a stable position. The iteration is guaranteed to converge, since each knot is only moved if that will increase the likelihood, and there is a lower bound on the size of the increase, given by $\min_{x,\lambda(x)\neq 0} |\lambda(x)|$.

The overall procedure for optimizing the boundary uses a flag for each knot to indicate whether it should be moved. Initially, all these flags are set. The procedure then considers each knot in turn around the circular list. If the flag is set for the knot, the optimization procedure described in the next paragraph is applied to it. If the knot does not move significantly (i.e., less than a threshold distance ϵ), then its flag is cleared. On the other hand, if it does move, the flags for the preceding and following knots are set, because these knots may be no longer at their optimum position.

Knots with both adjacent segments of linear form are optimized using different strategies to those used when one of the adjacent segments is of cubic form. When both adjacent segments are linear, advantage can be taken of the simpler geometry to make the optimization both more efficient and more thorough.

3.4.4.3 Optimization procedures for linear segments

The procedure for optimizing a knot without adjacent cubic segments involves a series of one-dimensional optimizations, which can be performed efficiently. Each one-dimensional optimization considers moving the knot along an 'optimization line', and chooses the position on the optimization line which gives the greatest likelihood. The optimization line is a straight line segment of limited length which passes through the current position of the knot. The steps involved in optimizing each knot are therefore:

(i) Choose an initial direction to move the knot.

(ii) Perform up to a specified number (usually 3) of the one-dimensional optimizations described in steps (iii) to (v); then stop.

(iii) Optimize the knot position in the current direction.

(iv) Optimize the knot position in the perpendicular direction.

(v) If the knot moved less than a threshold distance ϵ in steps (iii) and (iv), then stop.

(vi) Set the current direction to the overall direction in which the knot moved in steps (iii) and (iv), and return to step (iii).

The parameter ϵ controls the accuracy of the boundary and the time taken to converge. Its value is not at all critical. A value of 0.1 pixels gave excellent results without causing the optimization to become excessively slow.

It is quite important to prevent the boundary from intersecting itself or the border of the image. In the more general situation with many regions and boundaries, the boundary being optimized must also be prevented from intersecting any other boundary. Otherwise, the boundary may turn itself inside out, or go around the region twice, or include pixels that have already been assigned to the other side of a neighbouring boundary. There are therefore limits on how far a knot may be moved along a given optimization line in either direction.

For those knots far from any other boundary segment, there is an arbitrary limit $d_{\max}$ imposed on the distance moved. The optimization then proceeds more smoothly, since the knots tend to move in concert towards the correct boundary. Imposing the limit also reduces the computation required in each one-dimensional optimization, since a smaller area of the image is considered.

The optimization line is defined by the direction chosen and the distance limits determined as described in section 3.4.4.7. The one-dimensional optimization procedure, described in the next section, finds the position of maximum likelihood along this line. This maximum is not a global maximum, since the length of the optimization line is limited. Neither is it strictly a local maximum, since the procedure can pass over a local maximum close to the knot in favour of a higher maximum further along the optimization line. Thus the maximum could be called a 'regional' maximum. The limit on the distance moved, $d_{\max}$, should therefore not be too small, lest the optimization become stuck at a local maximum. A value of 10 pixels gave good results on the textures considered. The value used is not critical, but should be at least comparable to the scale of the texture being considered.

The initial direction is chosen by considering all positions around a circle of small radius, centered on the current position of the knot. However, if the knot is very close to another boundary segment, the direction parallel to that segment will be chosen instead; if an adjacent segment is very close to some other knot, the direction of that segment is used. These considerations ensure that a useful direction is chosen when the movement of the knot is constrained by other parts of the boundary.

The procedure outlined above works well, and shows little tendency to choose a solution which is globally below optimum. Partly this is because each one-dimensional optimization uses information in an extended part of the image, namely, all the pixels adjacent to the boundary between the neighbouring two knots. The sequence of directions used is also important. Considering two perpendicular directions will usually produce some improvement, even if the initial direction was badly chosen. The overall direction of movement is then a better estimate of the direction of steepest ascent; the third optimization can often move the knot much further than the previous two. It is not possible to choose a direction by considering derivatives, since the derivative is zero almost everywhere—the likelihood changes in discrete jumps as the boundary moves.

3.4.4.4 Choosing the initial direction

The procedure for choosing the initial direction of optimization considers a small circle of radius r (usually 2 pixels). The new position of the knot is defined by

$$x_1' = x_1 + r\cos\theta \tag{3.9a}$$

$$y_1' = y_1 + r\sin\theta \tag{3.9b}$$

The procedure then considers each pixel which will be crossed by the boundary as θ goes from 0 to 2π. For each such pixel, the θ values at which the boundary will cross the pixel are calculated and put on a list, along with the associated change in the log likelihood. This change equals λ_{ij} if pixel (i,j) is crossing from outside the boundary to inside, or $-\lambda_{ij}$ if it is crossing to the outside. The procedure then sorts the list by θ value, and then proceeds through the list, accumulating the associated log likelihood changes; the interval of θ giving the maximum total log likelihood is noted. The centre of this interval is used as the initial direction.

3.4.4.5 One-dimensional optimization of knot position

Given a knot and an optimization line, the one-dimensional optimization procedure divides the optimization line into intervals in which the boundary does not cross any pixels. There is only a finite number of these intervals, since there is only a finite number of pixels. The interval is chosen for which the likelihood is highest. The knot is moved to the centre of this interval, unless it is already within the interval.

Let the knot being moved be knot 1, at position $\mathbf{z}_1 = (x_1, y_1)$, and the preceding and following knots be knots 0 and 2, at positions $\mathbf{z}_0$ and $\mathbf{z}_2$, respectively. Only the two segments adjacent to a knot are affected by moving the knot, so the optimization requires knowledge of the positions of only the adjacent knots. Let $\mathbf{z}_d$ be a unit vector in the direction of the optimization line. Knot 1 will be moved to a position

$$\mathbf{z}_1' = \mathbf{z}_1 + \alpha \mathbf{z}_d \tag{3.10}$$

where α is the distance that the knot is moved. The procedure described in section 3.4.4.7 limits this distance, by limiting the range of α values to $[\alpha_{\min}, \alpha_{\max}]$.

The procedure then considers each pixel which will be crossed by the boundary as α goes from $\alpha_{\min}$ to $\alpha_{\max}$. For each such pixel, the α value(s) at which the boundary will cross the pixel are calculated and put on a list, along with the associated change in the log likelihood. The procedure then sorts the list by α value, and proceeds through the list, accumulating the associated log likelihood changes; the interval of α giving the maximum total log likelihood is noted. If this range does not include $\alpha = 0$, the knot is moved to the position corresponding to the mid-point of the range.

The mid-point is chosen because it is as far away as possible from the ends of the range, which correspond to positions where the boundary is crossing a point. This reduces problems associated with the computational uncertainty of determining whether a pixel is on the inside or outside of the boundary. If the range does include $\alpha = 0$, then no increase in likelihood is gained by moving the knot. However, if the knot is near one end of the range, there is an advantage in moving it to the centre, for the reasons just mentioned. (This does not prevent the optimization from converging in practice.)

3.4.4.6 Approximate one-dimensional optimization

An approximate technique can be used in the initial stages of the BLM algorithm, when the current hypothesis is not close to the true boundary. This technique is faster than the straight-line optimization described above, but less accurate, since it involves using an approximation to the log likelihood data. Here, the scan-lines of the row-sum image S_{ij} are approximated by piece-wise linear functions. The procedure to determine these approximate PWL functions uses a recursive strategy. Initially, the whole line is treated as one linear segment. If any point deviates by more than a pre-set threshold from the linear approximation, it is split into two pieces at the point of maximum deviation, and each piece is checked and split if necessary.

Given this piece-wise linear approximation, the log likelihood will be a continuous function of α in a straight-line optimization, with the derivative defined and continuous almost everywhere. The derivative will have jumps where the boundary crosses one of the breakpoints of the scan-line PWL approximations, and where $\mathbf{z}'_1$ crosses a scan-line. As before, the procedure makes a list of these α values, and the corresponding changes in the derivative, and then sorts the list and processes it in increasing order of α. A local maximum in the log likelihood will occur where the derivative crosses zero from positive to negative values, which can occur either at a jump or in the middle of a continuous section. The procedure chooses the local maximum with the greatest log likelihood. Problems can occur if one of the adjacent knots is on a scan-line; if the boundary segment between it and the knot being optimized became horizontal, there would be a step change in the log likelihood. To avoid this problem, the knot is moved slightly to one side of the scan-line.

Since the derivative is defined almost everywhere, it is also possible to calculate the direction of steepest increase in likelihood. This direction is used as the initial direction for the optimization line. It is calculated by considering the derivative for movements in four directions, namely, in both directions parallel to each of the two adjacent boundary segments. Some care needs to be taken if the boundary is at the position of a jump in the derivative, because the surface of the approximate log likelihood as a function of the knot position can be locally composed of up to four planes of different slopes. Using the derivatives calculated for movements in the four directions, the direction of greatest derivative can be calculated.

3.4.4.7 Constraining the movement of a knot

The procedure for determining how far a knot can be moved in a given direction is responsible for ensuring that the boundary does not intersect itself or another boundary. The current implementation of the BLM algorithm allows manual specification of a bounding polygon which the boundary may not cross. This enables attention to be restricted to a given part of the image. If no bounding polygon is specified, the rectangular border of the image is used. Thus, there are two boundaries to be considered—the boundary being optimized and the bounding polygon.

The procedure computes the allowable range of α values, given the knot positions and the direction of the optimization line. Let the knot being optimized be knot 1; the segments adjacent to it are then segments 0 and 1. Suppose that both of these segments are linear. In outline, the procedure considers all linear segments in the boundary being optimized, other than segments 0 and 1, and all linear segments in the bounding polygon. For each such segment, and for both segment 0 and 1, the procedure computes the range of α values for which the two segments will intersect. The union of all these ranges is taken. The interval containing zero which is not in this union is then the allowable range of α values. This range is also limited by the maximum length of the optimization line, as discussed above.

Figure 3.9 illustrates the procedure for determining when two linear segments will intersect. The segment from $\mathbf{z}_0$ to $\mathbf{z}'_1$ will intersect the segment from $\mathbf{z}_k$ to $\mathbf{z}_{k+1}$ if $\mathbf{z}'_1$ lies in the shaded region. Therefore, it is sufficient to find the intersection of the optimization line with the shaded region. Since the shaded region is always convex, the optimization line will have only one interval of intersection with the shaded region, giving one interval of α values for which the segments will intersect.

For the sake of efficiency, a simple test is first performed to determine whether the two segments can possibly intersect. This involves taking the bounding rectangle of the segment from $\mathbf{z}_k$ to $\mathbf{z}_{k+1}$, and the bounding rectangle of the quadrilateral defined by the points $\mathbf{z}_0$, $(\mathbf{z}_1 - d_{\max}\mathbf{z}_d)$, $\mathbf{z}_2$, and $(\mathbf{z}_1 + d_{\max}\mathbf{z}_d)$. If these bounding rectangles do not intersect, the segments cannot intersect, and the segment from $\mathbf{z}_k$ to $\mathbf{z}_{k+1}$ need not be considered any further.

The problem of determining when segment 0 or 1 will intersect a cubic segment is far more difficult. The cubic segment may not be contained within the sector defined

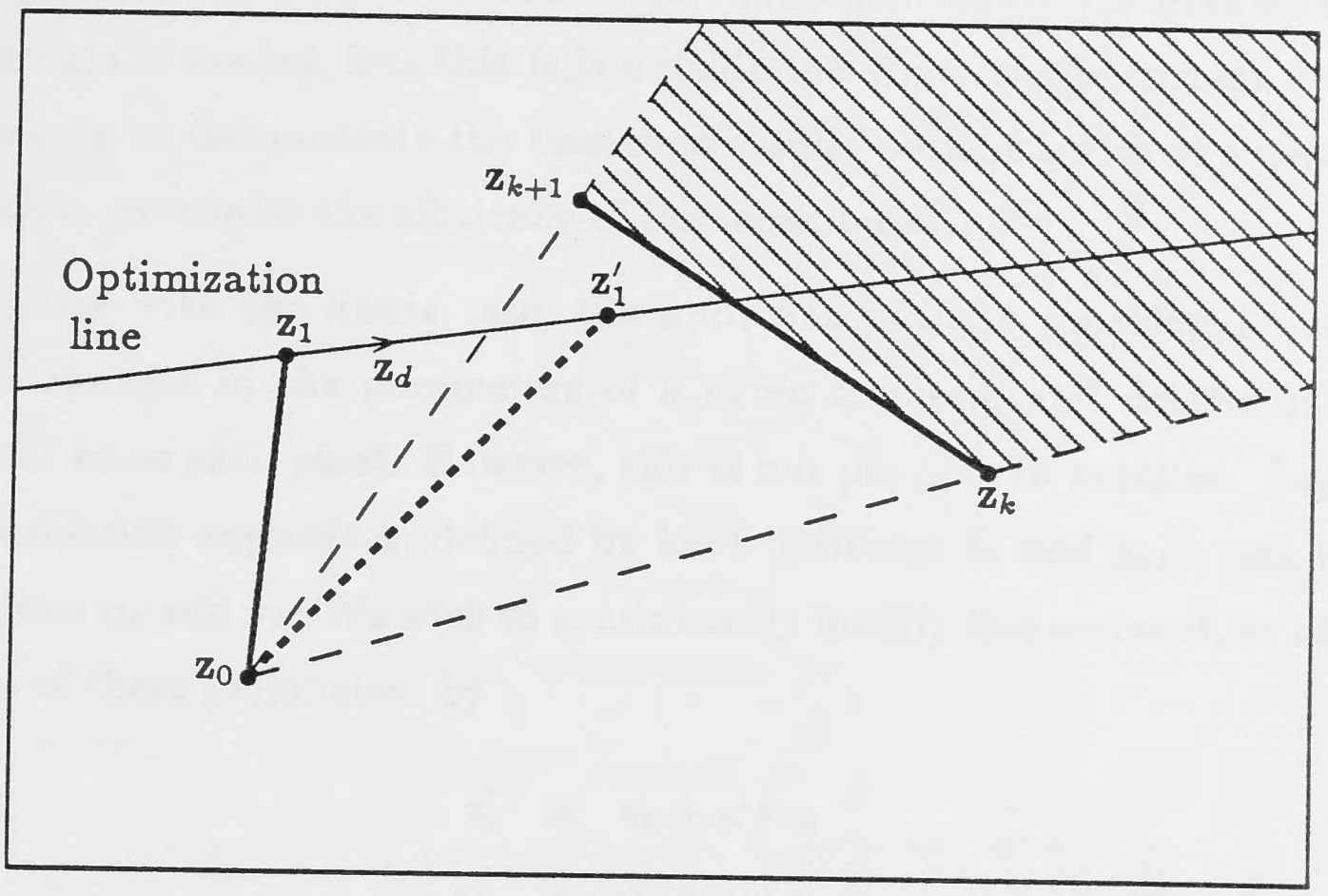


Figure 3.9: Determination of allowable range of positions of z'_1

by the endpoints in figure 3.9. Determining the sector within which the curve is contained involves finding points at which the tangent to the curve is parallel to the line from the curve to z_0 (assuming that we are considering segment 0). Locating these points involves solving a 5th-order polynomial. Therefore, an alternative approach is required. Cubic segments can be ignored at this stage, if each pixel considered in the one-dimensional optimization procedure is checked to ensure that it is on the expected side of the boundary. If it is not, it must be on the other side of a cubic segment, and its corresponding α value is therefore too large. In this way, the allowable range of α can be determined during optimization. The check for each pixel can be performed efficiently using a run-length encoded representation of the current region. The test using bounding rectangles can still be employed; pixels need only be checked during optimization if there is at least one cubic segment whose bounding rectangle intersects the bounding rectangle of the quadrilateral defined above.

3.4.4.8 Optimizing cubic segments

Optimizing a knot where one or both adjacent segments are of cubic form is more difficult than when both segments are linear, because it is not possible to take advantage of the geometry in the same way. The current implementation generally finds the correct

maximum, but is rather slow. Further research into efficient procedures for optimizing cubic segments is needed, but this falls outside the scope of the current work, whose primary aim is to demonstrate the improved results obtained with the new approach rather than to maximize the efficiency of implementation.

By analogy with the linear case, the approach initially considered was to make continuous changes in the parameters of a cubic segment, and determine when the segment will cross each pixel. However, this is not possible in practice. Suppose that we are considering segment k, defined by knot positions $\mathbf{z}_k$ and $\mathbf{z}_{k+1}$, and initial and final velocities $\mathbf{u}_k$ and $\mathbf{v}_k$. We wish to continuously modify this segment, so let us define new values of these parameters by

$$\mathbf{z}'_k = \mathbf{z}_k + \alpha\,\delta\mathbf{z}_k \tag{3.11a}$$

$$\mathbf{u}'_k = \mathbf{u}_k + \alpha\,\delta\mathbf{u}_k \tag{3.11b}$$

$$\mathbf{v}'_k = \mathbf{v}_k + \alpha\,\delta\mathbf{v}_k \tag{3.11c}$$

$$\mathbf{z}'_{k+1} = \mathbf{z}_{k+1} + \alpha\,\delta\mathbf{z}_{k+1} \tag{3.11d}$$

This corresponds to a one-dimensional optimization within the 8-dimensional space defined by the parameters, with $\delta\mathbf{z}_k$ etc. defining the direction of movement, and α defining the size of the movement as before. The new position of the segment is then given by

$$\mathbf{z}'_k(t) = \mathbf{z}_k(t) + \alpha\,\delta\mathbf{z}_k(t) \tag{3.12}$$

where $\delta\mathbf{z}_k(t)$ is defined from $\delta\mathbf{z}_k$, etc., in the same way as $\mathbf{z}_k(t)$ is defined from $\mathbf{z}_k$, etc. For a given pixel at location $\mathbf{z}$, we then wish to determine for which α there exists a t such that $\mathbf{z}'_k(t) = \mathbf{z}$. In its essential form, the problem can be stated as follows: for which α do the equations

$$p_1(t) + \alpha q_1(t) = 0 \tag{3.13a}$$

$$p_2(t) + \alpha q_2(t) = 0 \tag{3.13b}$$

have simultaneous solutions, when p_1, p_2, q_1, and q_2 are cubic polynomials? Multiplying the first by q_2 and the second by q_1 and subtracting gives the equation:

$$p_1(t)q_2(t) - p_2(t)q_1(t) = 0 \tag{3.14}$$

This is a sixth-order polynomial equation, and is consequently difficult to solve. It could be solved numerically, but it must be solved for each pixel in the area swept out

by the segment as it moves; this approach would be extremely time-consuming, and was therefore rejected.

The approach which has been adopted involves two elements:

(a) a procedure which uses an estimate of the deviations of the current boundary from the true boundary to update the segments, and

(b) a systematic procedure for searching the parameter space at a knot.

Rather than considering one cubic segment, we consider one knot, and the parameters which affect the curve near the knot. These parameters are the knot position $\mathbf{z}_k$, and either or both of the initial velocity $\mathbf{u}_k$ and the previous final velocity $\mathbf{v}_{k-1}$ (where $\mathbf{v}_{-1} = \mathbf{v}_{N-1}$), depending on which segments are cubic. There are four possible cases:

1. Two cubic segments, without slope continuity.
2. Two cubic segments with slope continuity.
3. A cubic segment followed by a linear segment.
4. A linear segment followed by a cubic segment.

Procedure (a) above estimates the deviations of the current boundary from the true boundary using the procedure for finding protrusions to check in the consistency test. This procedure, described in section 3.4.5.1, considers displacing short pieces of a boundary segment in the direction perpendicular to the segment, and finds the most likely position. When the current boundary is rather far from the true boundary, this procedure can give a reasonable indication of how the boundary shape should be changed. The protrusions determined by this procedure can be seen as outlining a boundary shape of higher likelihood.

Given this irregular, but more likely, boundary shape, procedure (a) above then takes the x- and y-components of the distance between the segment and the protrusions, as a function of t, for both of the segments adjacent to the knot. The t values for the segment before the knot are re-scaled to run from -1 to 0, giving the x- and y-deviations as a function of t, with t running from -1 at the previous knot to 1 at the following knot. The next step is to perform linear least-squares fits independently in x and y, using the available basis functions. (The set of basis functions depends on which of the above cases applies.) The parameters of the fitted functions give the changes to be made in the parameters defining the two segments.

The likelihood of the boundary with the changed parameters is then compared with the original likelihood. If it is greater, the changes are accepted. If not, the changes are used to define an optimization line (in a space of up to six dimensions); several positions on the line are tested, and the position giving the greatest likelihood increase is accepted. If none of the positions give an increase, the search procedure described below is invoked. Each separate position that is tested involves re-calculating the likelihood of the two segments.

This procedure works rather well when the current boundary is remote from the true boundary, but as the fit improves it becomes inappropriate in some ways. The most significant problem is that the x- and y-deviation functions to be approximated are not independent, and are not fixed. Changing either of the x- and y-polynomials will change the value of t at which a particular protrusion occurs, and therefore distort both of the deviation functions. The procedure is trying to hit a moving target. This problem is observed particularly when one segment needs to be extended.

Another problem arises because there are infinitely many functions $\mathbf{z}_k(t)$ representing a given curve, but it is possible that only one of them can be represented in cubic polynomial form. Even if the deviation functions represent accurately the difference between the true boundary and the current boundary, they still only represent one possible parametrization of the true boundary, which may not be appropriate for cubic representation. The deviation functions do not necessarily represent the true boundary, either, because they tend to be affected by variations in the textures in the regions.

Other significant problems arise because the log likelihood does not, in general, fall off quadratically as the boundary is displaced from the true boundary. Rather, it usually falls off linearly, assuming homogeneous regions, and it may have different slopes on the two sides of the true boundary. The unweighted least-squares criterion therefore gives too much weight to some protrusions, and not enough to others, and consequently cannot reliably determine the best direction to move when the current boundary is close to the true boundary.

Consequently, a systematic search procedure is used to search the space of parameters at the knot, once the curve-fitting procedure can produce no further gain. Although the space can be up to six-dimensional, the search is always four-dimensional. Six parameters only apply when both segments are cubic, and the slope is not continuous; in this case, the initial velocity $\mathbf{u}_k$ and the previous final velocity $\mathbf{v}_{k-1}$ can be optimized

separately for each value of the knot position $\mathbf{z}_k$ considered, because each velocity only affects one of the segments.

The search procedure considers a range of positions for the knot, defined by a set of square grids with decreasing spacing. These grids consist of nine points, centered on the original knot position. At each position, a somewhat similar search pattern is used to determine the best values for $\mathbf{u}_k$ and $\mathbf{v}_{k-1}$; if they provide an improvement in likelihood over the original value, the search terminates. The spacings used start at a maximum value (usually 10 pixels) and decrease by a factor of 2 each time until a minimum value is reached (usually 0.1 pixels). The central position only needs to be considered on the first iteration.

The searches in the two-dimensional subspaces for $\mathbf{u}_k$ and $\mathbf{v}_{k-1}$ also consider a set of square grids with decreasing spacing. These searches are slightly different from the search for the knot position, in that the second and subsequent grids are centered on the best position found with the previous grid, and the search proceeds to the smallest spacing. The maximum spacing is usually 30 pixels, and the minimum 0.3 pixels. The spacings are three times as large for the velocities as for the knot positions, because this corresponds to equal spacings for all four of the control points of the equivalent Bézier spline.

The techniques described above for preventing the boundary from intersecting itself are not applicable when the knot has an adjacent cubic segment, because of the difficulties of determining when a cubic segment will intersect another segment as its parameters are changed. Instead, the rotation number of each pixel is checked; if the rotation number is other than 0 or 1 for any pixel inside the bounding polygon, or other than 0 for any pixel outside it, that combination of parameters must be rejected. This test can be performed quite efficiently if run-length encoded representations are precalculated for the bounding polygon, and for those segments of the boundary that are not changing. As each combination of parameters is considered at a knot, a run-length representation of the two adjacent segments can be calculated and checked.

3.4.5 Consistency test

The boundary obtained by the likelihood maximization step will be the best obtainable with the given number of segments, but the number of segments may be inadequate. The next step is therefore to test whether the boundary hypothesis is consistent

with the image data. The consistency test was described in outline in section 3.2.3; this section considers implementation of the test in more detail.

In practice, the test is applied to each segment in turn. The three basic elements of the test are:

1. Find a patch to test.
2. Calculate the log likelihood ratio statistic for the patch.
3. Calculate the mean and variance of the log likelihood ratio statistic under the null hypothesis, and decide whether the null hypothesis can be rejected.

3.4.5.1 Finding patches to test

The procedure for finding patches to test searches along the length of the current boundary segment for areas adjacent to the segment which are more likely to belong on the other side. This procedure is as follows:

1. Divide the segment into short pieces.
2. Consider displacing each piece sideways (i.e., in the direction perpendicular to its length), and choose the most likely position. This creates a 'protrusion' on the boundary.
3. Collect together adjacent protrusions on the same side of the boundary to form patches to be tested.

Dividing the segment into pieces. Dividing the segment into short pieces is simple if the segment is linear; the length of the segment is divided evenly into a number of pieces, with the number chosen so that the length of the pieces is close to a pre-set value (usually 5 pixels). This value is not critical; smaller values generally involve more computation. If the value is much too large, the protrusions will not be able to follow details of the true boundary.

Cubic segments are divided into pieces which can be approximated by straight lines. A standard length for the pieces is determined as described above. The algorithm then starts at the beginning of the segment, and takes a piece of the standard length. If this piece can be approximated by a straight line (in the sense described below), it is accepted; otherwise the length is halved until it can be approximated by a straight line.

The next piece is then taken, starting at the end of the previous piece, and so on until the end of the segment is reached.

Each piece is described by four vectors: two giving the starting and ending points, and two giving the direction perpendicular to the curve at the starting and ending points. For linear segments, these perpendicular directions are constant, whereas they change along the length of a cubic segment. The endpoint and the perpendicular direction at the endpoint equal the starting point of the next piece and the perpendicular direction there.

A piece of a cubic segment can be approximated by a straight line if a line segment can be found which makes identical pixel assignments to the cubic piece, that is, if the area enclosed by the cubic piece and the approximating line contains no pixels. The approximating line is determined as follows: first, a line between the endpoints of the cubic piece is tested. In most cases, it will be an adequate approximation. In some, however, the cubic piece will curve around to enclose some pixels between the cubic and linear pieces. In such cases, the endpoints of the linear piece are displaced in the perpendicular directions until the linear piece crosses all of the enclosed pixels. If this causes the linear piece to also cross other pixels, the length must be reduced as described above. Otherwise, the linear piece is as accepted as an approximation of the cubic piece. Using the approximate linear piece means that the piece can be described by its endpoints alone.

Determining protrusions. The pieces into which the segments are divided then form the basis for finding areas adjacent to the boundary which are more likely to belong on the other side. These areas are called protrusions, and are defined as follows. Consider a piece of a segment, described by its endpoints $\mathbf{z}_1$ and $\mathbf{z}_2$, and its corresponding perpendicular directions $\mathbf{w}_1$ and $\mathbf{w}_2$. (These directions point towards the outside of the boundary.) A protrusion is the area enclosed by $\mathbf{z}_1$, $\mathbf{z}'_1$, $\mathbf{z}'_2$, and $\mathbf{z}_2$, where $\mathbf{z}'_1 = \mathbf{z}_1 + h\,\mathbf{w}_1$ and $\mathbf{z}'_2 = \mathbf{z}_2 + h\,\mathbf{w}_2$ for some h; here h is the height of the protrusion—see figure 3.10.

The task of finding the protrusions involves determining the value of h for which the log likelihood of the protrusion is a maximum. The log likelihood of the protrusion is evaluated from its boundary, using the procedures described above for evaluating the log likelihood of a boundary. If h is positive, the area of the protrusion is positive, because

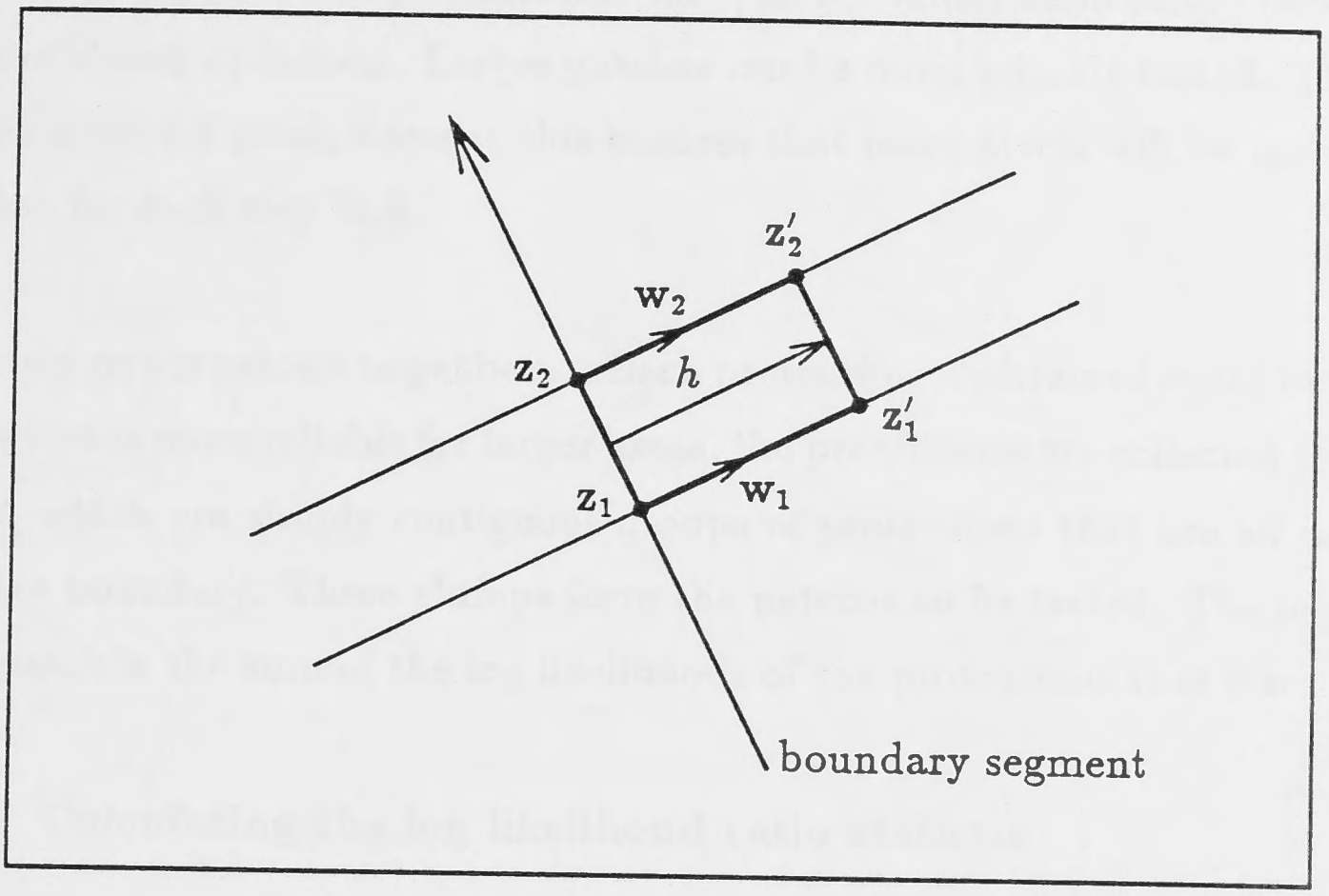


Figure 3.10: Outline of a protrusion

its boundary proceeds anti-clockwise; the pixels within the protrusion are currently assigned to region 2, and the log likelihood of the protrusion will be positive if the pixels in it are more likely to belong to region 1. On the other hand, if h is negative, then the boundary of the protrusion proceeds in a clockwise direction, and the log likelihood of the protrusion will be positive if the pixels in it are more likely to belong to region 2. The protrusion can be seen as a perturbation in the boundary shape; we seek the perturbation which will maximize the log likelihood of the boundary.

The best value of h is determined by simply considering a range of values between $-h_{\max}$ and $h_{\max}$, where $h_{\max}$ is a pre-set parameter. For each value of h, the log likelihood is calculated, and the value of h that gives the highest likelihood is taken. An alternative strategy is to seek only a local maximum, by proceeding out from $h = 0$ only while the log likelihood is increasing. The relative merits of these two strategies are discussed in section 5.1.3.2. As each value of h is considered, it is important to ensure that the protrusion does not intersect some other part of the boundary or the bounding polygon; therefore, any pixels in the protrusion that are not on the expected side of the boundary are removed, as are any that are outside the bounding polygon. This can be done efficiently using run-length encoding of the regions concerned. The usual value for $h_{\max}$ was 10 pixels. This parameter essentially controls a trade-off between sensitivity

of the consistency test and computation time; larger values allow larger patches, at the expense of slower operation. Larger patches can be more reliably tested. The step-size for h was always 1 pixel, because this ensures that more pixels will be included in the protrusion for each step in h.

Collecting protrusions together. Each protrusion so obtained could be tested; but since the test is more reliable for larger areas, the protrusions are collected together into 'clumps', which are simply contiguous groups of protrusions that are all on the same side of the boundary. These clumps form the patches to be tested. The log likelihood of each patch is the sum of the log likelihoods of the protrusions that form the patch.

3.4.5.2 Calculating the log likelihood ratio statistic

Suppose a patch P is to be tested, and that it belongs to region 1. The null hypothesis H_0 is that the pixels in P were generated from the distribution described by $p_1(x)$, and the alternative hypothesis H_A is that they were generated from the distribution described by $p_2(x)$, where $p_1(x)$ and $p_2(x)$ are estimates of the pixel intensity distributions for regions 1 and 2. The log likelihood ratio statistic for this test is

$$l = \ln\left(\frac{L_0}{L_A}\right) \tag{3.15a}$$

$$= \sum_{(i,j)\in P} \ln(p_1(I_{ij})) - \sum_{(i,j)\in P} \ln(p_2(I_{ij})) \tag{3.15b}$$

$$= \sum_{(i,j)\in P} \lambda_{ij} \tag{3.15c}$$

This statistic is therefore just the log likelihood for the patch. If the patch belongs to region 2, the expression for l is the negative of that shown above.

At this point, the pixels are still assumed to be independent. The correlation between pixels is taken into account later, in setting the threshold for rejection of H_0.

3.4.5.3 Setting the threshold

At this stage, we have a patch P currently assigned to region i (where $i = 1$ or 2), and its log likelihood ratio statistic l. The likelihood ratio test rejects the null hypothesis, thus declaring the segment to be inconsistent with the image data, if $l < K$, where K is a threshold set so as to give the desired false alarm rate for the test. In

general, setting this threshold requires knowledge of the distribution of l when the null hypothesis is true. This distribution is rather difficult and time-consuming to evaluate, especially when the pixels are not independent.

For this reason, the distribution of l is assumed to be Gaussian. This assumption allows the threshold to be set on the basis of the mean μ and variance σ^2 of the distribution, because a Gaussian distribution is characterized completely by its mean and variance. These will depend on the shape and size of the patch and the correlation between pixels. The methods used for calculating them are detailed below.

In principle, the threshold is then set at a given number of standard deviations from the mean, but in practice, the likelihood ratio statistic is normalized according to the formula

$$\nu = -\frac{(l-\mu)}{\sigma} \tag{3.16}$$

The null hypothesis is rejected if ν is greater than a fixed threshold, here called `siglev` because it sets the significance level of the test. This threshold is independent of the size and shape of the patch being tested and the textures in the regions. The usual value is 5.0; section 5.1.3.2 discusses the choice of this value.

It remains to estimate the mean μ and variance σ^2 of the distribution of l under the assumption that H_0 is true. The mean is easy to calculate, since it is proportional to the number of pixels in the patch. The variance is more difficult, since the pixels are correlated. If they were independent, the variance would be proportional to the number of pixels. However, independence cannot be assumed here; assuming independence usually gives values of ν which are too large, because adjacent pixels are positively correlated in most textures.

Let μ_1 and μ_2 be the mean values of the λ-image in regions 1 and 2, respectively. Then

$$\mu_i = \sum_{x \in G} p_i(x)\lambda(x) \tag{3.17}$$

where G is the set of quantized grey levels. If the patch P is assigned to region 1, then the mean μ is given by $\mu = |P|\mu_1$, where $|P|$ is the number of pixels in P. If the patch is assigned to region 2, then $\mu = -|P|\mu_2$.

The log likelihood ratio l is the sum of pixels in the λ-image, and is thus the sum of a set of correlated random variables. The variance of such a sum may be calculated from the variances and covariances of the individual random variables. Suppose we

have N random variables x_i; their sum is then

$$S = \sum_{i=1}^{N} x_i \tag{3.18}$$

with variance

$$\mathrm{Var}(S) = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{Cov}(x_i, x_j) \tag{3.19}$$

where $\mathrm{Cov}(x_i, x_j)$ is the covariance of x_i and x_j (and therefore $\mathrm{Cov}(x_i, x_i) = \mathrm{Var}(x_i)$). We therefore require the covariances of pixels in the λ-image. Since the texture in each region is assumed to be homogeneous, the covariance of two pixels should depend only on the vector separating them, not on their position. That is,

$$\mathrm{Cov}(\lambda_{ij}, \lambda_{kl}) = R_n(i-k, j-l) \tag{3.20}$$

in region n, where R_n is the autocovariance function for the λ-image in region n. The variance of l for patch P is therefore given by

$$\sigma^2 = \sum_{(i,j)\in P}\sum_{(k,l)\in P} R_n(i-k, j-l) \tag{3.21}$$

These autocovariances are estimated in the initial phase of the algorithm according to the formula

$$R_n(i,j) = \frac{\sum_{(k,l)\in S_n,(k+i,j+l)\in S_n}(\lambda_{kl}-\mu_n)(\lambda_{k+i,j+l}-\mu_n)}{\sum_{(k,l)\in S_n,(k+i,j+l)\in S_n} 1} \tag{3.22}$$

where S_n is the area used for evaluating statistics for region n. They are estimated only for $0 \le i < M$ and $|j| < M$, where M is a parameter, elsewhere called `autodist`, which specifies the maximum scale of intensity variation which will be considered to be texture. This parameter is discussed further in section 5.1.3.2. It is not necessary to estimate $R_n(i,j)$ for negative values of i, because $R_n(i,j) = R_n(-i,-j)$.

Equation (3.21) can be rearranged as

$$\sigma^2 = \sum_{i=1-M}^{M-1}\sum_{j=1-M}^{M-1} N(i,j)R_n(i,j) \tag{3.23}$$

where N is defined by

$$N(i,j) = \sum_{(k,l)\in P,(k+i,j+l)\in P} 1 \tag{3.24}$$

This can be calculated efficiently from a run-length encoded representation of P.

3.4.5.4 Other applications of consistency test—confidence measures

The consistency test can be used to obtain information about the clarity and sharpness of the boundary, expressed in terms of confidence intervals on the boundary position. These confidence intervals are obtained by considering a range of positions for a given section of the boundary, and applying the consistency test at each position. This has been implemented for knots, for single segments, and for sections of segments. The confidence intervals represent information which is potentially useful to later processing stages. If the data are unclear, a broad range of positions may be acceptable, indicating that later processes have some freedom in hypothesizing new positions to facilitate interpretation.

Confidence contours. For a knot, a confidence contour is obtained which indicates the area of acceptable positions for the knot. The knot is displaced outwards along radial lines until one of the adjacent segments becomes inconsistent. The radial lines are spaced at equal angles, thus providing a specification of the contour in polar form. Examples of these contours are given in chapter 4. Figure 3.11(a) shows how the boundary is deformed as the knot is displaced.

Rather than simply stepping out in small steps along each radius until the boundary becomes inconsistent, a binary search procedure is actually used. This is much more efficient, but it makes the assumption that the area of acceptable positions is connected, that is, that there exists a radial distance such that all closer distances are acceptable, and all further distances are not. The initial phase of this procedure searches outwards for a distance which is inconsistent, by starting at a distance of 1 pixel and doubling it until an inconsistent distance is found. Thereafter, the distance half-way between the largest acceptable and smallest inconsistent distances is tested. If that distance is acceptable, the acceptable distance is updated, otherwise the inconsistent distance is updated. When the acceptable and inconsistent distances become acceptably close (i.e., within 0.1 pixels of each other), the former is taken as the radius of the confidence contour at that angle.

'Cornerness' measure. The confidence contour for a knot can represent rather a large amount of data. It would be extremely useful to be able to extract from the contour a 'cornerness' measure, that is, an indication of whether the knot represents

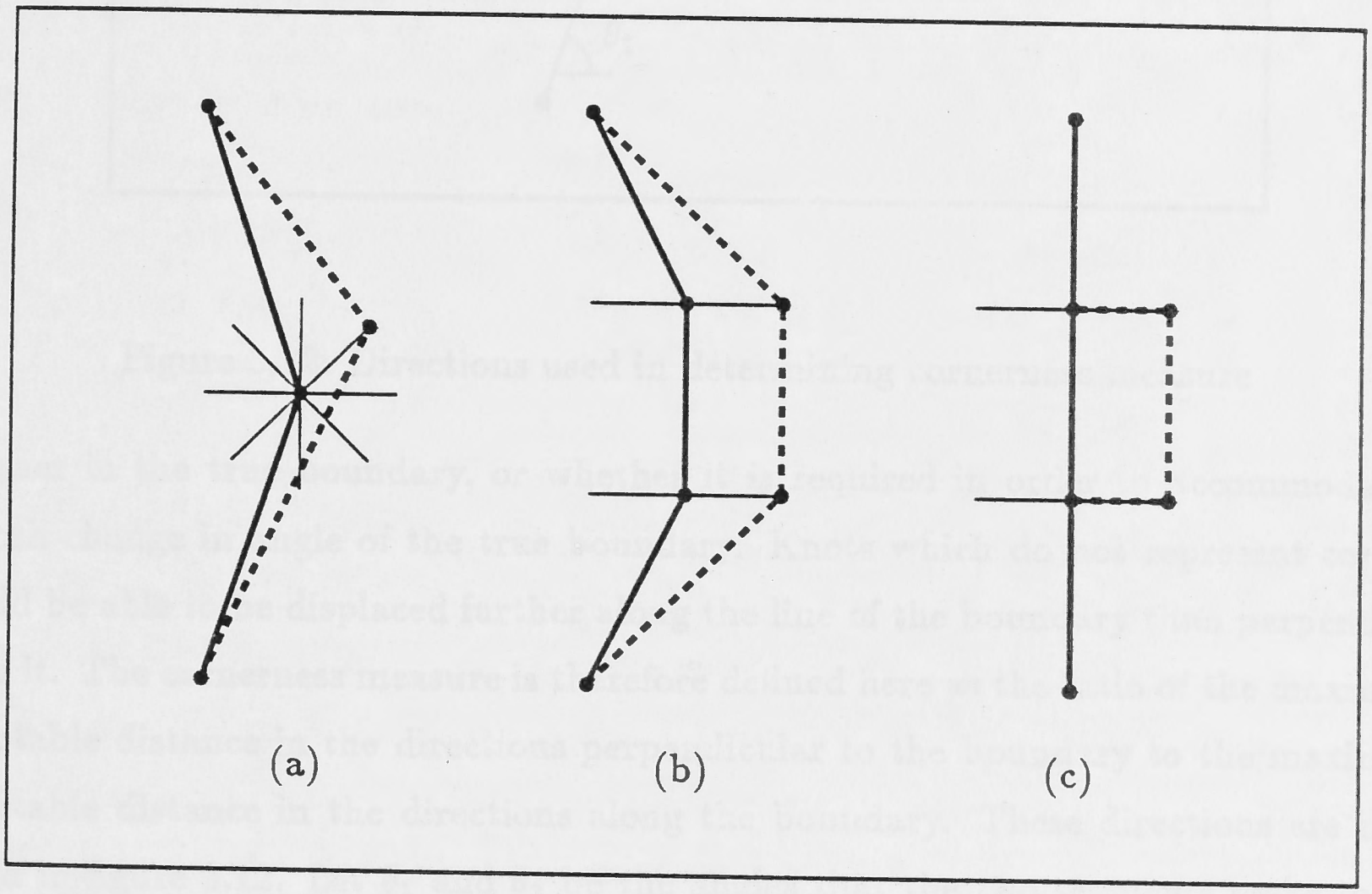


Figure 3.11: Knot movements for confidence contours and intervals: (a) knot confidence contour, (b) segment confidence interval, (c) confidence interval for section of segment.

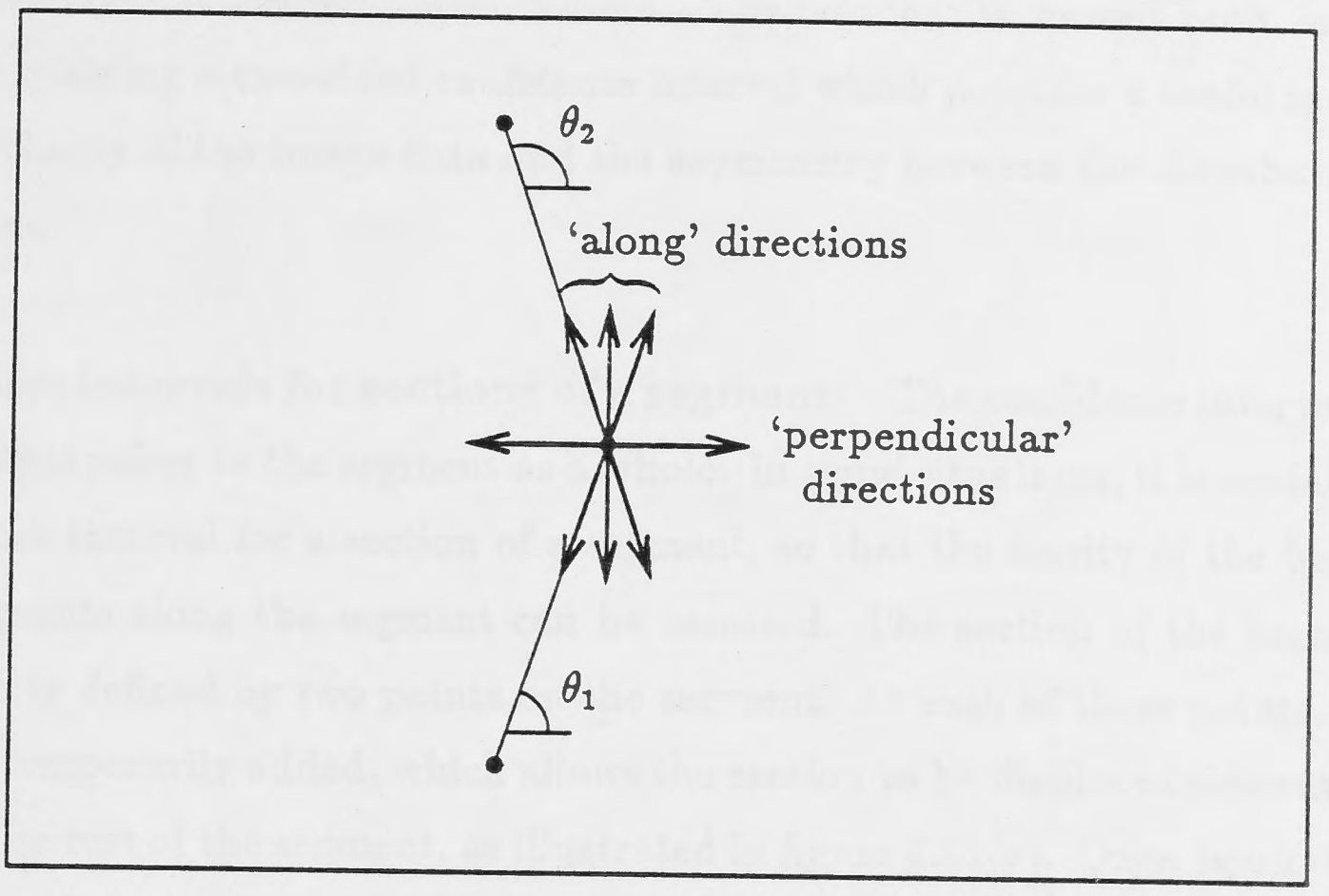


Figure 3.12: Directions used in determining cornerness measure

a corner in the true boundary, or whether it is required in order to accommodate a smooth change in angle of the true boundary. Knots which do not represent corners should be able to be displaced further along the line of the boundary than perpendicular to it. The cornerness measure is therefore defined here as the ratio of the maximum acceptable distance in the directions perpendicular to the boundary to the maximum acceptable distance in the directions along the boundary. These directions are illustrated in figure 3.12. Let θ_1 and θ_2 be the angles that the two segments make to the horizontal at the knot. The directions 'along' the boundary are then θ_1, $\theta_1 + \pi$, θ_2, $\theta_2 + \pi$, $(\theta_1 + \theta_2)/2$, and $(\theta_1 + \theta_2)/2 + \pi$. The directions perpendicular to the boundary are $(\theta_1 + \theta_2 + \pi)/2$ and $(\theta_1 + \theta_2 + 3\pi)/2$. Results obtained with this measure are given in chapter 4.

Confidence intervals for segments. The procedure for obtaining a confidence interval for the position of a segment moves the segment until either the segment or one of the two adjacent segments becomes inconsistent, as illustrated in figure 3.11(b). The segment is moved by displacing each endpoint by the same distance in the direction perpendicular to the segment at the endpoint. These directions will be identical for a linear segment, but different in general for a cubic segment. Once again, the binary

search procedure described above is used. The segment is moved both inwards and outwards, yielding a two-sided confidence interval which provides a useful indication of both the clarity of the image data and the asymmetry between the distributions in the two regions.

Confidence intervals for sections of a segment. The confidence interval obtained for a segment refers to the segment as a whole. In some situations, it is useful to obtain a confidence interval for a section of a segment, so that the clarity of the boundary at different points along the segment can be assessed. The section of the segment to be considered is defined by two points on the segment. At each of these points, two extra knots are temporarily added, which allows the section to be displaced sideways without affecting the rest of the segment, as illustrated in figure 3.11(c). Once again, the binary search procedure is used to determine the maximum acceptable displacement to each side, but in this case, both the displaced section and the two segments joining it to the original segment must be acceptable.

The confidence intervals and contours have been defined above for the movement of a single knot or segment, or section of a segment. There are many ways in which the boundary could be deformed, and confidence intervals could be obtained for any of them. Higher-level processes in an overall image interpretation system could therefore request that any given deformation could be tested in this manner. Such a facility should provide a very useful check to prevent these processes from hypothesizing interpretations which are contradicted by the image data.

3.4.6 Elaboration and simplification

The elaboration and simplification steps have only been implemented for piece-wise linear boundaries. The specific difficulties in implementing these steps when the option of using cubic segments exists are discussed in detail in section 5.1.3.3. The elaboration step is quite simple for piece-wise linear boundaries: a new knot is added at the middle of each inconsistent segment. This simple strategy gives good results, and is discussed further in section 5.1.3.3.

The algorithm used in the simplification step considers each knot in turn. If it passes an area test, described below, then the current state of the boundary is saved and the knot is removed; the two adjacent segments are replaced by a single linear

segment between the adjacent knots. The boundary is then re-optimized and tested for consistency. If it is not consistent, the previous state of the boundary is restored. The algorithm then proceeds to the next knot. The algorithm terminates once there are three or fewer knots or when all knots have been considered. The simplification step is discussed in more detail in section 5.1.3.4.

The area test is designed to detect quickly those knots which are likely to be essential in the boundary description. This test is based on the area enclosed by the triangle formed by the knot and the two adjacent knots. If the area is large, then the knot is most likely necessary in the boundary description, but if it is small, it may not be necessary, and it is a candidate for elimination. The threshold area is a settable parameter; its usual value is 200 pixels, but the value is not critical. Increasing this value will increase the computation time required. If it is too small, the boundary description obtained may in some cases be slightly more complex than required.

3.4.7 Split-and-merge procedure

The BLM algorithm requires sample patches for the two regions adjacent to the boundary of interest. These patches must be indicated by some process which detects the difference between the regions. In the current implementation, a split-and-merge algorithm is used for this purpose. The algorithm used here differs from the classical algorithm (Horowitz and Pavlidis, 1976) in the homogeneity test used. In keeping with the spirit of the BLM approach, the test used involves a likelihood ratio test of the hypothesis that each sub-region of interest belongs to the joint region. It also differs in that the 'merge' phase corresponds to the 'grouping' phase of Horowitz and Pavlidis, and their 'merge' phase is not used. The merge phase used here merges adjacent regions if they are sufficiently similar, and will thus eventually merge the four sub-blocks of a larger block if necessary; a separate step to merge these sub-blocks is not required.

The 'split' phase of the procedure begins with the image split into a pre-specified number of blocks (usually 16). Each block is recursively split until each sub-block is homogeneous or has reached a given minimum size (usually 16 pixels square). The initial block can be initially split to a given number of levels to reduce the processing time required, because the homogeneity test is slower on the larger blocks. The homogeneity test on a given block first estimates the distribution of pixels in the block. For each sub-block, it then tests the null hypothesis that the pixels in the sub-block were

generated by the distribution estimated for the whole block, against the alternative that they were generated by the distribution estimated from the sub-block alone. If the null hypothesis can be rejected for any sub-block, the block is split.

Two different forms of this test were used, corresponding to different assumptions about the distribution of pixel intensities. The first form used essentially the same test described above for the consistency test. That is, the pixel intensity distributions in the two regions were estimated from their histograms, autocovariances were calculated, and a normalized log likelihood ratio was obtained and compared with a threshold. This form of the test was rather slow, because the autocovariances have to be calculated for each sub-block, since they depend on the function $\lambda(x)$ and therefore on the histograms. In addition, the results of the test were sometimes counter-intuitive, because no account was taken of the brightness difference between pixels which fall into different bins of the histograms.

Consequently, a simpler form of the test was used, based on the assumption that the distribution of pixel intensities was Gaussian. The null hypothesis was rejected if the mean of the sub-block was outside the range expected from the mean and autocovariance of the pixels in the larger block, that is, if

$$\nu = \left| \frac{\mu_S - \mu}{\sigma} \right| > \texttt{siglev} \tag{3.25}$$

where μ_S is the mean pixel intensity in the sub-block, μ is the mean intensity in the larger block, and σ^2 is the variance expected for the mean of the sub-block, given the autocovariances of the pixels in the larger block. This test is faster than the one described previously (because the autocovariances for the larger block need only be calculated once), and it also gives results which are more in accordance with human perception.

The homogeneity test has two important parameters. The first is `siglev`, which was set at the relatively low value of 2.0 (standard deviations from the mean) in the split phase, to ensure that blocks were split when required; failing to split a block is a more serious error than splitting it unnecessarily, since the latter can be corrected in the merge phase. A higher value is used in the merge phase. The second parameter is `autodist`, which controls the scale of intensity variation which is considered texture. Values of between 2 and 4 were appropriate for the textures considered. Some problems were encountered in estimating the autocovariances for small blocks. When the

width of a block was less than approximately four times `autodist`, the estimates could be unreliable, leading to variance estimates which were too small, and in some cases negative. Therefore, the value of `autodist` was dynamically reduced to be at most one quarter of the width of the block being tested.

The merge phase considers pairs of adjacent regions, and merges them if both regions could belong to the joint region. Therefore, the statistics of the joint region are estimated, and the test described above is applied to each region, to test the null hypothesis that the pixels in the region were generated by the distribution estimated for the joint region. If the null hypothesis is accepted for both regions, the two regions are merged. A higher threshold for rejection of the test is used in the merge phase than that used in the split phase, to ensure that regions can be merged when appropriate. A value of 5.0 (standard deviations from the mean) gave good results.

The merging procedure starts with each block forming a separate region. It then considers each block in turn, and attempts to merge it with each block that adjoins on the right-hand side or below, unless they already belong to the same region. However, before merging any pair of blocks, it attempts to merge the second block with any blocks that adjoin it, using a lower threshold. The threshold is set to the value of ν obtained from the first two regions. Thus, the second block will only be merged with a third if the distribution of the second block is closer to that of the third than that of the first. This proceeds recursively, attempting to merge the third with blocks adjacent to it, using an even lower threshold, and so on. If the second block does in fact merge with any adjoining regions, it is necessary to recalculate ν for the first and second regions; if it is still less than the threshold, the two regions are then merged. This procedure tends to merge regions which are very similar early in the process, and thus creates larger regions for the later tests, which are then more reliable. It gave better results than the more straightforward procedure of merging two regions as soon as they pass the test.

This formulation of the split-and-merge algorithm has some interesting features. The first is that it explicitly takes account of the distribution of pixel intensities in a texture, and of the correlation between pixels. The values of the thresholds for splitting and merging are therefore largely independent of the textures considered. The other interesting feature is the merging strategy, which gave results which are less dependent on the order in which the blocks are considered than the straight-forward strategy.

Typical results from this procedure are given in chapter 4.

Chapter 4

Results

Previous chapters have demonstrated via examples that convolution edge detectors give inaccurate and confused results when applied to textured imagery. Further examples are given here of the results obtained with the Marr-Hildreth and Canny edge detectors, together with a quantitative evaluation of their accuracy. The consistency test described in the previous chapter is applied to the results from the convolution edge detectors, and as might be expected, it shows that the boundary descriptions obtained with these detectors are often inconsistent with the image data.

In contrast, the BLM algorithm produces excellent results on a variety of images. Results are shown for nine artificial images and four real-world images, including some which do not conform to the assumptions made in the current implementation of the BLM algorithm. These results are generally in very good agreement with human perception, and also demonstrate some interesting characteristics of the BLM algorithm.

The artificially generated images used have a single boundary between two homogeneous textured regions. The textures in the regions were natural textures, obtained from the set of digitized Brodatz textures (Brodatz, 1966) in the image data base distributed by the University of Southern California Image Processing Institute (Weber, 1983). Use of these artificially generated images has the advantage that the correct boundary position is known, so the accuracy of the reported edge position can easily be assessed. These images generally have a clear, sharp boundary which humans can locate accurately. The real-world images chosen each have a boundary between extended regions of texture. They include some images which were digitized from negatives during the course of this study, and also some images from the USCIPI data base. Each image has a descriptive name, which is shown in a 'typewriter' font (e.g.,

gmcorner). The name is intended to be descriptive; for example gmcorner comes from 'Grass Multiplied (by half) with a Corner'.

4.1 Evaluation of edge detector results

The two edge detectors which have been implemented, namely the Marr-Hildreth and Canny edge detectors, were described previously in chapter 2. Both edge detectors provide as output a map of edge pixels and their strengths. Many of the problems associated with the use of these edge detectors can be seen simply by a visual inspection of these maps, and comparison with the original image. The deficiencies which are visually obvious are confirmed by the quantitative measures of accuracy. Two test images are used, one artificial image (gmcorner) and one real-world image (rock). One further real-world image (lenna) is used as a visual check that the edge detectors have been correctly implemented.

In most cases, these edge maps are displayed as an image in which the edge pixels are shown as coloured points overlayed on the original image. This representation allows comparison of the location of the edge pixels with the true edge (or at least, the edge perceived by human observers). The edge maps must be thresholded for display in this form; the threshold has in each case been set manually at a level which gives the best results, suppressing the responses to texture as far as possible without causing the edge string of interest to break up.

In some cases, the edge maps are also shown in the form of a grey-scale image, in which edge pixels have an intensity proportional to their strength. No thresholding is required with this representation, but for display purposes the range of edge strengths has been truncated at the high end so that the weaker responses are visible. With this representation, the strength of the edge string of interest can be compared with the strength of responses to textural variations.

4.1.1 Quantitative tests of accuracy

Three quantitative tests are described below, which test the accuracy with which the boundary has been located. These are:

1. Number of misclassified pixels
2. Mean absolute error
3. Consistency

These tests are appropriate when a particular boundary of interest can be identified. Tests 1 and 2 above require knowledge of the true boundary, and are applied to the results from image **gmcorner**. The consistency test can be applied with or without knowledge of the true boundary, provided that appropriate regions on each side of the boundary can be identified for evaluating the necessary statistics.

4.1.1.1 Tracking edges and obtaining a segmented image

All three tests require that the boundary be represented in the form of a region mask (i.e., a segmented image). Deriving this representation from the edge map proceeds in three steps:

(a) Isolate the edge pixels corresponding to the boundary of interest,

(b) link them together to form a continuous chain, and

(c) form a region mask by determining, for each pixel in the image, which side of the chain it is on.

In the case of the artificial images, it might be expected that the edge pixels corresponding to the boundary of interest will be much stronger than all the others, and could therefore be isolated with a thresholding operation (possibly with hysteresis, as suggested by Canny (1986)). However, this is not generally true, especially at the smaller scales, as inspection of figures 4.2 and 4.12 shows. Therefore, where necessary, the string of edge pixels closest to the true boundary has been identified manually. This bypasses the inadequacy of the edge detectors in indicating the difference between a globally significant edge, and an edge due to textural variations, thus favouring the edge detectors.

Once a string of edge pixels has been isolated by thresholding and manual selection (if necessary), the edge pixels are linked together by a relatively simple procedure. This

involves starting at one end, and proceeding by stepping to neighbouring pixels until the other end is reached, thus producing a chain-code representation of the boundary. Gaps are bridged by searching out in squares of increasing size until another edge pixel is reached. The maximum size of these squares is limited, so that the edge tracker does not jump across to other edge strings. If the end of the string is sufficiently close to the beginning, the chain is closed by bridging from the end to the beginning.

Converting the chain-coded boundary so obtained to a region mask requires that the boundary be closed. If the boundary is not closed, as in the case of images where the boundary extends from one side of the image to the other, then it is arbitrarily closed by manually connecting the ends with straight lines to one of the margins of the image. The procedure for constructing the region mask looks at the number of times that the chain crosses the scan-line to the right of a given pixel. The rule is that the pixel is 'inside' if the number of times the chain crosses the scan-line to the right of the pixel, going upwards, differs from the number of times that it crosses going downwards. An arbitrary decision is made about the edge pixels, by displacing the chain a small amount upwards, and a smaller amount to the left (in much the same manner as described for the parametric boundaries in section 3.4.4.1).

Where the true boundary extends to the borders of the image, the string of edge pixels reported will usually not extend to the borders of the image, due to the problems that the edge detectors have in the regions of the image where the convolution mask extends beyond the borders. In such cases, the evaluation techniques are restricted to considering the range of rows or columns spanned by the string of edge pixels, thus bypassing this deficiency of the edge detectors.

4.1.1.2 Description of quantitative tests

For the artificial image `gmcorner`, it is possible to compare the region mask obtained as described above with the correct result, which is available in the form of the mask used to generate the image. The simplest comparison is to count the number of pixels classified differently. This can then be divided by the length of the boundary to give the mean absolute error (i.e., the average distance of the reported boundary from the true boundary).

Just as the results from the edge detectors do not satisfy the accuracy and simplicity criteria, they also do not satisfy the consistency criterion. Applying the consistency

test requires that patches be identified for testing. These patches are connected groups of pixels which are all on one side of the reported boundary. Two methods have been used to obtain these patches. The first method takes connected groups of pixels which are classified differently by the correct and reported boundaries. Each such group is tested as described in section 3.4.5.

The second method does not require knowledge of the correct result. Instead, the reported boundary is compared with a more likely boundary. This involves finding patches on each side of the boundary which are more likely to belong on the other side. This is done using a procedure which attempts to maximize the likelihood of the reported mask by iteratively re-classifying pixels that have a neighbour in the other region. Pixels are only re-classified if they are more likely to be in the other region, and if re-classifying them could not split either region into two pieces. This procedure is iterated until no more pixels can be re-classified. It will generally increase the likelihood substantially, but it is not guaranteed to find the global maximum of the likelihood. As before, connected groups of pixels which are classified differently in the reported and optimized masks form the patches for testing.

As this test does not require knowledge of the true boundary, it can be applied to results obtained from real-world images. Thus, the accuracy of an edge detector can be evaluated *without* knowing the true boundary. This is in contrast to previous methods for evaluating accuracy, which require knowledge of the true boundary (e.g., Abdou and Pratt, 1979).

4.1.1.3 Obtaining statistical characterizations

The consistency test requires statistical information about the regions adjacent to the boundary of interest. Thus, representative patches of these regions must be identified. These patches are used for evaluating the statistics of the two regions. They are obtained by eroding the two regions in the reported mask, as described in section 3.4.3. As noted there, the reason for eroding the regions is that the boundary is likely to be somewhat inaccurate, and there may therefore be pixels near the edge of each region which actually should belong to the other region. The eroded regions are then used for the extraction of the required statistical information, as described in sections 3.4.3 and 3.4.5.3.

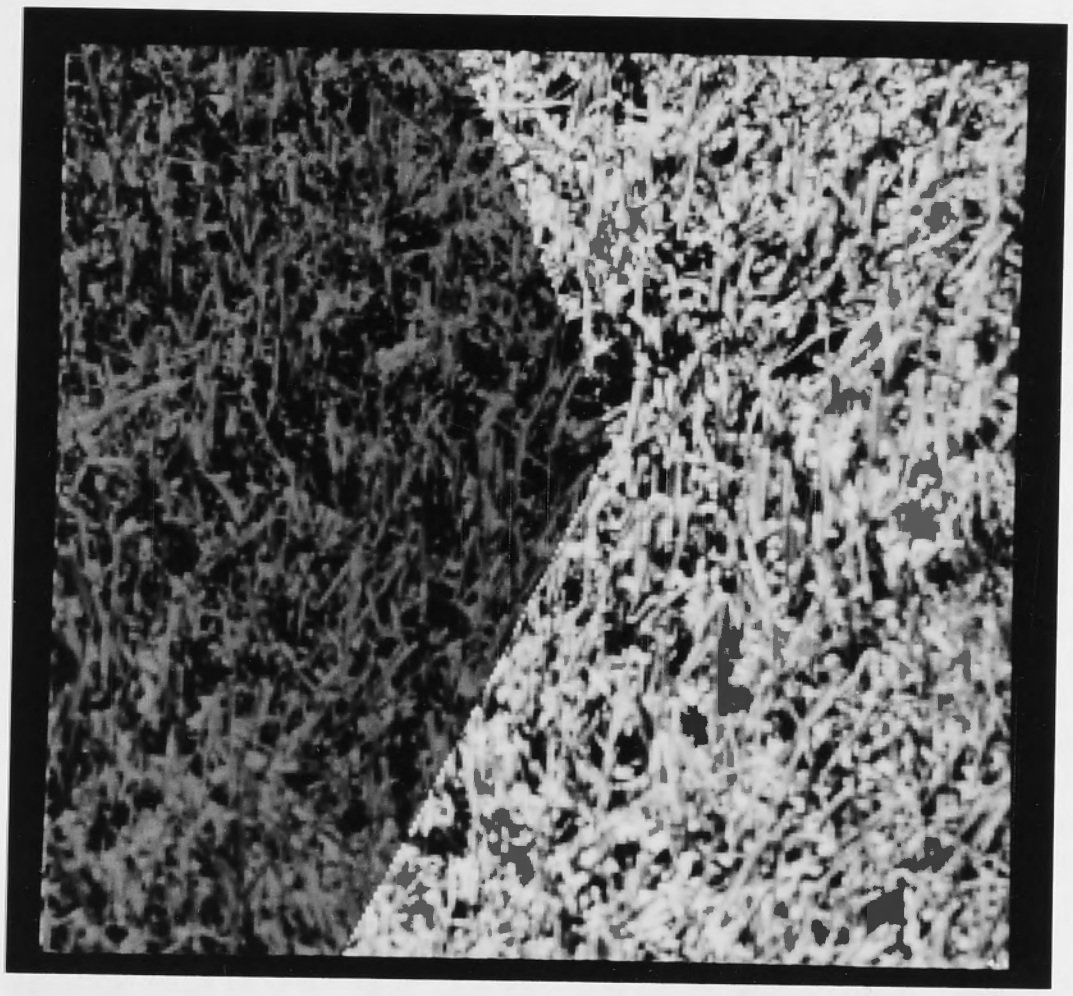


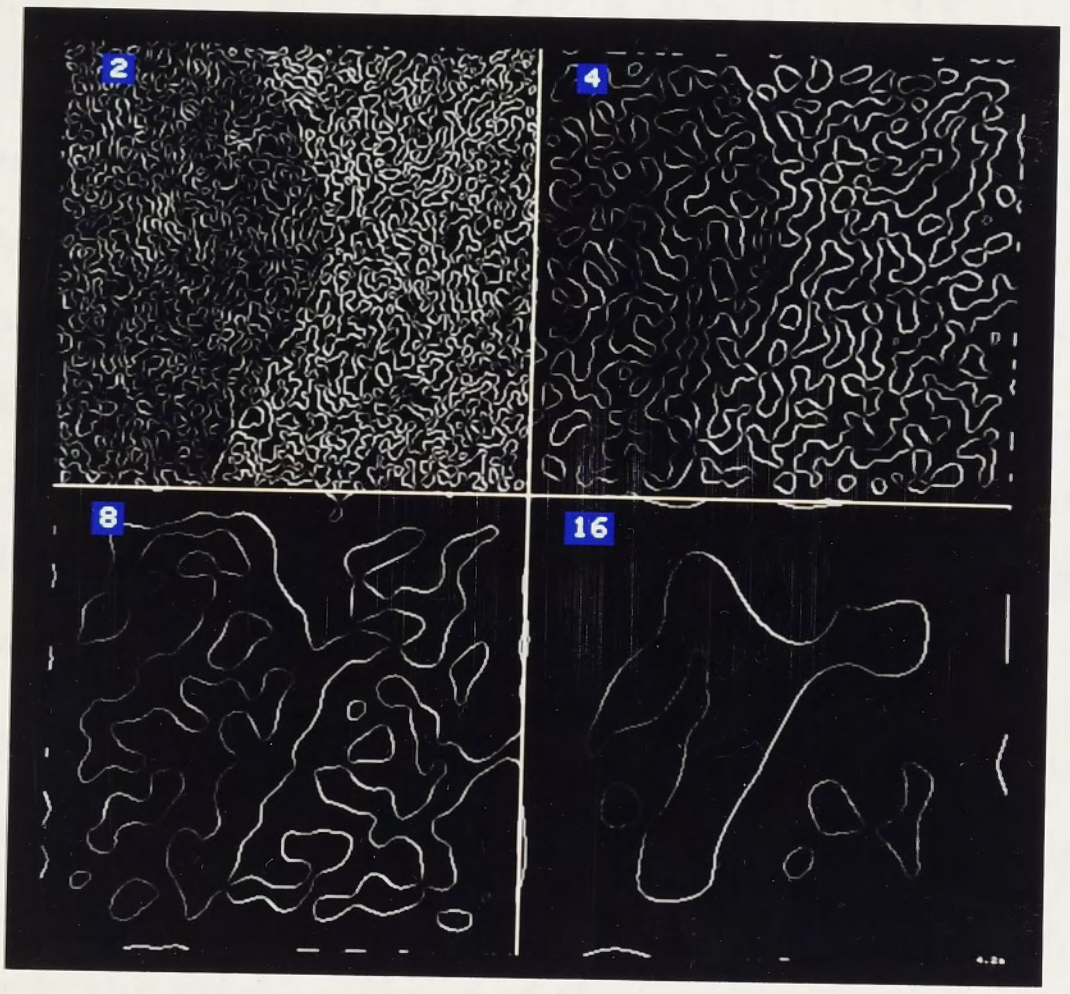
Figure 4.1: Image gmcorner

4.1.2 The Marr-Hildreth edge detector

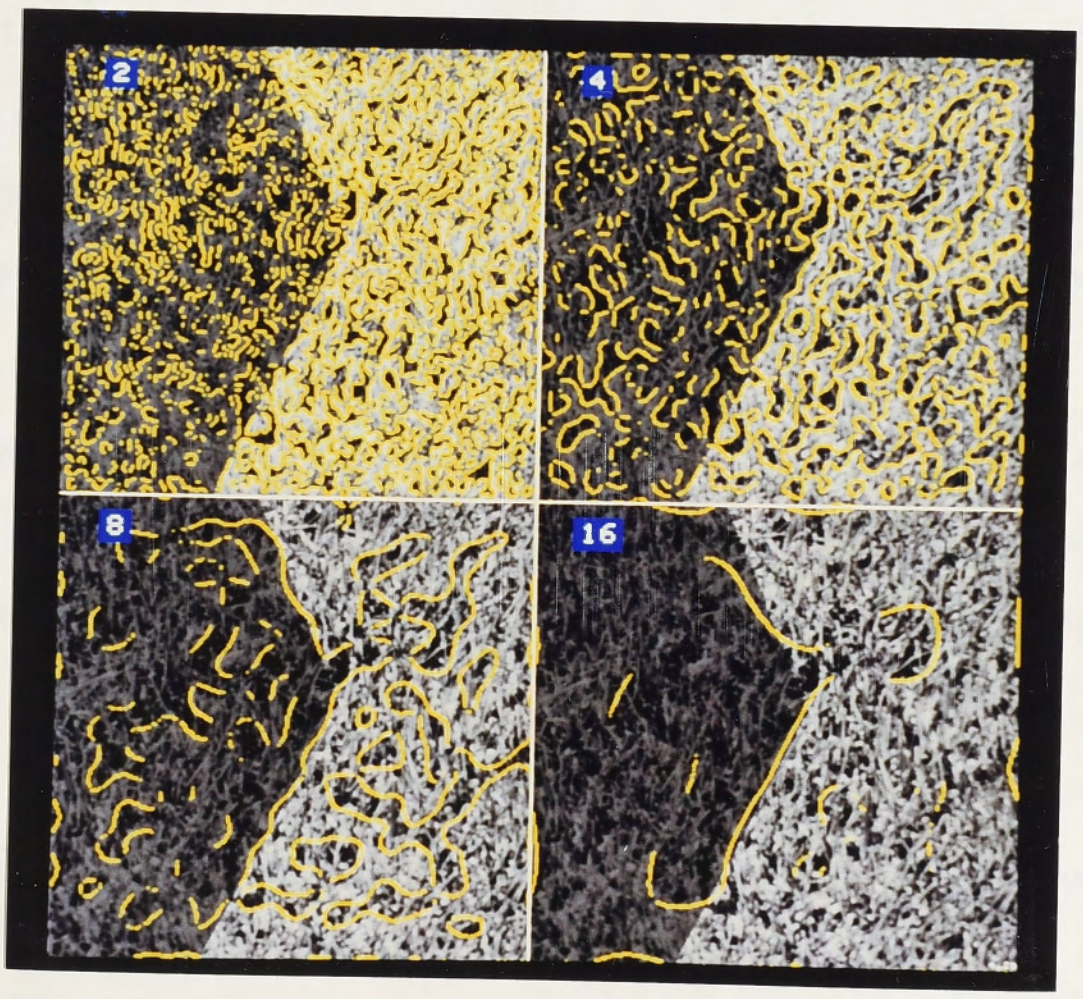
4.1.2.1 Image gmcorner

Figure 4.2 shows the results obtained with the Marr-Hildreth edge detector, at four different scales, from the image **gmcorner** shown in figure 4.1. This image was constructed from a uniform grass texture by halving the intensity of the pixels to the left of the boundary. The scales were $\sigma = 2$, 4, 8, and 16 pixels, as shown in the figures. Figure 4.2(a) shows the edge strengths in grey-scale form, while figure 4.2(b) shows the thresholded edges overlayed on the original image. Only at the largest scale is the boundary of interest unambiguously marked. At smaller scales, the deviation of the reported boundary from the true boundary is generally smaller, but the boundary of interest is weaker relative to the textural responses, and it also tends to break up where there are dark and light clumps adjacent to the boundary. Given the problems observed over the range of scales considered, it is unlikely that substantially better results would be obtained at any scale.

The original Marr-Hildreth theory (Marr and Hildreth, 1980) proposed that important edges in the image would be signalled by a coincidence of zero-crossings over a



(a) edges in grey-scale form



(b) edges overlaid on image

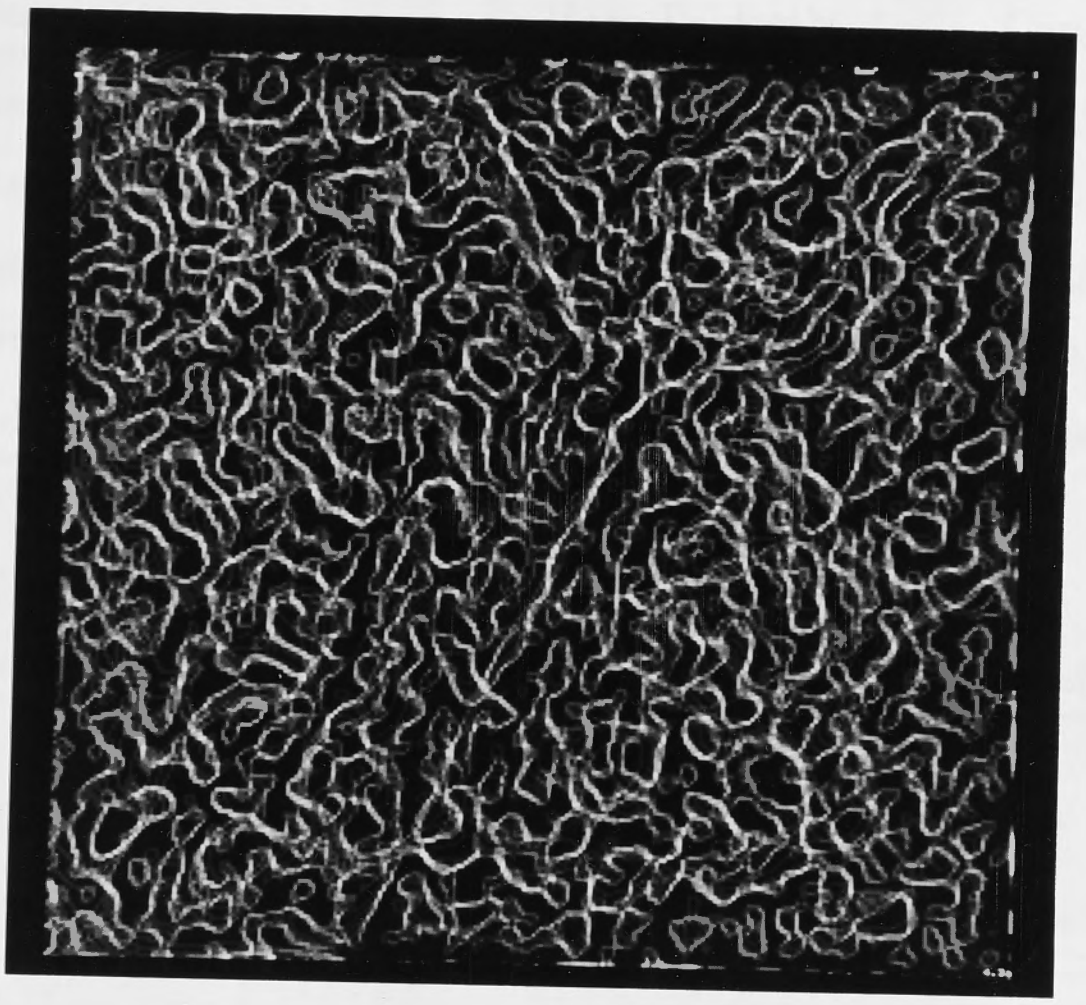
Figure 4.2: Marr-Hildreth edge detector results on image gmcorner

range of scales. To test this idea in the case of textured images, the Marr-Hildreth edge detector was applied at 13 scales between $\sigma = 2$ and $\sigma = 16$ at quarter-octave intervals (σ =2, 2.4, 2.8, 3.4, 4, 4.8, 5.7, 6.7, 8, 9.5, 11.3, 13.5, and 16). The image shown in figure 4.3(a) shows the coincidence of zero-crossings: each pixel has brightness proportional to the number of zero-crossings at that pixel. The boundary of interest is not at all clearly marked. Figure 4.3(b) shows the result obtained when only the largest seven scales were used; it is no better. The larger scales were used because the smaller scales respond to the texture elements as strongly as to the boundary of interest, and could therefore be interfering with the larger scales reporting the boundary of interest. However, it is evident that the proposed coincidence of zero-crossings does not occur even at the larger scales.

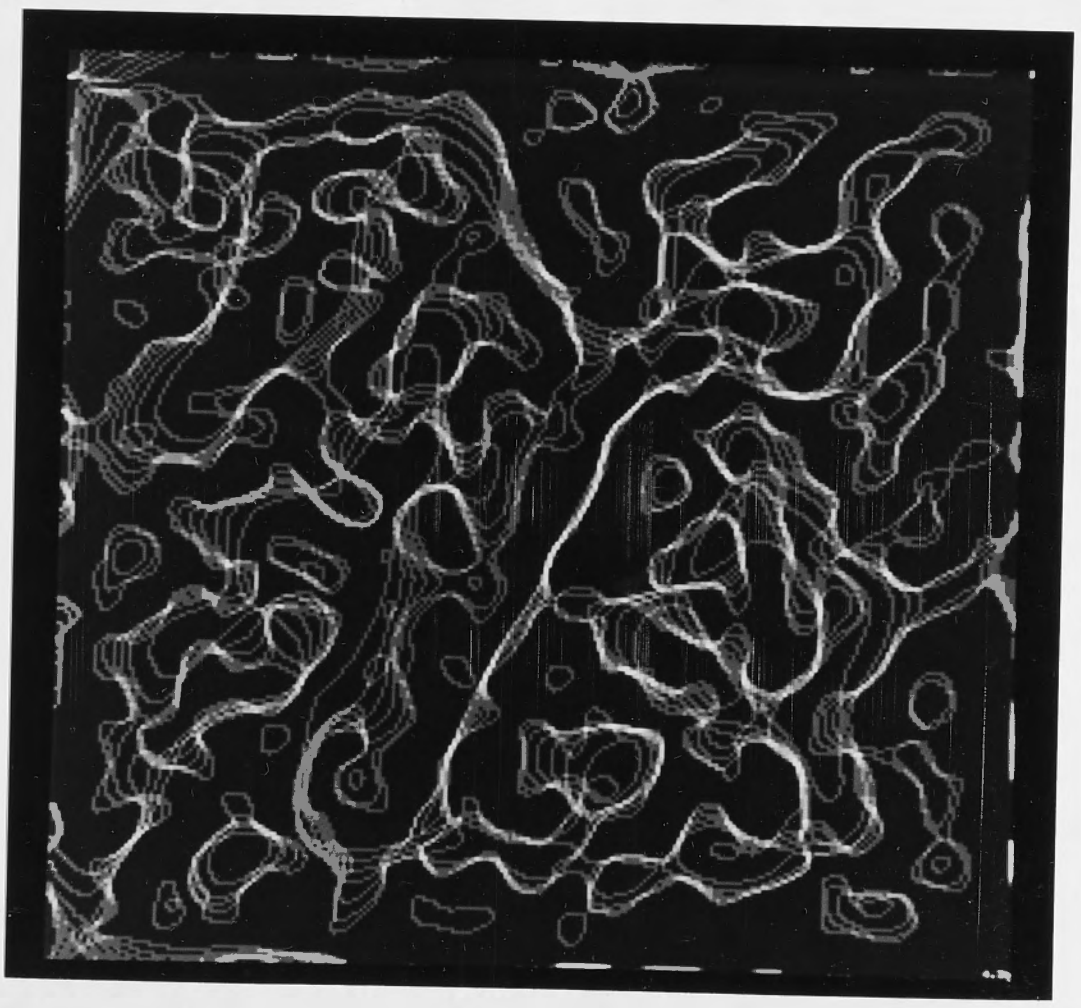
The scale $\sigma = 8$ was chosen for further quantitative evaluation, as being the scale which gave the best accuracy while still reporting the boundary of interest relatively clearly. The larger scale $\sigma = 16$ reports the main boundary clearly, but is quite inaccurate, especially in the neighbourhood of the corner. On the other hand, with the smaller scales, the boundary of interest is almost impossible to distinguish from the responses to textural detail.

The boundary in the output for $\sigma = 8$ was then selected manually and converted to a region mask in the manner described above. The edge string is shown in figure 4.4, which also shows the outlines of the eroded regions used for evaluation of statistics. Notice particularly that the edge string has been truncated where it turns away from the true boundary in the vicinity of the corner. A square of width 25 pixels was used for the erosion.

Table 4.1 shows the quantitative evaluation of the accuracy of both edge detectors on image `gmcorner`. The length of the true boundary is different for the two detectors because the edge pixel strings spanned different sets of scan-lines. (Note that all images are displayed with line 1 at the top.) This length is the sum of the distances from the corner to the points of intersection of the true boundary with the first and last scan-lines. The rows in the table labelled 'No. of patches failing test' show how many of the patches tested were found to be clearly on the wrong side of the boundary, and the rows labelled 'Maximum ν' give the normalized log likelihood ratio for the worst patch. The thresholds were 5.0 for the comparison with the correct mask, and 10.0 for the optimized mask. Section 5.1.3.2 discusses the choice of these thresholds. Patches



(a) 13 scales



(b) 7 scales

Figure 4.3: Coincidence of zero-crossings for image `gmcorner`

Edge detector	Marr-Hildreth	Canny
Scale σ (pixels)	8	13.6
Scan-lines processed	13–238	19–236
Area misclassified (pixels)	549	629
Length of true boundary (pixels)	246.8	238.2
Mean error distance (pixels)	2.22	2.64
Comparison with correct mask:		
No. of patches failing test	5	4
Maximum ν	30.3	35.6
Comparison with optimized mask:		
No. of patches failing test	7	7
Maximum ν	31.4	53.5

Table 4.1: Quantitative evaluation of edge detector results on image gmcorner

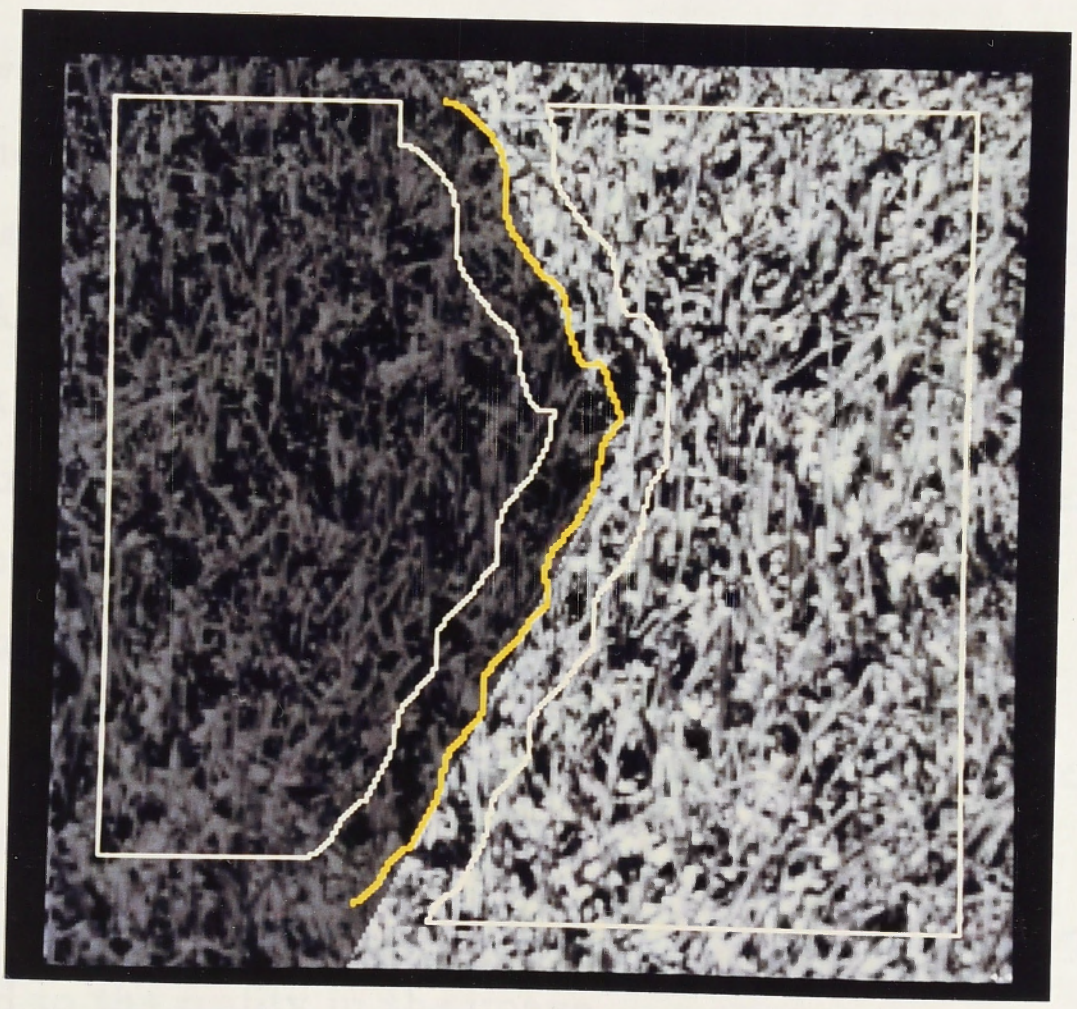


Figure 4.4: Selected edge string and patches for statistics for image gmcorner

which passed the test are not indicated in the table.

Figures 4.5 and 4.6 show where the reported mask for the Marr-Hildreth edge detector differs from the correct and optimized masks, respectively. The thin yellow line shows the position of the reported boundary. The colours of the patches indicate whether the patch failed the consistency test: red and magenta indicate fail, and blue and cyan indicate pass. Notice that where the reported boundary is to the left of the true boundary, even quite large patches can pass, but where the reported boundary is to the right, the test is much more sensitive. This is a consequence of the asymmetry in the distributions. The results shown in table 4.1 confirm the visual impression that the edge detector output is inaccurate. The reported boundary fails the consistency test quite badly—values of around 30.0 for ν indicate patches which are most definitely wrong.

The results given in table 4.1 indicate that the Canny edge detector is not as accurate as the Marr-Hildreth. This is probably due in part to the different scales used. However, the intention here is not to compare the two edge detectors, but rather to indicate that they both have serious deficiencies when used on textured images.

4.1.2.2 Image rock

Figure 4.7 shows image rock, which portrays a rock in front of some vegetation. The results from the Marr-Hildreth edge detector are shown in figure 4.8. Once again, the scale $\sigma = 8$ was chosen for quantitative evaluation. Figure 4.9 shows the selected edge pixel string. Attention was restricted to the left-hand portion of the boundary of the rock by placing a bounding rectangle on the image, also shown in figure 4.9. This portion of the boundary has relatively homogeneous regions on either side. Table 4.2 and figure 4.10 show the comparison of the reported mask with the optimized mask (no knowledge of the true boundary is assumed with this image).

This image was included to demonstrate that the problems which arise with the artificial images can be expected to arise also in real-world images. Once again, the results shown in table 4.2 confirm the visual impression that the reported boundary does not correspond to the reality in the image.

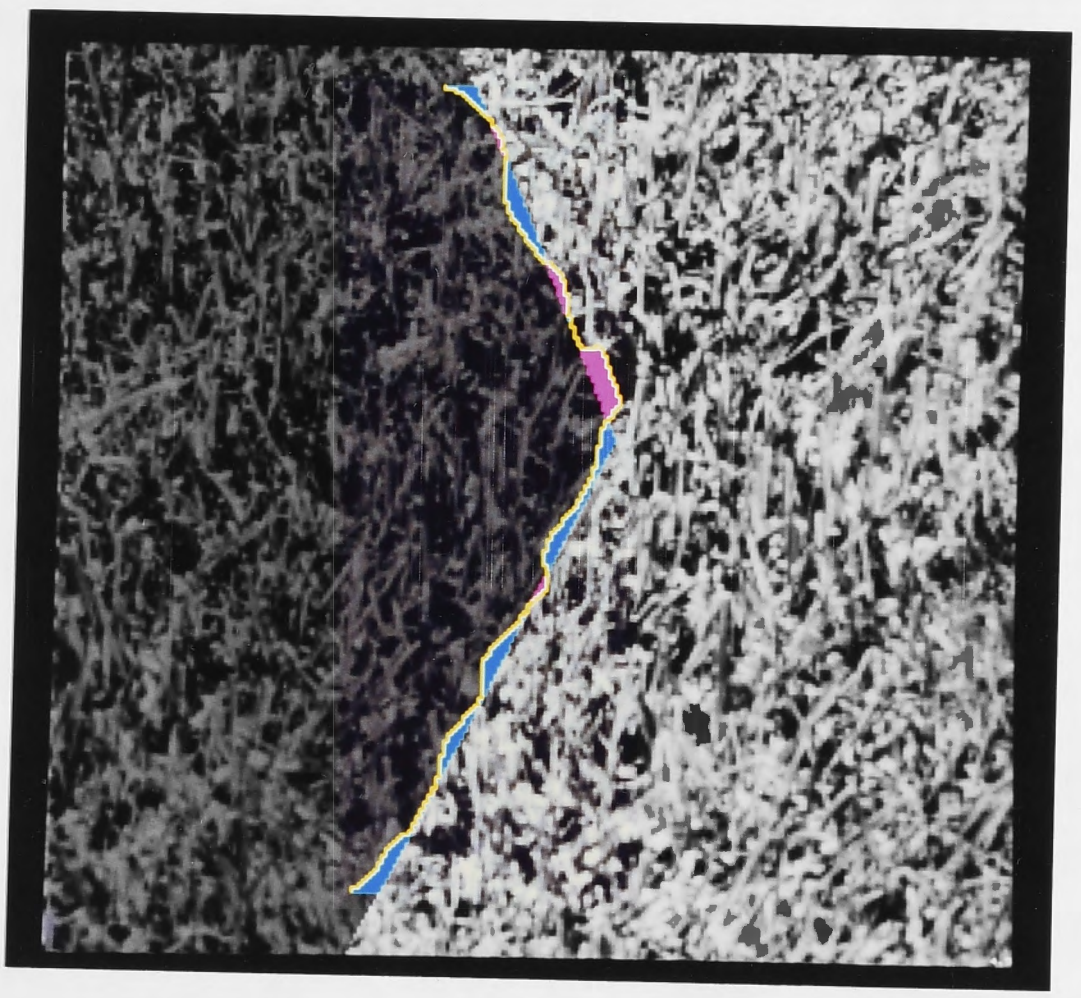


Figure 4.5: Comparison of Marr-Hildreth and correct results for image gmcorner

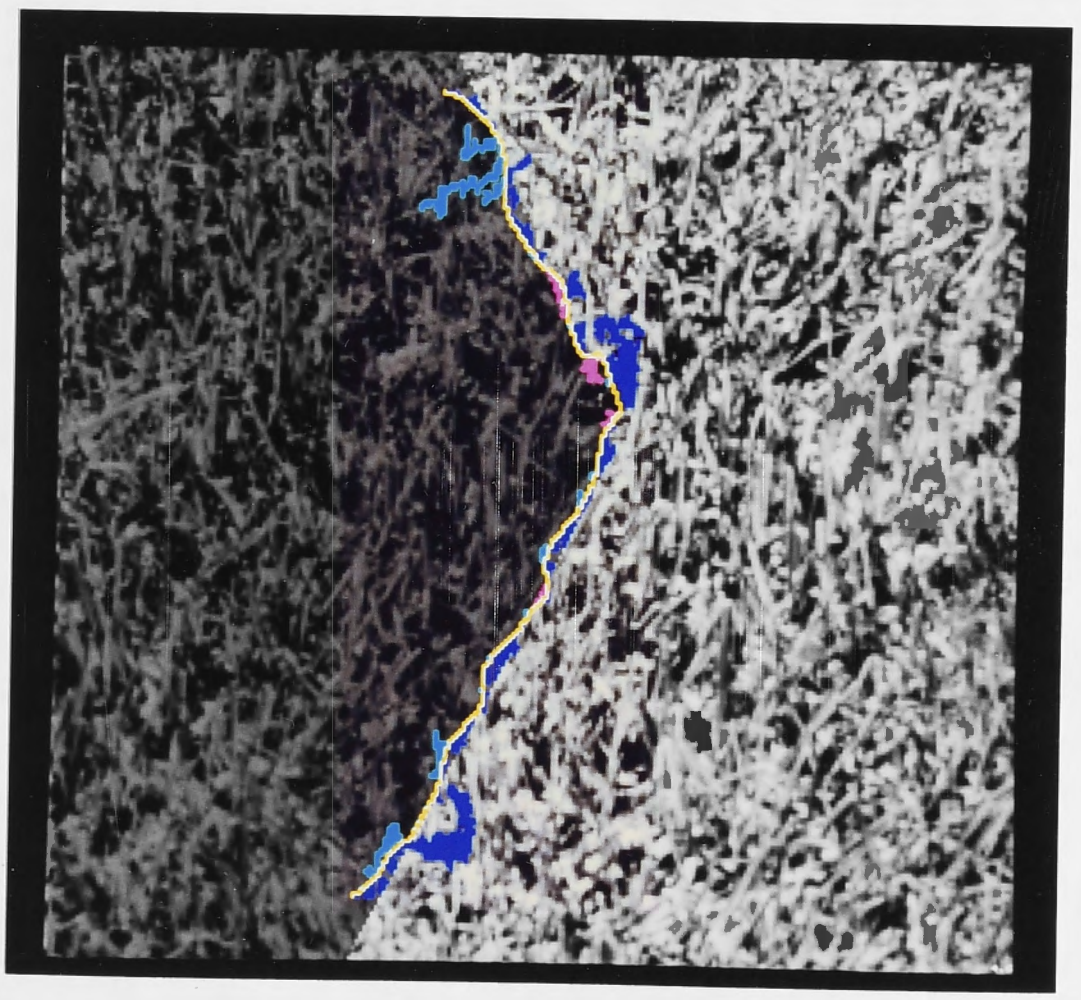


Figure 4.6: Comparison of Marr-Hildreth and optimized results for image gmcorner

Figure 4.7: Image rock

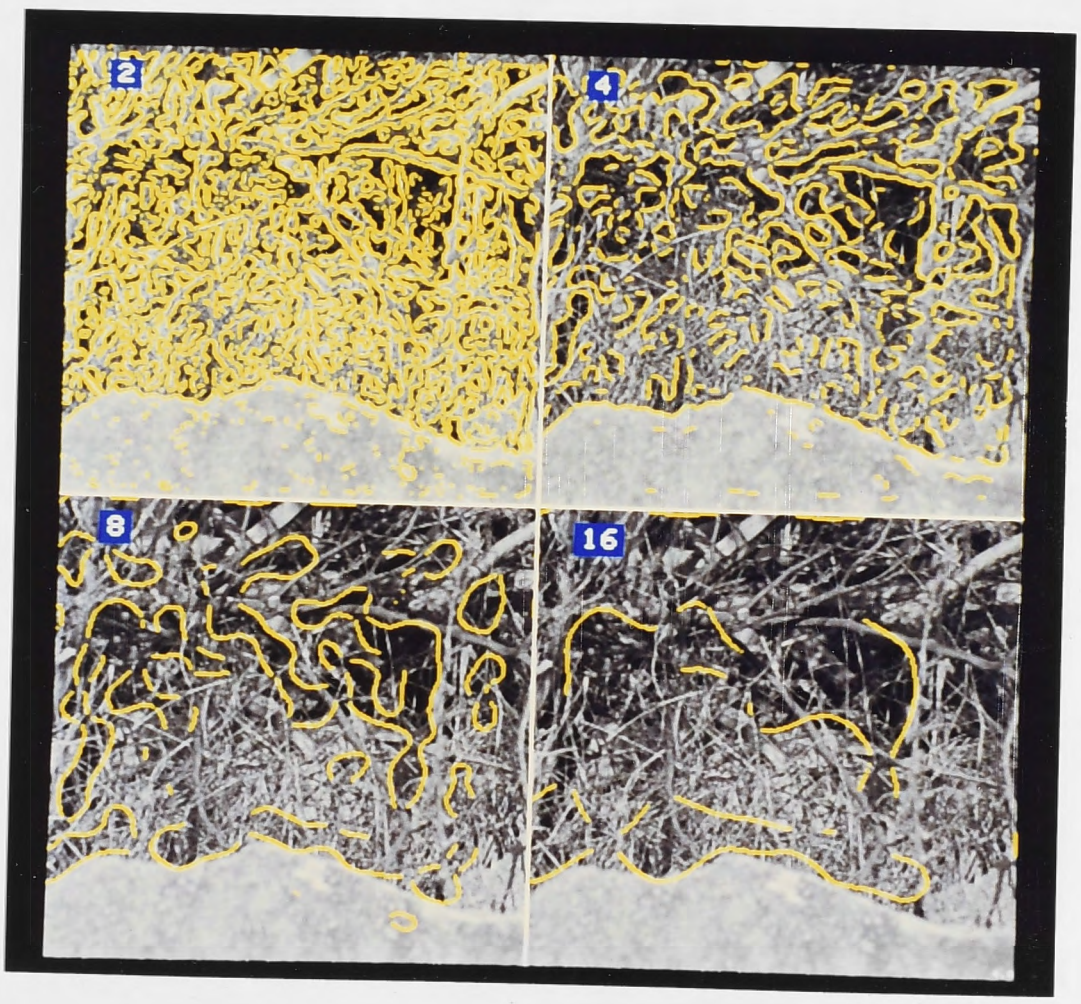


Figure 4.8: Results from Marr-Hildreth edge detector for image rock

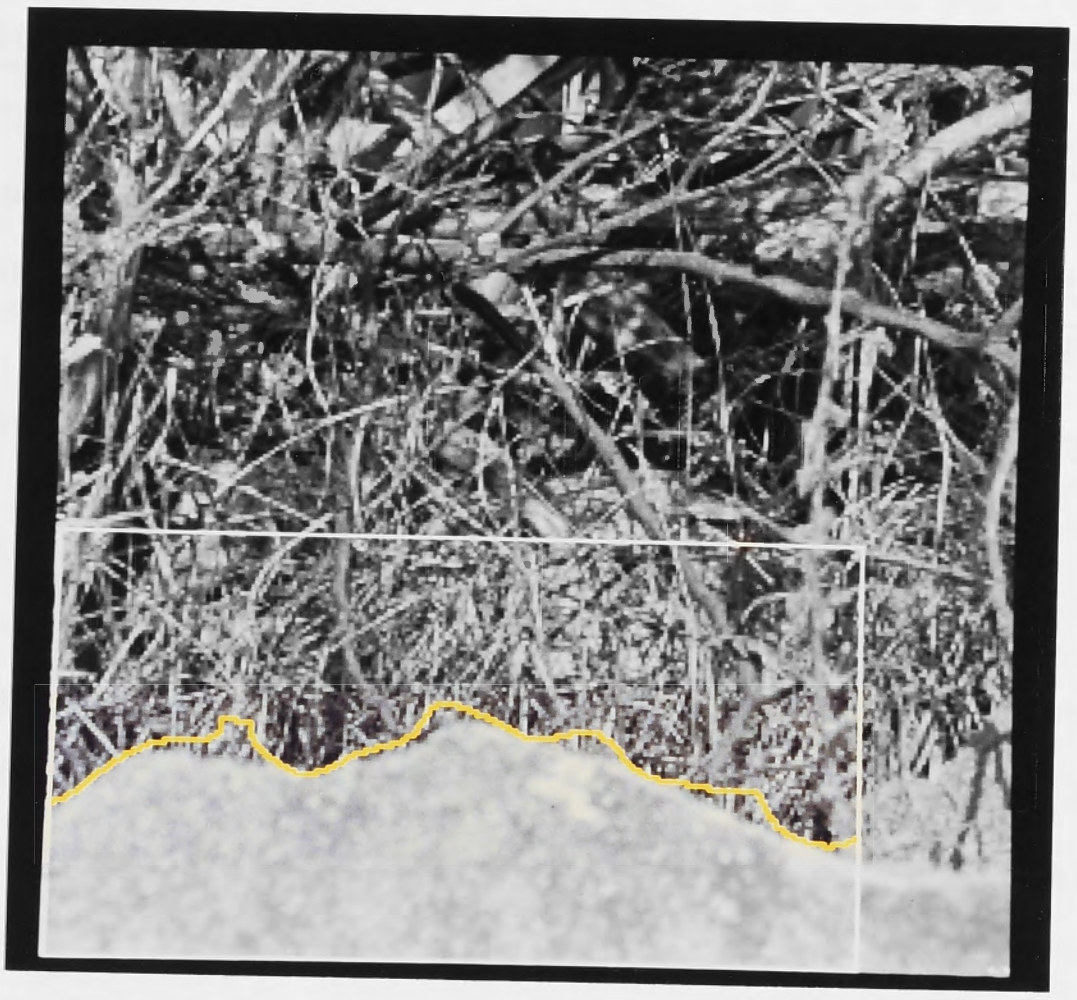


Figure 4.9: Selected edge string and bounding rectangle for image rock

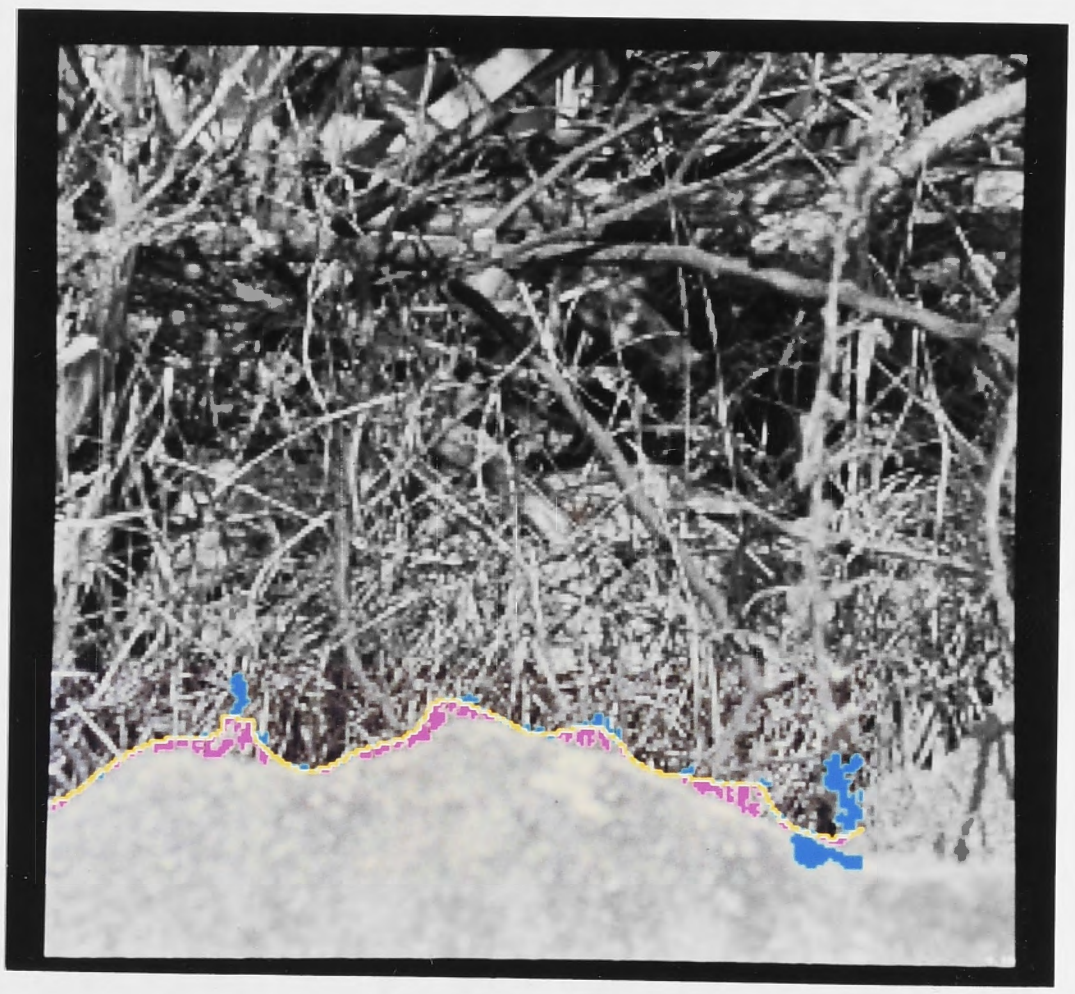


Figure 4.10: Comparison of Marr-Hildreth and optimized results for image rock

Edge detector	Marr-Hildreth	Canny
Scale σ (pixels)	8	6.8
Columns processed	2–218	2–213
Comparison with optimized mask:		
No. of patches failing test	31	21
Maximum ν	22.9	19.3

Table 4.2: Quantitative evaluation of edge detector results on image `rock`

4.1.2.3 Image `lenna`

Figure 4.11 shows the results obtained at four scales on the image `lenna`, which is from the USCIPI data base. Note that the edges of the light vertical bar in the upper-left corner of the image are correctly located at all scales, which is an indication that the implementation is correct, as the Marr-Hildreth edge detector can be expected to work well on isolated edges between areas of constant grey-level. However, notice that at the larger scales, the detected edges in the region of the face bear almost no resemblance to

Figure 4.11: Results from Marr-Hildreth edge detector for image `lenna`

the boundary of the face or features in it, because of interference between neighbouring edges (as described by Shah *et al.*, 1986).

4.1.3 The Canny edge detector

4.1.3.1 Image gmcorner

Figure 4.12 shows the results obtained with the Canny edge detector on the image gmcorner (figure 4.1) at four different scales; figure 4.12(a) shows the edge strengths in grey-scale form, while (b) shows the thresholded edges overlaid on the original image. The numbers in figure 4.12 give the half-width at half-height w of the underlying Gaussian.

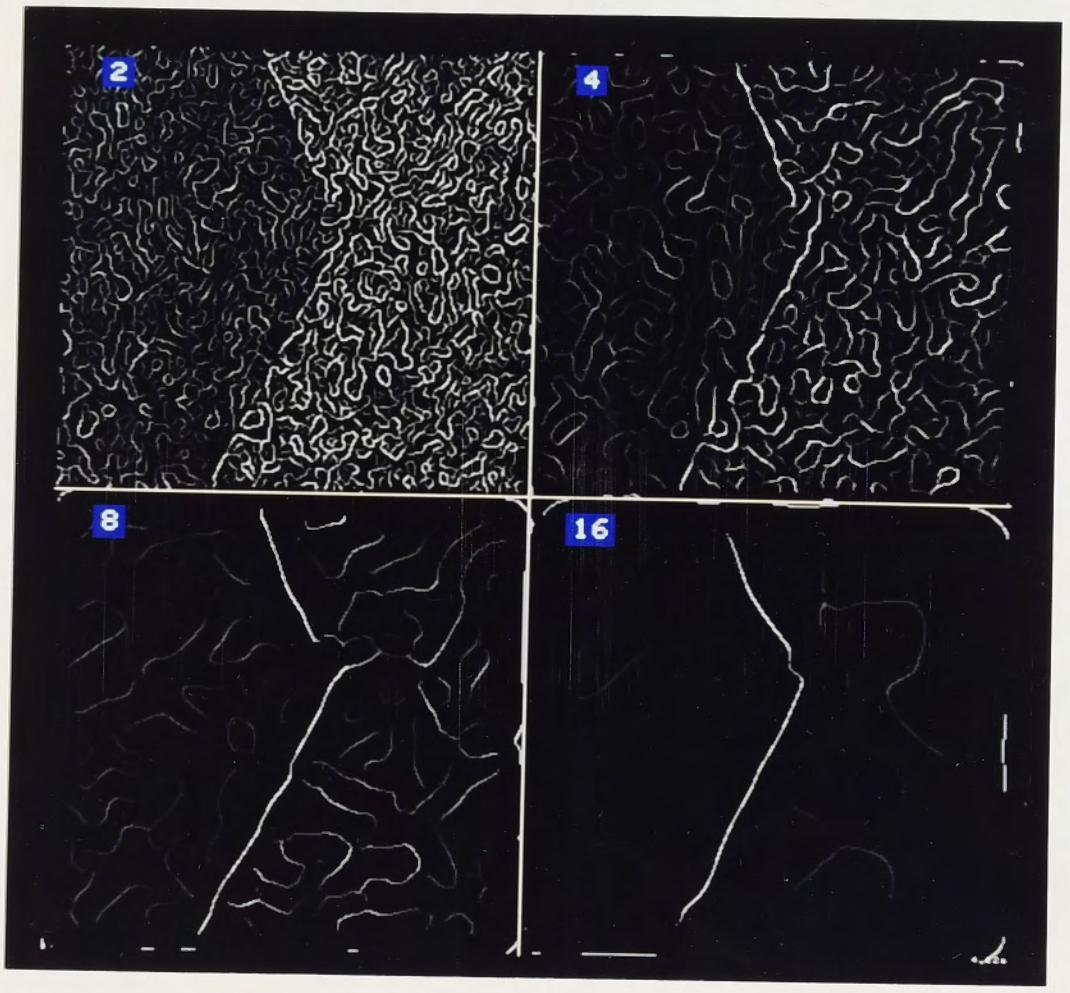
The scale $w = 16$ was chosen for further quantitative analysis of accuracy, as described previously. This scale was chosen because it gives the clearest description of the boundary; at smaller scales, the string of edge pixels tends to break up, especially at the corner. At no scale is the location of the boundary reported accurately. The selected string of edge pixels is shown in figure 4.13, along with the outlines of the areas used for evaluating statistics. Figures 4.14 and 4.15 show where the reported mask differs from the correct and optimized masks, respectively. Table 4.1 lists the numerical results.

4.1.3.2 Image rock

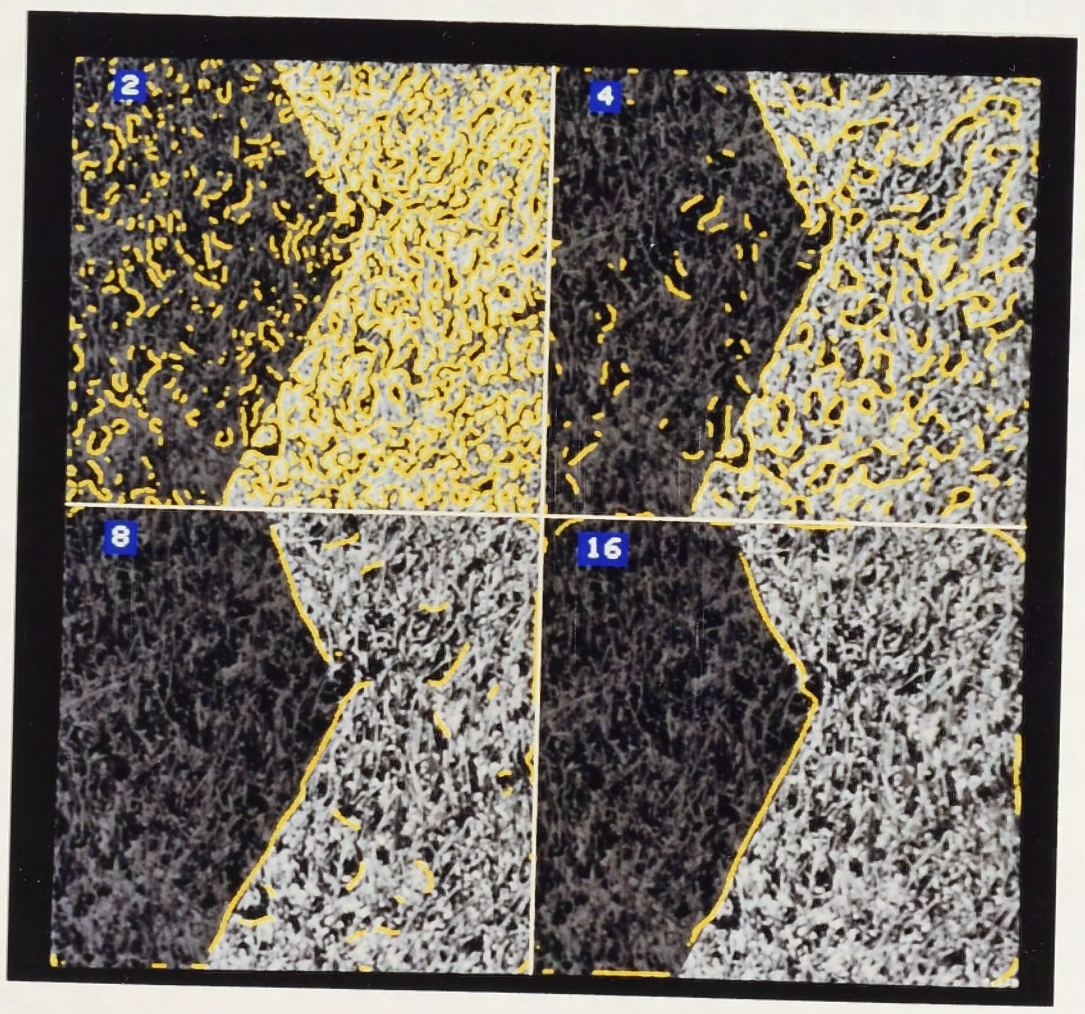
Figure 4.16 shows the results obtained at four scales from image rock (figure 4.7). In this case, the scale $w = 8$ was selected for further evaluation; figure 4.17 shows the selected string of edge pixels, and the bounding rectangle used to confine attention to the boundary of the rock. The results are shown numerically in table 4.2, and visually in figure 4.18.

4.1.3.3 Image lenna

Figure 4.19 shows the results at four scales of applying the Canny edge detector to the image lenna. Once again, the isolated edges are correctly located, indicating that the implementation of the edge detector is correct, while neighbouring edges interfere at the larger scales.



(a) edges in grey-scale form



(b) edges overlaid on image

Figure 4.12: Canny edge detector results on image gmcorner

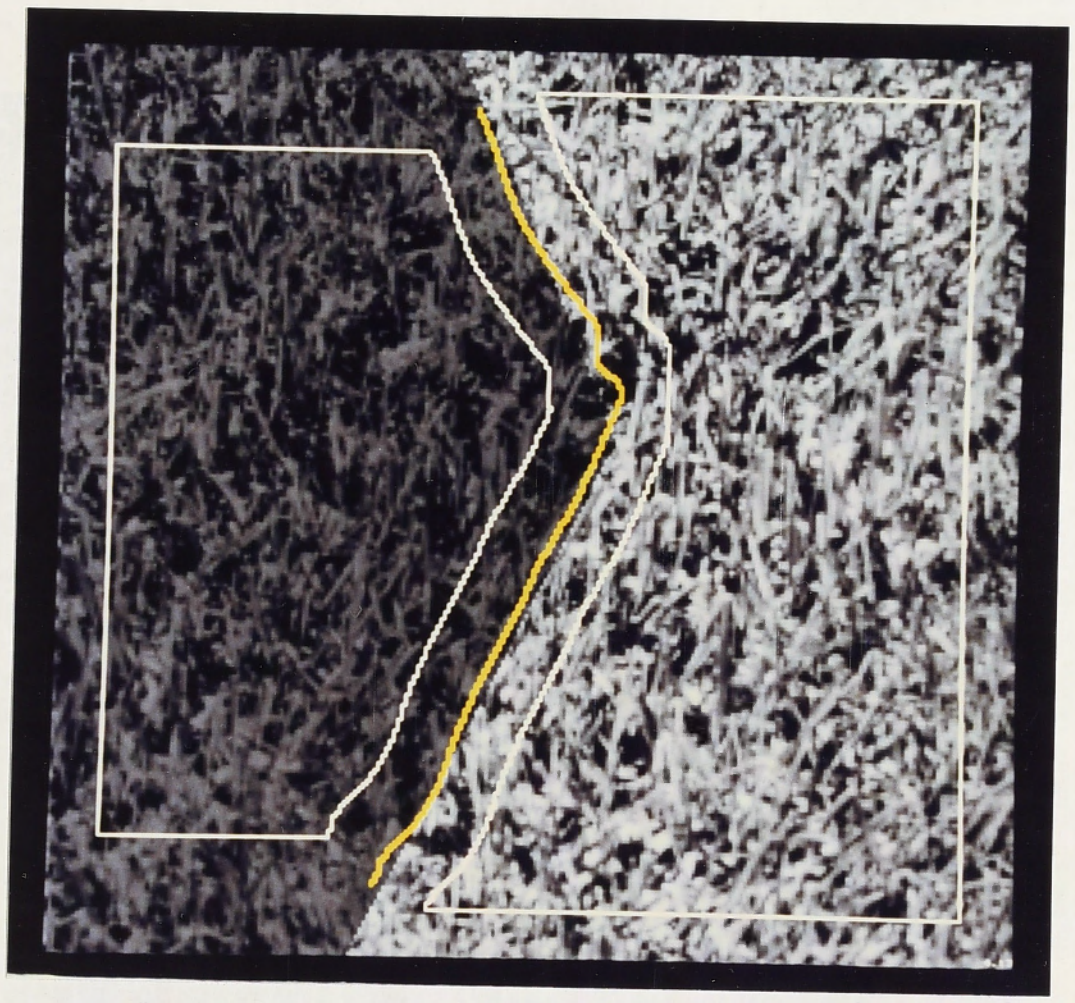


Figure 4.13: Selected edge string and patches for statistics for image `gmcorner`

4.1.4 Conclusions about convolution edge detectors

The main point to be drawn from the results given above is that these convolution edge detectors give quite inaccurate results when applied to textured images. At small scales, the responses to important boundaries are indistinguishable from responses to the texture, and are also fragmentary and irregular. At larger scales, the desired responses are more obvious, but are inaccurate. The inaccuracies are obvious visually, and have been confirmed by application of the consistency test. The consistency test is a useful way to check the accuracy of an edge detector without requiring knowledge of the true boundary location. The results from these convolution edge detectors satisfy none of the three criteria set out in chapter 3.

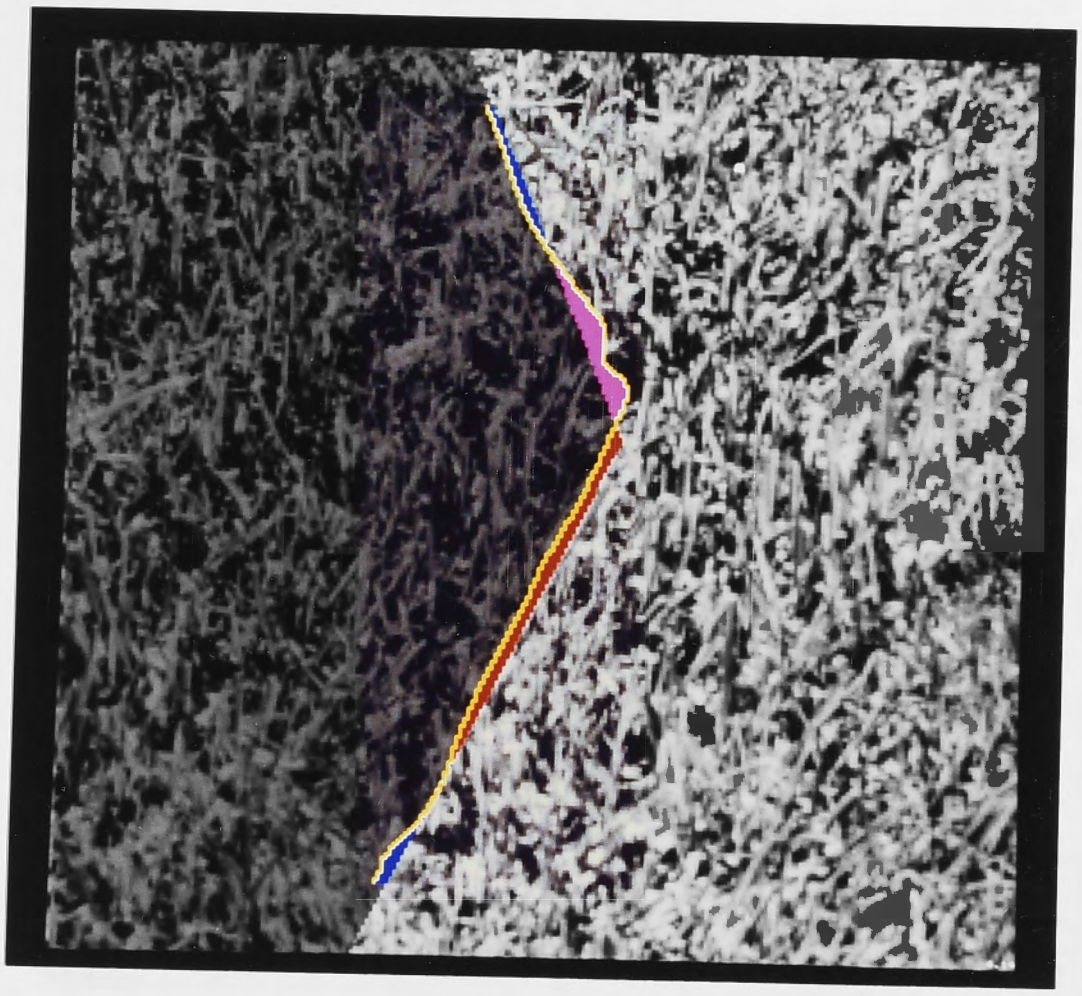


Figure 4.14: Comparison of Canny and correct results for image gmcorner

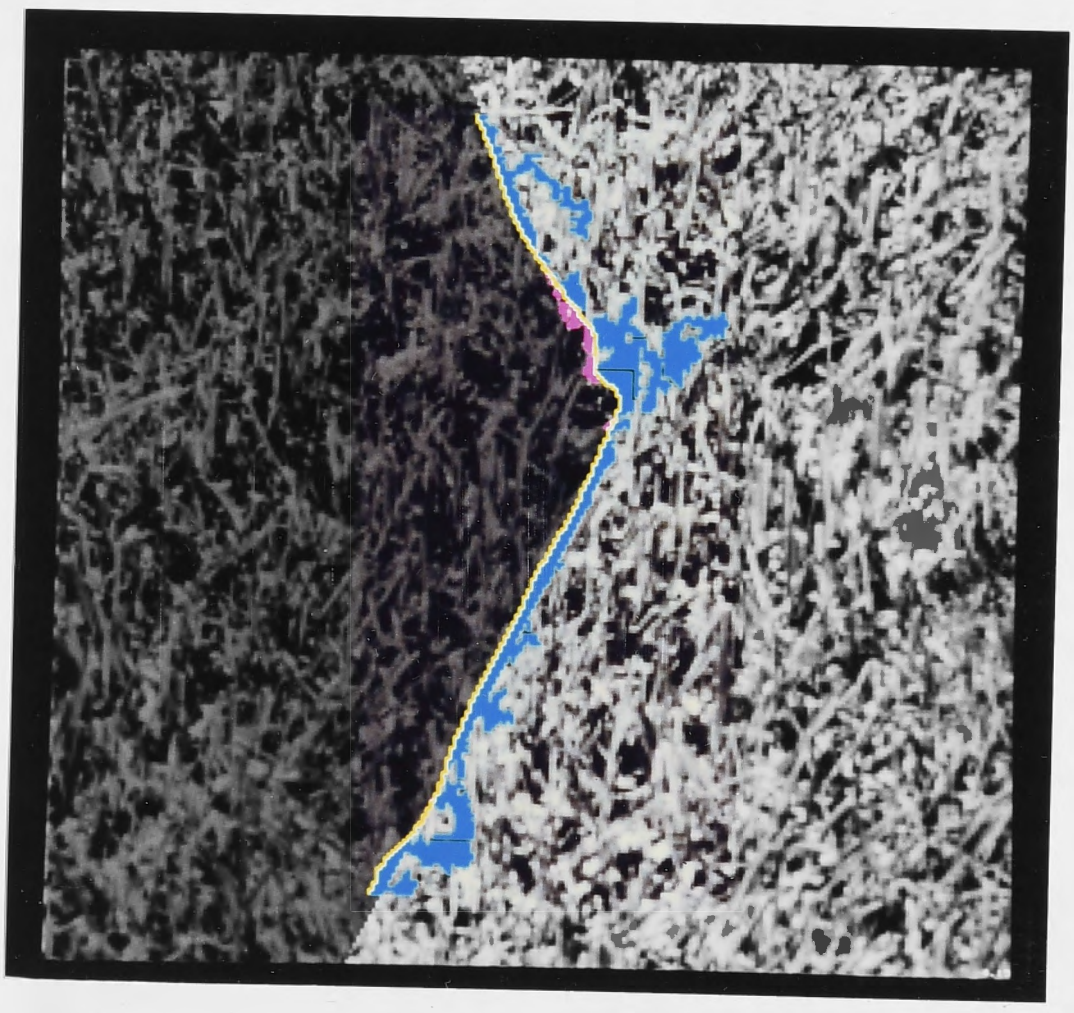


Figure 4.15: Comparison of Canny and optimized results for image gmcorner

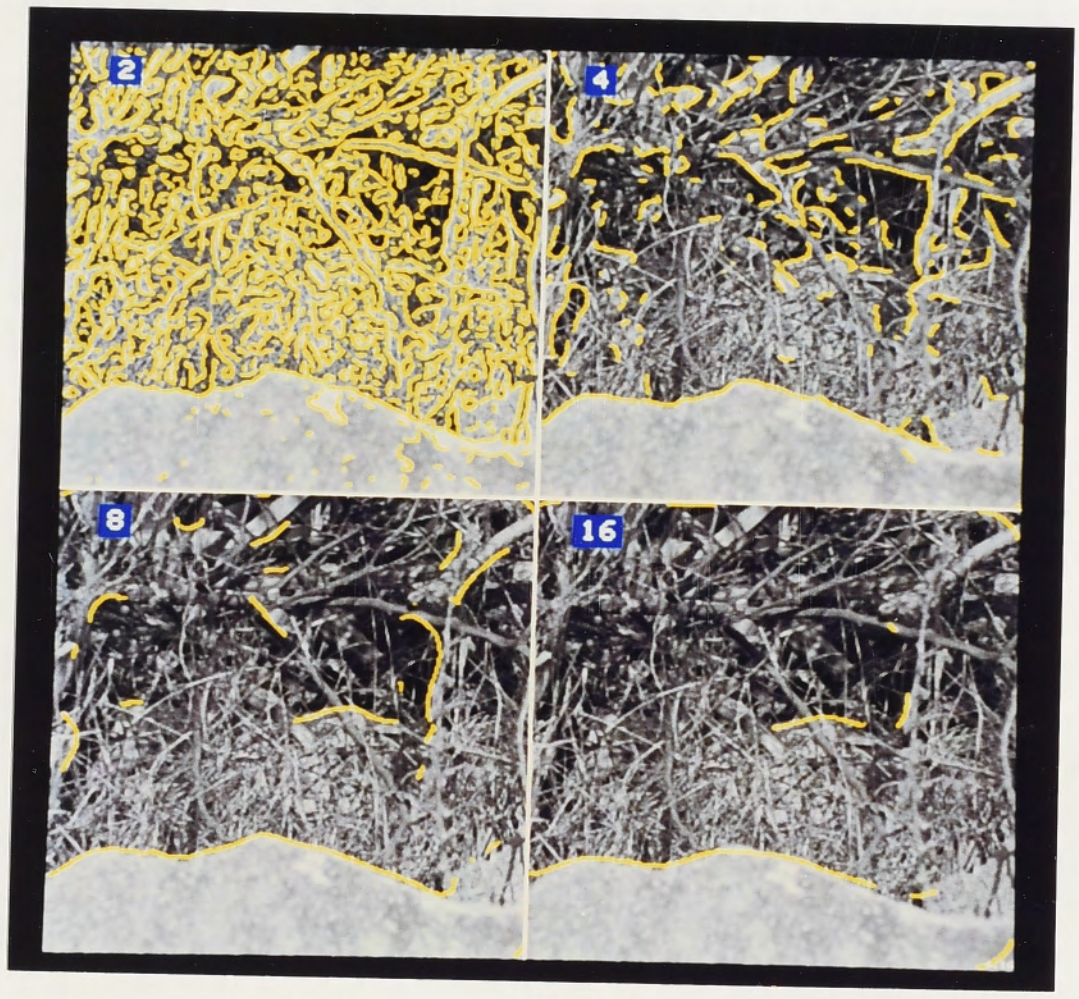


Figure 4.16: Results from Canny edge detector for image rock

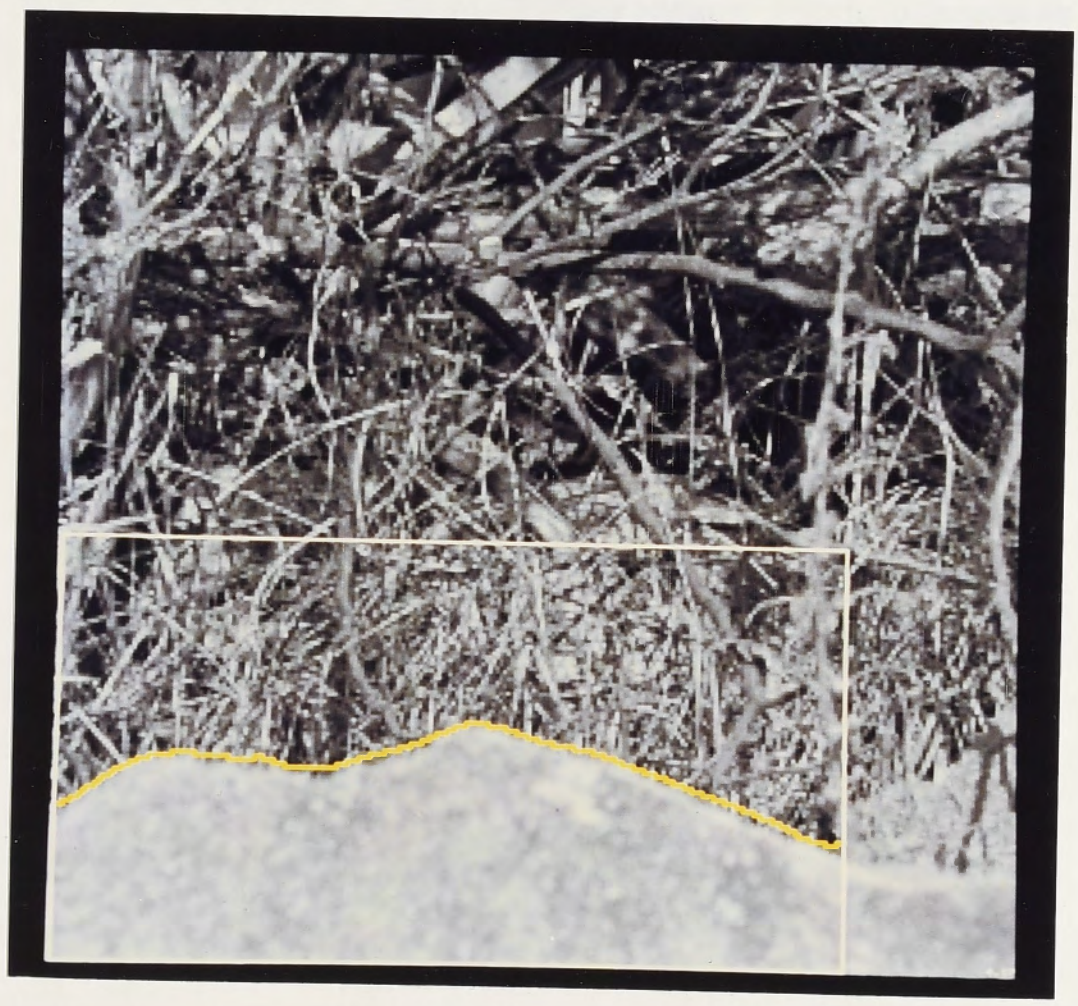


Figure 4.17: Selected edge string and bounding rectangle for image rock

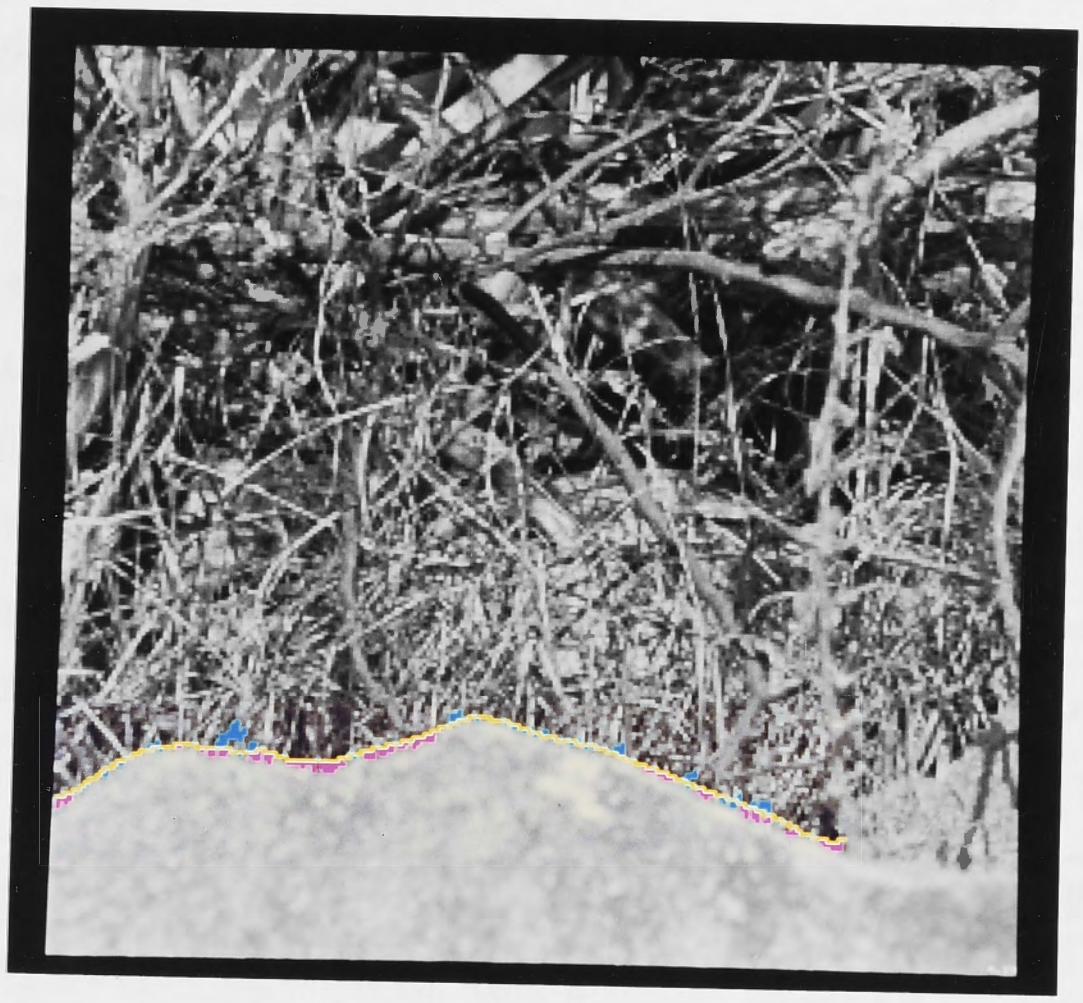


Figure 4.18: Comparison of Canny and optimized results for image rock

Figure 4.19: Results from Canny edge detector for image lenna

4.2 Evaluation of results obtained with BLM algorithm

In contrast to the convolution edge detectors discussed above, the BLM algorithm gives accurate and concise information about the locations of boundaries in textured images. As discussed in chapter 3, the current implementation of the BLM algorithm assumes that a single boundary is to be located, and that the regions on either side are homogeneous. The range of real-world images which are suitable for processing with the current implementation of the BLM algorithm is therefore rather restricted, since its operation is best demonstrated on boundaries of reasonable length with homogeneous regions on either side; section 5.3 discusses how the approach could be extended to remove these restrictions. Table 4.3 lists the images which were used. Confidence intervals on segment positions or confidence contours on knot positions are shown for

Name	Boundary shape	Region 1	Region 2
`gmcorner`	two lines with corner	dark grass	grass
`gmarc`	circular arc	dark grass	grass
`gmprot`	polygon	dark grass	grass
`gmwave`	perturbed line	grass	dark grass
`rcirc`	circle	dark raffia	raffia
`learaf`	circle	raffia	leather
`noise`	straight line	Gaussian noise	Gaussian noise
`dots`	circular arc	dots, density = 0.1	dots, density = 0.01
`straw`	filtered noise	straw	straw rotated 90°

(a) Artificial images

Name	Description
`rock`	Rock in front of vegetation.
`house`	Brick house.
`pinetree`	Outline of pine tree against sky.
`trunk`	Trunk of tree against grass.

(b) Real-world images

Table 4.3: Images used for testing the BLM algorithm

Image	Area misclassified	Boundary length	Mean error
gmcorner	3	280.9	0.01
gmarc (PWL)	144	287.5	0.50
(cubic)	9	287.1	0.03
gmprot	26	431.9	0.06
gmwave	783	303.9	2.58
rcirc (PWL)	69	422.8	0.16
(cubic)	9	420.7	0.02
learaf	862	429.2	2.01
noise	13	265.3	0.05
dots	244	133.2	1.83
straw	400	307.7	1.30

Table 4.4: Quantitative evaluation of BLM algorithm results

some representative images.

The split-and-merge procedure has been used to obtain the initial segmentation for all of the artificial images and all but one of the real-world images. The exception is image house, in which the BLM algorithm was set to find a very subtle boundary.

The methods which are appropriate for evaluating the BLM algorithm results are visual inspection, and for the artificial images, comparison of the correct and reported boundaries. The consistency test has already been applied in the course of the BLM algorithm, so the boundaries will be consistent with the image data. Therefore it is not necessary to check the results from the BLM algorithm for consistency.

The results of the quantitative comparison for the artificial images are shown in table 4.4. In this table, the 'Area misclassified' column gives the number of pixels classified differently by the reported and correct boundaries. The 'Boundary length' column gives the length of the reported boundary, excluding any segments which complete the boundary around the border of the image. The length of the reported boundary is used rather than the length of the correct boundary, because the former is easier to calculate, and should be very close to the latter. The 'Mean error' column gives the ratio of the area misclassified to the length of the reported boundary, as a measure of

the mean distance between the reported and correct boundaries.

Where the boundary extends from one border of the image to another, it is possible for the segments completing the boundary around the border of the image to move away from the border slightly if there are some atypical pixels adjacent to the border. The effect of this is to assign these pixels to region 2, when they should be assigned to region 1. The few pixels misclassified in this way have been excluded from consideration in table 4.4, since they do not occur along the boundary of interest.

4.2.1 Comments on individual images

4.2.1.1 Image gmcorner

This image, shown in figure 4.1, is one of those used for testing the convolution edge detectors. The results from the BLM algorithm, shown in figure 4.20, are identical to the boundary perceived by human observers. This is an interesting image for testing a boundary locator, for the following reasons:

(a) It is sufficiently textured to present problems for convolution edge detectors, yet human observers perceive a clear, sharp boundary.

(b) The asymmetry in the pixel intensity distributions means that the bright pixels are disproportionately important in fixing the location of the boundary (bright pixels appear only in the right-hand region, whereas dark pixels occur in both regions).

(c) The fortuitous presence of the dark blob just above the corner is a good test of the ability of any technique to accept that the blob is just part of the texture (as indeed the BLM algorithm does).

This image represents an easy test for the BLM algorithm, in that the boundary shape is composed of straight-line segments. However, it is worth noting that the BLM algorithm has placed a knot accurately at the position of the corner, rather than within the dark blob just above the corner. The algorithm has been able to give much more weight to the presence of the light pixels just above the corner than to the dark pixels within the blob, as indeed these light pixels deserve. Also, the dark blob does not cause the boundary to fail the consistency test.

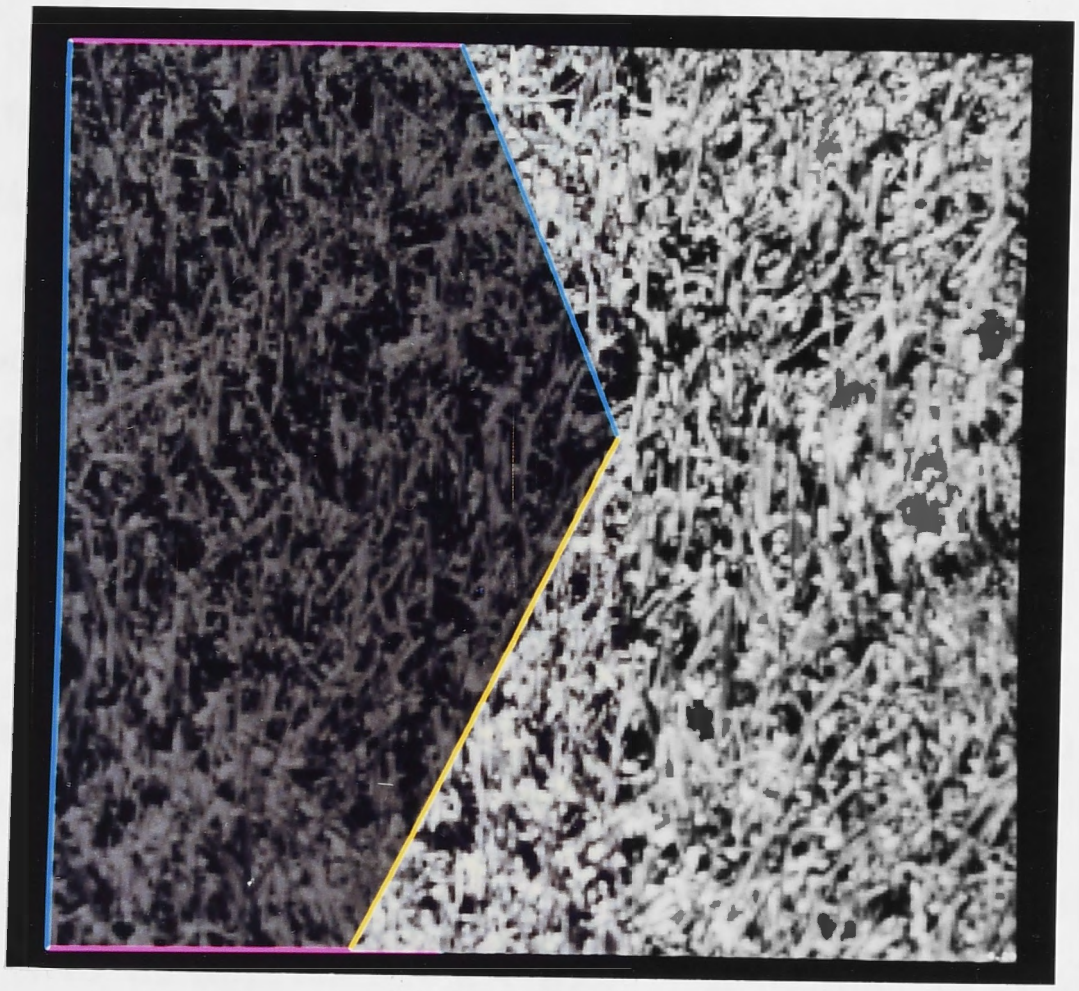


Figure 4.20: Results from BLM algorithm for image gmcorner

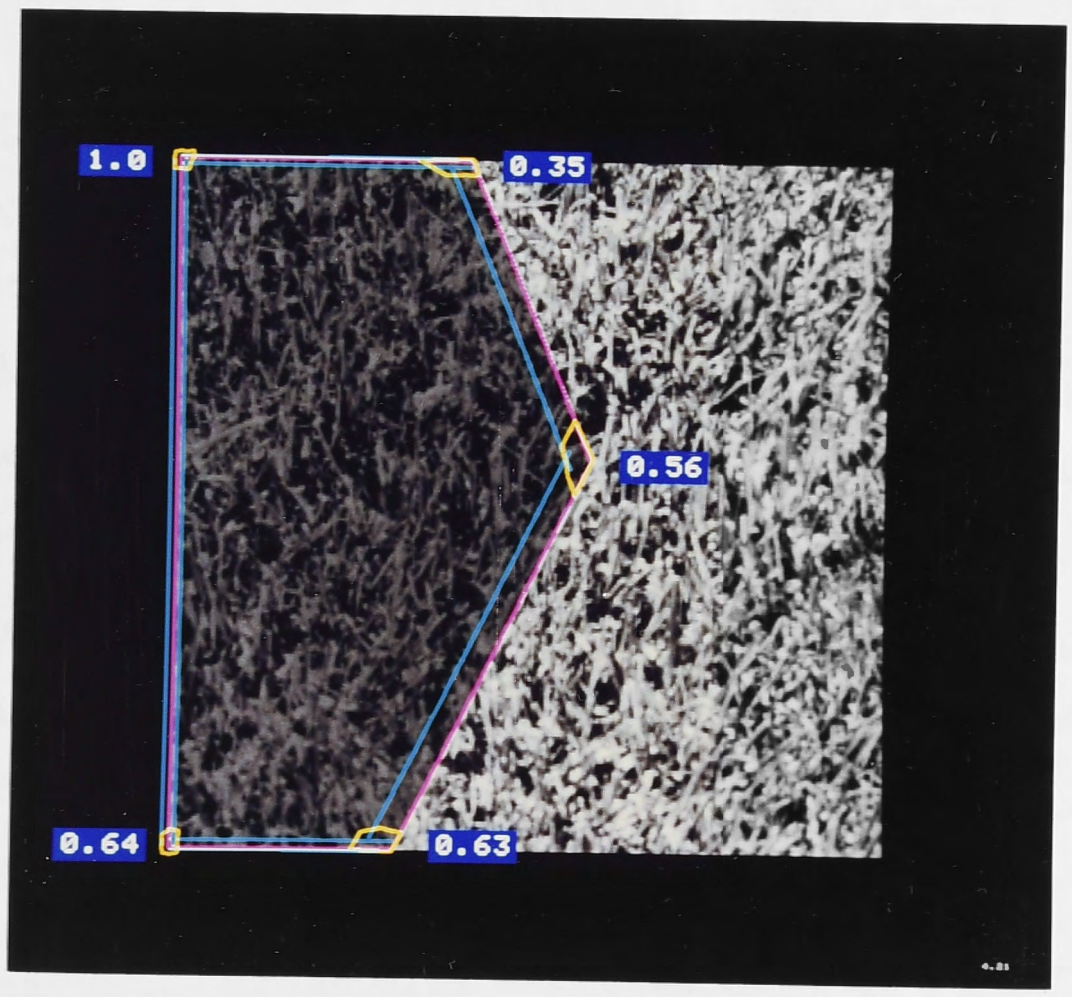


Figure 4.21: Confidence intervals and contours for image gmcorner (magnified by 10)

The confidence intervals and contours are shown in figure 4.21. The intervals and contours are shown magnified by a factor of 10 for ease of viewing; the regions are very distinct, and the boundary is tightly constrained. The confidence intervals are essentially zero on the right-hand side, and the boundary (shown in magenta) has thus overlaid the confidence interval (shown in cyan) in some cases. The confidence contours are shown in yellow. The boundary is much more tightly constrained on the right than on the left, due to the asymmetry in the pixel intensity distributions. The confidence intervals and contours are consequently asymmetrical, extending much further to the left then the right. Figure 4.21 also shows the cornerness measure at each knot.

4.2.1.2 Image gmarc

Figure 4.22 shows the original image, which was constructed in a similar manner to **gmcorner**, except that the shape of the boundary is an arc of a circle, rather than a polygon. This represents a more difficult test for the BLM algorithm, in that the smooth curve has to be approximated either by a number of straight-line segments or by a cubic segment. Inspecting the results, shown in figure 4.23, we see that the algorithm has found that six segments are sufficient to give a consistent boundary. Note that the knots have been placed generally at locations along the boundary where there is a small dark patch adjacent to the boundary in the right-hand region.

Using a cubic polynomial segment gives better results, shown in figure 4.24. The algorithm was able to obtain a consistent description of the boundary with only one cubic segment. This description is preferable to the piece-wise linear description, because it is simpler; eight parameters are required for the spline segment, compared to sixteen for the seven piece-wise linear segments (defined by eight knots). The program was instructed to use a cubic segment.

Figure 4.25 shows the confidence intervals and contours for the piece-wise linear boundary in figure 4.23, with a magnification of 3 on the intervals and 1 on the contours. Note that the confidence intervals are broader for the shorter segments, and that the confidence contours are elongated along the direction of the boundary. The cornerness measures for the knots along the arc are generally lower than those shown for image **gmcorner**, i.e., the knots are less 'corner-like'.

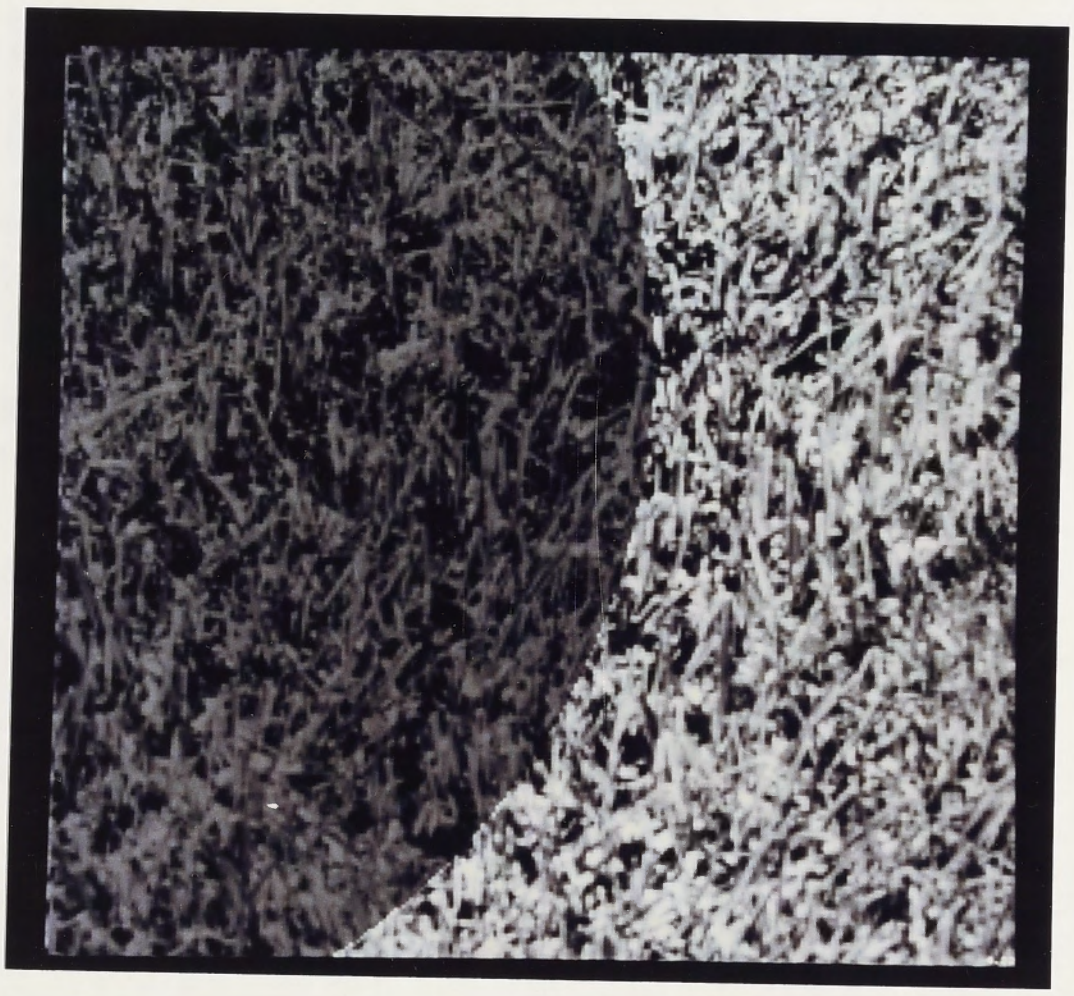


Figure 4.22: Image gmarc

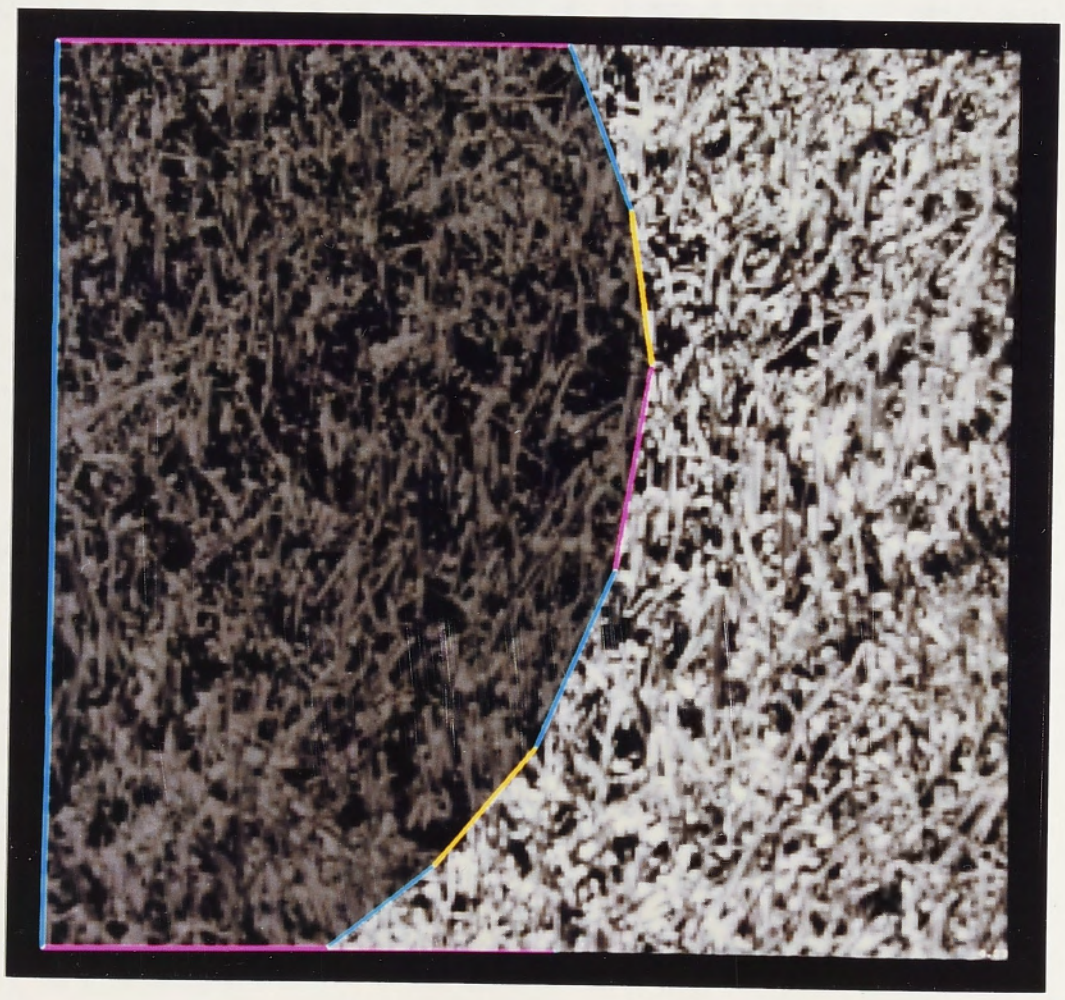


Figure 4.23: Results from BLM algorithm for image gmarc; piece-wise linear boundary

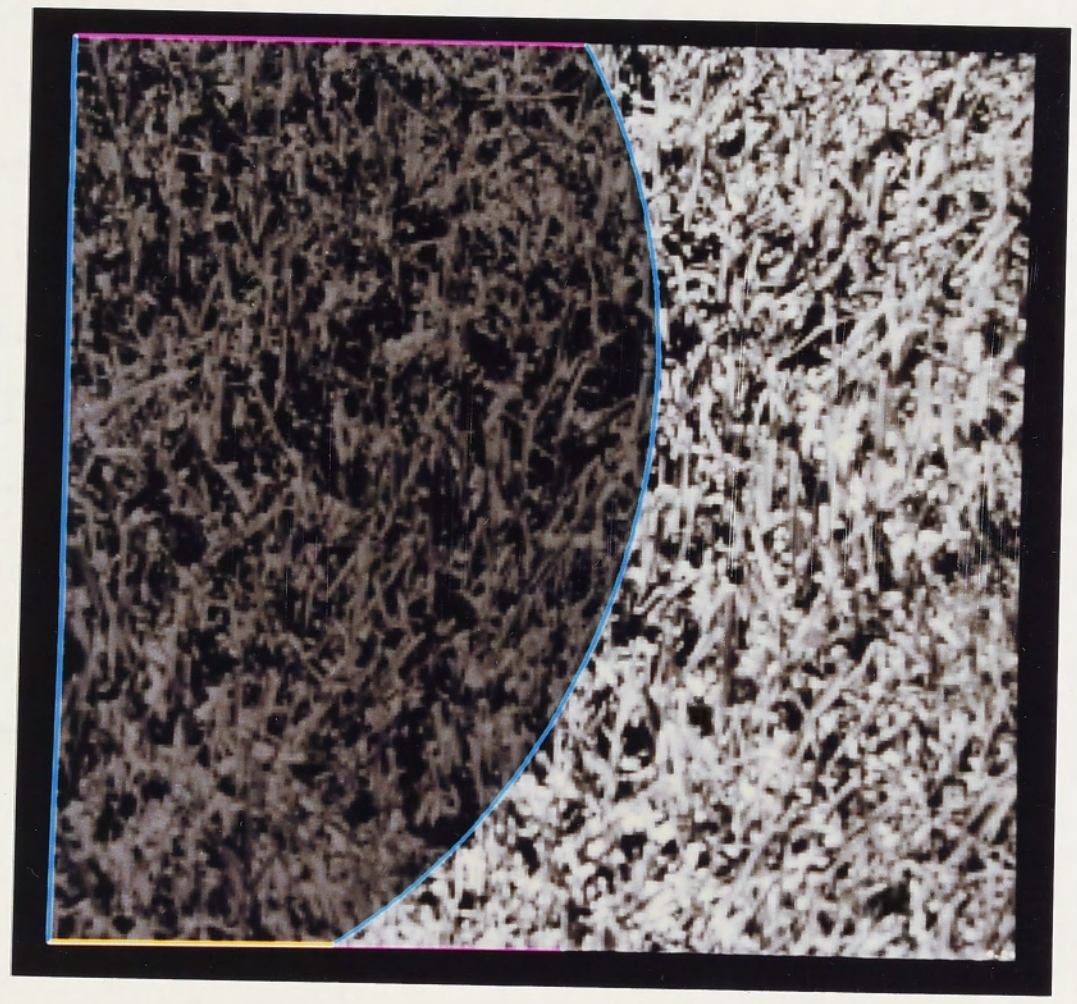


Figure 4.24: Results from BLM algorithm for image gmarc using cubic segment

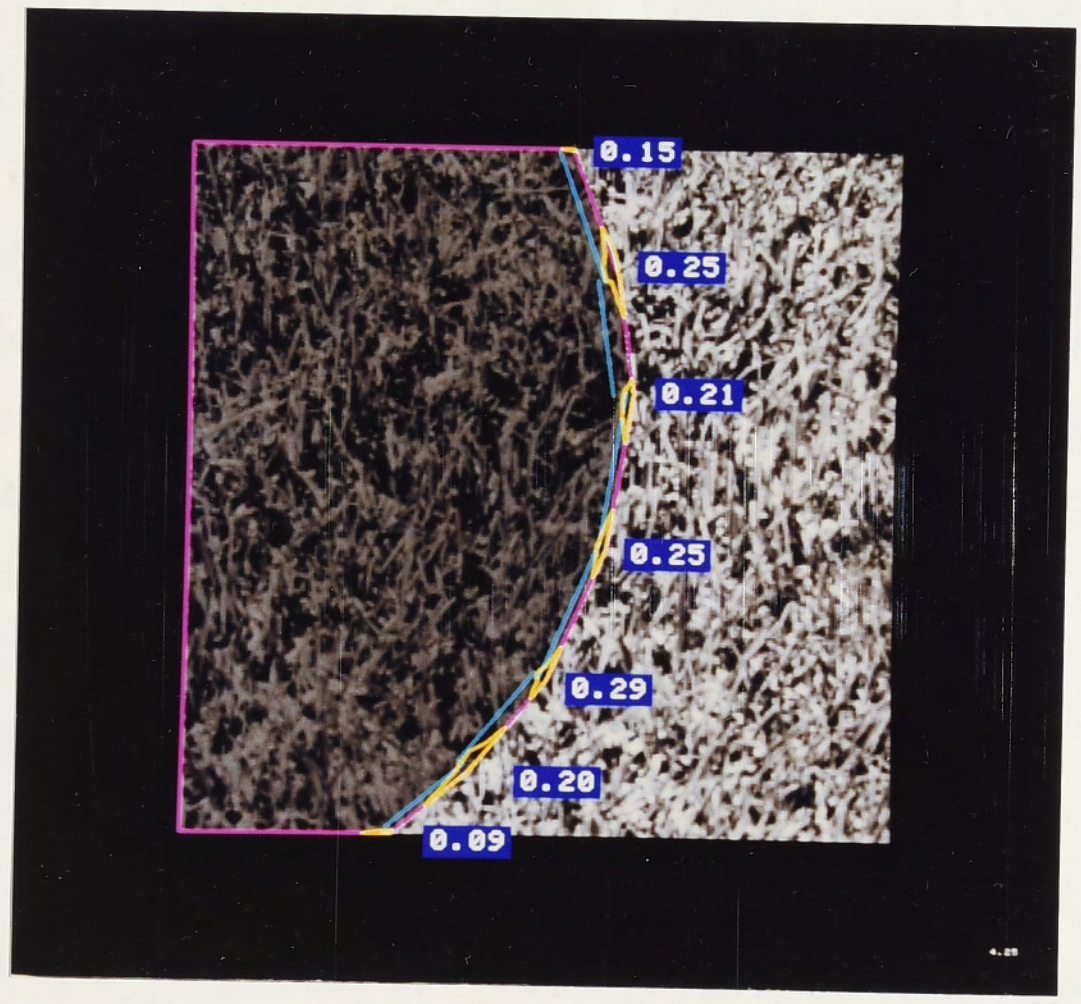


Figure 4.25: Confidence intervals (magnified by 3) and contours for image gmarc

4.2.1.3 Image gmprot

This image (figure 4.26) was constructed in a similar manner to gmcorner. It was deliberately chosen to check the consistency test and elaboration steps in the BLM algorithm. After two iterations of the algorithm, the top-right portion of the boundary has been fitted with a single segment, which is not consistent. The iteration has to proceed through three more cycles, adding three more knots on that segment, before further progress can be made. Then the middle of the three added knots moves up into the protrusion. Two more knots are added, which gives a consistent boundary. The simplification step yields the result shown in figure 4.26, which is correct. Parameter $h_{\max}$ had to be increased from 10 to 20 for this image (see section 3.4.5.1). The higher value could have been used for all images; the lower value was used in other cases because the processing time required was less.

4.2.1.4 Image gmwave

In generating this image (figure 4.27), the position of the true boundary on each scan-line was obtained by filtering white noise with a low-pass filter of Gaussian profile. Points to the right of the true boundary position were multiplied by 2/3. Figure 4.28 shows this image with the true boundary outlined. The true boundary contains more detail than can be resolved in the final image, because of the texture in the regions and the overlap in the distributions. Perceptually, the impression is of a boundary which is not clear and sharp (though still quite obvious). There appear to be texture elements in the left-hand region which overlap the right-hand region—this is a kind of boundary which is quite common in real-world images (for example the boundary between a concrete path and a lawn). The results, shown in figure 4.29, follow the true boundary remarkably well, given the relatively small contrast of the edge.

4.2.1.5 Image rcirc

This image is an example of one in which the boundary is very tightly constrained, because the regions are so distinct. This, combined with the curved boundary shape, forces the BLM algorithm to use many piece-wise linear segments. The result shown in figure 4.30 is a good approximation to the circle, using 18 segments.

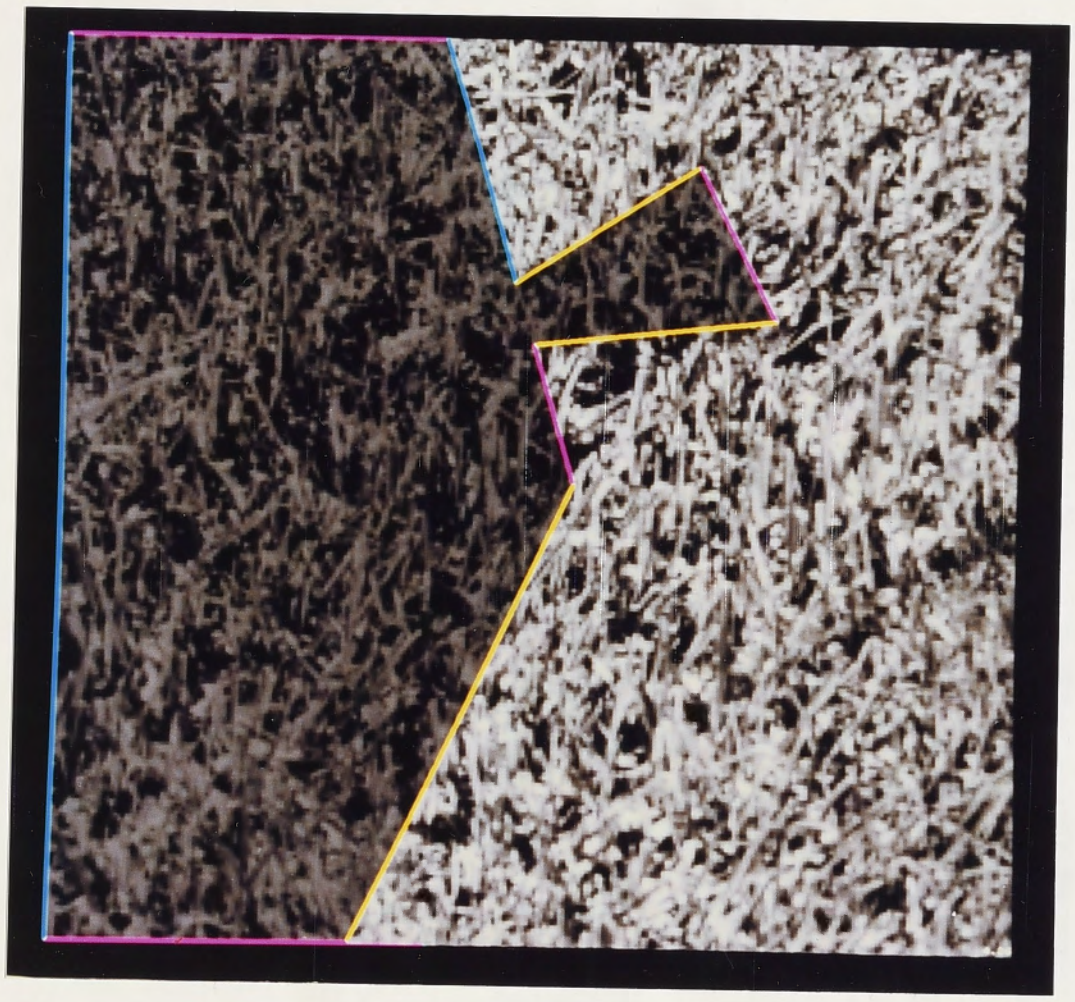


Figure 4.26: Results from BLM algorithm for image gmprot

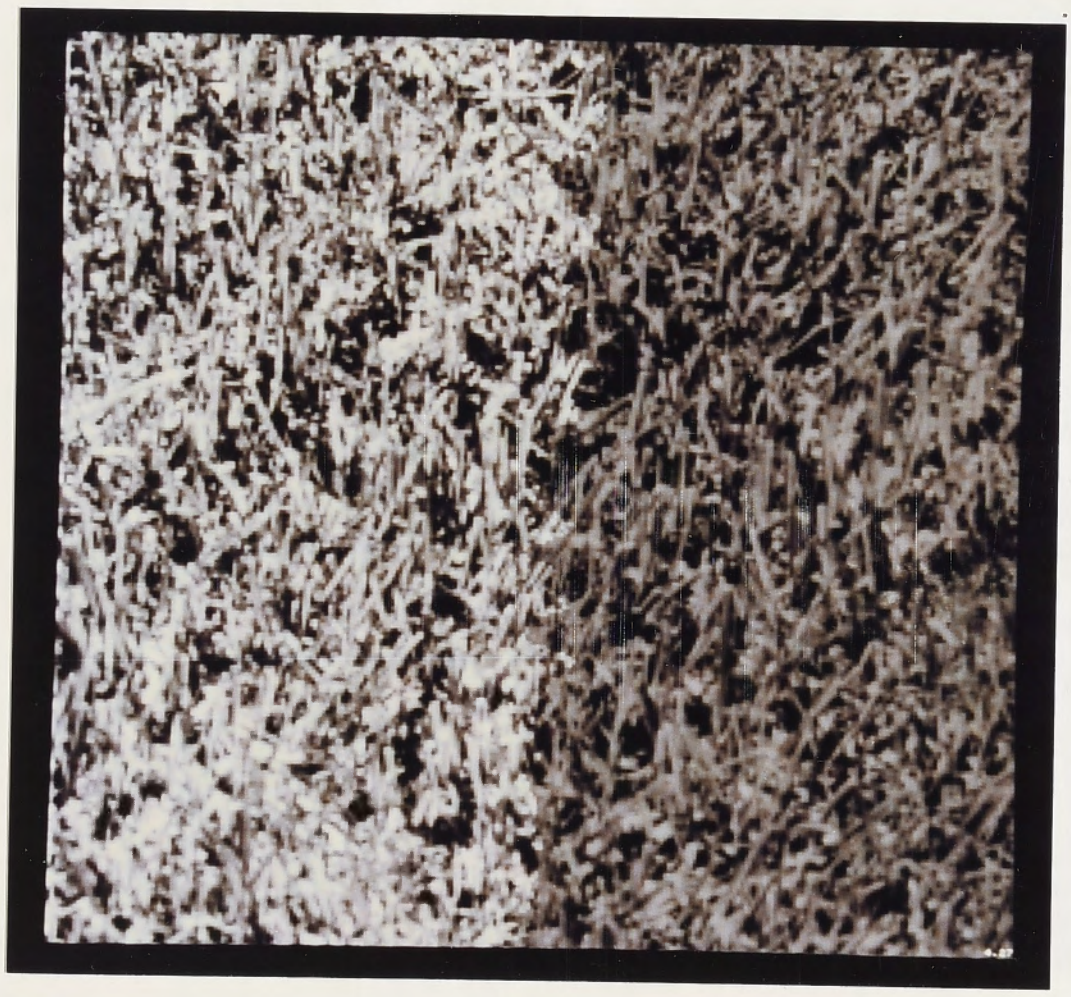


Figure 4.27: Image gmwave

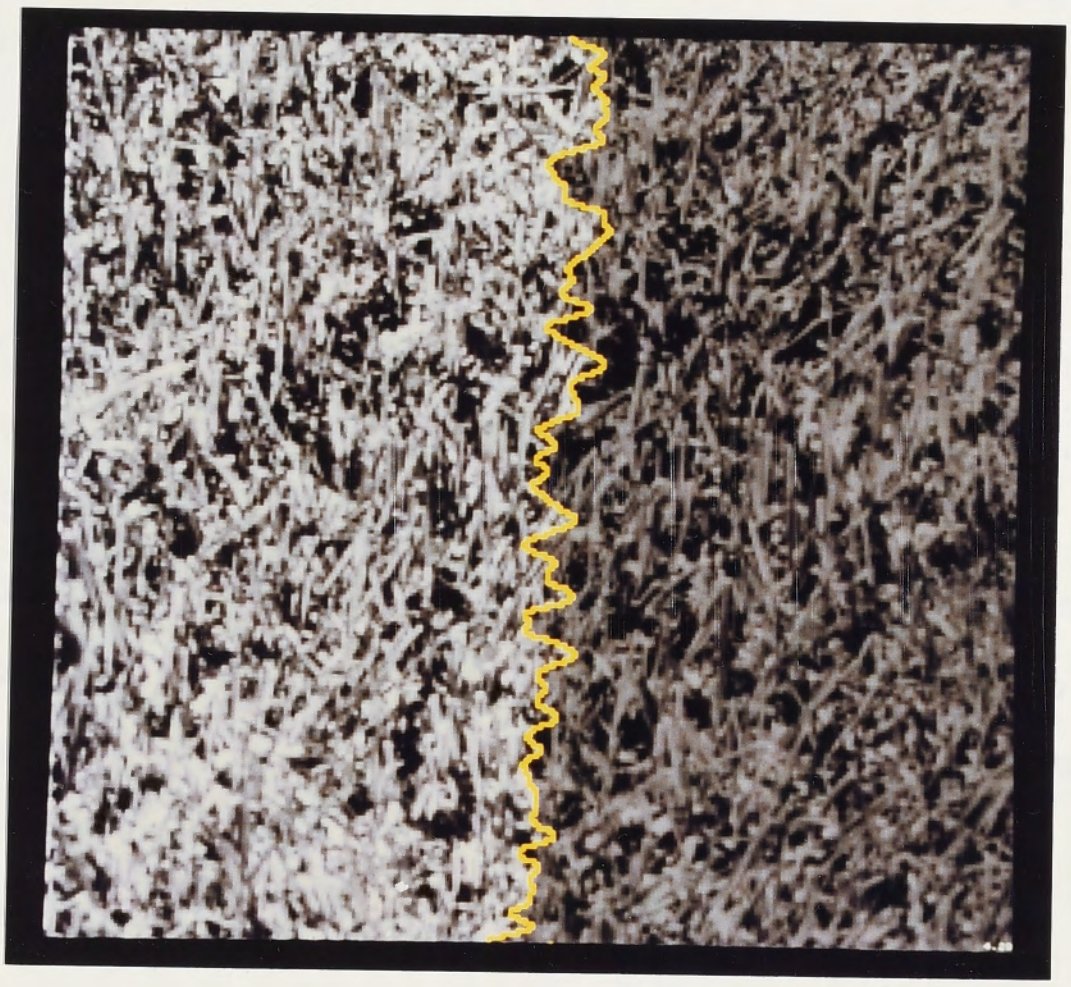


Figure 4.28: True boundary position for image gmwave

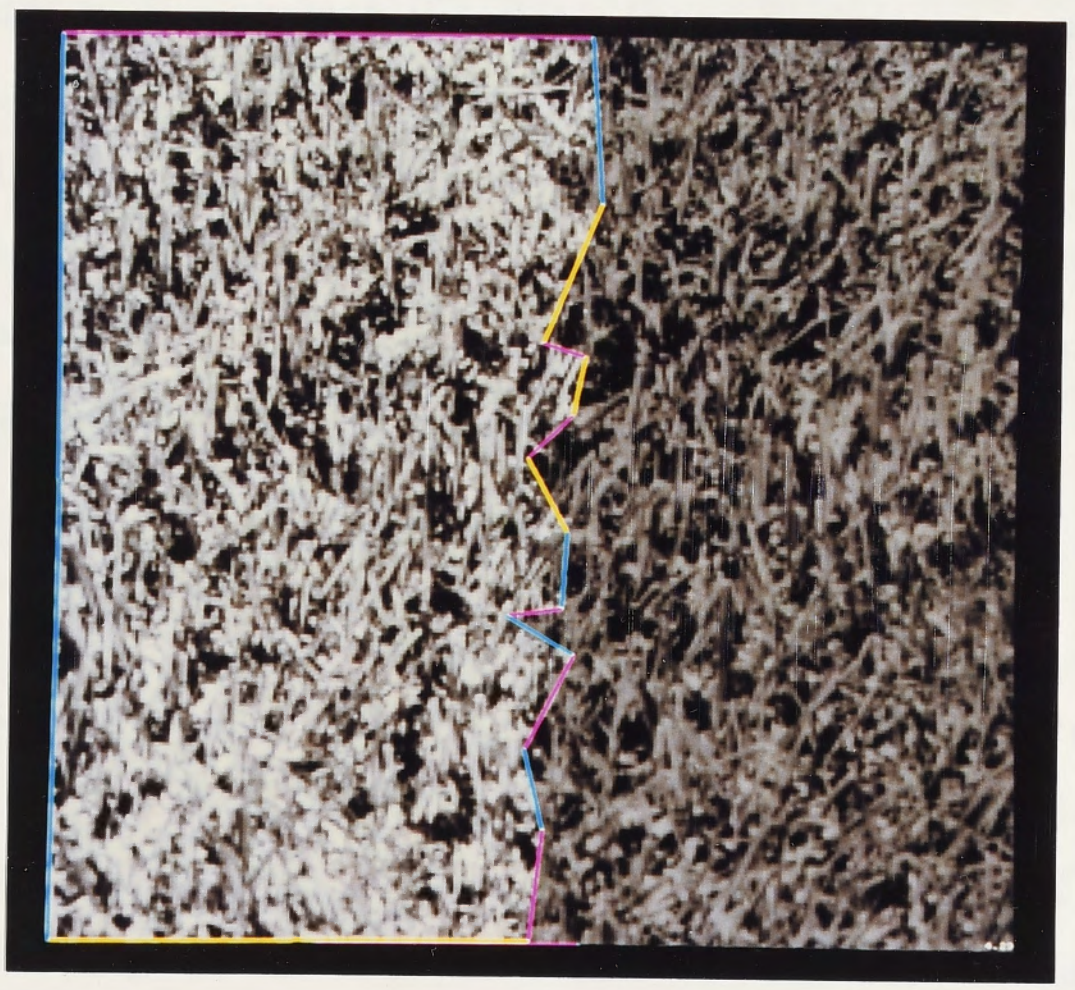


Figure 4.29: Results from BLM algorithm for image gmwave

A better approximation to the circle is obtained by using four cubic segments with slope continuity at each knot, as shown in figure 4.31. This is a better description of the boundary, because it is both simpler and more accurate. The program was instructed to use four cubic segments with slope continuity.

4.2.1.6 Image `learaf`

This image has the same boundary as `rcirc`, but the regions are much less distinct (the contrast is lower), and the textures in the regions are different. The results are shown in figure 4.32. Notice the error that the algorithm makes in the bottom left-hand part of the inner region. This is an example where non-homogeneity of the regions has caused problems. Human interpretation of this image appears to be based on the nature of the textures in the regions, as well as the grey-level distributions.

4.2.1.7 Image `noise`

Figure 4.33 shows this image, which has a straight boundary between two fields of white Gaussian noise. The standard deviations of pixel intensity in each region are equal, and are equal to the difference between the means, that is, the signal to noise ratio is 1. In this image, the percept of a clear, sharp boundary can only be obtained when a substantial length of the boundary is visible; if all but a short length is masked off (as in figure 5.1 on page 147), the boundary does not look at all sharp and clear. The percept of a sharp, clear boundary therefore seems to require that information be integrated along the length of the boundary, as indeed the BLM algorithm does. Its results, shown in figure 4.34, are correct.

4.2.1.8 Image `dots`

This image contains random dots, with density 0.1 to the left of the boundary, and density 0.01 to the right of the boundary. Figure 4.35 shows the results from the BLM algorithm, which correspond very well to the perceived boundary. The shape of the correct boundary is a circular arc; this is another example where the details of the shape of the correct boundary are obscured by the texture. The statistical fluctuations within the regions are rather extreme in this image; nevertheless, the BLM algorithm performs well.

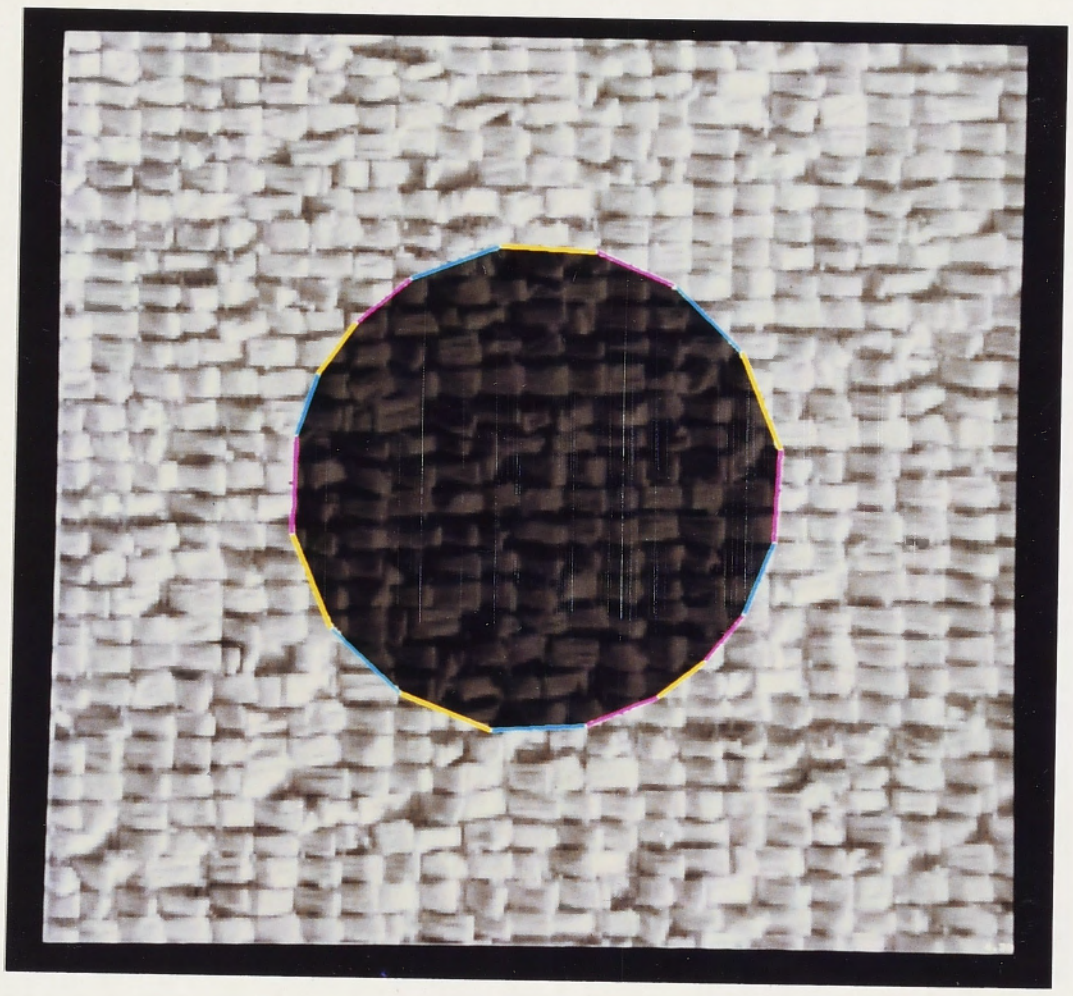


Figure 4.30: Results from BLM algorithm for image `rcirc`; piece-wise linear boundary

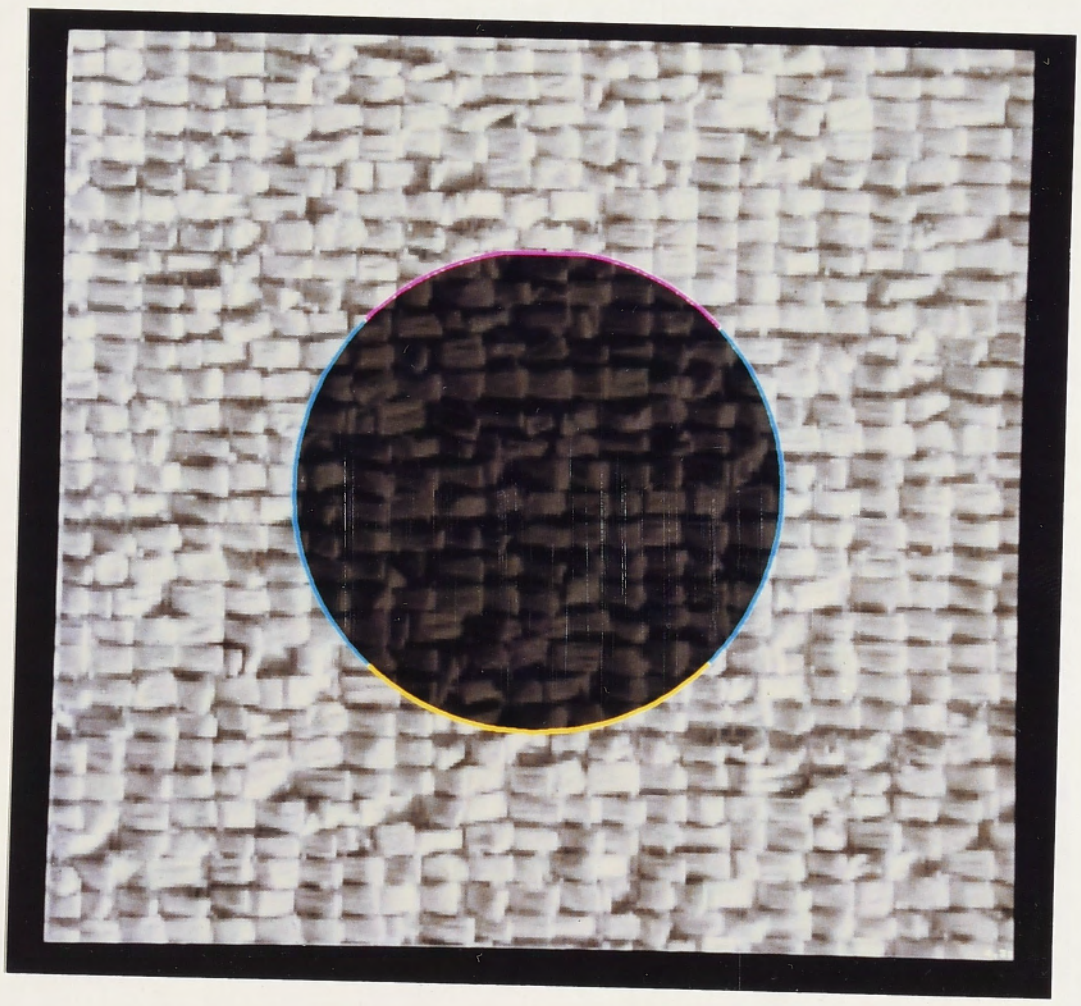


Figure 4.31: Results from BLM algorithm for image `rcirc` with four cubic segments

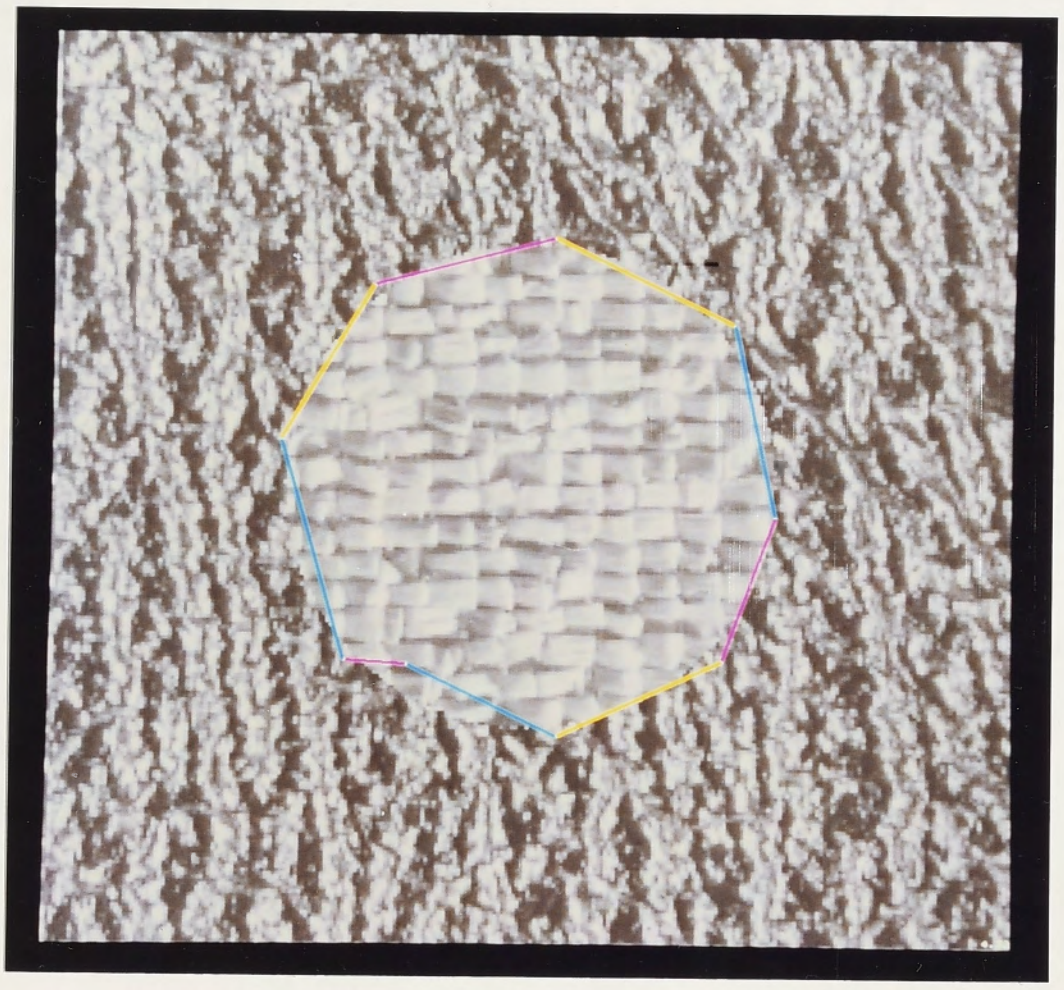


Figure 4.32: Results from BLM algorithm for image `learaf`

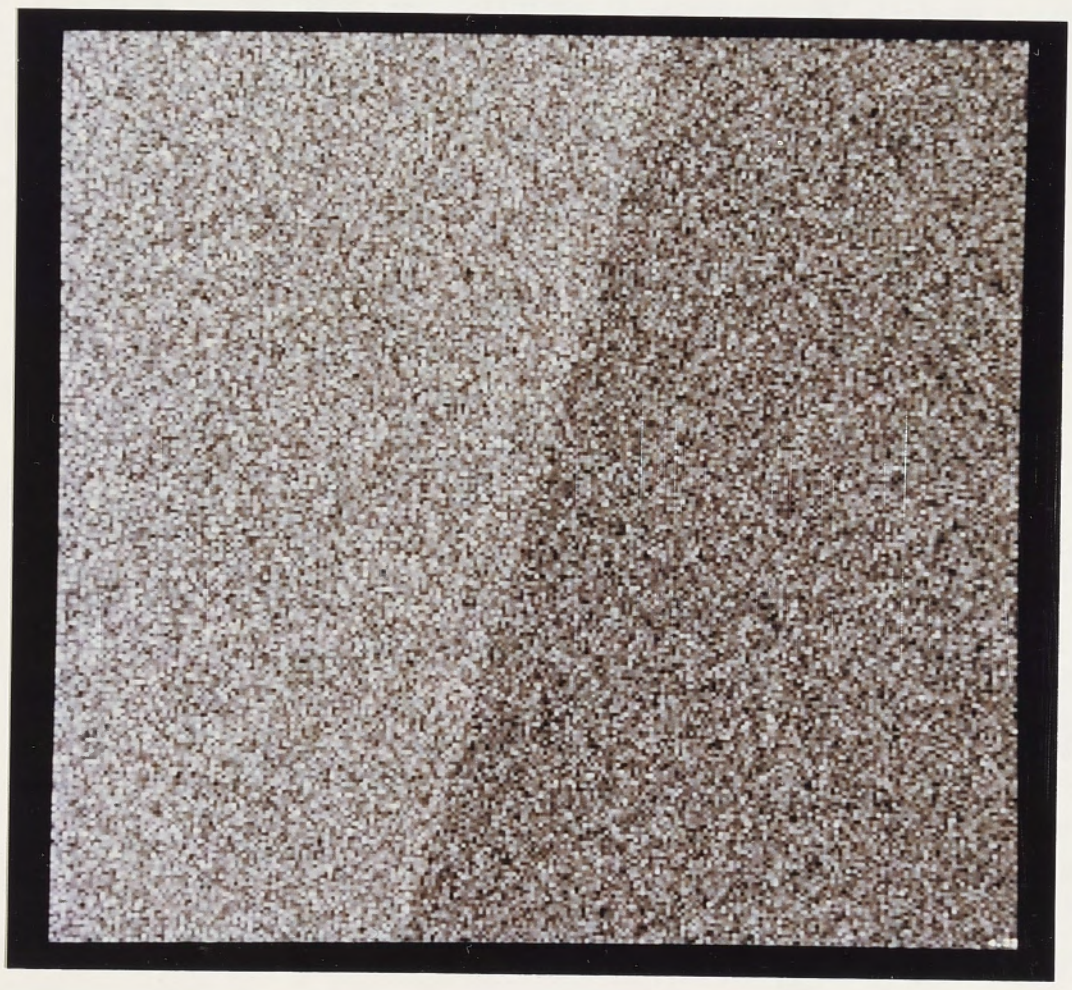


Figure 4.33: Image `noise`

Figure 4.34: Results from BLM algorithm for image `noise`

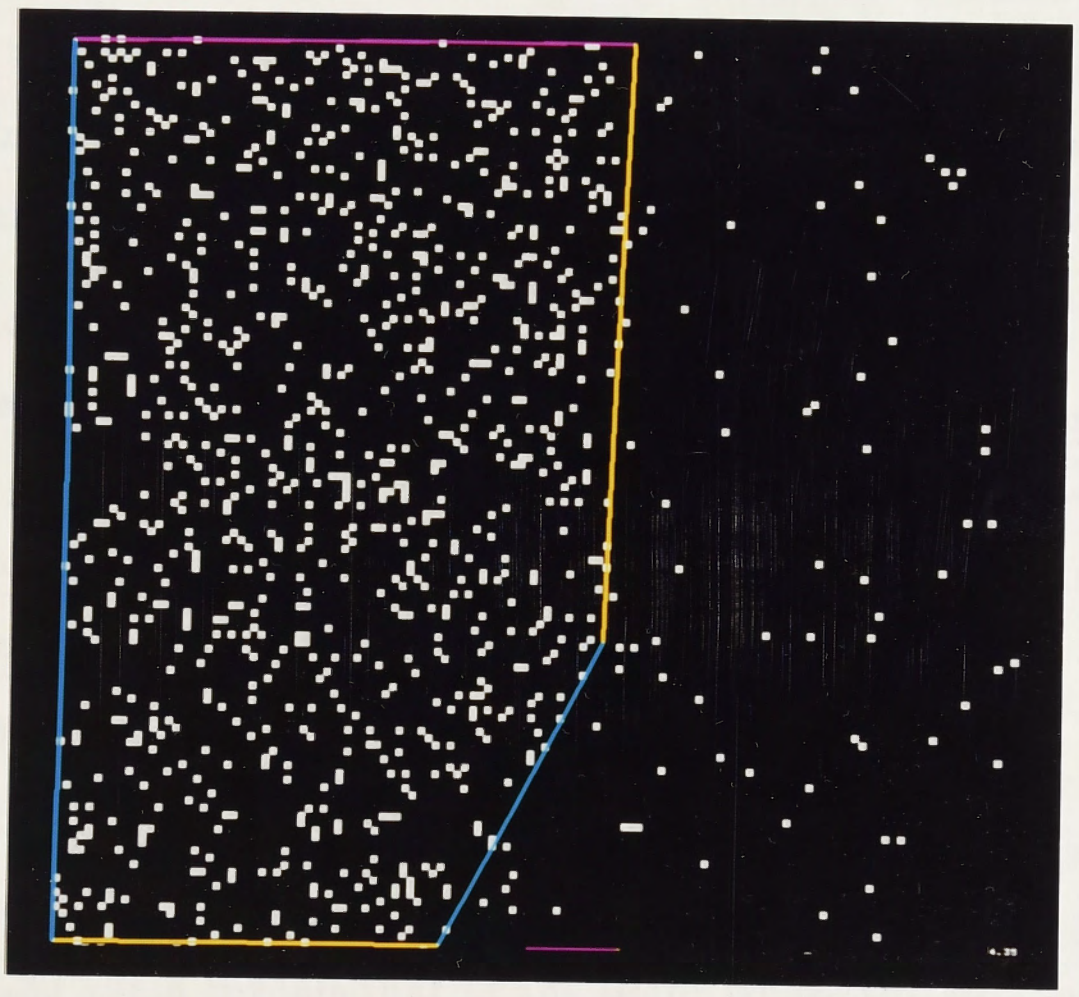


Figure 4.35: Results from BLM algorithm for image `dots`

The 'texton' theory of human texture perception (Julesz and Bergen, 1983) proposes that textures are discriminated on the basis of differences in density of textons, which are various types of salient image feature. If each texton of a given type was marked with a dot, the resulting image might be something like image `dots`. The BLM algorithm could then be used to locate the boundaries between distinct regions.

4.2.1.9 Image `straw`

This image illustrates the use of a texture filter prior to the BLM algorithm. The two halves of this image (figure 4.36) have the same intensity distributions, since they come from the same original image (the 'straw' texture in the Brodatz collection). The right-hand region comes from the straw texture rotated through 90°. The strong directionality of the straw texture gives rise to the discrimination of the two regions on the basis of the orientation difference. Human perception of the boundary location is also assisted by the endings of the pieces of straw (especially on close inspection). The true boundary in this image was obtained in the same manner as described for image `gmwave`, and is shown in figure 4.37.

The output of the texture filter is shown in figure 4.38. (The filter used is discussed below.) Note that the orientation difference has been converted to an intensity difference, but that the two regions are still textured. This filtered image was initially segmented with the split-and-merge procedure, giving the result shown in figure 4.39. The BLM algorithm then gave the result shown in figure 4.40.

The results shown in figure 4.40 appear to indicate more detail than is perceivable. However, the confidence intervals on the segment positions in figure 4.41 indicate that the segment positions are not tightly constrained. The confidence intervals are magnified by a factor of 3, and the contours by a factor of one. In particular, the lower segments are not as clear as the upper segments. This is partly because they are shorter, and partly because the boundary is quite fuzzy there.

The reported boundary deviates most from the true boundary in the segment at the top of the image, where a curved section is approximated by one long linear segment. Inspecting figure 4.36 again reveals that at the point where the true boundary is to the right of the reported boundary, the image detail has no strong directionality. This area is therefore somewhat ambiguous. Human perception appears to rely on the endings

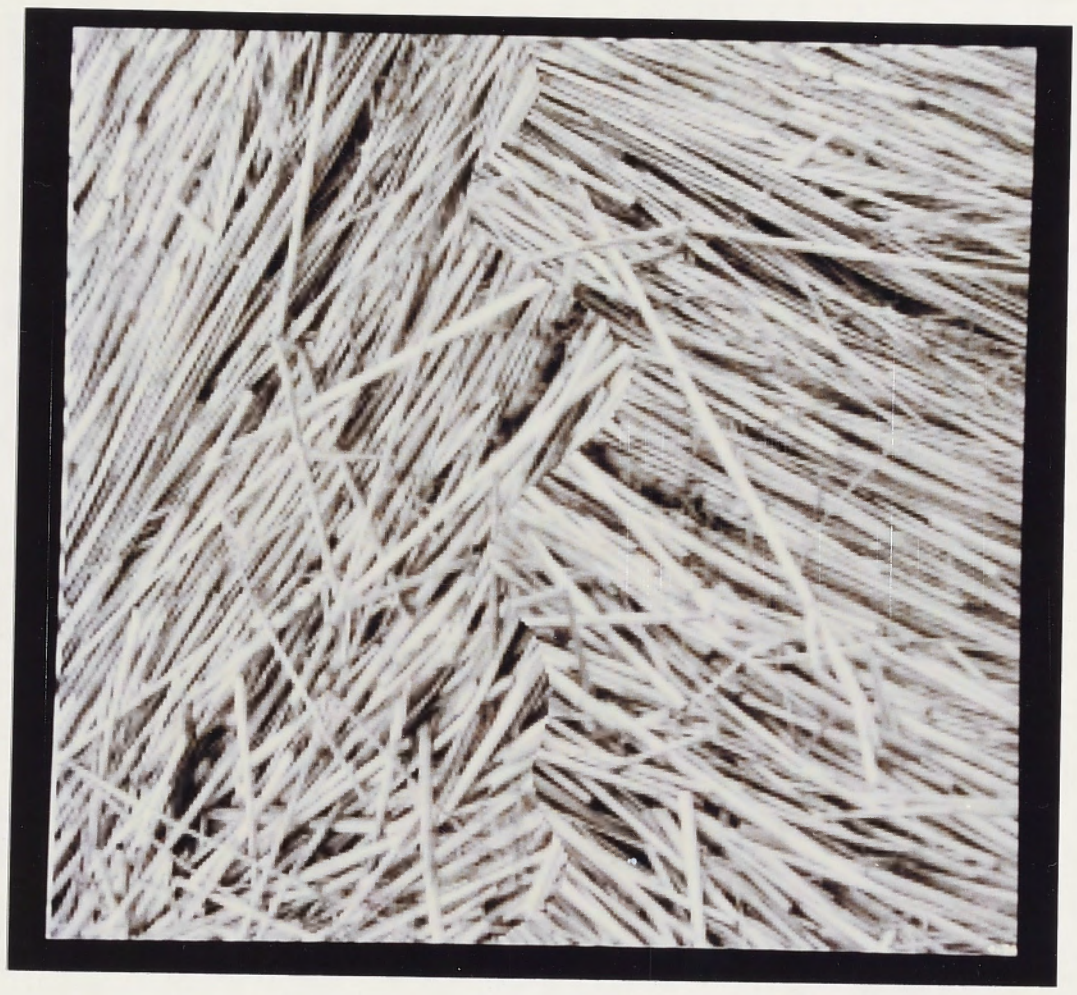


Figure 4.36: Image straw

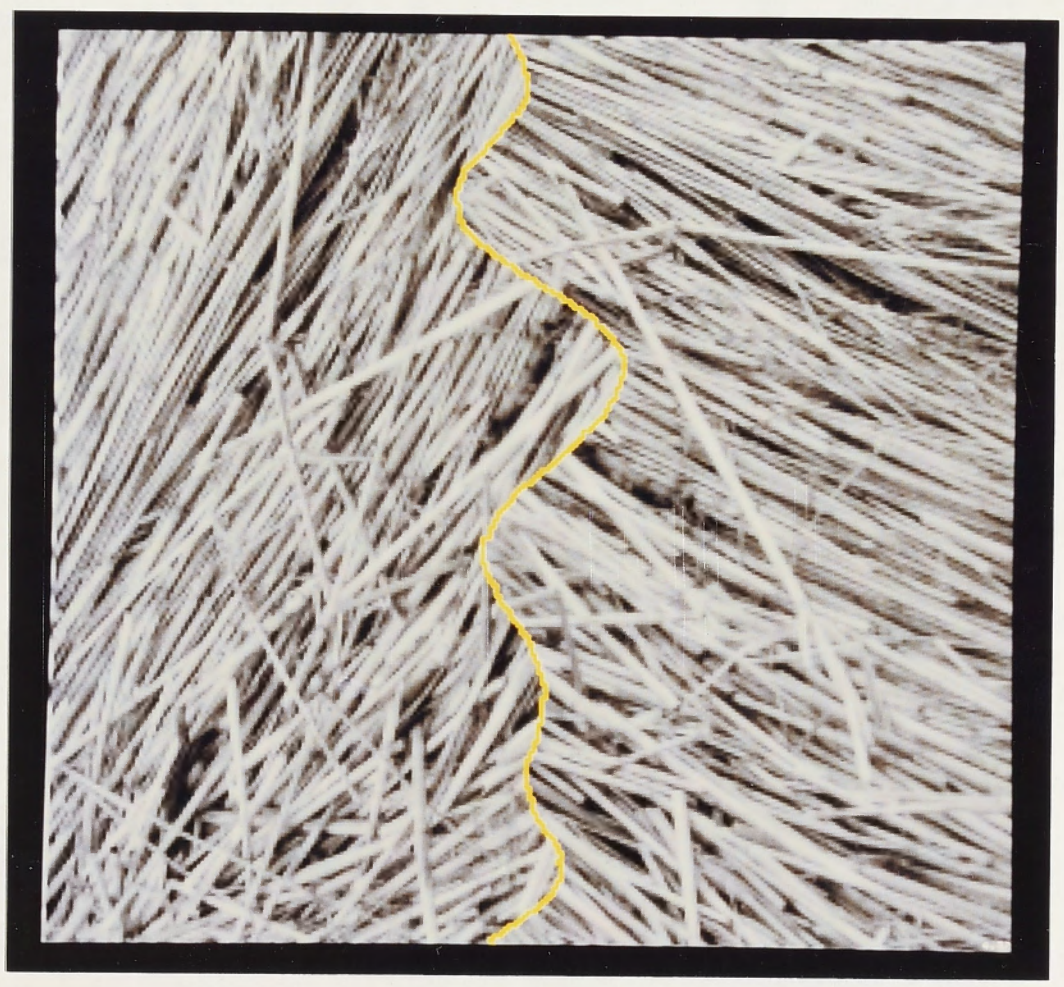


Figure 4.37: True boundary position in image straw

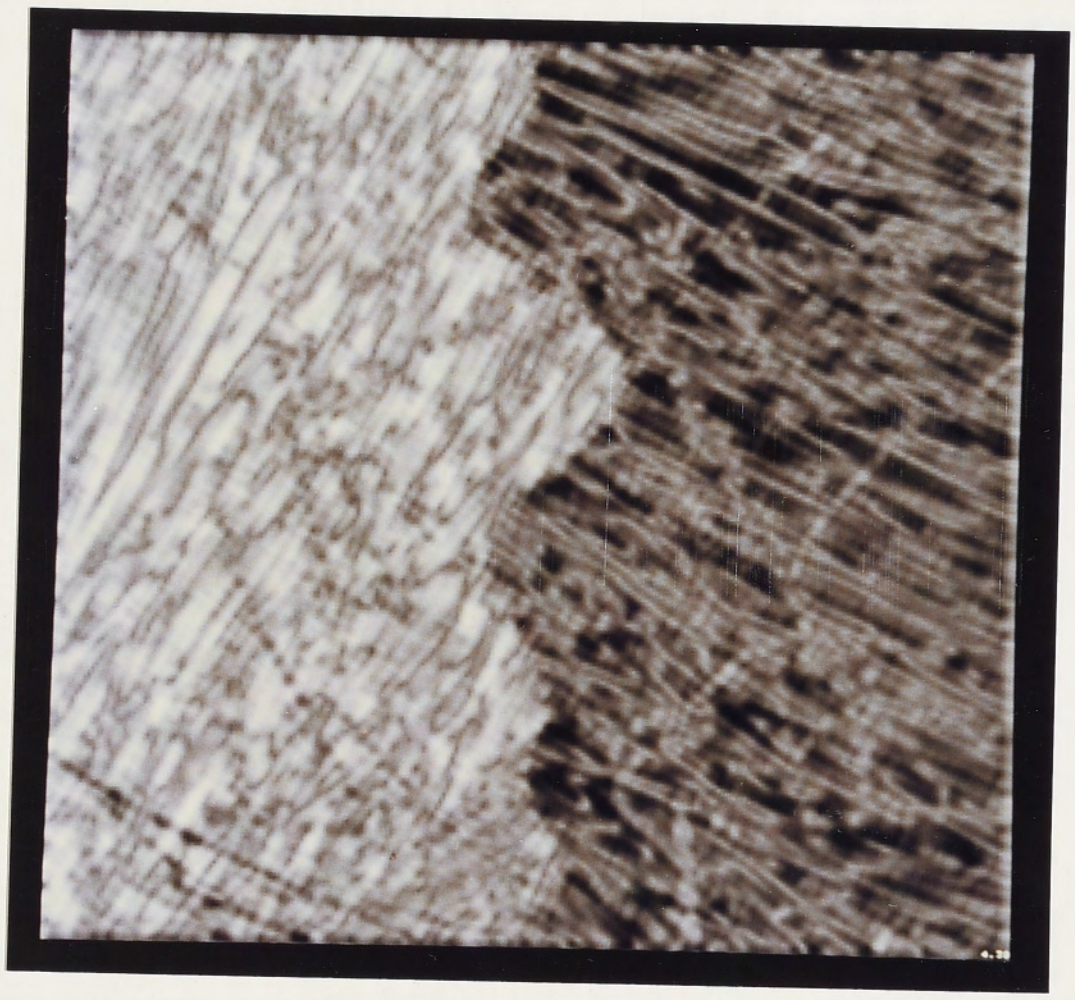


Figure 4.38: Texture filter output for image straw

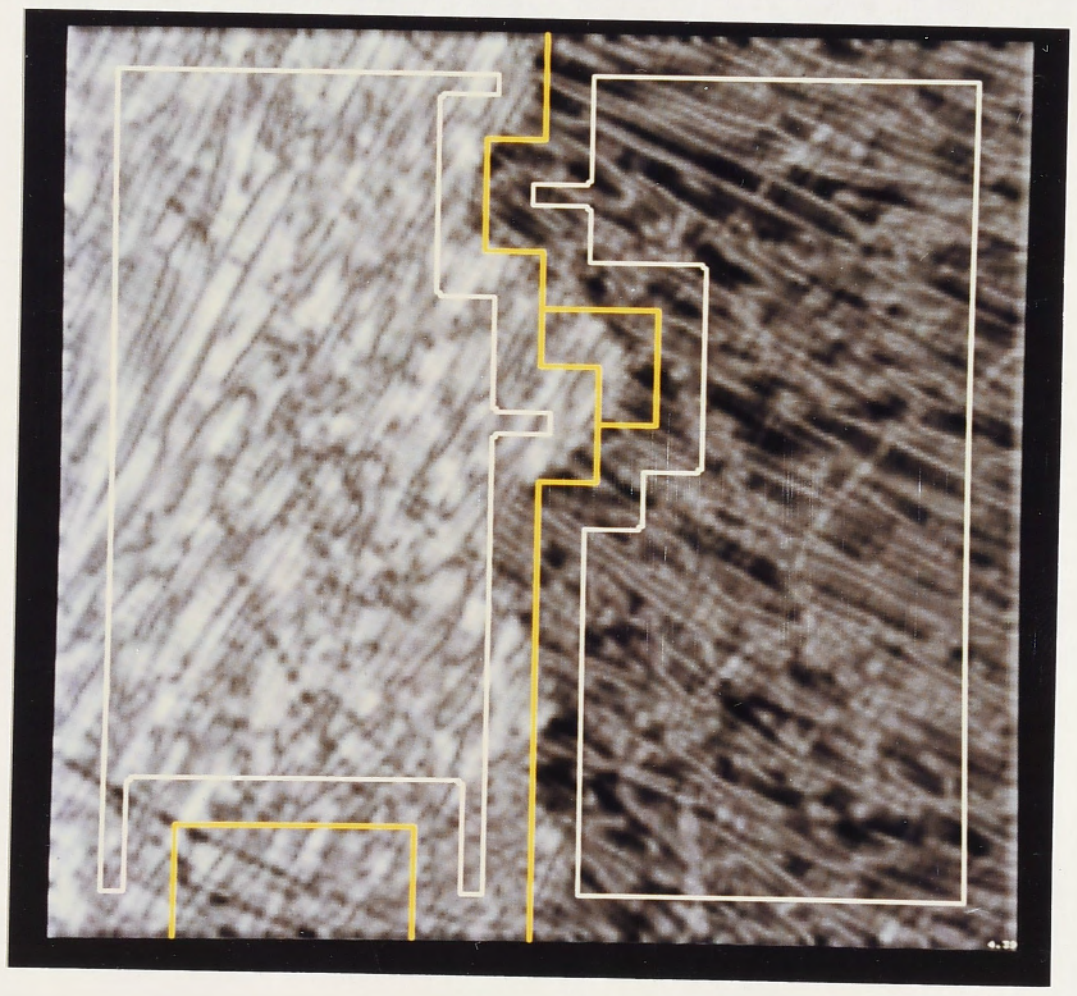


Figure 4.39: Split-and-merge results for the image in figure 4.38

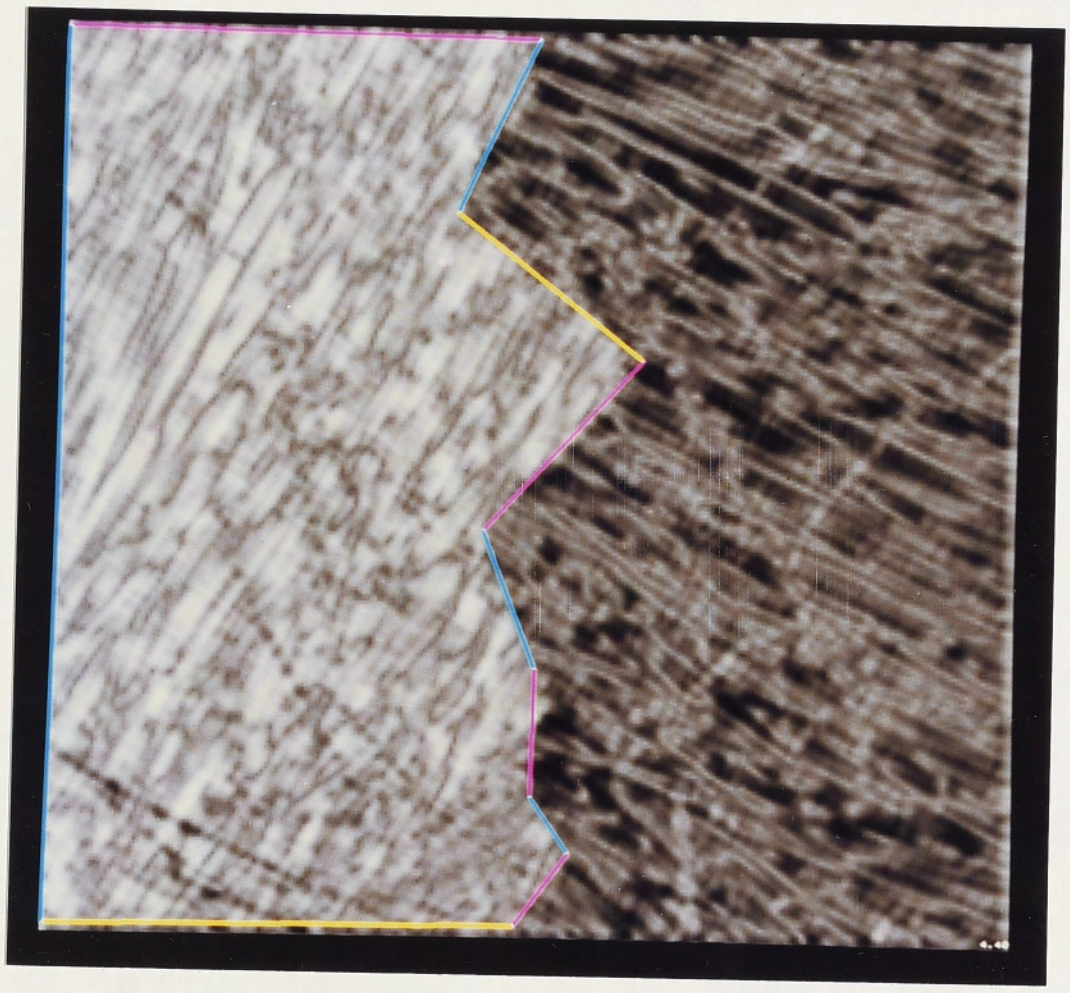


Figure 4.40: Results from BLM algorithm for the image in figure 4.38

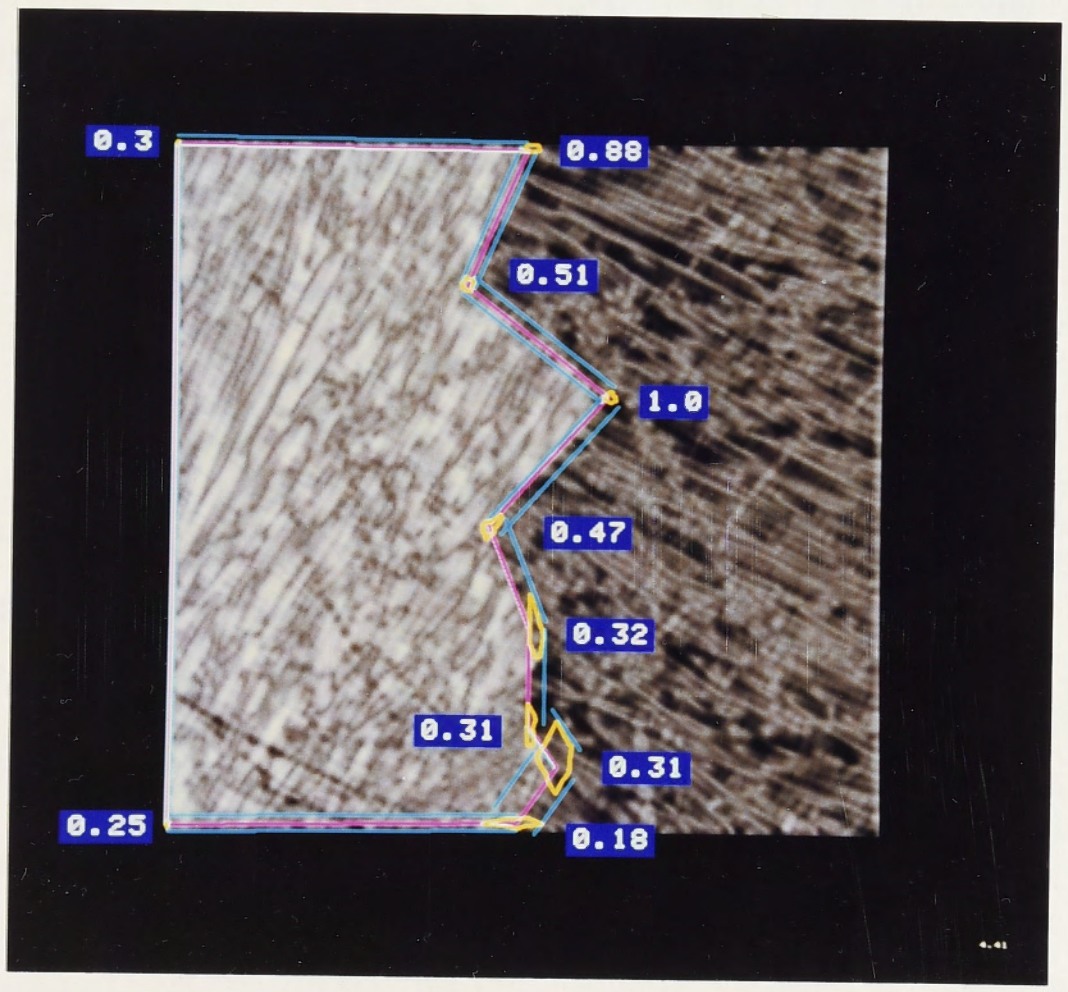


Figure 4.41: Confidence intervals (magnified by 3) and contours for figure 4.38

of the linear features in the image as a cue to the fine boundary location. The filter used here is not sensitive to such features.

The filter used involves convolution with Gabor function masks. These masks are two-dimensional sinusoidal gratings modulated by a Gaussian impulse. They give a good description of the properties of the 'simple' cells in the mammalian visual cortex (Marčelja, 1980). The masks are described by

$$G_C(x,y;\sigma,f,\phi) = \exp\left[-\frac{x^2+y^2}{\sigma^2}\right]\cos\left[2\pi f(x\sin\phi - y\cos\phi)\right] \qquad (4.1a)$$

$$G_S(x,y;\sigma,f,\phi) = \exp\left[-\frac{x^2+y^2}{\sigma^2}\right]\sin\left[2\pi f(x\sin\phi - y\cos\phi)\right] \qquad (4.1b)$$

where σ controls the extent of the Gaussian modulation, f gives the frequency in cycles/pixel, and ϕ gives the orientation; $\phi = 0$ gives a mask most sensitive to horizontal features. As the straw texture contains features at approximately 60° to the horizontal, two orientations were used: $\phi = 60°$ and 150°. The frequency f was 0.25 cycles/pixel, and σ was 1.8. These particular values were chosen on the basis of the characteristics of the textures. In order to reduce the computational effort required, only these values were used, rather than a range of values. In general, a bank of such filters at regular intervals of orientation, at a range of frequencies, would be used, as indeed the mammalian visual cortex appears to do (Sakitt and Barlow, 1982).

The image was filtered with four masks ('cos' and 'sin' masks at two orientations), giving images $C_{60°}$, $S_{60°}$, $C_{150°}$, and $S_{150°}$. These were combined into two images by taking the magnitude of the response at each orientation:

$$M_\phi(x,y) = \left(C_\phi^2(x,y) + S_\phi^2(x,y)\right)^{\frac{1}{2}} \qquad (4.2)$$

for $\phi = 60°$ and 150°. The final filtered output, shown in figure 4.38, is the difference $M_{60°} - M_{150°}$. This subtraction serves to sharpen the orientation specificity of the masks; it is quite similar to the phenomenon of cross-orientation inhibition in the mammalian visual cortex (Morrone *et al.*, 1982), where cells tuned to a particular orientation are inhibited by cells tuned to the perpendicular orientation.

Thus, this particular filter has been chosen to be similar to the filtering operations which occur in biological visual systems; only one filter has been used, but it is of a type which could be expected to be used in a practical texture-analysis system. Indeed, Turner (1986) obtained good results using Gabor functions for texture analysis; he also combined the 'cos' and 'sin' terms into a magnitude term as in equation (4.2).

4.2.1.10 Image rock

This image was used in testing the convolution edge detectors, and is shown in figure 4.7. The split-and-merge algorithm produced the results shown in yellow in figure 4.42, in which the region directly above the rock was split into two. These regions were combined into one, giving the areas outlined in white for evaluating statistics. This is necessary for the current implementation, but it actually increases the difficulty of the problem because the unified region is less homogeneous.

The piece-wise linear boundary found by the BLM algorithm (figure 4.43) follows the edge of the rock fairly closely. Some minor problems have occurred towards the right-hand edge of the image, however, where there is another rock, with some vegetation in front of it. The boundary has had to go around the dark patches, and has done this a little clumsily. The reason for these problems is that there are more than two distinct regions in this area of the image.

Figure 4.44 shows how the BLM algorithm fitted the left-hand portion of this boundary with five cubic segments, with slope continuity at two of the knots. Once again, the number of cubic segments and their slope continuity were specified manually. A bounding rectangle (not shown) was used to remove the confused area at the right-hand side of the picture from consideration.

4.2.1.11 Image house

This image, from the USCIPI data base, contains many distinct edges. However, we shall focus on the edge of the chimney, which manifests itself as a very subtle change in the texture of the brick wall. The edge of interest is the corner of the chimney, where the right-hand face perpendicular to the wall of the house meets the face parallel to the wall—see figure 4.45. The right-hand face has faint highlights.

This image also illustrates the use of a bounding polygon to restrict the BLM algorithm to the area of interest. This bounding polygon, entered manually, is also shown in figure 4.45, along with the areas for evaluating statistics outlined in white. Within this region, the boundary found by the BLM algorithm is shown in figure 4.46. Notice that the boundary of interest has been fitted with a single straight line, which is quite reasonable given the small difference between the two adjacent regions. The edge of the chimney might be expected to be vertical, but it is not; there is evidence in

Figure 4.42: Split-and-merge results for image `rock`

Figure 4.43: Results from BLM algorithm for image `rock`; piece-wise linear boundary

Figure 4.44: Results from BLM algorithm for image `rock`; five cubic segments

the image that the width of the chimney is not constant, but rather becomes thicker towards the bottom—as the shape of the shadow on the left-hand window shows. In fact, the edge of the chimney seems to be something like that shown in figure 4.47. This shape was entered manually on the basis of faint cues; the corners were positioned by following the lines of bricks across from the left-hand edge of the chimney, which is very slightly more distinct. Comparing this with the boundary found by the BLM algorithm, it is clear that the BLM algorithm has in fact done remarkably well, just on the basis of the difference in pixel grey-level distribution between the front and side of the chimney.

4.2.1.12 Image `pinetree`

Two aspects of the BLM algorithm are illustrated here: the ability of the piecewise linear boundary to represent a very complex shape, and some of the difficulties experienced when there are more than two connected regions. This image is taken from the top left-hand corner of an image from the USCIPI data base, showing a lake with pine trees on its shores. The results from the BLM algorithm are given in figure 4.48. The boundary of interest is that between the pine trees and the sky. This is a very

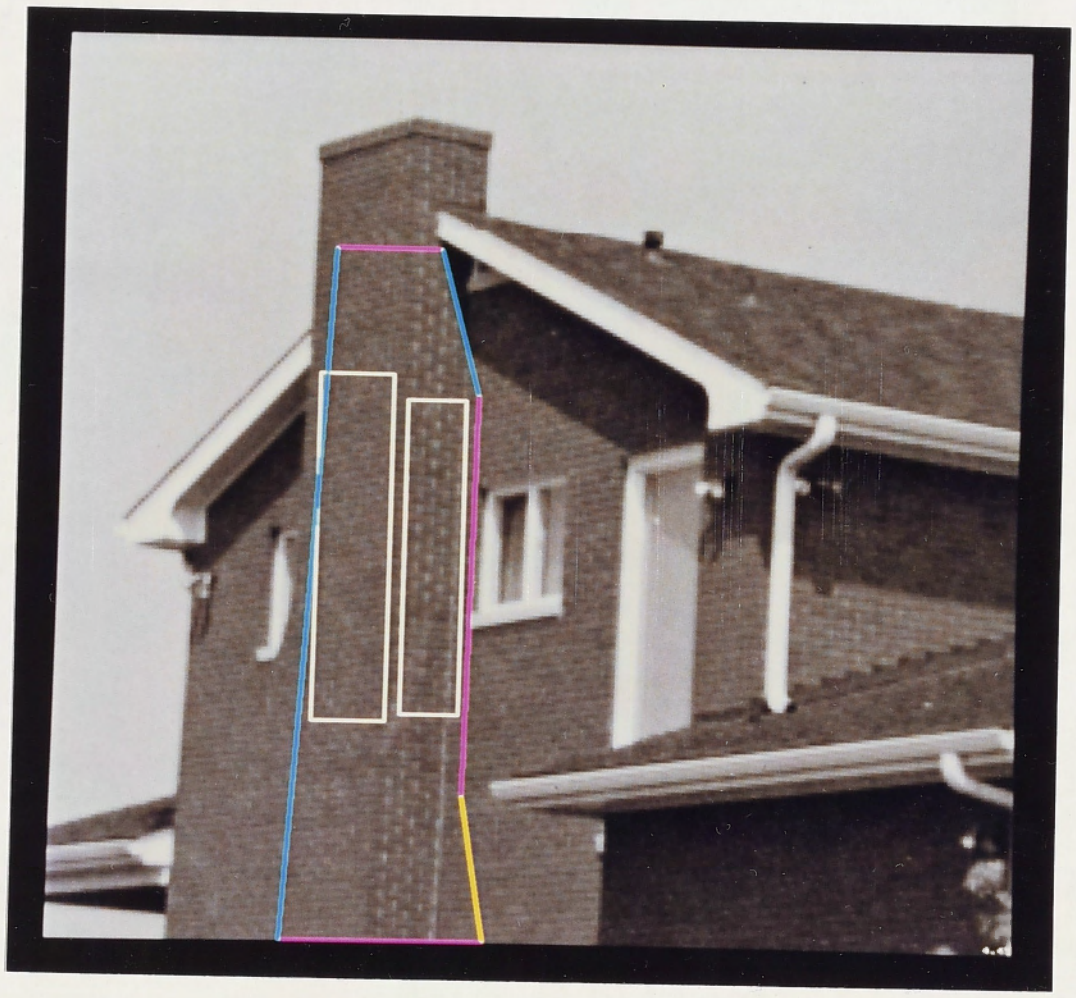


Figure 4.45: Image house with bounding polygon and areas for statistics

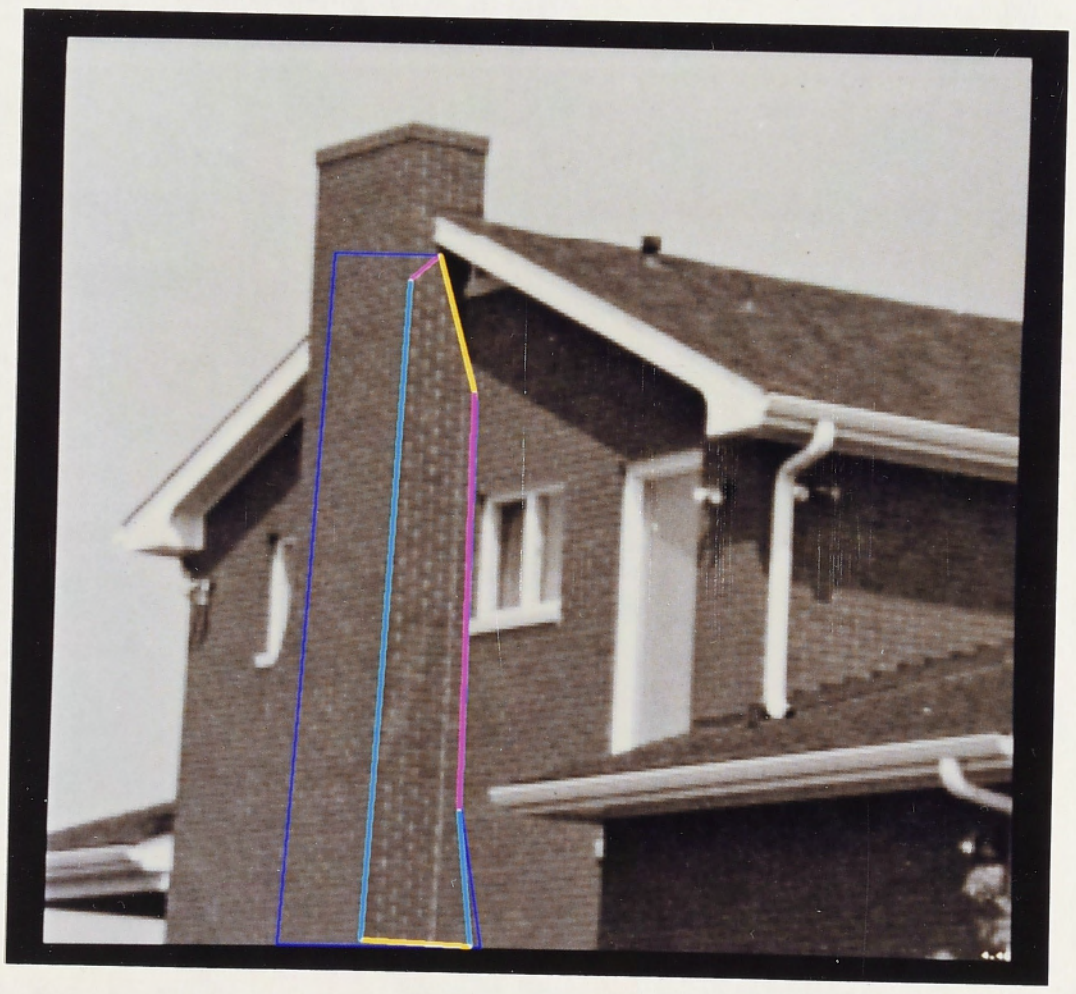


Figure 4.46: Results from BLM algorithm for image house

Figure 4.47: Edge of chimney in image house

Figure 4.48: Results from BLM algorithm for image pinetree

distinct, high contrast boundary, so there is very little ambiguity about which side any pixel is on; consequently, the boundary is tightly constrained, and many segments are needed (a total of 81 segments in the result shown in figure 4.48). Clearly, the reported boundary follows the edge of the pine trees quite closely, yet the result does not seem to correspond closely with human perception; probably because the tree is perceived as an aggregate of dark clumps, rather than as a region with a distinct boundary.

The split-and-merge algorithm split the right-hand region into two, corresponding to the clear sky and the clouds. As with image `rock`, these two regions were manually combined into one.

4.2.1.13 Image `trunk`

This image is part of a photograph of some Australian forest, showing a eucalyptus tree standing in a grassed area. For this image, a bounding polygon was used to restrict attention to the edge of the tree. The patches used to estimate statistics (obtained with the split-and-merge algorithm), and the bounding polygon, are shown in figure 4.49. The results (figure 4.50) show a boundary which follows the trunk of the tree quite closely. Four segments were sufficient, as the regions are not particularly distinct.

4.2.2 Conclusions about the BLM algorithm

The results shown above demonstrate that the BLM algorithm performs exceptionally well on a wide range of images. In most cases, the results correspond closely to human perception of the boundary shape. The reported boundary position is very accurate for the images, such as `gmcorner`, where the regions are distinct; for those, such as `gmwave`, where the true boundary is complex and the regions are not distinct, the results are as accurate as can reasonably be expected. The results are generally more complex for the images where the regions are more distinct, and in all cases are only as complex as can be justified by the clarity of the image data.

In comparison with traditional approaches, the BLM algorithm produces results which are dramatically superior, both in qualitative and quantitative terms.

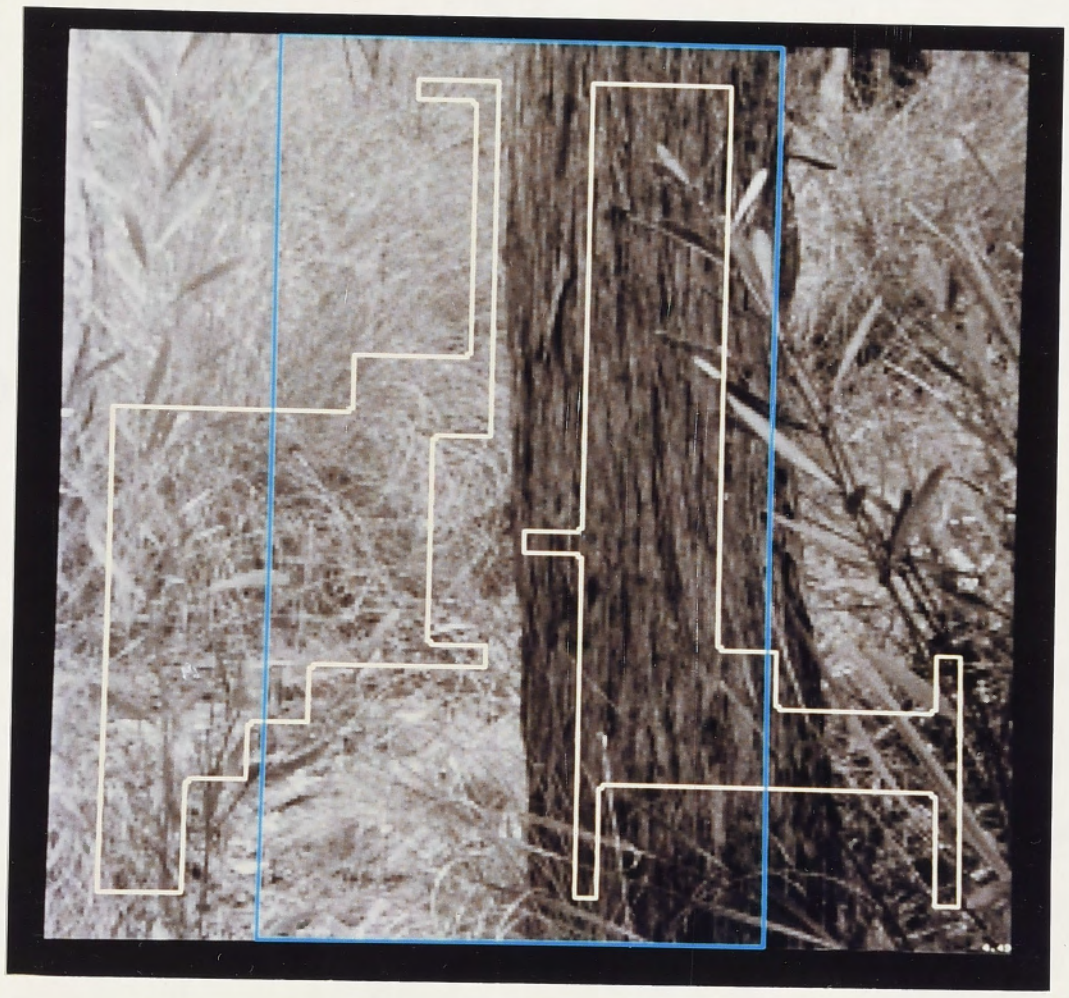


Figure 4.49: Image `trunk` with bounding polygon and patches for statistics

Figure 4.50: Results from BLM algorithm for image `trunk`

Chapter 5

Discussion

Previous chapters have presented a new approach to boundary location in textured imagery and contrasted the results of the the new approach with previous methods. The following sections examine some broader aspects of the BLM algorithm, the reasons for its improved performance, and possible extensions of the approach.

5.1 Appraisal of BLM algorithm

5.1.1 Discussion of performance

The results in the previous chapter demonstrate that the BLM algorithm performs very well on a wide range of images. Generally, the results are both simpler, and more accurate, than those obtained with other techniques. Some notable points are:

- The results are generally very accurate; they correspond closely to the true boundary in the artificial images. They show neither the unnecessary and incorrect undulations present in the output of convolution edge detectors, nor the 'blocky' characteristics of the boundaries of the regions found with the split-and-merge technique, nor the fragmentation of regions observed with thresholding or window classification techniques.

- The boundary found by the BLM algorithm is the simplest possible boundary which is still tenable (within the class of boundary shapes available to it). It is almost always simpler than the boundary found by the convolution edge detectors. It is certainly simpler in the sense that the parametric representation generally requires fewer bits than either a chain-coded or an image-based representation

the boundary (except possibly for an extremely intricate boundary, or for regions only a few pixels across).

- The boundary shape is *not* perturbed by irrelevant intensity variations within the textures.

- The agreement with human perception is generally excellent, and much better than previous techniques. There are, however, some discrepancies:
  - Where the boundary is unclear, human perception does not report a clear position (e.g., image **gmwave**), whereas the BLM algorithm has to report some definite position. However, the fact that the boundary is not clear can be determined from the confidence contours for the knots and/or the confidence intervals for the segments. This discrepancy is partly due, therefore, to limitations in the display of the reported boundary for human inspection.
  - Where a smooth undulating boundary is sufficiently unclear that the piece-wise linear representation is fairly coarse (e.g., image **straw**), the boundary will be perceived without the corners which are (apparently) present in the piece-wise linear representation. The confidence contours and intervals should indicate that the knots do not correspond to sharp corners in the true boundary. Clearly, the human perceptual apparatus does not use piece-wise linear boundary approximations; perhaps more extensive use of cubic polynomial segments would reduce this discrepancy.

- The current implementation can give incorrect results on images where the regions are not homogeneous, or where there are more than two regions, simply because these images do not conform to the assumptions that are made in the current implementation. Section 5.3 discusses how these limitations could be overcome (within the framework of the BLM algorithm).

5.1.2 Reasons for good performance

The major reason why the BLM algorithm produces such good boundary representations is that it explicitly attempts to find a good interpretation of the image data. The view of segmentation as being an act of interpretation is therefore central to the

success of this technique. Within this framework, four reasons for the good performance of the algorithm can be identified:

1. The criteria of simplicity, accuracy, and consistency successfully capture the essence of a 'good' boundary hypothesis.
2. The formalization of the criteria of accuracy and consistency in statistical terms is both appropriate and effective.
3. The framework of the BLM algorithm allows relevant statistical knowledge to be applied effectively.
4. The implementation of the algorithm is generally effective in finding a boundary which satisfies the three criteria.

These points are discussed in detail below.

The criteria of simplicity, accuracy, and consistency have considerable intuitive appeal. Unnecessary complexity in the boundary description can only create difficulties for further processing, and a description which is inaccurate, or inconsistent (that is, detectably wrong), is undesirable. The three criteria are each aspects of a desirable boundary description; the results indicate that, together, they are sufficient to constrain the boundary locator to produce excellent boundary descriptions.

Given these criteria, the next important aspect is how they are formalized and quantified for use in an algorithm. Formalizing the simplicity criterion virtually forces the use of a parametric boundary representation. Such representations generally have the advantage that the number of bits required depends on the shape of the boundary, not its orientation or size, and corresponds to an intuitive idea of how 'complicated' the boundary shape is. In contrast, neither chain-coded nor image-based representations have these desirable properties. The number of bits required for a chain-coded representation depends on the length of the boundary, not on how much it changes direction; image-based representations are even less appropriate, requiring a large, but fixed, number of bits.

The consistency and accuracy criteria require a statistical formalization when dealing with textures, because the regions do not contain a single intensity level, but rather a distribution of intensities. In most cases, the pixel intensity distributions will overlap, leading to ambiguity about which region some pixels belong in. The degree of ambiguity of a pixel will be a function of how often pixels of the same intensity occur in

each region. (This ambiguity is at the root of the small-region problem in region-based segmentation methods.) Quantifying this ambiguity as a likelihood ratio enables the BLM algorithm to give each pixel the weighting that it deserves, given its level of ambiguity. Formalizing the accuracy criterion in terms of maximizing the total likelihood of the boundary therefore means that the boundary represents the best possible resolution of the ambiguity of the pixels near the boundary. All the available information is taken into account, since each pixel near the boundary can contribute to determining its position, and so the most likely boundary can be expected to be accurate.

For similar reasons, the consistency criterion is formalized in terms of a likelihood-ratio test of hypothesis. Considerable care was taken in formulating the test to ensure that the 'false-alarm' rate was standardized. (The false-alarm rate is the probability that a patch of the image will be rejected, even though it does belong to the region it is in.) Successfully controlling this false-alarm rate requires that the correlation between nearby pixels be taken into account. The test used in the BLM algorithm does this, as described in section 3.4.5, which results in a test which is both powerful and reliable. It is reliable in the sense that the rate of false alarms is controlled to be virtually zero, and it is powerful in the sense that the threshold used in the test is always kept as low as possible, consistent with an acceptable rate of false alarms. Thus, the test adapts to the particular textures in the regions so that it is as sensitive as possible without making mistakes.

Statistical characterization of the regions is an essential ingredient in the BLM algorithm's performance. This statistical information represents extra knowledge which is brought to bear in locating boundaries. Application of this knowledge enables the BLM algorithm to take account of the textures, distinguishing the intensity variations which are merely part of the textures from those associated with the boundary. This statistical knowledge, and its application through the criteria of accuracy and consistency, is one of the main reasons why the BLM algorithm performs so much better than convolution edge detectors.

5.1.3 Appraisal of the implementation

It may seem that the BLM algorithm is complex, and likely to be computationally expensive. In fact, it is not particularly expensive, when compared to the time taken for convolution of the image with an edge detector profile of reasonably large support.

The processing time required for the BLM algorithm depends on the complexity of the boundary, but is usually of the order of minutes, on a VAX 11/750, for a piece-wise linear boundary on an image of size 256×256 pixels. This time is quite similar to the time required for the Marr-Hildreth and Canny edge detectors (typically 5 minutes for each convolution). The optimization routine for cubic segments is much slower, but there is considerable room for improvement. The BLM algorithm is certainly more complicated than a simple convolution edge detector; the current implementation occupies approximately 8000 lines of code in the 'C' language. However, it is probably no more complicated than the complete process of detecting edge pixels, linking them up, and approximating the resulting strings of edge pixels with some parametric representation. The approach which has been taken in this research is that it is more important to discover how to obtain accurate and reliable estimates of the locations of boundaries (since no previous methods give satisfactory results) than to restrict attention to fast algorithms.

5.1.3.1 Optimization phase

The results indicate that the optimization phase generally does very well in finding the desired maximum of the likelihood for piece-wise linear boundaries. To a large extent, its ability to pass over local maxima is provided by its ability to combine information over a reasonably large area of the image. This area is determined by the lengths of the segments adjacent to each knot and the length of the optimization lines. The parametric representations of the boundary are therefore a major factor in the good performance of the optimization phase; the segments in a parametric representation are generally much longer than they would be with a chain-coded representation.

The task of the optimization phase is not to find the globally most likely position over the whole image, but rather to find the most likely position for the boundary between the two regions of interest, which are known to be connected. This is a considerably easier task, because the boundary can generally be started at a rough approximation to its correct position, so that the likelihood is continually increasing as the boundary moves towards the correct position. The hill-climbing technique used in the current implementation is therefore quite adequate.

Note also that the use of the original image information in the optimization step, rather than an abstraction such as an edge map, makes the optimization much more

reliable and straightforward. An edge map only contains information near the true position of the boundary: if the current boundary is relatively far from the true boundary, there is little information to indicate which way the boundary should move, that is, the 'force' on the boundary is weak or non-existent. In contrast, using the original image information means that the 'force' is (on average) constant throughout the whole of each region adjacent to the true boundary. Consequently, for the case of two regions where a closed boundary is used, enclosing (say) the left-hand region, the optimization can be started with the initial boundary estimate enclosing only a small patch inside that region. Each segment will move outwards, since that increases the likelihood by an amount proportional, on average, to the area thus brought inside the boundary. Therefore, the boundary will continue to expand until it reaches the true boundary.

In the optimization phase, the pixels are assumed to be independent. While this assumption is not, strictly speaking, correct, the likelihood maximization still produces good results. The inter-dependence of the pixels means that the 'likelihood' is really only a pseudo-likelihood; calculating the true joint likelihood of a set of correlated pixels requires the use of a Markov random field characterization. The effect of assuming pixel independence is mainly that clumps of similar pixels are given more weight than they deserve. This has not caused any problems in practice; it should cause problems only when the lengths of the boundary segments are similar to the sizes of the clumps. For relatively long boundary segments, such clumps do not cause significant problems. The reason for this is that the effects of such clumps along the boundary tend to balance out.

Optimization of cubic segments in the boundary is quite slow, in the current implementation, due to the difficulty of searching the space of up to 6 dimensions for the parameters at each knot. Further research into efficient algorithms for optimizing cubic segments is required.

5.1.3.2 Consistency test

The consistency test is a key element of the BLM algorithm, since it essentially controls how many segments the final boundary has. The current implementation of the consistency test works very well. One of the best features is that the threshold `siglev`, controlling the significance level of the test, can be set at a constant value. During evaluation of the BLM algorithm, this threshold did not have to be changed

for any image. In all cases, the value of 5.0 standard deviations from the mean was used, which gives a sensitive test with a false-alarm rate of practically zero (certainly, no false alarms were observed).

The implementation of the consistency test is based on the assumption that the distribution of the log likelihood-ratio (LLR) statistic is Gaussian when the null hypothesis is true (i.e., when the patch really does belong on the side to which it has been assigned). This is quite a reasonable approximation for large patches (as would be expected from the Central Limit theorem (Freund, 1972, p206)) and for natural textures. It may, perhaps, break down for small patches of textures with strongly non-Gaussian intensity distributions.

The usual threshold value of 5.0 may seem rather high, given the assumed Gaussian distribution. One might expect a threshold value of 3.5 or 4.0 to give an acceptable false-alarm rate with better sensitivity. There are two reasons why the higher value is used, both relating to the fact that the rate of false alarms must be practically zero—a false alarm will introduce unnecessary complexity into the boundary. The first is that the distribution may deviate from the assumed Gaussian shape. The second is that, even when the Gaussian assumption is true (or a good approximation to the truth), the relationship between the observed false-alarm rate and the threshold depends on the method used to determine the patches to be tested. Because only patches which are likely to fail are tested, a much higher proportion will fail the test than the proportion that would fail if the patches were chosen purely at random. This proportion is given by the probability in the upper tail of a Gaussian distribution beyond the threshold. For example, the probability that a normalized Gaussian variate will be greater than 5.0 is only 2.9×10^{-7}, but the observed false-alarm rate in the test with a threshold of 5.0 will be higher than that, because the test only chooses patches with a relatively high LLR, and thus cuts off the lower end of the distribution.

This is the reason why a higher threshold had to be used in comparing the results from the convolution edge detectors with an optimized region mask. The optimization routine used in that case has much more freedom to search out patches with high values of the LLR, and therefore concentrates on the upper end of the (assumed) Gaussian distribution to a greater extent than the procedure used in the BLM algorithm.

Strategy for finding protrusions. The procedure used in the BLM considers displacing small pieces of a boundary segment out from the segment, searching for the most likely position for the piece. Two options exist: (1) find the most likely position within a given distance (a global optimum over the distance considered), which is the method currently used, or (2) move the piece away from the segment only while the likelihood increases (finding a local optimum). The first method gives a more sensitive test, and one which is less susceptible to noise (since it can pass over textural intensity variations in finding the largest possible area which is (apparently) on the wrong side of the boundary), but it can give rise to problems if one of the regions contains another embedded region, close to the part of the boundary being tested. This situation does not conform to the assumptions made in the current implementation of the BLM algorithm, but is reasonably common in real-world images. Section 5.3 describes an extension of the scheme to handle multiple regions; the second method would probably be preferable with the extended scheme.

Calculating autocovariances. Applying the test to a given patch requires calculating the variance of the LLR statistic under the null hypothesis. This variance is calculated using the autocovariances of the λ-image, as described in section 3.4.5.3. This approach works well, provided that a reasonable area of each texture is available for calculating the autocovariances, that is, an area at least four or five times as broad as the maximum span over which autocovariances are calculated (controlled by parameter `autodist`). If insufficient area is available, the estimates of the autocovariances will be unreliable, and it is possible for the calculated variance for a region to be negative. Since the variance must be positive, such a result clearly indicates that the autocovariances are incorrect, and that therefore `autodist` is too large. On the other hand, if substantial correlation exists over a considerable fraction of the breadth of the region, then there is a strong case for saying that the variations are too coarse to be considered as texture, that is, that the region is not homogeneous.

Note that these autocovariances are of the pixels in the λ-image, not the original image; thus, they depend on the function $\lambda(x)$, which depends on the estimated intensity distributions in *both* regions. In the case where there are more than two regions, the autocovariances will have to be calculated for each pair of adjacent regions (rather than just for each region). These autocovariances could be calculated from the grey-level dependence matrices (i.e., the second order statistics), but since a complete $N_G \times N_G$

matrix is required for each span for which the autocovariance is required, the matrices would consume a very considerable amount of storage. Using these matrices is therefore probably not a viable strategy.

5.1.3.3 Elaboration step

The task of the elaboration step is to increase the complexity of the boundary hypothesis if it is found to be inconsistent with the image data. For a piece-wise linear boundary, the only way to increase its complexity is to add more knots. The question is, how many, and where? The strategy in the current implementation is to add one extra knot in the middle of each inconsistent segment (i.e., half-way between the adjacent knots). Clearly, it makes most sense to add the extra knot(s) to inconsistent segments, but it is reasonable to ask, is there a more intelligent strategy than adding the extra knot at the half-way point? It may be possible to use the information contained in the locations and sizes of the inconsistent patches to guide where to place the extra knots, but experience indicates that such a strategy would have to be quite complex, and would probably be error-prone. The simple strategy of adding the knot at the half-way point was chosen because it has the advantage of tending to keep the segment lengths even, and it avoids the need for a complex algorithm to determine where to add the knots. This strategy may mean that up to three iterations are required for sufficient knots to be added in the right places to enable the boundary to move towards the true boundary, but it will eventually add the required knots. It may add more knots than necessary (as is discussed further in the following section).

In considering a more intelligent elaboration algorithm, at least three possible reasons for a segment to be inconsistent must be considered:

1. The true boundary is curved, and there are, as yet, too few segments to approximate it sufficiently well (e.g., image `gmarc` (figure 4.22, page 110)).
2. The true boundary contains a corner in the length spanned by the segment (e.g., image `gmcorner` (figure 4.1, page 89)).
3. The true boundary contains a protrusion, but is approximated for some of its length by the current hypothesis (e.g., image `gmprot` (figure 4.26, page 113)).

In case (1), the segment needs at least one extra knot, but there is no obvious way of determining whether one is enough (without trying it). One extra knot will, however,

allow some progress to be made. In case (2), one extra knot will probably suffice; in fact, there may be adjacent inconsistent segments which do not need an extra knot (i.e., this section of the boundary only needs one extra knot to be able to become consistent). In both these cases, adding the knot at the half-way point will usually be quite adequate, since the optimization phase should be able to move it to where it is required. However, in case (3), at least three extra knots are needed, and they need to be added near the protrusion, so that the middle one of the three can move up into the protrusion when the boundary is optimized.

There is no obvious way to determine which category an inconsistent segment belongs to. The shape and size of the inconsistent clumps may be some guide, but not a particularly reliable one. The reason for this is that the procedure which finds the clumps, being a relatively unconstrained likelihood maximization working on short pieces of the segment, tends to be influenced considerably by noise and textural variations, as well as by the position of the true boundary. (This is why the outline of the clumps is not particularly useful as a boundary hypothesis.) Consequently, it is difficult to decide where the best position to add extra knots is, or how many to add. The root of the problem seems to be that accurate knowledge of the true boundary is required to make the best choice, and this knowledge (almost by definition) is not available at this point. The simple strategy of adding the extra knot half-way along the inconsistent segment is therefore probably the best alternative.

Elaboration of cubic segments. When the alternative of using cubic segments exists, the elaboration step has now at least three choices: add extra piece-wise linear segments, add extra cubic segments, or change some segments from piece-wise linear form to cubic form. If the boundary already has cubic segments, more options exist, since the slope can be made continuous or discontinuous at a knot adjacent to a cubic segment. These options exist at each iteration of the algorithm, so the set of hypotheses to be considered forms a tree. The tree is not infinite, since a consistent hypothesis will be reached at some point down each branch, but it could be very large; a complete search of every option (i.e., a search with backtracking) is probably impracticable. It may be possible to use some kind of heuristic search. For the reasons outlined above, it would be very difficult to determine reliably whether a linear or cubic segment is required from the size and shape of the inconsistent patches.

Detailed algorithms for the elaboration and simplification steps, when the possibility of using cubic segments exists, have not yet been developed. There are two obvious strategies that could be used in searching for the simplest consistent hypothesis, to avoid the need for backtracking in the elaboration step: (a) use only piece-wise linear segments until a consistent hypothesis is established, and then simplify the hypothesis by using cubic segments where appropriate, or (b) use cubic segments exclusively until a consistent hypothesis is established, and then simplify by using straight-line segments where possible.

The first option would probably be the less expensive in terms of processing time, unless a drastic improvement can be made in the speed of optimizing cubic segments. It would have the advantage that information from the shape of the piece-wise linear boundary (i.e., the lengths of the segments and the angles between them) could be used to determine likely places for using one cubic segment in place of several linear segments, and also to give a good initial approximation for the shape of the cubic segment (which should speed the optimization). The 'cornerness' measure would also be very useful: those knots which do not correspond to definite, sharp corners are obvious candidates for being subsumed into a cubic segment. Note that a cubic segment (without slope continuity at either end) has the same complexity as three linear segments: the cubic segment requires two coordinates (four parameters) to specify its initial and final velocities. A cubic segment without slope continuity at either end, to be a preferable alternative, would therefore have to either (i) replace three linear segments, and have a higher likelihood, or (ii) replace four or more linear segments. Similar rules apply if slope continuity is maintained at one or both ends.

5.1.3.4 Simplification step

At first sight, it might seem that a simplification step should not be necessary, given that the BLM algorithm starts with a very simple hypothesis and elaborates it until it is consistent. However, as noted above, the elaboration step cannot always determine the least possible elaboration which will give a consistent boundary, since that would require knowledge of the true boundary which is not available. In practice, it may add more segments than are needed, in the process of adding those extra segments which are necessary.

The algorithm used for simplifying a piece-wise linear boundary, given in sec-

tion 3.4.6, represents a search with a limited degree of backtracking. Briefly, the algorithm for testing whether a knot can be removed is to remove it, re-optimize the boundary, and test for consistency. If the boundary is not consistent, it must be restored to the state before the knot was removed. In practice, this strategy is not particularly expensive in processing time. Those knots which are most likely to be necessary in the boundary are eliminated from consideration by a very quick test on the area enclosed in the triangle formed by the adjacent segments and a line between the adjacent knots. The knots which pass this test are therefore those where the adjacent segments are short, or where the angle between them is close to zero or 180°. The effect of removing such knots tends to be relatively local: that is, re-optimizing the boundary generally only moves a few knots near the knot which was removed; this optimization is therefore usually quite fast.

It is important to re-optimize the boundary when a knot is removed, before testing for consistency. A strategy which was used initially, and rejected, was to consider removing each knot, replacing it with a straight segment between the adjacent two knots, without re-optimization. If this segment was not consistent, the knot was restored. Furthermore, this procedure was performed on each iteration of the BLM algorithm. The problem with this strategy is that one could thus obtain a boundary which was simpler than the original, and consistent, but which was not the most likely. Subsequent re-optimization of the boundary could then give one which was more likely, but not consistent! Furthermore, when this simplification step was used within the iteration of the BLM algorithm, the possibility existed for the algorithm to get into an infinite loop adding and removing segments.

Where cubic segments are used, many more possibilities for simplification exist: for example, some of the additional options are:

- Convert a cubic segment to a linear segment
- Merge two adjacent cubic segments
- Merge a cubic segment with an adjacent linear segment
- Enforce slope continuity between two adjacent cubic segments
- Convert three or more linear segments to a cubic segment.

A detailed strategy for the simplification step under these circumstances has not yet been developed.

5.1.3.5 Boundary representation

In the framework of the BLM approach, the boundary representation has a critical role, because it determines which shapes are considered simple. Those shapes for which large sections can be fitted with a single segment will be considered simple, so to some extent the repertoire of segment types could be application-dependent. The segment types used in the current implementation of the BLM algorithm (linear and cubic) provide a versatile repertoire for fitting a wide range of shapes, and the complexity measure they provide corresponds well to an intuitive idea of how complex a given shape is. One possible exception to this statement is a circular arc—an arc would generally be considered simpler than an arbitrary cubic curve. This could be overcome by adding circular arcs to the repertoire of segment representations. Arcs would have five parameters: four for the endpoints, and one for the radius of curvature. (Four of these are shared with adjacent segments, giving effectively three parameters per segment.) If a section of the boundary was approximately circular, an arc would be preferable to a cubic segment, as being simpler (provided it was consistent).

However, increasing the repertoire of segment representations will increase the complexity of the elaboration and simplification steps. There are many possible segment representations which could be used, and it is necessary to select a suitable subset. Increasing the repertoire will enable a simpler description to be found, in some cases, at the cost of time spent searching through the repertoire for the simplest consistent description. If the images to be processed contain many instances of a particular type of curve, a representation for that type of curve should probably be added to the repertoire. In general, however, linear and cubic segments perform quite well (figure 4.24 on page 111 shows that a single cubic segment can approximate a circular arc very closely over quite a long distance).

5.1.4 Other aspects of the BLM algorithm

5.1.4.1 Behaviour as signal/noise ratio changes

An interesting and useful characteristic of the BLM algorithm is that the amount of detail in the boundary can be expected to *decrease* as the two regions become less distinct; that is, as the contrast between them decreases, or as the level of noise increases. With many other edge-finding algorithms, increasing noise will affect the boundary

shape more and more, causing the shape to become more complex, rather than less. However, with the BLM algorithm, a lower contrast between the regions will mean that the consistency test is less stringent, resulting in a less complex boundary. Thus, the BLM algorithm does not report more information than can reliably be extracted from the image. When the contrast between adjacent regions is low with respect to the variations within the regions (that is, the signal-to-noise ratio is low), the amount of information in the image is low, so those algorithms which report a complex boundary shape are reporting noise as 'signal', i.e., as useful information present in the image. This spurious information can only cause problems for subsequent processes using this data.

5.1.4.2 Boundaries with broad transitions

It is of interest to consider how a boundary with a broad transition between two regions could be handled. The BLM algorithm, as formulated, makes no provision for such transition regions: each pixel is assigned to one region or the other, and the regions are assumed (in the current implementation) to be homogeneous. In fact, if the distributions of the regions overlap, there is generally no serious problem. The pixels near the middle of the transition region will have characteristics intermediate between the two regions, and consequently will be ambiguous. That is, the corresponding pixels in the λ-image will have values close to zero. The location of the boundary will therefore not be tightly constrained; its likelihood will tend to have a broad maximum at the centre of the transition region. The fact that there is a broad transition region will be reflected in relatively large confidence intervals for the segment positions. However, this does not distinguish between the two cases where the boundary location is not tightly constrained: (a) where the noise or texture level is high (low contrast between regions), where the transition could be sharp, but its location cannot be accurately determined, and (b) where the noise level is low, but there is a broad transition.

If the two distributions are quite distinct, the intermediate pixels will tend to be such that they could not well belong to either region. Consequently, their log likelihood ratios will be essentially zero, and the boundary will have a flat-topped likelihood maximum in the vicinity of the transition region. Problems may arise in the consistency test, however, because it may be possible to find clumps which cannot belong to either region.

In the spirit of the BLM algorithm, and of the region modelling discussed in section 5.3.2, it would be best to model the transition region explicitly with a parametric model. If the transition region is very broad, it is probably best modelled as a region in its own right, with spatially varying characteristics. The boundaries of such a region would probably not be at all tightly constrained—perhaps a fixed width for the region would be appropriate. Narrower transition regions could be modelled by incorporating the transition as part of the boundary model (which would complicate the optimization phase). With a sharp transition, the likelihood changes according to those pixels which the boundary crosses as it moves. A broad transition would mean that all pixels within the transition region would have to be taken into account in determining how the likelihood changes when the boundary is moved.

5.1.4.3 Vector pixels

In many applications, the information available for each pixel has a number of components, that is, it forms a vector, rather than being a simple scalar value. The most obvious example of this is colour information (or more generally, multi-spectral information), where each pixel has red, green, and blue components. Such information could be handled quite easily within the framework of the BLM algorithm. This would require the statistical models to be extended to provide the likelihood of a pixel as a function of its vector value. These likelihoods would define the function $\lambda(\mathbf{x})$, which would be a scalar function of the vector pixel $\mathbf{x}$. Thus, the λ-image would still be a scalar image, and the autocovariances would be calculated as before.

The likelihoods could be defined in two ways. The simpler way is to make the approximation of assuming that the components of the vector are independent. Histograms can then be constructed for each component independently; the log likelihoods obtained for each component would simply be added. Pre-processing techniques such as principal components analysis (Rao, 1973, p590) could be used to reduce the correlation between the components, making the assumption of independent components a very good approximation to the truth. On the other hand, if this assumption cannot be justified, the likelihoods will have to be assigned using methods that take into account all the components at once, such as multi-dimensional histogramming.

One major advantage of using the BLM algorithm on vector data is that one segmentation is done, yielding one result. Other techniques which are only suitable for scalar

data would have to be applied to each component in turn, giving a set of segmentations which have to be combined in some way.

A bank of different texture filters could be used before the BLM algorithm to enable it to distinguish textures which do not differ in their pixel intensity distributions. The set of output images from these filters can naturally be regarded as forming one vector-valued image, which could be processed as described above. Therefore, using a large number of pre-filters need not add substantially to the complexity or expense of the BLM algorithm.

5.1.4.4 Correspondence with human perception

The results from the BLM algorithm generally correspond closely to the boundary perceived by humans. Image `noise` shows the ability of the BLM algorithm to integrate information globally, as humans seem to be able to do. When the whole length of the boundary is visible, as in figure 4.33 on page 117, the boundary seems fairly clear and sharp. However, if all except the middle tenth is masked, as in figure 5.1, the boundary is perceptually much less sharp. Obviously, the whole length of the boundary in figure 4.33 is required to obtain the perception of a sharp boundary. The BLM algorithm obtains corresponding results—the confidence interval for the middle tenth of the boundary is 4.4 pixels wide, whereas the confidence interval for the central 95% is only 1.2 pixels wide. (The central 95% section was used, rather than the whole length of the segment, to allow the section to be displaced some distance without intersecting the border of the image.)

Subjective contours. A subjective contour is a percept of an edge where there is no actual edge in the stimulus. Thus, the sharp boundary perceived by human observers in an image such as `gmcorner` (figure 4.1, page 89) has intervals where the contour is subjective, as there are intervals along the boundary where there is little or no change in intensity across the boundary. In this sense, the BLM algorithm can 'perceive' subjective contours. The intervals where there is no intensity change across the boundary can be quite long, as in the case of image `dots` (figure 4.35, page 118).

The BLM algorithm could also 'perceive' the subjective contour completing a boundary behind an occluding object, if the area occupied by the occluding object was identified, and the log likelihoods of all the pixels in that area were set to zero. This

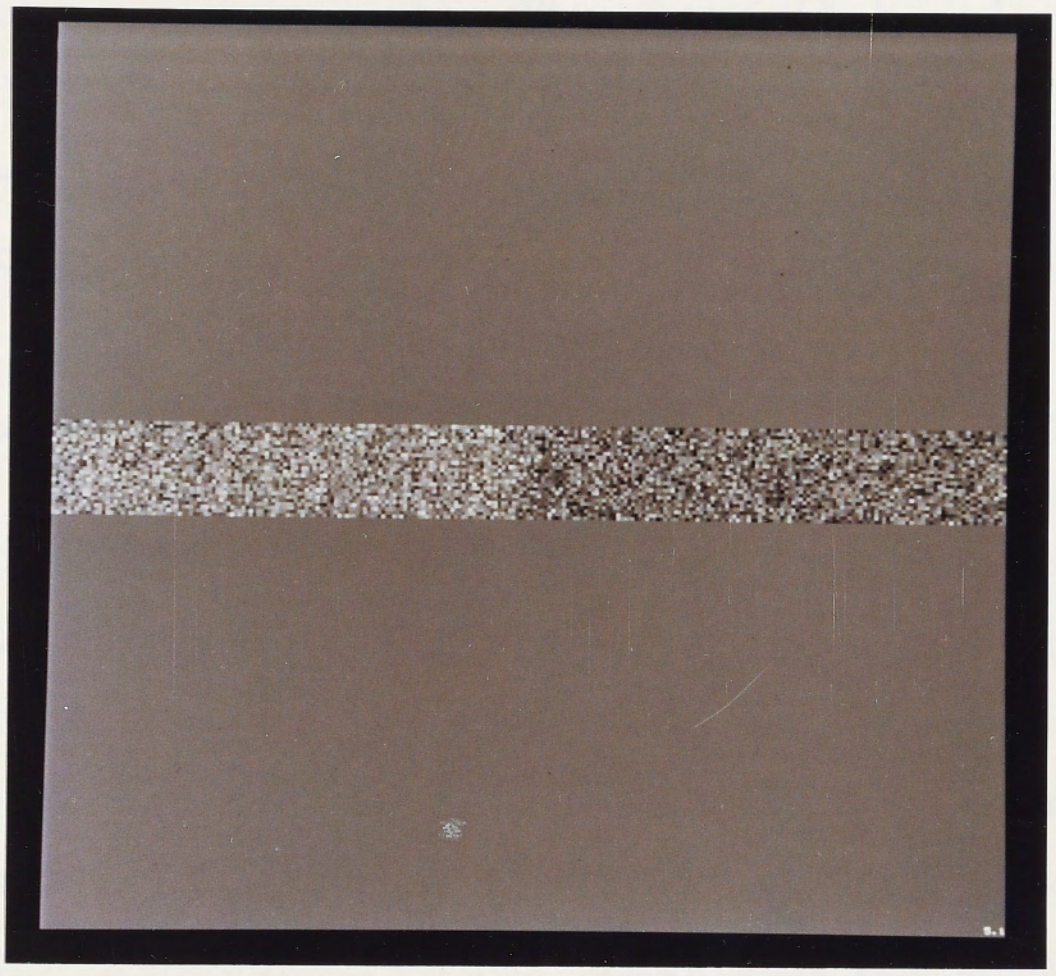


Figure 5.1: Middle tenth of image noise

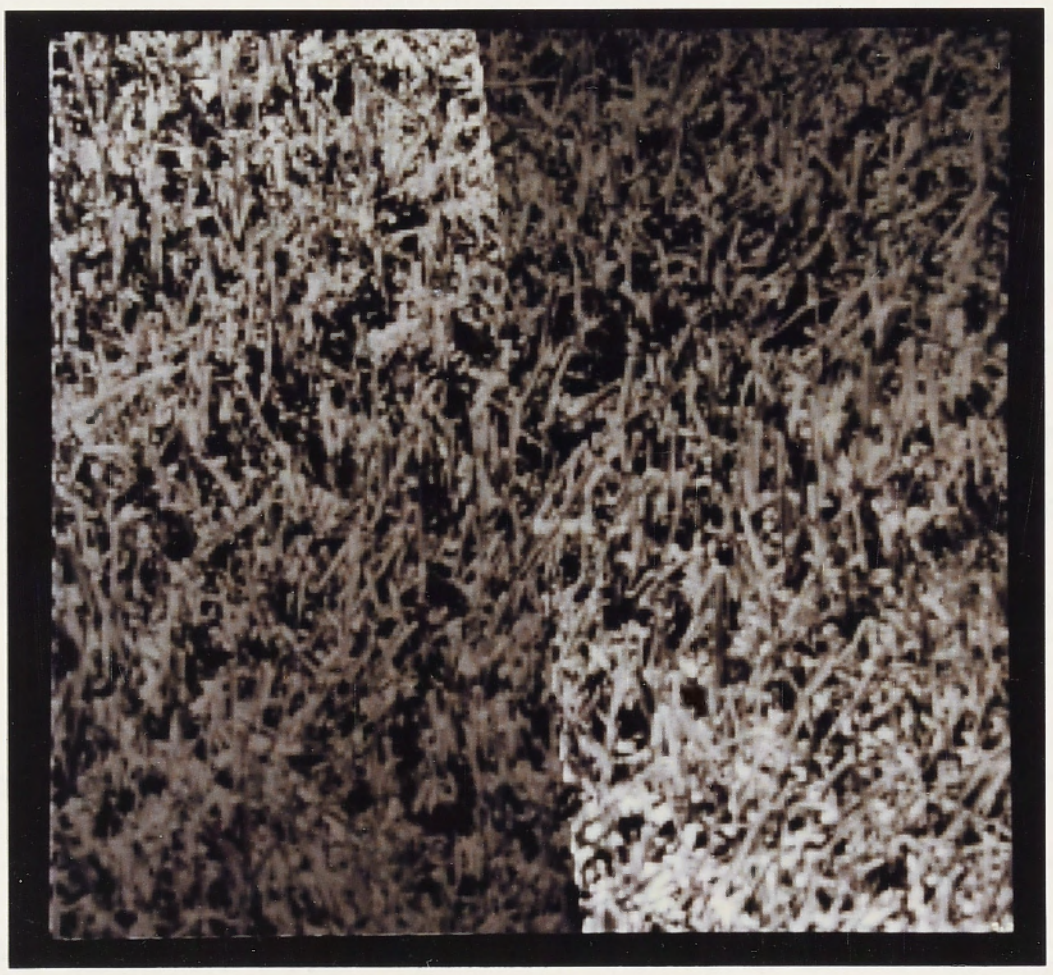


Figure 5.2: Image with spatially varying regions

would indicate that there is no information about where the boundary is, in the interval behind the occluding object. The BLM algorithm could then be used to find the simplest hypothesis which is consistent with the information which is available, i.e., the parts of the boundary which are visible. The part of this boundary hypothesis which is occluded would be a 'subjective' contour. If cubic splines were used, it might be necessary to introduce some method to encourage constant curvature in the absence of any likelihood information.

5.1.4.5 Architectures for fast implementation of the BLM algorithm

As stated previously, the aim of this study has been to identify methods for obtaining accurate and concise descriptions of boundaries, rather than necessarily to seek methods which can be implemented in real-time. There is some scope for improvement in the algorithms employed in the BLM approach. However, it is of interest to consider what types of computer architecture could be used to advantage in efficiently implementing the BLM approach. In particular, parallel computation is probably the only way to obtain dramatic increases in speed.

An array of processors, one per pixel, could be used to advantage in some parts of the BLM algorithm, notably in the likelihood optimization phase (where each processor could simultaneously calculate when it will be crossed as the boundary moves), and the consistency test (where the protrusions could be determined at each point along the boundary simultaneously). Such operations would require that the current boundary description and other information be broadcast to all processors. This appears to be essential, because any segment of the boundary can affect pixels over any portion of the image. A central facet of the BLM algorithm is that it relates two distinct entities: a parametric representation of the boundary shape, and the array of pixels forming the image. Opportunities for algorithms in which each processor communicates only with its neighbours are therefore limited.

Parallel computation could also be used in other ways, for instance, to optimize many segments at once (as long as no two segments being optimized are adjacent), or to try different classes of description simultaneously (e.g., one processor could obtain a piece-wise linear description while another uses cubic segments). Clearly, further research into efficient implementation of the BLM algorithm is required if it is to be used in a real-time machine vision system.

5.2 Comparison with other approaches

5.2.1 Edge-based approaches

With suitable pre- and post-processing, edge-based approaches are capable of giving reasonable results on simple images, such as 'blocks-world' images, in which the regions are of substantially uniform brightness. However, the assumptions on which they are based are not appropriate for textured images, as the results in chapter 4 show. The philosophy of edge-based approaches to segmentation is quite different from that of the BLM algorithm, in that it is assumed that the extent of difference between two areas of an image is purely a function of the difference in brightness between them. In contrast, the BLM algorithm uses statistical models for the regions, explicitly allowing for local intensity variations within regions.

The intensity variations within textured regions tend to cause unwanted responses, because edge detectors cannot distinguish intensity variations within a texture from the changes in intensity that occur at the boundaries between regions. (The BLM algorithm is not based on and does not attempt to make this distinction.) This problem of false responses can be overcome to some extent by smoothing the image with a low-pass filter, i.e., by averaging over large areas, prior to the derivative (or derivative-like) operation which compares the intensity in neighbouring parts of the image. This is based on an assumption that the intensity change at the boundary of a region will persist over a larger distance than the intensity changes within the texture in the region. This assumption is valid in simple cases, but it causes problems in many situations, such as where the boundary of the region has a sharp corner, at the junction of three or more edges, and near the edge of the image.

The two approaches differ in how the location of an edge is determined. Edge detectors usually mark edges at the points where the smoothed image is changing fastest. Intensity variations within the textures may be considerably reduced in the smoothed image, but they cannot be entirely eliminated, and these variations tend to perturb the position of maximum rate of change. Consequently, the reported edge position tends to wander around the true boundary position, as the results in chapter 4 show.

The output of an edge detector is usually a map of edge pixels, or possibly several maps for different scales. Combining the results for different scales is quite difficult, in

general, because the responses corresponding to the same edge in the original image will tend to occur in different places in the different edge maps. Given that the different scales are combined, there still remain the problems of identifying strings of edge pixels, linking them together, and (if necessary) approximating the resulting chains with a parametric representation.

These chains tend to be smooth when large amounts of smoothing are used on the image, but apart from that, there is no notion of preferring a 'simple' boundary. (Note that the large-scale detectors will report a smooth shape even when the true boundary has a sharp corner.) Converting the edge strings to a parametric representation therefore involves making approximations, in order to obtain a relatively simple shape, and also to try to remove unnecessary undulations caused by texture. These approximations are generally made without referring to the original image, with the consequent danger that the approximation could be even further from the true boundary than the chain of edge pixels.

The BLM algorithm does not suffer from such problems. Use of parametric representations allows 'simple' boundary shapes to be encouraged, without prohibiting corners. The parametric representation of the boundary is fitted directly to the original image data; the algorithm makes no arbitrary and untested approximations in determining the boundary shape. Multiple scales are not used, so there is no problem of combining the results at multiple scales. (There is a scale parameter, which specifies the maximum scale of textural variations, but setting this parameter to a large value does not cause the algorithm to give inaccurate results.) The BLM algorithm can actually combine information over much larger distances than are practicable for a convolution edge detector, without having to smooth the image.

Thus, we see that the edge-based approaches involve making a number of assumptions which are inappropriate for images which contain textured regions, chief of which is the assumption that the larger the difference in intensity between two patches, the more likely they are to belong to different regions. In contrast, the BLM algorithm combines explicit knowledge about the textures with a preference for simplicity to obtain a boundary hypothesis which is simple, consistent, and accurate.

5.2.1.1 Comparison with the approach of Mansouri *et al.*

The approach of Mansouri *et al.* (1987) has some interesting similarities with the BLM approach, but also some important differences. Their approach initially convolves the image with two masks of 3×3 pixels and obtains a gradient magnitude and orientation at each pixel. The basic idea underlying their approach is that pixels along a straight segment of an edge contour will have similar orientations, and that the hypothesis of a given straight-line segment can be tested by assessing the uniformity of the gradient direction along the length of the segment. Experience with the convolution edge detectors implemented in this study demonstrates that the gradient orientation will generally not be uniform along a straight edge in textured images, especially when small edge detection masks are used. For example, the results shown in figure 4.12 on page 100 for the smallest scale show that the gradient orientation oscillates wildly along the length of the true boundary. (The masks used for the smallest scale have a support of 15×15 pixels.) It is therefore doubtful whether the approach of Mansouri *et al.* could identify the true boundary in image `gmcorner` (figure 4.1, page 89). Another relatively minor point is that their hypothesis test is not formulated so as to have a fixed false-alarm rate; their test depends on pre-set thresholds, and the false alarm rate is not assessed.

5.2.1.2 Comparison with the approach of Kass *et al.*

The approach of Kass *et al.* (1987) is also similar in some ways to the BLM approach. Their approach finds a smooth curve which is influenced both by the image data and by a preference for smoothness, and is thus able to find a reasonable interpretation of the image data in many cases. In fact, their approach has most in common with the statistical estimation approaches discussed in section 2.4, although it is not formulated in statistical terms. Both approaches find an interpretation which combines an influence from the image data with a preference for smooth boundaries; the MAP approaches express this in terms of probabilities, whereas Kass *et al.* express it in terms of 'energy'. The relative weighting of the image data with respect to the preference for smoothness is a critical parameter in both approaches.

In one example given by Kass *et al.*, their procedure has found a smooth boundary around an object, despite the irregularity of the zero-crossing contours which it is using. If the internal forces were relaxed, the boundary would follow the zero-crossings

more closely, and would therefore become more irregular. The method thus gives no indication as to whether the boundary really is smooth or irregular. This illustrates the major deficiency of their approach—there is no consistency test. The BLM algorithm uses the consistency test to determine whether intensity fluctuations represent real deviations in the boundary, or whether the boundary is actually smooth, with the fluctuations being due to texture.

The energy minimization performed by Kass *et al.* is similar to the likelihood optimization step in the BLM algorithm, and their methods could perhaps be adapted to likelihood maximization by using an appropriate energy functional. The curve that their method finds is simple in the sense of being smooth, but not simple in the sense of being able to be described with few numbers. The curve is the solution of a set of differential equations, which is solved numerically. Describing the curve (independently of the original image) would therefore require either a large array of coordinates, or a chain code; in neither case is the description concise.

5.2.2 Region-based approaches

Region-based approaches generally tend to be able to take into account the statistical properties of regions more easily than edge detection algorithms, and in this sense they are closer to the BLM algorithm. However, region-based approaches differ considerably from the BLM approach in that very little attention is paid to the shapes of the boundaries of the regions obtained; rather, the aim is to identify the different regions in the image. In this way, the region-based approaches are complementary to the BLM algorithm, which assumes that the regions have already been identified (by a split-and-merge phase in the current implementation), and seeks to find the accurate location of the boundary between the regions.

Because region-based approaches regard the region boundaries as of secondary importance, they tend to produce rather poor information about the shapes of the boundaries. Pixel classification techniques (e.g., thresholding) tend to produce disconnected regions with irregular boundaries. Techniques which concentrate on larger blocks of the image (e.g., window classification, split-and-merge) tend to produce region boundaries which have a 'blocky' appearance, and may also tend to generate many small regions near the boundaries between large regions. These small regions cannot reliably be merged with either region. The BLM algorithm has an analogous difficulty to cope

with, in that it may be ambiguous as to which region an individual pixel (or clump of pixels) belongs. This problem is solved by seeking to make the best overall assignments, that is, by assigning pixels to one region or the other so as to maximize the overall likelihood.

Thus, region-based approaches and the BLM algorithm are concentrating on different aspects of the segmentation problem, and are perhaps not directly comparable. Section 5.3.2 discusses a segmentation technique, using the same basic principles as the BLM algorithm, which combines the region-finding and boundary-locating aspects of the segmentation problem in one algorithm.

5.2.3 Statistical approaches

Techniques which seek to segment an image by maximizing an *a posteriori* probability have more in common with the BLM algorithm than either edge-based or region-based approaches. These techniques are described in section 2.4. They are called MAP segmentation techniques (for Maximum *A Posteriori*). The underlying idea is to assign pixels to regions so as to maximize the probability of the segmentation, given the observed image; this probability is proportional to the product of the probability of the region assignments multiplied by the likelihood of the original image, given the region assignments.

There are thus two components to the statistical model: the model for the region shapes, which assigns an *a priori* probability to each possible segmentation, and the model for the texture in each region, which assigns a likelihood to each region, given the contents of that region of the image. The *a priori* component is used to introduce a preference for 'nice' region shapes, (e.g., connected regions with smooth boundaries). The likelihood component is, in principle, the same as that used in the BLM algorithm, although different statistical models are used.

The BLM and MAP approaches are therefore similar in that they both contain an element of seeking to maximize the likelihood of the image data, given the proposed assignment of pixels to regions. However, they differ in the way in which this likelihood-maximization is constrained in order to produce reasonable results. (Likelihood maximization, with the assumption of independent pixels, becomes merely pixel classification.) The MAP segmentation techniques seek to encourage 'good' or 'simple' region shapes by assigning them a high *a priori* probability, which acts to compensate

for the generally lower likelihood that such regions would have. In contrast, the BLM algorithm uses the criterion of consistency to define an acceptable set of segmentations, and chooses the simplest of these. Therefore, the BLM algorithm can choose a less likely segmentation in favour of a more likely one, provided that the less likely one is consistent and simpler. The MAP approaches can also choose a less likely segmentation in favour of a more likely one, if the *a priori* probability of the first is sufficiently greater than the second.

Both approaches can cope with texture in the regions, that is, with a distribution of pixel intensities rather than an almost constant intensity. If adjacent pixels are not statistically independent, the MAP approaches generally require Markov random field models to cope with the dependence. Use of such models usually involves considerable computational expense. Fortunately, they were not found to be necessary in the BLM algorithm.

In their current implementations, both methods require the statistical characteristics of the regions to be supplied before starting. That is, both require an initial segmentation, and so both can be described as boundary locators, rather than region detectors. So-called 'adaptive' algorithms for MAP segmentation, which do not require the region models to be initially specified, are an active area of research (e.g., Cohen and Cooper, 1987). Some suggestions for extending the BLM algorithm to eliminate the need for the initial segmentation are given in section 5.3.2.

The main difference between the BLM and MAP approaches lies in the criteria for the required segmentation. Whereas the BLM algorithm uses the consistency test to provide an absolute yes/no decision about whether a boundary hypothesis is acceptable, the MAP approaches only use relative information (i.e., one segmentation is better or worse than another). Consequently, there is no guarantee that the results obtained with a MAP segmentation technique will correspond to the image data within given limits; that is, the results may not be consistent with the image data. The results from MAP segmentation have not, in any of the studies cited, been checked against the image data with a consistency test, nor has a measure of confidence in the boundary location been derived.

Similarly, the shapes of the regions may be quite complex, even when there are much simpler shapes which are consistent (but perhaps of lower *a posteriori* probability). The region shapes that are obtained will depend on the relative strengths of the

likelihood and *a priori* probability contributions; if the likelihood contribution dominates, the region shapes will tend to be complex, whereas if the *a priori* contribution dominates, the region assignments may become inconsistent with the image data. Since the *a priori* probabilities are usually specified quite arbitrarily, they represent a set of parameters in the segmentation process whose values are quite critical to obtaining correct segmentations.

Published results from MAP segmentation techniques most often show segmentations of completely artificial images (e.g., Derin and Cole, 1986). Even so, the results often show rather irregular region boundaries, and sometimes also show numbers of small isolated regions. These deficiencies are mainly due to inadequate models for specifying the *a priori* probabilities of regions of various shapes. The best models that have been used to date are Markov random field models, with the parameters set so as to encourage neighbouring pixels to belong to the same region. However, practical considerations usually limit the set of neighbours of a pixel to the nearest four or eight pixels. While a dependence span of this size is capable of producing large, connected regions, it is not capable of expressing the difference between a region with a globally straight or smooth section of its boundary, and a region with a quite irregular boundary. These models for the region shapes therefore do not capture the essential qualities of 'simple' boundaries.

5.3 Possible extensions to the BLM algorithm

5.3.1 Handling multiple regions

As noted above, the current implementation is limited to considering two homogeneous regions. The limitation to two regions could be overcome quite easily. This would involve implementing an appropriate data structure to represent the regions, boundary segments, and knots. These would be linked together into a structure here called the 'RSK graph' (for Regions, Segments, and Knots). This structure is based on the 'RSV graph' (V for vertices) of Hanson and Riseman (1978). The following information would be associated with the various nodes:

- Each region would be linked to the boundary segments which define the boundary of the region, and to the adjacent regions. Each region would be associated with

a statistical model for the region, giving the estimated intensity distribution, and possibly the grey-level dependence matrices.

- Each pair of adjacent regions, say regions j and k, would have an associated log likelihood ratio function $\lambda_{jk}(x)$, plus the mean, variance, and autocovariance function for the λ_{jk}-image in both regions.
- Each boundary segment would be linked to a knot at each end, and to the regions on either side. Other information in the node would include the segment type (e.g., linear or cubic) and any other parameters needed to define its shape (e.g., initial and final velocities).
- Each knot would be linked to the segments which are attached to it. The information stored in the node would include the position of the knot, together with any constraints on the boundary shape; for example, collinearity of one or more pairs of segments.

With this representation, the structure of the graph corresponds directly to the structure of the regions and boundaries in the image. That is, the 'region' nodes would correspond to visible surfaces (or parts thereof), and the 'segment' nodes would correspond to events such as boundaries of surfaces, occlusion of one surface by another, shadows, or changes in surface properties. Knot nodes with more than two segments attached will be called 'junctions' here. They are quite important, because they essentially define the topology of the boundary structure. Each junction could be characterized as a T-junction, Y-junction, or X-junction, etc., according to the number of attached segments and any constraints between them. These different types of junction could correspond to events such as occlusion of a boundary or three-dimensional corners. The knot nodes with two segments attached are necessary to represent correctly the shape of the boundary between two regions. Thus, dealing with a string of segments joined by such knots essentially involves only two regions. As discussed above, these knots could either correspond to corners in the shape of the true boundary, or perhaps merely be necessary to represent the shape of the true boundary.

The BLM algorithm, extended in this way, could easily and naturally handle the junction of three or more edges, giving accurate and reliable results, without interference between adjacent edges. This is a situation where many edge detectors (in particular the Marr-Hildreth detector (Shah *et al.*, 1986)) fail badly.

Some minor changes to the BLM algorithm would be necessary to handle the RSK graph. These are:

- As before, the region characterizations and initial boundary hypothesis would be obtained from an initial segmentation. Obtaining the initial hypothesis would be slightly more complex than previously, since sufficient knots would have to be established that the initial segments around each region enclosed at least part of that region. This would include inserting all of the junctions which were necessary to represent the topology of the regions and their boundaries.

- Optimizing the positions of knots with two segments attached would be the same as previously. When optimizing a junction, there can be as many possibilities for the likelihood of some pixels as there are regions in the vicinity of the junction. Even so, as a knot is moved along an optimization line, the likelihood will change in discrete steps as before, according to the pixels crossed by any segment attached to the junction. Where there are constraints on some of the attached segments (such as at a T-junction), these constraints may affect the permissible movements of the knot, or may require that one or more other knots move in concert with the knots being optimized.

- The consistency test would operate as before, since it considers each segment individually.

- The elaboration and simplification steps would operate largely as before, since we are assuming at this point that all the regions have been identified, and thus all the necessary junctions have been inserted at the stage of forming the initial hypothesis. However, these steps would also have to consider simplification by adding constraints (e.g., a T-junction should be regarded as simpler than a Y-junction), or elaboration by removing constraints. This is very similar to the possibility of slope continuity at a knot between two cubic segments.

5.3.2 Spatially-varying regions

The current implementation assumes that the regions are statistically uniform—that is, the statistical characterization obtained in one part of each region applies to the whole region. Clearly, in real-world scenes, this is not the case; gradual changes

often occur across a region. The underlying philosophy of the BLM algorithm (i.e., obtaining the simplest hypothesis which is accurate and consistent with the image data), can be used to obtain models which account for gradual variations across regions. Such a method will also sub-divide the image into regions, and will interact with the boundary locator. Thus, this underlying philosophy extends to provide a complete image segmentation algorithm. (However, real-world images may still contain some detail, such as linear features, which cannot conveniently be described within a framework of disjoint regions with smooth variations.)

Some important causes of gradual variation across regions are:

- Variations in illumination on a surface due to
  - changes in aspect of the surface,
  - diffuse shadows, or
  - changing distance from the light source;
- Variations in spatial scale of texture due to changing aspect and/or distance from the viewer ('texture gradient' (Stevens, 1981; Witkin, 1981; Gibson, 1950)).

These variations are rich in information about the surfaces being viewed, their orientations, distances, and shapes. Algorithms such as the 'shape from shading' algorithm of Ikeuchi and Horn (1981) use these variations in three-dimensional reconstruction of a scene.

Thus it will be necessary in processing real-world scenes to take spatial variations within regions into account for two reasons: (a) so that the image can be correctly segmented, and (b) because the variations are themselves informative. The first reason relates to the incorrect results given in some cases by the current implementation when the regions are not uniform in their statistical characteristics. An extreme example of this is shown in figure 5.2. This image could not be correctly segmented into the two regions perceived by human observers by any technique which assumed that regions were homogeneous. Such images as this cause severe problems to other strategies such as edge detection (because the contrast of the edge becomes zero at the middle), and split-and-merge techniques (because regions are assumed to be homogeneous).

As a first step, the BLM algorithm could easily be extended to handle regions which are not homogeneous, given appropriate statistical models for the regions; the

BLM algorithm could then handle images such as figure 5.2 on page 147 correctly. A spatially-varying region model would require two components:

1. A probability distribution function for the intensity of each pixel. This distribution could be a function of the position of the pixel. Consequently, the log likelihood ratio function $\lambda_{jk}(x)$ for adjacent regions j and k would also become a function of position.

2. Information about the spatial dependence of the pixels. Sufficient information would be required to be able to calculate the variance of any given patch of the λ_{jk}-image for adjacent regions j and k. In general, this would also depend on the position of the patch. (Only patches near the boundary between regions j and k will be considered.) In principle, all the necessary information is expressed in the spatial grey-level dependence matrices, which would now be a function of position.

How could these statistical models for the regions be determined? In the spirit of the BLM algorithm, they could be determined by searching for the simplest hypothesis which is accurate and consistent, given the following elements:

(a) A set of parametric models for different types of spatial variation, each with an associated level of complexity. These would represent various classes of hypotheses about a region.

(b) A method for finding the most likely model in a given class for a particular region. This would involve estimating the parameters of the model by likelihood maximization.

(c) A method for testing whether a given model is consistent with the image data in a region, that is, for finding patches of the region which are inconsistent.

(d) A strategy for elaborating and simplifying region models.

Most of these elements would be rather more complex than in the case of the BLM algorithm, since they deal with two-dimensional areas rather than one-dimensional lines. (One-dimensional (time series) data could be segmented with this approach; these elements would then be much simpler than for two-dimensional images.) In particular, the consistency test involves searching for patches within a region which

fail the appropriate test of hypothesis. Given the astronomical number of possible connected patches within a region, it is clearly impossible to test them all; some kind of representative subset should be tested (perhaps partly overlapping square windows at a range of sizes).

These elements would be combined into an algorithm very similar in outline to the BLM algorithm, whose purpose is to find an overall, composite hypothesis for the whole image, which would incorporate parametric models for all of the regions and their boundaries. The overall hypothesis would be accurate and consistent when each component satisfied the criteria of accuracy (i.e., maximum likelihood) and consistency.

This algorithm would start with the simplest hypothesis: one homogeneous region for the whole image. This hypothesis would be fitted to the image data using likelihood maximization ((b) above), and then tested for consistency ((c) above). If the image is of a uniform texture, the model will be consistent, and the process terminates. On the other hand, if there are two (or more) regions, or a broad spatial variation across the region, then the consistency test will fail.

At this point, there are several options for the elaboration step. It can either use a more complex model for the region, or it can split the region into two (or more) pieces. Suppose that a spatially-varying model is chosen for the region, but that there are actually two distinct regions. Now, the spatially-varying models should allow for smooth variation across the region, but not for sharp changes within the region. Consequently, none of the spatially-varying models will be consistent with the image data, so the region will eventually have to broken into two pieces in the search for a consistent hypothesis. After it has been subdivided, there will be two regions, which then have a boundary between them. Each region would start with the simplest class of region model, which would be elaborated or subdivided as necessary in a recursive fashion.

On the other hand, suppose that the image consists of a single region with spatially-varying characteristics. It may be possible to obtain consistency with multiple, uniform regions. However, the option of using a single, spatially-varying model will also be consistent, and will be preferable because it is simpler. The classes of models which are available should correspond to the types of smooth variation which are expected in the scene, such as those listed above.

As the image is subdivided, initial hypotheses are formed about the boundaries between the subregions. Once reasonable hypotheses about the regions have been formed,

the BLM algorithm would be used to refine the boundary hypotheses. Modifying the boundary of a region may cause the estimates of its statistical properties to change, which may then affect the estimated boundary position. In this way, the region and boundary hypotheses would interact, leading to an iterative process for determining the best models for a region and its boundaries. As the total likelihood should never decrease, this iteration should converge.

The above procedure is somewhat similar in parts to the split-and-merge procedure; elaboration by subdivision of a region corresponds to the 'split' phase, and simplification by merging adjacent regions corresponds to the 'merge' phase. However, the new procedure has considerably more flexibility, in using spatially-varying region models, and in subdividing a region according to the shape and size of inconsistent patches within it rather than always subdividing it into four sub-squares.

There is an interesting analogy between the different classes of region models and boundary models. A uniform region model is analogous to a straight-line segment. If this is not consistent, one can use a spatially-varying region model, which is analogous to a cubic segment (change of angle being analogous to change in statistical properties). Alternatively, one can subdivide the region into two new regions, which is analogous to introducing another knot in a boundary representation.

Thus, we see that the notion of determining simple, consistent, and accurate hypotheses extends to the whole problem of image segmentation, leading to a segmentation technique which would yield parametric models for the texture in each region and for the boundaries between regions. Such a technique would (hopefully) have similar advantages to the BLM algorithm in practical applications. In particular, it would cope, explicitly and correctly, with spatial variations within regions, and describe them in parametric terms, suitable for use by subsequent processes in three-dimensional interpretation of the scene.

5.4 Use of the BLM algorithm in a machine vision system

The BLM algorithm is, of course, not an end in itself: it is useful only insofar as its results can be used by later processing stages, for performing such tasks as 3-dimensional analysis of a scene, locating objects, and recognizing them. In this context, the BLM algorithm is potentially quite useful as it stands, but with the further

extensions described in section 5.3, it could be of even greater value.

As it stands, the BLM algorithm would not replace other segmentation techniques, such as edge detectors or split-and-merge algorithms. Instead, it would form a subsequent stage to obtain accurate boundary location estimates, after the initial segmentation technique had tentatively divided the image up into regions. The extended scheme described above, which forms a segmentation by obtaining the simplest consistent hypotheses for the regions and their boundaries, would do away with the need for the initial segmentation.

Several advantages of using the BLM algorithm, rather than using the output of an edge detector or a split-and-merge algorithm, are listed below. Some advantages relate to the algorithm as it stands, and some to extended versions.

- The boundary reported by the BLM algorithm is generally simpler. This should simplify later processing, since there is less information to deal with.

- The boundary reported by the BLM algorithm is generally more accurate. Once again, this should simplify further processing. In particular, the absence of unnecessary and incorrect undulations in the boundary should make recognition of objects considerably easier. The boundary shows neither the 'blockiness' present in the output of the split-and-merge technique, nor the following defects, present in the outputs of the convolution edge detectors used in chapter 4:
  - Unnecessary undulations, due to the influence of variations in brightness within the textures
  - Breaking-up of the boundary
  - Problems at the edge of the image and at the junction of edges

- There is no need to convert from an image-based or chain-coded representation to a parametric representation. Such conversion generally requires making approximations (so that the parametric representation is not unduly complex). An important point is that these approximations are generally made without reference to the original image. In contrast, the BLM algorithm fits the parametric representation directly to the original image, and is thus much more accurate. For example, an edge map represents less information than is in the original image (assuming that a thesholding or non-maxmimum suppression step has been

performed). The BLM algorithm is therefore in a position to use much more of the relevant information.

- The results from the BLM algorithm are reliable; they are guaranteed to be consistent with the image data. Later processing can thus proceed with confidence that the reported boundary corresponds with what is in the original image. This is quite important, since subsequent stages would rarely go back to the original image, relying instead on the reported boundaries.

- There is no need to combine the results over multiple scales to obtain results which are both accurate and reliable, as required for the convolution edge detectors. There are, indeed, parameters for the BLM algorithm which relate to the maximum scale of variation which can be considered as texture, but the algorithm does not give inaccurate results when this scale is large. In contrast, convolution edge detectors generally give inaccurate results at large scales, and unreliable results at small scales.

- There are no critical parameter or threshold settings. There are only two important parameters: the significance level in the consistency test, and the maximum scale of texture. The significance level can be set to a constant value, and the maximum scale of texture should depend on the sizes of the regions.

- Other information, potentially useful to later processing stages, can be obtained. For example, the confidence contours and intervals (section 3.4.5.4) should be useful, as an indication of the sharpness or clarity of the boundary, and also as an indication of the range of acceptable positions for the boundary. The latter could be very useful if some later process wishes to hypothesize a slightly different position for a boundary segment, in order to obtain a simple description at that level. For example, some later process may wish to hypothesize that two boundary segments are, in fact, collinear, even though their reported positions and orientations don't give collinearity. It should be possible to evaluate whether this hypothesis is consistent with the image data using the reported confidence intervals and contours. Similarly, the 'cornerness' measure should be useful in guiding the interpretation of knots in a piece-wise linear boundary as representing corners in the true boundary, or as being a descriptive convenience.

- Parts of the BLM algorithm could be used in other ways: for example, the consistency test could be used to test a boundary hypothesis suggested at some later stage in the interpretation. Or, if ambiguity exists, the likelihoods of two slightly different boundary hypotheses could be compared.

- The BLM algorithm is quite amenable to control by higher levels of processing. For example, in the version extended to handle multiple regions, it would be quite easy to place constraints on the angles that segments form at knots. Specifying that two segments must be collinear at a knot with three adjacent segments would give a 'T' junction at that knot.

- The BLM algorithm could easily be adapted to make use of statistical knowledge about the forms of the pixel grey-level distributions. For instance, if the distributions were known to be Gaussian, the mean and variance of each region could be estimated rather than the entire distribution. Likelihoods would then be calculated according to the Gaussian distribution.

- The BLM algorithm could also be used to obtain an approximate description of a boundary, if the consistency test was adapted to only consider those protrusions whose length perpendicular to the boundary is longer than a given distance. This would allow the true boundary to deviate from the reported boundary by (at most) that distance, thus producing a simpler, but approximate, boundary description.

- Where ambiguity exists, it should be possible to produce more than one hypothesis. In this context, the criteria of simplicity and accuracy should be seen as relative rather than absolute: that is, an acceptable hypothesis should be relatively simple, not necessarily the simplest; or of relatively high likelihood, not necessarily the highest. The likelihood should still be the highest for the given configuration of linear and cubic segments; however, if two boundaries were of similar complexity and likelihood but of different form, then they would both be tenable. Possible hypotheses must still, of course, be consistent.

Chapter 6

Conclusions

Existing approaches to the problem of segmentation do not give accurate, concise, or reliable information about the locations of boundaries in textured images. Inspection of sample results shows that current segmentation algorithms fall far short of human performance. In particular, the Marr-Hildreth and Canny edge detectors give boundary contours which are irregular and inaccurate when applied to test images containing smooth boundaries between regions of texture.

These deficiencies are due to intensity variations within the textures. Textured regions may contain a broad range of intensities, and nearby pixels may be correlated. The intensity changes within a textured region can be larger than the intensity changes at the border of the region. Generally, edge-based methods tend to be perturbed by the intensity variations within the regions, and thus yield inaccurate results. Large-scale edge detectors can distinguish between local intensity variations due to texture and changes in average intensity at the border of a region, because the latter tend to persist over larger distances than the former. However, the textural intensity variations still affect the positioning of the edge elements along the region boundary. On the other hand, small-scale edge detectors cannot distinguish between textural variations and region boundaries; their results are so confused as to be virtually useless.

Traditional region-based methods cannot give fine location information because the characterization of a block of texture becomes less reliable as the block becomes smaller. The classification of small blocks near the boundary between two textured regions can therefore be ambiguous, especially if the distributions of the properties of the two regions overlap. This ambiguity makes it difficult to obtain an accurate and reliable estimate of the boundary position. Consequently, most region-based methods impose

a minimum size on the blocks considered, and the boundaries of the regions obtained therefore tend to be "blocky" in outline.

Many of the problems with region-based methods are due to the independent classification of pixels or small blocks of pixels. This deficiency can be overcome to some extent by using statistical estimation methods which make the classification of a pixel dependent on the classification of its neighbours. These methods use statistical models for the shapes of the regions which express a preference for having each pixel classified to the same region as its neighbours. However, practical considerations limit the neighbourhood size to the nearest four or eight pixels, which is insufficient to express a preference for boundary shapes which are globally simple.

The approach presented in this study (as embodied in the BLM algorithm) overcomes these difficulties. Use of a statistical model for the texture in the regions allows for the intensity variations within the regions, and use of a parametric representation of the boundary enables the BLM algorithm to determine the description which is the simplest possible in a global sense. Ambiguous pixels near the boundary cause few problems, because the maximum-likelihood estimate of the true boundary position takes account of the relative degree of ambiguity of all pixels near the boundary. Finally, the boundary estimate obtained by the BLM algorithm is reliable, because it has been tested for consistency with the image data.

The results obtained with the BLM algorithm are consequently concise and accurate, and generally correspond well with the results of human perception. The BLM algorithm has been applied to a wide range of imagery, including both real-world images and images that have been artificially constructed from natural textures. The results display no more complexity than the observable level of detail in the image warrants, and represent a substantial improvement over conventional techniques.

The BLM algorithm is driven by three criteria for a good boundary description: accuracy, simplicity, and consistency. These criteria were motivated by the characteristics of human perception of boundaries in textured images, and are formulated in mathematical terms. The appropriateness and effectiveness of these criteria are demonstrated by the excellent results obtained with the BLM algorithm. These three criteria together capture the essence of a good description of a boundary.

The formulation of the accuracy criterion in terms of maximum-likelihood estimation of the boundary position gives good results. Maximum-likelihood estimation takes

into account all of the relevant statistical information. The assumption of pixel independence used in calculating likelihoods does not cause significant problems, although it does require that the regions differ in their pixel intensity distributions. Regions with the same pixel intensity distribution but with different textures can be segmented by applying an appropriate texture filter before using the BLM algorithm.

Use of a parametric representation for the boundary has several advantages. The parametric representation allows a natural measure of the complexity of a boundary, which depends only its shape, not on its size or on the pixel resolution of the image. This complexity measure corresponds well to human assessment of the complexity of a boundary. There is also a significant advantage in fitting a parametric representation directly to the original image data, as the BLM algorithm does, rather than fitting it to some abstraction such as an edge map. Fitting a parametric representation to an edge map or other abstractions inevitably involves making approximations which are not tested against the image data, and which therefore may be inconsistent with the image.

Perhaps the single feature which most clearly distinguishes the present approach from previous approaches is the use of a consistency test. The likelihood-ratio test used in testing whether a boundary is consistent with the image data works well. One of its best features is that the false-alarm rate is controlled explicitly by a single threshold, yielding a test which is as powerful as possible, consistent with an essentially zero false-alarm rate, and which does not require critical setting of any thresholds. The test successfully takes the correlation between pixels into account. The results from the BLM algorithm show that the consistency test judges correctly, given the level of observable detail in the boundary. Thus, the BLM algorithm has the desirable characteristic of producing a less complex boundary description in situations where the regions are less distinct, because the consistency test is then less stringent. The approximation of normality of the log likelihood ratio statistic does not cause any significant problems.

In contrast to most previous studies, the results obtained with the BLM algorithm demonstrate that it is possible to obtain good descriptions of boundaries without requiring fine choice of values for critical parameters. The behaviour of the algorithm is controlled by two main parameters: the significance level in the consistency test, and the maximum scale of intensity variation which will be considered as texture. *Both*

were set at the same values for all the images processed. The scale parameter is at the discretion of higher-level processes; however, large values do not cause the algorithm to give incorrect or inaccurate results on fine textures.

The current implementation of the BLM algorithm assumes that there are only two regions, and that the regions are homogeneously textured. The implementation adequately demonstrates the concept of using the three criteria to drive the process of locating and describing boundaries, and is sufficient to obtain excellent results on a wide variety of images. However, these assumptions are somewhat restrictive in dealing with real-world images, and the implementation needs to be extended for use in a general-purpose vision system. The necessary extensions to handle multiple regions are straightforward.

The BLM approach has many implications for the overall image segmentation problem, and also for the problem of machine vision in general. The three criteria can be applied to descriptions of other entities, such as spatially-varying regions of an image. The approach developed in this study not only makes a significant contribution to the problem of locating boundaries in textured imagery; it also provides a clear and coherent framework for characterizing regions as well, leading to a complete image segmentation procedure.

Bibliography

I.E. Abdou and W.K. Pratt, 'Quantitative design and evaluation of enhancement/thresholding edge detectors', *Proc. IEEE* **67**(5), 753–763, 1979.

D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982.

J. Besag, 'Spatial interaction and the statistical analysis of lattice systems', *J. Roy. Statistical Soc.* **B36**, 192–236, 1974.

A.C. Bovik, T.S. Huang, and D.C. Munson, 'Nonparametric tests for edge detection in noise', *Pattern Recognition* **19**(3), 209–219, 1986.

A.C. Bovik and D.C. Munson, 'Edge detection using median comparisons', *Computer Vision, Graphics, and Image Processing* **33**, 377–389, 1986.

P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover Publications, New York, 1966.

T. Caelli, 'Three processing characteristics of visual texture segmentation', *Spatial Vision* **1**(1), 19–30, 1985.

J. Canny, 'A computational approach to edge detection', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-8**(6), 679–698, 1986.

P.C. Chen and T. Pavlidis, 'Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm', *Computer Graphics and Image Processing* **10**, 172–182 (1979).

P.C. Chen and T. Pavlidis, 'Image segmentation as an estimation problem', *Computer Graphics and Image Processing* **12**, 153–172 (1980).

M. Clark and A.C. Bovik, 'Texture discrimination using a model of visual cortex', 6pp. in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, Atlanta, October 1986.

F.S. Cohen and D.B. Cooper, 'Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-9**(2), 195–219, 1987.

R.W. Conners, M.M. Trivedi, and C.A. Harlow, 'Segmentation of a high-resolution urban scene using texture operators', *Computer Vision, Graphics, and Image Processing* **25**, 273–310, 1984.

D.B. Cooper and H. Elliott, 'A maximum likelihood framework for boundary estimation in noisy images', pp. 25–31 in *Proc. IEEE Computer Soc. Conf. on Pattern Recognition and Image Processing*, Chicago, May/June 1978.

D.B. Cooper, H. Elliott, F. Cohen, L. Reiss, and P. Symosek, 'Stochastic boundary estimation and object recognition', *Computer Graphics and Image Processing* **12**, 326–356, 1980.

G.R. Cross and A.K. Jain, 'Markov random field texture models', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-5**(1), 25–39, 1983.

L.S. Davis, 'A survey of edge detection techniques', *Computer Graphics and Image Processing* 4, 248–270, 1975.

L.S. Davis and A. Mitiche, 'MITES: a model-driven, iterative texture segmentation algorithm', *Computer Graphics and Image Processing* **19**, 95–110, 1982.

L.S. Davis and A. Rosenfeld, 'Detection of step edges in noisy one-dimensional data', Tech. report TR-303, Uni. of Maryland Computer Science Center, College Park, Maryland, 1974.

H. Derin and W.S. Cole, 'Segmentation of textured images using Gibbs random fields', *Computer Vision, Graphics, and Image Processing* **35**, 72–98, 1986.

H. Derin and H. Elliott, 'Modeling and segmentation of noisy and textured images using Gibbs random fields', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-9**(1), 39–55, 1987.

H. Elliott, D.B. Cooper, F.S. Cohen, and P.F. Symosek, 'Implementation, interpretation, and analysis of a suboptimal boundary finding algorithm', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-4**(2), 167–182, 1982.

W. Frei and C-C. Chen, 'Fast boundary detection: a generalization and a new algorithm', *IEEE Trans. Computers* **C-26**(10), 988–998, 1977.

J.E. Freund, *Mathematical Statistics*, 2nd ed., Prentice-Hall International, London, 1972.

A. Gagalowicz and S.D. Ma, 'Sequential synthesis of natural textures', *Computer Vision, Graphics, and Image Processing* **30**, 289–315, 1985.

S. Geman and D. Geman, 'Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-6**(6), 721–741, 1984.

J.J. Gibson, *The Perception of the Visual World*, Houghton-Mifflin, Boston, 1950.

A.R. Hanson and E.M. Riseman, 'Segmentation of natural scenes', pp. 129–163 in *Computer Vision Systems*, ed. A.R. Hanson and E.M. Riseman, Academic Press, Inc., Orlando, Florida, 1978.

R.M. Haralick, 'Statistical and structural approaches to texture', *Proc. IEEE* **67**(5), 786–804, 1979.

R.M. Haralick and L.G. Shapiro, 'Image segmentation techniques', *Computer Vision, Graphics, and Image Processing* **29**, 100–132, 1985.

M. Hassner and J. Sklansky, 'The use of Markov random fields as models of texture', *Computer Graphics and Image Processing* **12**, 357–370, 1980.

E.C. Hildreth, 'Edge detection', A.I. memo no. 858, MIT Artificial Intelligence Lab., 1985.

S.L. Horowitz and T. Pavlidis, 'Picture segmentation by a tree traversal algorithm', *J. Ass. Computing Machinery* **23**(2), 368–388, 1976.

M.H. Hueckel, 'An operator which locates edges in digitized pictures', *J. Ass. Computing Machinery* **18**(1), 113–125, 1971.

M.H. Hueckel, 'A local visual operator which recognizes edges and lines', *J. Ass. Computing Machinery* **20**(4), 634–647, 1973.

K. Ikeuchi and B.K.P. Horn, 'Numerical shape from shading and occluding boundaries', *Artificial Intelligence* **17**, 141–184, 1981.

B. Julesz and J. R. Bergen, 'Textons, the fundamental elements in preattentive vision and perception of textures', *Bell System Tech. J.* **62**(6), 1619–1645, 1983.

R.L. Kashyap and A. Khotanzad, 'A stochastic model based technique for texture segmentation', pp. 1202–1205 in *Proc. 7th International Conference on Pattern Recognition*, Montreal, July/August 1984.

M. Kass, A. Witkin, and D. Terzopoulos, 'Snakes: active contour models', pp. 259–268 in *Proc. First International Conf. on Computer Vision*, London, June 1987.

S. Levialdi, 'Finding the edge', pp. 105–148 in *Digital Image Processing*, ed. J.C. Simon and R.M. Haralick, D. Reidel Publ. Co., Dordrecht, Holland, 1981.

W.H.H.J. Lunscher, 'The asymptotic optimal frequency domain filter for edge detection', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-5**(6), 678–680, 1983.

R. Machuca and A.L. Gilbert, 'Finding edges in noisy scenes', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-3**(1), 103–111, 1981.

I.D.G. Macleod, 'Comments on "Techniques for edge detection" ', *Proc. IEEE* **60**(3), 344, 1972.

A-R. Mansouri, A.S. Malowany, and M.D. Levine, 'Line detection in digital pictures: a hypothesis prediction/verification paradigm', *Computer Vision, Graphics, and Image Processing* 40(1), 95–114, 1987.

S. Marčelja, 'Mathematical description of the responses of simple cortical cells', *J. Optical Soc. Am.* 70(11), 1297–1300, 1980.

D. Marr and E.C. Hildreth, 'Theory of edge detection', *Proc. Roy. Soc. Lond.* B207, 187–217, 1980.

D. Marr, *Vision*, W.H. Freeman and Co., San Francisco, 1982.

M. Mazumdar, B.K. Sinha, and C.C. Li, 'A comparison of several estimates of edge point in noisy digital data across a step edge', pp. 27–33 in *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, San Francisco, June 1985.

M.C. Morrone, D.C. Burr, and L. Maffei, 'Functional implications of cross-orientation inhibition of cortical visual cells. I. Neurophysiological evidence', *Proc. Roy. Soc. Lond.* B216, 335–354, 1982.

V.S. Nalwa and T.O. Binford, 'On detecting edges', *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-8(6), 699–714, 1986.

M. Pietikäinen and A. Rosenfeld, 'Image segmentation by texture using pyramid node linking', *IEEE Trans. Systems, Man, and Cybernetics* SMC-11(12), 822–825, 1981.

M. Pietikäinen, A. Rosenfeld, and L.S. Davis, 'Texture classification using averages of local pattern matches', pp. 301–303 in *Proc. 6th International Conference on Pattern Recognition*, Münich, October 1982.

C.R. Rao, *Linear Statistical Inference and its Applications*, 2nd ed., John Wiley & Sons, New York, 1973.

L.G. Roberts, 'Machine perception of three-dimensional solids', in *Optical and Electro-Optical Information Processing*, ed. J.T. Tippett *et al.*, MIT Press, Cambridge, Mass., 1965.

B. Sakitt and H.B. Barlow, 'A model for the economical encoding of the visual image in cerebral cortex', *Biological Cybernetics* **43**, 97–108, 1982.

M. Shah, A. Sood, and R. Jain, 'Pulse and staircase edge models', *Computer Vision, Graphics, and Image Processing* **34**, 321–343, 1986.

K.A. Stevens, 'The information content of texture gradients', *Biological Cybernetics* **42**(2), 95–105, 1981.

R.E. Suciu and A.P. Reeves, 'A comparison of differential and moment based edge detectors', pp. 97–102 in *Proc. IEEE Computer Soc. Conf. on Pattern Recognition and Image Processing*, Las Vegas, Nevada, June 1982.

A.J. Tabatabai and O.R. Mitchell, 'Edge location to subpixel values in digital imagery', *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-6**(2), 188–201, 1984.

W.B. Thompson, 'Textural boundary analysis', *IEEE Trans. Computers* **C-26**(3), 272–276, 1977.

F. Tomita and S. Tsuji, 'Extraction of multiple regions by smoothing in selected neighbourhoods', *IEEE Trans. Systems, Man, and Cybernetics* **SMC-7**, 107–109, 1977.

M.R. Turner, 'Texture discrimination by Gabor functions', *Biological Cybernetics* **55**, 71–82, 1986.

L. Van Gool, P. Dewaele, and A. Oosterlinck, 'Texture analysis anno 1983', *Computer Vision, Graphics, and Image Processing* **29**, 336–357, 1985.

A. Weber, 'Image data base', USCIPI report 1070, Image Processing Institute, Uni. of Southern California, Los Angeles, California, 1983.

H. Wechsler, 'Texture analysis—a survey', *Signal Processing* **2**, 271–282, 1980.

D. Wermser and C-E. Liedtke, 'Texture analysis using a model of the visual system', pp. 1078–1080 in *Proc. 6th International Conference on Pattern Recognition*, Münich, October 1982.

D. Wermser, 'Unsupervised segmentation by use of a texture gradient', pp. 1114–1116 in *Proc. 7th International Conference on Pattern Recognition*, Montreal, July/August 1984.

A.P. Witkin, 'Recovering surface shape and orientation from shading and occluding boundaries', *Artificial Intelligence* **17**, 17–45, 1981.

M. Yachida, M. Ikeda, and S. Tsuji, 'Boundary detection of textured regions', pp. 992–994 in *Proc. 6th International Joint Conference on Artificial Intelligence*, Tokyo, August 1979.

A.L. Zobrist and W.B. Thompson, 'Building a distance function for gestalt grouping', *IEEE Trans. Computers* **C-24**(7), 718–728, 1975.

S.W. Zucker, R.A. Hummel, and A. Rosenfeld, 'An application of relaxation labeling to line and curve enhancement', *IEEE Trans. Computers* **C-26**(4), 394–403, 1977.