University of Southampton

Colour Object Recognition Using Shape-based Aspects

by

Richard Ian Taylor

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

September 1992

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Colour Object Recognition Using Shape-based Aspects

by Richard Ian Taylor

One of the main aims of Computer Vision research is the recognition of three-dimensional objects in two-dimensional images. Object recognition can only be achieved if the system has some knowledge, or a model, of the objects that may appear in the scene.

This thesis is concerned with the development of a model for three-dimensional objects which is amenable to the identification of objects in real, full-colour, images. The model presented is viewer-centred, representing the object as a list of unique two-dimensional views, or *Aspects*. Each *Aspect* represents all the views in which a given set of object surfaces are visible, either fully or partially. An *Aspect* is thus described primarily in terms of the two-dimensional shapes which correspond to the projections of the object surfaces. Additional surface information, such as colour and texture, can easily be attached to the representation since it is face-based rather than edge-based. The decomposition of complex objects into simpler *parts* is also addressed.

The recognition of objects in real images is demonstrated by first segmenting the colour image into consistent regions using a novel combination of region-growing and the TekHVC colour space. Regions which potentially correspond to the same object surface are then grouped together and the shape of the boundary is examined using a new two-dimensional shape descriptor called the *Fractal Shape Signature*. Identified shapes are then compiled into a data-structure called the *Shape Association Graph* which has the same form as an object *Aspect* and can therefore be matched without any complex transformations.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost I would like to thank my supervisor, Dr Paul Lewis, for his constant support over the last three years. Without his interest, encouragement and foresight I would not have completed this thesis on time, if at all. Cheers Paul.

Thanks must also go to my colleagues in the Image and Media lab who have offered constructive criticism and support, especially Mark Dobie.

Thanks also to Dr Joann Taylor of Tektronix Laboratories for providing all the technical details of her team's new colour space.

I would also like to thank the authors of the free software which has proved invaluable to my research. In particular I am most grateful for the time and efforts of the Free Software Foundation, Jef Poskanzer for PBMPLUS, all X-Windows contributors and the authors and maintainers of LaTeX.

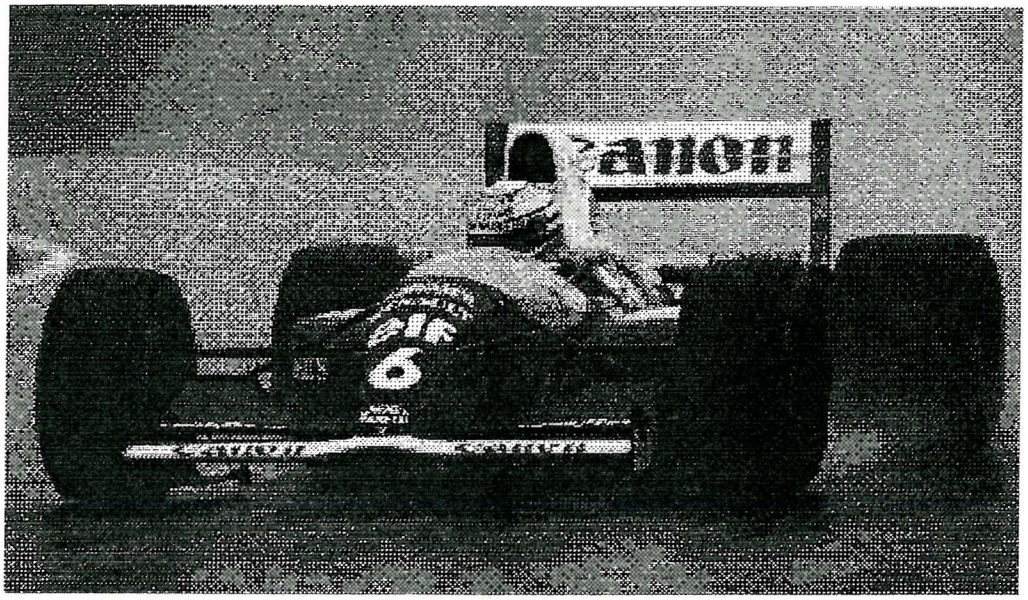
Finally, I would like to acknowledge the financial support of the Science and Engineering Research Council.

1 Introduction

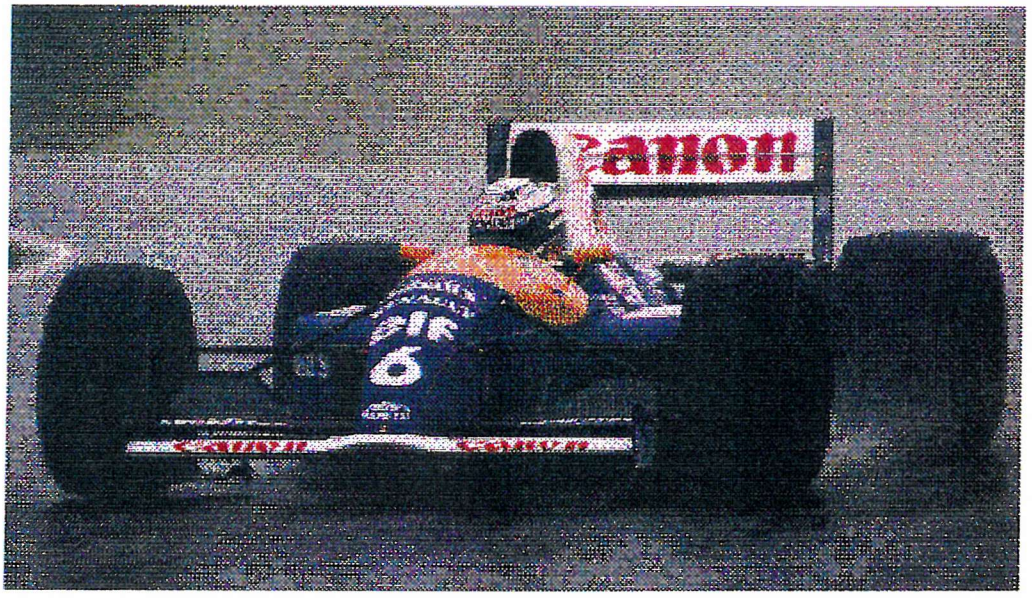
Vision is something we all take for granted but this one sense is so useful that we would be completely lost without it. Over the last few decades many researchers have been working towards the goal of giving computers sight: the field of Computer Vision has come a long way from the early experiments with poor quality images but it still has a long way to go to achieve even a small fraction of the performance of the human visual system.

One of the problems in Computer Vision which has attracted a lot of attention is the recognition of three-dimensional objects in images. There are many applications where this ability is desirable and of course it is essential for general-purpose vision. As a step toward general object recognition, the aim of the work described in this thesis is to develop a compact model of three-dimensional objects and to show how objects described by such models can be identified in real images.

Over the last few years there has been significant improvement in the performance (and a reduction in cost) of imaging hardware. We now have access to good quality, full colour, images which are believed to be optically superior to those captured by the human visual system. Although colour cannot solve all our problems, it can often make things much easier. In a very tightly constrained system we can often recognise objects solely by their colour (eg. the balls in snooker): in a less constrained system colour is more likely to be used as well as other features like shape and texture. Consider the two images in figure 1.1, both show the same view of the same car. It is quite easy to see that both images contain a racing car but in the colour image, if we know that a Williams is blue and yellow with bold red letters on the wings, then we can tell very quickly what type of racing car is in the picture.



(a) Monochrome image of racing car



(b) Colour image of racing car

Figure 1.1: Monochrome vs Colour

A digital image is just a huge array of thousands of pixels. In order to get a more manageable representation, the first step in most Computer Vision systems is to segment the image into consistent parts. By consistent we mean areas which share some common feature such as colour or texture. This thesis will concentrate on the use of colour as the feature for segmentation but will bear in mind the extension of the techniques described to the handling of textured surfaces.

Image segmentation usually involves looking for groups of pixels where the features are changing (ie. edges) or where the features are similar (ie. regions). Therefore, in order to segment an image on the basis of colour we need a model of colour which will enable the consistent measurement of differences in colour between pixels. This thesis is therefore initially concerned with the problems of colour measurement and the application of colour metrics to image segmentation. A novel approach to colour measurement is proposed and algorithms for colour segmentation are evaluated.

Given a segmented image there is little agreement on the best way forward to recognising objects. Any approach to object recognition is greatly influenced, if not completely determined, by the way in which objects are modelled. We have already noted that in the general case colour or texture is not enough and that some characterisation of an object's shape is likely to be required. Since recognition is the process of finding instances of models in images, it is logical that models which use features that appear directly in images (ie. two-dimensional features) will be easier to identify than models which require complicated transformations between two-dimensional (2D) image features and three-dimensional (3D) model features.

The approach proposed in this thesis is therefore to model 3D objects as a number of 2D views, or *Aspects*, which fully describe the object in a manner which is amenable to locating it in an image. The 2D features chosen are the outlines of the segmented image regions, which correspond to the projections of object surfaces. The second part of this thesis will deal with the description of these 2D boundaries, bearing in mind that they are the projections of 3D surfaces, and a new 2D shape description based on the principles of Fractal Geometry will be presented. The third part of the thesis will then use this description to develop image and object representations with the same

form and will demonstrate how they can be matched to achieve object recognition.

Using these representations we can think of a scene as being described by four layers as illustrated in figure 1.2. The bottom layer (pixels) is the starting point and the top layer (objects) is the final objective: in between we have regions (dotted lines) and shapes (dashed lines). How we move from one layer to the next is discussed in the following chapters but we should not limit ourselves to thinking of this as four separate stages. There is no fundamental reason why the image should be completely segmented into regions, just as there is no reason why all the regions should be tested for all shapes or why all shapes should be tested for all objects.

There is a tendency for Computer Vision systems to slip into a pipeline approach simply because processes like segmentation and shape analysis are thought of in isolation without consideration of a global visual process. Although the following chapters consider the transitions between *layers* in sequence, we should bear in mind that all the layers can interact and in the final chapter we will discuss this interaction in more detail.

The remainder of this thesis is divided up as follows,

Chapter 2, a review of the currently available colour models and some of the algorithms which have employed them for segmenting colour images.

Chapter 3, two new colour segmentation algorithms are presented and demonstrated using three different colour models. A full comparison is made and the best combination is selected for this application.

Chapter 4, a review of existing approaches to the description of two-dimensional shapes. The problems of consistent representation are discussed and the specifications of an ideal shape description are drawn up.

Chapter 5, a new shape description, the *fractal signature*, is described and compared to an established method. Results are also given for the recognition of two-dimensional shapes from three-dimensional viewpoints to facilitate the extension of the technique to identifying the 2D projections of general 3D surfaces.

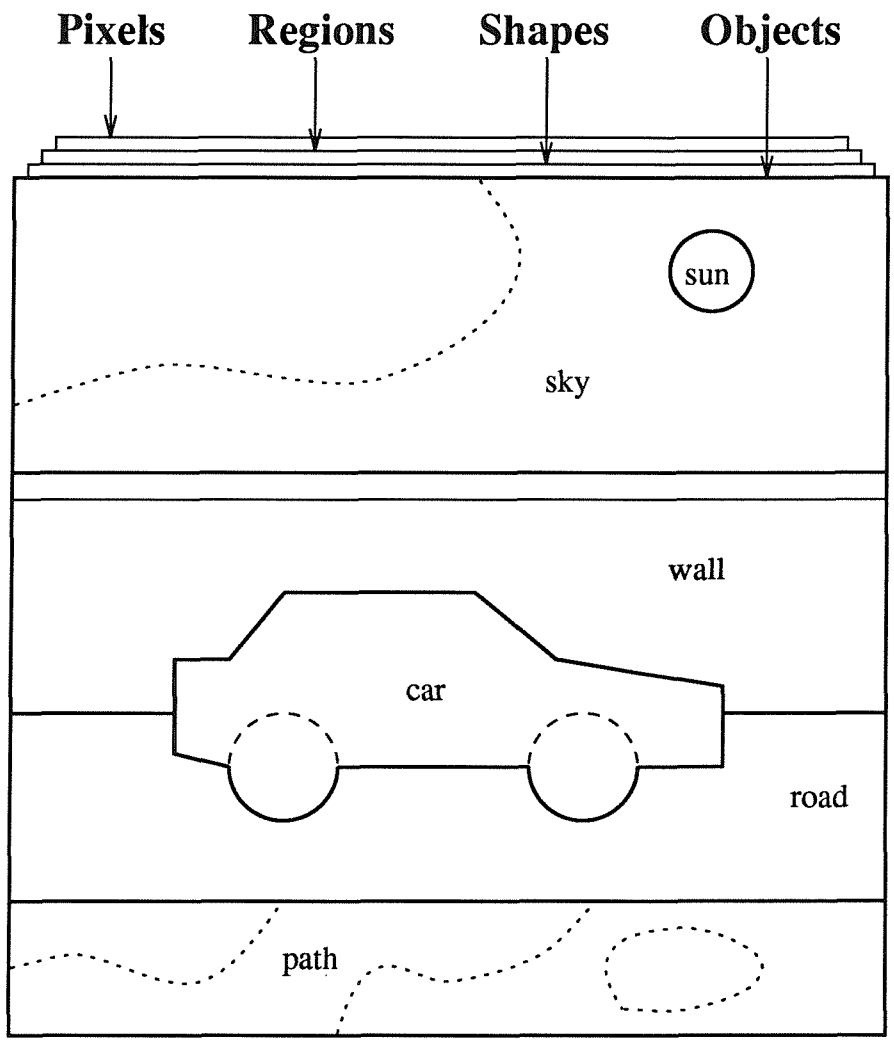


Figure 1.2: Overview

Chapter 6, a review of the literature on previous three-dimensional object recognition systems. Both object-centred and viewer-centred paradigms are discussed and compared.

Chapter 7, a new object recognition system is described which uses viewer-centred object models. Multiple views, or *aspects*, are used to describe objects in a manner similar to the description extracted from the image. Matching of object models to extracted image features is discussed and demonstrated.

Chapter 8, the previous chapters are summarised and applications of the newly presented algorithms are discussed.

2 Colour Image Segmentation

Early computer vision systems were restricted to the use of monochrome or so-called grey-level images. Now that hardware that can handle colour images is readily available, researchers have begun to take advantage of the strong visual cues provided by colour. Indeed, colour is the *main* cue in Ballard's renowned *Animate Vision* [4] system where objects are represented by their colour histograms.

In chapter 1 we discussed the need for defining exactly what colour is so that we can segment parts of an image which are "uniformly" coloured. In the following sections we will first look at how colour has been defined in the past and then go on to see how these definitions have been used in the segmentation of colour images.

2.1 Colour Spaces

A colour space is basically a coordinate system for specifying a set of colours. In a monochrome system our colour space is simply a straight line representing a range of intensities ie. a 1-dimensional space. In a colour system we have a chromatic content as well as an intensity content, this chromatic content is itself two-dimensional since colours have the two independent properties of hue and purity. Hence, a colour space will be 3-dimensional to allow colours to vary independently in intensity (light-dark), hue (red, yellow, green, cyan, blue, magenta) and purity (pale-bright).

Colour is a subjective feature and much work has been done on colour theory and perceptual models [22, 63, 64]. Many colour spaces have been developed but here we will concentrate on those spaces which have been used, or show promise for use, in computer vision systems.

2.1.1 CIE Colour Standard

For the purpose of standardisation the Commission Internationale de l'Éclairage (CIE) defined, in 1931, the wavelengths of the so-called *primary* colours: red at 700nm, green at 546.1nm and blue at 435.8nm. However, these are not true *primaries* since they cannot be combined additively to produce *all* the spectral colours. Hence, the CIE also defined three primary colours (X,Y,Z) which could be combined, all with positive weights, to produce all visible colours (the weights to produce all the visible spectral colours were tabulated in 1nm steps). These primaries have become an international standard for specifying colour.

The CIE colour space is shown in figure 2.1. Planes $X+Y+Z=k$ contain colours of equal intensity k. The three primaries are themselves invisible, all visible colours are contained within the horseshoe-base-pyramid shown. The bright, saturated, colours run around the edge of the base of the pyramid and get paler towards the white-point in the "centre". Greys run from the white-point to the origin ie. the grey-scale is a simple projection of the three dimensional colour space onto a one dimensional line.

By normalising colours for intensity we can produce *chromaticity* values which are independent of the amount of luminous energy,

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z} \tag{2.1}$$

ie. we project all colours onto the plane $x+y+z=1$. Given x and y we can determine z since $z=1-x-y$, therefore z is really redundant. If we project this plane onto the plane $z=0$ then we obtain the CIE chromaticity diagram show in figure 2.2.

The curved part of the horse-shoe boundary contains all the spectrally pure colours (wavelengths given in nanometres) and the interior (plus the straight part of the boundary) contains combinations of them. The whole region represents *all* visible colours, discounting luminance-related effects eg. brown is "dark orange" just as black is "dark white".

A useful property of the CIE chromaticity diagram is the way colours combine.

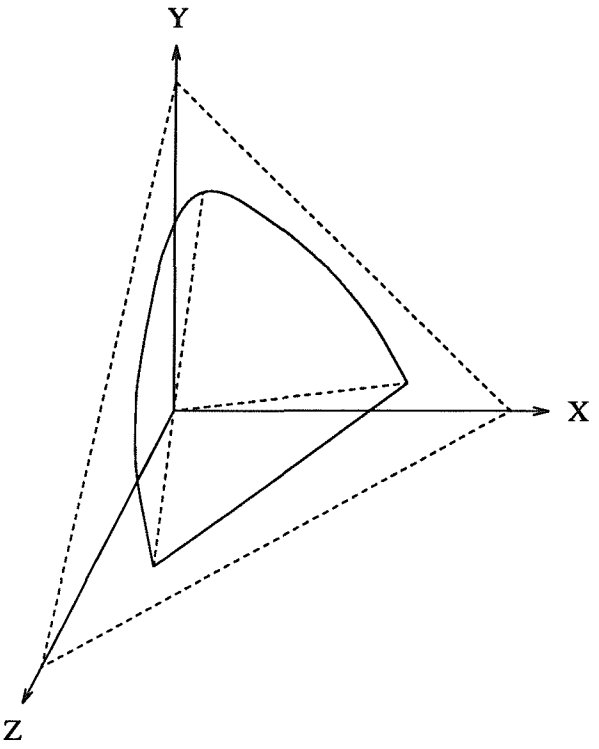


Figure 2.1: CIE Colour Space

Given a pair of colours, any colour on the line between them can be generated by combining them with some positive weights. This means that given three "primary" colours, the gamut of colours that can be generated from them (with positive weights) are those colours lying in the triangle with vertices at those primary colours. Since the base of the CIE space is convex and not triangular, it follows that there is no set of three visible primaries which can generate *all* visible colours. This is not generally a problem since a reasonable choice of primaries can cover *most* of the visible colours. To illustrate this, figure 2.3 shows the part of the diagram obtained from a typical set of red, green and blue primaries.

Whilst CIE space is a useful reference, there is a problem in that equal geometric distances within the space do not, in general, correspond to equal perceptual changes in colour. In 1976 the CIE developed the CIELUV space [30] to meet this need. If the position of *white* is defined as (X_w, Y_w, Z_w) then the space is defined for $X \leq X_w$,

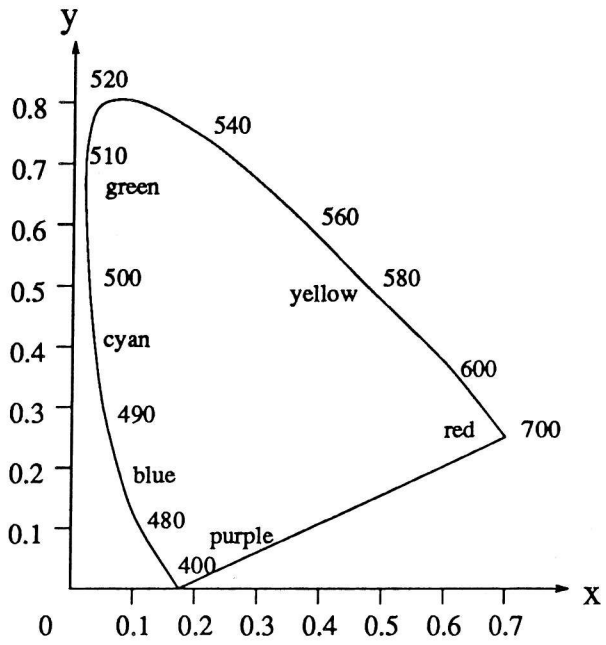


Figure 2.2: CIE Chromaticity Diagram

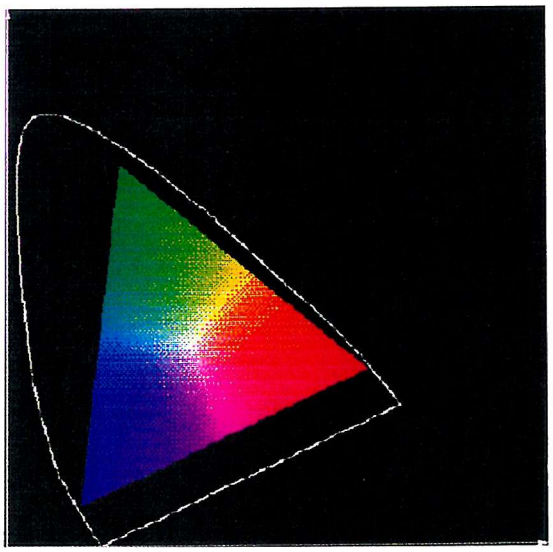


Figure 2.3: RGB Gamut

$Y \leq Y_w$ and $Z \leq Z_w$ as

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_w}\right)^{\frac{1}{3}} - 16 & \frac{Y}{Y_w} \geq 0.01 \\ 903\left(\frac{Y}{Y_w}\right) & \frac{Y}{Y_w} < 0.01 \end{cases} \tag{2.2}$$

$$u^* = 13(u' - u'_w)L^* \tag{2.3}$$

$$v^* = 13(v' - v'_w)L^* \tag{2.4}$$

where,

$$u' = \frac{4X}{X + 15Y + 3Z} \qquad v' = \frac{9Y}{X + 15Y + 3Z} \tag{2.5}$$

$$u'_w = \frac{4X_w}{X_w + 15Y_w + 3Z_w} \qquad v'_w = \frac{9Y_w}{X_w + 15Y_w + 3Z_w} \tag{2.6}$$

This CIELUV space is based on the same opponent-colour theory that has been used to describe human colour vision [50]. The principle behind opponent-colour theory is that colours are encoded as light-dark, red-green and yellow-blue ie. a colour cannot be light *and* dark, or red *and* green, or yellow *and* blue, but a colour can be red *and* yellow, or light *and* blue. In CIELUV, L^* represents lightness, zero means black and the more positive the value the lighter the colour, up to 100 which is white; positive values of u^* represent reds and negative values represent greens; similarly, positive v^* corresponds to the yellows and negative values correspond to the blues.

The shape of the CIELUV space is an irregular spheroid which is a rounded version of the CIE space (figure 2.1). Corresponding to the CIELUV space is the Uniform Chromaticity Scales (UCS) diagram: this is a plot of u' against v' and is a linear transformation of the xy Chromaticity diagram. Standard colours (independent of intensity) can therefore be described as a (u', v') point or an (x, y) point.

The deviations from uniformity in the UCS diagram occur mostly at the limits of the spectral locus (ie. the boundary between visible and invisible colours). Fortunately, the colour gamut used by most computer vision hardware lies in the most uniform part of the diagram.

As well as the LUV space (of which there are several variations), the CIE also defined

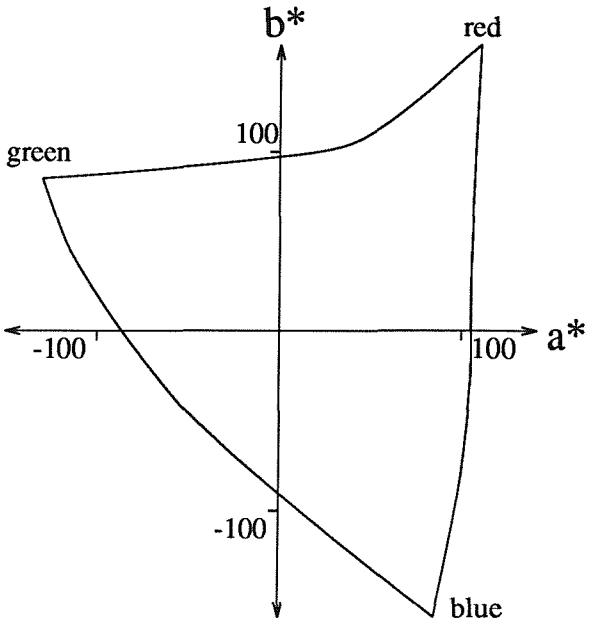


Figure 2.4: CIELAB Colour Space

a colour space called CIELAB. This has similar properties to CIELUV but was designed to correspond to the empirical Munsell Colour System [73], it is defined as

$$a^* = 500\left[\left(\frac{X}{X_w}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_w}\right)^{\frac{1}{3}}\right]$$
$$b^* = 200\left[\left(\frac{Y}{Y_w}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_w}\right)^{\frac{1}{3}}\right]$$

L^* is the same as in CIELUV. This is clearly a non-linear transformation of the chromaticity diagram, as illustrated by figure 2.4 which shows the transformation into the a^*b^* plane of the RGB triangle in figure 2.3 for $Y = 50$. The CIELAB space is believed to be a slightly less uniform than CIELUV but, never the less, is often used because the Munsell system has been widely used in industry.

2.1.2 RGB Colour Model

Most commercial colour imaging systems mimic a colour image by using three filters which respond to the red, green and blue components of the incident light, producing three images. The three (R,G,B) images appear as a single image when displayed on

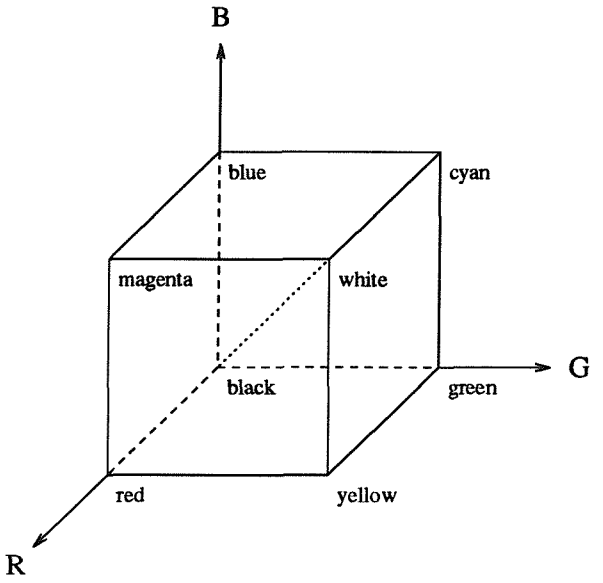


Figure 2.5: RGB Colour Cube

a monitor which *adds* the component images together. This is therefore the starting point of most colour image processing systems.

The Red, Green, Blue model is shown in figure 2.5. Each colour is specified by an (R,G,B) triple which is a vector to a point in the cube. The greys lie on the dotted line shown.

This is a hardware model. The primaries red, green and blue are chosen because they lie in the "corners" of the CIE space and therefore cover the great majority of the visible colours. Whilst the (R,G,B) triple describes the colour completely (and in an easily implementable fashion) it does not carry direct semantic information about the colour ie. a person cannot visualise a colour given its (R,G,B) triple, they can only tell how red, green or blue it is. Additionally, the RGB space is not perceptually linear so it is not useful for colour comparison based on geometric separation within the cube.

Given the positions of each of the primaries in CIE space, the RGB values $[0 \ldots 1]$ can

be converted into CIE coordinates.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.7}$$

For the NTSC [1] primaries used by our camera we have the following chromaticities

	Red	Green	Blue	White
x	0.67	0.21	0.14	0.31
y	0.33	0.71	0.08	0.32

Remembering equation 2.1 we know that for the red primary

$$x_r = \frac{X_r}{X_r + Y_r + Z_r} \, , \; y_r = \frac{Y_r}{X_r + Y_r + Z_r} \, , \; z_r = \frac{Z_r}{X_r + Y_r + Z_r}$$

and defining $S_r = X_r + Y_r + Z_r$ we can write

$$X_r = x_r S_r \, , \; Y_r = y_r S_r \, , \; Z_r = z_r S_r$$

with similar definitions for the other two primaries.

Equation 2.7 now becomes

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_r S_r & x_g S_g & x_b S_b \\ y_r S_r & y_g S_g & y_b S_b \\ z_r S_r & z_g S_g & z_b S_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.8}$$

$$= \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{bmatrix} \begin{bmatrix} S_r R \\ S_g G \\ S_b B \end{bmatrix} \tag{2.9}$$

The unknowns S_r, S_g, S_b can be found by considering the white-point, which results

[1] The American "National Television System Committee".

from $R = G = B = 1$.

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{bmatrix} \begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} \tag{2.10}$$

We can arbitrarily select the luminance of the white-point, Y_w, and hence it follows that

$$X_w = \frac{x_w}{y_w} Y_w \ , \ Z_w = \frac{z_w}{y_w} Y_w$$

Choosing $Y_w = 1$ we get $X_w = 0.98$ and $Z_w = 1.18$. If we substitute the values given into equation 2.10 and solve for S_r, S_g, S_b then we can substitute into equation 2.8 and finally we get

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.61 & 0.17 & 0.20 \\ 0.30 & 0.59 & 0.11 \\ 0.00 & 0.07 & 1.12 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Hence, we can convert the RGB values produced by our hardware to CIEXYZ coordinates and therefore into any of the perceptually uniform colour spaces defined in those terms.

2.1.3 YIQ Colour Model

This is the model used for colour TV broadcasting. It is a simple transformation of the (standard NTSC) RGB space.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.30 & 0.59 & 0.11 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The Y component (same as CIE Y) carries the intensity information and is used by black and white TV. The I and Q components carry the remainder of the colour

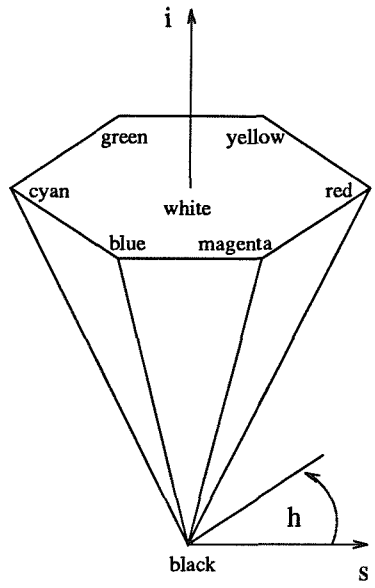


Figure 2.6: HSI Colour Hexcone

information. Like RGB this model is for hardware use and is primarily designed for efficient transmission of colour signals rather than their interpretation.

2.1.4 HSI Colour Model

The Hue, Saturation and Intensity colour model due to Smith [92] (also known as HSV or HSB - V for value, B for brightness) is based on the hexcone shown in figure 2.6. In this colour space, intensity is a rectangular coordinate and hue and saturation are polar coordinates.

Hue represents the colour name eg. red, yellow, green, cyan, blue, magenta. Saturation is a measure of the purity of the colour ie. how much white is in it, a low saturation means a pale colour (lots of white) and a high saturation a pure or bright colour (very little white). Intensity is the analogue of grey-level from monochrome images, it is a measure of how light or dark the colour is. The advantage of the HSI system is that it is a *user based* model, in which the axes all have some semantic meaning.

The HSI space is derived directly from the RGB cube and hence there is no calibration

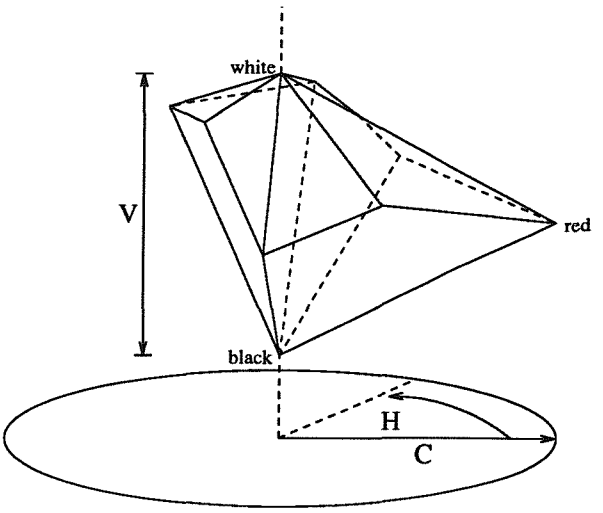


Figure 2.7: TekHVC Colour Space

of the primaries. The hexagon of intensity i is obtained by projecting the planes $R = i$, $G = i$ and $B = i$ onto the plane perpendicular to the white-black line eg. the intensity=1 hexagon is what you see if you look at the whole RGB cube along the white-black line.

2.1.5 HLS Colour Model

The Hue, Lightness and Saturation model is very similar to the HSI model. It is based on a double hexcone obtained by stretching the white point of HSI upwards to twice the original height. This is an attempt to improve on the intensity/value/brightness coordinate. In HSI all the most intense colours eg. white, red, green, blue, are not perceived to be equally bright, white is clearly brighter than all the others which, themselves, are not equally bright. In HLS, white is made *twice* as *light* as red, green, blue etc. In some ways this is just as bad, if not worse, than the HSI intensity since the colours red, green, blue etc. still have the same lightness and additionally are equally as light as 50% grey.

2.1.6 HVC Colour Model

The Hue, Value and Chroma colour model has recently (1988) been developed by Tektronix Laboratories [97]. Known as TekHVC^{TM}, or more simply HVC, it is a

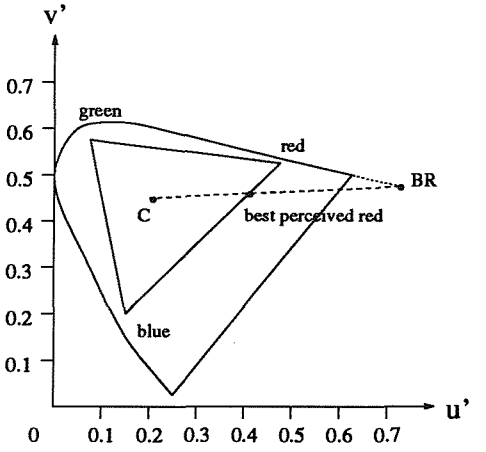


Figure 2.8: CIE UCS Diagram

linear transformation of the CIELUV space with a semantically meaningful coordinate system. Hue is an angle ($0 \ldots 360^o$) representing the name of the colour (red, yellow etc.); Value is the equivalent of CIELUV's lightness L^* ($0 \ldots 100$); Chroma is roughly equivalent to saturation or purity except that it is an absolute measure which allows some colours to exhibit a larger range of Chroma than others eg. red$\approx$($0 \ldots 130$) and green$\approx$($0 \ldots 105$) and blue $\approx$($0 \ldots 80$).

An approximation of the space is shown in figure 2.7. The HVC system is patented but is offered as an open standard. The definition is

$$H = \tan^{-1}\frac{v^*}{u^*} - \psi \tag{2.11}$$

$$V = L^* \tag{2.12}$$

$$C = (7.50725)\sqrt{u^{*2} + v^{*2}} \tag{2.13}$$

The hue offset, ψ, in equation 2.11 is chosen such that $H = 0$ represents the "best" achievable red. Tektronix have conducted psychophysical experiments which determine the average position of the best red to be $u' = 0.7127$, $v' = 0.4931$. This is not actually a visible colour but the intersection of a line joining the white-point to the "best perceived red" and the extension of the UCS curve as shown in figure 2.8.

Given the position of the best red, the value of the hue offset is the angle between

the line joining $(0.7127, 0.4931)$ to the white-point and the u' axis in the UCS diagram.

$$\psi = \tan^{-1} \frac{0.4931 - v'_w}{0.7127 - u'_w}$$

For the standard C illuminant ($u' = 0.2009$, $v' = 0.4608$) the value of the hue offset is 3.6112^o.

In equation 2.13 the constant 7.50725 is also the result of Tektronix's psychophysical experiments and is believed to make the perceptual impact of a numeric change in Value and Chroma equal. This is the major difference between HVC and CIELUV since it actually alters the distances between colours in the space, effectively *widening* it by a factor of 7.50725. Without conducting our own psychophysical experiments we must assume that HVC is now the most perceptually uniform colour space available.

Figures 2.9 and 2.10 show a series of sections through the part of the HVC space obtainable from the NTSC RGB primaries using the C illuminant as the white-point (ie. the colours contained within the triangle in figure 2.8). In figure 2.9 we have (from left-right and top-bottom) the planes $H = 0^o, 180^o$ to $H = 169^o, 349^o$ and in figure 2.10 we have the planes $V = 10$ to $V = 99$.

Clearly, the great advantage of HVC over the previously discussed colour spaces is that it is both perceptually uniform *and* has a semantically meaningful coordinate system.

2.2 Colour Segmentation Techniques

Early attempts at colour image segmentation simply applied the techniques that had been successful in monochrome images to each of the RGB images separately and then combined the responses. This was almost invariably found to give unsatisfactory results. Tajima [95] pointed out that since many operations on images are based on colour differences it is to be expected that using uniform colour spaces will produce results closer to human expectation than using the RGB space.

Some researchers have employed the RGB model more successfully in statistical

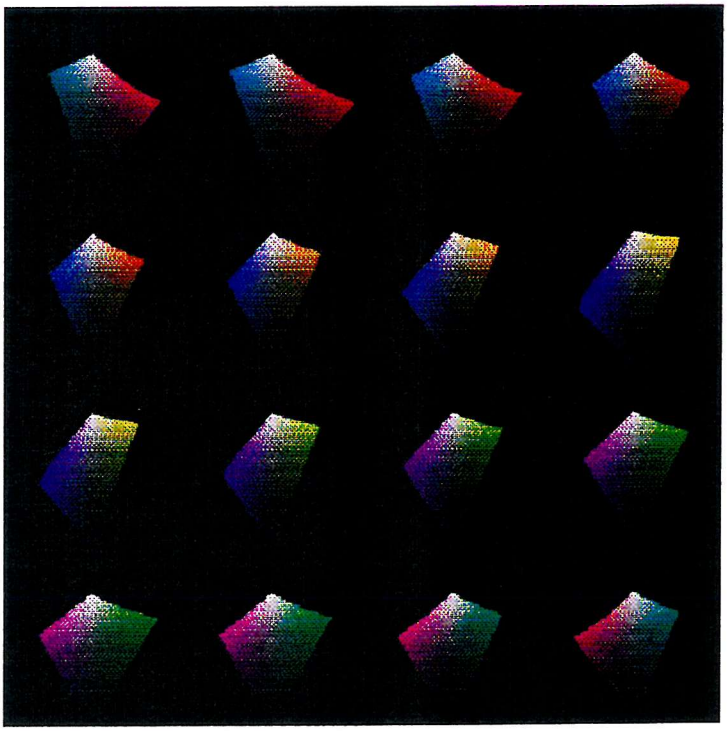


Figure 2.9: TekHVC Space (Vertical Sections)

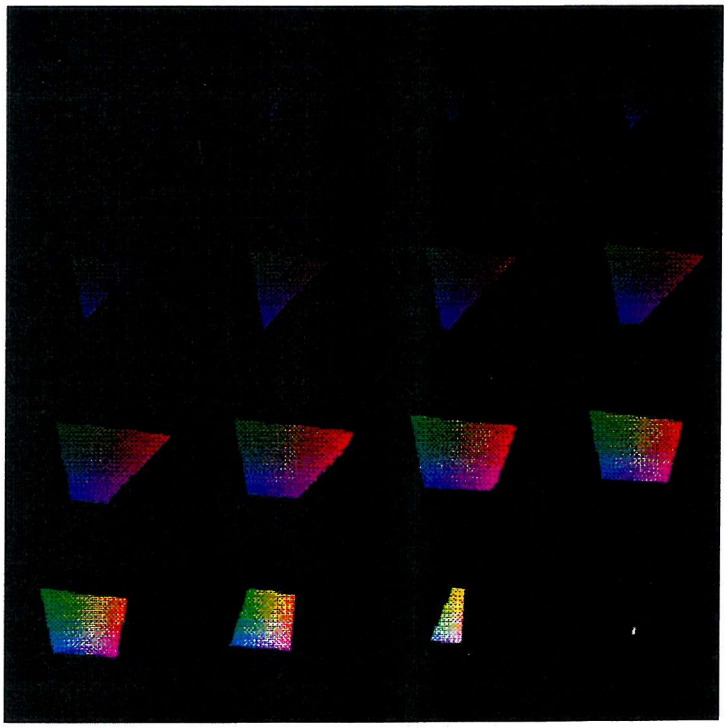


Figure 2.10: TekHVC Space (Horizontal Sections)

techniques where the segmentation task is modelled as data fusion from different, but related, sources. Wright [110] proposed the use of a Markov random field approach to maximise the conditional probabilities of region labels and boundaries, assuming that the boundaries within a colour plane are related to those in other planes. The process is computationally intensive since it involves a simulated annealing process to optimise a cost function. Since the results, even on artificial images, are not outstanding we must conclude that the initial assumptions are flawed.

Early research by Feldman and Yakimovsky [39] into knowledge-guided segmentation used a statistical approach to partitioning a colour image into meaningful regions. The region properties used included colour hue, saturation and intensity as well as several geometric properties. The colour analysis is quite crude because the colour space is coarsely quantised with only eight values of hue and four each of saturation and intensity.

A later knowledge-guided segmentation technique due to Nazif and Levine [74] also utilised colour information. This is a rule-based approach so the knowledge is expressed explicitly and is intended to embody the in-built knowledge of many segmentation heuristics in a controllable manner. Little colour knowledge is actually used and the reasonably good performance of the system as a whole is a credit to the other knowledge employed. The RGB model is used to compare the colours of regions and they are merged if the differences in each of the components is less than some threshold. The large amount of other low-level knowledge is required because, as the authors say, "Analysing an image using only this rule as the merging criterion produces many incorrect data configurations."

2.2.1 Colour Clustering

A frequently referenced early colour image segmentation technique is that due to Ohlander, Price and Reddy [76]. They used a recursive region splitting method based on histograms in three different colour spaces ie. RGB, HSI and YIQ. The technique basically takes a region and calculates nine histograms, one for each plane of each

colour space, from these histograms the best [2] of all the peaks in all the histograms is used to threshold the region. After thresholding, connected pixels which have the same label (ie. 0 or 1) are grouped into new regions which can then be split further.

Ohlander, Price and Reddy's technique is an example of segmentation based on *colour clustering*. Three colour spaces are used to try and avoid the problem of clusters which are easily separable in 3D but are not separable in any of the three individual colour planes. Colour clustering in 3D has been performed by Sarabi and Aggarwal [88] using one colour space (xyY) but at great computational expense.

More recently, Celenk [23] has proposed a clustering technique based on the perceptually uniform CIELAB space (except he uses the polar form of a^*,b^*). In order to avoid the immense computational cost of three-dimensional clustering without simply thresholding in each colour plane separately (as Ohlander et al. do) Celenk first uses the 1D histograms to determine a set of decision volumes and then projects the volumes onto a minimum-error line for 1D thresholding. This enables all the colour information to be used at once in the thresholding process.

The results of Ohlander et al., and later of Celenk, depend very much on the type of image under consideration. For images containing a few distinctly coloured areas the results are generally very good but less evenly coloured regions tend to be split or merged arbitrarily. The major drawback with clustering type segmentation techniques is that they do not use any spatial information when deriving the clusters. Only when a cluster has been determined are the pixels with an appropriate colour linked together to form regions. This can lead to attempts to form regions from pixels which are not connected.

2.2.2 Colour Edge Detection

Nevatia [75] developed an early colour edge detector based on a YT_1T_2 colour model. Y is the luminance (the implication is that it is the same Y as in YIQ) and T_1 and T_2 are

[2] The best peak is determined by a combination of seven criteria which relate to the position, height and width of the peaks.

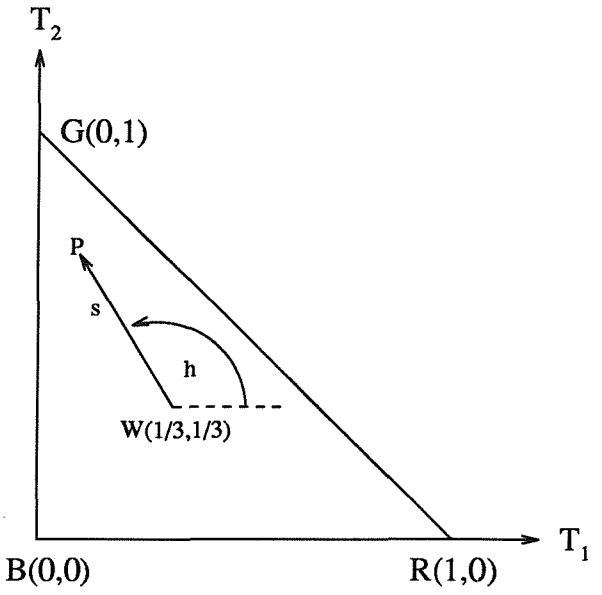


Figure 2.11: YT_1T_2 Colour Triangle

defined as

$$T_1 = \frac{R}{R+G+B}$$
$$T_2 = \frac{G}{R+G+B}$$

This colour triangle is shown in figure 2.11. Hue and saturation are defined as the polar position relative to the white point (1/3,1/3). Colour edges are detected by applying the Hueckel edge detector to the YT_1T_2 components with the constraint that the response to each component must have the same orientation. Edges in the hue and saturation components are inferred from the T_1T_2 edges (although Nevatia states that there is little point in this).

Nevatia concluded that most of the edge information lay in the intensity image except for scenes with low contrast or poor illumination. Furthermore, he suggests that colour is useful as a description for edges rather than a source of new edges. To some extent Nevatia's results are prejudiced by his colour model since the strengths of his chromatic edges are unreliable because of the non-uniformity of the YT_1T_2 colour triangle.

Colour has been utilised successfully in edge detection by several other researchers, the main problem being to produce a reliable definition of colour difference. Claxton and Kwok [31] defined colour difference in terms of separation in the HSI colour space with various modifications: for two colours (h_1, s_1, i_1) and (h_2, s_2, i_2) they define the colour difference ΔD as

$$\Delta D = \sqrt{(i_1 - i_2)^2 + \left(\ln \frac{|i_1 + i_2|}{2}\right)^2 (s_1^2 + s_2^2 - 2s_1 s_2 cos[h_1 - h_2])C}$$

where, C is the saturation index. The logarithmic term is an attempt to reduce the quasi-random effect of the chrominance term at low luminance. What this really means is that the authors are trying to make their colour space more perceptually uniform! By combining their colour difference with a Laplacian operator Claxton and Kwok produce a colour edge detector which gives quite good results in their sample outdoor scene.

Forsyth [40] proposed that the principles of human colour vision could be used to detect changes in colour. Whilst arguing that a considerable part of a colour signal is due to the illumination, Forsyth concludes that a sharp change in the colour signal is much more likely to be due to a sharp change in the reflected signal than a sharp change in the illumination. For his colour edge detector he combines the Fleck edge detector with the double-opponent cell believed to exist in humans. This cell responds to normalised Blue-Yellow and Red-Green signals and can thus detect spectral crosspoints. The results given are good although the images used are of coloured squares on a constant background and are therefore quite artificial.

2.2.3 Colour Region Analysis

Region-based techniques have also been developed to use colour. Chassery and Garbay [25] developed an iterative region growing technique for the segmentation of images of cells. In this constrained environment it is easy to find starting points by simple thresholding since cell nuclei are quite distinct from the rest of the cell and the background. The approach then proceeds to grow regions around the nuclei by

adding pixels which have a similar colour (defined by Euclidean distances in a variant of HSI space) to the present mean colour. Since the cells may touch, Chassery and Garbay also use a convexity criterion to prevent cell merging.

Meyer [71] has used a flooding process which *grows* regions at a constant rate from a set of marker points until they meet. This effectively means that pixels adjacent to the edge of a current region are added to the "nearest" region in distance order, "closest" first. Here, closest and nearest are defined as the smallest Euclidean distance, in the HLS colour space, between the pixel in question and its neighbour in the region. Meyer applies his technique successfully to the segmentation of paintings but the success of the method is very dependent on the selection of the initial markers (or seed points) and at present this must be done manually.

Vlachos and Constantinides [104] have recently proposed a graph-based region merging approach using the CIELAB colour model. Initially, each pixel is a separate node in the graph, linked to its (4-connected) neighbours by arcs which have a "strength" of half the Euclidean colour distance between the pixels. Nodes are then merged by sequentially removing the "weakest" arc and updating the node colours and the strengths of all the arcs connected to either of the merging nodes ie. if nodes i and j are merged and node k is connected to one of them then

$$\begin{aligned} \boldsymbol{C}_i(t+1) &= \frac{N_i(t)\boldsymbol{C}_i(t) + N_j(t)\boldsymbol{C}_j(t)}{N_i(t) + N_j(t)} \\ N_i(t+1) &= N_i(t) + N_j(t) \\ S_{i,k}(t+1) &= \frac{N_i(t+1)N_k(t+1)}{N_i(t+1) + N_k(t+1)} \left|\boldsymbol{C}_i(t+1) - \boldsymbol{C}_k(t+1)\right|^2 \end{aligned}$$

where $\boldsymbol{C}_i$ is the colour (L^*, a^*, b^*) of node i; N_i is the area of region i and $S_{i,k}$ is the strength of the arc between nodes i and k.

Hence, the representative colour for a region is the mean position of all the pixels in the colour space. The arc strength $S_{i,k}$ is actually the total increase in the *Square Euclidean Distance* (SED) between the segmentation and the original image which would result from merging nodes i and k. Thus, by removing the weakest arc, Vlachos and Constantinides always perform the merge which increases the SED by the least

amount (at that point in the segmentation). This clearly favours the merging of small regions before larger regions with the same colour difference.

The results given by Vlachos and Constantinides show an image segmented into exactly 4000 regions, this is not bad but appears over-segmented (ie. still too many regions). It is not clear whether or not they are proposing a fixed number of merges but it would clearly be better to merge regions until either the total SED reaches some limit or the weakest arc exceeds some minimum strength. The computational issues are also unaddressed since this approach, though appealing, is presumably highly computationally intensive due to the potentially huge size of the graph under consideration.

2.3 Conclusions

Of all the colour models available, the RGB model has probably received the most attention since it is "forced" on us by our hardware. Convenience is a poor substitute for good results and researchers soon realised that other colour spaces were worth exploring. The popular colour spaces are either user-based, with semantically meaningful co-ordinate systems, or close approximations to a perceptually uniform space. Until recently there were no colour spaces which had both of these properties. As such a space, TekHVC is likely to be widely used in the future.

One problem that was mentioned by Forsyth [40] is the dependence of the colour signal on the illumination ie. the colour we see is due to a combination of the reflectance properties of the object surface and the light shining on it. With regards to segmentation this is not generally a problem since (as Forsyth concludes) *changes* in colour are nearly always due to changing object properties rather than changing illumination except for effects like shadows. Problems arise when we try to compare the colour of a segmented region to some value stored in a database. Unless the illumination is the same then the two colours may be very different, even if they correspond to the same surface of the same object viewed from the same position. This problem of colour constancy has recently been addressed by several researchers; work by Tsukada and Ohta [102] and

Ho, Funt and Drew [47] has made significant progress in this difficult area.

Even given the limitations of the colour spaces used there has been no definitive general-purpose segmentation algorithm produced. This reflects the experience with monochrome images, the requirements of the *ideal segmentation* are image and application dependent, the desired edge or region properties being largely contradictory. As such it is very difficult to compare segmentation algorithms. Even if all the algorithms described in the previous section were applied to the same image then we would require some well defined criteria to compare them directly.

Since no *standard* images or comparison criteria can be agreed among researchers, we must compare results in a purely subjective manner. Some algorithms can of course be criticised on grounds other than segmentation quality, for instance the computational complexity of some algorithms makes them impractical for use in present-day systems.

Ohta, Kanade and Sakai [77] encountered the above problems when they attempted to evaluate colour spaces for region segmentation. Firstly, they used the Ohlander region splitting algorithm on eight different types of images with the Karhunen Loeve transformation of R, G and B as the features. Since these features are decorrelated (but very expensive to calculate) they are assumed to be ideal for the algorithm. The segmentations are then repeated using RGB, XYZ, YIQ, Lab, $U^*V^*W^*$([3]), YT_1T_2, HSI and $I_1I_2'I_3'$([4]) colour spaces. The results are then compared in a subjective manner since, as the authors state, "No quantitative evaluation procedure has been established for segmentation of natural scenes. We adopted eyeballs as the most reliable tool at present."

Using their visual comparison (which rightly, in my view, criticised undersegmentation more severely than oversegmentation) Ohta et al. concluded that their own $I_1I_2'I_3'$ space and the Lab space were the most suited to region segmentation using the Ohlander algorithm. This is not surprising since histogram-based techniques require features that are significantly decorrelated; as such, highly correlated spaces such as RGB and XYZ will do very poorly indeed. Other spaces are simply not suited to

[3]This colour space is similar to the CIELUV space.
[4]Defined by Ohta et al. as $I_1 = (R+G+B)/3, I_2' = (R-B), I_3' = (2G-R-B)/2$.

histogramming at all, for instance, any notion of hue will not be usable in very light or dark areas since hue can vary through its whole range without producing a significant change in colour.

The graph-based approach of Vlachos and Constantinides, if computationally feasible, would initially appear quite appealing. On further inspection the use of the SED seems less wise because it means that in the later stages of the process small regions with large colour differences will be merged ahead of much larger regions with much smaller colour differences. This would explain why the segmentation appears to be stopped "early" by the authors, since further segmentation would start merging in this unwanted fashion. This is clearly undesirable so the algorithm should be altered so that no pair of regions will be merged if the difference in colour is greater than some threshold.

Of the algorithms considered, Meyer's region grower gives the most appealing results. However, the manual selection of the marker points is a serious drawback. What is needed is either a means of automatically selecting the optimal marker points or a stopping criteria for growing regions from non-optimal marker points. These issues are discussed in the following chapter.

3 New Colour Segmentation Techniques

In the previous chapter I introduced several colour spaces and described how some of them have been used by other researchers to implement colour image segmentation techniques. We saw how the choice of colour space was, to a large extent, just as important as the choice of segmentation algorithm; this is caused by most segmentation algorithms being defined in terms of *differences* in colour between pixels.

The following sections of this chapter define exactly what we mean by terms like colour difference and average colour, with respect to the HVC colour space. We then use these definitions to describe two segmentation algorithms which have been developed by extending existing monochrome techniques. These extensions involve both changes to the basic algorithms as well as the change from monochrome to colour.

Both algorithms have been implemented using the RGB, HSI and HVC colour spaces so that they can be fully compared and their performance discussed with respect to the criteria introduced in chapter 2. The results of this work were presented at the *11th IAPR Conference 1992* [99].

3.1 Colour Metrics

Since the HVC space is perceptually uniform, we can define the scalar difference between two colours as being the Euclidean distance between their positions in the space ie. for two colours (h_1, v_1, c_1) and (h_2, v_2, c_2) the colour difference ΔC is given by

$$\Delta C = \sqrt{(v_1 - v_2)^2 + c_1^2 + c_2^2 - 2c_1c_2\cos(h_1 - h_2)} \tag{3.1}$$

This metric can now be used in a similar manner to the comparison of grey-levels in monochrome image processing: for instance, we could perform simple edge detection by calculating colour differences between pixels; or we could do simple region growing by linking pixels with small enough colour differences.

Using a perceptually uniform colour space also means that we can define the average colour $(\hat{h}, \hat{v}, \hat{c})$ of a set of N colours (h_i, v_i, c_i) to be the centroid of the positions in the colour space ie.

$$\hat{h} = \tan^{-1} \frac{\bar{q}}{\bar{p}} \tag{3.2}$$

$$\hat{v} = \frac{1}{N} \sum_{i=1}^{N} v_i \tag{3.3}$$

$$\hat{c} = \sqrt{\bar{p}^2 + \bar{q}^2} \tag{3.4}$$

where,

$$\bar{p} = \frac{1}{N} \sum_{i=1}^{N} c_i \cos h_i \tag{3.5}$$

$$\bar{q} = \frac{1}{N} \sum_{i=1}^{N} c_i \sin h_i \tag{3.6}$$

A logical progression from the colour difference and average colour is a metric to define the colour spread throughout a region: the most intuitive definition of colour spread is the RMS colour difference from the average colour ie.

$$D = \sqrt{\left(\frac{1}{N} \sum_{i=1}^{N} \Delta C([h_i, v_i, c_i], [\hat{h}, \hat{v}, \hat{c}])^2 \right)} \tag{3.7}$$

These simple metrics give us a consistent means of comparing individual colours and groups of colours as we shall see in the following sections. One question which we will repeatedly encounter is "how far apart can two colours get before they *look* different?". Using a perceptually uniform colour space we have the advantage of knowing that this distance will be the same (or very nearly the same) throughout the colour space. To illustrate the probable range of this "just different" distance figures 3.1 and 3.2

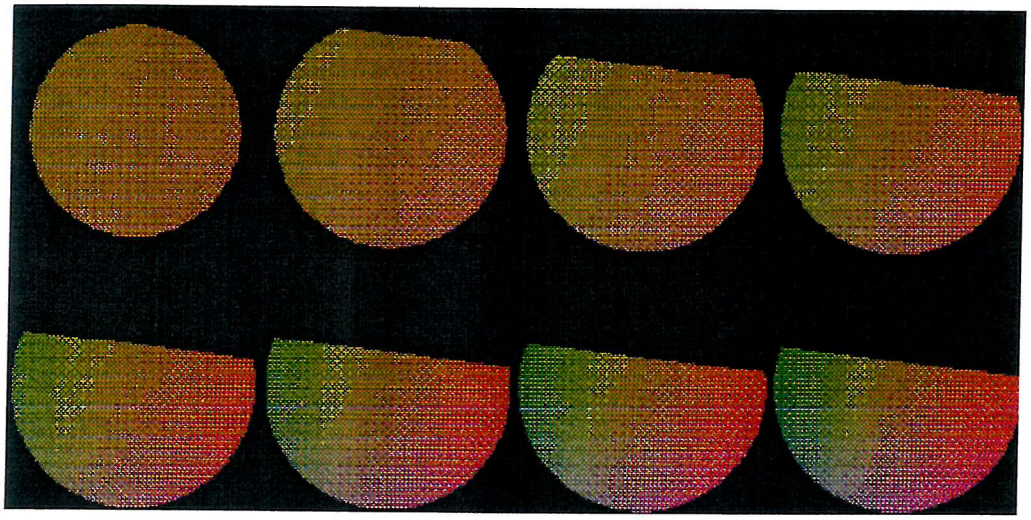


Figure 3.1: Constant Value Discs, Radius 5-40

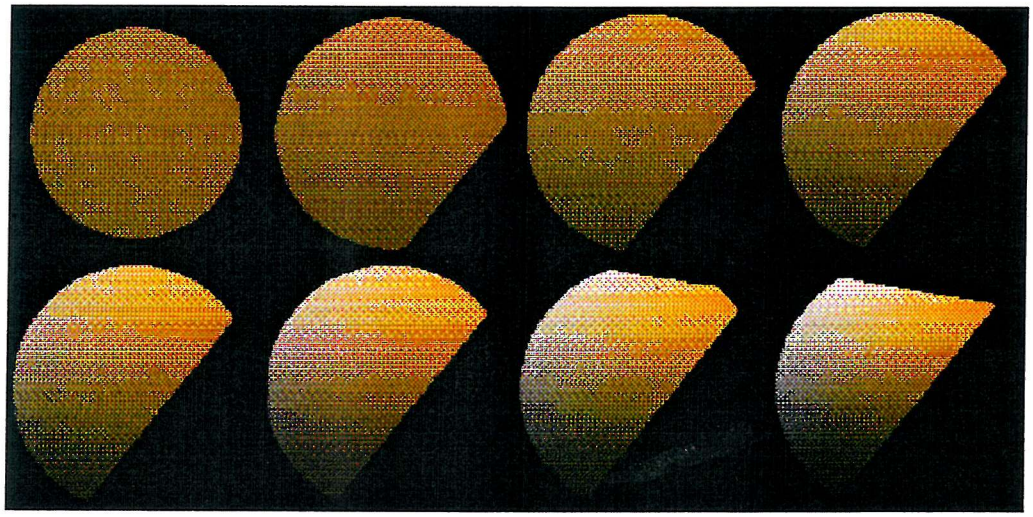


Figure 3.2: Constant Hue Discs, Radius 5-40

S/σ	D/S	$\rho(S)$
0	0.77	1.00
1	0.75	0.61
2	0.68	0.14
3	0.55	0.01
4	0.43	0.00

Table 3.1: Colour Spread with Gaussian Distribution

show various colour "discs" which are all centred on the HVC colour $(60°, 60, 40)$. The radii of the top row of discs are 5,10,15,20 and the bottom row are 25,30,35,40. In both the constant-value discs and constant-hue discs of radius 5 there is no distinguishable difference in colour across the disc: in the radius 10 discs the colour at the edges of the discs are just differentiable, hence we can make the assertion that colours separated by a distance of less than 20 will appear the same.

Relating colour spread to uniformity of colour obviously depends on the actual distribution of the set of colours. If the colours are distributed in a normal manner around the average, ie. the density $\rho(r)$ of colours at a distance r from the mean is given by

$$\rho(r) \propto \exp(-\frac{r^2}{2\sigma^2})$$

where σ is the standard deviation, then if the maximum distance is S, table 3.1 shows the spreads resulting from different values of S/σ. Exactly how we choose a maximum spread depends on the segmentation algorithm we are using and will be explained in the following sections.

3.1.1 Implementation of Colour Metrics

All the images used in this thesis were captured using a colour CCD camera. The images obtained contain 24bits of colour information per pixel (8bits for each of red, green and blue). These RGB values are first converted to floating point numbers (scaled to the range $[0 \ldots 1]$) and then the triple is transformed into a CIEXYZ triple using equation 2.11.

Once we have a CIEXYZ triple we can convert it easily to a CIELUV triple using

equations 2.2 to 2.6 with the standard C illuminant ($X_w = 0.98, Y_w = 1.00, Z_w = 1.18$).

Given CIELUV values the obvious next step would be to convert to HVC and then use directly the colour metrics defined above. However, the polar nature of the HVC coordinate system makes the calculations of colour difference, average colour and colour spread quite expensive. Since these are measurements that will be performed many times in low-level segmentation routines it is desirable to use an intermediate colour space for these purposes.

We define this intermediate space as a triple pqV where (p, q) is basically the cartesian equivalent of $[H, C]$. In terms of the CIELUV triple we define

$$p = 7.50725\, u^* \tag{3.8}$$

$$q = 7.50725\, v^* \tag{3.9}$$

$$V = L^* \tag{3.10}$$

From equations 2.11 and 2.13 we have

$$H = \tan^{-1}\frac{q}{p} - \psi \tag{3.11}$$

$$C = \sqrt{p^2 + q^2} \tag{3.12}$$

Since all colour distances are the same in pqV space as HVC space we can do our repetitive calculations using the pqV values. The only times we will need to convert to HVC values are when we need to refer to the colours themselves rather than just the differences between them or their spread.

Our colour difference equation now becomes

$$\Delta C = \sqrt{(p_1 - p_2)^2 + (q_1 - q_2)^2 + (V_1 - V_2)^2} \tag{3.13}$$

which is much more simple than equation 3.1.

Our average colour $(\hat{p}, \hat{q}, \hat{V})$ is now simply

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{3.14}$$

$$\hat{q} = \frac{1}{N}\sum_{i=1}^{N} q_i \tag{3.15}$$

$$\hat{V} = \frac{1}{N}\sum_{i=1}^{N} V_i \tag{3.16}$$

And our colour spread D is given by

$$D^2 = \frac{1}{N}(\sum p^2 + \sum q^2 + \sum V^2) - \frac{1}{N^2}[(\sum p)^2 + (\sum q)^2 + (\sum V)^2] \tag{3.17}$$

The advantage of equations 3.14 to 3.17 is that they express the average colour and colour spread only in terms of the number of colours N and the sums and sums of squares of p,q and V. This means that if we only store N and the six sums for each set of colours in which we are interested, then we can easily calculate the average colour and colour spread for a combined set of colours by simply adding each of the seven terms and inserting the values into equations 3.14 to 3.17.

Using this intermediate colour space we can significantly speed up our low-level segmentation routines and can then easily convert our colours to HVC values as we require.

3.2 Split and Merge Segmentation

Given our colour model and our colour metrics we can make an initial attempt at colour image segmentation using the standard split and merge (SAM) method (originally due to Horowitz and Pavlidis [49]) as defined in Ballard and Brown [5].

1. Start with the whole image as one region
2. If any region does not satisfy a particular *homogeneity function* then split the region into four quarters

3. If any pair of adjacent regions can satisfy the *homogeneity function* as a single region then merge those two regions
4. Repeat from step 2 until no more splits or merges can be performed

The key to this method is the choice of an appropriate *homogeneity function*. We have to apply the function to each pair of regions we are trying to merge and therefore it is desirable for the function to be defined in terms of region attributes which can be quickly calculated for the combined region.

To calculate the average colour of a region we require only the number of pixels and three sums; to calculate the colour spread we need a further three sums. Hence, by storing seven parameters for each region we can calculate the average colour and colour spread of any combination of regions using those parameters alone.

If we were dealing with a monochrome image then we could also determine the furthest value from the average by simply storing the maximum and minimum grey-levels included: for a combined region the new maximum is clearly the greatest of the old maxima and the minimum is the smallest of the minima. For a colour image though we are working in three dimensions so we would require the maximum distance in *all directions* for two regions in order to determine the furthest value from the average for the combined region. This is clearly not practical without making some drastic approximations so the only way to determine the furthest value would be to search the whole of both regions, this is equally impractical.

Therefore, in order to produce an algorithm which will work in reasonable time we must construct a homogeneity function in terms of colour spread alone. We thus define our homogeneity function $H(R_i)$ for region R_i as

$$H(R_i) = \begin{cases} true & \text{if } D(R_i) < D_{max} \\ false & \text{otherwise} \end{cases}$$

where $D(R_i)$ is the colour spread of region R_i (see equation 3.7) and D_{max} is a constant.

One of our conclusions in section 3.1 was that colours in HVC space are similar as long as they are separated by a distance of less than 20. To ensure that our region R_i is uniformly coloured using $H(R_i)$ alone we must find a suitable value of D_{max} which will give a high probability that all the colours lie within a distance of 10 from the average colour. Returning to table 3.1 we note that a set of colours uniformly distributed in a sphere around an average colour will have a spread of $0.77S$ ie. 7.7 so we might think this would be a reasonable value for D_{max}. However, it is obvious that deforming such a "colour sphere" (moving some colours towards the average and others away) could give the same spread but with many colours further than 10 from the average. It is preferable at this stage to over-segment rather than under-segment so the important consideration is therefore the density of colours at the extreme of the distribution, $\rho(S)$. From table 3.1 we can see that $\rho(S)$ becomes negligible for $D \approx 0.5S$ and thus we can be reasonably confident that using $D_{max} = 5$ will give perceptually uniform coloured regions.

3.2.1 Relaxing the Split and Merge

One problem with the standard split and merge is that the segmentation can be *blocky* ie. we get straight edged regions with sharp corners in gradually changing areas. This is clearly undesirable since these edges and corners are artifacts of the segmentation method and not properties of the image. The cause is the non-ideal nature of our compromise homogeneity function which pronounces regions as homogeneous when, in fact, small areas are clearly a different colour to the rest of the region ie. where we have a systematic error in colour rather than a random error.

In order to overcome, or at least reduce, the problem we can apply a relaxation technique to the initial segmentation. This situation is suited to a relaxation approach [54] since we have an initial estimate (the split and merge) and a means for iteratively updating the estimate (moving pixels from one region to another). For each iteration the relaxation technique looks at each pixel in turn, if the pixel is in region R_a and adjacent to region R_b and moving the pixel from R_a to R_b reduces the sum $D(R_a) + D(R_b)$ and maintains both $D(R_a) < D_{max}$ and $D(R_b) < D_{max}$ then the pixel

is moved. If there is more than one possible move then the one giving the lowest $\sum D$, across all regions involved, is performed. This boundary relaxation has the effect of *smoothing out* the undesirable artifacts.

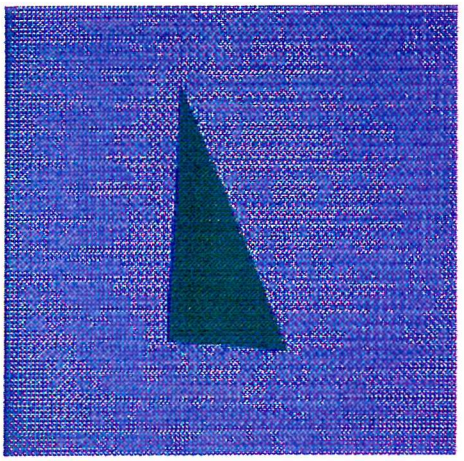
To allow boundaries to move equally well in all directions during the relaxation, pixels are examined left to right and top to bottom for odd numbered iterations and in the opposite direction for even numbered iterations.

A potential problem with the current implementation is that of hidden splitting. By this we mean the situation where a region gets thinner and thinner at some point until it breaks into two. If this were to happen successively to many regions then we might find that some ares of the image regressed back toward single pixel regions. In the results presented later we shall see that this does not happen and indeed has not happened with any of the images I have tested.

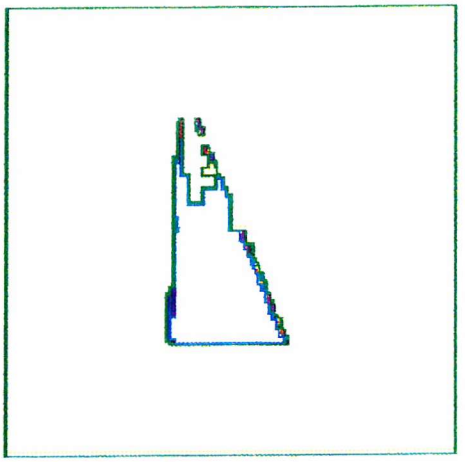
One further problem is how many iterations of the relaxation to perform. In some cases the segmentation will "settle down" quite quickly and no pixels will subsequently move, in other cases the segmentation may oscillate between two or more different states. Since we cannot determine quantitatively whether one segmentation is better than another, without knowing the *ideal*, it is very difficult to determine when to stop processing.

As an example, we can consider figure 3.3(a) which shows a single object on a plain background (256×256). The initial split and merge of this image (figure 3.3(b)) shows the tendency for boundaries to be horizontal or vertical as well as a large "missing" area at the top of the triangle. In some senses this a worst case because none of the true boundaries match the splitting lines (horizontal and vertical) and the object is big, but small enough to lose a large area to the background.

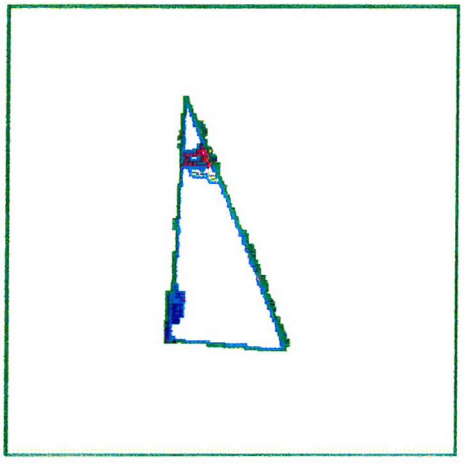
In figures 3.3(c)-(f) we can see the improvement to the initial segmentation after 2, 4, 6 and 8 iterations of the relaxation process. After just 2 iterations 3.3(c) the background has retreated into the correct place but some of the small noise regions have spread across the triangle near the top. Over the next few iterations the noise regions dissipate into the main triangle region, after 8 iterations the segmentation is stable and further



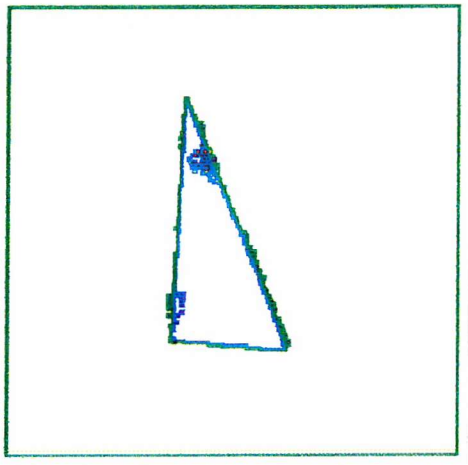
(a) Original Image



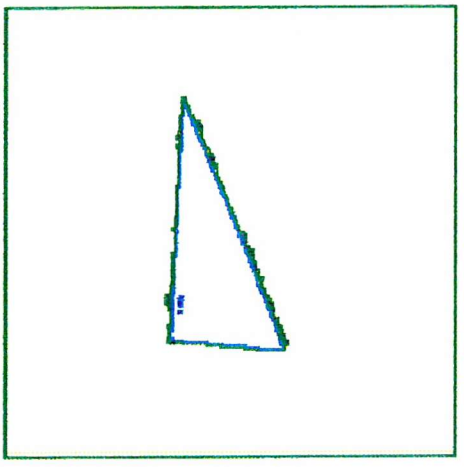
(b) Initial SAM



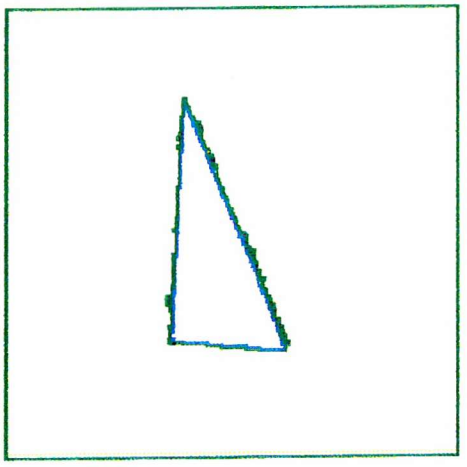
(c) 2 iterations



(d) 4 iterations



(e) 6 iterations



(f) 8 iterations

Figure 3.3: Relaxation Steps

iterations produce no change.

The example shows that when the objects are distinct the relaxation process produces improved boundaries very quickly and further iterations serve only to merge away noise. Where the objects are less distinct, as we shall see later, the boundaries tend to wander much more subtly as many regions "compete" for pixels. Obviously, if no pixels move then we stop relaxing immediately but the number of pixels moving, in general, is not a good indicator of the progress of the segmentation because of the complex swapping back and forth of pixels between regions. Experience has shown that performing a fixed number of iterations (20 in the results in section 3.4) gives significant improvement on the initial segmentation and that any further iterations will not generally provide any substantial changes.

3.3 Optimal Region Grower

As an alternative to the split and merge segmentation I have also implemented a colour region grower. Region growing in monochrome images has been widely used [46] but, as with the split and merge, we will attempt to improve upon the basic algorithm as well as extending the technique to colour images. The basic algorithm involves selecting a seed point and then extending the region by adding neighbouring pixels which are *similar enough*: when the region cannot be extended any further a new seed is selected and the procedure repeated until the whole image has been segmented.

In many cases the segmentation is not performed on a region-by-region basis but by scanning sequentially through the image, row by row. This scanning can produce a poor segmentation if many pixels, which just meet the similarity criteria, are joined early on and are thus prevented from later being added to regions in which they would be more suitable.

Clearly the major issues to be addressed when constructing a region grower are -

- How to select a seed point;
- How to go about adding pixels to the region;

- What do we mean by similar enough.

To produce a coherently coloured region it seems sensible to start with a coherently coloured seed point. Rather than using a single pixel as a seed we will use a 2×2 region (This precludes regions with areas less than four and lines which are only one pixel wide from our segmentation but this is not seen as a problem) which has the lowest colour spread, provided this is less than D_{max} (the value of which we will come to in a moment). From our seed region we will add the neighbouring pixel which has the nearest colour to the present average colour of the region, provided that the colour spread remains less than D_{max} and the colour of the pixel is less than 10 from the average colour. The region growing process, starting from a seed region R, is therefore defined as,

1. Find the set of pixels $P = \{p_1, p_2 \ldots, p_n\}$ which are 4-connected to region R;
2. If the average colour of the pixels in region R is $(\hat{h}, \hat{v}, \hat{c})$ and the colours of the pixels in P are $\{(h_1, v_1, c_1), (h_2, v_2, c_2), \ldots, (h_n, v_n, c_n)\}$ then determine the colour differences $\Delta C_i = \Delta C([h_i, v_i, c_i], [\hat{h}, \hat{v}, \hat{c}])$;
3. Find k such that $\Delta C_k = \min(\Delta C_i)$;
4. If $\Delta C_k < 10$ and $D(R + p_k) < D_{max}$ then add pixel p_k to the region R, update the average colour $(\hat{h}, \hat{v}, \hat{c})$ and return to step 1; otherwise terminate the growing process.

Since we are always starting at the most coherent point and extend outward along the "path of least resistance" we will get coherent regions as long as our similarity constraints are suitable. Unlike the split and merge we are not testing a set of pixels for similarity, we are starting with four similar pixels and then adding other similar pixels in order. As a result we can be much more confident in our average colour actually being a true representation of the whole region. By adding pixels which are most similar first we can also be confident that the average colour will not change very much as the region grows.

The increased confidence in the average colour might suggest that testing candidates for being within 10 of the average colour might be sufficient to ensure a coherent region.

Colour Model	black-white	spread (SAM)	spread (RG)	max distance
HVC	100.0	5.0	7.7	10.0
HSI	255.0	12.8	19.6	25.5
RGB	441.7	22.1	34.0	44.2

Table 3.2: Segmentation Parameters for Different Colour Models

However, there may still be cases where the average colour "wanders" and so we will continue to use the colour spread but in a less restrictive manner than in the split and merge: referring again to table 3.1 we see that an even distribution of colours ($\sigma \rightarrow \infty$) gives a spread of $0.77S$ or 7.7 and this is the value we shall use for D_{max}.

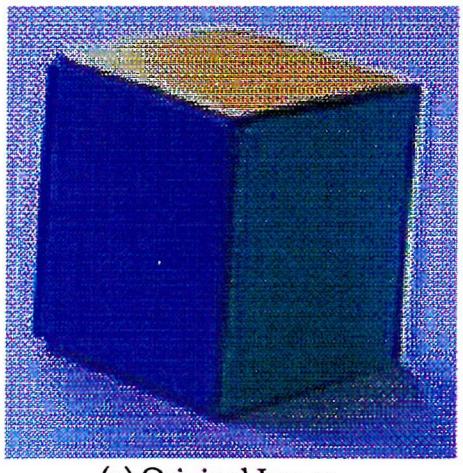
Clearly, we have approached the implementation of the colour region grower as a direct comparison with the split and merge, hence the choice of seed region as the most coherent 2×2 area and the repeated application until as much of the image has been segmented as possible. Since we have a region-by-region approach we can be more flexible if we wish, in both the selection of seed region and the number of regions we segment. These issues will be discussed in detail in later sections.

3.4 Comparison of Algorithms and Colour Models

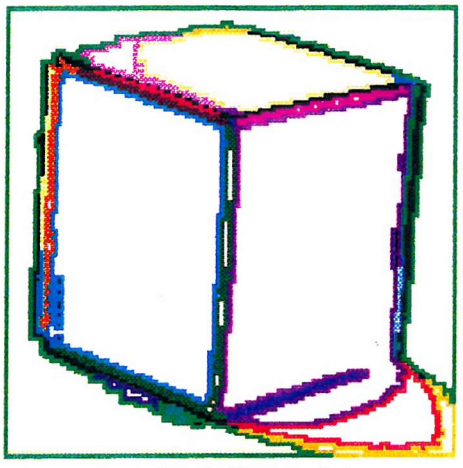
Both the region grower and split and merge algorithms have been implemented with three different colour models, RGB, HSI and HVC. For the RGB and HSI colour models we define colour metrics in the same way as we did for HVC in section 3.1 ie.

- Colour difference - Euclidean separation in the colour space;
- Average colour - the mean position in the colour space;
- Colour spread - the RMS distance from the average colour.

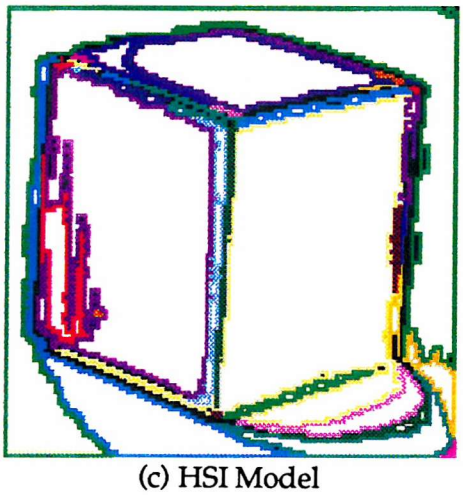
With regard to the parameters used, we will simply scale those used for the HVC space in proportion to the relative "sizes" of the other spaces. Since all the spaces are obviously different shapes, we will use the black-white distance as a measure of this "size". Using this approach we obtain the values in table 3.2 for our parameters.



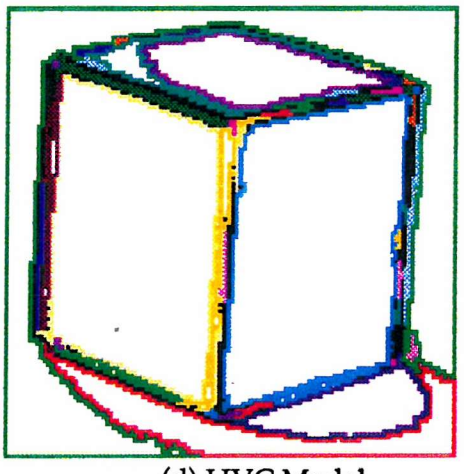
(a) Original Image



(b) RGB Model

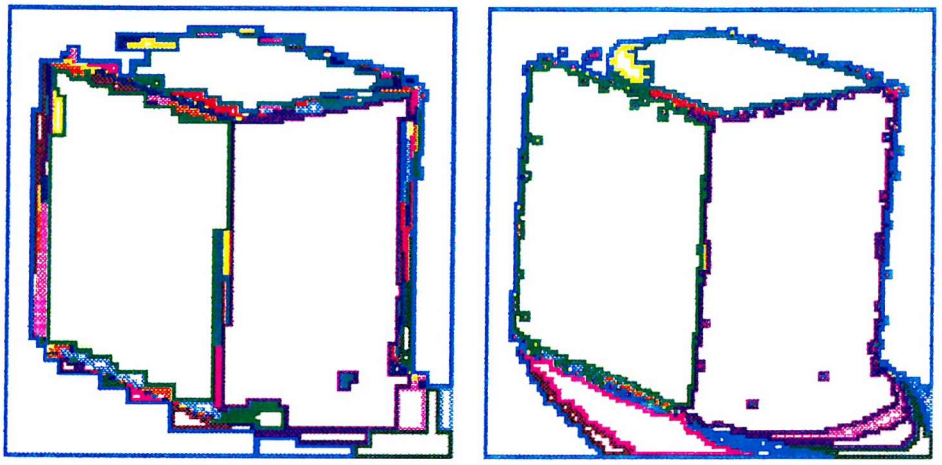


(c) HSI Model

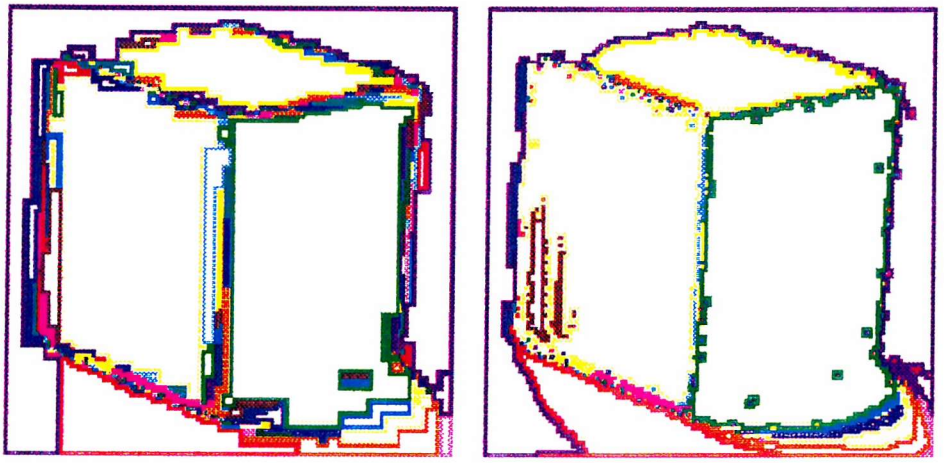


(d) HVC Model

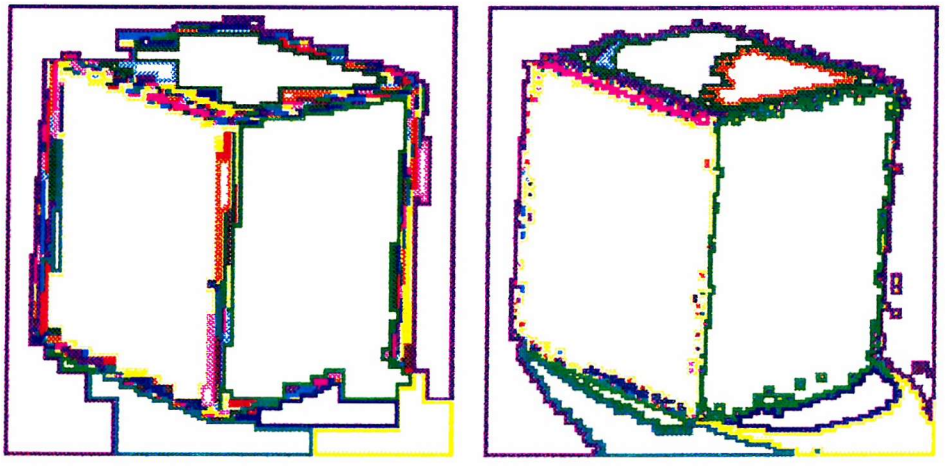
Figure 3.4: Region Growing using Different Colour Models



(a) RGB Model (b) RGB Model (20 iterations)



(c) HSI Model (d) HSI Model (20 iterations)



(e) HVC Model (f) HVC Model (20 iterations)

Figure 3.5: Split and Merge using Different Colour Models

3.4.1 Region Growing Results

Figure 3.4 shows the original image of a coloured cube (128×128) along with segmentation images [1] produced by the region growing algorithm. Of the three visible faces in the original image, the green and blue faces appear to be uniformly coloured but the yellow face has a distinctly brighter patch in the left-hand corner. There are clearly some distinct differences in the performance of the different colour models,

- Blue face - RGB and HVC almost extract a single region (with some small holes), HSI divides the face into two regions even though there is no visible difference in their colours;
- Yellow face - all three models separate the face into bright and normal patches with a small region in the far corner;
- Green face - RGB and HSI models have merged the green face and part of the shadow at the base of the cube (with HSI the gap is only 2 pixels wide), the HVC model has successfully separated the green face and shadow;
- Shadows - there are no easily determinable locations for the shadows since the background is gradually changing in these areas. The results produced by all the models are quite acceptable but the HVC model has produced the simplest result since the shadow is divided into only two regions, one clearly darker than the other.

Clearly the major fault with the RGB and HSI models is that they cannot separate the green face of the cube and the adjacent shadow. It is possible to "tweak" the parameters in order to achieve a separation but then we would find other regions divided where they should not be. These results therefore show that this segmentation algorithm is only suited to the HVC model since the other two spaces are not uniform enough to uphold the basic assumptions of colour similarity.

[1] In all the segmentation images presented, region boundaries are brightly coloured and region interiors are white: pixels which have not been labelled by the algorithm appear in black.

The algorithm itself (working with the HVC model as intended) performs well. All the distinct parts of the image have been segmented out much as expected.

3.4.2 Split and Merge Results

Using the same image as in the previous section we can see the results of the split and merge algorithm in figure 3.5. The left-hand column of results show the *initial* split and merge and the right-hand column shows the same segmentations after 20 iterations of the relaxation process. The profusion of straight boundary sections and sharp corners in the initial split and merge segmentations is obvious. Note also how some of the bright part of the yellow face has been merged with the background and some of the shadow merged with the green face with all of the colour models. Taking the initial segmentations alone we would have to say that the results with all three models were equally bad! The differences, and the potential of the algorithm, become apparent when we apply the relaxation process,

- Blue face - as with the region growing, the RGB and HVC models produce a reasonable segmentation on the blue face but the HSI model has problems at the bottom left edge;
- Yellow face - RGB and HSI models have failed to regain all of the bright patch from the background, the HSI model has merged most of the yellow region into one and the RGB model has produced a boundary between bright and normal which is arguably in the wrong place. Interestingly, the HVC model has separated the patch into two small regions but one of the noise regions (orange border) has gained pixels from the main region rather than being swallowed up;
- Green face - RGB and HSI models have again merged the green face with part of the shadow; the HVC model has separated the green face and the shadow by returning pixels from the initial "green face region" to the "shadow region";
- Shadows - again, the shadows are covered reasonably well by all three models after the relaxation process.

The relaxation process clearly improves the initial segmentations significantly. The handling of the yellow face using the HVC model is not ideal but, as stated previously, it is better to oversegment than undersegment since regions can always be combined at a later stage of processing.

The relative performance of the different colour models is much the same as with the region grower. The RGB and HSI models perform less well than the HVC model because in these spaces a constant colour spread encompasses a larger or smaller range of colours depending on the actual colours under consideration.

These results also show the major pitfalls of the split and merge algorithm. By using colour spread alone we have allowed some large regions to merge in small regions of a significantly different colour. In most cases though this deficiency is rectified by the relaxation process as it returns many of the wrongly placed pixels to their correct regions. Of course this can only happen if there is some part of the "correct region" to return the pixels to: if a large region were to completely swallow up a distinct region then there would be no boundary between the two "regions" across which to move pixels.

3.4.3 Region Growing vs Split and Merge

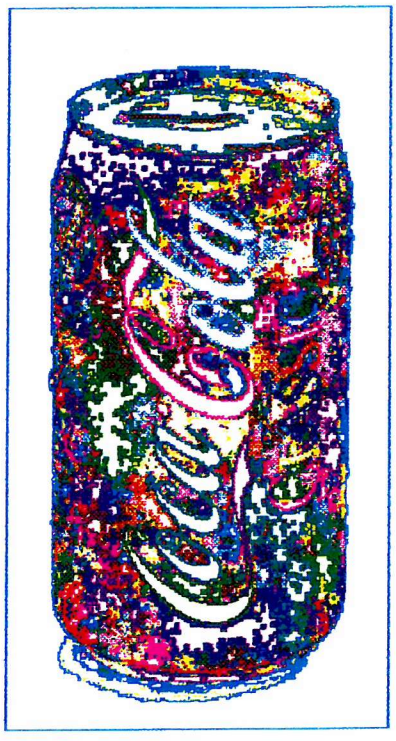
3.4.3.1 Segmentation Quality

In the previous two sections we have seen that both of the segmentation algorithms described perform best using the HVC colour model. Comparison of figures 3.4(d) and 3.5(f) suggests that the Region Grower is superior to the Split and Merge since all the faces of the cube are segmented out correctly.

Two, more varied, examples of the two algorithms' performance with the HVC colour model are given in figures 3.6 and 3.7. The drink-can image 3.6(a) (227×425) is a complex object (with some smoothly coloured and some textured areas) on a smooth background. The outdoor scene 3.7(a) (256×384) has few smoothly coloured areas and has poor contrast.

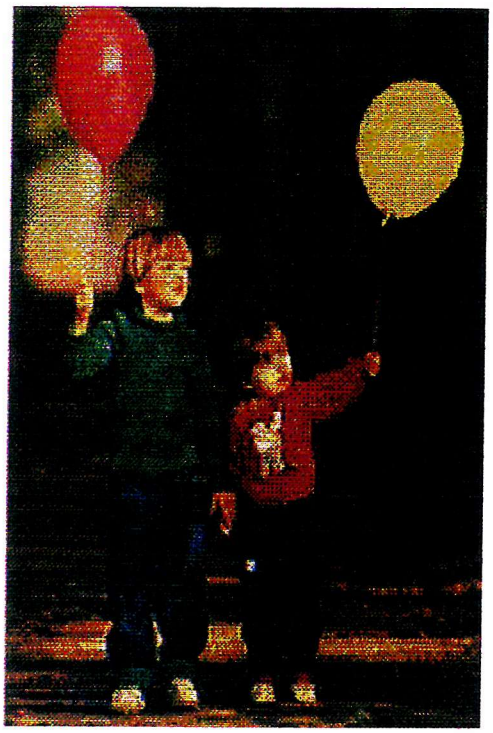
(a) Real Thing

(b) Region Grower



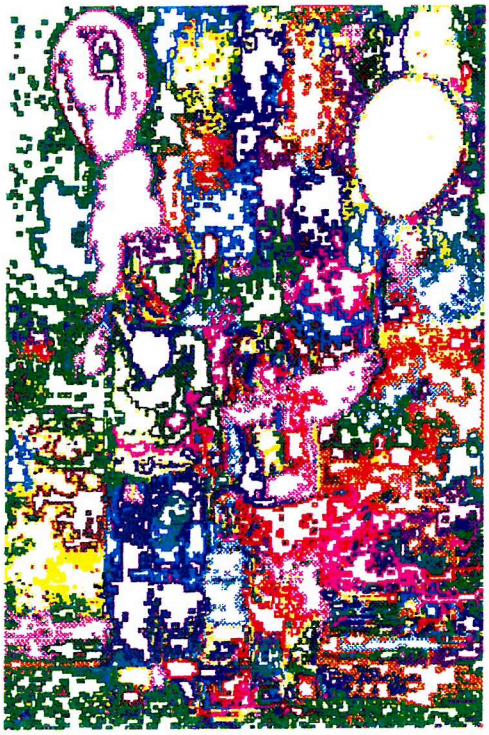
(c) Split and Merge

Figure 3.6: Region Grower vs Split and Merge



(a) Children and Balloons

(b) Region Grower



(c) Split and Merge

Figure 3.7: Region Grower vs Split and Merge

One obvious difference in the handling of the difficult regions is that the region grower tends to produce large regions which have lots of "holes" whereas the split and merge produces more compact, less perforated, regions. For our purposes, the former behaviour is more desirable since it is easier to trace the outline of a region (ignoring the holes) than to stick together many regions and trace the outline of the combined region.

In the highly textured areas (eg. the tiny writing on the can) the region grower gives no label (coloured black) but the split and merge produces a large number of single-pixel regions. Again, the former is more desirable since unlabelled areas can be subsequently examined by other techniques but single-pixel regions are of little use.

Other areas where the split and merge does badly are (i) the highlight on the yellow balloon - we may indeed wish to ignore this region but it is the job of higher-level parts of the system to do this, the region is a different colour from its surroundings and should be segmented out; (ii) some of the large letters on the can have been split - most notable is the capital "C" at the bottom which has been successfully segmented by the region grower but has lost its tail in the split and merge.

Based on the above observations we can say that, for our purposes, the colour region grower produces better overall results than the split and merge.

3.4.3.2 Flexibility

As well as producing preferable results for a "complete" segmentation, the region grower has the advantage of greater flexibility. Since the process is defined on a region-by-region basis we can be very selective in our choice of seed region and in the number/area of regions segmented, unlike the split and merge which is limited to segmenting the whole image in one go.

There are many approaches we might take to selecting a seed region, depending on our knowledge of the image content at the time eg.

- Position - it may be sensible, initially, to segment regions which are in the centre

of the image rather than at the extremes. Once some regions have been extracted we might want to find other regions in the same neighbourhood;

- Colour - at some stage of the segmentation we may decide to segment particularly light or dark regions, bright or pale regions or regions with a specific hue.

This flexibility is an important advantage since it enables us to get away from the restrictions of treating segmentation as an isolated process, enabling us to use partial segmentations in an intelligent and efficient manner.

3.5 Conclusions

Using segmentation algorithms which employ the concepts of colour difference, average colour and colour spread, we have seen that the perceptually uniform TekHVC colour space performs better than the non-uniform HSI and RGB colour spaces. This is due to the definitions of the colour metrics being consistent throughout the whole colour space. The performance of the HSI and RGB models could be improved by modifying the colour metrics (eg. Claxton and Kwok [31]) but this would be equivalent to transforming the colour space into a more uniform one, hence we might as well start with a uniform space and use the more simple metrics.

As previously stated by myself, and other researchers, there is no generally accepted procedure for comparing segmentation algorithms on the same platform, let alone via published results. Hence, it is very difficult to justify claims that one algorithm performs better than another, for some class of image, without testing both on the same image. However, general comparisons can be made on the assumptions which underly the algorithms; for instance, as indicated in the previous paragraph, we can assume that any algorithm which employs a colour metric will be improved if that metric is made more consistent.

In this chapter we focussed on the need to produce regions which were coherently coloured, even if this means that we are left with regions that contain many small holes. Algorithms which try to avoid holes often end up producing undesirable

results because they allow the merging of small regions with relatively large colour differences eg. the approach of Vlachos and Constantinides [104].

The region splitting method due to Ohlander et al. [76] was improved to some extent by Ohta et al. [77] and their $I_1 I_2' I_3'$ colour features. However, the drawback with clustering techniques remains that they do not necessarily produce coherently coloured regions because they are primarily concerned with detecting conspicuous peaks in histograms, which is often unreliable eg. for a region with a gradual change from one distinct colour to another we would expect to be able to divide it into two or more parts, but we will be unable to if the histograms are all "flat".

Meyer's region grower [71] produces good results if given a good enough set of marker points. Unfortunately there is no scope for partial segmentation since boundaries are only determined when regions "collide". Both of these problems are avoided by the region grower presented here since the marker points are determined automatically and regions only grow if there are pixels of a similar enough colour to be added.

The two algorithms presented in this chapter both improve on their more "standard" monochrome definitions as well as being extended to colour images. The split and merge algorithm is substantially improved by the addition of a boundary relaxation step and the region grower is made more flexible and consistent by using the region-by-region and *nearest colour first* approaches. The results given show that the region grower is most suited to our requirements and therefore will be used in the following chapters.

4 Plane Shape Analysis

In chapter 3 I described how a colour image could be segmented into a number of consistently coloured regions. Once we have a set of regions, we require a means of identifying their shapes in terms of previously stored (or learned) shapes. By tracing the boundary of a region we can obtain a closed 2D curve that is not self-intersecting. Hence, we require a means of characterising 2D plane shapes.

There have been several surveys of shape representation and matching. Notably by Pavlidis [80, 81] and later by Marshall [68] and Davis [113].

The prime objective of shape matching is to derive a measure of similarity between shapes, enabling *unknown* shapes to be compared to a set of knowns (either specific examples or class representatives). This is obviously heavily dependent on the shape representation employed.

Shape Matching can be achieved using Neural Networks which are *taught* shapes and can then recognise them. This is a subject in itself and will not be covered here but it is an important research area.

Whilst the concept of shape is easy to understand, it is very difficult to define and describe. As a result, there are a very large number of shape representation schemes in the literature. Pavlidis suggests several criteria for classifying different techniques

- Internal or External - internal methods examine the whole of the object but external methods only trace the boundary of the object.
- Scalar Transform or Space Domain - scalar transforms produce a set of scalar features from the object but space domain techniques produce another 2D picture. Scalar transforms are suited for input to statistical pattern recognisers

whereas space domain results are generally used in structural or syntactic pattern recognition.

- Information preserving and non-preserving - whether the original shape can be reconstructed (to arbitrary accuracy) from its description or not. If the representation is to be used for data compression then it must be information preserving, but if we are only interested in recognition then the representation need only hold enough information to distinguish the shape from similar shapes that may be encountered.

A further criterion, concerned with recognition, is whether the representation is scale, rotation and translation invariant ie. whether the representation of a shape remains the same irrespective of its size, orientation and position within an image. This invariance may be inherent within the representation but is more usually *designed in* using some form of normalisation (eg. Fourier Transform techniques) or even obtained by using a fusion of two or more techniques (eg. the work of Wu and Stark [111]).

4.1 Internal Scalar Transform Techniques

The best known of these methods is that of moments. The 2D central moments of an image function $f(x, y)$ are defined as

$$m(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)(x - \overline{x})^u (y - \overline{y})^v \, dx \, dy$$

These moments are position invariant but not rotation and scale invariant. Dudani, Breeding and McGhee [35] have used seven combinations of the second and third order moments for identifying aircraft. These *moment invariants* are position, rotation, scale and reflection invariant. Dudani et al. found that an additional set of moments could be calculated using just the boundary of the shape. These two pairs of seven invariants contain different information about the shape, the boundary giving finer detail information and the silhouette giving structural information. All fourteen invariants were then used as a feature vector in a statistical classifier.

Whilst the results of Dudani et al. are good, they only used six different types of aircraft. Also, the shape of an aircraft in a 256×180 image is not very complex since much of the fine detail is lost. In general, techniques of moments are not believed to perform well with more complicated shapes.

A novel description developed by Skliar and Loew [91] uses a diffusion-type procedure. Initially each boundary pixel is assigned an equal number of *particles;* then the diffusion of those particles into the region is modelled assuming isotropic diffusion (equal in all directions and parts of the region). If $N(i,j,t)$ is the number of particles at pixel (i,j) at time t then

$$\begin{aligned} N(i,j,t+1) &= N(i,j,t) - K.C(i,j)N(i,j,t) \\ &\quad +K.[N(i-1,j,t)+N(i+1,j,t)+N(i,j+1,t)+N(i,j-1,t)] \end{aligned}$$

where $C(i,j)$ is the number of 4-neighbours of pixel (i,j) within the region (ie. no particles diffuse in or out of the region) and K is the diffusion constant ($0 < K < 1$). The value of K determines the detail and computation time of the process and is set to 0.01 which gives a reasonable compromise.

The diffusion consists of an initial transient response followed by a steady- state condition. During the transient stage, the number of particles at each boundary pixel depends on the shape of the boundary: the concentration is higher in concavities than convexities. The variation in concentration around the boundary can thus be used as a shape signature. The time at which the signature is taken is when the largest difference between the maximum and minimum concentration on the boundary exists.

Few results are given and it is unclear as to whether the descriptor can be made invariant under rotation. The authors claim that the descriptor is not sensitive to noise and that it has potential for handling partial occlusion.

Even if this method cannot be used directly as a shape descriptor, it does seem to have a great deal of potential for finding the curvature maxima of a boundary and therefore may be of use in curve fitting or boundary segmentation.

4.2 External Scalar Transform Techniques

4.2.1 Fourier Transform Techniques

The FT of the shape boundary has been used by many authors. Zahn and Roskies [115] define their Fourier Descriptors (FDs) in terms of the fourier series of the cumulative angular bend versus arc length. For a polygonal curve of length L, with m vertices at $V_0 ... V_{m-1}$ and edge lengths $(V_{i-1}, V_i) = \Delta l_i$; if the change in angular direction at vertex V_i is $\Delta\phi_i$ then Zahn and Roskies define the cumulative angular change $\phi(l)$ at length l as

$$\phi(l) = \begin{cases} \sum_{i=1}^{k} \Delta\phi_i & for \quad \sum_{i=1}^{k} \Delta l_i \leq l < \sum_{i=1}^{k+1} \Delta l_i \\ 0 & for \quad 0 \leq l < \Delta l_1 \end{cases}$$

Then, the normalised form ϕ^* is defined as

$$\phi^*(t) = \phi(\frac{Lt}{2\pi}) + t$$

so that the domain of ϕ^* is $[0, 2\pi]$ and $\phi^*(0) = \phi^*(2\pi) = 0$. ϕ^* is then expanded as a Fourier series

$$\phi^*(t) = \mu_0 + \sum_{n=1}^{\infty}(a_n \cos nt + b_n \sin nt)$$

After lengthy calculation the coefficients are determined to be

$$\mu_0 = -\pi - \frac{1}{L}\sum_{k=1}^{m} l_k \Delta\phi_k \tag{4.1}$$

$$a_n = -\frac{1}{n\pi}\sum_{k=1}^{m} \Delta\phi_k \sin\frac{2\pi n l_k}{L} \tag{4.2}$$

$$b_n = \frac{1}{n\pi}\sum_{k=1}^{m} \Delta\phi_k \cos\frac{2\pi n l_k}{L} \tag{4.3}$$

where, $l_k = \sum_{i=1}^{k} \Delta l_i$.

The amplitude/phase form of the series is given by

$$\phi^*(t) = \mu_0 + \sum_{n=1}^{\infty} A_n \cos(nt - \alpha_n)$$

where (A_n, α_n) is the polar form of (a_n, b_n).

Zahn and Roskies show that the harmonics' amplitudes A_n are invariant under changes in size, orientation, position and starting point. The harmonic phase angles α_n are, however, variant under change in starting point. Whilst the amplitudes alone would be very useful as shape features, Zahn and Roskies define a form-invariant, based on the phase angles, unaffected by changes in starting point.

$$F_{kj} = \frac{j}{gcd(j,k)}\alpha_k - \frac{k}{gcd(j,k)}\alpha j$$

They then go on to show how a curve can be reconstructed from its descriptors. It is interesting to note that the reconstructed curve is generally not closed, since the original function has discontinuities. The number of harmonics used is ten in the examples given.

Whilst the FDs are good shape discriminators, in many cases, they are very expensive to calculate and using this method you have the additional problem of first fitting a polygonal approximation to the curve (a useful shape description in itself!). Since the digital image is already an approximation, the additional errors are compounded but the computational effort saved in reducing the number of vertices is substantial.

Persoon and Fu [86] define FDs in terms of the boundary as a complex function $u(l) = x(l) + jy(l)$ which has period L. Using the same notation as Zahn and Roskies, the complex FDs a_n for a polygonal curve are given by

$$a_n = \frac{L}{(2n\pi)^2}\sum_{k=1}^{m}\frac{V_k - V_{k-1}}{|V_k - V_{k-1}|}[\exp(-jn\frac{2\pi l_k}{L}) - \exp(-jn\frac{2\pi l_{k-1}}{L})]$$

These FDs are normalised for scale, rotation, position and starting point by making $a_0 = 0$ and multiplying the other a_n by $se^{j(\phi+n\alpha)}$. Where s,ϕ and α are chosen such that a_1 and a_{-1} become pure imaginary numbers and the modulus of their sum is unity. Alternatively, the FDs can be left as they are (except $a_0 = 0$) and normalisation can be built into the matching process by varying s,ϕ and α to produce the best match. This has the advantage that errors in a_1 and a_{-1} will not produce "faulty" normalisation

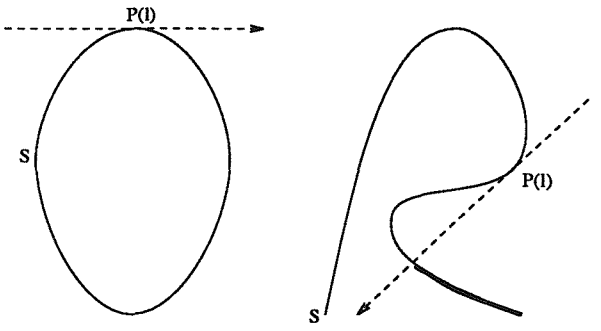


Figure 4.1: O'Rourke's Shape Signature

but the disadvantage that comparing FDs will take a lot longer.

Curves reproduced from this formulation of the FDs are always closed. This is an advantage over the previous formulation but with the added disadvantage of a much more complicated normalisation process.

In his review, Pavlidis [81] states that the advantage of FDs is that they are backed by the well developed theory of Fourier Transforms. The disadvantages are their inability to describe local information and to handle partially hidden objects.

Recent work by Arbter et al. [1] used normalisation in the Fourier domain to produce FDs which are constant under *any* affine transform. These FDs were then used to match aircraft silhouettes from 3D models. The technique was found to be more robust than ordinary FDs since 3D motion produces *nearly* affine transformations rather than similarity transformations (changes in position, orientation and size only).

4.2.2 Shape Signatures

In general, a shape signature is a 1D curve which is derived from a 2D boundary. Signatures are usually parametrised by a length or an angle and therefore can be considered to be continuous (unlike a sequence of FDs).

O'Rourke [78] proposed a shape signature which can be applied to both open and closed curves. Given some starting point S on a curve of length L (see figure 4.1) the value of the signature $\Sigma(s)$ corresponding to a point P (which is a distance l along the curve) is the fraction of the length of the curve which lies to the left of the directed

tangent at point P, where s is given by $s = l/L$. The length to the left of the tangent is shown in bold in the figure.

This representation is translation, rotation and scale invariant although care must be taken in defining the starting point and the direction of travel around the curve. The immediately obvious disadvantage of this representation is that all convex closed curves will have the same signature since $\Sigma(s) = 0, \forall s$.

O'Rourke's main application is the analysis of joined handwriting and his representation is useful in this domain, it does not apply well to general shapes though.

Wolfson [108] has used a translation and rotation invariant shape signature in order to find the longest common sub-curve between a model curve and an occluded image curve. From some arbitrary starting point, Wolfson first calculates a graph of arclength, s, versus total turning angle, $\theta(s)$ (called the cumulative angular bend by Zahn and Roskies in section 4.2.1). The graph is then sampled at n equally spaced points $s_1 \ldots s_n$ and the shape signature is defined as an averaged difference over these samples,

$$\phi_i = \frac{1}{k} \sum_{m=0}^{k-1} [\theta(s_i + m\delta + \Delta s) - \theta(s_i + m\delta)]$$

where $\Delta s, k, \delta$ are determined experimentally depending on the nature of the curve.

Using this signature Wolfson demonstrates algorithms for finding the longest common sequences between two signatures and hence matching models to occluded shapes in images. The signature itself could be made scale invariant by normalising the length but this would make the matching of common substrings very difficult so the method is best suited to applications where the scale is known eg. objects on a conveyor belt a known distance from the camera.

4.3 Internal Space Domain Techniques

The Medial Axis Transform (MAT) proposed by Blum [18] is the most widely used technique in this class. If S is a set of points in the plane and B is its boundary, then for

each point X in S, it is possible to find its closest neighbour in B: if there exists more than one such neighbour then X lies on the *medial axis* of S. The MAT can be used to derive shape information but its computation with discrete data can be difficult and time consuming and it is very sensitive to noise: a small change in the shape can produce a very large change in the MAT representation. It is usually necessary to fit a polygonal approximation to the curve first.

The MAT is an example of skeletonisation ie. reducing a 2D shape to an axial representation. Recently Wright [109] has proposed a method of skeletonisation based on the Euclidean distance transform. Each pixel within the shape is assigned a value which is equal to the distance from that point to the nearest point on the boundary. Extracting the skeleton is then a matter of finding local maxima in the distance map, a process which Wright compares to edge detection in grey-level images and demonstrates using the Marr-Hildreth operator and the Petrou operator.

What is not clear with all skeletonisation techniques is how skeletons can be matched efficiently. Difficult graph matching problems arise in which there is little extra information to constrain the matching process.

4.4 External Space Domain Techniques

Early examples of this class are the Freeman chain-code [42] and Davis's polygonal approximation [32]. These techniques, and related curve parsing methods, all suffer greatly from problems of scale in the definition of boundary curvature (see section 4.5).

A common approach is to first smooth the boundary and then try and segment it into sections by finding maxima and minima of curvature. This is the approach taken by Liu and Srinath [60] and Li, Ireton and Xydeas [59], both groups report that the number of segments depends on the amount of smoothing.

In cases where the boundary can be segmented reliably the breakpoints provide a very useful description of the shape. Because there are relatively few points it is possible to match shapes when some of the points are missing due to occlusion [58, 60]

or when one of the shapes undergoes an arbitrary affine transform [57].

4.4.1 Hough Transform Methods

Illingworth and Kittler [51] survey Hough techniques and state that their major advantage is the ability to recognise shapes in the presence of noise and occlusions.

The Generalised Hough Transform (GHT) [3] is an evidence gathering or voting procedure. A shape is described by a list of template vectors from the centroid of the template shape to its boundary. Translated instances of the shape can be characterised by the position of their centroid. Each image point is compared to every entry of the template list and votes for the corresponding centroid by incrementing a cell in a two-dimensional accumulator array. When all image points have been processed, the accumulator cells with the highest values give the most likely positions of the shape within the image. The method can also be used to search for scaled and rotated versions of the shape by adding a further two dimensions to the accumulator array.

Each image point generates or votes for all parameter instances that could have produced it. Only sets of image points which belong to a shape will vote coherently and produce peaks in the accumulator array. Thus extraneous data only adds to the distributed background votes and missing data leads only to a reduction in the height of a peak. The GHT is also inherently parallel since each image point votes independently. The main disadvantages of the GHT are the large computation and storage requirements.

The GHT is useful where a single object must be found since it is a model driven approach. When there is a large number of possible objects the amount of computation required is prohibitive since we must search the whole image for all possible instances of all possible objects.

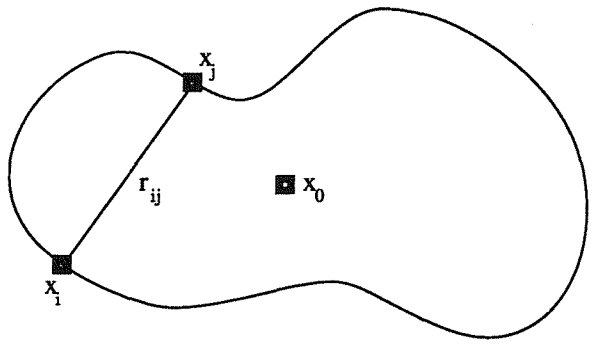


Figure 4.2: Chord Length Distribution

4.4.2 Probabilistic Techniques

Taylor and Cooper [96] have developed a shape representation scheme for naturally variable shapes. The chord length distribution (CLD) is defined in terms of probabilities rather than absolute measures and its application has some similarities to the GHT.

A shape is first defined in terms of a reference point x_0 and n boundary points $x_1 \ldots x_n$, the only constraint is that there must be a consistent way of finding the points. Given a set of points the CLD is then defined as the set of probability distributions $P(r_{ij}) : i, j = 0 \ldots n, i \neq j$ where r_{ij} is the distance between points x_i and x_j as shown in figure 4.2.

The probability distributions are estimated from a set of training images which display all the likely variations in the shape. At runtime the probabilities are used to iteratively update the hypothesis that the shape is present at a particular position. The examples given show that the system does converge quickly to the correct solution for simple shapes but the conditions for convergence are not understood.

4.5 Multi-Scale Techniques

The problems of scale have always been a source of difficulty in shape analysis. Witkin [107] states that any non-trivial local measurement has to depend on the value of the signal at two or more points, situated on some neighbourhood around the nominal

point of measurement. This means that to calculate say the curvature of a contour at a point, we have to select an arbitrary scale at which to perform the calculation. Different scales will, in general, yield different results and there is no single scale which can be said to be correct.

The proposal of Witkin is a *scale-space* which represents an image over a broad range of scales. The scale-space is obtained by convolving the image with a Gaussian, the standard deviation of which is used as the scale parameter.

The concept of a scale-space has been used by Mokhtarian and Mackworth [72] to describe and match planar curves. They first define the curve as $C = \{x(t), y(t)\}$ where t is the normalised path length $[0 \ldots 1]$ from some arbitrary starting point. The curvature $\kappa(t)$ is then given by

$$\kappa(t) = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{4.4}$$

In order to calculate the curvature at different scales, $x(t)$ and $y(t)$ are convolved with the Gaussian $g(t, \sigma)$

$$\begin{aligned} X(t,\sigma) &= x(t) \bigotimes g(t,\sigma) \\ &= \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} \exp^{-(t-u)^2/2\sigma^2} du \end{aligned}$$

and hence,

$$\begin{aligned} \dot{X}(t,\sigma) &= \frac{\partial X(t,\sigma)}{\partial t} \\ &= x(t) \bigotimes \left(\frac{\partial g(t,\sigma)}{\partial t} \right) \end{aligned}$$

$$\begin{aligned} \ddot{X}(t,\sigma) &= \frac{\partial^2 X(t,\sigma)}{\partial t^2} \\ &= x(t) \bigotimes \left(\frac{\partial^2 g(t,\sigma)}{\partial t^2} \right) \end{aligned}$$

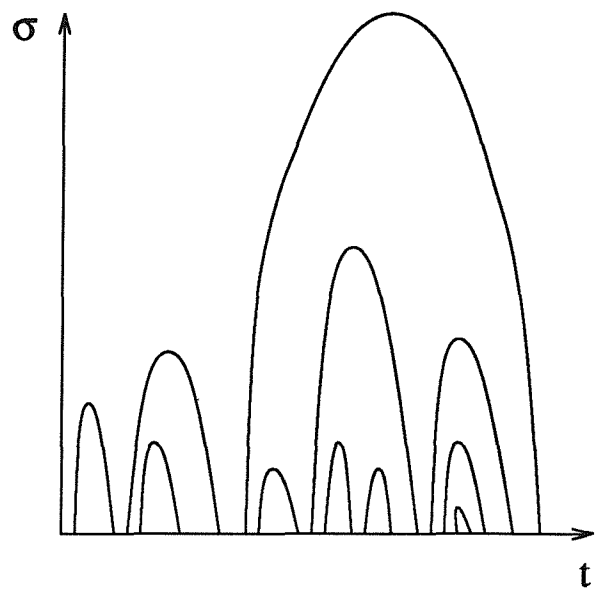


Figure 4.3: Generalised Scale-Space Image

and similarly for Y, $\dot{Y}$ and $\ddot{Y}$. Substituting $\dot{X}$ for $\dot{x}$ etc. in equation 4.4 gives

$$K(t,\sigma) = \frac{\dot{X}\ddot{Y} - \dot{Y}\ddot{X}}{(\dot{X}^2 + \dot{Y}^2)^{\frac{3}{2}}}$$

The zero-crossings of the function $K(t,\sigma)$ on the t-axis correspond to the positions of the points of inflexion of the curve at the scale σ. The generalised scale-space image is then defined as a binary image $I(t,\sigma)$ which is unity if (t,σ) is a zero-crossing of $K(t,\sigma)$ and zero otherwise. An example image $I(t,\sigma)$ is shown in figure 4.3. Clearly, this representation cannot distinguish between convex curves since they all have no points of inflexion at all.

Mokhtarian and Mackworth then go on to use the Uniform Cost Algorithm to match pairs of scale-space images. Using a VAX 11/780 with 4MBytes of memory the matching of 2 images with 4 or 5 contours takes about 20 seconds but a more realistic pair of 20 contours takes about 300 seconds. Hence the matching time is between $O(n^2)$ and $O(n^3)$. On top of this, the generation of the scale-space image is also computationally intensive, although certain assumptions can be made which reduce the computation time from hours to minutes!

Bengtsson and Eklundh [10] have used the ideas behind scale-space to produce a multi-scale contour approximation. They begin with a polygon $P_0 = \{(x_i, y_i) : i = 1, \ldots, N\}$ which has a vertex for every point on the digital boundary; then they calculate a series of polygons $P_1 \ldots P_S$ such that

$$P_0 \supseteq P_1 \supseteq \ldots \supseteq P_{S-1} \supseteq P_S \tag{4.5}$$

and P_k is a polygon which approximates P_0 to within a tolerance of $k\epsilon$. Hence, k can be thought of as the scale of the approximation and equation 4.5 means that as we increase the scale no new features (vertices) appear and small-scale features disappear before large-scale features.

Given a multi-scale polygonal approximation Bengtsson and Eklundh then classify all the points and segments of the polygons P_k as corners, points of inflexion, straight segments or segments with positive or negative curvature. Significant features are those which are stable over a wide range of scales so Bengtsson and Eklundh plot *occurrences of feature* vs k and look for plateaus. They then rank the approximations P_k according to their life-length (width of plateau) of various features.

The approach is demonstrated on test polygons and the most stable polygon is clearly a very intuitive fit to the original data. It is not clear whether the approach will be as successful with real edges from images where there are likely to be structures at many different salient scales.

4.6 Fractal Shape Description

Fractal Geometry [65, 66] has been used to model image functions as 3D fractal surfaces. The principle described by Pentland [84] is that fractal functions provide a good model for describing the 3D surfaces typical of natural scenes. Pentland's qualitative claim is that the imaging process produces an image surface which is also fractal. Given that the image (grey-level) surface contains areas which are fractal, authors have used the fractal dimension as a measure of texture [28, 70, 112]. Whilst the texture of a region is

useful for segmentation, it does not provide a means of shape description that is useful for matching.

A fractal related system that is useful for 2D shape representation is described by Giles et al. [44]. The Iterated Function System (IFS) is a recursive description of shape in which contractive affine transformations of the original shape are used as primitives in its decomposition. This means that primitives are not arbitrarily selected prior to encoding.

The reconstruction of the shape from the IFS is known as the *attractor* of the IFS. The IFS consists of a set of mappings w_n and an associated set of probabilities p_n. The attractor is then generated using the following algorithm:

1. Take an initial point in the plane x_0.
2. Choose a transformation from the IFS at random. The probability of choosing w_n is p_n.
3. Apply the transformation to get point x_1.
4. Choose another transformation to get from x_1 to x_2 and so on for a large number of points N.

The resultant set of points $(x_0, x_1, ..., x_N)$ is an approximation to the attractor of the IFS and tends to the attractor as N tends to infinity.

The method described by Giles et al. for determining the mappings w_n is as follows:

1. Given a list of boundary points, calculate the distance r of each point from the centroid and the arc length s from the start point.
2. For each point calculate $\frac{dr}{ds}$ and then smooth with a Gaussian with zero mean and standard deviation 1.7
3. Segment the boundary into arcs whose endpoints correspond to the zero crossings of the smoothed $\frac{dr}{ds}$ curve.
4. Classify arcs as linear, concave or convex by fitting quadratic functions in s using a least squares method.

5. Calculate contractive transformations to match arcs to the same type of arc. Consider only transformations with a contractivity of less than 0.8 in order to get *fast* convergence.
6. If the best fit for a given arc is better than a given threshold then accept the transformation and remove the arc from the segment queue. If the fit is not good enough then halve the arc and add the new segments to the end of the segment queue. Continue until the queue is empty.

By choosing the centroid of the shape as the origin of the coordinate system, it is very likely that the point $(0, 0)$ will lie on the attractor so this is used as the starting point x_0.

The probabilities p_n are determined by restricting the choice of transformations to affine transformations for which the determinant of the transform matrix is equal to the ratio of the area of the transformed region to the area of the original. Choosing probabilities to be proportional to the determinant of the transform matrix leads to an even spread of points over the attractor.

The results given by Giles et al. are not very impressive but do show that a 2D shape can be coded as an IFS. Since the IFS codes generated are not scale or orientation invariant, the present scheme cannot be used for shape matching. In addition, the code generation is rather crude and slow and occasionally produces regenerated boundaries that have large *gaps*. These problems are currently being investigated by the authors.

IFS have been used for image compression by Barnsley [8]. Barnsley has not fully published his algorithms (for commercial reasons) and his early claims were treated with suspicion by many people in the field. The hardware produced by Barnsley's company has since demonstrated that large compression ratios can be achieved using IFS techniques. The quality of the reconstructed images suggests that the heart of the system consists of a very good algorithm for finding the optimal IFS codes for general images.

Scott [89] states that IFS are a *special case* of the search for natural basis functions and transformation between and within shapes. The general parametric form used by

Scott to describe closed continuous 2D curves is

$$\begin{aligned} x &= a_0 G_0(\theta) + a_1 G_1(\theta) + a_2 G_2(\theta) + \cdots \\ y &= b_0 G_0(\theta) + b_1 G_1(\theta) + b_2 G_2(\theta) + \cdots \end{aligned}$$

where θ is a continuous parameter and the number of terms is finite. Without loss of generality the functions G_i may be required to be orthogonal over θ.

Given a set of training shapes, Scott derives a set of *natural basis functions* which can expand them all. It is assumed that the shapes are affine sets and that the correspondence problem can be solved. For such a case with N curves there are N vectors **a** for which $\boldsymbol{a}.\boldsymbol{p} = \mathbf{0}$, where **p** is the vector $(1, x_1, y_1, x_2, y_2, \ldots, x_N, y_N)$. The values of **a** may be generated by determining the eigenvalues and eigenvectors of the matrix $\sum \boldsymbol{p}\boldsymbol{p}^T$ where the sum is over sets of corresponding points. Scott calls the values associated with zero eigenvalues, *invariant vectors* and those associated with non-zero eigenvalues *basis vectors*. For these basis vectors the dot product $\boldsymbol{a}.\boldsymbol{p}$ is generally non-zero and varies from one set of corresponding points to another. These are the values that Scott uses for $G_0(\theta), G_1(\theta), \ldots$ in descending order of eigenvalues.

Unfortunately Scott, like Giles et al. above, only gives results of reconstruction. There is no indication as to whether the description is useful for shape matching.

4.7 Conclusions

In this chapter we have seen many different approaches to the description and matching of 2D shapes. What all these techniques attempt to produce is a consistent and readily computable representation of shape. Unfortunately, it is often the case that the most consistent representations are the least readily computable and/or applicable to matching.

Since the basic shape of a region is independent of its position and size, it is desirable for a shape representation to be translation and scale invariant. In most applications we also require a rotation invariant descriptor, an exception would be character recognition

where it is important to be able to distinguish an "M" from a "W" and a "6" from a "9". In other applications we may require further invariances, such as reflection invariance for recognising mirror images of shapes or affine invariance for recognising shapes that have undergone general linear transformations.

External techniques (those which examine the boundary only) are more common than Internal techniques (those which consider the whole area) because in most applications the shape is characterised completely by its boundary and any internal detail is either irrelevant or misleading. Character recognition again provides a counter-example, it is much easier to distinguish a "0" from an "8" using internal techniques. It is interesting to note though that Dudani et al. [35] chose to use a combination of internal and external moments because the internal moments alone could only capture the large scale structure of the shape and not its detail. Internal Space Domain techniques such as skeletonisation [18, 109] are quite appealing in theory but their implementation for real shapes in digital images reveals many problems due to their excessive sensitivity to noise.

Fourier Transform techniques have received a lot of attention as an extension of the large body of knowledge which has been built up from analysing 1D signals. Indeed, the approach of Zahn and Roskies [115] is to first describe a boundary as a 1D signal (the cumulative angular bend function) and then determine the FDs for that. Problems arise with boundaries which have corners because this produces discontinuities in the bend function which are not handled well by Fourier Theory. Perhaps a more intuitive strategy is that used by Persoon and Fu [86] where the boundary is modelled as a complex function (therefore 2D) rather than a real function. Although this produces FDs which better describe the original 2D shape they are unfortunately much more difficult to normalise for scale, rotation and starting point.

One criticism of many shape representations is that they are global ie. they describe the shape as a whole. Global representations are not useful if we need to identify shapes which may be partially obscured since the representation of the original shape and the obscured shape will generally be completely different.

Hough Transform type techniques overcome (or at least greatly reduce) the occlusion problem by effectively using all the boundary points independently. Whilst Hough techniques often produce good results there are serious drawbacks in the large amounts of memory required for the accumulator array and the large amount of processing required to calculate the "votes" and find the peaks.

A common approach to local shape description has been to segment the boundary into sections which can be described separately. The idea is that if some of the sections are obscured then the shape may still be identified from the remaining sections. Whilst this is a good idea in principle, all these techniques suffer from scale problems in the definition of curvature. Problems with curvature arise because the boundary must be segmented in a consistent manner, invariably this leads to attempts to find points which are minima/maxima/inflexions of the curvature function. Since the number and position of such points varies with scale then the selection of an arbitrary scale produces an inconsistent representation.

Witkin's scale-space [107] goes a long way to producing a consistent shape representation by capturing curvature information across *all* scales. As used by Mokhtarian and Mackworth [72] the generalised scale-space image is of limited use because it only tracks inflexions in the curvature function rather than maxima and minima. This is presumably because it is much easier to find zero-crossings than maxima and minima. Even given this simplification, scale-space representations are very expensive to calculate and match.

Finally, the approach with perhaps the most potential for consistent shape description is that of Fractal Geometry. The basic principles of fractals embody the need to describe shape at all scales and yet yield expressions which are inherently simple. Much work to date has concentrated on using fractals to describe the shape of 3D surfaces as a measure of texture or, on the 2D front, IFS and Scott's natural basis functions [89] have largely sought to represent shape for compression and reconstruction rather than matching. What is required is a fractal-based 2D shape description which can be used for matching and this is discussed in the next chapter.

5 The Fractal Shape Signature

As we saw in chapter 4, there are many 2D shape analysis methods in the literature. In this chapter I will describe a novel method of shape analysis based on the principles of Fractal Geometry. The preliminary results of this work were presented at the *British Machine Vision Conference 1991* [98].

In the following sections we will see how the *Fractal Shape Signature* is constructed by modelling a shape's boundary as a fractional Brownian function (FBF) [65, 66]. We will then go on to examine the properties of the signature and demonstrate its application to 2D shape matching.

5.1 Fractional Brownian Model

The science of fractals was developed by Mandelbrot [65, 66] to enable complex natural shapes such as mountains, trees and clouds to be described mathematically. Although the shapes are very complicated, Mandelbrot found that they could be generated from very simple functions. The important properties of fractals, which enable them to be modelled so simply, are self-similarity and statistical invariance over wide ranges of scale.

One example of a physical system which exhibits fractal behaviour is Brownian motion, the apparently random motion of a particle bombarded by other particles. Whilst the motion is random (in that it cannot be predicted) it is statistically invariant over wide ranges of scale. This invariance is modelled using a fractional Brownian function (FBF).

A function $f(\boldsymbol{x})$ is an FBF if for all vectors $\boldsymbol{x}$ and $\Delta\boldsymbol{x}$,

$$Pr\left(\frac{f(\boldsymbol{x}+\Delta\boldsymbol{x})-f(\boldsymbol{x})}{\|\Delta\boldsymbol{x}\|^H} < t\right) = F(t) \tag{5.1}$$

where $F(t)$ is a cumulative distribution function [65, 66] and H is a constant. The function f is some measurement of the system and $\boldsymbol{x}$ is some parameter such as time or position, the term $\Delta\boldsymbol{x}$ is the scale over which the change in the measure is being considered. What equation 5.1 says is that the change in the measure divided by the scale at which the measure is made (to the power H) has the same probability distribution for *all* scales.

Clearly, the constant H is an important property of the system we are modelling. Mandelbrot introduced the concept of a *fractional dimension* [1] as an extension of the notion of integral dimension to allow the consistent measurement of features over different scales. The constant H makes our measurement in equation 5.1 consistent over different scales and is therefore related to the fractional dimension of the system.

If the vector $\boldsymbol{x}$ represents a point in E-dimensional Euclidean space $\Re^E$ then the constant H is related to the fractional dimension D of the system by, $D = E + 1 - H$. Hence, if $\boldsymbol{x}$ is a scalar then $D = 2 - H$.

From equation 5.1 it follows that for a given scale Δx,

$$E[\,|f(x+\Delta x)-f(x)|\,]\cdot|\Delta x|^{-H} = K \tag{5.2}$$

where $E[z]$ is the expected value of z and the constant K is the expected value of the modulus of the random variable t eg. for a zero-mean Gaussian distribution $N(0,\sigma^2)$ where,

$$F(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}\sigma} \exp\frac{-s^2}{2\sigma^2} ds$$

then,

$$K = E[\,|t|\,]$$

[1] Often known as the fractal dimension.

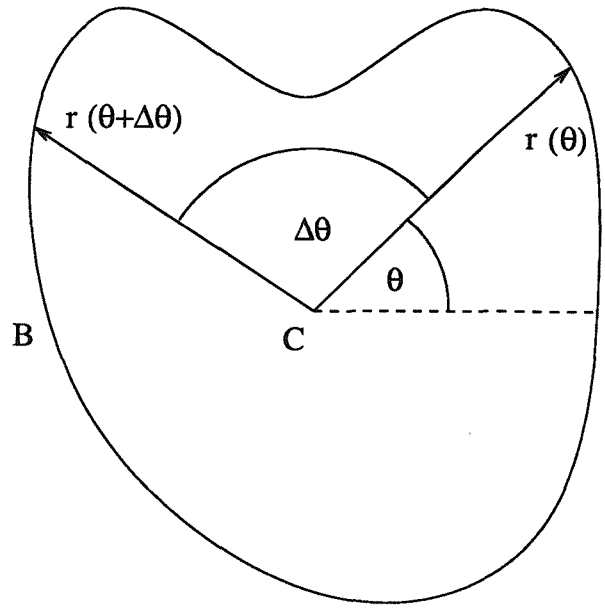


Figure 5.1: Continuous Plane Shape

$$= \int_{-\infty}^{\infty} |t| \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-t^2}{2\sigma^2} dt$$
$$= \sqrt{\frac{2}{\pi}} \cdot \sigma$$

We can now see how we might model a 2D shape boundary as an FBF. If we consider the boundary B in figure 5.1 then we can define a function $\epsilon(B, \Delta\theta)$ as

$$\epsilon(B, \Delta\theta) = E[\ |r(\theta + \Delta\theta) - r(\theta)|\ ]$$

where $E[\ |r(\theta + \Delta\theta) - r(\theta)|\ ]$ is the expected value of the difference in radius (from centroid C) for any two points separated by an angle $\Delta\theta$. Note that $r(\theta)$ may be multi-valued. Substituting into equation 5.2 gives,

$$\epsilon(B, \Delta\theta).\Delta\theta^{-H} = K \tag{5.3}$$

and equation 5.3 is equivalent to

$$\log \epsilon(B, \Delta\theta) - H \log \Delta\theta = \log K \tag{5.4}$$

For a true FBF, if we plot $\log \epsilon(B, \Delta\theta)$ against $\log \Delta\theta$ then (from equation 5.4) we expect to get a straight line with a gradient H and intercept $\log K$. However, the $\log - \log$ graph for any shape is a useful measure of the shape since we can think of our shape as being represented by different FBFs for different ranges of scales.

Other researchers (eg. [84, 112]) have tried to characterise shape by extracting a single fractal dimension D which is representative over a large range of scales. In the real world, some shapes will fit into this scheme better than others. To some extent a deviation from the model tells us as much about the shape as an agreement with the model. Therefore, I propose that the whole $\log - \log$ graph can be used as a *shape signature* ie. a sequence of ϵ values are used to represent the shape rather than the gradient of any one section. In the following sections we will see how this idea can be applied to the description and matching of plane shapes in images.

5.2 Shapes in Digital Images

The theory of the preceding section was derived in terms of a continuous boundary. When a region is extracted from an image we get a boundary that is made up of a list of discrete points. If we assume that the boundary is straight between points (see figure 5.2) then we can scan the list and generate, by interpolating where necessary, a list of polar coordinates $(r_i, n_i\theta_{min})$ $i = 1, 2, \ldots, N$ where n_i is an integer and θ_{min} is a constant. The value of N will be $2\pi/\theta_{min}$ if the boundary function is *single valued* but greater if it is *multi-valued*. Note that for a fractal boundary it is strictly invalid to interpolate linearly between points: however, since our boundary is the border of a segmented region, the interpolation is simply a re-sampling of the observed data.

We can then calculate $\epsilon(B, \Delta\theta)$ for integral multiples k of θ_{min} by scanning the polar list. As we are comparing all pairs of points this takes a time of $O(N^2)$.

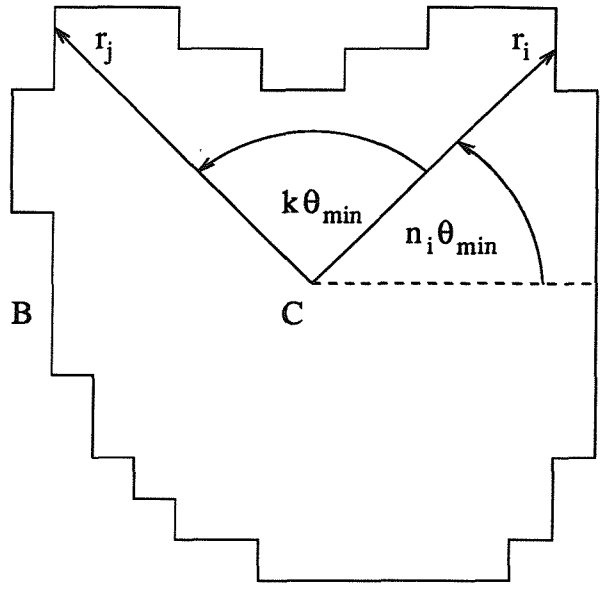


Figure 5.2: Digital Plane Shape

$$\epsilon(B, k\theta_{min}) = \frac{\sum_{i,j} P(i,j)|r_i - r_j|}{\sum_{i,j} P(i,j)}$$

where,

$$P(i,j) = \begin{cases} 1 & \text{if } n_i - n_j = k \\ 0 & \text{otherwise} \end{cases}$$

Clearly $\epsilon(B, \Delta\theta)$ will have period 2π and $\epsilon(B, \Delta\theta) = \epsilon(B, \pi - \Delta\theta)$ so we only need to look at values of $\Delta\theta$ between 0 and π to get a "complete" signature.

Figure 5.3 shows the signatures of three simple shapes obtained using $\theta_{min} = \pi/180$ and $k = 1 \ldots 180$ ie. we have a sequence of 180 ϵ values. These three curves are clearly distinct and yet the longest straight section for each of them has very nearly the same slope. We can therefore see that using the whole of the log – log graph gives us a better characterisation of the shape than the gradient of any one part.

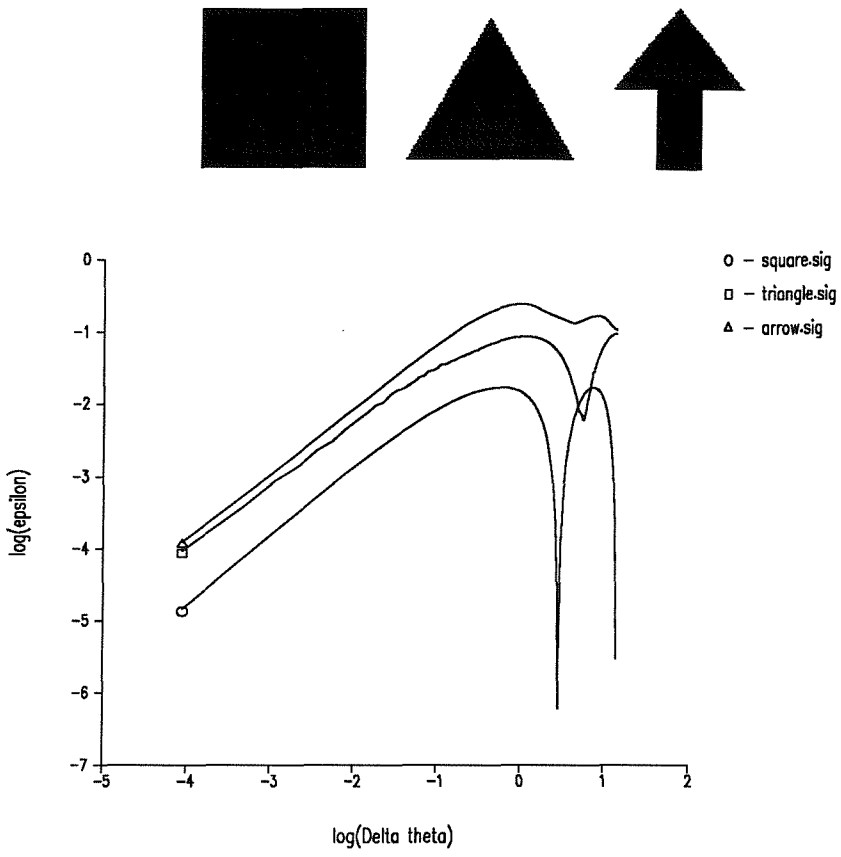


Figure 5.3: Simple Shapes and their Signatures

5.3 Invariance of Fractal Signatures

As discussed in chapter 4 we require our shape measure to be invariant under certain simple transformations of the shape. In particular we require invariance under translation, rotation and scale ie. we want a shape to have the same signature irrespective of its position and orientation in the image as well as its size.

Since $\epsilon(B, \Delta\theta)$ is measured relative to the centroid of the shape, the fractal signature is invariant under translation. The measure used involves the difference of radii so the signature is both rotation and reflection invariant.

Scaling the shape would result in a similar scaling of the ϵ values and thus a constant offset on the log graph. In order to overcome this, the signature can be *normalised* by dividing by the mean radius ie.

$$\epsilon'(B, \Delta\theta) = \frac{E[\ |r(\theta + \Delta\theta) - r(\theta)|\ ]}{E[r(\theta)]} \tag{5.5}$$

This *normalisation* also cures the problem that the signatures of $r_1 = f(\theta)$ and $r_2 = f(\theta) + constant$ would have been the same.

From here on, when we refer to the fractal signature of a shape we will mean the normalised signature defined above in equation 5.5.

5.3.1 Effects of Perspective Transformation

If we are viewing a 2D plane shape from a point which can move in 3D then we would like to know how the signature is affected by changes in that viewpoint. To examine this we will consider a viewing scheme such as that shown in figure 5.4. The point **O** is the centroid of the shape and the "lens" is at position **L** so we will call the distance OL the viewing distance. The viewing plane has its origin at point **P** and is perpendicular to the line **OL**, the focal length of the lens is therefore the distance LP. The figure shows an overhead view and a general viewing position which is identified by the viewing distance OL' and the two angles α and β. Altering the focal length of the lens simply scales the projected shape so this is fixed at unity ($L'P' = 1$).

To illustrate the effects of changes in viewing position we will use a unit square. Figures 5.5 and 5.6 show how the signature of the square changes when viewed from a position ($\alpha = 45°, \beta = 0$) at distances of 1 to 1024. There is a gradual change in signature as the viewing distance increases from 1-8 but then there is very little change as the viewing distance increases further. Therefore, if we discount "very close" views we can assume that viewing distance will have no effect on the signature of the viewed shape and we will be effectively considering orthographic projection. Additionally, from a far viewpoint any changes in the signature due to translations of the viewpoint will also be negligible. This "distant viewing constraint" is a common assumption since it is unlikely that the camera will be very close up to an object.

Using the same square we can see the effects of varying the viewing angles α

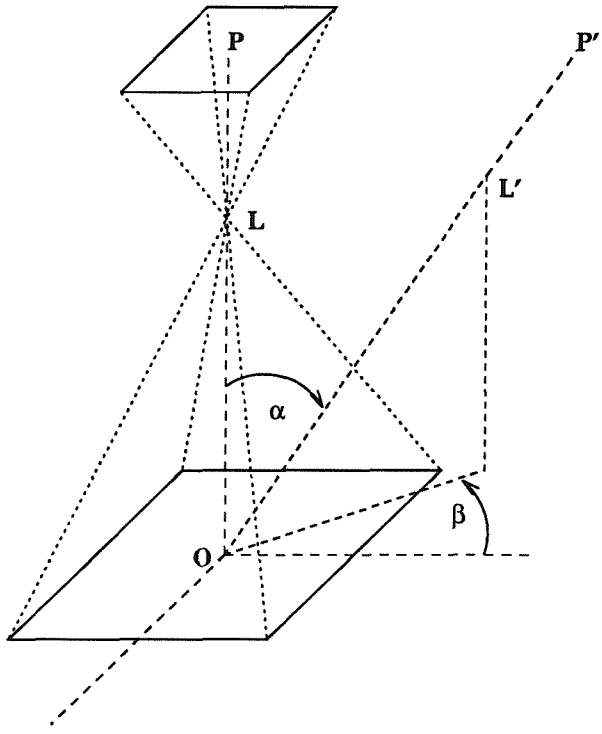


Figure 5.4: Perspective Transformation

(figure 5.7) and β (figure 5.8) with a constant viewing distance of 1000.

If we think of our original signature as a point in a 180-dimensional hyper-space then we can think of changing the viewpoint as moving the signature-point along a hyper-surface [2]. We can see that this surface is smooth because the signature changes gradually as the viewpoint moves.

Ideally we would like to be able to calculate the hyper-surface from the original signature and the viewpoint. This would allow us to decide whether an unknown signature was due to viewing a known shape from any arbitrary viewpoint. As we will see later, such calculations are extremely difficult, if not impossible, to perform.

As an alternative, I propose a type of *property sphere* scheme (see chapter 6,section 6.2) for encapsulating the viewpoint variability of the signature. We can view each known shape from a grid of equally spaced points on the viewing hemisphere and calculate

[2]If we allowed different viewing distances then we would get a volume, effectively the hyper-surface would have some "thickness".

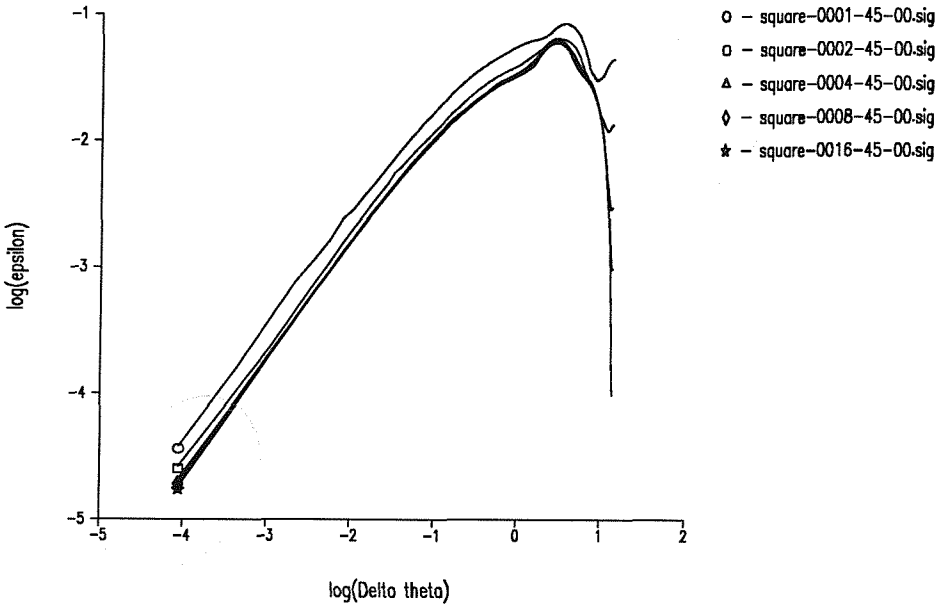


Figure 5.5: Effect of Changing Viewing Distance 1-16

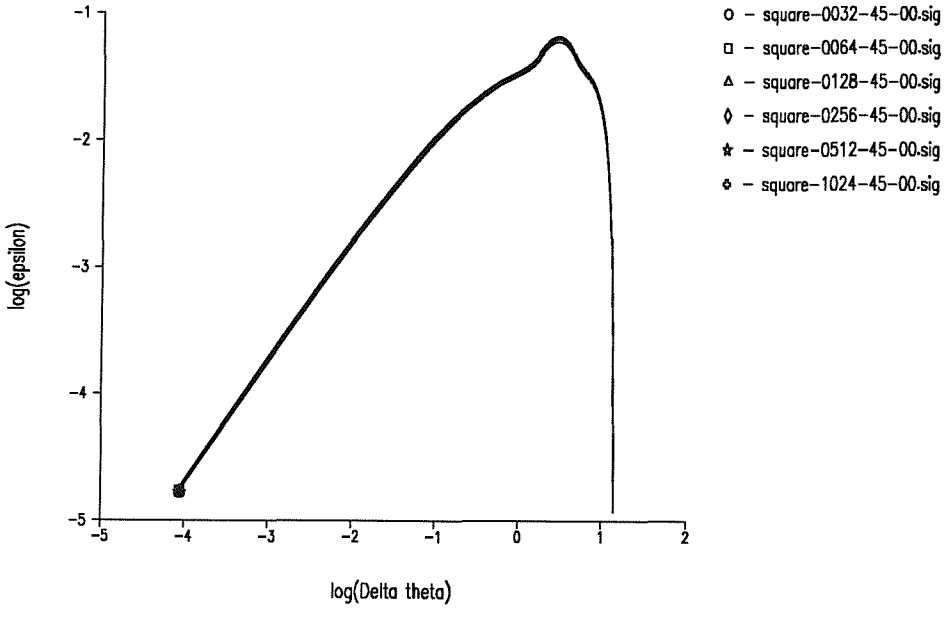


Figure 5.6: Effect of Changing Viewing Distance 32-1024

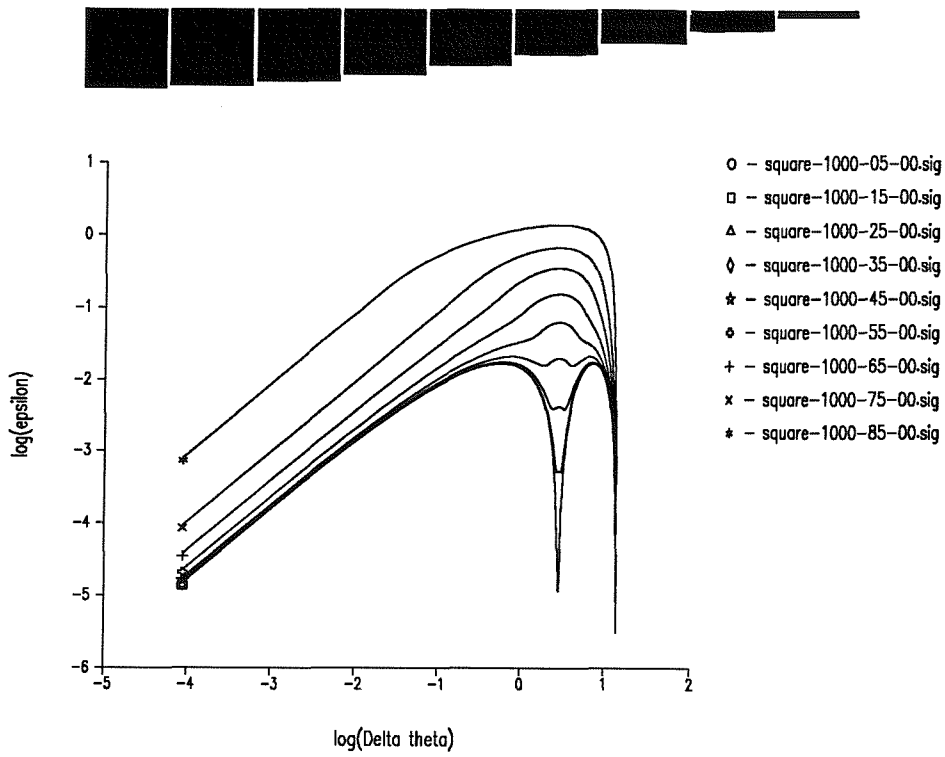


Figure 5.7: Effect of Changing Viewing Angle α

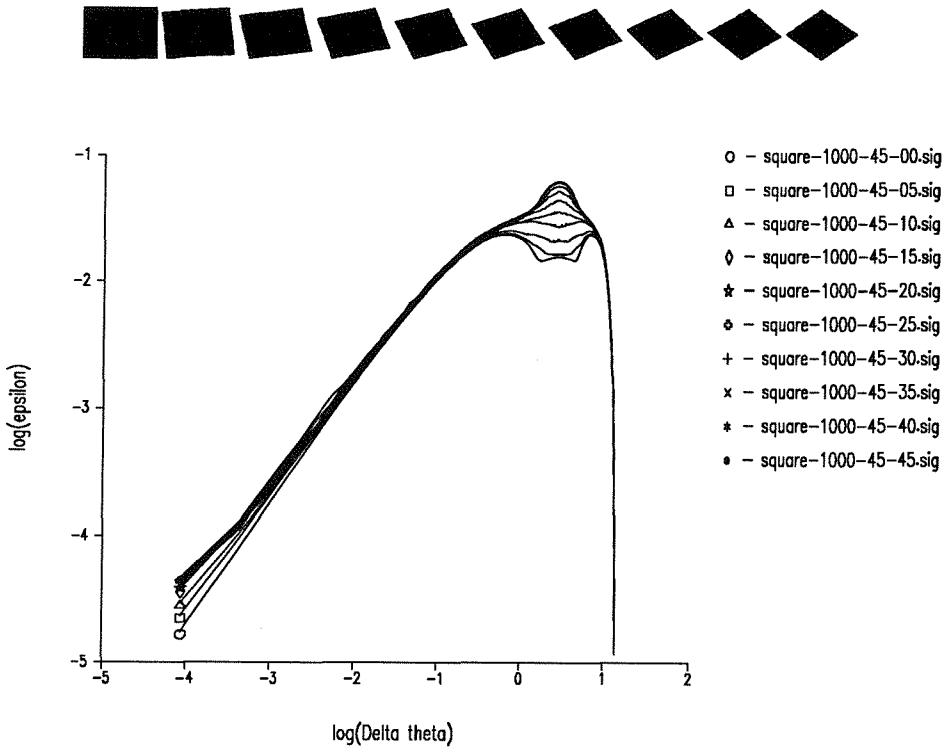


Figure 5.8: Effect of Changing Viewing Angle β

its signature, labelling it with the viewpoint. Given some unknown signature we can search the stored signatures for the closest match (we will discuss how signatures can be matched in the following section). Since the signature hyper-surface is smooth we can be confident that we are in the right part of the hemisphere as long as the sampling is fine enough.

Given an approximate match we can "zoom in" on the correct viewpoint by projecting the stored shape from a small neighbourhood around the current estimate of the viewpoint at a finer resolution. For these new projections we can compare their signatures to the unknown and get a new estimate of the viewpoint. This process can be repeated recursively until we are satisfied that the unknown shape is or is not the stored shape.

5.4 Matching Fractal Signatures

Unknown shapes can be matched against a library of known shapes by comparing fractal signatures. Whilst the comparison of signatures is not trivial it should be much simpler than direct comparison of the boundaries since we have removed the effects of scaling, rotation, reflection and translation.

I have used two different methods for comparing signatures

1. Euclidean Distance - treating the signatures as 180-dimensional vectors and calculating their separation.

$$E = \sqrt{\sum_{k=1}^{180} [\log \epsilon'(B_1, k\theta_{min}) - \log \epsilon'(B_2, k\theta_{min})]^2}$$

2. Bounded Area - treating the signatures as continuous curves and calculating the (absolute) area between them.

$$A = \int_{\log \theta_{min}}^{\pi} |\log \epsilon'(B_1, \theta) - \log \epsilon'(B_2, \theta)| \frac{d\theta}{\theta}$$

Tables 5.1 and 5.2 show the values of the two metrics for the signatures of the outlines

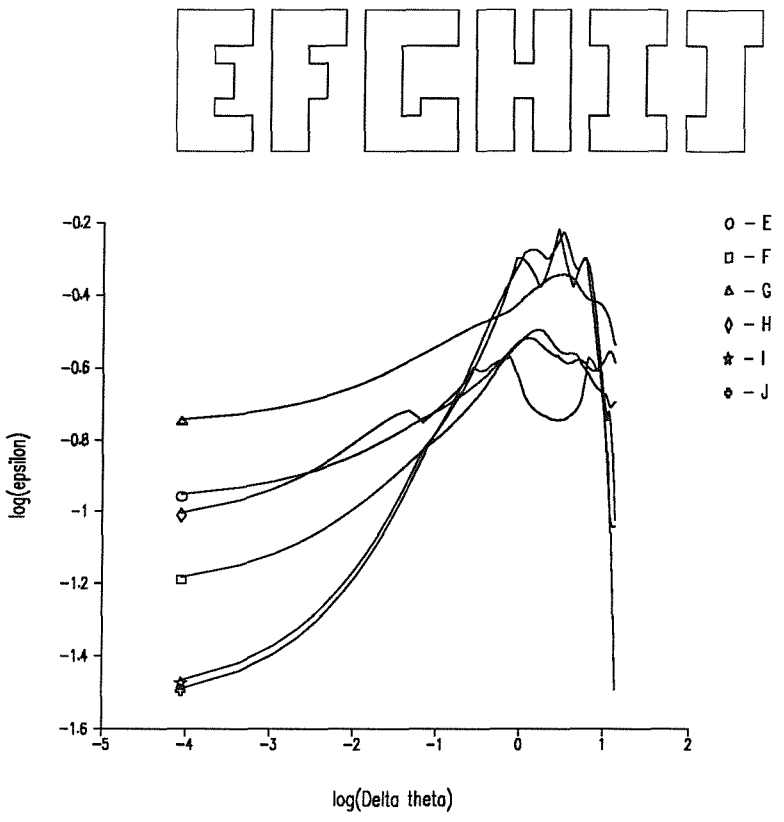


Figure 5.9: Signatures of Letters E-J

of the letters in figure 5.9.

The Euclidean metric appears to be more appropriate, since the bounded area does not handle the *I-J* case very well: the two signatures are quite different near $\Delta\theta = \pi$ but this adds very little to the area. The Euclidean metric though appears to have a reasonable value for all the letter outlines. We shall therefore use the Euclidean metric as a measure of the distance between two signatures.

5.4.1 Shape Confidence

Since it is often more convenient to have a measure of similarity rather than a measure of dissimilarity, we will define an additional metric. I define the confidence C_i that an

	E	F	G	H	I	J
E	0	0.856	1.869	1.460	3.511	3.418
F		0	2.387	1.599	3.072	3.032
G			0	2.889	3.424	3.098
H				0	3.840	4.015
I					0	0.958
J						0

Table 5.1: Euclidean Distance

	E	F	G	H	I	J
E	0	0.606	0.950	0.279	1.478	1.578
F		0	1.523	0.743	0.947	1.053
G			0	1.101	1.973	2.035
H				0	1.568	1.664
I					0	0.142
J						0

Table 5.2: Bounded Area

unknown shape S corresponds to a library shape L_i as

$$C_i = \frac{1}{1 + k \cdot E_i} \tag{5.6}$$

where E_i is the Euclidean Distance between the signatures of S and L_i. The positive constant k is chosen such that $C_i = 0.5$ represents a sensible intermediate between 1.0 which represents complete confidence in the match and 0.0 which represents no confidence in the match at all.

To determine the value of k we will make an arbitrary decision, based on table 5.1, that a Euclidean distance of 1.0 will correspond to a confidence of 0.5 ie. we are just more than 50% confident that an "E" is an "F" and an "I" is a "J". From equation 5.6 it follows that $k = 1$.

Considering a more complicated shape, we can see that this value of k does give us a sensible value for our confidence measure. The two outlines in figure 5.10 are identical apart from the rear spoiler, the confidence that both signatures represent the same shape is 0.60. Since the difference in shape is less than that between the letters E and F this seems a reasonable value. If we now also consider the signatures of the

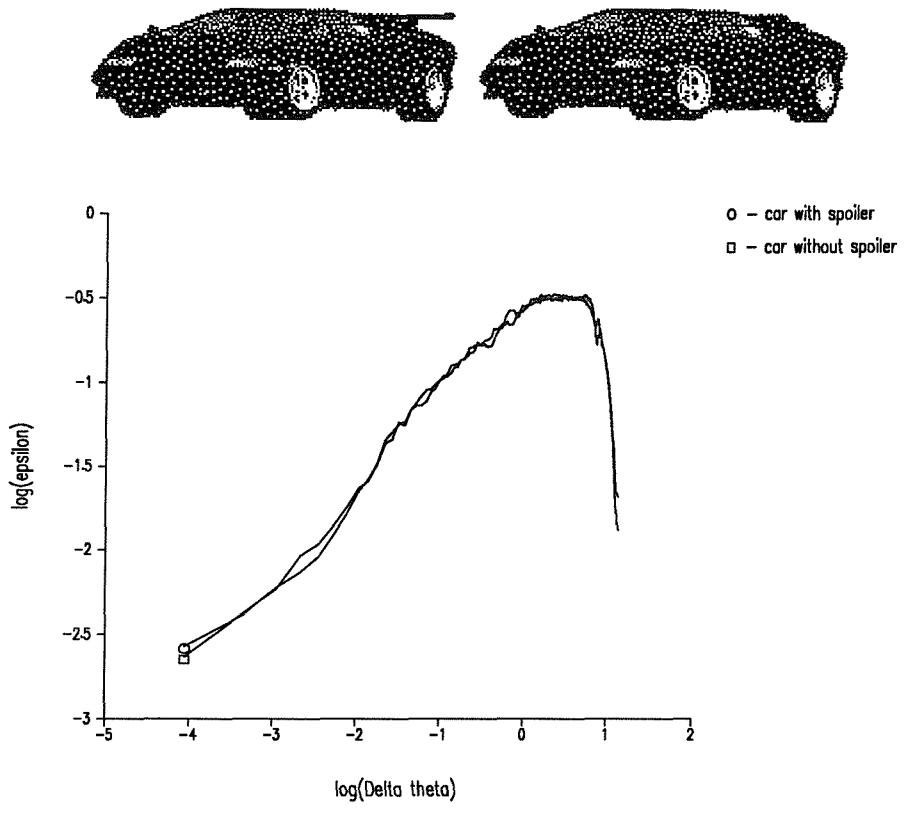


Figure 5.10: Signatures of Car with and without Spoiler

clearly different shapes in figure 5.3 then we get the following confidences that any pair of signatures represents the same shape

	Triangle	Arrow	Car
Square	0.06	0.04	0.04
Triangle		0.11	0.08
Arrow			0.18

As expected, all these values are low since the shapes (and their signatures) are very different. Hence, we appear to have a reasonable measure of shape similarity.

5.4.2 Matching From Unknown Viewpoints

We can now use the confidence measure defined above to examine the algorithm proposed in section 5.3.1 for handling perspective transformations. For simplicity

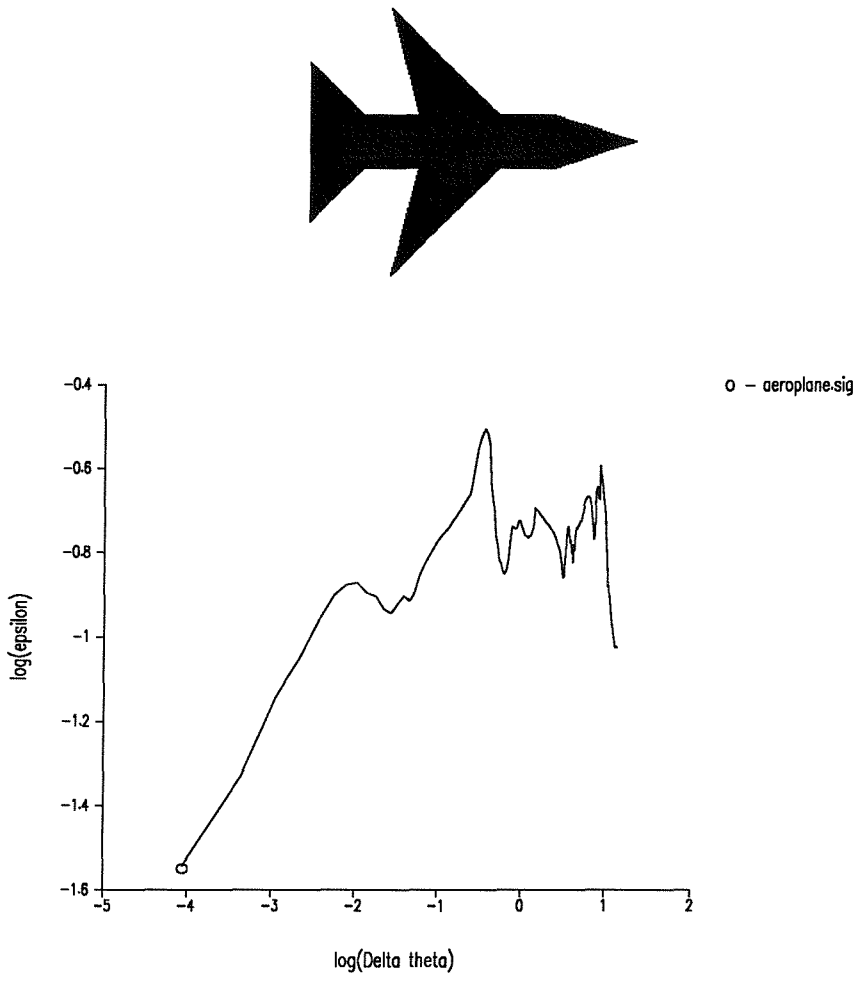
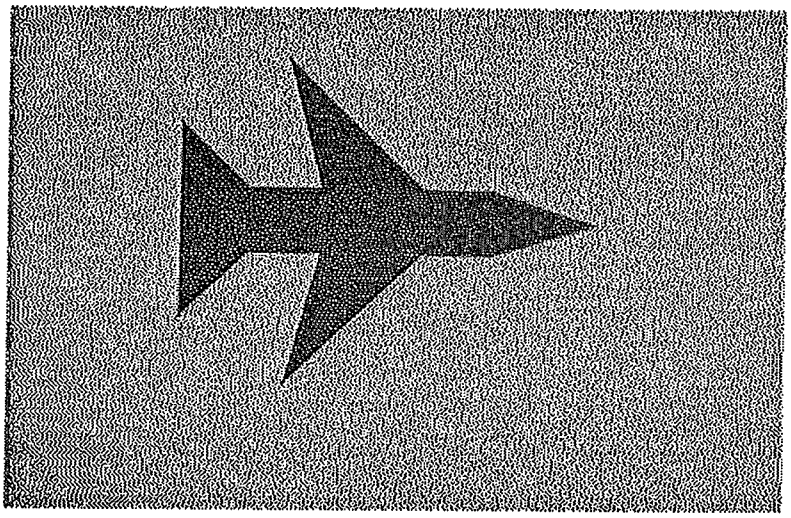


Figure 5.11: Aeroplane Shape and Signature

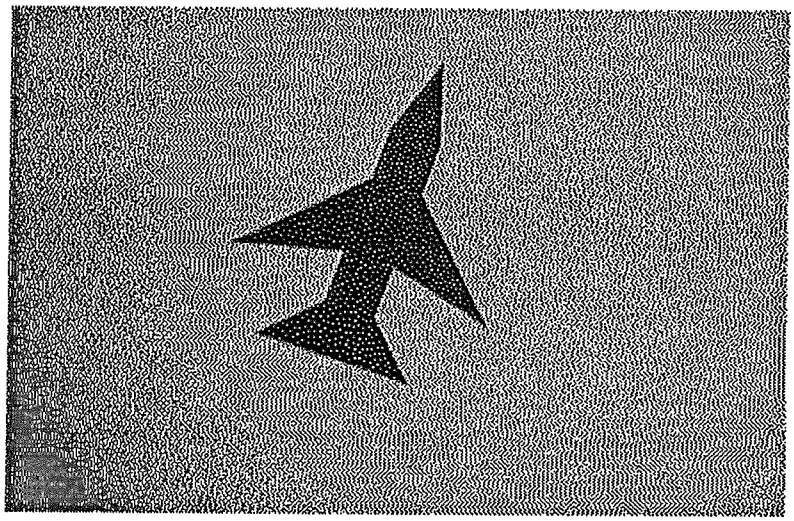
I shall only consider a single viewing distance (far from the object) although the algorithm as described can be generalised to handle all possible viewpoints.

Figure 5.11 shows a simple outline of an aeroplane and figures 5.12 and 5.13 shows the same outline viewed from four different points. The fractal signature is reflection invariant so we will only get unique signatures for viewpoints $0^o \leq \alpha < 90^o$, $-180^o \leq \beta \leq 0^o$. If we choose an initial resolution of 10^o in each direction then we need to store $1 + 8 \times 19 = 153$ signatures (if the shape has no axis of symmetry we need 289).

Given an unknown view we perform the following steps,

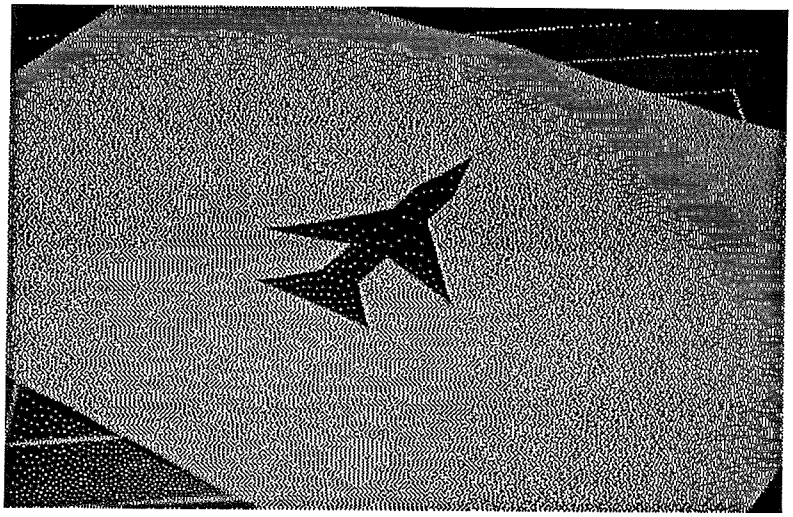


(a) Aeroplane 1

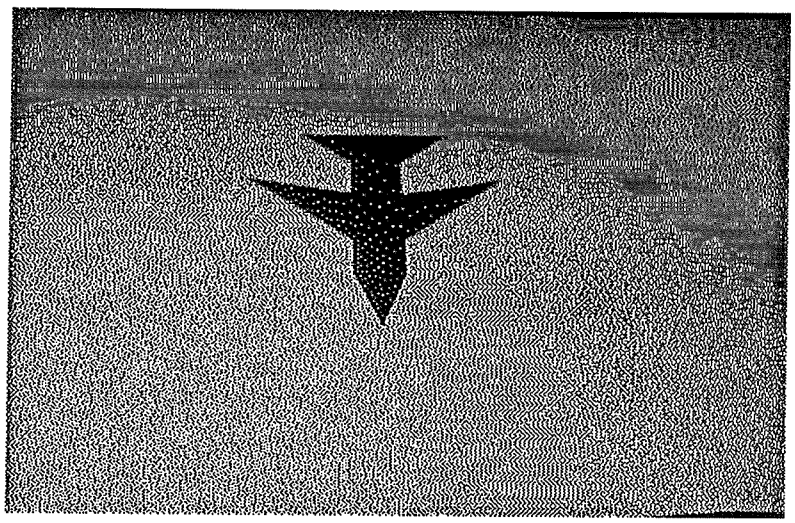


(b) Aeroplane 2

Figure 5.12: Aeroplane Shape from Different Viewpoints



(c) Aeroplane 3



(d) Aeroplane 4

Figure 5.13: Aeroplane Shape from Different Viewpoints

View	Step	(α, β)	Confidence
1	10	10,-90	0.77
	5	5,-90	0.78
	2	5,-90	0.78
	1	5,-90	0.78
2	10	10,-160	0.67
	5	10,-160	0.67
	2	10,-160	0.67
	1	9,-159	0.68
3	10	50,-140	0.50
	5	50,-135	0.55
	2	52,-133	0.56
	1	52,-133	0.56
4	10	50,0	0.50
	5	55,0	0.53
	2	53,0	0.55
	1	53,-1	0.62

Table 5.3: Viewpoint Determination for Aeroplane Shapes

1. Extract the shape boundary and calculate its signature S;
2. Compare S with the stored signatures and select the nearest, ie. highest confidence, match;
3. Given the best viewpoint, re-project the original outline from the 8 surrounding viewpoints 5^o away in α or β or both and calculate the signatures;
4. From these 8 signatures and the central one, select the nearest to S;
5. Repeat steps 3 and 4 with increments of 2^o and 1^o.

Table 5.3 summarises the results of applying the above procedure to each of the four views in figures 5.12 and 5.13.

All four views appear to have been pinpointed to approximately the correct viewpoints. Physical estimates of the actual viewpoint could only be made crudely to within about $\pm 5^o$ but these estimates agree with table 5.3. More precise experiments have not been carried out because the accurate determination of viewpoint is not the objective here: what is important is the confidence that the unknown shape corresponds to a view of the known shape from *some* viewpoint.

All the confidences in table 5.3 are high but perhaps not as high as we might like. One

problem that was encountered in these experiments was that of the camera's aspect ratio. The quoted figure for our hardware is 5:6 but a quick experiment revealed that it was actually closer to 2:3. Additionally, this aspect ratio is not constant across the whole image, resulting in some camera distortion. In the above experiments the average aspect ratio was assumed to be 2:3.

Note that views 3 and 4 have less confident matches than views 1 and 2. The most likely explanation for this is that any errors in the boundary location (due to noise, non-optimal segmentation or camera distortion) are more significant for "lower" views (ie. greater α) because the same distance in the image plane corresponds to a larger distance in the object plane.

The major disadvantage of using the signature in this manner, rather than trying to produce some sort of invariant, is the large number of signatures required for the initial viewpoint estimate. In the following section we will investigate the possibility of calculating invariant properties from a single signature.

5.5 Uniqueness of the Fractal Signature

After examining many examples the normalised signature appears to be unique ie. no two clearly different shapes have been found which have very similar signatures. A more precise mathematical analysis of the fractal signature's properties is extremely difficult due to the presence of the *modulus* term (equation 5.5). In this section we will investigate the possibility of predicting certain properties of the fractal signature by replacing this term with a *square*.

5.5.1 Simplifying The Problem

If we have a simple, single-valued, shape $r(\theta)$ then we can write it as a Fourier Series in θ ie.

$$r(\theta) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos n\theta + b_n \sin n\theta$$

where,

$$\begin{aligned} a_n &= \frac{1}{\pi}\int_0^{2\pi} r(\theta)\cos n\theta \, d\theta \\ b_n &= \frac{1}{\pi}\int_0^{2\pi} r(\theta)\sin n\theta \, d\theta \end{aligned}$$

Our fractal signature $\epsilon'(\Delta\theta)$ is defined as

$$\epsilon'(\Delta\theta) = \frac{E[\,|\rho|\,]}{E[r(\theta)]}$$

where,

$$\rho = r(\theta) - r(\theta + \Delta\theta)$$

It is difficult, if not impossible, to calculate $E[\,|\rho|\,]$ in terms of $\Delta\theta$, a_n and b_n, but it is possible for $E[\,\rho^2\,]$, and $E[r(\theta)]$ is simply $a_0/2$. Hence, we can analyse the properties of the mean of the square and then see if they hold for the mean of the absolute value. We can define a new signature $S(\Delta\theta)$ as

$$\begin{aligned} S(\Delta\theta) &= \frac{E[\,\rho^2\,]}{E[r(\theta)]^2} \\ &= \frac{4}{a_0^2}\cdot\frac{1}{2\pi}\int_0^{2\pi}\Big(\sum_{n=1}^{\infty} a_n[\cos n\theta - \cos n(\theta+\Delta\theta)] + \\ &\qquad\qquad b_n[\sin n\theta - \sin n(\theta+\Delta\theta)]\Big)^2 \, d\theta \\ &= \frac{4}{a_0^2}\sum_{n=1}^{\infty}(a_n^2 + b_n^2)[1 - \cos n\Delta\theta] \end{aligned} \tag{5.7}$$

Hence, we can calculate the function $S(\Delta\theta)$ for any simple shape given its Fourier coefficients. Conversely, we can reconstruct any shapes which may have identical functions $S(\Delta\theta)$ and test them to see if they also have similar fractal signatures.

5.5.2 Implications

If the Fourier coefficient pairs are considered as cartesian points (a_n,b_n) then equation 5.7 means that $S(\Delta\theta)$ will be unchanged if any of the points (a_n,b_n) are rotated about

(a) Square and Approximation

(b) Approximation with 4th harmonic rotated by $\pi/4$, $\pi/2$, $3\pi/4$ and π

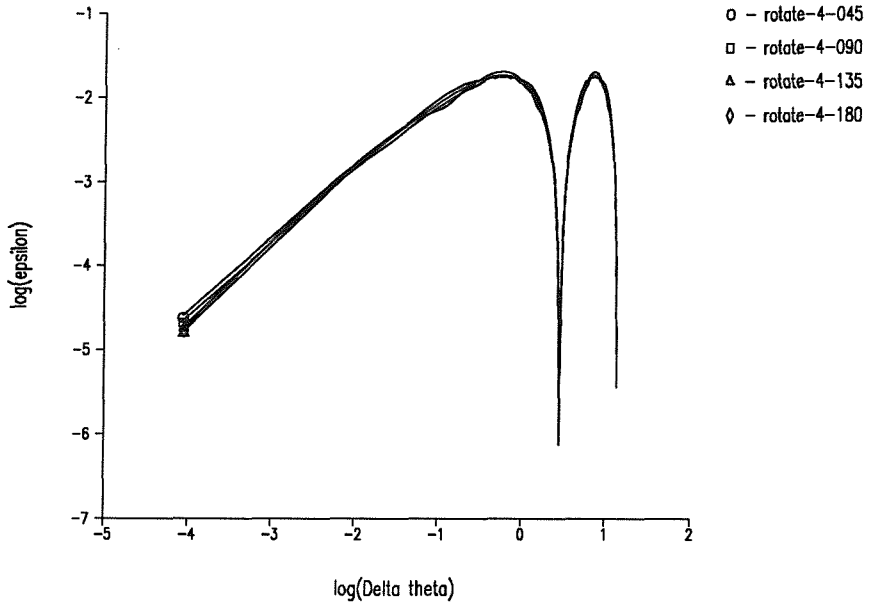
(c) Approximation with 8th harmonic rotated by $\pi/4$, $\pi/2$, $3\pi/4$ and π

Figure 5.14: Harmonic Variations on a Square

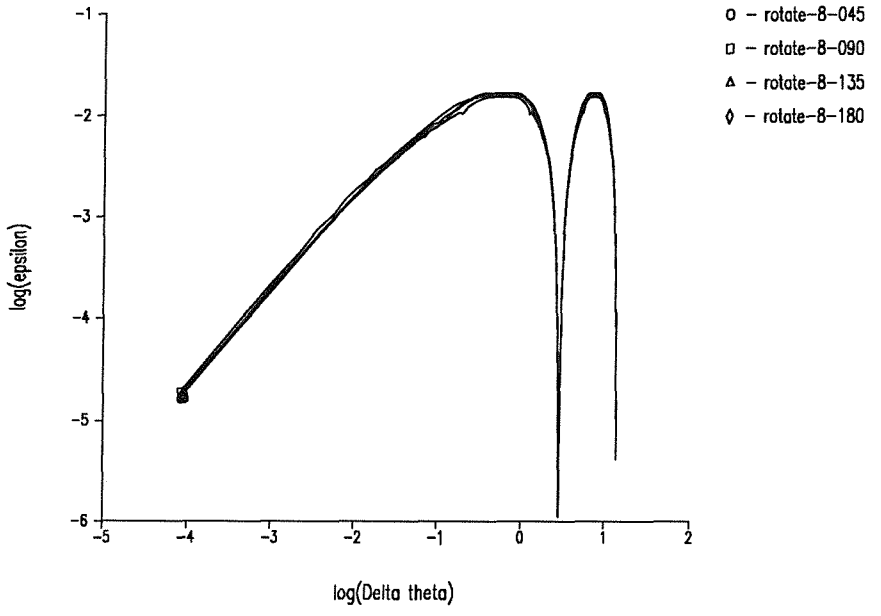
the origin by any angle, since $a_n^2 + b_n^2$ is the square of the distance from the point to the origin. Examples of this are shown in figure 5.14: the approximation to the square shape in 5.14(a) is made from the Fourier Series up to $n = 32$ (only every fourth coefficient is non-zero). In 5.14(b) the pair (a_4,b_4) has been rotated by multiples of $\pi/4$ and in 5.14(c) the pair (a_8,b_8) was rotated. Note that the higher the harmonic, the smaller the effect of the rotation on the shape.

All the shapes in figure 5.14 have the same value of $S(\Delta\theta)$ for all $\Delta\theta$ so this suggests that our signature $\epsilon'(\Delta\theta)$ may be unable to distinguish between them. However, this is not the case: when we calculate the signatures of the shapes in figure 5.14 we get the graphs in figure 5.15. These curves are close, but by no means indistinguishable. The Euclidean separations of the signatures in 5.15(a) range from 0.62 to 0.90 and for those in 5.15(b) the range is 0.36 to 0.84. For all the signatures, the separation from the signature of the approximate square ranges from 0.67 to 1.14.

The variations in the Fractal Signatures of the given class of shapes are completely compatible with the other results in this chapter. It is therefore clear that the prediction



(a) Signatures of approximate square with 4th harmonic rotations



(b) Signatures of approximate square with 8th harmonic rotations

Figure 5.15: Signatures of Approximate Square and Harmonic Variations

made, as a result of analysing $S(\Delta\theta)$, is not true, or even approximately true, for $\epsilon'(\Delta\theta)$.

5.5.3 Summary

We have seen that replacing the *modulus* in the Fractal Signature definition by a *square* enables predictions to be made but that these predictions may not necessarily hold. Since the prediction was that a given class of shapes would have the same Fractal Signature it is encouraging to find that in this case the prediction is incorrect: however, we have still not proved that the Fractal Signature is unique. On the one hand we have shown that the Fractal Signature is better than first thought, but on the other hand we are left with no firm predictions about its properties.

5.6 Shape Reconstruction from Fractal Signature

Since the primary use of our fractal signature is for shape matching it is not necessary for us to be able to reconstruct a shape's boundary from its fractal signature. For completeness, though, it would be nice if this could be done.

In the previous section we saw that an approximation to the signature could be expressed in terms of the Fourier series of the shape. Unfortunately, we also saw that this approximation was not valid and therefore unlikely to be useful. Given the presence of the modulus term in the definition of the fractal signature it is unlikely that the signature itself can be expressed in terms of a series.

Work on the simulation of fractals by other researchers (eg. Fournier [41]) has shown that it is possible, given a fractal dimension, to compute examples of boundaries or surfaces which have approximately that fractal dimension. I am not aware of any work on the reconstruction of multi-fractals given a set of fractal dimensions for different scales.

I believe that it will be possible, at some stage, to reconstruct shapes to some degree of accuracy from their fractal signatures: however, this will not be pursued further in this thesis.

5.7 Effects of Noise and Occlusion

Due to imperfections in the image capture process and the limitations of our segmentation algorithms, it is very likely that a shape extracted from an image will be distorted somewhat by noise. In addition, our shape may be partially occluded by some other object in the scene.

As each ϵ' value is an average over the whole shape, minor occlusions and deformities are smoothed out to some extent. Therefore the signature will change gradually as a shape is occluded or becomes deformed by noise. However, we would expect the change in the signature to be smaller for some random variation (ie. noise), which may cancel out to some extent, than for a systematic variation (ie. occlusion) which will not cancel.

Figure 5.16 shows the E boundary (originally 65×130 pixels) with noise added to the x and y positions of all the points. The noise is zero-mean Gaussian with the given standard deviation and the coordinates rounded to the nearest integer. The noisy boundary is then considered to be the polygon joining the points in the same order in which they originally existed.

In figure 5.17 we can see the Euclidean distance between the signature of the noisy boundary and the signatures of the original shapes. Since the noise is *random* we will obviously have some variation in the distances, the values plotted are an average for 10 examples.

Note that for all the noisy E boundaries the closest signature is that of the original E. However, for noise with a standard deviation of more than 0.6 pixels the distance to the E is greater than the distance between the original E and F.

It is also the case for the other five letters that the signature of a noisy example is always closer to the signature of the same letter than to any of the others. These results show that for this test set there would be no classification errors due to noise of the magnitudes considered.

Figure 5.19 shows the greater degradation caused by occlusion. The signature of the

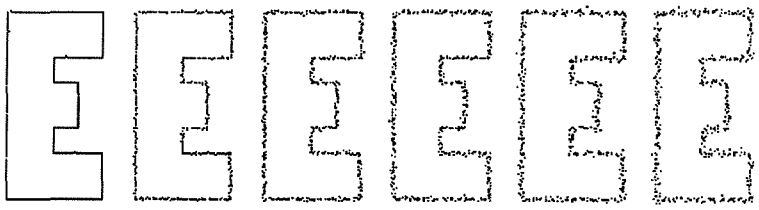


Figure 5.16: E with noise s.d. 0.2, 0.4, 0.6, 0.8, 1.0 and 1.2

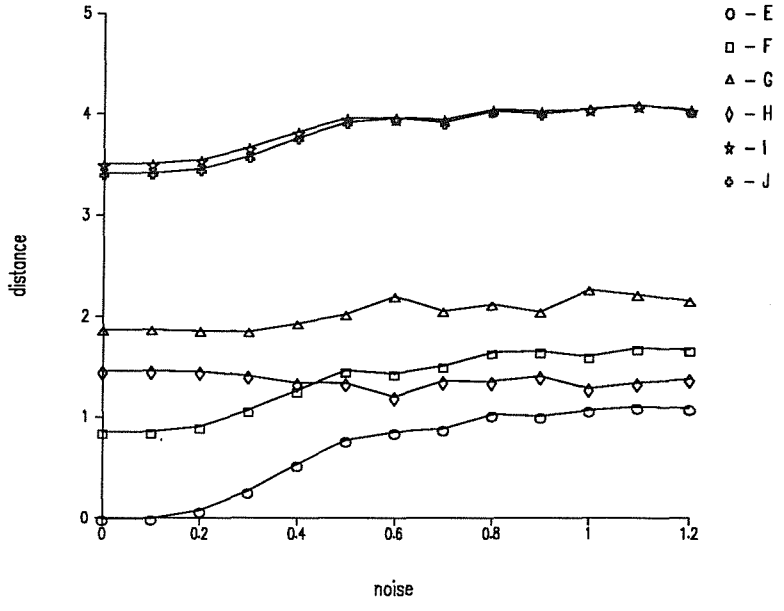


Figure 5.17: Euclidean metric on E's with noise

occluded E becomes closer to the original F than the original E after just over 25% of the lower stroke is occluded. When the shape is occluded in other ways the effects will be equally unpredictable.

Since our shape measure is global (ie. measured over the whole shape) we cannot expect it to be unchanged when the shape is occluded by any significant amount. Indeed, we would not want it to since we want different shapes to have different signatures. The best that we can manage is that the signature changes gradually as the shape is occluded by larger amounts.

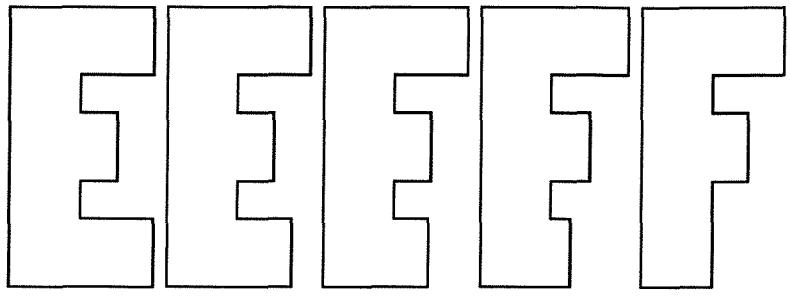


Figure 5.18: E occluded 0,1,2,3,4

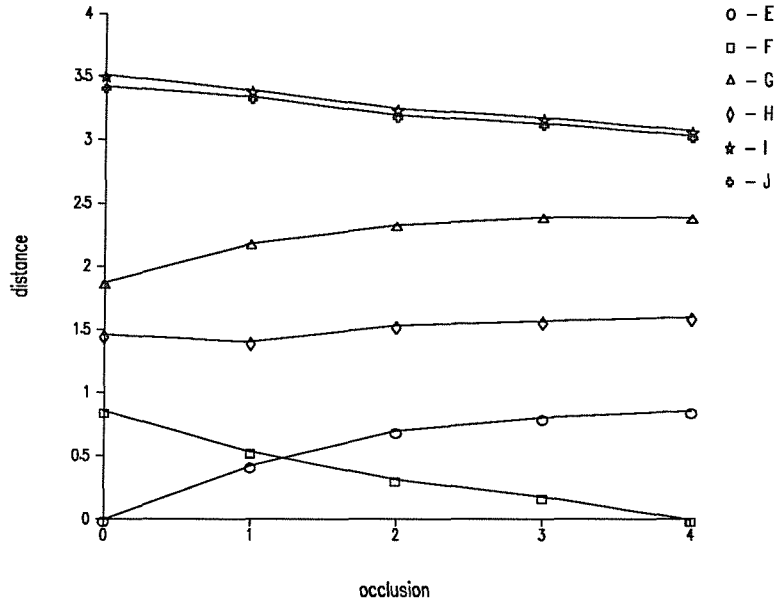


Figure 5.19: Euclidean metric on occluded E's

5.8 Comparison of Fractal Signature and Fourier Descriptors

In the previous section we added moderate amounts of noise to shapes from a small test set and found that no classification errors were made. In order to compare the performance to that of an established technique (the FDs of Persoon and Fu [86] described in section 4.2.1) I chose a larger test set and a higher noise limit.

The test set is shown in figure 5.20 and their signatures are shown in figures 5.21 and 5.22. The digits shown are from the *Courier Bold* font and are approximately 32×32 pixels in size. As a demonstration of the amount of noise used, figure 5.23 shows the

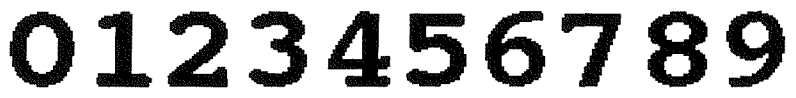


Figure 5.20: Sample shapes of digits

digit 3 with noise at standard deviations 0.2 to 1.0 pixels. Since the shape is roughly a quarter of the size of the previous example, this represents a signal-noise ratio of four times less.

All of these noisy threes are recognisable by a human being since we can selectively ignore points which are clearly in the wrong place. Figure 5.23 shows the original digit 3 and an example of one with noise of standard deviation 0.6 added. Clearly, the process of connecting the boundary points in the same order as the noise-free case makes recognition more difficult since this order is often unnatural. However, this is the simplest definition of a boundary with a given amount of noise and we have a fair comparison since both methods will use the same boundaries.

For the FDs 16 harmonics were calculated and normalised for starting point, position, rotation and scale as described in section 4.2.1. In Persoon and Fu's experiments (which also used digits 0-9) they used only 8 harmonics but we have used more here for a fairer comparison because the fractal signature has a much larger amount of data: harmonics higher than 16 would be useless because their magnitudes are very small and they are difficult to calculate reliably.

Given two sets of 16 complex coefficients $a_0 \ldots a_{15}$ and $\alpha_0 \ldots \alpha_{15}$ the distance between them is defined as

$$d = \sqrt{\sum_{n=0}^{15} |a_n - \alpha_n|^2}$$

which is conveniently similar to the distance measure for fractal signatures.

The following procedure was used to compare the robustness of the Fractal Signature and the Fourier Descriptors -

1. For each digit [3], calculate its fractal signature and store it as a reference;

[3]The bitmaps for the digits 6 and 9 were found to be 180° rotations of each other and were thus considered to be a single shape.

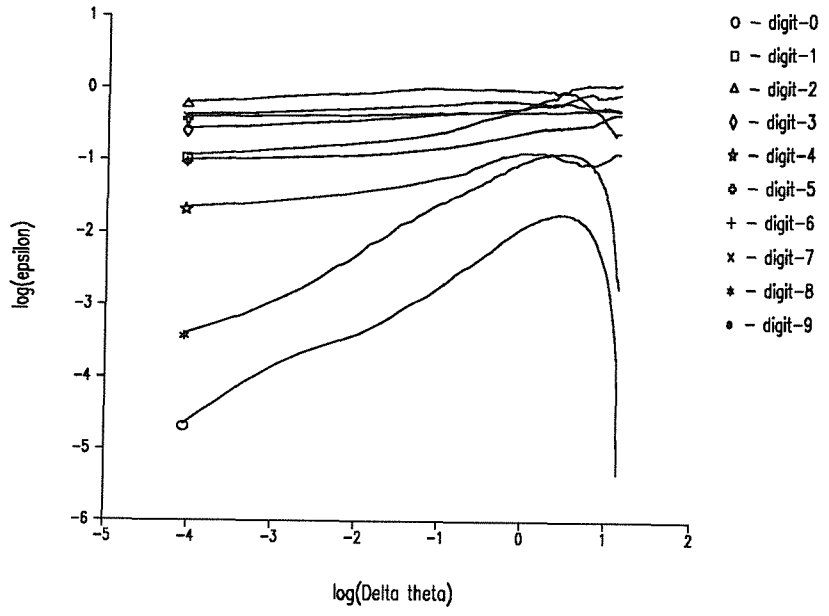


Figure 5.21: Signatures of digits 0-9

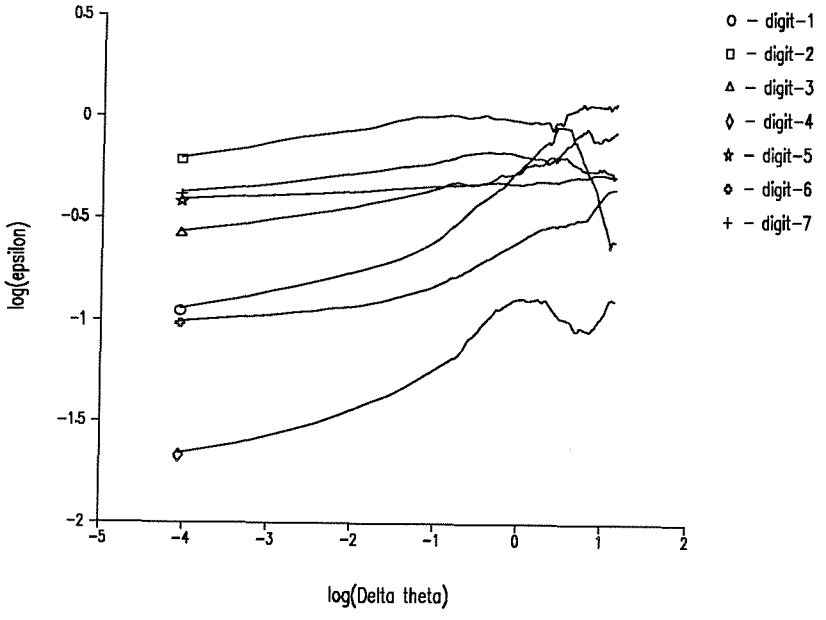
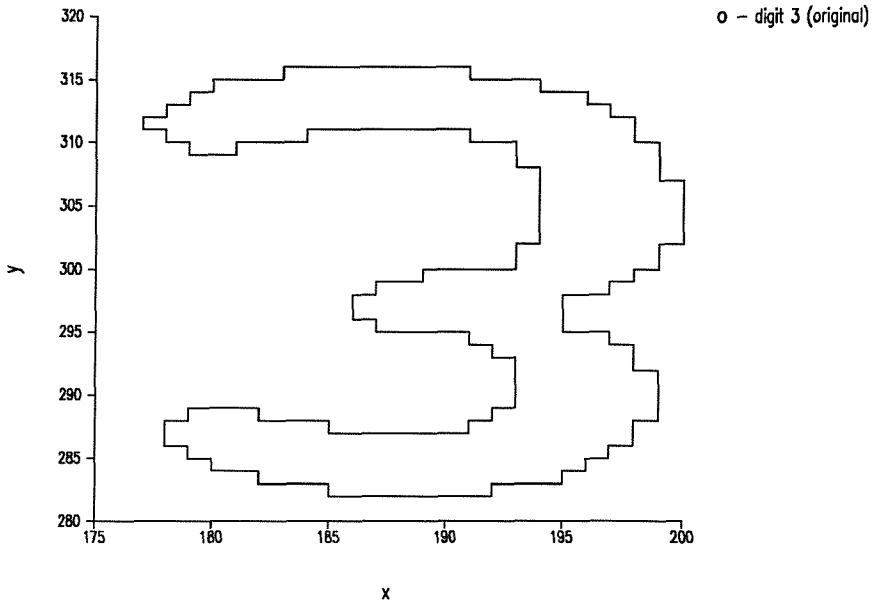
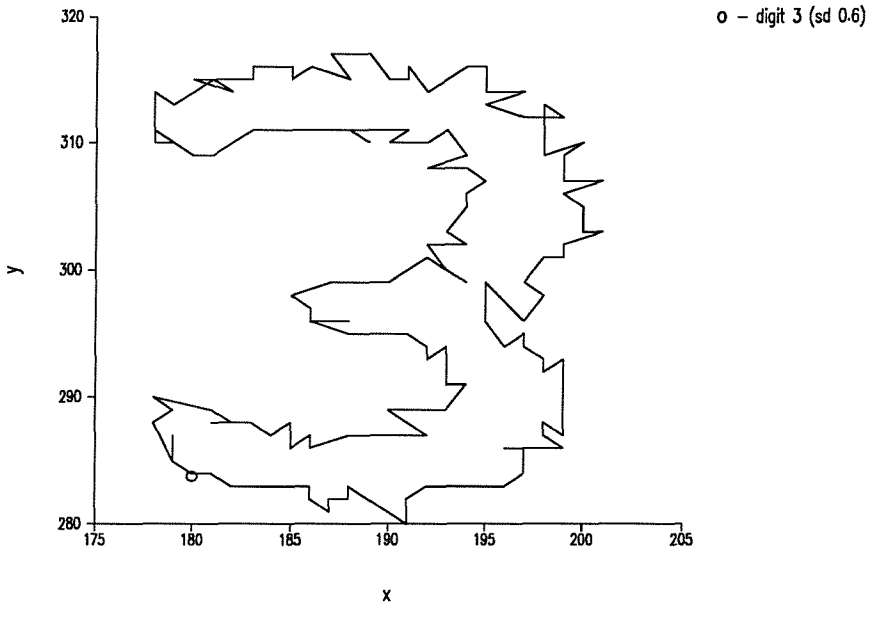


Figure 5.22: Signatures of digits 1-7

(a) Digit 3 with noise s.d. 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0



(b) Original Digit 3



(c) Digit 3 with noise 0.6

Figure 5.23: Effect of noise on digit 3

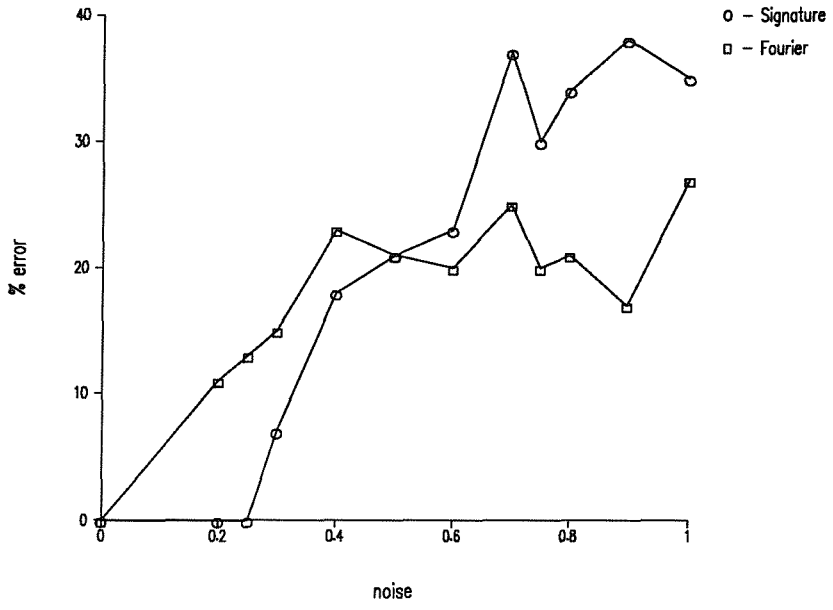


Figure 5.24: Robustness of Signature and Fourier Descriptors

2. For each noise level, add noise to each digit ten times to produce a large number of noisy shapes;
3. For each noisy shape, calculate its fractal signature and classify the shape as being the digit whose reference signature is *closest* to the calculated signature;
4. Calculate the percentage of incorrect classifications;
5. Repeat for the Fourier Descriptors.

The results of the above analysis can be seen in figure 5.24. It is interesting to note that both methods produce a classification error of 21% for noise of standard deviation 0.5 pixels: below that level the fractal signature produces less classification errors than the FDs but for higher noise levels the performances are reversed.

Adding increasing amounts of noise to a shape affects the smaller scale (or higher frequency) appearance of the shape much more significantly than the larger scale (lower frequency) appearance. Because the FDs decrease in magnitude rapidly with frequency a large proportional change in the high frequency coefficients only

contributes a small numerical increase in distance from the original. On the other hand, the fractal signature represents all scales evenly so a large proportional change in any value will produce a large change in the distance from the original.

If we examine the distances between all the original digits in feature space for both representations then we find the following

	Signature	FD
mean	11.0	0.50
sd	21.2	0.12
max	32.7	0.78

These figures become significant when we note that the average effect of noise at s.d. 0.25 pixels on the digits is of the order of 0.5 for the fractal signature and 0.05 for the FD. Thus, this level of noise represents about 5% of the average separation for the fractal signature but 10% for the FDs.

I conclude that the fractal signature itself is better at discriminating between the shapes in the test set than the FDs. However, its performance degrades more rapidly with noise because the simple matching process used does not take into account the fact that noise will affect the smaller scale values more significantly than the larger scale values.

5.9 Conclusions

In this chapter I have presented a new shape descriptor based on Fractal Geometry. By modelling the boundary as a Fractional Brownian Function (FBF) the *fractal shape signature* captures the statistical variations of a shape across *all scales*. By using the information from all scales, rather than trying to extract a single "characteristic fractal dimension" for the shape, the fractal signature can easily distinguish between shapes which have the same dimension. This means that the fractal signature is equally applicable to all types of shapes since those with a "clear" fractal dimension (ie. a largely linear signature) are modelled equally as well as those without.

From its definition the fractal signature is invariant under translations, rotations and reflections of the shape boundary. Whilst translation and rotation invariance are essential for our application, reflection invariance is neither essential nor a problem (although it could be in some cases). The fractal signature is also easily made invariant to the size of the shape: in circumstances when we require absolute size information this can easily be inferred from the length of the boundary.

In the previous chapter I mentioned techniques which were invariant under perspective transformation. The fractal signature is not invariant under such transformations and probably cannot be made so. To capture the viewpoint variance of the signature it is necessary to use a set of signatures from a large number of viewpoints. This initially seems a major disadvantage in comparison with projective invariant descriptions. However, we must remember that the 2D shapes we are describing are the projections into an image of 3D surfaces: if those surfaces are flat then the projections from different views will indeed be related by perspective transformations; if the surfaces are not flat then their projections will not be projectively invariant. Therefore, for describing the projections of general surfaces we will need to use multiple signatures anyway (as will be demonstrated in chapter 7).

The major criticism of fractal-based techniques such as the IFS was that they did not lend themselves easily to shape matching. The fractal signature presented here has been specifically designed for matching, the premise being that matching two 1D translation, rotation and scale independent signals is easier than matching two 2D boundaries with different positions, orientations and sizes. The matching strategy described in section 5.4, based on the Euclidean separation of the signatures, was chosen mainly for its simplicity. It is intended only to demonstrate the potential of the fractal signature. Whilst the derived shape confidence measure is useful in determining whether shapes are similar or dissimilar it is not so useful where the choice is less clear cut.

A major setback in making firm predictions about properties of the fractal signature has been the difficulty in performing a precise mathematical analysis. This has meant that I have been unable to prove whether the fractal signature is unique or not and have

also been unable to analyse the effects of affine transformations. It has been suggested that such problems would be solved by redefining the signature in terms of a *squared difference* rather than a plain *difference*. Unfortunately, we saw in section 5.5 that this was a blind alley.

Whilst the signature is a consistent representation of shape it is a global description and therefore cannot handle large amounts of noise and occlusion. I have demonstrated that the fractal signature can perform better than a comparable global shape descriptor (ie. Fourier Descriptors) for moderate amounts of noise and have suggested that the performance for larger amounts of noise could be improved by a more sophisticated matching technique. The same cannot be said for occlusion, the global nature of the fractal signature means that significant occlusions will render it unrecognisable. Therefore, other means of handling occlusion are required and some possibilities will be discussed in chapter 7.

6 Three-dimensional Object Recognition

The problems of modelling 3D objects and finding instances of those models in images have received much attention, particularly over the last decade. Several comprehensive surveys have been published, notably by Binford [17], Besl and Jain [12], Besl [11], Chin and Dyer [29]. In this chapter we will see some of the more popular, and practical, approaches to 3D modelling and matching. Some researchers [26,37,52,79] have used range images to some degree of success but we will concentrate here on approaches that are applied to ordinary chromatic images.

6.1 Object-centred Approaches to 3D Modelling

An object-centred representation is one in which no (or little) account is taken of the way that the object may be viewed. The object is specified in terms of 3D points, lines, surfaces or volumes which are all defined using some internal co-ordinate system.

Common examples of object-centred representations are those which are closely tied to Computer Aided Design (CAD) systems. Perhaps the most obvious CAD-based representation for polyhedral objects is the wire-frame, illustrated in figure 6.1. The object is defined as a simple list of edges, but this has the problem of ambiguity, as shown.

Other, non-ambiguous, CAD-based representation schemes are reviewed by Bhanu and Ho [13] including

- surface points and normals;
- surface curvatures;

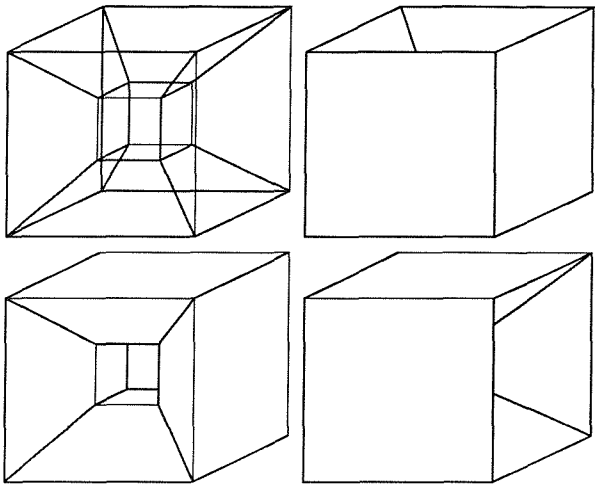


Figure 6.1: Ambiguity of Wire-Frame Representation

- generalised sweep;
- extended Gaussian image;
- object decomposition (eg. Constructive Solid Geometry).

They assess the different schemes according to Marr's [67] shape representation criteria - scope, accessibility, conciseness, uniqueness, stability and sensitivity. The favoured representation is surface points and normals which scores well in all respects except conciseness. In a further paper, Bhanu, Ho and Henderson [14] expand on their approach and describe how to produce a 3D point-normal model from a standard CAD-model to a given resolution.

A related approach by Falcidieno and Giannini [36] produced a face-based representation of solid objects from CAD models. Here an object is decomposed into a hierarchy of components such as slots, pockets, holes and protrusions, which are described by face adjacency hyper-graphs (FAHs). A FAH is a graph in which nodes represent faces, arcs connect nodes which share a common edge and hyper-arcs connect nodes which share a common vertex.

The motivation for Falcidieno and Giannini's work is that a face-based representation of objects is required in order to integrate information about surface finish. This idea also applies to Vision applications where we need to describe the colour and texture

of parts of objects.

For the purposes of object recognition an object-centred modelling system requires an algorithm for detecting the 3D translation, 3D orientation and the scale which allows the model to be projected correctly into the image. This involves the determination of seven parameters. Ballard and Sabbah [6] have extended the Generalised Hough Transform (section 4.4.1) to 3D to solve for six parameters (they assume orthographic projection so the absolute depth is lost). The process is divided into two stages, the orientation parameters (three) are determined in the first step and then the translation parameters (two) and the scale parameter are determined by the second step. Like the 2D GHT, this method has the advantage of insensitivity to noise and occlusion but the disadvantage of large computation and storage requirements. Stockman [94] describes a system which utilises a similar approach (although he calls it "pose clustering") to determine pose from both 2D and 3D data to within $1mm$ for $400 \times 400 \times 200mm$ objects but states that the approach is only suitable when the number of models is small and that hardware development is required to speed up the computations.

Brooks has produced an object recognition system called ACRONYM [21] which has been used to interpret aerial images of airfields. Objects (eg. aircraft) are modelled as subpart hierarchies of generalised cones. A generalised cone is a 3D curve (the spine) along which a plane shape (the cross section) is swept: the cross section is kept at a constant angle to the tangent of the spine and is deformed according to a deformation function known as the sweeping rule. An example of a generalised cone, representing a wing section, is shown in figure 6.2, here the spine is a straight line and the cross-section is a spline function, the sweeping rule moves one of the control points of the spline away from the spine in a linear manner.

Each generalised cone therefore has its own local coordinate system which must be related to the object's coordinate system. Additionally, cameras are modelled as coordinate systems and are placed in the *world* with the modelled objects by constraining the transforms between their local coordinate systems and the world coordinate system.

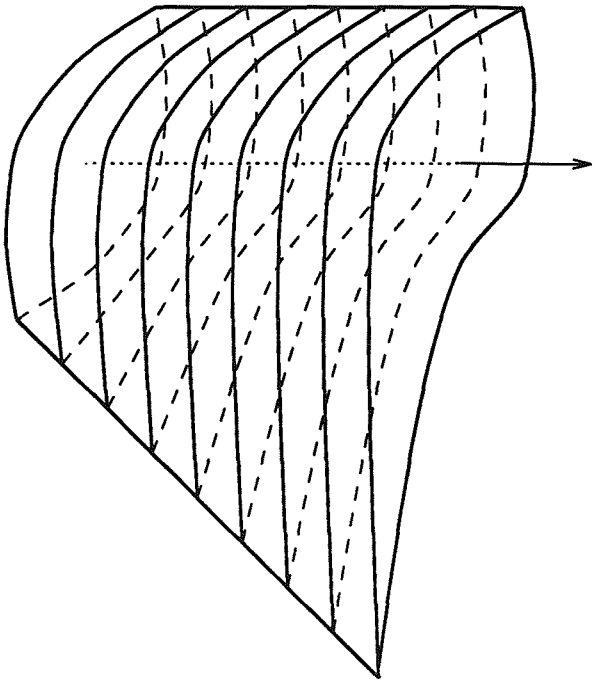


Figure 6.2: Generalised Cone Representation

Given this model of the world, ACRONYM first extracts a description of the image in terms of ribbons (the 2D projection of generalised cones, restricted to having straight spines and linear sweeping rules) and ellipses. Using these primitives, predictions are made as to the appearance of object components in the image. The predictions are initially constrained by rules derived from the 3D parts of the model, matched primitives are then used to further constrain the size and orientation of the 3D parts, matched parts are then grouped into objects to further constrain the pose of the object.

Unlike ACRONYM, which matches 2D image features to predicted 2D features, the 3D MOSAIC system described by Walker and Herman [105] performs matching of 3D structures. The application area here is the extraction of 3D wire-frame models from aerial images of complex urban scenes. Once 2D lines and junctions have been extracted from an image the system uses geometric reasoning to extract 3D structures such as edges and vertices.

The geometric reasoning employed by MOSAIC stems from the assumption that all surfaces are either horizontal or vertical. Once lines have been labelled as either

horizontal or vertical then the positions of their endpoints are determined by assuming that one end of certain vertical lines must lie on the (known) ground plane. Objects are then completed using knowledge about the type of objects that will appear. This knowledge is implicit and some user interaction is required; however, the authors state that the system could be improved by using explicit domain knowledge.

Lowe [61, 62] proposes object recognition based on the process of *perceptual organisation* which detects groupings and structures in the image that are likely to be invariant over wide ranges of viewpoint. The approach presented is limited to straight line segments but can theoretically be extended to arbitrary curves (some progress in this area has been made by Horaud et al. [48]).

The suggestion is that recognition of 3D objects need not proceed via explicit derivation of depth information. This is supported by psychological evidence which shows almost identical reaction times for humans recognising line drawings and full-colour slides of the same objects from the same viewpoints. In addition Lowe states that recognition of objects from complete depth images has not been shown to be easier than for systems using only two-dimensional images.

The model (only one object!) used is a set of 3D line segments (wire-frame) with a simple hidden-line marking scheme. The basic procedure is as follows

1. Find lines in the image
2. Group lines on the basis of
   - Proximity
   - Parallelism
   - Collinearity
3. Rank the groupings in order of probability that they are non-accidental
4. Compare the groupings to the model
5. Determine the viewpoint
6. Project the model into the image and check that all the model parts are consistent with the viewpoint

7. Accept or reject the match

The key operations are the perceptual grouping and the viewpoint consistency constraint. By only using groupings of three or more line segments a significant speed up is achieved since such groupings will contain enough information to completely solve for viewpoint based on the initial match.

The operations described are far too expensive (a few minutes per $\sim$ 20 line model) to perform in a serial fashion, model by model. Lowe suggests that the ranking mechanism employed to aid the search be used to control the distribution of processing to a parallel processing system.

A related approach is the work of Goad [45] which also involves matching between edges in a 3D model and edges in the image to constrain the viewpoint. However, Goad produced high runtime speed by "compiling" the object model to produce a fast program. This involves precalculating, for each position and orientation [1] of each model line, the relative position in the image of every other model line. In addition, a depth-first search tree is also precompiled and optimised so that the edges most likely to be present are examined first.

At runtime, Goad's system predicts an initial match between an image and model edge and then uses the search tree to predict the position of other edges in the image relative to the first edge. As model edges are matched to image edges the position of the image edge is used (back-projected) to refine the current estimate of the camera position (and therefore the object position).

Exact viewpoint determination for images of known polyhedra has also been demonstrated by Yu et al. [114]. Although an unknown view of a known triangle can only be solved to within twelve possible viewpoints, two adjoining triangles with a known angle between their normals yield a single solution. The method is only demonstrated with triangle pairs but generalises to polyhedra since any polyhedral object can be decomposed into a collection of triangles. Like all precise geometric solutions, it is

[1] More precisely, the model is assumed fixed in space and edges are viewed from 218 different positions on the viewing sphere.

not clear whether lines extracted from images will be accurate enough to achieve a consistent solution.

A completely new approach has recently been demonstrated by Terzopoulos, Witkin and Kass [100]. They have moved away from thinking of models as being rigid and trying to match them to image data by simply scaling and re-orientating them in 3D. Instead their models can be though of as being made of an elastic material which can be deformed. From an initial guess the model is acted upon by external forces which are derived from the image. These forces deform the model so that its projection into the image plane is consistent with the object.

The models presented have a natural "preference" for axial symmetry and are initialised by a user manually placing a spine on the image. The model starts off as a thin tube around the spine and then "inflates" to fit the object. The forces which inflate the model are defined in terms of the intensity gradients in the image, such that they pull the edges of the tube out towards occluding contours in the image.

The examples given are quite simple but the general approach of deformable models is quite appealing because it allows the modelling of objects with quite a complex geometry (eg. the potato and squash demonstrated) without explicitly stating their geometry in the model.

6.2 Viewer-centred Approaches to 3D Modelling

The motivation behind a viewer-centred, essentially 2D, representation of 3D objects is the ability of the human visual system to recognise objects from simple 2D line drawings. Since there is no direct 3D information (in the form of shading or texture) in a line drawing we must be able to infer 3D information from properties of the lines themselves. Barrow and Tennenbaum [9], Kanade [53] and later Barnard and Pentland [7] have all investigated ways of deriving 3D shape from single line drawings. Other researchers believe that such 3D inferences are achieved using high-level knowledge and that initial object recognition is achieved by comparing the image to a number of

stored representative views of the object. Experiments performed by Koenderink and van Doorn [55] support this view and they suggested that the human visual system uses some enumeration of all the fundamentally different views for object recognition.

The basis of viewer-centred 3D modelling is therefore that we represent objects in terms of 2D image features rather than 3D object features, using a set of different views, or aspects, in order to describe the object fully. In an object-centred system we either project a 3D model into a 2D image and match 2D features or extract 3D information from the image and match with 3D model features: hence, by storing a set of 2D features as our model and matching directly with 2D image features, a viewer-centred approach avoids the difficult problems inherent in the 2D$\leftrightarrow$3D transformations.

Early work by Underwood and Coates [103] yielded a system which could *learn* graphical descriptions of convex polyhedra when shown a series of views and then recognise objects from unknown views. Faces are represented by simple geometric invariants so that a view can simply be represented by a graph in which the nodes represent the faces and the arcs represent face adjacency. As the system is shown a series of views (which contain each face at least once) it builds up a representation of which faces are connected to which. Given an unknown view, the system simply generates the representative graph and compares it to its learned representation of the object.

Thorpe and Shafer [101] analysed the topological changes in line drawings of trihedral polyhedra with changing viewpoint. They developed a set of transition tables which show the allowable changes in junction appearance as the viewpoint moves. Using this information they demonstrate an algorithm which tracks vertices from one view to the next. Although restricted to simple trihedral polyhedra this work is important because it leads the way to generating all the distinct views of an object.

Wang and Freeman [106] have built on this and the work by Chakravarty and Freeman [24] on the recognition of planar-faced solids (PFS) using characteristic views. Given a PFS there are an infinite number of views. These views can, however, be partitioned into a finite number of regions called *characteristic view domains* (CVDs)

where views in the same domain are topologically equivalent and views in adjacent domains are topologically distinct.

Each model is represented as a graph of all its CVDs where branches correspond to the number of visible faces, the numbers of visible vertices and the various junction types. The CVs themselves are represented as graphs with nodes storing junction types and arcs representing the line segments. This representation reduces the number of graph matching operations that are required but means that occlusion cannot be handled.

Since all possible viewing positions are considered, there are a very large number of CVDs for even a simple object (eg. solid "L" has 37). Many of these views are due to very close positions and should really be discounted since they are (i) unlikely and (ii) not readily identifiable by a human being. In order to try and provide better model indexing, Chen and Freeman [27] recently defined a subset of the CVDs (called the dominant views, DVs) which encompasses maximal visual information about the solid. This basically means that the DVs are the minimum set of views in which each surface appears at least once.

The concept of multiple views has also been suggested by Fekete and Davis [38] and is embodied in their *property sphere* representation. A property sphere is basically a viewing sphere which is sampled in some regular fashion (Fekete and Davis use triangular subdivision of an icosahedron to give 320 viewpoints) and at each viewing point considered some object properties viewable from that point are stored. Fekete and Davis use two properties per viewpoint, the first and second moments of the silhouette. Given an unknown silhouette the property sphere is searched for viewpoints which give similar property values to the unknown.

The idea of property spheres was extended by Korn and Dyer [56] to use more qualitative properties which do not vary continuously across the view sphere. This means that viewpoints on the sphere can be grouped together into regions (like CVDs) and Korn and Dyer define a region growing process on the property sphere which can achieve this. As an example, the property of visible faces of a cube is used to produce

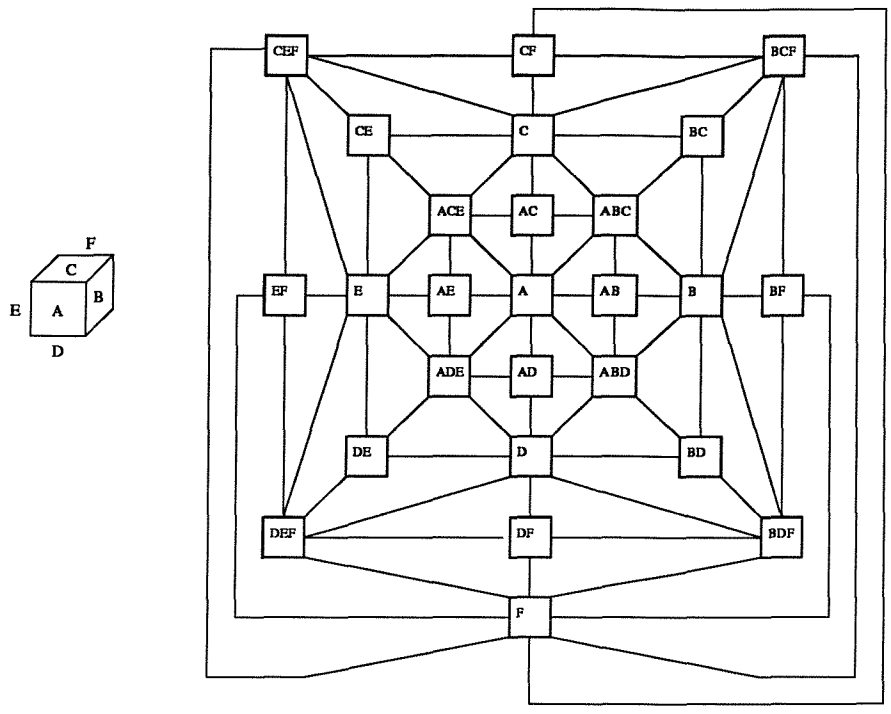


Figure 6.3: Visible-Face Aspect Graph of a Cube

a "segmented" property sphere which can be represented as an *aspect graph*. Figure 6.3 shows the derived graph, each node represents an area of the property sphere over which the given faces are visible and each arc represents a boundary between two adjacent areas.

Using this approach the number of different views is effectively reduced from 320 to just 26, although the number of views is obviously dependent on the object. We have traded accuracy of viewpoint determination (now a large region instead of a small region which approximated a point) for compactness of representation.

Plantigna and Dyer [87] investigated the aspect graphs of polyhedral objects under orthographic projection and perspective projection. By listing the ways in which aspect can change they show how to calculate the boundaries of visibility for each face of the object. They call the areas within these boundaries the regions of constant aspect and these form the nodes of the aspect graph. For a non-convex polyhedron with n faces they show that the upper bound on the size of the aspect graph is $O(n^6)$ under orthographic projection and $O(n^9)$ under perspective projection. This is extremely

large and they conclude that for general objects it would be necessary to construct some approximation of the aspect graph which contained the most important changes in visibility.

A different analysis of the same problem by Gigus and Malik [43] produced the same predictions for the upper bound although they claim that their algorithm would be faster in the average case. Another theoretical paper by Sripradisvarakul and Jain [93] carries out an analysis of the additional visual events which may occur with curved objects.

Some real systems using aspect graphs have been implemented. Bowyer et al. [20] have produced a system which attempts to identify convex polyhedra and give some estimate of their position and orientation. In their formulation, each node of the aspect graph has the following attributes,

- A definition of the corresponding 3D cell of viewing space;
- A definition of which faces are visible from that cell;
- The coordinates of a "central viewpoint" in the cell.

After processing the image to obtain a line drawing of the object, the system calculates the Fourier Descriptors (FDs) of all the unique closed loops (those which correspond to faces of the polyhedron. Bowyer et al. then use an optimisation technique to match the observed set of faces to aspect graphs which contain the same number of faces.

In their experiments, Bowyer et al. report only 3 classification errors from a set of 100 randomly generated views (25 views of each of 4 objects). They state that algorithms are required for non-convex polyhedra and for curved objects. Given that the aspect graphs tend to be large, they also state that some concept of node equivalence will be required to reduce the size of the representation.

Dickinson, Pentland and Rosenfeld [34, 33] have recently moved even further away from accurate viewpoint determination in their *qualitative 3D shape reconstruction* system. No attempt is made to find the exact pose of the object in 3D; it is assumed

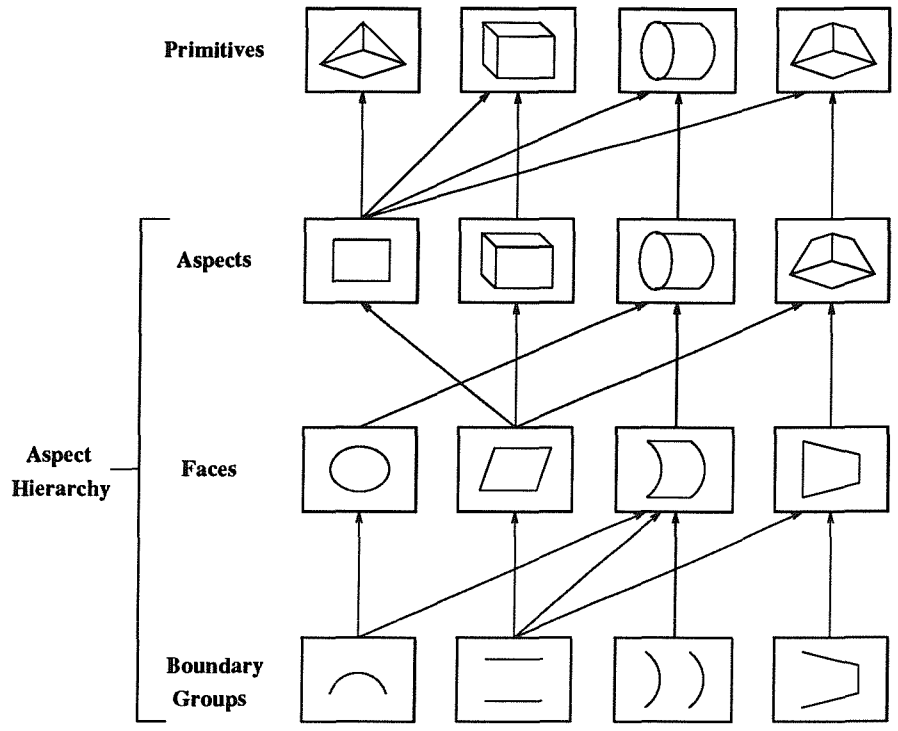


Figure 6.4: A Subset of the Aspect Hierarchy

that when an object is recognised in the image its pose can then be calculated by other means.

Dickinson et al. also state that the general approach of aspect graphs would require a prohibitively large number of aspects to model a large database of objects. In order to overcome this problem, objects are first decomposed into 3D volumetric primitives and then these primitives are represented by aspects.

The advantage of this approach is that, once a suitable set of primitives has been selected to model all the objects required, the number of aspects required to model *all* objects is then fixed. Since it is highly unlikely that a given primitive will always appear unoccluded in the scene, the aspects for each primitive are defined in a hierarchical manner as illustrated in figure 6.4. This sample of the hierarchy shows that aspects are constructed from faces which in turn are constructed from boundary groups. If an aspect is occluded then it may still be identifiable from the faces that are present; if a face is occluded then it may still be identifiable from the boundary groups that

are present. Thus, the problem of occlusion is tackled, by necessity, head on; most aspect, or multiview, approaches so far have ignored the problem of object occlusion completely.

Using a set of 10 volumetric primitives Dickinson et al. produce 648 views per primitive but then they exploit the symmetry of the primitives to throw away equivalent views (in terms of faces and boundary groups) and reduce the total number of views for all 10 primitives to 688. This is actually more like Chakravarty and Freeman's [24] characteristic views than a traditional aspect graph since we are throwing away the *visual events* and just keeping the unique aspects (nodes). As an example, the block primitive would have an aspect graph like that shown in figure 6.3 but only three of the nodes are qualitatively distinct, therefore the whole graph is reduced to a set of three aspects ie. one, two or three faces showing.

Associated with each link in the aspect hierarchy is a conditional probability which is an estimate of the likelihood that one item is a child of the other eg. the probability that we have a cylinder given that we have found a rectangle. These probabilities are generated from all the views of all the primitives and assumes that all primitives occur with the same frequency and that all orientations of the primitives are also equally likely. This clearly produces only an estimate of the true probabilities since these assumptions are not likely to be true.

Given an aspect hierarchy and a set of objects modelled by a connection of primitives, the recognition of objects in an image is a matter of,

1. Extracting a set of contours from the image;
2. Grouping contours into faces;
3. Classifying faces;
4. Grouping classified faces into aspects;
5. Matching aspects to primitives;
6. Grouping apparently connected primitives into object parts;
7. Matching parts to objects.

The conditional probabilities are used to constrain the classification, grouping and matching processes in all but the last step. The matching of parts to objects is done using a hash table to avoid a linear search through the whole of the object database: unfortunately this means that if some of the primitives are unrecoverable then the parts constructed will not index to the correct model (if any).

The approach is demonstrated on images of objects which are conveniently constructed of two or three of the primitives with limited occlusion.

In response to the increasing interest in techniques involving *aspect graphs*, in one guise or another, Bowyer [19] set up a workshop to examine the practicalities of the approach. The technique is criticised by Faugeras, Mundy and Ahuja but defended by Dyer, Pentland, Jain and Ikeuchi. The major criticisms of the aspect graph approach are,

- Computing the aspect graph has a high complexity and much of the work to date has been theoretical or limited to simple objects;
- The number of aspects is generally very large so matching an unknown view to one of the aspects produces a very difficult indexing problem;
- Many of the approaches rely on edges and junctions but these often cannot be recovered reliably from images;
- Most approaches do not return the 3D pose of the object;
- Most approaches produce impoverished representations with topological data only and no grey-level or relative size information.

In response, the following defences are fielded,

- Aspects based on edges and vertices were an important first step, now researchers are beginning to redefine the aspect graph in terms of other features which are more detectable and result in smaller numbers of aspects;
- Complexity and size can be reduced by applying the aspect graph approach to the component parts of the object rather than the object as a whole;

- Representations can be made richer by moving from edge-based aspects to face-based aspects;
- Generation of aspect graphs is done off-line so it does not matter if it is slow. What is important is that recognition is fast, aspect graphs have potential to achieve this because we are matching 2D views to 2D images, unlike traditional object-centred approaches which attempt to match 3D models to 2D images at runtime;
- Using recoverable features that are also effective in indexing will also result in features that are accurate in pose recovery.

The conclusions to be drawn from the workshop appear to be that the early research into aspect graph approaches has shown the potential and some of the problems. Current and future work will determine whether the full potential can be realised or not.

6.3 Recognition by Components

With the notable exception of Dickinson et al. [33], most of the approaches to object recognition considered so far have attempted to model objects as single indivisible entities. However, there is evidence that suggests it may be beneficial to represent objects as a collection of connected components.

Biederman [15, 16] has claimed that all objects recognisable from line drawings can be encoded using a set of only 36 generalised cone components (geons). This limited number arises because humans (in primary recognition) only use simple qualitative measures such as "straight" and "curved". He also proposes that object recognition is a matter of recognising these geons and then relating the componential description to complete objects. (Distinction is made between *count nouns* such as chair and table which are recognised on the basis of shape and *mass nouns* such as sky and water that are recognised on the basis of colour and texture.) The key phrase behind this

approach is "Although objects can be highly complex and irregular, the units by which objects are identified are simple and regular."

An interesting question is whether geon identification is 2D or 3D. Biederman states that although all 36 geons have a clear subjective volumetric identification they can often be uniquely specified from their 2D image properties. Consequently, recognition need not follow the construction of an object-centred 3D interpretation of each volume. The possibility is also noted that "despite the subjective componential interpretation given to the arrangement of image features as simple volumes, it is the image features themselves, in specified relationships, that mediate perception."

On the total number of distinguishable objects, Biederman estimates (liberally) that adult humans can distinguish 3,000 basic-level categories with an average of 10 varieties per category ie. a total of 30,000 readily discriminable objects. Given that the average number of objects available from two geons is $36 \times 36 \times 57.6 = 74,649$ (where 57.6 is the average number of ways that two geons can be connected) there are easily enough representations available and object space is sparse enough to enable most objects to be recognised on the basis of only a few geons.

The subject of unusual views is also discussed and examples are given. It is suggested that primary recognition cannot identify objects seen from unfamiliar viewpoints - in these cases recognition is achieved only by resorting to scene context or guesswork. It is therefore reasonable to assume that humans only store a limited description of objects and rely heavily on context.

Experimental systems for extracting 3D *parts* from range images have been demonstrated by Bajcsy and Solina [2] and Pentland [82, 83]. In each case, the parts are approximated by deformations of superquadrics. Superquadrics are a family of 3D shapes which are defined parametrically as

$$\boldsymbol{X}(\eta,\omega) = \begin{pmatrix} \cos^{\epsilon_1}(\eta)\cos^{\epsilon_2}(\omega) \\ \cos^{\epsilon_1}(\eta)\sin^{\epsilon_2}(\omega) \\ \sin^{\epsilon_1}(\eta) \end{pmatrix}$$

The basic shape is determined by the two parameters ϵ_1 and ϵ_2 (hence quadric), for example $\epsilon_1 = \epsilon_2 = 1$ gives an ellipsoid. All the basic primitive shapes commonly used for object modelling can be produced eg. blocks, cylinders, spheres, diamonds and pyramids. These basic shapes are then deformed by stretching, bending, twisting or tapering.

Pentland [85] has also demonstrated that simple superquadrics can be recovered from chromatic images using 3D shape from shading techniques. However, these techniques are notoriously unreliable when applied to real images due to complex illumination effects and the variable texture of surfaces.

Since extraction of parts has not been shown to be simple from 3D data it would appear that the 2D aspect approach due to Dickinson, Pentland and Rosenfeld [34, 33] (described in the previous section) is the best way forward.

6.4 Conclusions

Of the large number of 3D object recognition schemes reported in the literature the majority have been object-centred. This is largely a consequence of the development of CAD and graphics systems, where the models are used to generate pictures of the object from any desired viewpoint. The use of such models in the inverse problem (ie. pictures to models) has not been as successful!

A major failing in simple object-centred modelling systems is their inability to carry surface information such as colour and texture. Indeed, in some schemes it is difficult enough to work out where the surfaces are let alone what specific properties they have.

The fact that an image is two-dimensional and an object-centred model is three-dimensional means that at some stage some transformation has to be made. Either the 3D model is projected into the image or 3D information is derived from the image; there are significant problems with both approaches (i) there are an infinite number of ways to project an object into an image and a 3D object has seven degrees of freedom (ii) deriving 3D information from an image can only be achieved using assumptions

about the scene and is often unreliable.

Due to the complexities of the 2D↔3D transformation most systems using object-centred models have been limited to objects which were mainly polyhedral or cylindrical. Additionally, the exact geometry of the object must be known in order to construct the model; although, Martin and Aggarwal [69] have shown that for a large class of objects an object-centred representation can be derived from an object's occluding contours in a number of views if the viewpoints are known. The recent work of Terzopoulos et al. [100] with deformable models has addressed both problems by creating models which can assume a complex geometry as a result of forces derived from the image: problems still remain in the selection and initialisation of such models but the basic idea has great potential.

Given the problems of matching 3D object-centred models to 2D images, it is not surprising that there is increasing interest in viewer-centred models. Clearly, the main advantage of a viewer-centred modelling system is that the model is a description of the object which is similar to the image, both are two-dimensional and therefore no 2D↔3D transformations are required. The drawback is, that to capture the three-dimensionality of an object we need to store a potentially very large number of 2D views.

The earlier viewer-centred approaches based on lines and junctions were as impoverished as the early object-centred models and often required hundreds of views. Much more promising are face-based approaches which provide richer descriptions, may require fewer stored views and are more reliably recovered from images.

Aspect graphs have received a lot of attention because they not only contain the views of an object but how the view changes as the viewpoint changes. This information could be very useful in an active vision scenario. For purely passive vision systems Dickinson et al. [33] quite rightly point out that the visual events are of no consequence and that the size of the representation can quite drastically be reduced by omitting them and retaining only the distinct *aspects*.

Another feature of the Dickinson et al. approach is that they describe objects as

a collection of parts rather than as complete, indivisible, entities. This builds on Biederman's [16] theory of recognition by components (RBC) in which all objects are described by only 36 different primitives. The appeal of this approach is obviously that the aspects of the primitives can be calculated initially and then no more aspects are required: this means that the total number of aspects required to model all objects is fixed and much smaller than the number of aspects which would be required to model whole objects.

Where RBC approaches fall down is in situations where the objects cannot be conveniently decomposed into the chosen primitives. What is required is a more general approach in which objects can be described in a compact viewer-centred manner, using decomposition into primitives only when appropriate. This is the approach taken in the following chapter.

7 New Aspects of Aspect Graphs

In chapter 6 we saw a myriad of techniques for 3D object recognition and came to the conclusion that the most promising approaches were the viewer-centred methods using *Aspect Graphs*.

In this chapter I will show how the colour and shape analysis techniques described in chapters 3 and 5 can be combined with the principle of *aspects* to achieve object recognition. We shall describe each view of an object, or sub-component, in terms of the shapes of the 2D image projections of its visible surfaces. This will allow us to easily attach colour information to our representation where it may be useful. The ideas presented have been partially implemented to demonstrate the feasibility of the approach.

7.1 Shape-based Aspects

Initially we will not concern ourselves about the type of objects that we are trying to represent ie. whether they are complete objects or components of larger objects. For the time being an *object* is simply some three-dimensional volume.

For our purposes we define an aspect of an object to be a unique view of the object in terms of the shapes present. Here, by shape we mean the 2D projection of a surface. With this definition we get just three distinct aspects for a simple cubic block object, as illustrated in figure 7.1. In our aspect graph the nodes represent the constituent shapes and the arcs represent adjacency of shapes. In the block example all the nodes represent the same shape ie. a perspective projection of a square.

We can immediately see that this representation is far more compact than the "full"

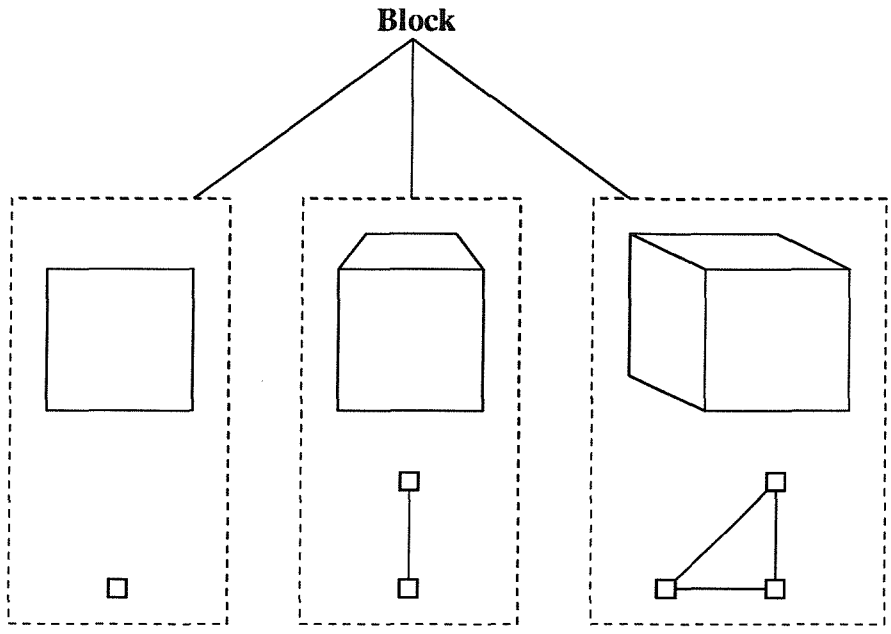


Figure 7.1: Shape-based Aspects of a Block

aspect graph for the cube that we saw in figure 6.3. What we have thrown away is the information contained in the arcs of figure 6.3 ie. the visual events which indicate how the views change as our viewpoint moves around the object. In an active vision system the visual events could be very useful but we are concerned with recognising an object from a single view so the visual events are of no use to us since we will not experience any.

In our scheme each *aspect*, or view, is represented by a graph and the object is represented by a list of aspects, in contrast to the normal aspect graph approach where the object is represented by a graph of aspects. Henceforth I shall use the term *aspect* to refer to a graph which represents a single representative view of an object.

There is no reason why each of the nodes of our aspect should contain only a single attribute (so far just the shape of the surface). We can easily attach additional information to the nodes such as colour, texture or relative size. These features may be single or multi-valued eg. an orange will always be orange but an apple may be red or green or both. We can also attach information to the aspect as a whole to tell us something about the particular view. For objects with strong orientation, eg. a car,

we may label an aspect as viewing from the side/front/rear/above/below to give us information about the 3D pose.

7.2 Extracting Shapes from Images

If we are to match imaged objects to our stored shape-based aspects then we must first be able to extract shapes reliably from an image. In the simplest case, where the shape corresponds to a single region of our segmentation, the identification of the shape merely involves calculating the fractal signature of the region boundary and comparing it to our library of signatures as described in chapter 5. We then assign to the shape a list of all the shape names (in descending order of confidence) that have a match confidence above some lower limit.

7.2.1 Handling Shadows and Specularities

In a real scene, it is likely that aspect shapes will not correspond directly to a single region in the segmentation because of lighting effects such as shadow and specularity or highlight. Since our segmentation method is completely data-driven and designed to over-segment the image, we expect shapes which exhibit such lighting effects to comprise more than one region. An example of this is shown in figure 7.2: the square shape has been over-segmented into three regions, a main part R a shadow S_1 and a specularity S_2. If all of these regions belong to the same, equally coloured, surface then they will have similar colour hue but may differ quite significantly in colour value and chroma. In general a shadow region will have a lower colour value because it is darker and a highlight region will have a larger colour value and a lower chroma because it is brighter and less saturated.

In a situation such as that shown in our example (figure 7.2) it is obvious that we should simply group regions R and S_1 together into one region and examine the shape of the combined region. The highlight region S_2 is irrelevant, unless it is recognised as a shape itself, because it is totally enclosed within region R and thus does not affect

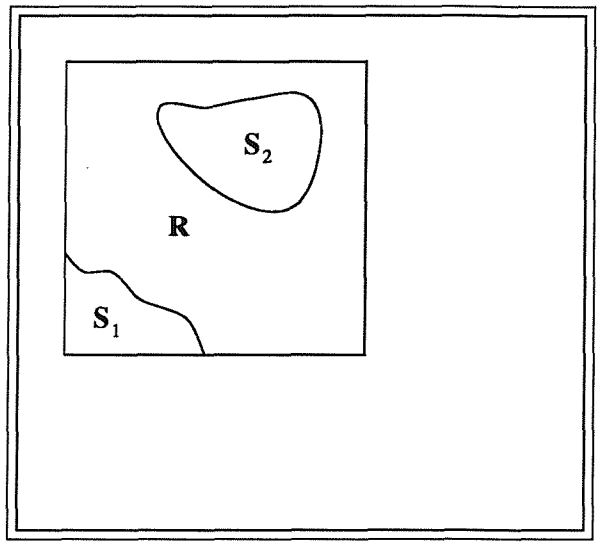


Figure 7.2: Region Group Example

the shape of any combined region by its inclusion or omission. This is often the case with highlights although we did see a counter example in figure 3.4 during our colour image segmentation experiments.

Given the assumptions outlined above on the effects of lighting on the colour of regions, I have used the following algorithm to group regions in a given locality into shapes

1. Produce a Region Adjacency Graph (RAG) in which the nodes represent image regions and arcs connect adjacent regions;
2. Select a seed region R (usually the largest);
3. Find all the connected [1] regions S such that

$$|H_R - H_S| \leq \Delta H_{max} \tag{7.1}$$

$$|V_R - V_S| \leq \Delta V_{max} \tag{7.2}$$

$$|C_R - C_S| \leq \Delta C_{max} \tag{7.3}$$

where (H_R, V_R, C_R) is the average colour of region R and (H_S, V_S, C_S) is the

[1] A region S_i is connected to R if it is adjacent to R or if it is adjacent to another region S_j which is connected to R.

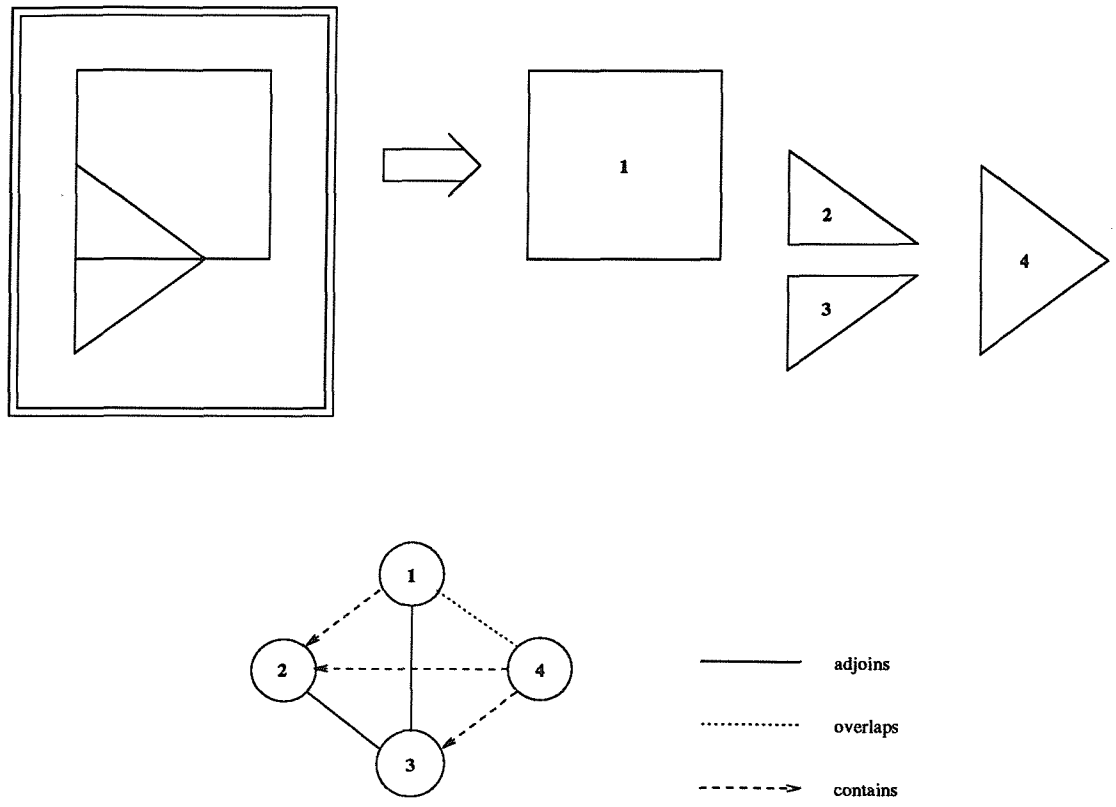


Figure 7.3: Shape Association Graph

average colour of region S as defined in section 3.1.

4. For each of the groupings involving R for which no region is completely enclosed (eg. for figure 7.2 these are $\{ R, R + S_1 \}$) calculate the best shape confidences of all the boundaries;

5. If any confidence $C_i > C_{min}$ then add the boundary to the shapes data-structure (see below).

The constants ΔH_{max}, ΔV_{max} and ΔC_{max} are set to 30°, 50 and 65 respectively. This gives a relatively narrow hue band and large value and chroma bands as required. As with any *magic parameters* the values are somewhat arbitrary, if the range is too small then some regions will be left out but if it is too large then we may group regions which are unrelated. Excessive grouping is less of a problem than failing to group (because unrelated regions are unlikely to produce coherent shapes) so the given values are "generous".

The value of C_{min} is also set on the generous side (at 0.30) so that we record shapes which even vaguely match one of the known shapes. In section 5.4 we saw that very different shapes matched with confidences of the order of 0.10 whereas moderately similar shapes matched with confidences of 0.50 upwards. Hence, a confidence of 0.30 can be thought of as being in the middle of the "grey area".

In the majority of cases we could store the identified shapes in a graph with the nodes representing the shape and the arcs representing adjacency. However, for cases such as that shown in figure 7.3 we need a slightly more complicated data structure. In this case, excluding the background, there are three regions but these are shown to potentially make up four different shapes. Shape 1 is a square, shapes 2 and 3 are 3/4/5 triangles and shape 4 is a 6/5/5 triangle. Clearly, the representation of such an arrangement must make clear which shapes overlap and which are entirely contained within other shapes. The graph in figure 7.3 accomplishes this by joining the nodes, which represent the shapes, by three different types of arc

1. Adjoins (bi-directional) - the two shapes touch;
2. Overlaps (bi-directional) - the two shapes overlap;
3. Contains (uni-directional) - the first shape entirely contains the second.

I call this data structure a *Shape Association Graph* (SAG) since it contains information about the known shapes present in the image and how their appearance is inter-related. The SAG can be thought of as being built upon the underlying *Region Adjacency Graph* (RAG), since each node of the SAG corresponds to one or more nodes of the RAG: this relation will be utilised in later sections.

7.2.2 Handling Occlusion

Another reason why a shape may not correspond directly to a region in our segmentation is that it may be occluded by some other object in the scene (or even by part of the same object). Occlusion is a major problem in object recognition and many researchers state quite openly that their algorithms cannot cope with it. Since occlusion is effectively hiding information from us, we need to employ some higher-

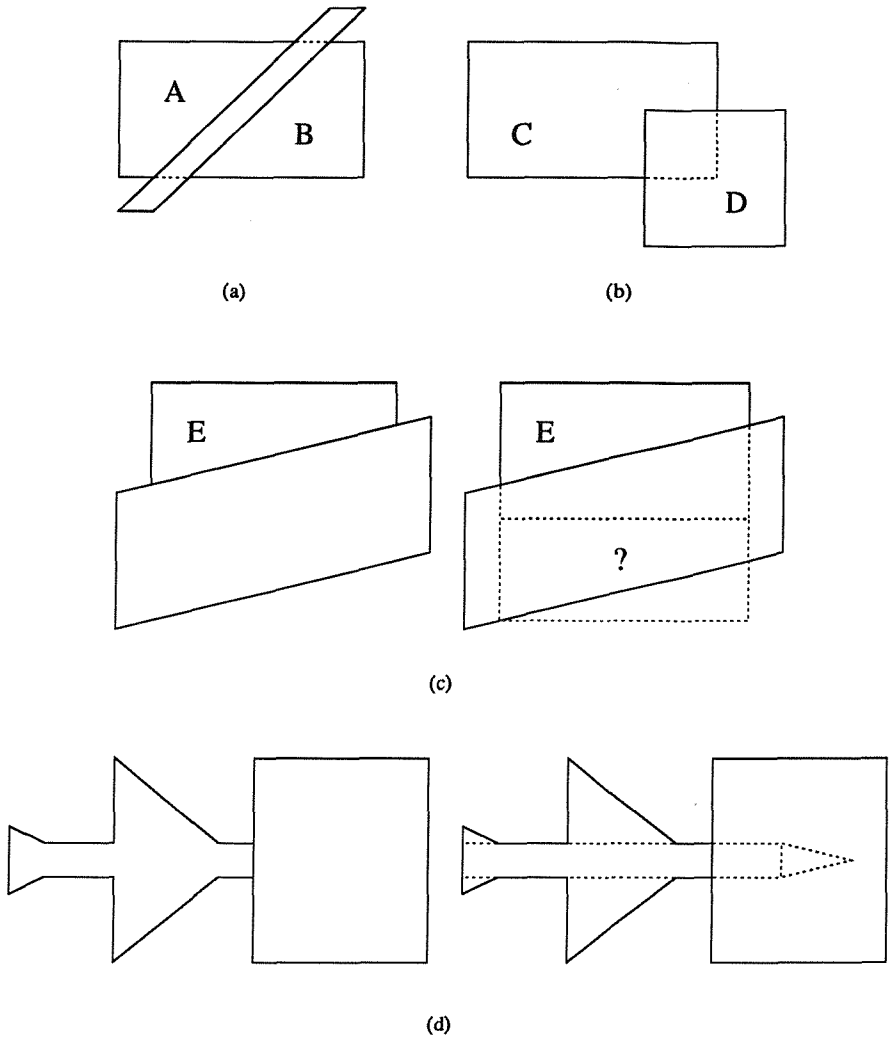


Figure 7.4: Types of Occlusion

level knowledge (or heuristics) to overcome the problem, as we did with the handling of shadows and highlights. In general we rely on the fact that humans tend to interpret occluded shapes so that the simplest possible arrangement results.

Some examples of the different ways in which a shape can be occluded are shown in figure 7.4. In the first example 7.4 (a) if regions A and B have the "same" colour then, in the absence of any other information, a human observer would assume that they were part of the same shape. In this case we can join the two regions by extending the tangents to the boundary at the region intersections (dotted lines). If the resulting boundary can be confidently matched to a library shape then it is very likely that we

have an occluded shape. Similarly, in the second example 7.4 (b), region C can be extended under region D to form a known shape.

The middle example (7.4 (c)) differs from the other two in that a whole side of the shape has been occluded. Hence, it is not immediately obvious, even to a human observer, what the exact shape of region E should be. In conjunction with the previous two examples we would tend to think that E was the same rectangle as AB and C. Interestingly, in isolation there is a stronger suggestion that E is a square, even though there is not room for a square to fit under the occluding shape. Here we are using the appeal of symmetry to produce a "simple" interpretation of the occluded scene.

In the final example (7.4 (d)) we have a case where we can do nothing simple to recover the occluded boundary. You and I can see that we have something like an aircraft under the rectangle because we can see two wings and two tail fins joined by a fuselage. What we are doing is partitioning the shape into sub-shapes which have some functional meaning. In cases like this it is not necessary to determine the occluded boundary since the visible part contains enough information for recognition.

My basic claim is that for a large number of cases it is not necessary to store any partial shape representation (equivalent to the boundary groups used by Dickinson et al. [34, 33]) because the "complete" boundary (or at least the most likely) can be reconstructed using some simple heuristics/knowledge. Such an approach has recently been demonstrated by Shimaya et al. [90].

7.3 Shape-based Aspects Revisited

In section 7.1 we defined an *aspect* as a graph in which the nodes represented a shape and the arcs represented adjacency. For convex objects this approach is satisfactory since one part of the object cannot occlude another.

When we are dealing with non-convex objects we must address the problem of self-occlusion. In figure 7.5 we see two slightly different views of an L-shaped block. The projections of the faces G and H are not the same shape viewed from different angles

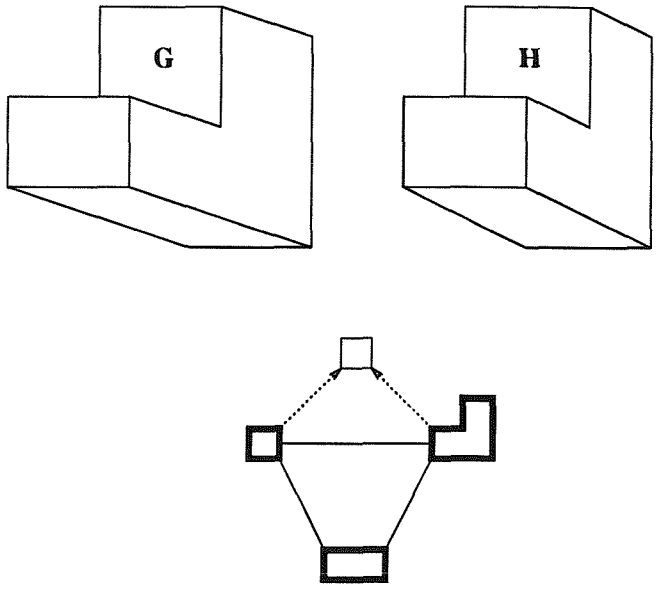


Figure 7.5: Self-Occlusion in Shape-based Aspects

because of the occlusion, even though both views are topologically identical.

Just as we saw that we needed the SAG for representing shapes extracted from images, we also need the SAG to describe the shapes in our aspects. The lower part of figure 7.5 shows the SAG form of the aspect which encompasses both of these views. The bold nodes represent shapes which are normally (in the absence of other occluding objects) fully visible and the non-bold node represents a shape which is occluded by the object itself and therefore cannot be fully visible. The arc types have the same meaning as in the SAG, solid arcs represent adjacency and dotted arcs represent overlaps. Note that, unlike the image extracted SAG, overlap arcs are uni-directional ie. shapes GH are overlapped by other faces but do not overlap those faces.

We now have a slightly more complicated graph structure to represent our aspects although it is still compact, the number of aspects required to fully describe the L-block is as low as 11 (if all the rectangles have the same aspect ratio, see figure 7.6) compared to the 37 used by Wang and Freeman [106].

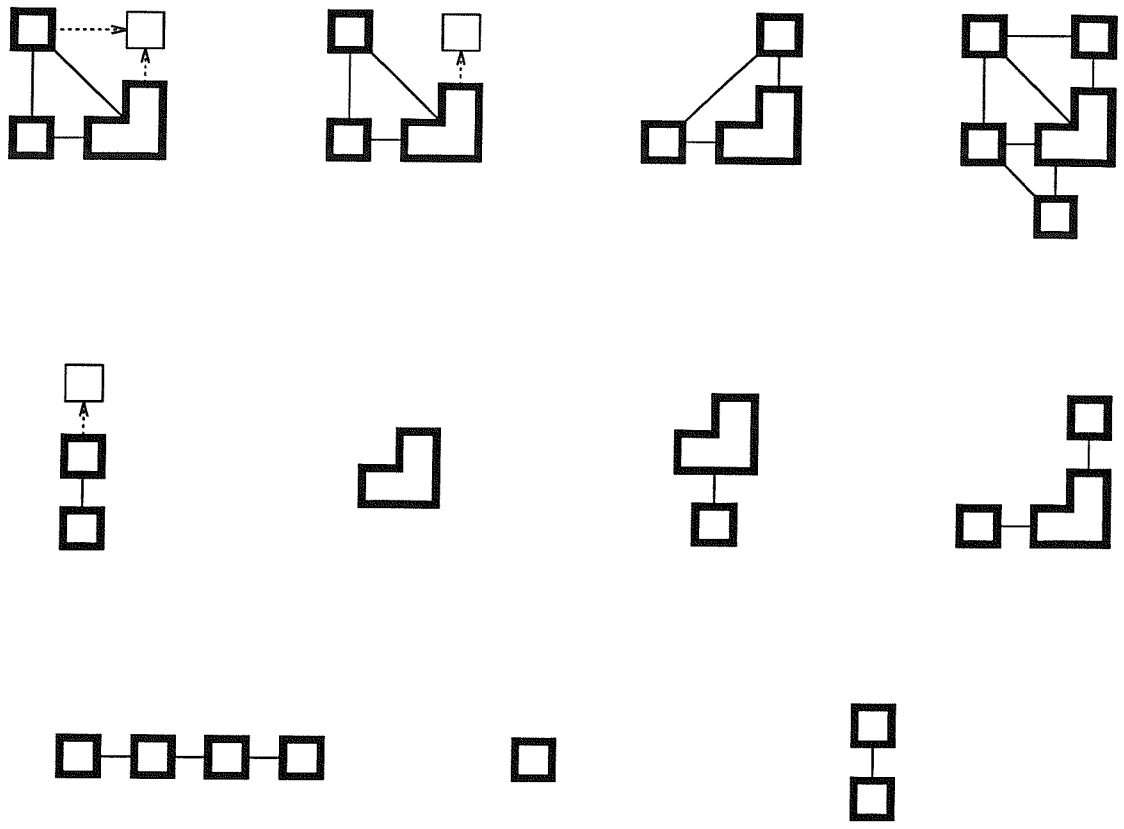


Figure 7.6: All Possible Aspects of a Simple L-Block

7.4 Matching SAGs to Aspects

We now have a scheme in which we can describe views of objects and images containing those views in an almost identical manner.

For simple cases the aspect and the SAG will be identical, as in the example of figure 7.7. This view of an L-shaped block is represented by a simple five shape aspect 7.7(b) with *adjoins* arcs only. Figure 7.7(c) shows the SAG (bold shapes) extracted from an ideal image laid over the corresponding RAG (circles). As in the aspect, all the arcs of the SAG are of the *adjoins* type. To match the SAG to the aspect we only have to prove that the two graphs are isomorphic.

When we have non-convex objects, recognition can be complicated by self-occlusion. Figure 7.8(a) shows a slightly different view of the previous L-block in which one of the faces is occluded by two of the others. The aspect for this view (as discussed in the previous section) is shown in 7.8(b). Figure 7.8(c) shows the SAG laid over the ideal RAG; this time there is not a one to one correspondence between SAG and RAG nodes/arcs so SAG arcs are shown in bold.

In order to match this SAG and aspect we first note that the SAG is isomorphic to the subgraph of the aspect consisting of *adjoins* arcs only. In other words, the recovered faces in the SAG correspond directly to the fully visible faces in the aspect. Given this initial match we can see from the aspect that nodes 1 and 2 are each connected by an *overlaps* arc to node 3. By descending to the RAG we can also see that the corresponding nodes A and B are connected to node C whose shape has not been identified. We can therefore match the SAG to the aspect if we can be convinced that nodes 3 and C represent the same surface: there are several ways that this might be done

1. Blindly assume that since the region C is connected to the correct regions A and B it must represent the occluded shape;
2. Try and complete the occluded shape using high-level knowledge;
3. Compare the colour/texture of region C to that specified in aspect node 3.

The approach chosen will depend to some extent on both the complexity of the

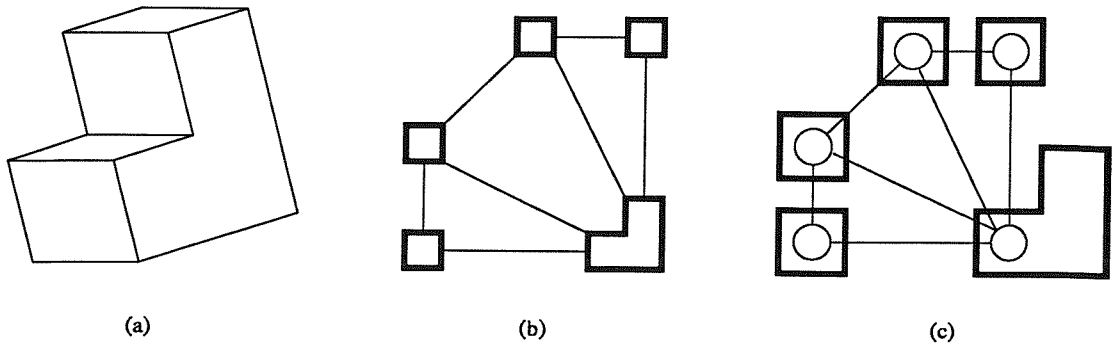


Figure 7.7: An Unoccluded Object, Model and Image Representation

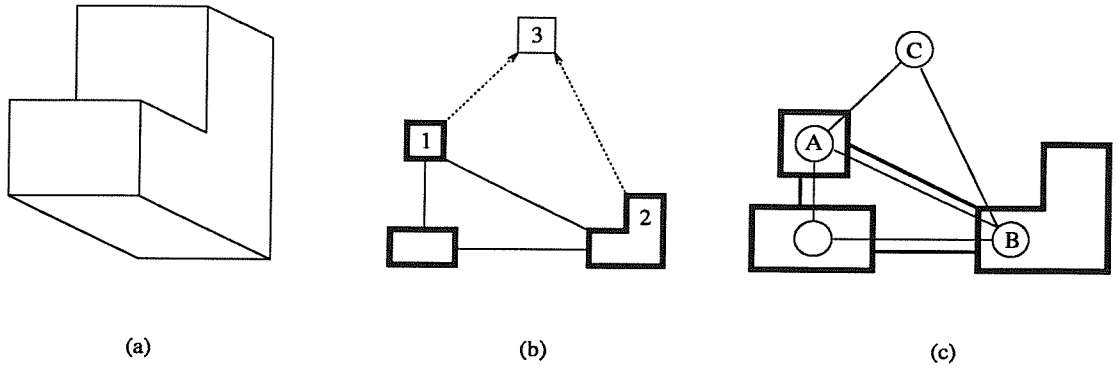


Figure 7.8: A Self-Occluded Object, Model and Image Representation

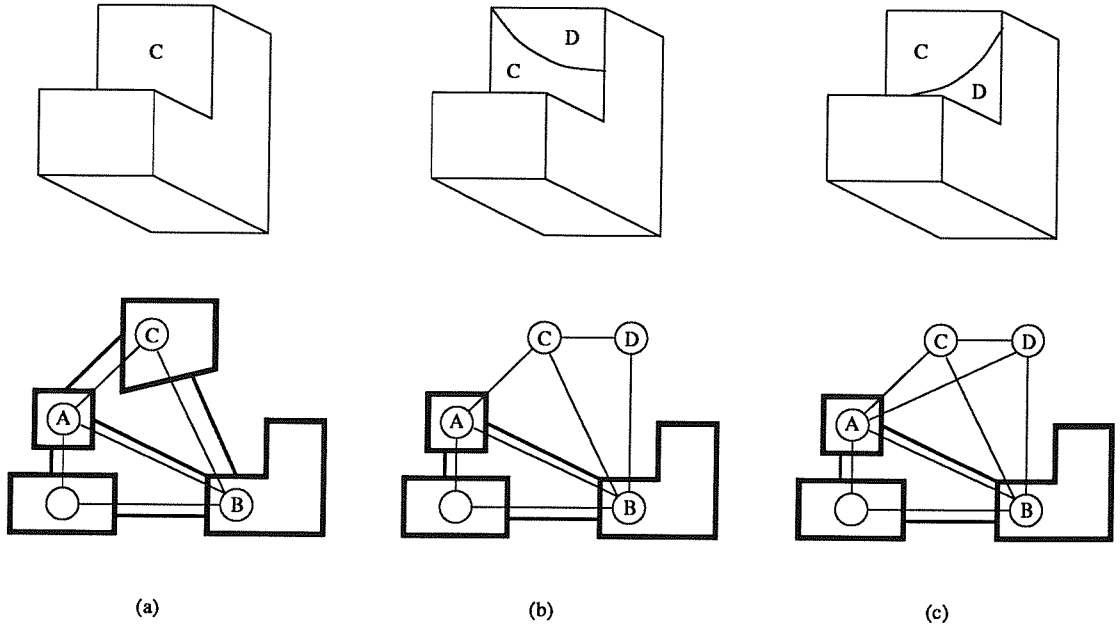


Figure 7.9: Self-Occluded Object, Alternative Image Representations

object and its description. Obviously, colour and texture information can only be used if it is supplied; similarly, shape completion is not generally possible without some "hints" in the description. If the object is sufficiently complex such that many surfaces are positively identified and few remain occluded then the first option may be satisfactory.

Unfortunately, it is likely that the SAG for this object will be more complicated than that illustrated by figure 7.8(c). Three likely alternatives are shown in figure 7.9: (a) region C is identified as some known shape, (b) and (c) the occluded face is divided into two regions.

In the case of 7.9(a) we can immediately match to the aspect if we consider node 3 to be a "wildcard" shape which may match with any known shape satisfying node 3's other criteria (eg. colour/texture). This is a necessary extension because as the number of shapes increases so does the probability that an occluded surface will appear to have a known (but incorrect) shape.

Finally, examples 7.9(b) and 7.9(c) require an extension of the region grouping algorithm discussed in section 7.2.1. Given the initial aspect-SAG match we expect there to be a single region (corresponding to the occluded face) connected to regions A and B; where we find more than one region, ie. C and D, we can attempt to group them using the criteria in section 7.2.1. Once the regions are grouped then we are back to the "simple" situation illustrated in figure 7.8(c).

Clearly, a simple graph-matching algorithm may well be unable to match aspects to SAGs efficiently. What is required to perform these operations is an intelligent graph matcher which can use all the information available to guide the matching process.

7.5 Results

In this section I will demonstrate the feasibility of the approach described above. The mechanisms which are intended for implementation as a knowledge-based system (eg. restoration of occluded shapes, intelligent graph matching) have not as yet been

implemented. The aim here is to demonstrate that shapes can be extracted from real images and that the derived SAGs potentially match object aspects.

The library of shapes used in the following experiments consisted of all the shapes required to describe all the objects as well as some of the simple shapes used in chapter 5 (ie. projection of a square, triangle, arrow). Since there can be several hundred signatures per shape (for a 10^{o} sampling of the viewing sphere) a serial search through all signatures takes some time so the number of shapes has been kept deliberately small. In chapter 8 I will suggest how this restriction may be removed to allow the development of a more practical system.

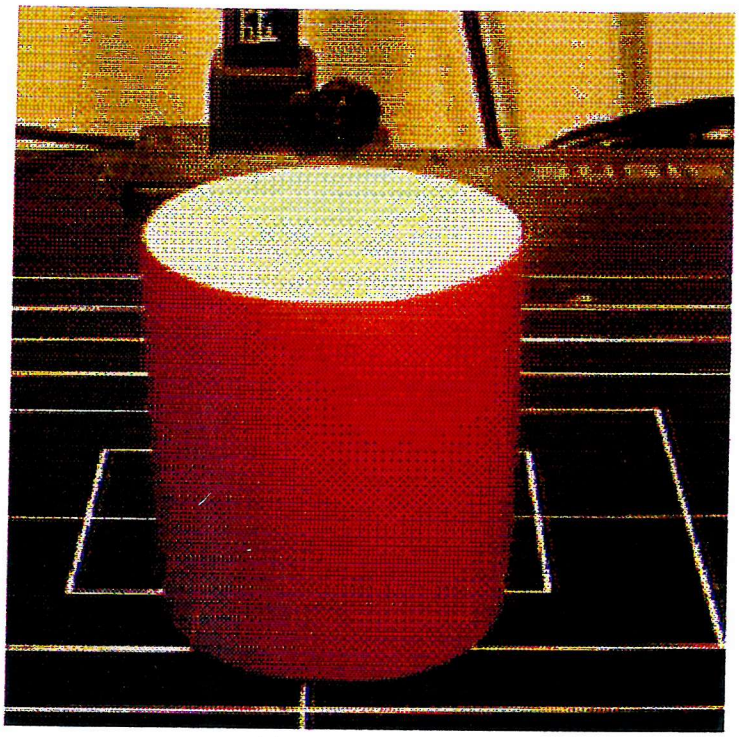
7.5.1 Object1 - cup

The first object chosen was the cup shown in figure 7.10(a). This is a very simple object since it appears as a simple cylinder from most viewpoints. The inside of the cup is all the "same" colour and the outside is almost all the same apart from a dark patch at the bottom-left. This colouring is reflected correctly in the segmentation image 7.10(b).

In the segmentation image it can be seen quite clearly that there are several small dark regions adjoining the large dark region which is part of the side of the cup. The simple region grouping algorithm described in section 7.2.1 examines a large fraction of all the possible combinations of a set of similarly coloured regions; hence, the number of boundaries tested increases almost exponentially with the number of regions. It is therefore highly desirable to reduce the number of regions which may be considered for region grouping. This has been achieved by applying two additional constraints to the region grouping algorithm

1. Regions connected to the edge of the image are ignored since they are unlikely to represent surfaces which are fully contained within the image;
2. Regions whose area is less than 100 are ignored since they are unlikely to make much difference to the shape of a combined region.

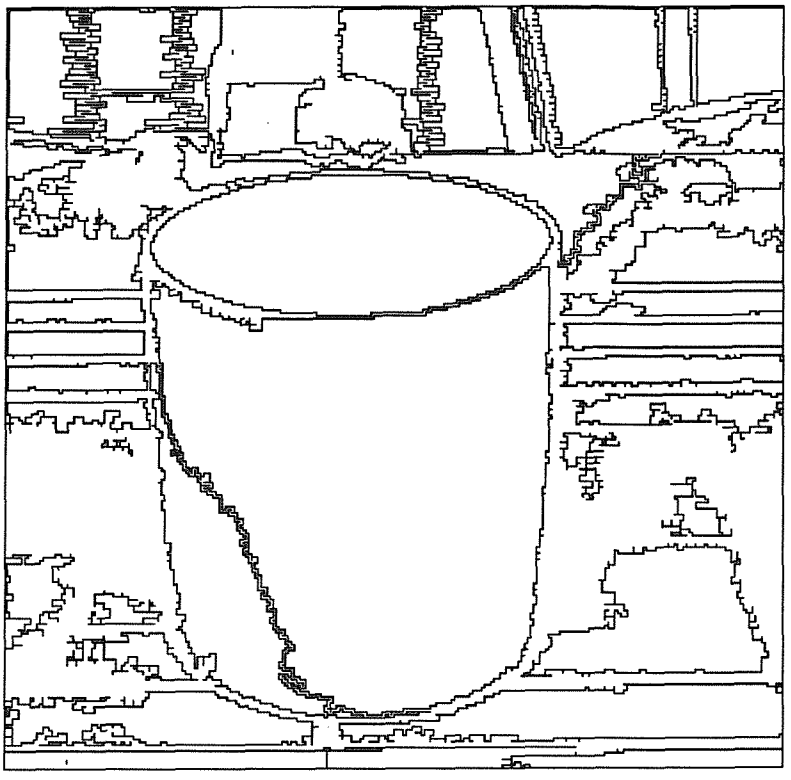
The second constraint is slightly less robust because a large number of small regions



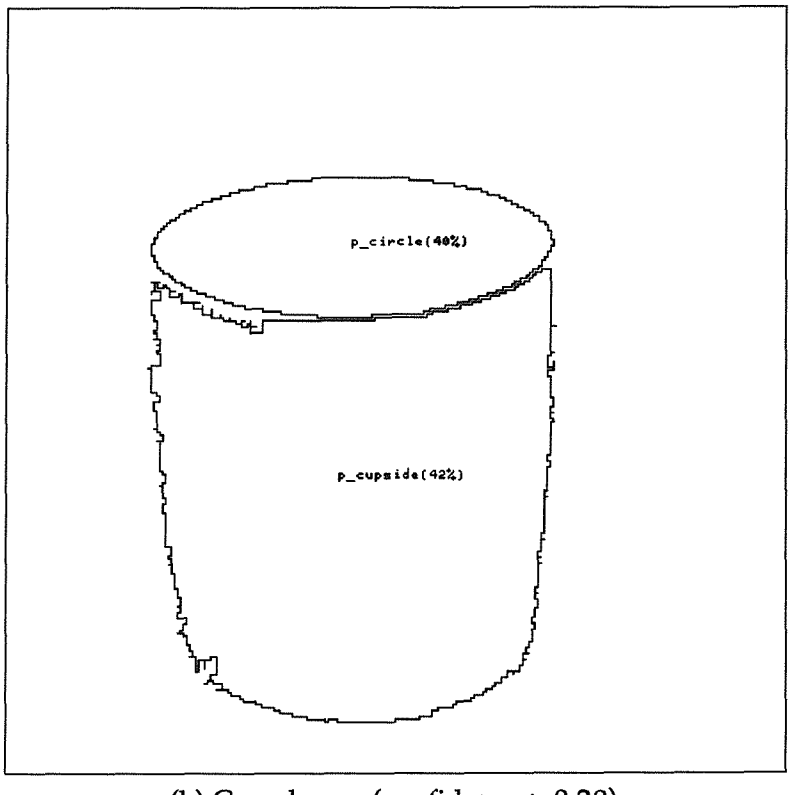
(a) Image of Cup

(b) Segmentation of Cup image

Figure 7.10: Cup



(a) Cup regions (area > 100)



(b) Cup shapes (confidence > 0.30)

Figure 7.11: Cup (cont.)

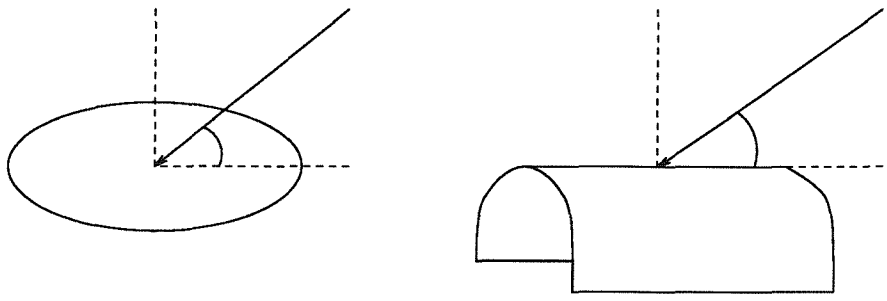


Figure 7.12: Unique Viewpoints for Cylinder Model

taken together would make a significant difference to the shape of a combined region. This problem can only be handled properly by an intelligent system since it is impossible to tell *a priori* whether a group of small regions is significant or not.

Figure 7.11(a) shows the outlines of regions which have an area greater than 100: of the discarded regions most are unwanted but some are part of the cup, obvious examples are the small regions at the top-left of the cup's side and the regions due to the reflection of the white line at the bottom-left of the cup.

7.5.1.1 The Model

The cup was modelled as a simple cylinder with a circular top and straight sides. The two shapes seen in the given view are therefore a flat circle and a curved side, both seen from some unknown viewpoint. The fractal signatures required to describe the shapes were generated by a script which was given the 3D boundary of each surface and the unique viewpoints from which to consider them. Hence the circle and curved side were only viewed from nine different viewpoints ($[0^o, 0^o], [10^o, 0^o], [20^o, 0^o] \ldots [80^o, 0^o]$) since the signature is independent of the viewing angle β because of the symmetry of the object (see figure 7.12).

7.5.1.2 Identified Shapes

The shapes found amongst the regions of figure 7.11 which have a confidence greater than 0.30 are shown in figure 7.11(b). For each particular region group only the most

confident match is shown.

Both shapes have been correctly identified and therefore the SAG and aspect are identical ie. an adjoining circle and cup-side. Both confidences are quite low; the cup-side (0.42) is largely due to the raggedness of the boundary caused by leaving out small regions; the circle (0.40) is partly due to the viewpoint lying in the middle of two sampling points and partly due to the extracted region not being as symmetrical as the modelled shape, this results in a larger signature difference than expected.

7.5.2 Object2 - padlock

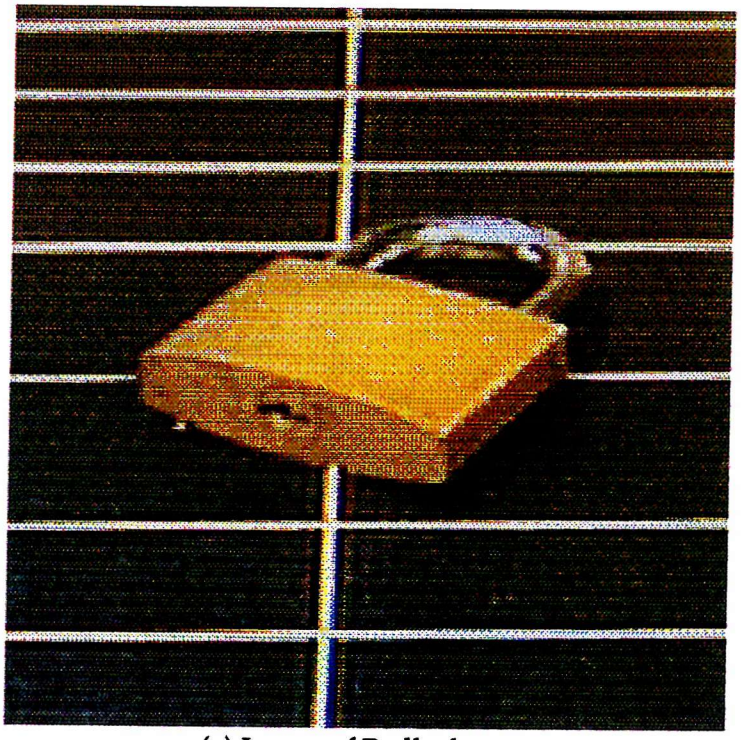
The second object considered is the padlock pictured in two different poses in figures 7.13 (a) and 7.15 (a). This is a more complicated object than the cup since there are seven different surfaces and two distinct parts.

Segmentation images 7.13 (b) and 7.15 (b) clearly show that the shiny loop of the lock is segmented into a large number of regions. Even worse, there are some areas where the loop and background have been merged because there is no visible boundary between them. From these segmentations we can see that we should not expect to identify the loop by simple region grouping: this is a classic example of the interdependence of image layers mentioned in chapter 1, we should first identify the body of the lock and then use other techniques to identify the loop given that we expect to find it joined to the top of the body.

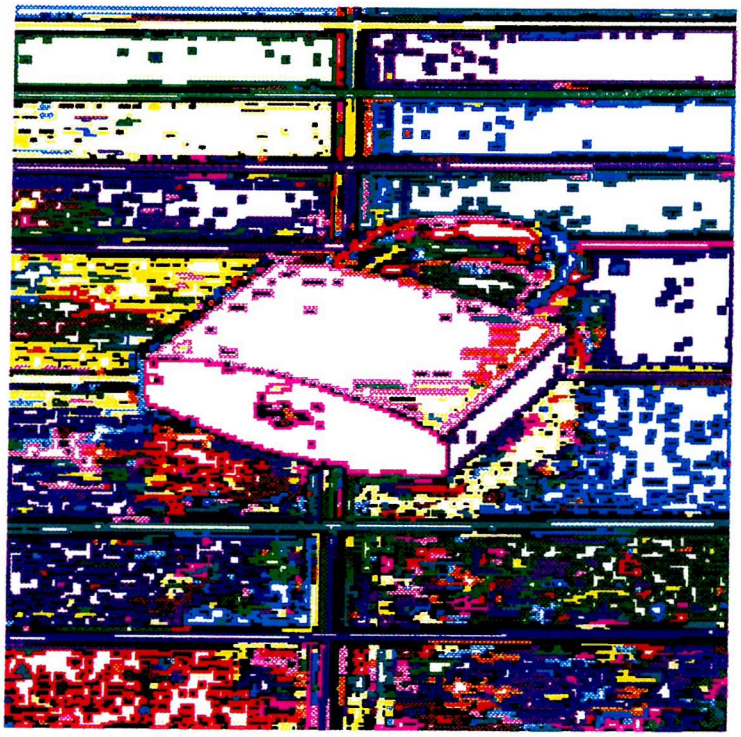
7.5.2.1 The Model

The padlock was modelled as five flat surfaces (base, barrel, edge, chamfer and top) and two curved surfaces (side and loop). The barrel of the lock was simply modelled as a circle since the keyhole was felt to be too small to be recognised by its shape. The other surfaces were physically measured and their signatures calculated in 10^{o} steps as previously described.

The manually generated aspects for the two poses are shown in figures 7.17(b) and

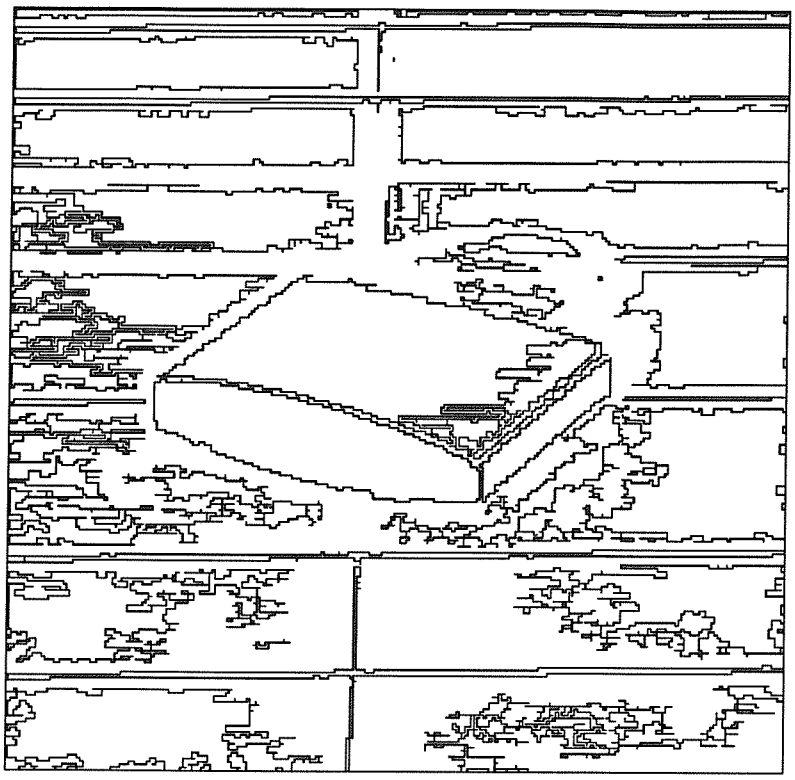


(a) Image of Padlock

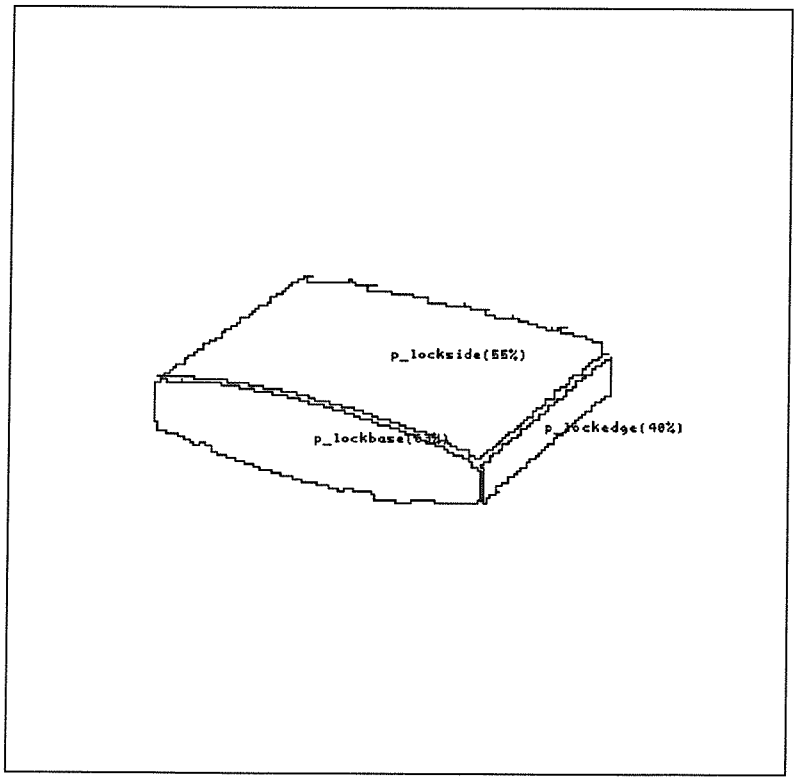


(b) Segmentation of Padlock image

Figure 7.13: Padlock (lower view)

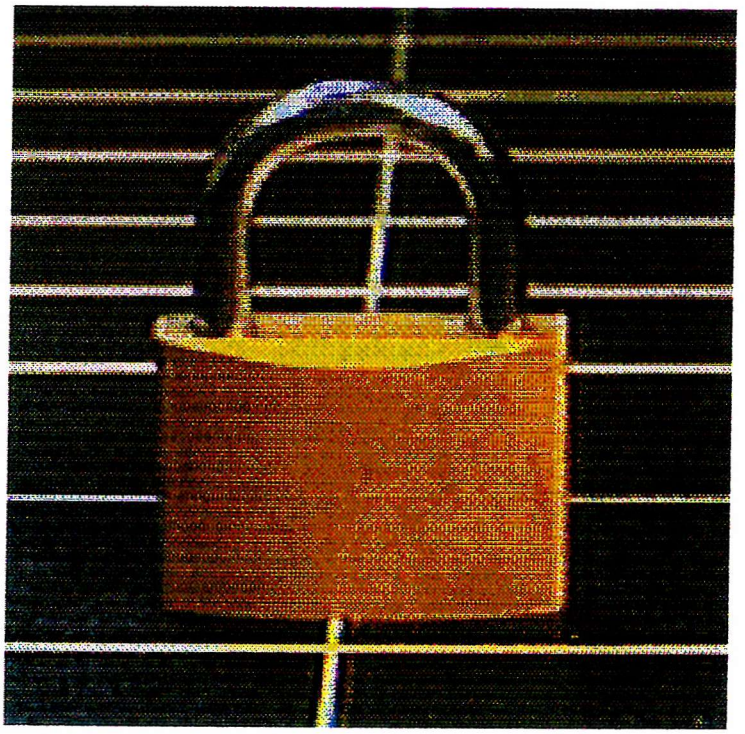


(a) Padlock regions (area > 100)

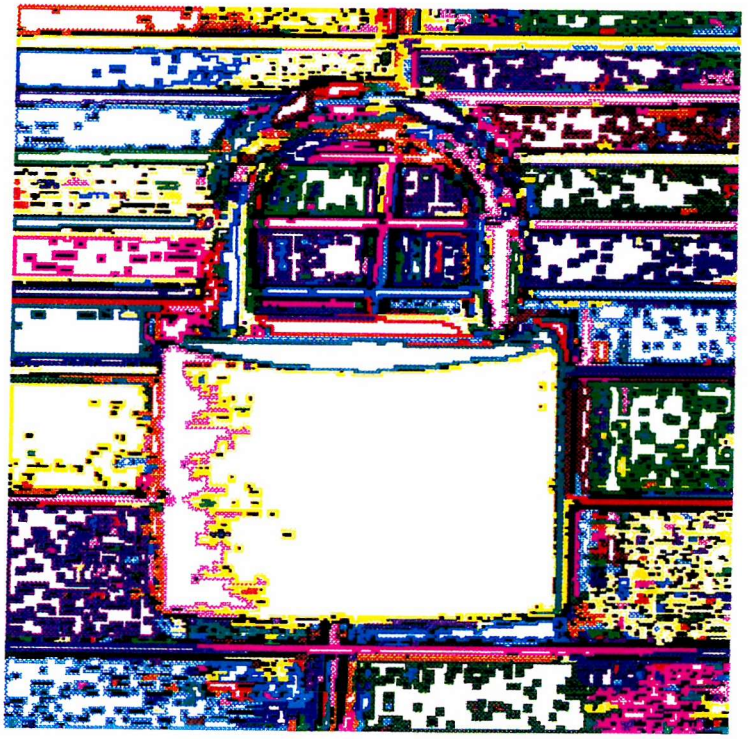


(b) Padlock shapes (confidence > 0.30)

Figure 7.14: Padlock (lower view cont.)

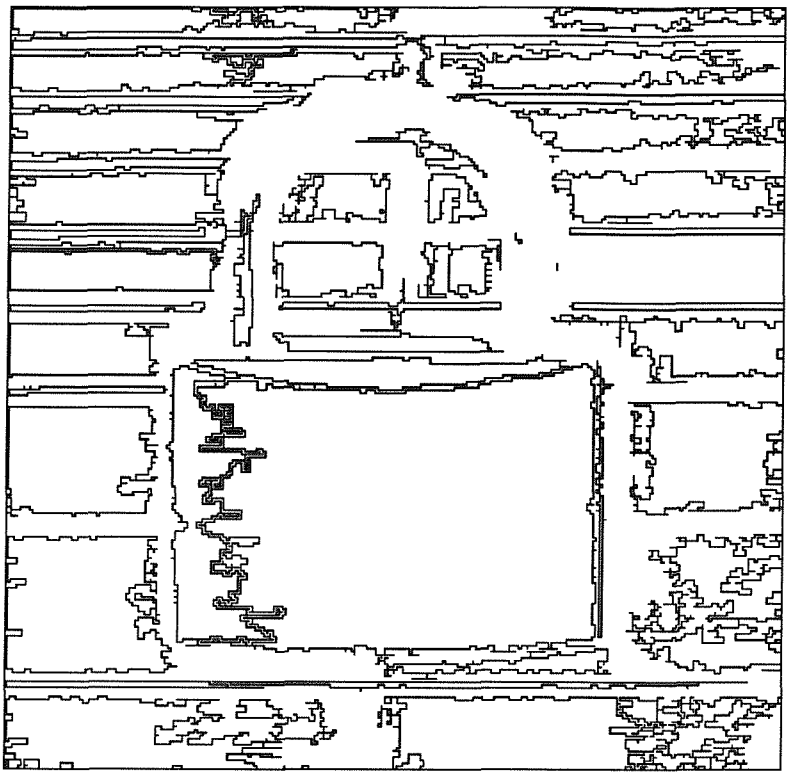


(a) Image of Padlock

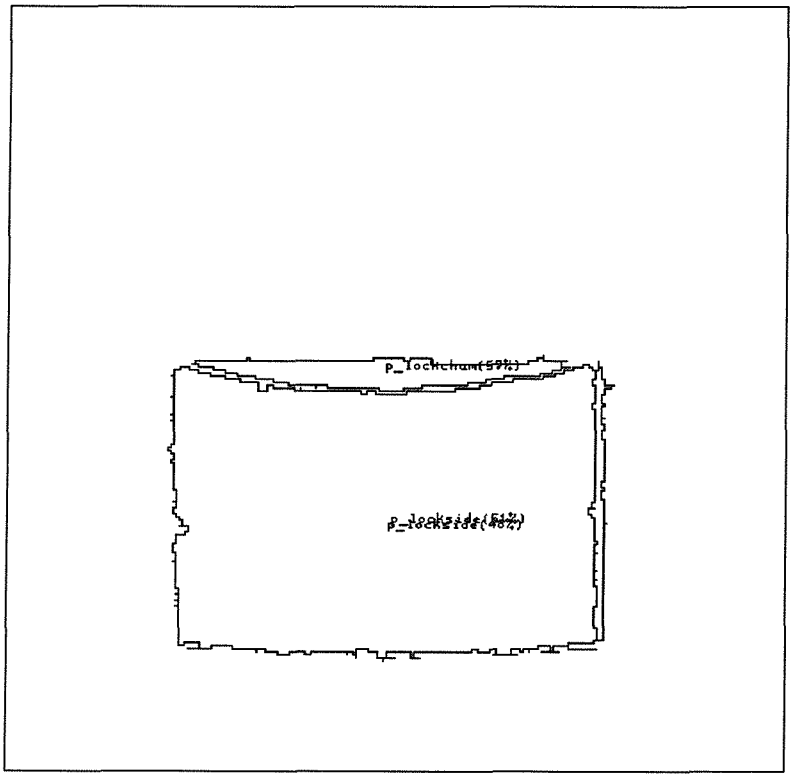


(b) Segmentation of Padlock image

Figure 7.15: Padlock (front view)



(a) Padlock regions (area > 150)



(b) Padlock shapes (confidence > 0.40)

Figure 7.16: Padlock (front view cont.)

7.18(b). Note that the body of the lock and the loop of the lock are divided into separate parts as indicated by the dotted boxes. This is more of a functional decomposition than a structural decomposition; meaning the two parts are naturally separate because, for example, they are made of different materials and can move apart, rather than being separate because their shapes approximate some predefined volumetric primitives. This more natural decomposition has the advantage over the approach of Dickinson et al. [33] that real objects can be modelled more accurately. For instance, the lock used by Dickinson et al. was simply a block connected to a bent cylinder, the base of the lock used here is more complicated and does not conveniently fit any simple primitives.

Additionally, this formulation of the notion of parts allows the connections between parts to be made explicit. Most part-based approaches simply indicate that two parts are adjacent but here we have indicated which nodes of which aspects are connected and how.

7.5.2.2 Identified Shapes

For the first view of the padlock the identified shapes are shown in figure 7.14(b). For the reasons described above, the shiny loop of the lock has not been identified. The three main visible shapes of the body have been correctly identified but the barrel has not because its boundary is not clear in the image and it has been merged with the base.

The shape confidences are reasonably high for the base (0.63) and the side (0.55) but the edge (0.40) is quite low. This is again due to the over simple matching technique since the boundary of the extracted *lockedge* shape is significantly more jagged than the training shape and therefore their signatures have quite a large difference at the smallest scales even though they are very similar for all other scales.

The three extracted shapes form the SAG shown in figure 7.17(a) next to the corresponding aspect 7.17(b). The subgraphs of SAG and aspect which contain only *adjoins* arcs are clearly isomorphic. The circle contained within the base has not been matched but there are a number of small regions contained within the "lockbase

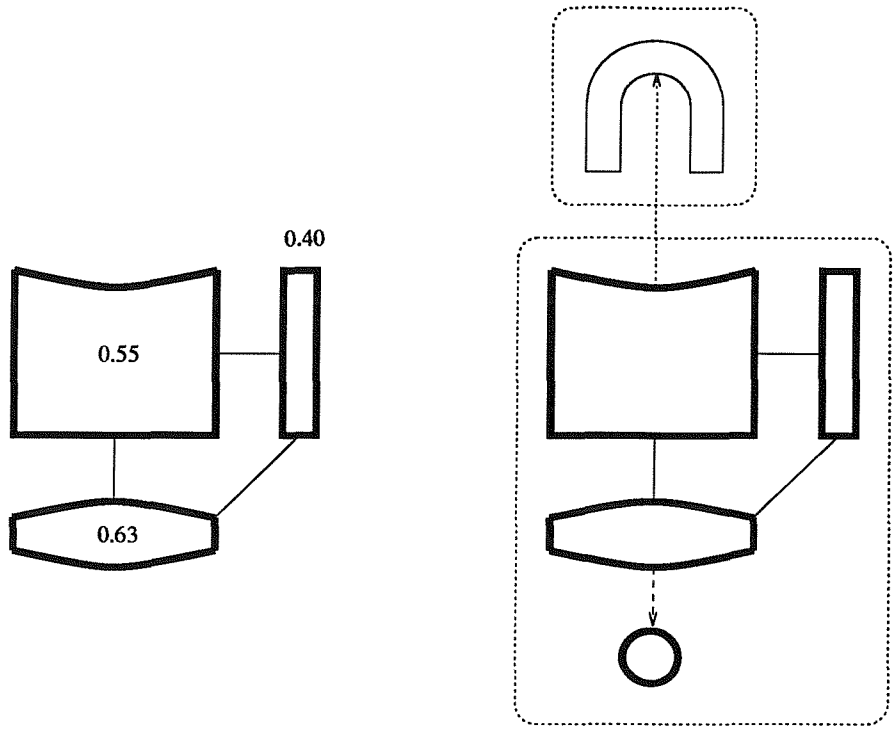


Figure 7.17: Image and Model, Padlock (lower view)

region" which lend evidence of its presence. Given that the body of the lock has been identified we know where to look for the loop part using other techniques.

For the second view of the padlock I have used regions with areas greater than 150 rather than 100. This is because there are several regions connected to the large central region which have an area just over 100, resulting in an explosion in the number of shape combinations. Additionally, the shapes shown in figure 7.16(b) are those found with confidence greater than 0.40 rather than 0.30 because there were five other region combinations which resulted in *lockside* matches with confidences between these values. These extra matches are discarded simply to make the following pictures easier to follow, where two overlapping *locksides* are shown we should remember that there are actually seven!

The extracted shapes form the SAG shown in figure 7.18(a) next to the corresponding aspect 7.18(b). Clearly an *overlaps* arc in the SAG must be considered as an "either

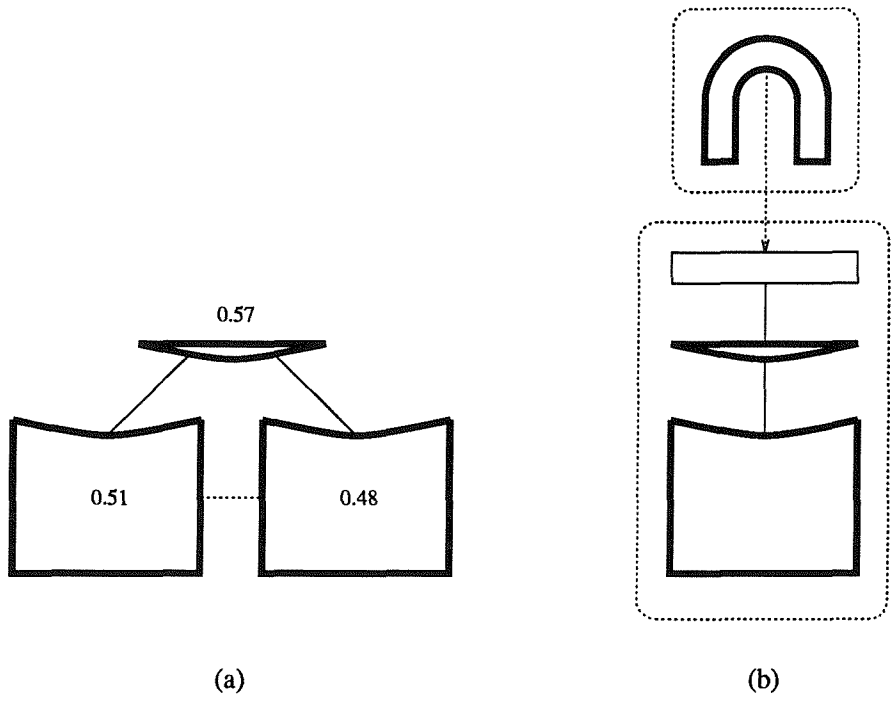


Figure 7.18: Image and Model, Padlock (front view)

or" operator since only one of the overlapping shapes can be matched to an aspect shape. Since the overlapping shapes are actually the same shape, the SAG immediately reduces to a two node graph of adjoining *lockcham* and *lockside* nodes. Given this reduction we again have an isomorphism between the subgraphs of SAG and aspect made up of fully visible nodes connected by *adjoins* arcs.

Note though that there is a node in the aspect (corresponding to the *locktop* shape) which is joined to nodes in the same part of the aspect but is overlapped by a shape in the aspect of the connected part (the loop). The presence or absence of this shape is therefore an indication of the presence or absence of the other part. Since the *locktop* has not been found but there are candidate regions adjacent to the "lockcham region" this is a strong indication that the loop is present.

7.5.3 Summary

Overall we can see that in each of the examples above, the object description derived from the image (SAG) and the viewer-centred model (aspect) can be matched. In

simple cases, matching involves proving that the two graphs are isomorphic but more complex arrangements require a more sophisticated approach which lends itself to knowledge-based methods.

7.6 Conclusions

In this chapter I have presented a new approach to the representation of three-dimensional objects which uses Shape-based Aspects (SBAs). By using a list of SBAs rather than an Aspect Graph we have the advantage of a much more compact representation: the information we have discarded (ie. the visual events) is of no real use to us since we are concerned with how an object appears in a particular image rather than how its appearance changes as our viewpoint moves.

Describing aspects in terms of shapes (surface projections) rather than lines and junctions (edge projections) means that it is easier to attach additional information to the representation. For instance, features such as colour and texture are properties of the surfaces of an object rather than its edges. Another important reason for using shapes is that they can generally be extracted more reliably from real images than lines and junctions.

A further advantage of the SBA approach is that it lends itself quite easily to the decomposition of objects into parts. Most contemporary Aspect Graph approaches treat an object as an indivisible entity, producing a very large number of aspects for a modest number of objects. Dickinson et al. [33] avoided this problem by describing objects in terms of a fixed number of volumetric primitives; although this gives a fixed number of aspects the approach cannot be easily extended beyond objects which can be conveniently described in terms of the chosen primitives. The representation described in this chapter combines the generality of a complete description with the compactness of a part description: there is no fixed set of primitives so new objects can be modelled accurately but distinct parts can be treated distinctly and re-used (like primitives) where possible to reduce the total number of aspects across all objects.

The motivation behind developing a viewer-centred representation was that object models and image models could be matched without performing any 2D↔3D transformations. Using the colour segmentation algorithm (chapter 3) and the Fractal Shape Signature (chapter 5) we saw how an image could be described by a Shape Association Graph (SAG) which has the same form as the graph used to describe SBAs.

In order to recognise objects in images it is therefore necessary to match the SAG extracted from the image to the SBA models of the objects. The matching process for real examples is ideally suited to a knowledge-based approach since several *rules* are required to determine which nodes and types of arcs may be compared at any one stage of the process. Such a system has not been implemented here but the types of operations required have been illustrated using real examples.

As indicated in chapter 1, the representations developed in this chapter should be considered as part of a global visual process. In order to co-ordinate all the processes discussed so far in a coherent and efficient manner, some form of intelligence will inevitably be required. The requirements of such an intelligent system will be discussed in the final chapter of this thesis.

8 Conclusions and Future Work

8.1 Conclusions

The aim of this work was to develop a compact model for three-dimensional objects and show how objects described by such models could be identified in real images. The three processes identified in chapter 1 and developed in chapters 3, 5 and 7 will be summarised in the following sections and a final section will identify areas for further research.

8.1.1 Colour Image Segmentation

The majority of segmentation algorithms require some definition of colour difference and many also need some notion of an average colour or a colour spread. We therefore began by defining colour metrics in terms of the most perceptually uniform colour space available, TekHVC. Since HVC space is perceptually uniform we expect these metrics to be more consistent than non-uniform spaces such as RGB and HSI: this was demonstrated in chapter 3.

Two segmentation algorithms were presented and compared: (i) a split and merge algorithm using colour spread as its homogeneity measure, followed by a relaxation process to remove unwanted artifacts; (ii) a region-grower which adds adjacent pixels in an order determined by the closeness of their colour to the present average. The region-growing technique was considered to be superior on the grounds of segmentation quality and flexibility.

Unlike many segmentation algorithms, the region-grower presented in this thesis places most emphasis on producing coherently coloured regions. This can lead to the

production of many small regions in noisy areas or where boundaries are indistinct rather than sharp. Since it is easier to subsequently merge regions than split them, over-segmentation is preferable to under-segmentation, particularly since many of the small regions are completely contained within larger regions and therefore have no effect on the shape of the boundary.

In general, I do not believe that a segmentation algorithm which has no knowledge of the objects in the scene can ever produce an *ideal* segmentation. However, it is highly unlikely that adjacent pixels which appear to have the same colour, should belong to different regions. The output of the region-grower is therefore a good starting point for higher-level processing.

8.1.2 Plane Shape Analysis

The *Fractal Shape Signature* was presented as a new method of representing and matching 2D shapes. By modelling a digital boundary as a fractional Brownian function and using information from *all* scales we obtain a description which is consistent and yet inherently simple. As required, the signature is invariant to changes in a shape's position, orientation and size; additionally, it is invariant under reflection, so mirror images are considered to be the same shape.

A simple matching technique was developed to provide numeric values for the difference and similarity between two signatures. Using this technique the signature was shown to be more robust than a competing method for moderate amounts of noise.

Since the 2D shapes considered are the projections of 3D surfaces, the changes in signature due to varying 3D viewpoint were examined. Rather than seek to develop a projective invariant (which would only work for flat surfaces) a more general approach based on the notion of a *property sphere* was proposed. The disadvantage of this method is the large number of signatures that are required to describe a general surface.

As a global shape descriptor, the main drawback of the fractal signature is its inability to cope with partially occluded shapes. However, other researchers have

not demonstrated that partial shape descriptions can be extracted reliably from real images of general shapes. A promising alternative, discussed in chapter 7, is to use high-level knowledge to "complete" the occluded shape so that a global descriptor can be used.

8.1.3 Object Modelling and Recognition

A new viewer-centred model for three-dimensional objects was presented. In contrast to existing *Aspect Graph* techniques, the visual events have been discarded, leaving a list of unique *aspects*. This results in a much more compact representation for many objects. A further difference between this and most previous approaches is that the *aspects* are face-based rather edge-based. This allows properties which are associated with surfaces (eg. colour) to be easily attached to the representation as well as the shape information.

Modelling each object as a single, indivisible, entity would produce a huge number of *aspects* for even a modest number of different objects. On the other hand, approximating objects as a conjunction of volumetric primitives will not always give good results; particularly when the objects cannot be decomposed easily or uniquely into the chosen primitives. The approach suggested in this thesis is a compromise between these extremes: objects are separated into re-usable *parts* when appropriate without limiting the total number of different parts.

Using a region grouping algorithm based on the colour metrics of chapter 3 and the shape analysis of chapter 5, a structure called a *Shape Association Graph* (SAG) was extracted from real images. This SAG was designed to have the same form as the graph used to describe object *aspects*, enabling matching to be performed without any complex transformations.

Matching between SAGs and *aspects* has been performed manually as a demonstration of the types of rules that are required. In the following section I will discuss how the matching process may be automated.

8.2 Future Work

The three processes summarised above, successfully achieve the transitions between the image layers that were described in chapter 1, ie. pixels→regions, regions→shapes and shapes→objects. Object recognition *can* be achieved by applying these processes sequentially to all the data available, as demonstrated in this thesis.

Improvements could be made to each of the stages to make them more general. As indicated in the text, texture is an important surface feature that could be combined with colour to give better image segmentation. Fractal signatures would be more useful if an affine invariant could be derived; this would enable shapes to be described more qualitatively since all rectangles, ellipses, triangles etc. would have single descriptions. Aspects were compared solely on the basis of shape; in general, colour, texture and relative size would also be very useful for matching.

To take this work forward, a more intelligent control strategy is required. At the simplest level, this system would contain the knowledge required to perform *region grouping*, *signature matching* and *SAG-Aspect matching* more generally, rather than using arbitrary heuristics.

At a higher level, an organisational layer will be required to apply the lower-level operations efficiently as the number of shapes and objects increases. Here, knowledge about how objects appear in scenes and how they interact with each other will be required. The completion of occluded shapes and the construction of partial segmentations are examples mentioned in previous chapters.

A general-purpose vision system will require a large amount of general knowledge about the world. However, I believe that the major failing in Computer Vision and Artificial Intelligence is not the amount of knowledge that can be incorporated but the way in which knowledge is organised and applied. Much progress in this area is required before a machine can truly *see* the world.

Bibliography

[1] K Arbter, W E Snyder, H Burkhardt, and G Hirzinger. Application of affine-invariant Fourier descriptors to recognition of 3D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:640–647, July 1990.

[2] R Bajcsy and F Solina. Three dimensional object recognition revisited. In *Proceedings of the 1st International Conference on Computer Vision*, pages 231–240, London, England, June 1987.

[3] D H Ballard. Generalising the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[4] D H Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.

[5] D H Ballard and C M Brown. *Computer Vision*. Prentice Hall, 1982.

[6] D H Ballard and D Sabbah. Viewer independent shape representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:653–660, November 1983.

[7] S T Barnard and A P Pentland. Three-dimensional shape from line drawings. In *Proceedings of the IJCAI*, volume 2, pages 1062–1064, Karlsruhe, Germany, August 1983.

[8] M Barnsley and A D Sloan. A better way to compress images. *Byte*, pages 215–223, January 1988.

[9] H G Barrow and J M Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[10] A Bengtsson and J-O Eklundh. Shape representation by multiscale contour approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:85–93, January 1991.

[11] P Besl. Geometric modelling and computer vision. *Proceedings of the IEEE*, 76:936–958, August 1988.

[12] P Besl and R Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17:75–145, March 1985.

[13] B Bhanu and C Ho. CAD-based 3D object representation for robot vision. *IEEE Transactions on Computers*, 20:19–35, August 1987.

[14] B Bhanu, C Ho, and T Henderson. 3D model building for computer vision. *Pattern Recognition Letters*, 5:349–356, May 1987.

[15] I Biederman. Matching image edges to object memory. In *Proceedings of the 1st International Conference on Computer Vision*, pages 384–392, London, England, June 1987.

[16] I Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[17] T Binford. Survey of model-based image analysis systems. *International Journal of Robotics Research*, 1:18–64, 1982.

[18] H Blum. A transformation for extracting new descriptions of shape. In *Symposium on models for the perception of speech and visual form*, Cambridge, MA, 1964.

[19] K Bowyer, O Faugeras, J Mundy, N Ahuja, C Dyer, A Pentland, R Jain, and K Ikeuchi. Workshop panel report: Why aspect graphs are not (yet) practical for computer vision. *CVGIP: Image Understanding*, 55:212–218, March 1992.

[20] K Bowyer, J Stewman, L Stark, and D Eggert. ERRORS-2: A 3D object recognition system using aspect graphs. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 6–10, November 1988.

[21] R Brooks. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:140–150, March 1983.

[22] F Bumbaca and K C Smith. Design and implementation of a colour vision model for computer vision applications. *Computer Vision, Graphics and Image Processing*, 39:226–245, 1987.

[23] M Celenk. A colour clustering technique for image segmentation. *Computer Vision, Graphics and Image Processing*, 52:145–170, 1990.

[24] I Chakravarty and H Freeman. Characteristic views as a basis for three-dimensional object recognition. *Proceedings of the SPIE on Robot Vision*, 336:37–45, 1982.

[25] J M Chassery and C Garbay. An iterative segmentation method based on a contextual colour and shape criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:794–800, November 1984.

[26] C Chen and A Kak. A robot vision system for recognising 3D objects in low-order polynomial time. *IEEE Transactions on Systems, Man and Cybernetics*, 19:1535–1563, November 1989.

[27] S Chen and H Freeman. The dominant views of solid objects. In *Proceedings of the 11th IAPR Conference on Pattern Recognition*, volume 1, pages 322–336, The Hague, Netherlands, August 1992.

[28] S S Chen, J M Keller, and R M Crownover. Shape from fractal geometry. *Artificial Intelligence*, 43:199–218, 1990.

[29] R T Chin and C R Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18:65–108, March 1986.

[30] CIE. CIE publication no.15(E-1.3.1),1971/(TC-1.3). Bureau Central de la CIE, Paris, France, 1978.

[31] P R Claxton and E K Y Kwok. The use of colour to segment and label images. In *Proceedings of the 3rd Alvey Vision Conference*, Sheffield, England, September 1987.

[32] L S Davis. Understanding shape: Angles and sides. *IEEE Transactions on Computers*, 26:236–242, 1977.

[33] S Dickinson, A Pentland, and A Rosenfeld. From volumes to views: An approach to 3D object recognition. *CVGIP: Image Understanding*, 55:130–154, March 1992.

[34] S J Dickinson, A P Pentland, and A Rosenfeld. Qualitative 3D shape reconstruction using distributed aspect graph matching. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 257–262, Osaka, Japan, December 1990.

[35] S A Dudani, K J Breeding, and R B McGhee. Aircraft identification by moment invariants. *IEEE Transactions on Computers*, 26:39–46, January 1977.

[36] B Falcidieno and F Giannini. Automatic recognition and representation of shape-based features in a geometric modelling system. *Computer Vision, Graphics and Image Processing*, 48:93–123, 1989.

[37] T Fan, G Medioni, and R Nevatia. Recognising 3D objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1140–1157, November 1989.

[38] G Fekete and L Davis. Property spheres: A new representation for 3D object recognition. In *Proceedings of the IEEE Computer Society Workshop on Computer Vision, Representation and Control*, pages 192–201, Annapolis, MD, April 1984.

[39] J A Feldman and Y Yakimovsky. Decision theory and artificial intelligence: 1. a semantics-based region analyser. *Artificial Intelligence*, 5:349–371, 1974.

[40] D A Forsyth. A system for finding changes in colour. In *Proceedings of the 3rd Alvey Vision Conference*, Sheffield, England, September 1987.

[41] A Fournier. Computer rendering of stochastic models. *Communications of the ACM*, 26(3):371–384, 1982.

[42] H Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, pages 260–268, June 1961.

[43] Z Gigus and J Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:113–122, February 1990.

[44] P A Giles, A Purvis, D Waugh, and R Garigliano. Iterated function systems and two-dimensional shape representation. In *Proceedings of the 5th Alvey Vision Conference*, pages 49–53, Reading, England, September 1989.

[45] C Goad. Fast 3D model-based vision. In A Pentland, editor, *From Pixels to Predicates*, pages 371–391. Ablex Publishing Co., 1986.

[46] R M Haralick and L G Shapiro. Survey: Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:100–132, 1985.

[47] J Ho, B V Funt, and M S Drew. Separating a colour signal into illumination and surface reflectance components: theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:966–977, October 1990.

[48] R Horaud, F Veillon, and T Skordas. Finding geometric and relational structures in an image. In *Proceedings of the 1st European Conference on Computer Vision*, pages 374–384, Antibes, France, April 1990.

[49] S J Horowitz and T Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *Proceedings of the 2nd IJCPR*, pages 424–433, August 1974.

[50] L M Hurvich and D Jameson. Some quantitative aspects of an opponent-colours theory. II. brightness, saturation and hue in normal and dichromatic vision. *Journal of the Optical Society of America*, 45:602–616, 1955.

[51] J Illingworth and J Kittler. A survey of efficient Hough Transform methods. In *Proceedings of the 3rd Alvey Vision Conference*, Sheffield, England, September 1987.

[52] A Jain and R Hoffman. Evidence-based recognition of 3D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:783–802, November 1988.

[53] T Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[54] J Kittler and J Illingworth. Relaxation labelling algorithms - a review. *Image and Vision Computing*, 3:206–216, November 1985.

[55] J Koenderink and A van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[56] M Korn and C Dyer. 3D multiview object representations for model-based object recognition. *Pattern Recognition*, 20:91–103, 1987.

[57] Y Lamdan, J T Schwartz, and H J Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6:578–589, October 1990.

[58] H Lee and R-H Park. Relaxation algorithm for shape matching of two-dimensional objects. *Pattern Recognition Letters*, 10:309–313, November 1989.

[59] X Li, M A Ireton, and C S Xydeas. Detection of the extreme points of closed curves. *IEE Proceedings-I*, 139:198–205, April 1992.

[60] H-C Liu and M D Srinath. Partial shape classification using contour matching in distance transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:1072–1079, November 1990.

[61] D Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[62] D Lowe. Stabilised solution for 3D model parameters. In *Proceedings of the 1st European Conference on Computer Vision*, pages 408–412, Antibes, France, April 1990.

[63] M R Luo. Perceptual colour models: An introductory review. *Alvey Project MMI/146 Technical Report*, 1986.

[64] L W MacDonald, M R Luo, and S A R Scrivener. Factors affecting the appearance of coloured images on a video display monitor. In *RPS Conference on 'Quantification of images'*, Cambridge, England, September 1989.

[65] B B Mandelbrot. *Fractals: Form, Chance, and Dimension.* Freeman, 1977.

[66] B B Mandelbrot. *The Fractal Geometry Of Nature.* Freeman, 1982.

[67] D Marr. *Vision.* Freeman, New York, 1982.

[68] S Marshall. Review of shape coding techniques. *Image and Vision Computing*, 7:281–294, November 1989.

[69] W Martin and J Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:150–158, March 1983.

[70] G G Medioni and Y Yasumoto. A note on using the fractal dimension for segmentation. In *Proceedings of the Workshop on Computer Vision: Representation and Control*, pages 25–30, Annapolis, Maryland, April 1984.

[71] F Meyer. Colour image segmentation. In *Proceedings of the 4th International Conference on Image Processing and its Applications*, pages 303–306, Maastricht, The Netherlands, April 1992.

[72] F Mokhtarian and A Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:34–43, January 1986.

[73] A H Munsell. *Munsell Book of Colour.* Munsell Colour Company Inc., 1970.

[74] A Nazif and M Levine. Low-level image segmentation : an expert system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:555–577, September 1984.

[75] R Nevatia. A colour edge detector and its use in scene segmentation. *IEEE Transactions on Systems, Man and Cybernetics*, 7:820–826, 1977.

[76] R Ohlander, K Price, and D Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[77] Y Ohta, T Kanade, and T Sakai. Colour information for region segmentation. *Computer Graphics and Image Processing*, 13:222–241, 1980.

[78] J O'Rourke. The signature of a plane curve. *SIAM Journal of Computing*, 15:34–51, February 1986.

[79] M Oshima and Y Shirai. A scene description method using three-dimensional information. *Pattern Recognition*, 11:9–17, 1979.

[80] T Pavlidis. SURVEY. a review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[81] T Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:301–312, July 1980.

[82] A Pentland. Recognition by parts. In *Proceedings of the 1st International Conference on Computer Vision*, pages 612–620, London, England, June 1987.

[83] A Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4:107–126, 1990.

[84] A P Pentland. Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:661–674, 1984.

[85] A P Pentland. Perceptual organisation and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.

[86] E Persoon and K-S Fu. Shape discrimination using fourier descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:388–397, May 1986.

[87] H Plantinga and C R Dyer. Visibility, occlusion and the aspect graph. Technical Report #736, Computer Sciences Department, University of Wisconsin-Madison, 1987.

[88] A Sarabi and J K Aggarwal. Segmentation of chromatic images. *Pattern Recognition*, 13:417–427, 1981.

[89] G Scott. Derivation of natural basis functions for a group of shapes. In *Proceedings of the 5th Alvey Vision Conference*, pages 43–48, Reading, England, September 1989.

[90] A Shimaya, I Yoroizawa, and M Kosugi. An interpretative model of overlapped figures. In *Proceedings of the 11th IAPR Conference on Pattern Recognition*, volume 2, pages 170–173, The Hague, Netherlands, August 1992.

[91] O Skliar and M H Loew. A new method for characterisation of shape. *Pattern Recognition Letters*, 3:335–341, September 1985.

[92] A R Smith. Colour gamut transform pairs. In *Proceedings of SIGGRAPH 78*, pages 12–19, Atlanta, Georgia, August 1978.

[93] T Sripradisvarakul and R Jain. Generating aspect graphs for curved objects. In *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115, November 1989.

[94] G Stockman. Object recognition and localisation via pose clustering. *Computer Vision, Graphics and Image Processing*, 40:361–387, 1987.

[95] J Tajima. Uniform colour scale applications to computer graphics. *Computer Vision, Graphics and Image Processing*, 21:305–325, 1983.

[96] C J Taylor and D H Cooper. Shape verification using belief updating. In *British Machine Vision Conference*, pages 61–66, Oxford, UK, September 1990.

[97] J M Taylor, G M Murch, and P A McManus. TekHVC^{TM}: A uniform perceptual colour system for display users. *Proceedings of the SID*, 30:15–21, 1989.

[98] R I Taylor and P H Lewis. A fractal shape signature. In *British Machine Vision Conference*, pages 178–184, University of Glasgow, Scotland, September 1991.

[99] R I Taylor and P H Lewis. Colour image segmentation using boundary relaxation. In *Proceedings of the 11th IAPR Conference on Pattern Recognition*, volume 3, pages 721–724, The Hague, Netherlands, August 1992.

[100] D Terzopoulos, A Witkin, and M Kass. Symmetry-seeking models for 3D object reconstruction. In *Proceedings of the 1st International Conference on Computer Vision*, pages 269–276, London, England, June 1987.

[101] C Thorpe and S Shafer. Topological correspondence in line drawings of multiple views of objects. Technical Report CMU-CS-83-113, Carnegie-Mellon University, 1983.

[102] M Tsukada and Y Ohta. An approach to colour constancy using multiple images. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 385–389, Osaka, Japan, December 1990.

[103] S A Underwood and C L Coates. Visual learning from multiple views. *IEEE Transactions on Computers*, 24:651–661, June 1975.

[104] T Vlachos and G Constantinides. A graph-theoretic approach to colour image segmentation and contour classification. In *Proceedings of the 4th International Conference on Image Processing and its Applications*, pages 298–302, Maastricht, The Netherlands, April 1992.

[105] E L Walker and M Herman. Geometric reasoning for constructing 3D scene descriptions from images. *Artificial Intelligence*, 37:275–290, 1988.

[106] R Wang and H Freeman. Object recognition based on characteristic view classes. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 8–12, 1990.

[107] A P Witkin. Scale-space filtering: a new approach to multi-scale description. In *Image Understanding*, 1984.

[108] H J Wolfson. On curve matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:483–489, May 1990.

[109] M W Wright. Skeletonisation as model based feature detection. In *Proceedings of the 4th International Conference on Image Processing and its Applications*, pages 254–257, Maastricht, The Netherlands, April 1992.

[110] W A Wright. A Markov random field approach to data fusion and colour segmentation. *Image and Vision Computing*, 7:144–150, May 1989.

[111] R-H Wu and H Stark. Rotation and scale invariant recognition of images. In *Proceedings of the Eighth International Conference on Pattern Recognition*, pages 92–95, Paris, France, October 1986.

[112] N Yokoya and K Yamamoto. Fractal-based analysis and interpolation of 3D natural surface shapes and their application to terrain modelling. *Computer Vision, Graphics and Image Processing*, 46:284–302, 1989.

[113] T Y Young and K-S Fu, editors. *Handbook of Pattern Recognition and Image Processing*. Academic Press, 1986.

[114] C Yu, K C Wong, and J Kittler. The recognition of triangle-pairs and quadrilaterals from a single perspective view. In *Proceedings of the 4th International Conference on Image Processing and its Applications*, pages 425–432, Maastricht, The Netherlands, April 1992.

[115] C T Zahn and R Z Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, 21:269–281, March 1972.