COMP27112
Computer Graphics and Image Processing
Part 2: Image Processing

Tim Morris

Chapter 1
Introduction

1.1 What are image processing and computer vision?

Many definitions for "image processing" and "computer vision" have been proposed but it is usually accepted that image processing involves manipulating a digital image to generate a second image that differs in some respects from the first, and computer vision involves extracting numerical or symbolic information from images, see figure 1.1.

For example, a weather station might capture an image of rainfall intensity using the appropriate radar sensor. Each pixel in this image has a value that is proportional to the amount of rain falling in the corresponding area. This is usually presented by converting the data to a colour image (image processing). The image might also be interpreted to recognise areas of higher than average rainfall and potential flood risk (computer vision) or to recognise weather systems.

It is important to realise that alternative definitions exist: these are young disciplines and there is, as yet, incomplete agreement over the subjects' boundaries. Indeed, the boundary is least distinct between image processing and computer vision, hence the overlap in figure 1.1.

In this text, we are going to examine the methods by which digital images are processed to, ultimately, extract useful information.

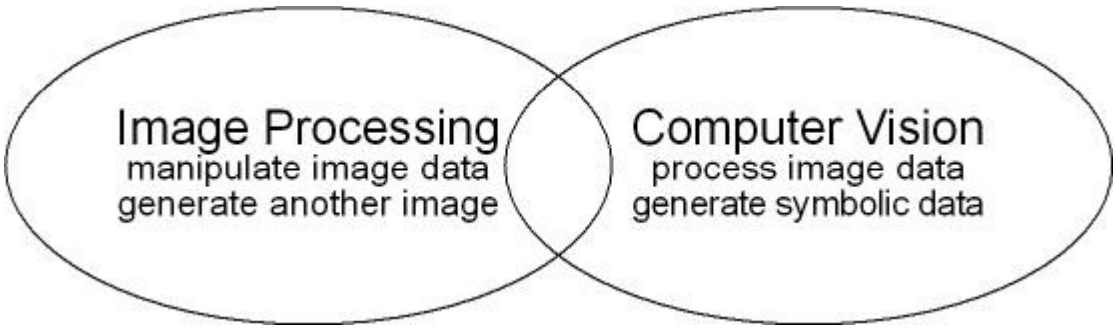


Figure 1.1: Definitions of image processing and computer vision

1.2 Historical overview

Although image processing and computer vision are relatively new subjects, the first digital image was created in the 1920s when photographs were converted to a numerical format for transmission by Morse code across the Atlantic by telegraph. In this process, the photograph was divided into small areas that were assigned a numerical code that depended on the distribution of brightnesses in them. The code was transmitted and the photograph reconstructed using specially designed printing blocks. This was a time consuming process, each photograph would take several hours to code, transmit and print. However, the alternative was to send the photograph across the ocean by steamship, which could take over a week.

The process of coding, transmitting by telephone and printing photographs did not change conceptually for some 50 years, but the time taken to complete the process became shorter as the hardware that was used became more sophisticated.

Major advances in image processing began to be made in the 1960s. The US and Russian space programmes began to generate large amounts of image data that needed to have artefacts removed and to be enhanced for viewing. Simultaneously, advances in the technology meant that computers became powerful enough to store images and to process them within realistic time frames and to display the images.

Since then, we have seen the cost and availability of computer hardware change to make powerful computers available to the mass market – sufficiently powerful for many image processing or computer vision applications. With the increased availability of computing power has come a vast increase in the range of applications of image processing and computer vision systems. From being restricted to government, university or industrial laboratories, these systems have emerged into everyday life; we see their output in our entertainment and news, in medical imaging devices, in scientific research, exploration and many others activities.

1.3 Sample applications

Three simple applications of vision systems will be described. These have been selected on the basis of their familiarity with everyday situations.

1.3.1 Optical Character Recognition

Systems that recognise characters in an image have been in existence for many years. Their applications range from simple interpretation of printed text to reading postcodes on envelopes to recognising signatures on cheques to reading car registration plates. In these systems, we may identify three levels of difficulty:

- recognising printed text,
- recognising hand printed text,
- recognising cursively written text.

Printed text is characterised by its quality and regularity. Any printed character will have a consistent appearance and it can therefore be recognised by comparison with a prototype. Figure 1.2 illustrates the process. A page of text was scanned. The area surrounding one character was copied into the prototype. The output image is the measure of similarity between the prototype and the different parts of the input image. The final part of figure 1.2 shows a profile along the line indicated in 1.2a: it is apparent that the largest response occurs where the prototype and image actually match, but large responses are also seen at locations where there are partial matches. A real character recognition system would require a prototype for every object that is to be recognised: each character, lower and upper case, and each punctuation mark. A set of prototypes is required for each font and font size. Any practical system will require a huge number of prototypes if it is to use this technique. Fortunately, other techniques are available.

Hand printed text is of a lower quality and is less regular. Hand printed text would need to be recognised by systems used for automatic sorting of the post: letters are sorted using the postcode, or by systems that read forms filled in by hand. Figure 1.3 shows samples of a letter written by one person over a period of time. Due to the irregularity of the character, we cannot recognise it by comparison with a template. Instead, we must use the structural characteristics to identify the letter. For example, the upper case letter “A” is composed of a horizontal stroke in the middle of the character and two diagonal strokes flaring outwards from the top central portion of the character. In fact, every character can be broken down to a set of strokes. If we are able to process the character image and identify those strokes, then we will be able to recognise the character. The other major problem associated with recognising hand printed text is actually separating each character from its neighbours: adjacent characters can overlap each other and can therefore be difficult to separate; it is also possible for the strokes of an individual character to be separated and to appear as two separate characters.

Recognition of handwritten (cursive) text is a problem that has not yet been solved. The text is so irregular that templates cannot be used, neither is it always possible to isolate the individual strokes that make up a character, see figure 1.4.

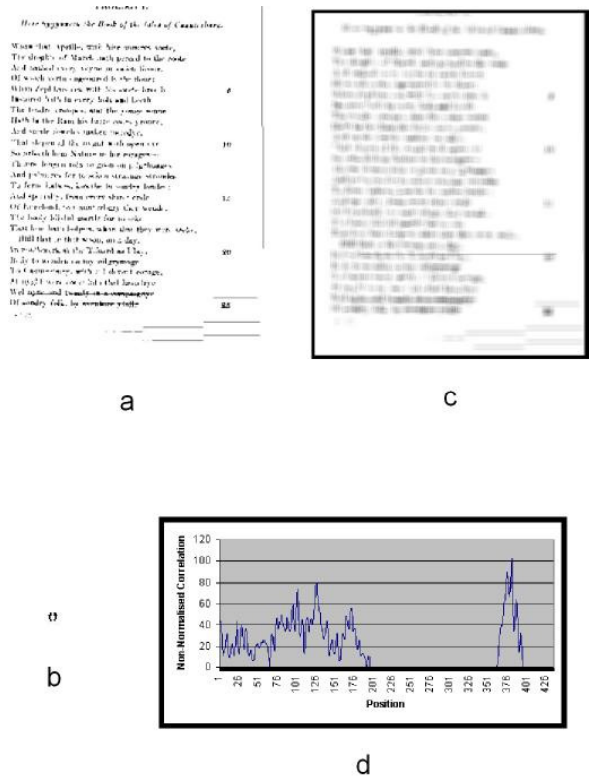


Figure 1.2: Character recognition by comparing each character with a prototype.
a) a scanned page of text,
b) a prototype for the character "0" derived from this image,
c) output of the comparison: the darker the pixel, the better the match between the scanned image and the prototype.
d) Magnitude of the result along the line indicated in c.

Figure 1.3: Samples of one handwritten character written by one person over a period of time.

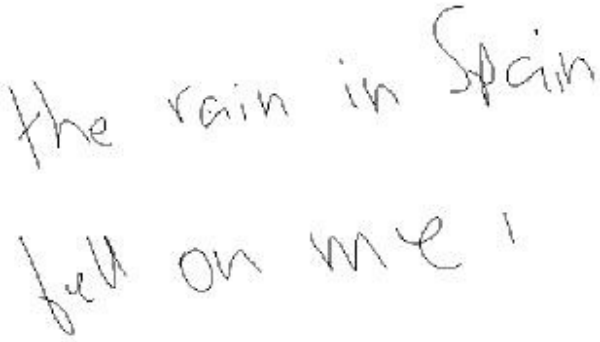


Figure 1.4: A sample of handwritten (cursive) text.

1.3.2 Biometry

Biometry is the science of identifying people from their physical characteristics. In the context of image processing and computer vision, this will imply recognition using images of the physical characteristics. Recognising individuals from images of their fingerprints is a problem that has been investigated for many years, as has recognition using the individual's face. One characteristic that has received attention recently is the iris. The striations on the iris are unique to each individual and are therefore ideally suited as a means of identification, for example, to control access or for personal banking, figure 1.5 illustrates an overview of the system.

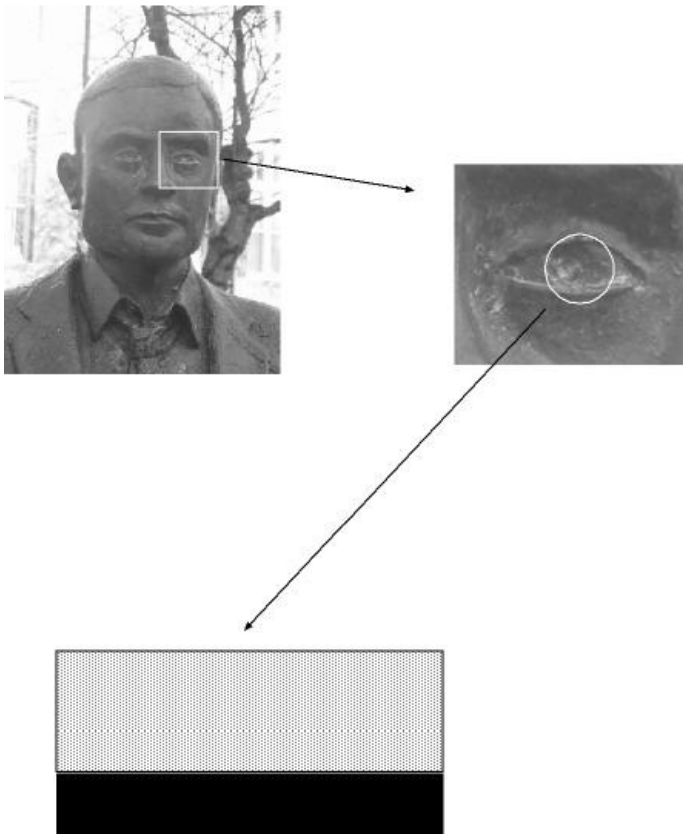


Figure 1.5: Processing stages of the iris recognition system. The eye is identified in the head and shoulders image and resampled to give a size independent representation of the iris.

An iris scanner will firstly identify the subject's eyes in a head and shoulders image. The iris region is then identified and resampled, converting the annular region into a rectangular one. Different lighting conditions will cause the pupil to dilate or contract hence changing the size of the iris, the resampling will ensure that the radial dimension is sampled with a constant number of samples, irrespective of its size. The iris region is then numerically transformed and represented by a small number of coefficients. These coefficients have been shown to be unique to each individual.

1.3.3 Aerial photography

Aerial photography finds many applications: mapping, land use analysis, land use change, entertainment, etc. Whilst images may be captured from altitudes ranging from a few hundred metres to a few hundred kilometres, they are all treated in a similar fashion to generate the view that is seen when looking vertically towards the ground.

When an image is captured, we must record the exact location and attitude of the camera, that is, where the camera is situated relative to a fixed datum, and what direction it is pointing in. The captured image will be distorted if the camera is not orientated vertically, since that would capture an oblique view of the ground, see figure 1.6. The image will also be distorted, to a greater or lesser degree, by the

camera's optical system: a short focal length lens, having a wide field of view will distort objects such that those towards the periphery of the image appear to be larger than they actually are (other, altitude-related distortions are also apparent).

Figure 1.6: Aerial photography, images must be captured with the camera pointing vertically downwards to avoid distorted images, unlike this example of an image taken at an oblique angle from a tall building.

If, as is often the case, we wish to assemble a number of images into a larger mosaic, we must either ensure that the camera is moved along a level, straight path and images are taken at regular intervals, or we must record the exact positions of the camera as images are captured and subsequently modify the image data. The latter alternative effectively resamples the data to compute the image that would be captured if the camera had been moved along a level and straight path, see figure 1.7. The images will also be corrected for any optical distortions.

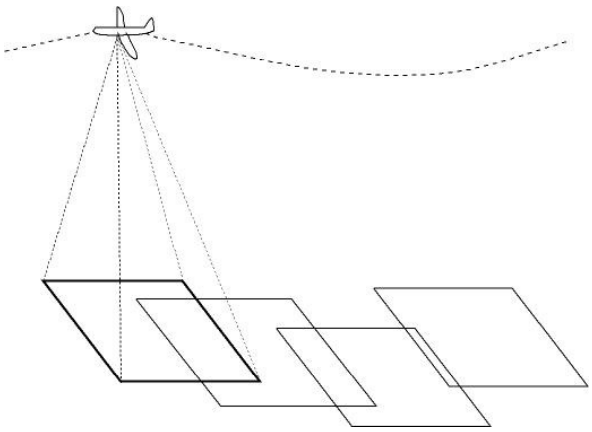


Figure 1.7: Building a mosaic image from a set of aerial photographs.

1.4 Routemap to the book

A computer vision system will include some or all of the components shown in figure 1.8 in the order shown. In this text we shall discuss the properties of image capture devices but will not examine any specific device. The starting point is a captured digital image. Chapter two deals with **Image Representation**: what do we require of an image capture device in order to have suitable data for the task at hand? Chapters three to seven deal with the components of the vision system presented in figure 1.8.

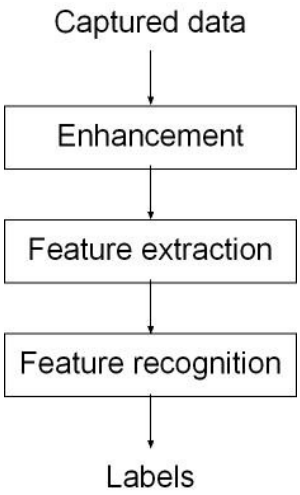


Figure 1.8: Overview of a computer vision system

Enhancement is any operation that is applied to the data to remove the degradations that might have been introduced by the capture process: optical degradations mentioned above, the effect of camera movement or any loss of clarity caused by atmospheric conditions or lighting. Figure 1.9 presents some examples.

Feature extraction operations aim to identify structures in the image that are useful for recognising the objects being analysed. They might include the pixels that constitute an area, or the straight boundaries between regions, figure 1.10.

Feature recognition processes will use the features just extracted to assign a label to the image or to parts of the image, by comparing the features against those derived from previously labelled objects.

Figure 9: Image distortions caused by camera shake, blurring, non-ideal illumination.

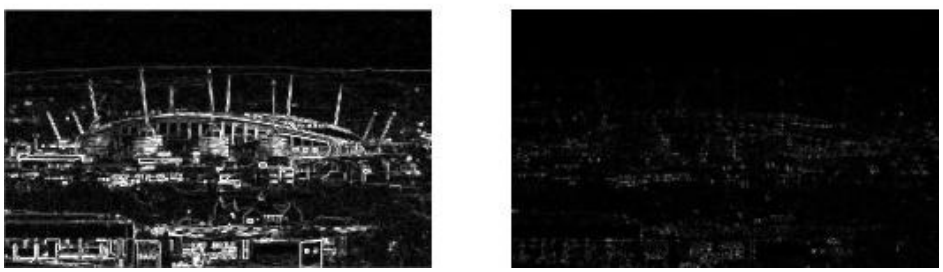


Figure 10: Sample feature extraction processes

1.5 Bibliography

Gonzalez and Woods (1993) present a detailed historical review of the development of the subject. Additionally, Castleman (1996), Haralick and Shapiro (1992), Jain et al. (1995), Schalkoff (1989) and Forsyth and Ponce (2003) provide a more advanced coverage of the topics covered in this text.

A number of websites and newsgroups have useful information. Probably the most important is the Computer Vision homepage at http://www-2.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html, which has links to most computer vision research groups worldwide, research reports and demonstration software.

Three useful newsgroups are also active: sci.image.processing, comp.ai.vision and sci.med.vision

1.6 Exercises

1. Suggest everyday problems that an image processing system could sensibly solve. Outline the tasks that the system would be required to perform. What difficulties would the system encounter? Are non-vision system solutions better?

2. Describe in outline an image processing system you have encountered. Identify the separate components of the system

Chapter 2
Image Representation

This chapter presents material that is necessary to understand how a digital image is stored and represented: specifically the number of pixels that can or should be used to represent the image, and the number of shades of grey or number of colours that can or should be used. We also consider camera calibration, which is important when we must relate image co-ordinates to a location in the real world. Finally, we examine the parallels between computer vision systems and the human visual system.

After reading this chapter, you should be able to specify the image resolution required for a vision system to achieve a specific task. You should also understand how image and world co-ordinates are related – this will be used in later chapters that discuss methods of determining the shape of objects. Finally, you should appreciate the similarities and differences between human and computer vision systems.

2.1 Introduction

Many authors suggest that parallels exist between computer vision systems and the human visual system. There is no doubt that there are similarities, but there are also subtle differences. The human visual system is examined in this chapter since an understanding of the human visual system can help in appreciating how computer vision systems might be designed. In particular, an examination of how humans solve visual problems can be beneficial when designing algorithms for processing visual data.

The real starting point of our examination of computer vision systems is with a captured digital image that is ready for processing. The reason for this is that images are captured using a multitude of devices and it is therefore not practicable to discuss them all. However, the processing of images cannot be considered without also considering the capture of the data: the data that is captured will determine what information can be extracted from it. If the image has so small a resolution that objects cannot be seen clearly, then this will obviously limit the amount of information that can be gleaned from it. An early part of this chapter discusses some of the image capture modalities, the remainder of the book assumes that camera-based systems are used for capture: either still of video cameras.

Of primary importance in the design of any computer vision system is a consideration of the requirements of image capture without reference to how the data is collected. That is, we must consider spatial and brightness (or colour) resolutions and relate these to what are required of a particular task in order to capture images that contain the information required to solve the given problem. We also consider factors that limit the resolution that can be achieved by a specific image capture system.

We shall examine how captured data is represented. Monochrome image data is invariably represented using one storage unit per pixel, whether this storage unit is an eight bit value or greater. Colour image data, following the old tristimulus theory is represented using a triplet of storage units for each pixel. However, the triplet may be organised in a variety of ways. These are discussed and analysed. Reasons for preferring one representation over any others are presented.

Camera calibration is discussed. This is the process by which relationships are established between image co-ordinates and a co-ordinate system tied to the real world. The process involves two steps, one relating the location and orientation of the camera to the world co-ordinate frame, and the second relating image co-ordinates to co-ordinates in a frame tied to the camera which is essentially the process of correcting the image data for the defects introduced by the camera's optical system.

2.2 The human visual system

The human visual system provides a useful analogy with computer vision systems and therefore a good starting point in our discussion of them. Often, when designing a computer vision system, the fact that a human can solve a visual problem is the only indication that the problem is indeed soluble.

The visual world is perceived because light reflected from surfaces in it is focused on to light sensitive cells in the eye. These convert the light energy to electrical signals that are transmitted to the brain and

there interpreted as the objects we "see". Figure 2.1 illustrates schematically the gross anatomy of the eye and will be used to explain this mechanism. The eye contains structures whose purposes are first to regulate the amount of light entering it, second to focus rays of light from objects at varying ranges onto the retina and third the retina itself whose purpose is to encode the nerve pattern into nerve signals.

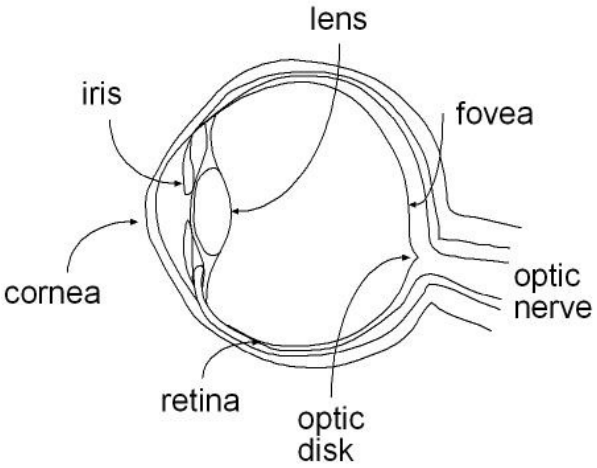


Figure 2.1: Schematic cross section though a human eye.

Light reflected from a surface in the scene being viewed is focussed on the retina in a two-stage process. The curved outer surface of the eyeball provides the primary focussing, but the variable focussing that is required to enable us to see objects at different ranges is achieved by changing the refractive power of the eye's lens, a process known as accommodation.

The iris regulates the amount of light entering the eye. In dark conditions, more light is needed to enable us to see clearly and consequently the iris dilates. The iris contracts in bright conditions so admitting less light. However, the iris diameter changes only by a factor of four, its area therefore changes by a factor of sixteen, which is much less than the range of brightness levels over which we can comfortably see. The changing iris size provides a fine control over the eye's adaptation, the major control is achieved by the retina itself changing its sensitivity: in bright conditions, the retina is actually less sensitive to light. This is analogous to the case of a photographic camera; exposure control is achieved by choosing film of varying sensitivity and by altering the aperture of the lens.

Anatomical studies reveal that the retina contains some hundreds of millions of light sensitive cells, whilst the optic nerve that transmits information from the eye to the brain has only some hundreds of thousands of individual nerve fibres. A significant amount of data reduction must therefore occur in the retina. It is suggested that the retina sending difference signals onwards to the brain achieves the data reduction.

Spatial and temporal differences are computed. Spatial differences are simply the differences in the image data between nearby regions of an image; the computer vision analogy is edge detection. Temporal differences are the differences between the same region in temporally adjacent images. The latter effect can by observed by staring fixedly at a static scene without blinking or moving the eyes. Gradually the scene will fade to grey as the temporal differences fall to zero. Any movement, in the scene or the eyes, will result in the scene being regenerated.

When we view a scene, three factors influence the quality of the perceived data: the angular resolution of the eyes, the colour sensitivity and the brightness sensitivity. These in turn are determined by the eye's physiology: principally the number and distribution over the retina of light sensitive cells and how these cells react to light of different brightness and wavelength.

The retina consists of two types of light sensitive cell called rods and cones because of their shapes (figure 2.2). The rods are generally sensitive to the intensity of the light striking them and therefore give a monochromatic response. They are also more sensitive in lower light levels than the cones; hence we are unable to see colours at night. The cones are differentially sensitive to light of varying wavelength. Figure 2.3 is a sketch graph of the three types of cones' reactivity to light. It is apparent that there are cones that are more sensitive in the red region, the green region and the blue region of the spectrum, and that each type of cone is relatively insensate to light outside of its specialised wavelength range. Although it is incorrect to say so, we often suggest that the cones are red, green or blue sensitive.

The rods and cones are distributed across the retina as shown in figure 2.4. The colour sensitive cones are more densely packed on or near the optic axis and are largely absent from the remainder of the retina. It is not surprising that the processing power of the brain's visual centres is largely devoted to data received from this region. A consequence of this is that our peripheral vision is almost monochrome. Conversely, the rods are most highly concentrated in an annulus surrounding the eye's optical axis and are distributed more evenly over the rest of the retinal surface. The rod's distribution has two consequences: our spatial resolution in the peripheral visual regions is very poor, an object must be large or fast moving to be seen; and since the rods have their highest concentration just off-axis, we can see dimly illuminated objects most clearly by viewing them slightly askance.

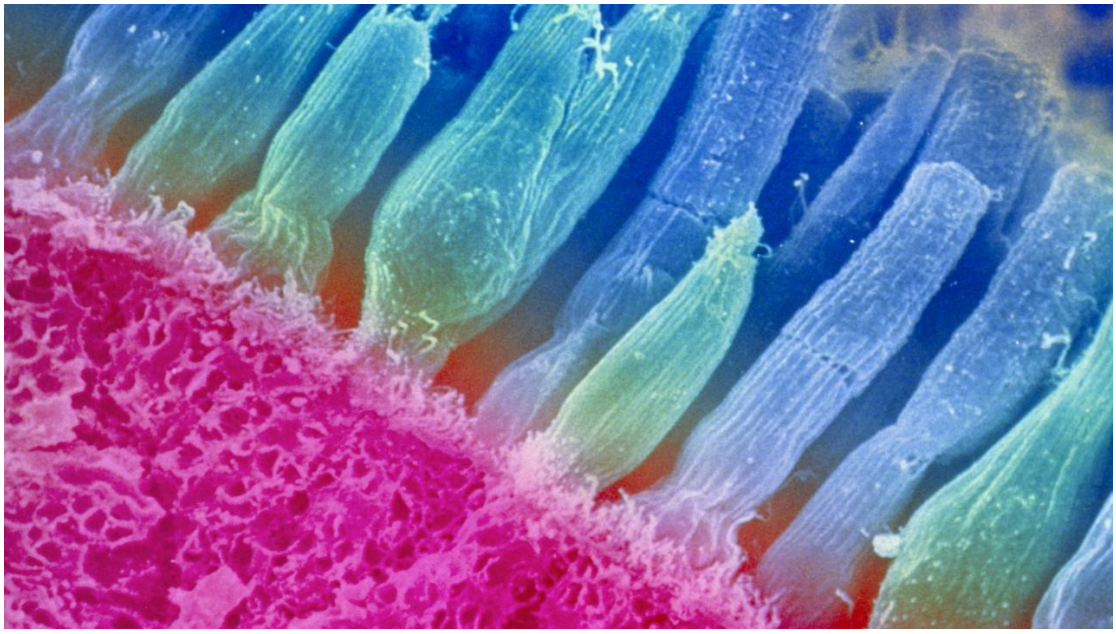


Figure 2.2: Photomicrograph of the retina's surface. in false colour.

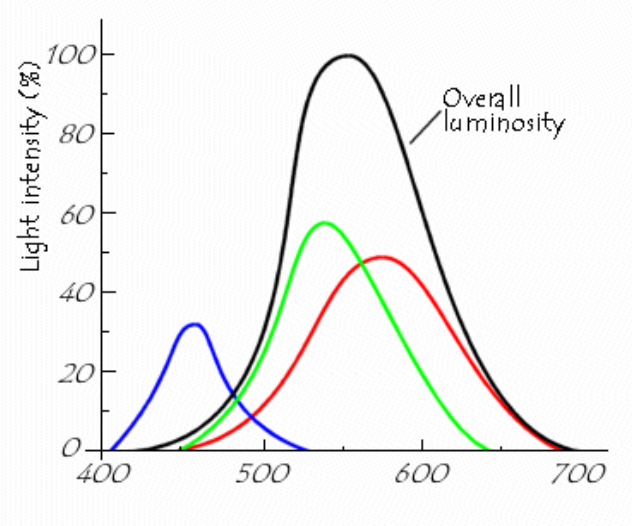


Figure 2.3: Sensitivity of the three types of cone to light – the horizontal axis is wavelength in nm.

The maximum concentration of light cells is in the foveal region on the eye's optic axis. The cells in this region are exclusively colour sensitive cones, there may be approximately 2000 cones in a region that is approximately 100 µm across. The sizes of the cones varies slightly, the smallest are 1 µm across and subtend an angle of 20 seconds of arc. This figure will suggest a size for the smallest discernible object of 0.2 mm at a range of 1 m. In fact our visual acuity can be better than this because we make use of information from more than a single rod or cone in creating our understanding of the

world.

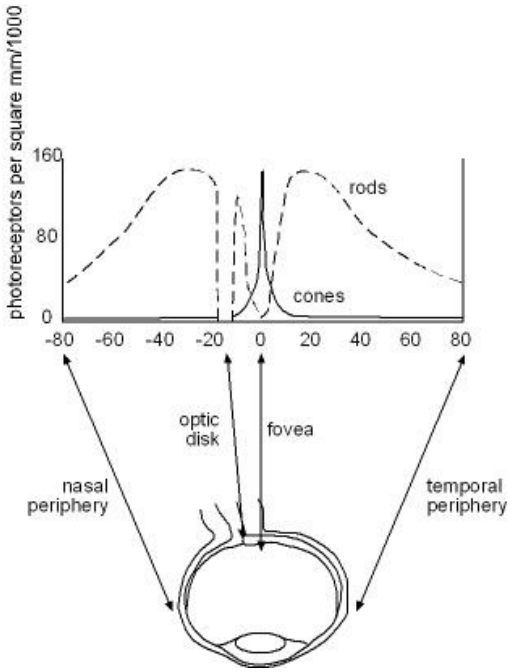


Figure 2.4: Distribution of rods and cones across the retina.

Physiological studies have shown that at any one brightness adaptation, only about 40 shades of brightness may be distinguished. If you were to fixate on a single spot in an image, you would distinguish just those 40 shades. We are able to differentiate much more than this number as we scan over an image by using the differential information, in a sense our brightness adaptation alters as we scan over the image.

Colour vision is a more complex phenomenon. Newton suggested that there were seven colours of light that generated white when mixed. On that basis it would have been sensible to suggest cells sensitive to all seven colours. However, Thomas Young in 1801 proposed that there were in fact three identifiably different colour receptors. The basis of Young's theory was the observation that most colours could be matched by mixing different proportions of a red, a green and a blue primary source. These ideas are still in use, in fact they form the origin of the colour representation models to be discussed below.

2.3 Image capture

Images may be captured using a wide range of instruments, ranging from radio telescopes imaging the far reaches of the universe to electron microscopes investigating the smallest scale phenomena. More familiar will be, for example, the images collected by weather satellites and displayed daily during weather forecasts. Despite the range of data capture modalities, the majority of computer vision applications capture images generated by video cameras.

Whilst there are innumerable data capture methods, they all share the property of being able to transform some type of energy into a visible form. For example, a radio telescope will measure the intensities of radio waves received from various orientations in the sky and assemble an image; an ultrasound scanner will measure the intensity of the sound pulse reflected from structures within a body, again, the data will be assembled into an image.

An illuminated object is illustrated schematically in figure 2.5. Light energy from a source falls on the object. A proportion of that light is reflected and some may fall on the light transducer (e.g. the detector). It should be noted that this is a generalised scenario; self-illuminated objects exist that do not require a source of illumination. The detector could be a video camera, a radio telescope or whatever instrument is used to convert energy into an image.

If a video camera is used for image conversion, it will transform the illuminated scene into an electrical signal. An analogue video camera will output a standard PAL or NTSC signal; a digital camera will output digitised data. The analogue signal must then be converted into a digital format. A frame

grabbing card that is mounted in the computer usually accomplishes this task. It will place the digitised image in the computer's memory or memory situated on the card ready to be transferred to the computer.

Whatever type of detector is used to capture the image data, it will transform a continuous brightness function into a discretely sampled image. For convenience, the sampling is usually performed on a rectangular grid, that is, the samples are equally separated in two orthogonal directions (figure 2.6), but other alternatives have been used. The number of samples and the spatial separations of the samples are considered below.

We are now able to describe a simple model that explains image formation. The following discussion is restricted to monochrome (black and white) images, but is readily extended to explain colour image formation. An image is formed by the capture of radiant energy that has been reflected from surfaces that are viewed. We may therefore define upper and lower bounds for the values that the reflected energy, *f(i, j)*, can take:

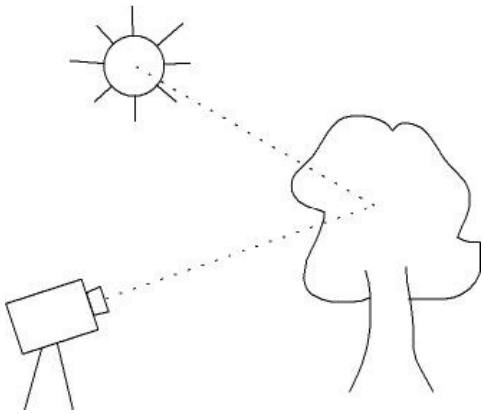


Figure 2.5: Image capture. A scene is illuminated by energy from a light source. Reflected light may fall on the detector and be captured.

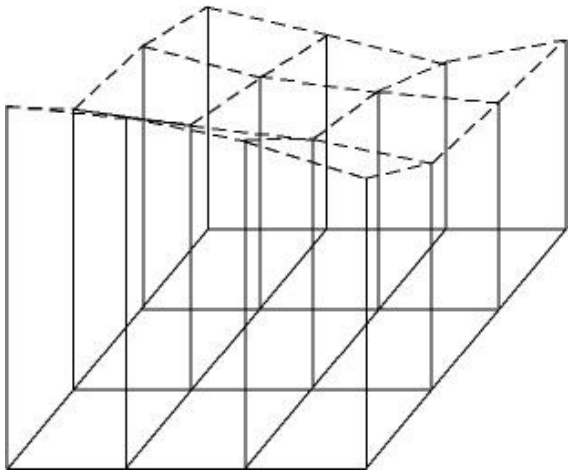


Figure 2.6: Rectangular sampling grid.

$$0 \le f(x, y) < \infty \qquad 2.1$$

Clearly the value will be zero if no energy is reflected, and the amount of energy reflected must also be finite.

The amount of reflected energy is determined by two functions: the amount of light falling on a scene and the reflectivity of the various surfaces in the scene. The two functions are known as the illumination, *i(x,y)*, and reflectance, *r(x,y)*, components, they combine to give:

$$f(x,y) = i(x,y)r(x,y) \qquad 2.2$$

The illumination function can theoretically take values between zero and infinity. In practice, the upper bound is set by the maximum ambient illumination, whether this is a well lit room indoors, a sunlit scene or a night time scene.

The reflectivity function can take values within the range zero (complete absorption of incident energy) to one (complete reflection of incident energy). In practice, the complete range is not realised, black velvet has a reflection coefficient of about 0.01 and snow a coefficient of approximately 0.95.

Thus the brightness values, *l*, in a scene can take values in the range:

$$i_{\min}(x,y)r_{\min}(x,y) \le l \le i_{\max}(x,y)r_{\max}(x,y) \qquad 2.3$$

When an image of this scene is captured, it is usual practice to clamp the minimum value to zero and the maximum value to whatever maximum is allowed by the resolution of the hardware, typically 255. A value of zero will equate to black and 255 to white. Intervening values will equate to shades of grey of varying intensity. This range of values is termed the grey scale.

2.4 Image quality measurements

Any system that measures a physical property can only do so to a finite accuracy which is determined in part by the property being measured and in part by the measuring equipment. For example, when measuring distances, an appropriate scale is used. One would not use a metre rule to determine the separation between two cities nor the thickness of a piece of paper. Indeed, even if the metre rule was being used to measure an appropriate distance, such as the width of a doorway, the measurement that is obtained could still be erroneous – is the metre rule accurate? Can I read the distance accurately?

These problems are equally applicable to image data. Imaging devices focus radiant energy from specific orientations onto a photosensitive detector. The quantity of energy that is absorbed by the detector in a specified time is measured digitally. The measurements are assembled into an image. All of these processes are subject to inaccuracies in one form or another and the net effect is that the image we derive is not strictly accurate – there is some measurement uncertainty that must be associated with each pixel value. The uncertainty is usually quantified by measuring the system's signal to noise ratio. In this section we shall discuss some of the causes of the loss of image quality and the measurement of image quality.

2.4.1 Sources of degradation

A major cause of loss of quality in the image has been alluded to: noise in the electronics that converts radiant energy to an electrical signal. This is the easiest degradation to measure and is usually quoted by camera manufacturers. However it is not the sole source of degradation, the camera's lenses also distort the image and the detector itself may not respond as we would like it to. Several sources of degradation may be identified.

Geometric distortion is a problem with all lenses. It arises because the best focus of the scene in front of the lens is in fact on spherical surface behind it. In practice it is not possible to fabricate detectors of this shape, so the image is projected onto a planar detector, with an associated distortion to the image (recall the problem of drawing a map of the world). The distortions become more severe as we use lenses with progressively shorter focal lengths to give wider angles of view. Figure 2.7 illustrates the effect of imaging a rectangular grid: the distortions are apparent around the periphery of the image.

Scattering occurs when rays of light reflected by the surface to be viewed are further dispersed by material in the optical path between the surface and the detector. Fog is the obvious example; it gives rise to a common name for this phenomenon: fogging.

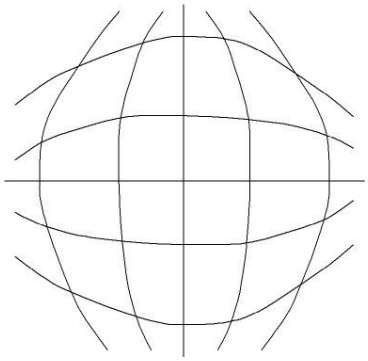


Figure 2.7: Schematic image of a rectangular grid captured using a very short focal length lens.

The refractive index of the glass or plastic from which a lens is constructed is a function of the wavelength (colour) of the light being refracted. This is demonstrated by passing a ray of white light through a prism, the white light is split into the constituent colours. The consequence of this for an optical system that should focus light onto a fixed point is serious: it cannot be done. The effect of this is that white objects will have a coloured fringe and sharp boundaries are blurred.

Blooming is the effect that is seen when a bright point source is imaged, the image is blurred slightly due to two causes: the camera's optical system will blur the image slightly and neighbouring cells in the detector will respond to the received impulse, see figure 2.8.

In all of our discussions, we have assumed that the sensitivity of the detector is uniform, i.e. all cells of the detector will respond equally to the same impulse. This is often not the case. The uniformity of a detector may be determined by capturing an image of a uniform scene (as described below). For accurate work, it may be necessary to derive a calibration function for each pixel.

Overflow or clipping of a signal occurs whenever the signal is of excessive amplitude. In such cases the signal might be clipped to the maximum value, or it might overflow resulting in the values being much smaller than they should. Overflow is not normally a problem with the solid state detectors currently used, but the automatic exposure employed does cause a loss of image detail in under-illuminated regions, as shown in figures 2.8 and 2.9.

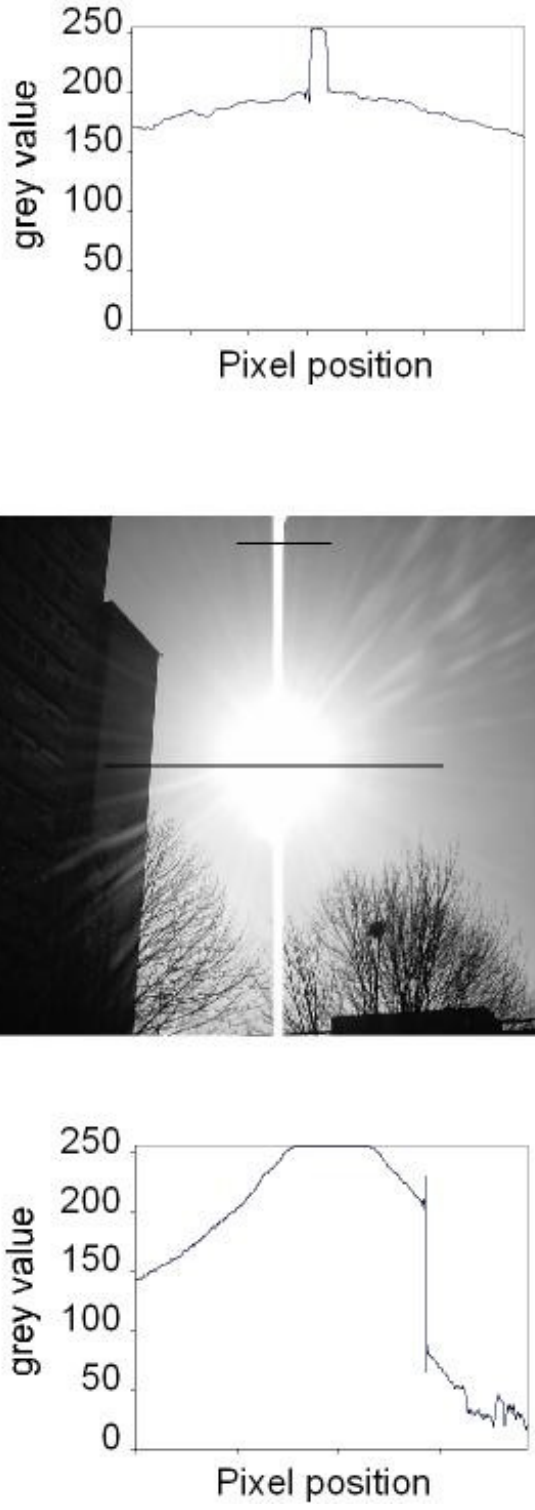


Figure 2.8: Blooming and saturation. The image contains a highly overexposed region. Pixels within this region have the maximum possible grey value, shown in the lower profile. The upper profile shows an exposure artefact.

Figure 2.9: Underexposure. Large regions of the image are underexposed due to the bright light source just out of view at the top of the image.

2.4.2 Signal to noise ratio

Noise in an imaging system is defined as any deviation of the signal from its expected value. It may be observed as speckle in the image where one would expect to see areas of uniform colour or brightness and many other effects.

Two types of speckle effect are seen; one is more common than the other. The less commonly observed type is called salt and pepper noise. Its effect is to randomly introduce pixels of pure black and pure white into the image. The more commonly seen type of noise is termed Gaussian noise; its amplitude distribution is described by a Gaussian or normal distribution. Gaussian noise is commonly observed in all systems, it arises due to randomness superimposed on the signal being captured or processed.

Gaussian noise is described by the Gaussian or normal distribution (figure 2.10). The noise will usually have zero mean and finite amplitude. The consequence of the image being corrupted in this way is that any pixel will have a value that differs from the expected one: it is equally likely to be increased or decreased and by an amount that is random but whose maximum is related to the deviation (width) of the distribution.

Video camera manufacturers will often quote their devices' signal to noise ratios. Whilst this is a useful quantity to know, it ignores half of the imaging system: the conversion of the video signal to an analogue format. If is more useful to measure the signal to noise ratio of the image capture system. Imaging a completely uniform scene and measuring the variation in the image data can achieve this.

Finding a completely uniform scene is not the difficult problem it might seem. A uniformly painted wall or a piece of card are suitable "scenes", provided that they are illuminated uniformly, or we may direct the camera at the sky, provided it is a uniform colour. It is also useful to defocus the camera viewing the scene slightly as this has the effect of reducing the influence of any non-uniformity in the scene. Having captured the image, we may measure the amount of noise in it by computing the image histogram (defined more rigorously in the following chapter). The histogram describes the frequency with which each grey level or each colour occurs in the image. The standard deviation of the histogram may be used as a measure of the noise amplitude. Figure 2.11 illustrates a uniform image captured using a low quality webcam and the image's histogram.

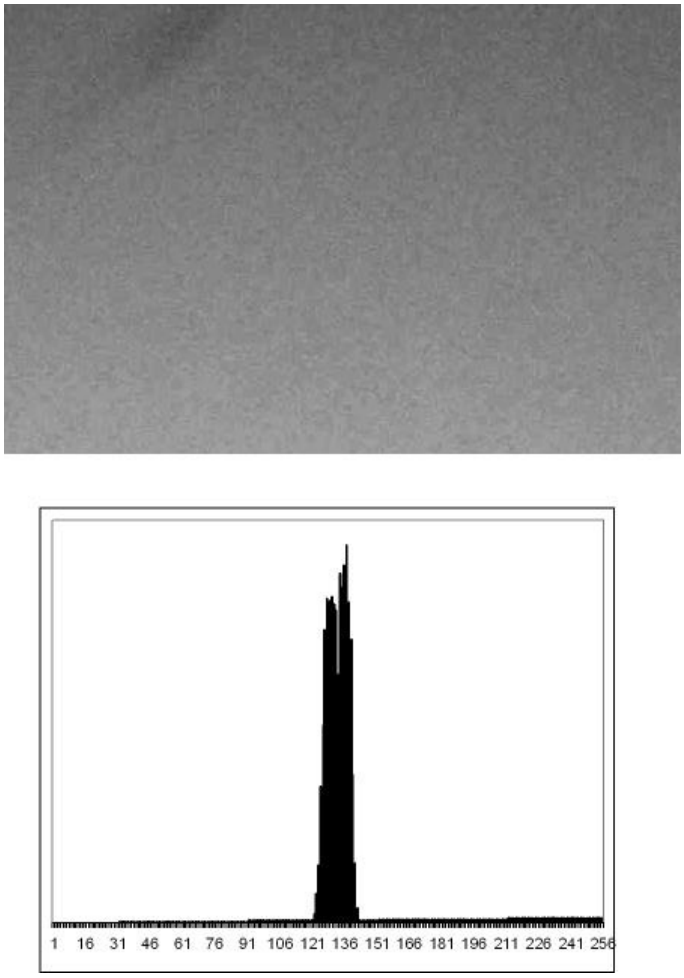


Figure 2.10: Gaussian (or normal) noise distribution.

A single measurement of the noise amplitude is not instructive, as it does not inform us of the influence of the noise on our data. This extra information can be determined by comparing the noise amplitude just measured against the amplitude of the noise free data. This comparison is called the signal to noise ratio, as its name suggests it is simply the ratio of the signal's amplitude (the maximum image value that could be observed) to the noise amplitude that we have measured. The ratio is usually quoted in decibels,

$$SNR = 20\log_{10}\frac{signal}{noise} \qquad 2.4$$

The signal to noise ratio also has consequences for the system's amplitude resolution that will be discussed below. The value used for *signal* is usually the maximum it can take: 255 in most instances.

2.5 Image resolution

When a vision system is being designed, the resolution of the images that are to be processed must be specified. This determines what objects can be seen in the data. Three resolutions must be defined: spatial (loosely the number of pixels), brightness (the number of shades of grey or colours in the image) and temporal (the number of frames captured or processed per second).

Whilst these resolutions are defined and discussed separately, they do have a mutual dependency which will also be discussed.

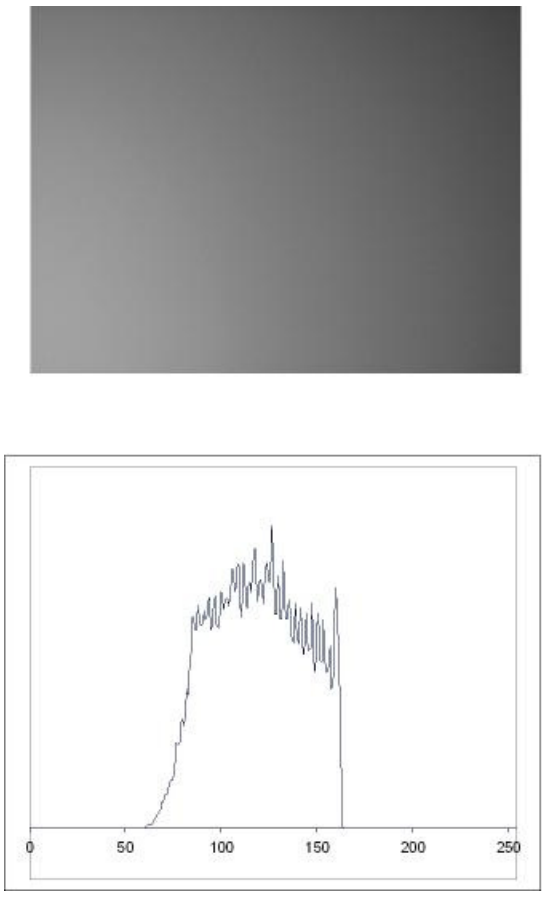


Figure 2.11: Image of a uniform scene captured using a low-quality webcam and the histogram of the image.

2.5.1 Spatial

It is tempting to define spatial resolution as the number of pixels in an image. Thus an image with 600 000 pixels is said to have a higher resolution than one with only 300 000. However, this figure is misleading as it ignores factors such as the distance of the object from the camera and the field of view of the system. Rather, the definition of spatial resolution must consider the interaction between the optics of the camera and the detector.

Figure 2.21 illustrates the interaction between a camera's field of view and the light sensitive element that converts incoming photons into an electrical signal. The field of view is the angle subtended by rays of light that hit the detector at the extreme edges of the image. To fully define the optical system, we must define two fields of view, in the planes containing the detector's edges and parallel to the camera's optical axis. If these angles are divided by the number of light sensitive elements across the detector in those directions (that is, the number of pixels in the digitised image in these directions) then we can compute the system's angular resolution. Although there are two angular resolutions, imaging systems are designed such that they are equal. An imaging system that has a horizontal field of view of 15° and 760 pixels across the image, will have a horizontal angular resolution of just over one arc minute ($1/60^{th}$ of a degree).

The angular resolution can be used to specify how close an object must be for it to be resolved (seen), or what objects can be resolved at a certain range. Figure 2.11 illustrates this relationship. The angular resolution, θ, measured in radians, is equal to the ratio of the smallest resolvable object, r, to the range, Z:

$$\theta = \frac{r}{Z} \qquad 2.5$$

Therefore, to resolve an object 2 mm in diameter at a range of 1 m requires a system with an angular resolution of 0.002 radians (0.1°).

An alternative way of specifying the spatial resolution of a system by-passes all consideration of the optics and uses a variant of Nyquist's theorem. Nyquist was the first to formally set a limit on sampling rates for representing a digital signal. He observed that to represent a periodic signal unambiguously required at least two samples per period (see figure 2.13). This observation can be translated into a requirement for a vision system: two pixels must span the smallest dimension of an object in order for it to be seen in the image.

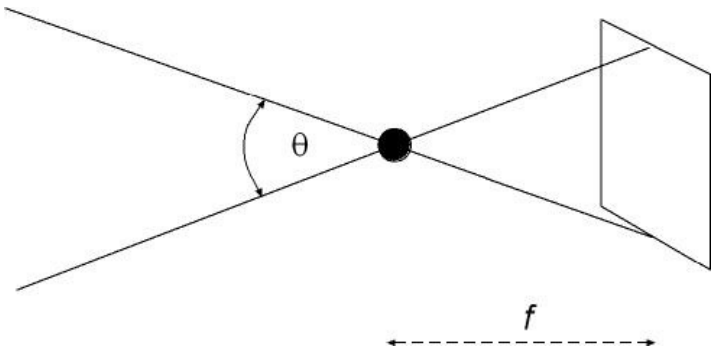


Figure 2.12: Relationship between field of view, θ, and camera's detector and focal length, *f*

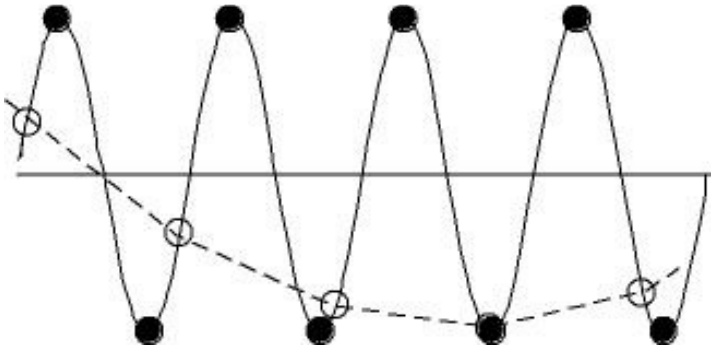


Figure 2.13: Sampling a continuous function at and below the Nyquist frequency.

If a signal is sampled at the Nyquist rate, little information is gained, other than knowledge of the presence of the signal. Similarly, if an image is sampled at this rate, the presence of objects of this size is determined. To discover more about the objects requires higher sampling rates. Figure 2.14 illustrates the effect of progressively increasing the number of samples that span a simple image. Reading across the diagram each image has four times the number of samples of the previous image, i.e. the number of samples in the images increases from *n* by *m* to 2*n* by 2*m* from one image to the next. It is apparent that objects can be discerned in the earlier images but not recognised until the later ones.

Figure 2.14: Sampling an image with progressively more pixels.

2.5.2 Brightness and colour

The brightness resolution of a system can be defined as the number of shades of grey that may be discerned in a monochrome image, or the number of distinct colours that may be observed in a colour image.

Given particular hardware, we may compute the brightness resolution of the system. Alternatively, we may investigate what a vision system is required to achieve and thus define the brightness resolution that will enable this task to be achieved.

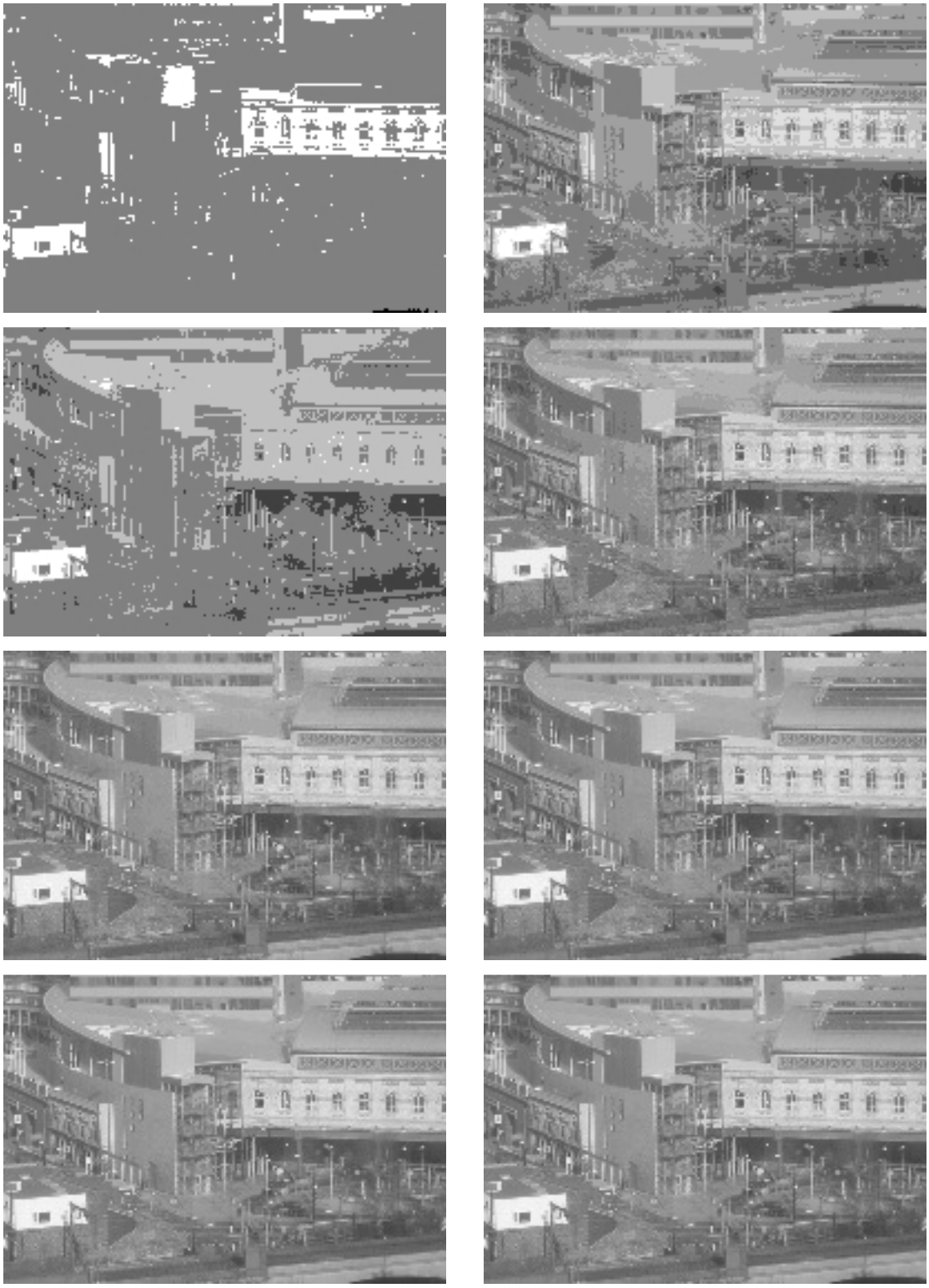


Figure 2.16: Sampling an image with progressively more grey values (bits) per pixel.

Recall that the signal to noise ratio was defined using the ratio of the noise amplitude to the maximum amplitude in an image. The major contributor of noise in a digital system is the camera; the analogue to digital conversion adds very little noise. If we assume that the noise in a system is described by a Normal (Gaussian) distribution, and the noise amplitude is measured as the standard deviation of this distribution, then our knowledge of statistics tells us that the probability of observing noise with amplitude more than three standard deviations from the mean is approximately 0.001, implying that there is a one in one thousand probability of observing a noise deviation with amplitude outside of this range, which is an unlikely occurrence. Therefore, if the total brightness scale is divided into bins of

size six times the noise amplitude, then we can be sure that a sample that falls into the centre of one bin cannot be confused with a sample that falls into a neighbouring bin due to noise perturbing one value sufficiently for it to be confused with the other, figure 2.15.

As an example, consider a black and white camera that has a signal to noise ratio of 55 dB. This equates to a ratio of signal to noise amplitudes of 562. If this magnitude is divided by six, we obtain the maximum number of grey values that this camera can give, without confusion: 94.

Alternatively, by viewing typical images using varying numbers of shades of grey or colour we may also estimate the number of grey values required in an application. This is illustrated for a monochrome image in figure 2.16; the first image has two shades, the second four, and the final one 256 shades. It is apparent that more detail becomes visible as the number of shades is increased. False contours are visible in areas of gradually changing brightness, when there are insufficient shades of grey. The last few images do not show any apparent differences. Although there is a limited resolution in the reproduction, this effect is due to the human visual system being unable to differentiate more than about 40 shades of grey. This is often given as justification for having this as a lower limit to the number of shades, but it ignores the requirement to allow for latitude in exposure.

It should be noted finally, that image capture devices are limited in their capacities by what the manufacturers will provide. Most monochrome images are captured using eight bit values, and colour images using three eight bit values.

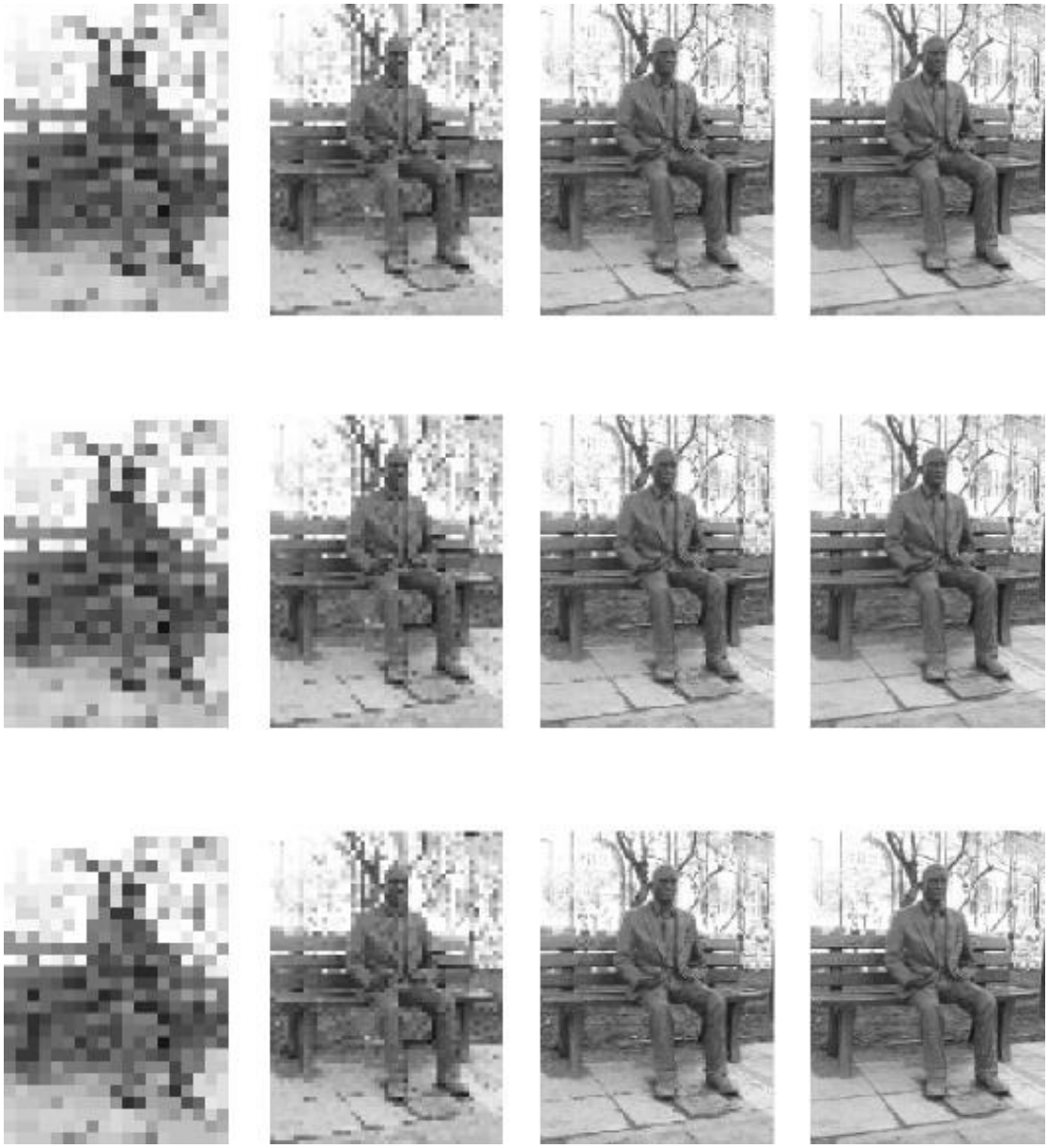
2.5.3 Temporal

Temporal resolution is defined either as the number of images that are captured and stored per second, or as the number of images that are captured and processed per second. When capturing analogue video, the national standards dictate that a maximum of 25 frames per second are available (or 30 in the US). Digital video data may be captured at higher frame rates, the exact rate will depend on the rate that data may be transferred between the camera and the host computer and the size of the images that are captured.

When processing digital video, a lower limit on the rate at which results should be calculated can be set if we require the appearance of real time processing. Psychological studies have suggested that this rate is in the region of twelve frames per second.

2.5.4 Interactions

It is helpful to consider spatial and brightness resolution separately, to avoid confusing the effects of these causes. However, the two resolutions can interact in affecting the overall perception of an image: a poor spatial resolution can sometimes be compensated for by a good brightness resolution. In fact for each brightness resolution it is usually possible to define a threshold spatial resolution above which the image appears of an acceptable quality and below which it does not. The combinations of resolutions will define a border between acceptable and unacceptable qualities. This border will be different for all types of images, as an example, consider the images of figures 2.17 and 2.18. In each of these diagrams we have combinations of differing brightness and spatial resolutions that represent different points in the resolution space. It is apparent that somewhere along a diagonal of these diagrams lies the border between the acceptable and unacceptable images.



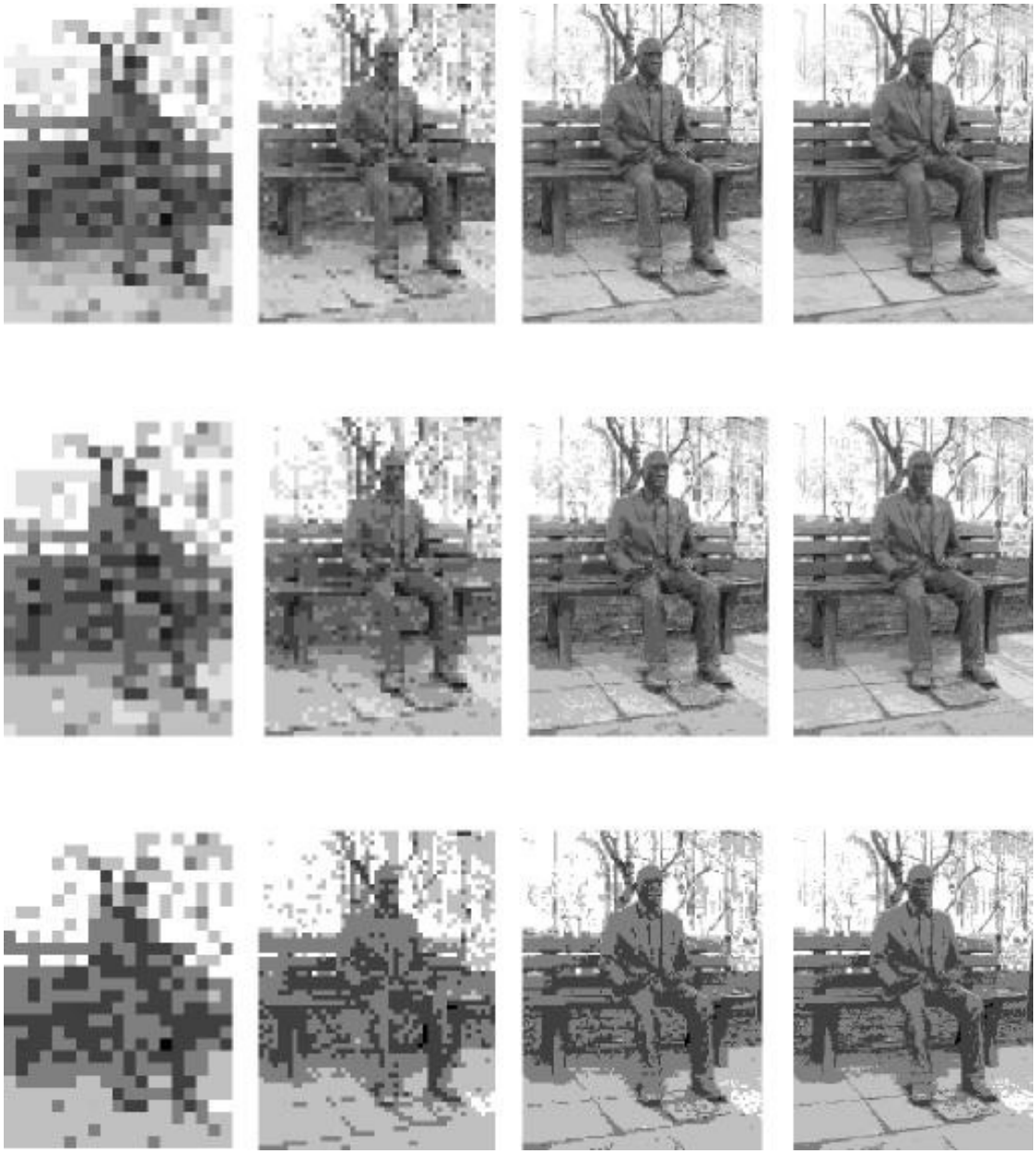


Figure 2.17 (Caption below)

Figure 2.18: Images presented with varying sample densities and grey values. Each line of the diagram has half the number of grey values per pixel as the line above. The columns have 10%, 25% and 50% of the samples per unit length of the images in the rightmost column.

2.6 Connectivity and distance

In many cases when processing a digital image, one must decide whether one pixel is connected to another, that is, can one pixel be reached from another by tracing a path between connected pixels? The question can be reduced to "how do we decide whether two adjacent pixels are connected or not?"

Figure 2.19a illustrates the problem. Is the central pixel connected to the four nearest neighbours (north, south, east and west) or to all eight nearest neighbours? If four-connectivity is assumed, then the diagonal line of shaded pixels in figure 2.19b is not connected, and the underlying line will not be detected. However, the two regions on either side of the diagonal line will not be connected, since no two of the closest neighbours are adjacent, which is intuitively the correct answer.

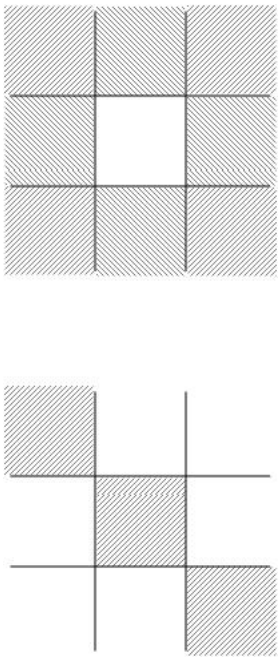


Figure 2.19: Pixel connectivity.
a) a 3 x 3 grid f pixels: is the central pixel connected to the four or eight nearest neighbours?
b) a diagonal line of shaded pixels: are the pixels on the line and either side of the line connected?

If eight-connectivity is assumed, that is a pixel may be connected to the four nearest neighbours and the four next nearest, then the diagonally adjacent pixels of figure 2.19b are connected. However, the two regions on either side of the line are also connected, since pixels in one region are diagonally adjacent to pixels in the region on the line's other side.

The solution to this contradiction is to be flexible in choosing a connectivity model. For some purposes four-connectivity is appropriate, for others eight-connectivity is a better approach.

Similarly, there are a multitude of methods of measuring distances in digital images. The Euclidean distance, *D*, between a pair of pixels (co-ordinates (*x,y*)) is measured by Pythagoras' theorem:

$$D(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad 2.6$$

In some cases an approximation to this distance may be made, the so-called city-block distance:

$$D(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| \qquad 2.7$$

Which distance measure is used will be a compromise between speed of computation and accuracy.

Other measures of distance have been suggested, but are not widely used.

2.7 Colour representation models

Young's tristimulus theory was mentioned above. An important consequence of the theory is that it suggested that colours could be synthesised by suitable combinations of a red, a green and a blue primary colour. This was formalised by the CIE (Commission Internationale d'Eclairage) who, in 1931, specified the spectral characteristics of the red, R, green, G and blue, B, primary sources to be monochromatic light of wavelengths 700nm, 546.1 nm and 435.8 nm respectively. These light sources could be combined linearly to match almost any colour, C:

$$C = rR + gG + bB \qquad 2.8$$

Where *r*, *g* and *b* denote the amounts of each primary. The relative amounts of these primary sources required to match the range of visible colours was shown in figure 2.3. Note that negative amounts of

the red primary are required to match certain colours. To overcome this problem the CIE defined the curves shown in the graph sketched in figure 2.20. These show the amount of each idealised primary colour, X, Y and Z, which is required to match any arbitrary test colour. It is common for the amounts of X, Y and Z to be normalised and to present the two of them, as the so-called CIE chromaticity diagram. In this diagram, the values of *x* and *y* are presented:

$$x = \frac{X}{X+Y+Z} \quad and \quad y = \frac{Y}{X+Y+Z} \qquad 2.9$$

The diagram is shown in figure 2.21. Around the outside of the horseshoe are shown the equivalent wavelengths of monochromatic light that are generated by these combinations of the CIE primaries. White lies at the co-ordinates (1/3, 1/3).

Although using the RGB colour space is convenient as colour images are captured in this format, it is not the only colour representation method, and is indeed probably not the best. A significant problem with the RGB colour space is that it is perceptually non-linear. This means that if we change a colour's RGB value by a fixed amount, the perceived effect it generates is dependent on the original RGB values. It is also difficult to predict the change that will occur to a colour by changing one of its components by a certain amount: what colour will result if we change the red component of the colour represented by the RGB triplet (0.57, 0.37, 0.06) by 0.30?

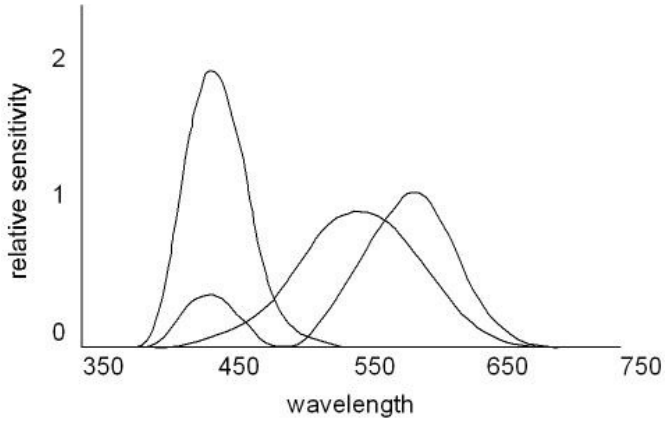


Figure 2.20: CIE colour matching functions for the standard colourimetric observer

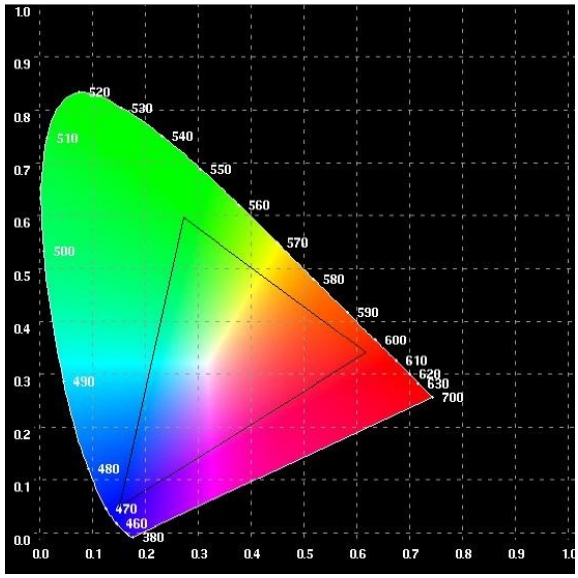


Figure 2.21: CIE chromaticity diagram. The range of colours realisable by a standard monitor is indicated by the triangle. Numbers around the horseshoe indicate the wavelength of monochromatic light of the equivalent colour.

It was this type of consideration that led to the adoption of perceptual colour spaces. One of these, HSV, uses three components that represent

- the intensity of the sample or its brightness (V),
- the underlying colour of the sample, the hue (H),
- and the saturation or depth of the sample's colour (S).

Thus reddish colours will have similar hue values but will be differentiated according to their saturations. The HSV colour space is represented by the cone of colours in figure 2.22.

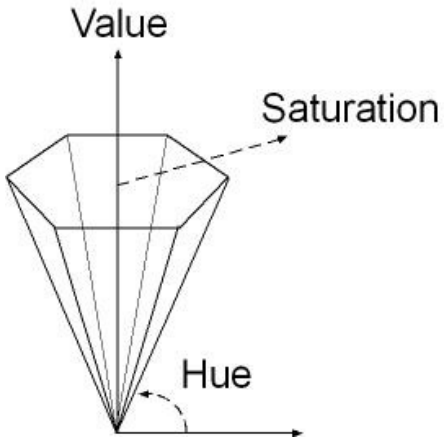


Figure 2.22: Definition of HSV colour space.

Finally, it should be noted that colour data is broadcast using colour difference values. These are again a triplet of values, one represents the brightness to be assigned a pixel, the other two components represent colour difference values.

2.8 Camera calibration

In the final section of this chapter we examine the problem of camera calibration, that is, how can we relate the co-ordinates of pixels to co-ordinates in a frame of reference in the real world?

The calibration process divides into two stages, one that relates image co-ordinates to a co-ordinate frame aligned to the camera, and one that relates the camera co-ordinate frame to an arbitrary co-ordinate frame fixed to some point in the physical world. Related to these two transformations are two sets of parameters, extrinsic parameters relate to the camera to world co-ordinate transform and intrinsic parameters relate to the image to camera co-ordinate transform (figure 2.23 summarises the process).

The extrinsic parameters simply define the translation and rotation of the camera co-ordinate frame such that it aligns with the world co-ordinate frame. In principle it is sufficient to determine the location of the camera's optical centre with respect to the origin of the world co-ordinates and the orientation of the camera's optical axis and image plane with respect to the world co-ordinate frame's axes, figure 2.24. (The camera's optical centre is the point in the optical system through which all rays of light will pass undeviated. The optical axis is the line that is perpendicular to the image plane and passes through the optical centre.)

The intrinsic parameters of the camera are slightly more complex. We shall consider two cases, one in which there are no optical aberrations, i.e. the imaging process does not distort the image, and the other case in which distortions are introduced.

If the imaging process does not distort the image and the image is sampled uniformly in the x and y directions, then a linear relationship exists between the image and camera co-ordinates, we must offset the origin, (o_x, o_y), (since the camera co-ordinate system's origin coincides with the centre of the image) and scale the data (by s_x and s_y along the x and y axes):

$$x = -(x_{im} - o_x)s_x$$
$$y = -(y_{im} - o_y)s_y \qquad 2.10$$

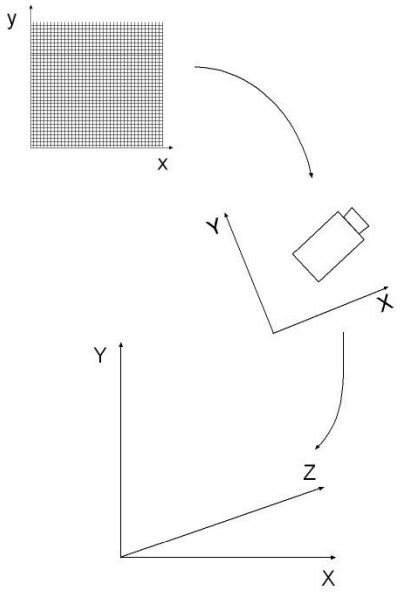
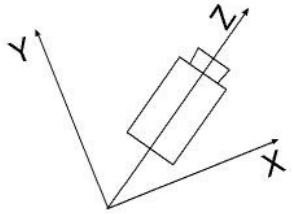


Figure 2.23: Co-ordinate frame transformations: image → camera and camera → world.



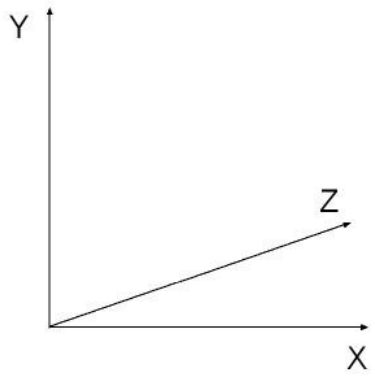


Figure 2.24: World to camera transformation parameters.

If the imaging process does distort the data, it is usually sufficient to assume that a first order spherical correction is sufficient:

$$x = \left(1 + k_x\left(x^2 + y^2\right)\right)x_d$$
$$y = \left(1 + k_y\left(x^2 + y^2\right)\right)y_d \qquad 2.11$$

The *d* subscripts refer to distorted co-ordinates, *k* are the distortion coefficients. This correction must be applied prior to the linear shifting and scaling. Its purpose is to remove the warping that is inevitable when an image is generated. Figure 2.25 illustrates its use; the original image plus the corrected one are shown. Dewarping is a topic that will be examined in greater detail in the following chapter.

Figure 2.25: Unwarped and warped images

Examples of the use of camera calibration will be examined in future chapters.

2.9 Summary

In this chapter we have attempted to lay foundations for the rest of the text. We have introduced image processing by analogy with the human visual system. We have examined the general principles that must be considered when specifying what data is to be captured and how for a specific task. Generally, these requirements reduce to the angular resolution of the system, the number of pixels an image should contain and the number of discrete brightness or colour values. Camera calibration was also considered.

2.10 Bibliography

Newton (1931), Young (1807) and Helmholtz (1881) laid the foundations for colour perception.
Gregory (1997) and Frisby (1979) present readable accounts of visual perception.
Hecht (1998) provides a modern introduction to optics and optical processes.
Stroud (1956) and others gave a justification for interpreting “real-time” from a human perspective.

Haralick and Shapiro (1992) discuss most aspects of camera calibration.

2.11 Exercises

1. Take an image and experiment with the interrelationship between spatial resolution (number of pixels) and dynamic range (number of grey values). Is it possible to identify a dynamic range at each spatial resolution at which the image's subject matter becomes recognisable? Repeat the experiment with images of different subject matter (faces, outdoor scenes, etc.).

2. An image of a fence is captured. The fence is made of panels 10cm wide separated by a gap of 10cm. A length of 20m of fence appears in the image. To what minimum resolution should the image be captured in order to observe the fence without error? Describe the errors that will be introduced if the data is sampled to a lower resolution.

3. Take a noise free image and add random noise of varying amplitudes that affects a varying numbers of pixels to create a sequence of images with varying signal to noise ratios. Score each of the images on a scale of one to five according to the image's perceived quality. What relationship can you discover between quality, signal to noise ratio and number of pixels affected?

Chapter 3
Image Transforms and Feature Extraction

The subject of image processing can be defined to include all operations that manipulate image data. These will range from simple operations that manipulate pixel's values, to more complex ones that transform an image from one representation into an alternative.

Many schemes have been proposed for classifying image transformation algorithms. Some authors organise them according to the area of the input image that is involved: whether it be individual pixels, small groups of pixels of the whole image. Others use the sequence of operations that commonly occur when images are processed: algorithms that correct the image for any defects introduced during capture are presented first, and so on. In this treatment, we are taking the former option.

Feature extraction is defined as any operation that extracts information from an image. This information would normally then be used in further processing stages of a system to identify objects etc.

After studying this chapter you will be familiar with a range of commonly used methods for processing images and extracting information from them. Feature extraction is an important topic that will reappear in the following two chapters.

3.1 A classification of image transforms

An image processing operation is defined simply as an operation that somehow manipulates image data to generate another image. The size of the image will not be altered, but the size of each storage unit (pixel) might, see figure 3.1. A large multitude of image transforms has been defined, and they can be highly confusing when presented to the novice all at once. All authors therefore attempt a systematic presentation. Some classify algorithms according to their task, thereby creating a recipe book of operations. This has its advantages in that if we are able to identify the task we are attempting to perform, then we can discover what algorithms could perform it. The approach does however assume some prior knowledge: the reader is expected to be able to determine what tasks are necessary when designing a system; this is probably beyond the ability of a novice approaching this subject for the first time. It is also possible to present algorithms according to their place in the processing hierarchy. Thus the description would follow the logical order of the algorithm's use. This is what we have used in the organisation of this text where we have chapters devoted to image manipulation, feature extraction and feature recognition, but it is not necessarily appropriate when investigating image manipulation itself. Rather, in this chapter we have organised image processing functions according to the area of the input image that is involved. Thus we discuss point transforms that involve a single pixel; local transforms that involve a small region of pixels surrounding the one of interest and global transforms that generally involve the entire image in computing the transformed value of a single pixel.

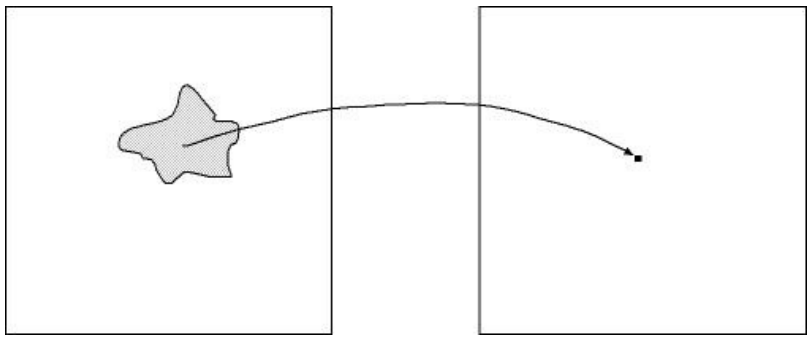


Figure 3.1: Image to image transformation, possible changes and invariants.

The aims of the image transformation being discussed are as follows:

- to remove or correct for degradations introduced by the image capture process,
- to improve the appearance of the image for human perception or for further processing,
- to identify and quantify structures in the image that may be indicative of the objects in the scene being viewed and
- to transform the image into an alternative representation in which some operations may be performed more efficiently.

After studying this chapter you should be able to design and implement systems that improve the appearance of images: either by removing exposure defects, or correcting for optical aberrations. You should also be able to implement simple enhancement systems that will form the basis of the further processing to be discussed in subsequent chapters.

3.2 Point transforms

As discussed above, a point transform will manipulate the grey or colour value of an individual pixel without considering the neighbouring values (see figure 3.2). Since the only information that an individual pixel carries is its brightness or colour, it is only this information that can be manipulated. This is achieved by a mapping between input and output grey values, see figure 3.3. The mapping must define an output value (along the y axis) for each input value (along the x axis). Every input value must have an associated output, but this need not be unique: more than one input might share the same output.

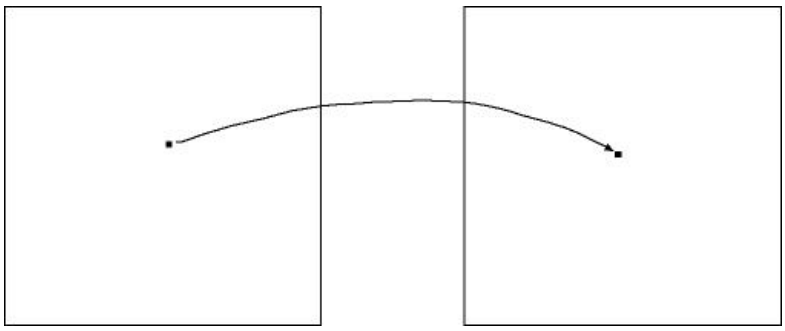


Figure 3.2: Point transform.

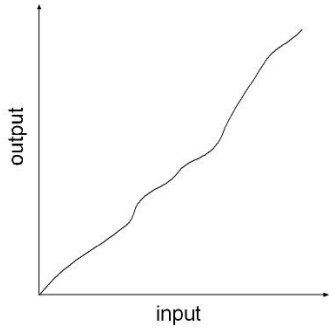


Figure 3.3: Mapping between input and output values.

The material of this section deals with how various mappings might be determined.

3.2.1 Grey scale manipulation

The grey values of an image may be manipulated in many ways to modify the image's appearance. The methods that are applied are generally interactive, that is, the user of a vision system must decide how the modification is to be performed, rather than have the system determine the mapping automatically.

The two simplest methods of modifying the image are to add a constant and to scale the values by a constant multiplier, figure 3.4 and 3.5. This might be necessary if, for example, the exposure of the image had been incorrect. It is important to emphasise that this will not alter the information content of the image but it will make that information easier for a human observer to perceive.

These methods are quite arbitrary: how much should be added to the image or what scaling factor ought to be used is determined specifically for each image being processed according to the desired outcome. To reduce this arbitrariness, the images are often manipulated such that the mean grey level is matched to a previously determined value. Thus we would add an amount equal to the difference between the target and actual grey values, or scale by a factor equal to the ratio of the target to actual values. Some applications also scale the image such that the variance of the data takes a standard value. Such normalising operations, although simple, are quite important in removing a source of variation when we're trying to recognise patches of an image.

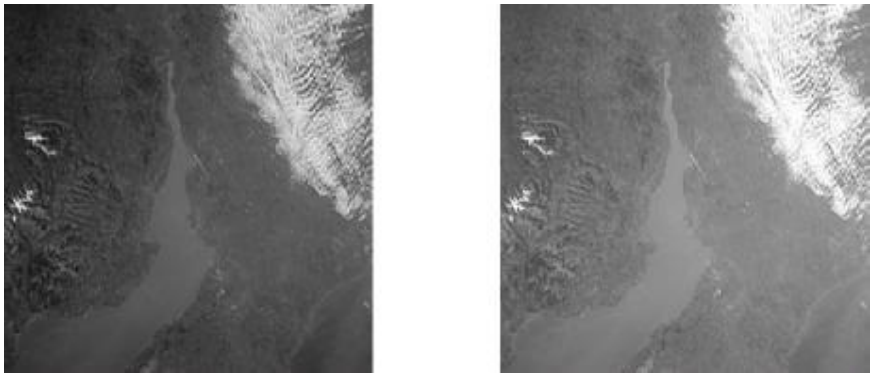


Figure 3.4: Adding a constant to each pixel's value.

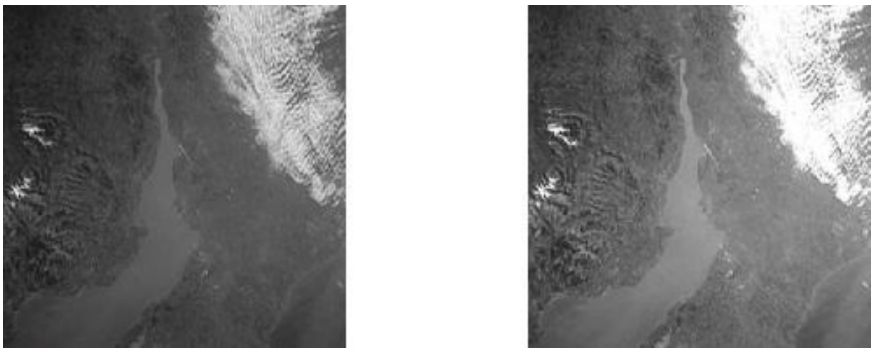


Figure 3.5: Multiplying each pixel's value by a constant

The grey scale of the image can also be manipulated interactively such that certain portions of it are expanded at the expense of others that are compressed. This must be done manually as the ranges to be manipulated cannot be determined automatically. The mechanism of this process is illustrated in figure 3.6a. The left column indicates the input grey scale and the right column the output. Without any manipulation, the columns will match exactly, grey values on one column map to the same grey value

on the other. The user may choose to connect values on the two columns to indicate that they should be mapped to each other: in this example value *a* has been connected to value *a'*, indicating that the input grey value *a* is mapped onto the output grey value *a'*. The extreme values of the scale, normally 0 and 255, will always remain unchanged. Thus the input values between 0 and *a* will be mapped onto the output values between 0 and *a'*. Since in this example, *a'* is less than *a*, this range of grey values has been compressed, which introduces the possibility that all or part of the range from *a* to 255 could be expanded. The user has also connected values *b* and *b'*, which indicates that the values in the range [*a*, *b*] will map to the values [*a'*, *b'*]. And since the range [*a'*, *b'*] contains more grey values than the input range, there is an expansion of the grey scale over this interval, and because the data is discrete, there will be values in [a', b'] that are not used. Finally, the input range [*b*, 255] is mapped to the output range [*b'*, 255]. Figure 3.6b indicates the grey scale mapping that is generated and figure 3.7 an image manipulated with this transform.

The aim of manipulating the grey map in this way is to expand certain portions of its range and thus make the objects that have this range of brightnesses appear more clearly in the manipulated image. The problem with the techniques discussed is that they require an operator to recognise what range of brightness should be expanded. Two methods exist that achieve the same aim automatically.

False contouring applies a grey scale map of the type shown in figure 3.8 to an image; the result is shown in figure 3.9. The aim of this transformation is to enhance gradual changes in brightness by changing them into a succession of rapid changes from dark values to bright values. An alternative is to map the input grey values into a set of colour values, a rainbow set of colours, gradations through shades of yellow or blue have been used, figure 3.10 presents an example using shades of red and orange.

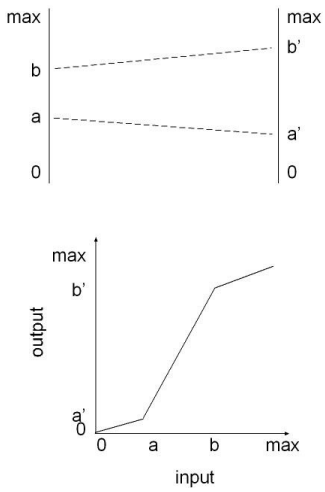


Figure 3.6: Interactive grey scale manipulation.
a) mapping between input and output values,
b) grey scale mapping

Figure 3.7: Image manipulated using this transform

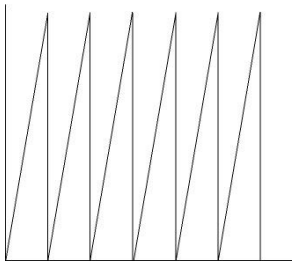


Figure 3.8: Sawtooth mapping

Figure 3.9: False contouring, the image has had its grey scale mapped according to figure 3.8

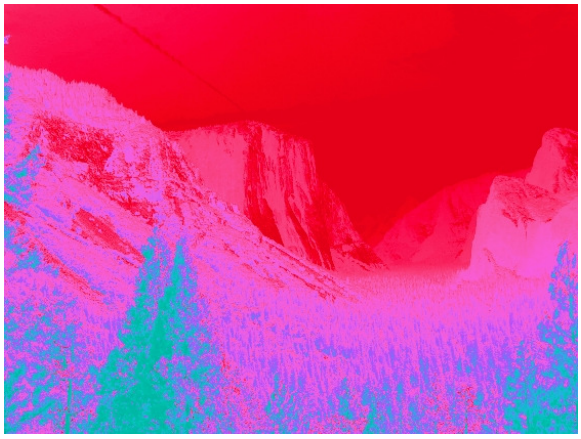


Figure 3.10: A false colour mapping.

3.2.2 Thresholding

From the discussion of the previous section, it should be apparent that many of the grey value manipulation algorithms are designed to improve the appearance of an image, or to make certain structures more obvious. One family of techniques that do this is so important that it deserves a separate discussion, and that is Thresholding.

Thresholding is the operation that transforms an input image, be it a monochrome or colour one, into a binary image, that is an image whose pixels can take one of two values only. The following discussion is of monochrome data only, but the methods are readily adapted to colour data. To perform thresholding, a threshold value, θ, is defined. The thresholding operation is achieved by comparing each pixel value, *f(i, j)*, against the threshold and setting the output appropriately (*g(i,j)* is the output image):

$$g(i,j) = \begin{cases} 0 & \text{if } f(i,j) \le \theta \\ 1 & \text{otherwise} \end{cases} \qquad 3.1$$

Thresholding has a multitude of uses in image processing, mainly in making decisions regarding the identity of objects or pixels. Crucial to the operation is deciding a suitable value for θ. This will be discussed in a later chapter.

3.2.3 Histogram manipulation

Transforming the grey scale of an image using the methods defined above is not necessarily an intuitive process. By manipulating the histogram we may achieve certain well defined and well founded results. Notable amongst these manipulations is histogram equalisation.

Histogram equalisation is based on the argument that the image's appearance will be improved if the distribution of pixels over the available grey levels is even. This transform would change the grey level histogram of figure 3.11a, which is shown with the image from which it was derived, into the image and histogram of figure 3.11b. Two things are apparent: one is that the histogram is not even, the second is that the image's appearance has been improved in some respects.

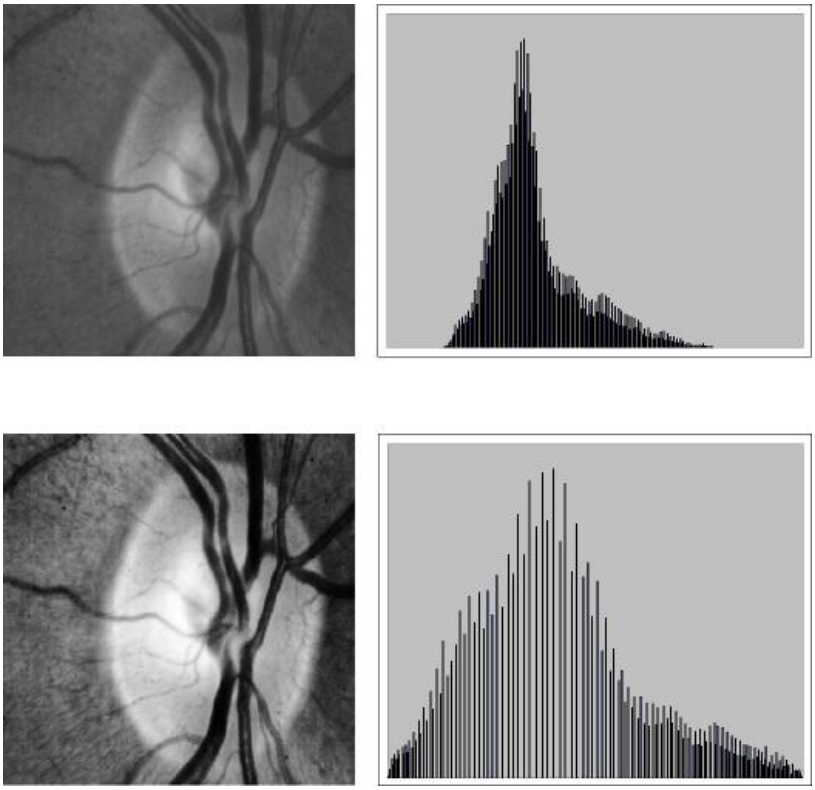


Figure 3.11: Histogram equalisation.
top) image and its histogram,
bottom) equalised image and its histogram.

The transform is derived from the cumulative histogram which is, in turn, derived from the image's grey level histogram. The grey level histogram records the frequency of occurrence of each grey value in the image. The cumulative histogram records the number of pixels in the image with grey values less or equal to each index. The cumulative histogram, *C(i)*, is derived from the grey level histogram, *g(i)*:

$$C(i) = \sum_{k=0}^{k=i} g(k) \qquad 3.2$$

One interpretation of the equalisation transformation is that it warps the horizontal axis of the histogram, stretching it in some regions and compressing it in others, such that the frequencies become more uniform. The compressed regions correspond to those grey levels that are less occupied than they should be, as shown in figure 3.12.

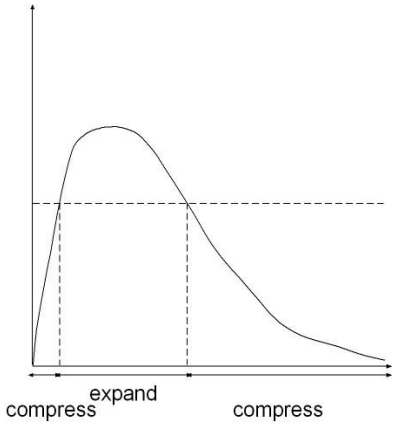


Figure 3.12: Expanding and compressing a histogram to make it more uniform.

In the ideal case the equalised image will contain an equal number of pixels having each grey value. This number will be the ratio of the number of pixels in the image, *N*, to the number of grey values, *l*: *N/l*. Thus, the j^{th} entry of the cumulative histogram will have the value *jN/l*. This must be equated to an entry in the input image's cumulative histogram, *C(i)*. This will define the mapping between input, *i*, and output, *j*, grey values that achieves equalisation, so we may equate:

$$\frac{jN}{l} = C(i) \qquad 3.3$$

And rearrange:

$$j = \frac{l}{N} C(i) \qquad 3.4$$

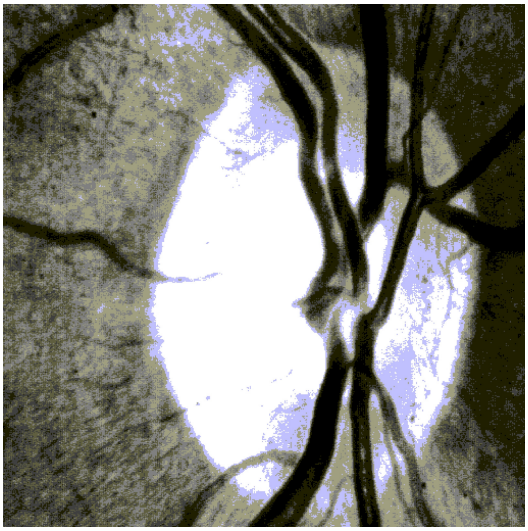


Figure 3.13: Example of the effect of equalisation.

One proviso must be observed, the range of grey values is 0 to *l*-1, and the maximum value of *j* derived using this formula is *l*. It is therefore appropriate to subtract 1 from the result to obtain the correct

maximum. The consequence of this however, is that the minimum value of j could be -1 rather than zero. This is readily corrected. The full expression for the transform is:

$$j = \max\left(0, \frac{l}{N} C(i) - 1\right) \qquad 3.5$$

It has also been suggested that equalising the histogram is not the best method of improving the image's appearance. Rather, the histogram should be hyperbolised, figure 3.13. Despite the apparent improvement in the data, this operation is rarely used.

3.3 Local transforms

Local transforms are those that are computed using a small region surrounding the pixel of interest, figure 3.14. "Small" is obviously a term that must be interpreted flexibly as there could be some overlap between these and global transforms discussed below, but it can be assumed that the region involved in a local transform is much smaller than the image.

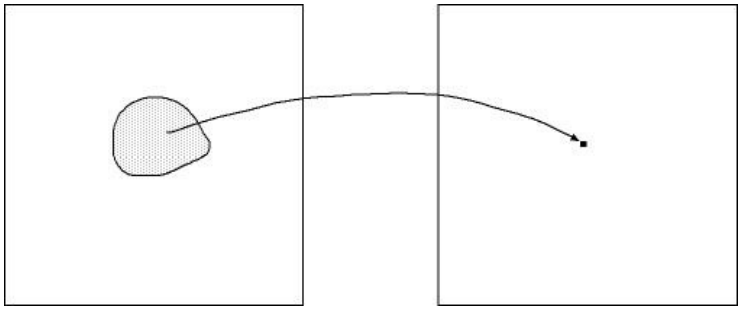


Figure 3.14: Local transformations.

3.3.1 Convolution

This operation is fundamental to local transforms. Mathematically it is defined by equation 3.6:

$$g(r,c) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(r-x, c-y)t(x,y) \qquad 3.6$$

The effect of this operation is that a template, $t(x,y)$, is placed over all possible image locations. At each location, the product of an image value and the overlapping template value is computed. The products are summed to give the output value at that location. Optional further steps are to subtract an offset, to guarantee that the minimum output value is zero and to scale the result such that the maximum value will lie within the range allocated to an image value. The offset to be subtracted would be the smallest result that could be computed. The multiplicative scaling factor could be the ratio of the maximum image value to the largest result that could be expected or it could be the sum of the template values.

Whilst the limits of the expression extend to $\pm\infty$ in both x and y directions, this does not pose a problem in practice. Firstly the image values are undefined (zero) outside of the image region, secondly, the same can be said of the template, and the template is generally quite small in comparison to the image. Thus the dimensions of the template define the practical limits of the summation. As an example of convolution, the image of figure 3.15 illustrates the result of convolving an image and a small template of nine elements.

Expressed in equation 3.6, convolution requires n^2 multiplications and additions per pixel, where n is the size of the template. Obviously, as n increases, n^2 will rapidly become large, indicating that

alternative methods of computation are desirable. Two methods are used. One of these makes use of the Fourier transform and is outside the scope of this course. The other method can be used if the template can be separated into two simpler, one dimensional templates. Thus a convolution of the image with one n by n element template is reduced to convolution with a 1 by n template and an n by 1 element template. In this way, n^2 additions and multiplications are reduced to two $2n$ pairs of operations.

Separating the template may or may not be a simple operation. For example, figure 3.16 shows the separation of two frequently used templates that can be derived analytically. Figure 3.16a is the separation of the Laplacian template; convolution of an image with this template is equivalent to double differentiation. The combination of the separated templates is intuitively equivalent to the original template, as will be discussed in the following paragraph. Figure 3.16b shows the separation of the Laplacian of Gaussian (LoG) template, which is used in edge enhancement and will also be described below. Since LoG templates can be large, it is desirable to find efficient ways of realising this convolution. The template is defined analytically, as is its separation: the diagram shows an oblique view of the two-dimensional template and profiles of the two one-dimensional templates.

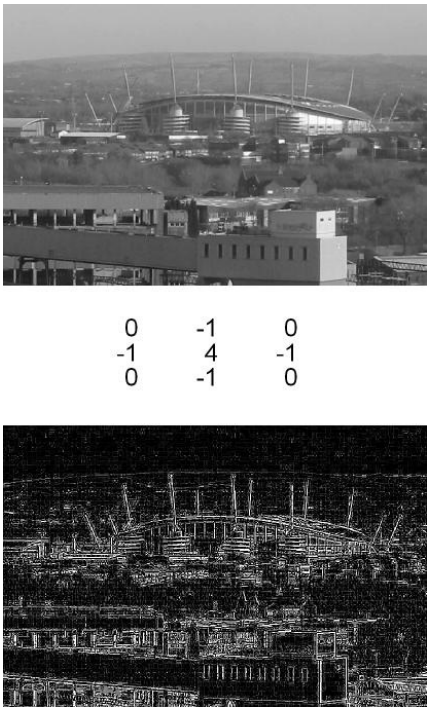
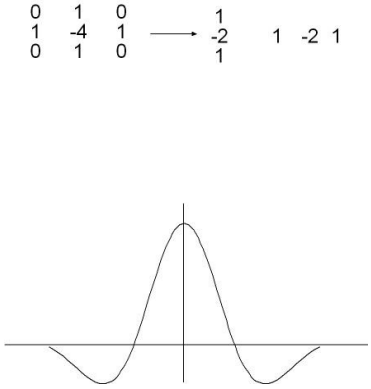


Figure 3.15: Simple example of convolution



3.16: Separable templates:
top) Laplacian operator,
bottom) Laplacian of Gaussian

Separating a template into two one-dimensional operations is only possible due to the distributative nature of convolution. That is, convolution of three templates can be performed in any order:

$$A \otimes (B \otimes C) \equiv (A \otimes B) \otimes C \qquad 3.7$$

Therefore, if B and C are the two separated templates, their convolution is equal to the combined template. And the result obtained by convolving the image, A, with B (*n* multiplications and additions) and convolving the intermediate result with C (a further *n* multiplications and divisions) is identical to what is obtained by the original method.

The usefulness of the convolution operation lies in the results that it generates. Four applications will be covered, smoothing, sharpening (also known as edge detection), corner detection and template matching (in a later chapter).

3.3.2 Smoothing

The effect of smoothing on an image is to remove sharp, sudden changes in the brightness function. These might be caused by noise in the image capture device or small objects in the scene that might obscure the larger objects of the scene. The result of smoothing an image is another image with an improved signal to noise ratio, or an image in which the effect of distracting artefacts has been reduced.

The value of a smoothed pixel, *S*, in an image is computed by combining a selection of the neighbouring pixels, *P*. The simplest method of achieving the combination is simply to average the values in the neighbourhood, *N*:

$$S(i,j) = \sum_{u,v \in N} P(u.v) \qquad 3.7$$

This can also be achieved by convolving the image with an appropriate template. The neighbourhood that is to be used defines the size and shape of the template. The template values are simply the reciprocal of the number of values included in the neighbourhood. So a neighbourhood of nine pixels would use template values of 1/9. Setting all template values to one and scaling the result of the convolution by 1/9, we achieve the same effect more efficiently.

In figure 3.17, a three by three element template is used to smooth an image, before and after versions are shown to illustrate the effect of the process. It is apparent that the smoothing has achieved the required effect, in that random noise fluctuations have been reduced. This is especially apparent in the smoother regions of the image. It is also apparent that some blurring of the image has taken place: sharp edges are no longer as sharp as they were. This is an undesirable side effect.

Figure 3.17: Smoothing an image using a 3x3 element template/kernel

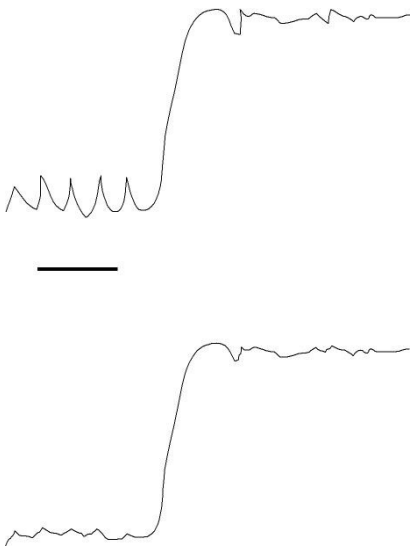


Figure 3.18: Schematic of thresholded smoothing – profile through an edge.

Figure 3.19: Example of thresholded smoothing. The lower image is a smoothed version of the upper one, smoothing has been performed only where the difference between the suggested smoothed value and the input value is below a threshold, in this case 5. Subtle differences are apparent in the vicinity of sharp discontinuities, e.g. window frames.

That this operation reduces the amount of noise in an image should not be a surprise. Earlier, some of the properties of image noise were defined. Amongst them was the property that the mean amplitude of the noise was zero. Therefore, the sum of a set of noise values is likely to be closer to zero than the original values.

The deleterious blurring may be reduced by conditional smoothing. In this variant of the smoothing algorithm, the smoothed value, *S(i,j)*, is computed, but it will only replace the original, *P(i,j)*, if it differs by less than a preset value, δ, equation 3.8.

$$S(i,j) = \begin{cases} S(i,j) & if\ |S(i,j) - P(i,j)| < \delta \\ P(i,j) & otherwise \end{cases} \qquad 3.8$$

The reason for this may be understood by inspecting figure 3.18 which shows a profile through an image feature. The brightness values are markedly different on either side of the feature; they are uniform but are corrupted by noise. The horizontal bar beneath the profile indicates the size of the smoothing neighbourhood. When the neighbourhood lies entirely within either of the uniform regions, the smoothed value and the value that might be replaced are similar; they will differ by less than the threshold value that is selected. However, when the neighbourhood overlies the image feature, the smoothed and original values will be more different and smoothing will not occur. Thus, the original image features should be preserved, without smoothing, as shown in figure 3.19. The minimum amplitude of the feature that is preserved is dependent on the threshold that is selected and the size of

the neighbourhood.

These two operators are reasonably efficient, although they do not necessarily produce the best results. The so-called rank filters are often used in preference, as they yield a better result but at a higher computational cost.

Figure 3.20: An image smoothed by median filtering

A rank filter will select one value as the smoothed value from the set of pixels in a neighbourhood that have been ranked according to their magnitudes. For example, the median filter will select the value that is central in the ranked list or the average of the two central values. Figure 3.20 illustrates this filter used for smoothing, and figure 3.21 its use for removing small scale objects from an image. In this case, the "small" objects were blood vessels in an image of a retina, they were removed by using a neighbourhood of 15 by 15 pixels: significantly larger than the vessels' diameters.

The rank filters are computationally expensive, however, rank filtering using a square neighbourhood may be approximated by two applications of one dimensional rank filters. Thus, a filter using an n by n neighbourhood may be approximated by using one with a 1 by n neighbourhood followed by an n by 1 neighbourhood. The number of pixels to be sorted to compute one filtered output is thereby reduced from n^2 to $2n$. The benefit that this gives far outweighs the fact that the result is not quite the same as obtained by the two dimensional filter.

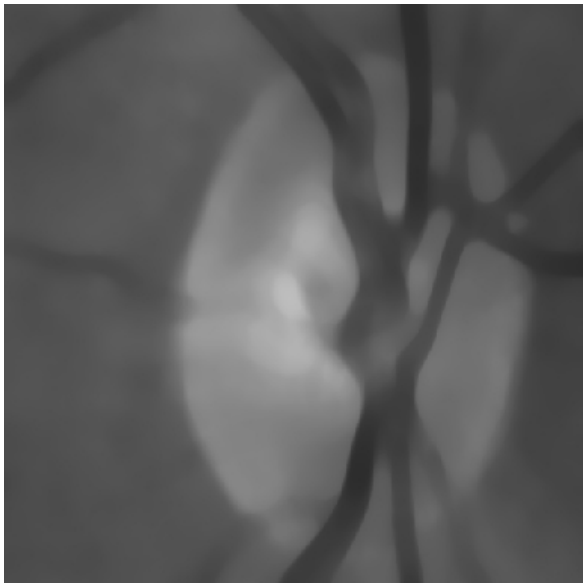


Figure 3.21: Median filter used to remove “small” objects

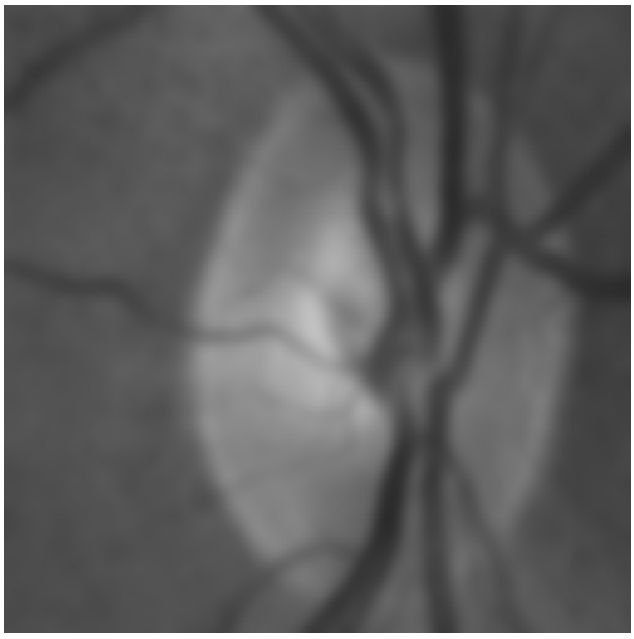


Figure 3.22: An image smoothed by Gaussian smoothing

Many smoothing filters introduce a ringing artefact that is especially visible in the neighbourhood of sharp boundaries: it is manifested as a sequence of echoes of the boundary. The cause of this artefact is due to the shape of the template, which has a sharp spatial cutoff. If the template’s values decayed smoothly to zero at its boundaries, then the echoing effect could be reduced, possibly even removed.

Such a template can be easily realised, a multitude of templates have been defined whose values are determined analytically. A common method is to use a Gaussian (Normal) distribution to compute the template’s elements:

$$T(r) = k \exp - \left(\frac{r^2}{2\sigma^2} \right) \qquad 3.9$$

where r is the distance of the template element from the centre of the template, k is a normalising constant and σ defines the width and size of the template. An example of the use of this filter is shown in figure 3.22.

3.3.3 Sharpening and Edge Enhancement

Sharpening is often presented as the converse of smoothing: smoothing is intended to reduce the effect of significant local variations in the image data whilst sharpening will exaggerate them. Sharpening is often referred to as edge enhancement.

Edges are extremely important structures in images and in image processing for two reasons. Firstly it has been shown that the human visual system contains very specific edge detecting subsystems. It has been shown that the retina performs a very low level edge detecting operation, but there are higher level visual processes that combine this information in many ways to detect different types of edge structure: primarily differences in orientation and velocity in moving edges. So, if edge detection is such an important process for the human visual system, it ought to be important in computer vision systems.

Secondly, and consequently to the first reason, edges define the significant structures in a scene: the outlines of objects and parts of objects. An example of this is our ability to comprehend cartoons – a line drawing that includes only the significant structures of objects. This implies that scenes could be comprehended if only the edge information was present, which will allow us to process significantly less information than would otherwise be the case.

Many types of edge have been defined. An early catalogue defined three idealised types: step edges, line edges and roof edges, figure 3.23. In practice the step edge is the one most commonly encountered, however, the step is seldom as steep as the ideal case and the image values are not completely smooth on either side of the step. The grey level profile through a real edge is more likely to be similar to that shown in figure 3.24.

For all practical purposes, we may define an edge:

An edge is a *significant*, *local, extended* change in image intensity.

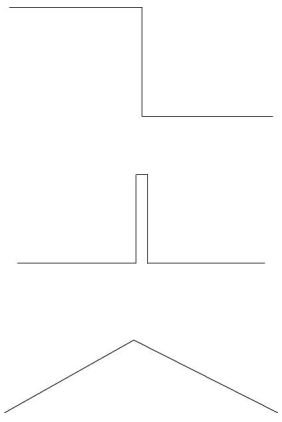


Figure 3.23: Schematic step, line and roof edges.

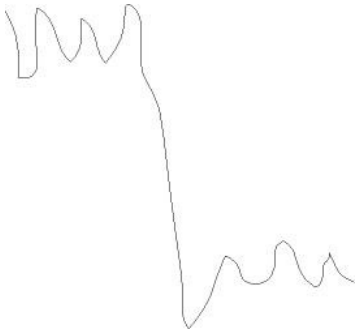


Figure 3.24: A schematic profile through a real step edge.

What is a significant difference? This question must be answered by investigating the difference between two pixels in relation to their brightness. If two monochrome pixels differ by 5 brightness values, this would be significant if their average value was 5, but probably not if their average value was 150. There will be a monotonic change from significant to insignificant as the proportional change in image intensity across the edge decreases.

What is a local change? A difference between two adjacent pixels will be of greater significance than the same difference between two pixels separated by the image's width. Again, there will be a monotonic increase in significance as the pixels being compared approach each other's locations.

Extended implies that the edge is not simply a local variation that could have arisen by random fluctuations of the data.

Given this definition, it is not surprising that most edge enhancement operations involve differencing image values. The earliest was defined in 1965 by Roberts. This operator required the image to be convolved with two templates, figure 3.25. This resulted in two images in which edges with orthogonal orientations were enhanced. These two images were then combined in two ways, the first combined edge magnitudes to estimate the edge strength, and the second used the ratio of the edge magnitudes to estimate edge orientation, eq 3.10:

$$E_{mag} = \sqrt{E_1^2 + E_2^2}$$
$$E_{orient} = \tan^{-1}\frac{E_1}{E_2} \qquad 3.10$$

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

Figure 3.25: Roberts edge templates.

The process is summarised in figure 3.26.

The Roberts operator is possibly the simplest edge-enhancing operator that can be designed. It is however very susceptible to image noise, a simple analysis suggests that the same amount of noise present in a pixel value is also present in the estimate of edge magnitude.

To counter this problem, an element of smoothing can be introduced. We would therefore estimate the edge strength at a location using the difference between the averages of groups of pixels on either side of the edge. Two very simple operators have been defined: the Prewitt and Sobel, they differ only in the

weightings used in computing the averages. Figures 3.27 and 3.28 show the templates defined by these operators, the horizontal and vertical components of the edges they compute and the combined edges.

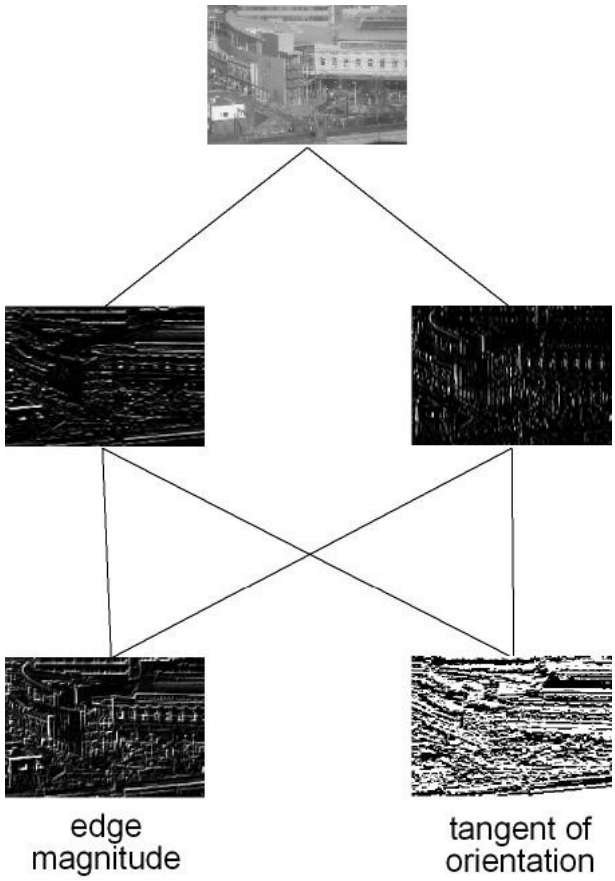


Figure 3.26: Edge enhancement process. An image is convolved with two templates that enhance edges with orthogonal orientations. The enhanced images are combined to yield edge magnitude and edge orientation images.

The Prewitt and Sobel edge enhancement operators are slightly less sensitive to noise than is the Roberts operator. The cost of this benefit is that these operators are slightly less sensitive to small (on the pixel scale) fluctuations in the data; this is not normally a problem.

Is it possible to select one of these operators as superior to the others? One study compared the operators according to their responses to the outline of a circular target and their performance when processing noisy images. Figure 3.29 was typical of the results that were obtained. The rows show the response of each operator when applied to an image containing a circular target on a contrasting background. Each column was derived from a source image with progressively more noise than the image used for the previous one. The interpretation of the result is that the Sobel operator has a superior edge detection performance to edges at all orientations, and a superior behaviour in the presence of noise.

The Sobel operator, whilst it partially addresses the problem of image noise, does not completely solve it. Neither does it address a further problem in image processing: scale.

When one of the simple three by three element templates is used to enhance edge features, they respond to features with a typical dimension of one pixel. However, when humans view a scene, they see objects at a range of scales: for example a tree is composed of a trunk with branches, the branches have smaller branches and ultimately twigs, all of which carry leaves. The tree can be observed at the leaf, twig, branch or whole tree scale. This is an alternative view of “local”, previously, local was defined as a small pixel distance, the present definition suggests that “local” must be interpreted according to the object being viewed.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

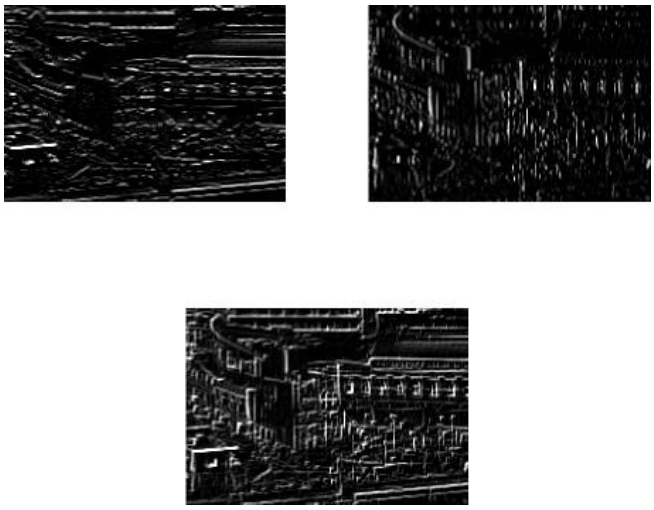


Figure 3.27 Prewitt edge enhancing template. Responses to temple and the combined response.

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

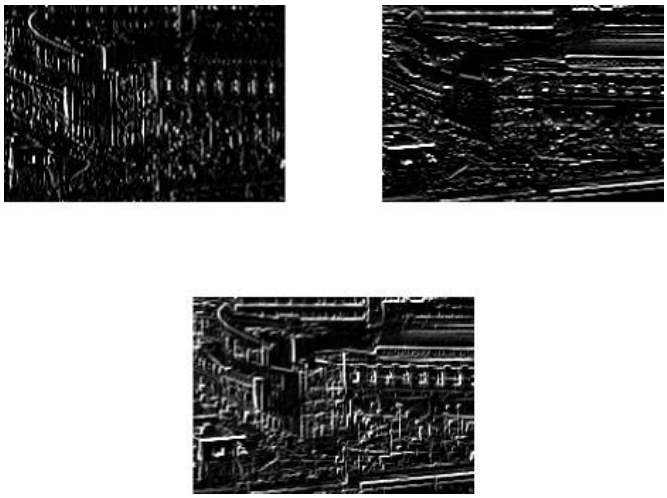


Figure 3.28: Sobel edge enhancing template. Responses to temple and the combined response.

Edge enhancement operators have been suggested that address these issues. Computationally, they involve smoothing the image, using a Gaussian template which includes σ, the scale parameter, followed by differentiation or double differentiation. Canny suggested the former, Marr and Hildreth the latter. Interestingly, the scale behaviour of the operator was not the primary goal of these investigators. Canny was seeking an edge enhancement operator that had three properties: it responded to edges, it yielded accurate edge localisation and it gave a single response to each edge. Marr and Hildreth were investigating properties of the human visual system. Figures 3.30 and 3.31 illustrate the response of these two operators to an image using a range of scale parameters. One useful property is that the operators respond to objects of a different size as the scale parameter is varied. To illustrate this, figure 3.32 shows a cross section through what is often termed a "scale space". The scale space is generated by convolving the image with the templates generated using all values of σ. Thus we generate a volume of data, two dimensions correspond to the image's dimensions and the third to the scale parameter. The cross section illustrates how all objects are detected at small values of σ, but smaller ones are removed by the processing as σ increases. Ultimately, only the largest objects survive. The scale space will be revisited in a later chapter.

Processing in the scale space provides a more efficient method of implementing many image recognition and feature detection operations, as will be seen in subsequent chapters.

The Marr-Hildreth operator included a double differential operator. The operation of double differentiation can result in more accurate localisation of an edge pixel. Consider the profile across a real step edge, figure 3.33a. If the profile is differentiated (figure 3.33b), regions of constant grey value yield a zero, or approximately zero response. The region where the grey value changes, which will correspond to the edge region – a region that will contain the point we wish to label as the edge, will yield a finite result. Often there will be a local maximum within this region, this defines the location of the edge. If there is no local maximum, the region of greatest magnitude is defined to contain the edge pixel which will be determined subsequently by other methods. However, if the profile is double differentiated, we obtain the result of figure 3.33c. Again, uniform regions yield a zero result and the edge region a finite result. In this case, the result is positive where the profile's gradient is increasing and negative where it is decreasing, the position of maximum slope is located where the doubly differentiated profile crosses the zero. This position may or may not coincide with the centre of a pixel, in fact we may use curve fitting techniques to locate the edge position to sub-pixel accuracy, should this be desirable. Either way, the zero crossing defines the location of the edge unambiguously.

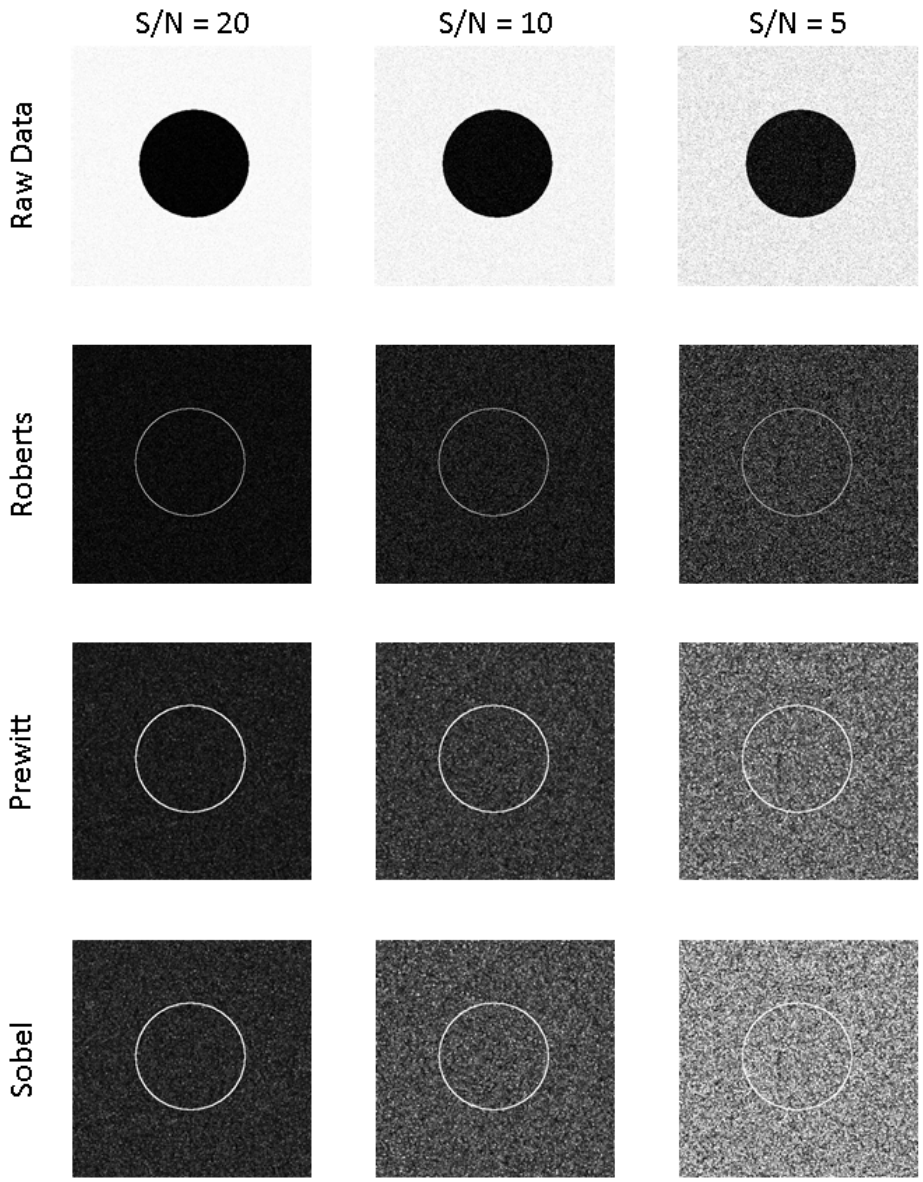


Figure 3.29: Responses of Roberts, Prewitt and Sobel edge enhancement operators in detecting the outline of a circular object in the presence of noise at increasing magnitudes.

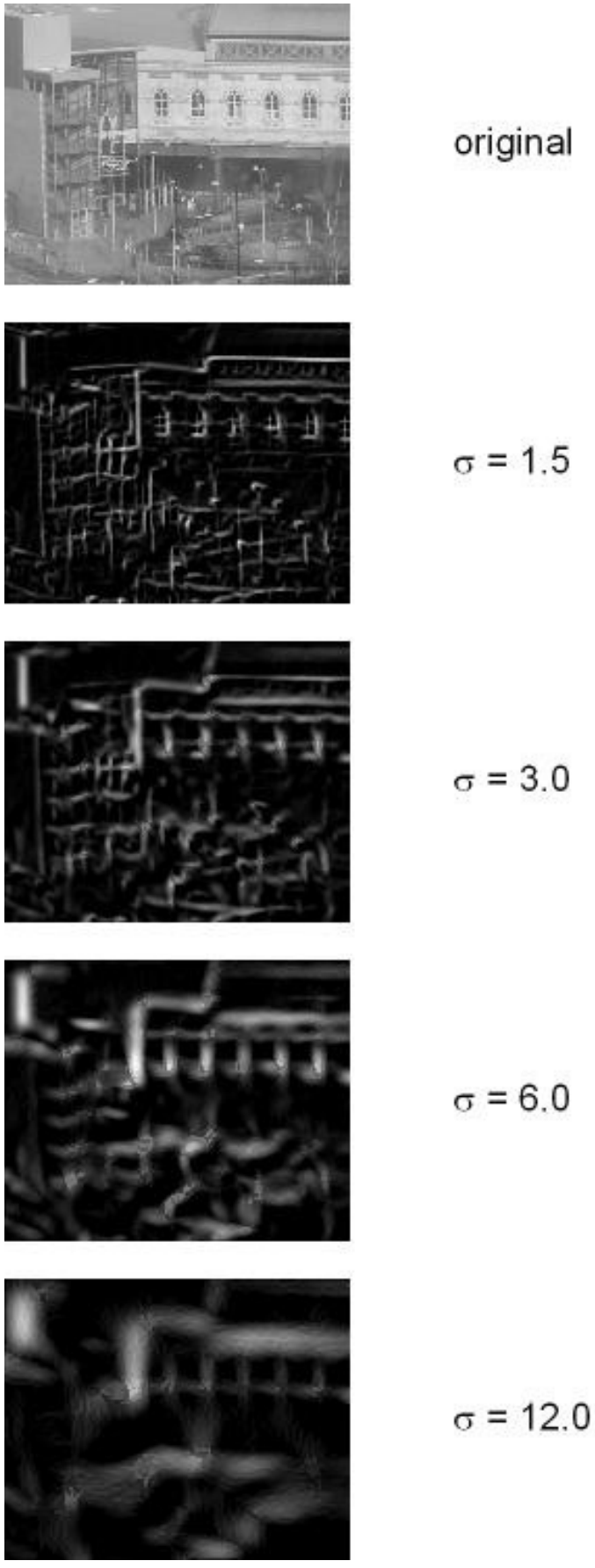


Figure 3.30: Canny response to an image with operators of increasing widths

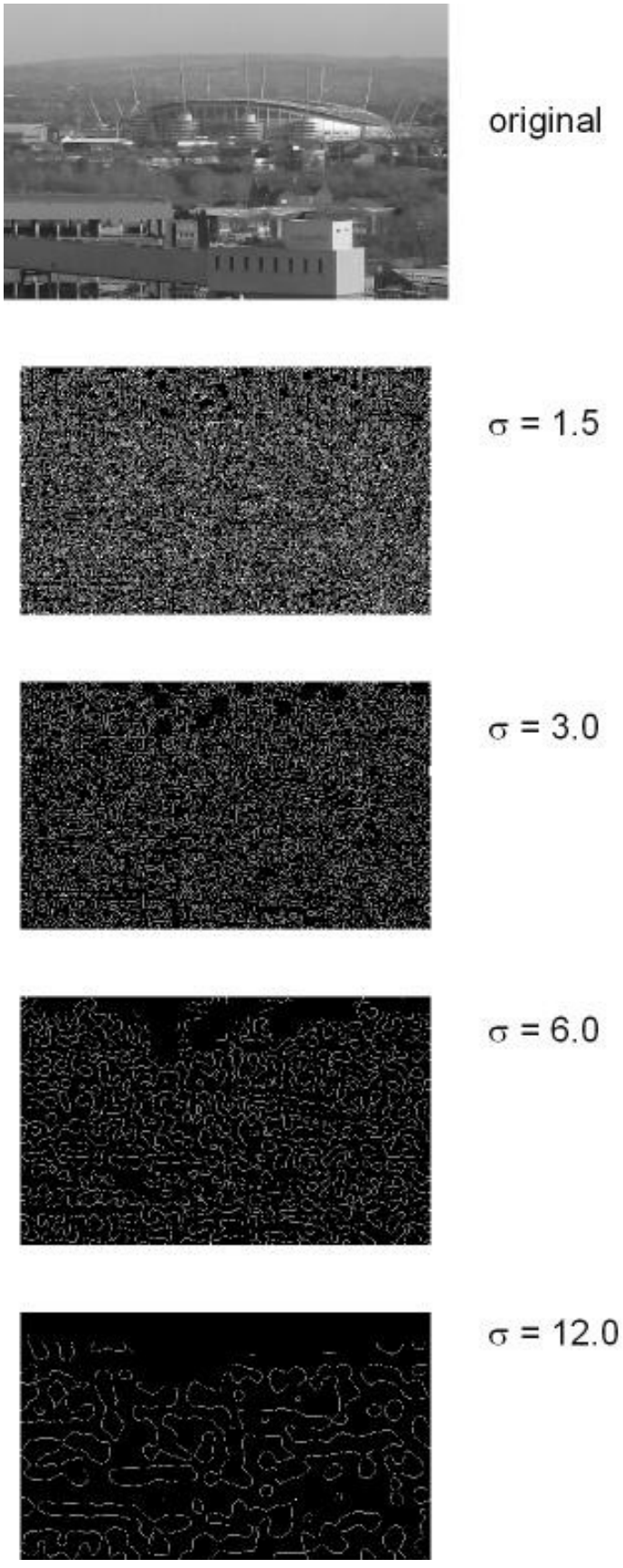
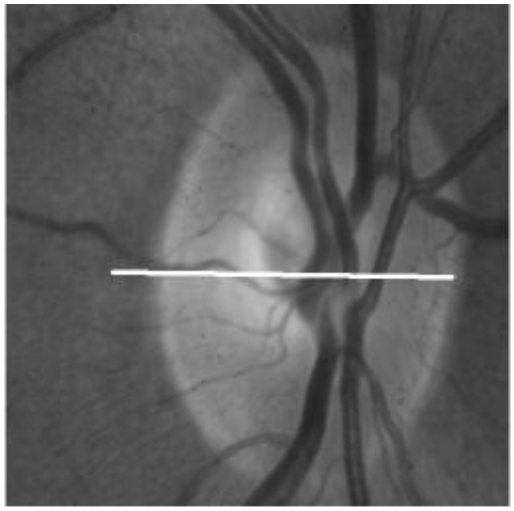


Figure 3.31: Marr-Hildreth with increasing widths.



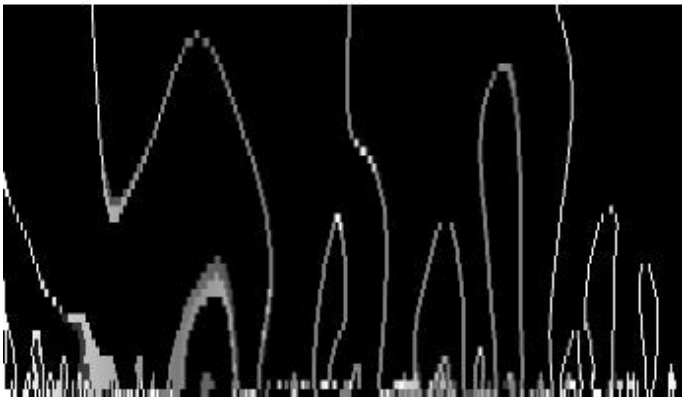


Figure 3.32: Cross section through a scale space. The sample location is indicated. The scale space was computed by convolving the image with the Marr-Hildreth template with increasing σ, and detecting the zero crossings.

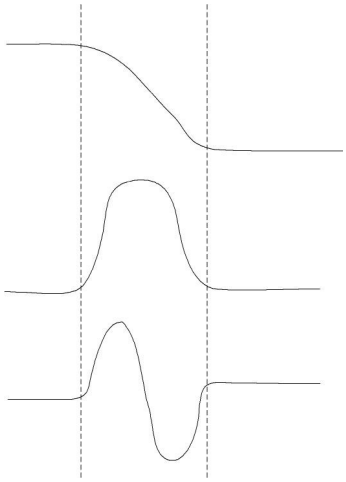


Figure 3.33: Edge detection by double differentiation. top) a noise free but blurred edge, middle) its differentiation and bottom) its double differentiation.

3.4 Global Transforms

3.4.1 Hough transform

The Hough transform was first proposed as a means of detecting linear features in an image. It depends on being able to represent the object being sought in an analytical manner using a small number of parameters. Each point in the image that might contribute to the object is then transformed into the set of parameters corresponding to all of the objects it might contribute to. Having transformed all of the image points in this way, the sets of parameters are examined; those that have contributions from many points are assumed to be the parameters of objects that really are present in the image.

The transform is best explained with an example.

Consider a straight line. It may be represented by equation 3.18. This has two parameters, *m*, the line's gradient and *c*, the intercept (the point where the line and the vertical axis intersect).

$$y = mx + c \qquad 3.18$$

We may rearrange this equation:

$$c = mx - y \qquad 3.19$$

If we define a space using *m* and *c* as co-ordinates, instead of *x* and *y*, then this equation also represents a straight line, with gradient *x* and intercept *–y*. We can therefore make a transformation between the image space and the Hough accumulator: the image has co-ordinates *x* and *y*, the accumulator has *m* and *c*. A point in the image, with a specific *x, y* combination will transform to a line in the accumulator, with that gradient (*x*) and intercept (*-y*), figure 3.40. Different *m, c* combinations along this line will correspond to the lines passing through the *x ,y* point in the image (an infinite number of them).

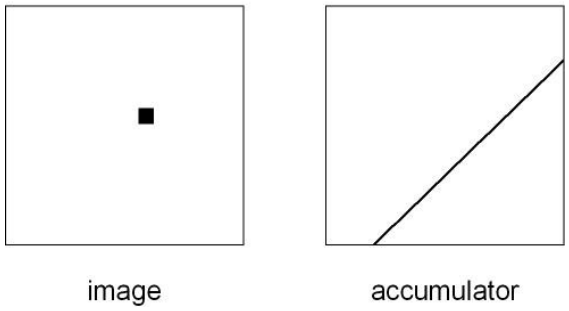


Figure 3.40: Correspondence between an image and the Hough accumulator.

The accumulator is a simple array of counters used to record the number of contributions made to each parameter combination. When it is designed, the implementer must decide the appropriate range of parameter values and the resolution of values, i.e. the difference in value between adjacent cells.

The Hough transform is used to locate lines in an image by firstly enhancing the edges in the image and then thresholding them. We then have a set of points, some of which lie on the lines that exist in the image, others do not. Each point is considered in turn, the line it defines in the accumulator is computed and the values in the cells in the accumulator that it passes through are incremented. Having processed all points in this way, the accumulator is searched for local maxima, these define the parameters of lines in the image, and the number of points that contributed to this line. No information is available that relates to the start and end points of the line. These must be determined from the original image data.

In practice, this form of the line's equation is not used as it does not represent vertical lines: these have infinite gradient and intercept which clearly cannot be represented in the accumulator. Instead the line

representation of figure 3.41 is used. In this, r is the perpendicular distance from the line to the origin and θ is the angle between the perpendicular and the x axis and we have an r, θ accumulator:

$$r = x \cos\theta + y \sin\theta \qquad 3.20$$

The Hough transform can be defined for all analytically defined shapes. For a circle, we would use the form:

$$r^2 = (x - x_o)^2 + (y - y_o)^2 \qquad 3.21$$

Where r, x_o and y_o represent the circle's radius and origin. The parameter space would be three-dimensional and points would transform into a conical surface.

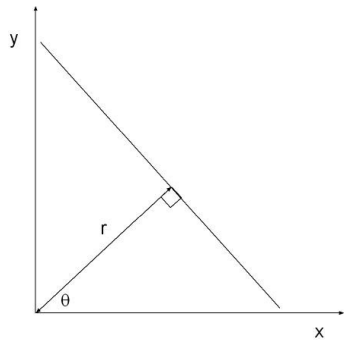


Figure 3.41: Line equation used for the Hough transform.

The transform has also been extended for arbitrarily shaped objects by defining a look-up table that dictates the cells to be incremented. This is not often used.

3.5 Geometrical transformations

Previous sections of this chapter have examined methods that process the values of pixels *in situ*, that is, the location of the pixel is not affected. In this section we shall discuss two sets of transforms that change pixels' locations in a carefully defined manner: linear and non-linear transforms.

Having moved a pixel, we often find that its new location does not correspond to one of the allowed locations; we often find that the new location is a non-integer co-ordinate. We must examine methods of overcoming this problem.

3.5.1 Linear transformations

An arbitrary geometrical transformation, T, will move a pixel at co-ordinates (x, y) to co-ordinates (x', y'). The pairs of co-ordinates are related by the pair of transformation equations:

$$\begin{aligned} x' &= T_x(x, y) \\ y' &= T_y(x, y) \end{aligned} \qquad 3.22$$

For all linear transformations, the two equations will be expanded to the linear polynomial:

$$x' = a_0 x + a_1 y + a_2$$
$$y' = b_0 x + b_1 y + b_2 \qquad 3.23$$

And these equations can in turn be expressed in matrix terms as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad 3.24$$

The co-ordinates are represented using homogeneous co-ordinates. This is simply so that the whole set of transforms can be represented using a single matrix product, rather than a matrix multiplication followed by a matrix addition.

By using the appropriate values for the six parameters, the various linear transformations of translation, rotation, scaling and shearing can be realised. The values are listed in table 3.1.

Table 3.1: Transformation coefficient values for Linear transformations

Transformation	a_0	a_1	a_2	b_0	b_1	b_2
Translation by (x,y)	1	0	x	0	1	y
Rotation by θ	cos θ	-sin θ	0	sin θ	cos θ	0
Uniform scaling by a factor s	s	0	0	0	s	0
Vertical shear by a factor s	1	0	0	0	s	0

These transformations may be applied in sequence and the resulting transformation will also be a linear one. Thus we may translate the image, rotate it and translate it again. The net effect will be a rotation about a point other than the origin. Further, rather than apply each transformation to the image, we may compute a compound transformation matrix by multiplying the individual matrices and simply transform the image using this single transformation.

One important effect of linear transformations is that straight lines remain as straight lines and parallel lines remain parallel. The angle between a pair of intersecting lines might be changed, as might the area enclosed by a polygon

.

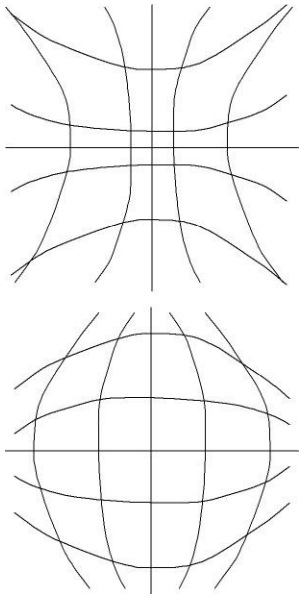


Figure 3.53: Pincushion and barrel distortion.

3.5.2 Non-linear transformations

Non-linear transformations will involve higher order terms in the polynomial expansion of equation 3.22. We might have a pair of second order transformation equations:

$$\begin{aligned} x' &= a_0 x^2 + a_1 xy + a_2 y^2 + a_3 x + a_4 y + a_5 \\ y' &= b_0 x^2 + b_1 xy + b_2 y^2 + b_3 x + b_4 y + b_5 \end{aligned} \qquad 3.23$$

The effect of this type of transformation is to introduce a degree of warping to the data. Depending on the relative magnitudes of the first three coefficients (subscripts 0, 1 and 2) the image will be distorted according to figure 3.53: pincushion (in which the sides of the image are distorted less than the corners) or barrel distorted. Not only does this transformation produce interesting graphic effects, it also models very accurately the types of distortion introduced by many camera lenses. For accurate measurement work, or for mosaicking a set of images, these distortions must be measured and removed: it is a non-linear transformation that achieves this correction. Figure 3.54 illustrates this process, showing a pair of images and their best overlap without correction.

Figure 3.54: Mosaicking – these two images are overlapped without any correction for the different image planes. The misalignment should be obvious at the join, especially in the profile of the mountain.

3.5.3 Image resampling

Thus far, image transformations have been discussed that map an input image to an output image. Is this a sensible manner to implement the transformation?

Consider a rotation, this involves multiplying each co-ordinate value with a sine or cosine. Only four angles have integer values of these functions: 0°, 90°, 180° and 270°. All other angles have non-integer

values. Therefore, unless a rotation through these specific angles is required, each input pixel might be transformed to a non-integer output pixel.

Several methods have been suggested for solving this problem: truncating or rounding the co-ordinate values, or sharing the pixel's value amongst the neighbouring destination locations. All of these methods suffer from the problem that many pixels in the output image will not be addressed: this will give blank pixels in the result image. Perhaps the most elegant solution is to apply the transformation in reverse. In this case, we would take an output location and use the reverse transformation to determine where it originated in the input image. This location might still be non-integer, but we are able to guarantee that all output pixels will be populated. Figure 3.55 illustrates the problem, the output pixels *(x′, y′)* are reverse mapped to the input pixels *(x, y)* which might be non-integer.

How then should the output pixels' values be computed? Two solutions are widely used.

Nearest neighbour interpolation simply rounds *(x, y)* to the nearest neighbour. Its value is given to the output *(x′, y′)*.

Bilinear interpolation takes a weighted sum of the four nearest neighbours to *(x, y)*. The values of the four neighbours are weighted inversely by their distance from *(x, y)*. Whilst this transformation requires more computations than simple nearest neighbour interpolation, it results in visually more pleasing output, specifically, the jagged effects generated by rotating straight edges are reduced.

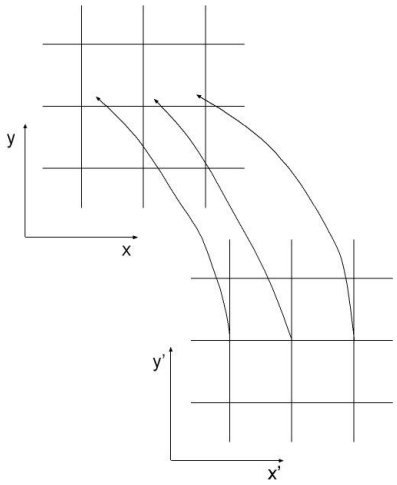


Figure 3.55: Forward and backward mapping for resampling

3.6 Summary

This chapter has discussed the most significant image transformations. These are the basic building blocks of all image processing/computer vision systems. Their place in such systems will become more apparent as we negotiate the following chapters.

3.7 Bibliography

Mylar and Weeks (1993) and Umbaugh (1998) provide useful accounts of low level image processing. Marr and Hildreth (1980) and Canny (1986) were instrumental in originating what have become optimal edge detectors. Kittler et al. (1983) presented an early account of the analysis of the accuracy of simple edge detectors. The analysis has been repeated often since, looking at many different detectors. Hubbel (1988) performed much early research into understanding the primate visual system, which had a strong influence on the design of edge detectors and indeed, on computer vision systems.

3.8 Exercises

1. The perception of the grey values of an image may be improved by changing the grey scale from a linear one to a logarithmic scale, in which the output value is proportional to the logarithm of the input value. Suggest circumstances in which this mapping would be useful and compare the processing times required to perform the transform by (a) using the logarithm function and (b) using a look-up table.

2. Implement the histogram equalisation algorithm and verify that it improves the appearance of images under certain conditions.

3. Implement the thresholding algorithms and use them to segment a simple image. By how much do the results differ? Are these differences significant?

4. Implement software that convolves an image with a three by three element template. Explore the effects that can be achieved by using various values in the template.

5. If a 2-D template can be separated into two 1-D templates, estimate the numbers of operations that must be performed to convolve an N x N pixel image with the 2-D template and with the two 1-D templates.

6. The Laplacian of Gaussian (the Marr-Hildreth) operator can be approximated by subtracting two Gaussians with different values of σ. For a given LoG, estimate the two values of σ that give the best match and the difference between the LoG and this approximation.

Chapter 4
Morphology

Morphology is defined as the "study of the form of things", that is the structure of objects. Morphological transforms are those that are designed to elucidate this structure. Morphological techniques are also frequently used to improve the appearance of a thresholded image: we shall see that thresholding seldom, if ever, gives perfect results and how these operators may redeem the situation.

Whilst morphological algorithms are usually applied to binary (bilevel) images, they can be adapted to greyscale images. This chapter will describe thresholding, binary morphology and grey level morphology. It concludes with an examination of some specific transforms for extracting structural information from a region: the distance transform, skeletonisation and the convex hull.

4.1 Thresholding

4.1.1 Thresholding monochrome images

Thresholding is the process of reducing the grey scale of a monochrome image to two values. It is defined according to equation 3.1 which is repeated here:

$$g(x,y) = \begin{cases} 0 & \text{if} f(x,y) \le \theta \\ 1 & \text{otherwise} \end{cases} \qquad 4.1$$

Where *f(x,y)* and *g(x,y)* are the input and output images respectively and θ is a threshold value. The two output values need not be zero and one; any two different values are sufficient, 0 and 255 are often used as this maximises the contrast between the values.

Thresholding is an operation that can also be applied to colour images, which is discussed below.

The execution of the thresholding function is simple; the difficulty lies in selecting the correct value for the threshold. This is done by using the grey level histogram, which records the frequency of occurrence of each grey value in the image. In the ideal case, the threshold would clearly distinguish between contrasting object (foreground) and background. For example, the typed dark letter on the white background, or the dark coloured toy on the light background in figure 4.1. Figure 4.2 shows the histograms derived from these images. Clearly, we may select some value in the trough between the two peaks as our threshold. If we then applied equation 4.1, we could distinguish between object and background (but would not be able to identify which was which).

Identifying the threshold in this way is an operation that may be performed manually, or it is a process that is easily automated. This method of determining the threshold is known as the modal method (even though this is a misuse of the term since the mode is defined as the value that occurs most frequently).

Figure 4.1: Two very different objects

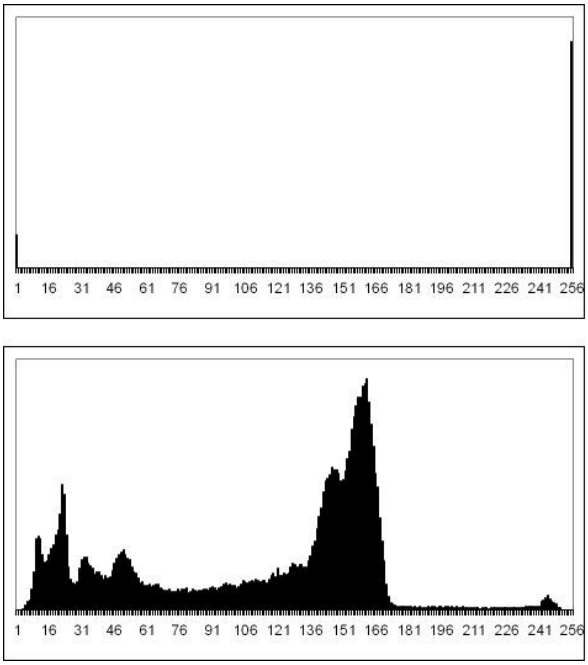


Figure 4.2: The objects' histograms

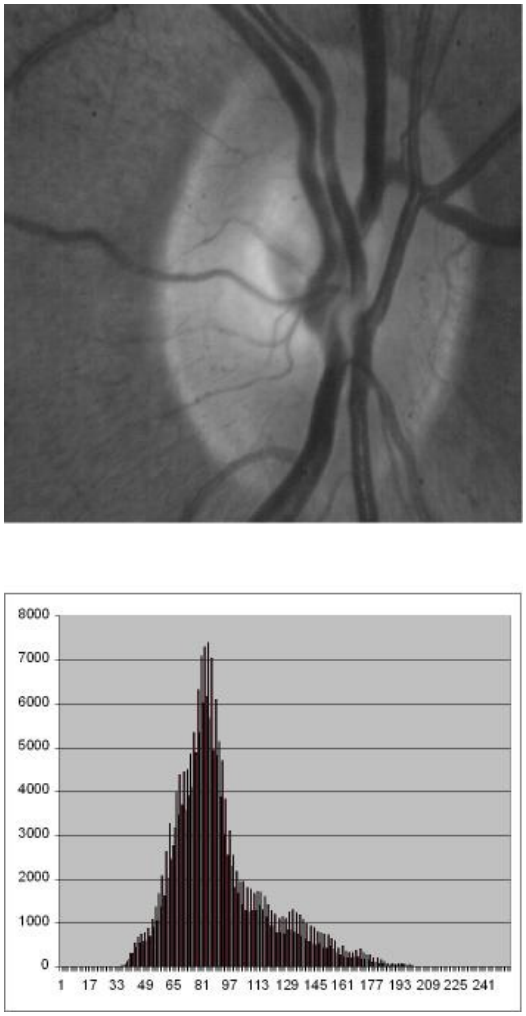


Figure 4.3: top) Optic nerve head image and bottom) its histogram

But what would happen if there were no clear division between the foreground and background. Figure 4.3a shows an image of part of the retina called the optic nerve head, it is the region where blood vessels and nerve fibres pass into and out of the eye. It is possible to distinguish between the brighter nerve head region and the darker regions that are the retina. However, figure 4.3b, the histogram of this image, reveals no such clear division. The same effect is observed if the values of the foreground and background are very similar: imagine the peaks in figure 4.2b sliding towards each other until they overlap. In these cases we cannot threshold at the mode, indeed, the modal threshold may not exist.

A simple method of determining the threshold relies on knowing what proportion of the image is occupied by the foreground and whether the foreground is brighter or darker than the background. This method simply defines the threshold as that grey level which selects the correct proportion of the grey levels. If the foreground occupies P% of the image and is darker than the background, then the threshold we require is the one shown in figure 4.4, it divides the area under the histogram into one region containing (100-P)% of the pixels corresponding to the background and a region containing P% of the pixels: the foreground. Naturally, this method is only effective if the size of the object is known in advance.

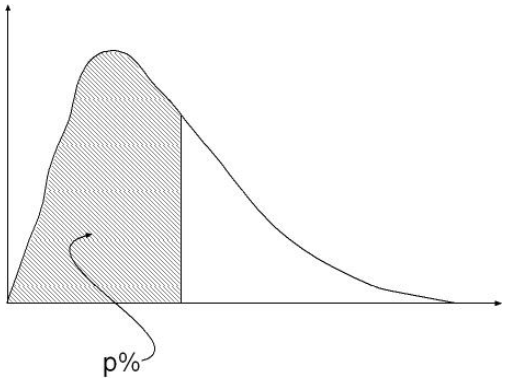


Figure 4.4: Histogram and P-tile threshold

In other cases, two iterative methods have been suggested, both are equally effective and differ only in detail.

The first method searches incrementally through the histogram for a threshold. Starting at the lower end of the histogram, we initialise a suggested threshold as the lowest value in the image; we compute the average of the pixels with grey values less than this threshold, call this *L*, and the average of the pixels with grey values greater than the suggested threshold, call this *G*. We then compute the average of *L* and *G*. This value will be the threshold if it is equal to the suggested threshold, otherwise we increment the suggested threshold and repeat the process.

The second method searches the histogram more purposefully. An initial threshold value is suggested, the average of the image's four corner pixels is a suitable choice. The average values of the pixels whose values are less than and greater than the initial threshold are computed, *L* and *G* respectively. If the average of *L* and *G* equals the threshold, then we have found the threshold value, as before, otherwise the search continues with an updated value of the threshold. The difference between the two methods lies in the updating of the suggested threshold: in this case the updated value is equal to the average of *L* and *G*. Figure 4.5 illustrates the results obtained by applying both methods to the image of figure 4.3, there is no significant difference between the two methods.

A problem will also arise when the local average in an image varies in a systematic manner across the image. This can happen if for example the scene is illuminated strongly from one side: the side nearer the light source will be better illuminated and hence this side of the image will be brighter. Any threshold value that is computed using statistics derived from the whole image will therefore be correct in a few image locations, but incorrect over most of the image.

The solution to this problem is to estimate the value of the threshold at each pixel. This is usually achieved by using a region surrounding the pixel and computing a threshold using one of the methods discussed. The size of the region to be inspected should be decided by experimentation. It should also be realised that inspecting all of the image's pixels is an expensive operation, a subset could be chosen and the intervening thresholds could be interpolated. Again, the number of thresholds to be computed should be determined by experimentation.

Figure 4.5: Thresholded image of figure 4.3 using the two iterative threshold finding methods.

There are occasions when the foreground occupies a range of grey values that is between two ranges of background. That is, there are background pixels that are darker and brighter than the foreground. The solution to this difficulty is to define two threshold values, θ_1 and θ_2, the thresholding rule of equation 4.1 now becomes:

$$g(x,y) = \begin{cases} 1 & \text{if} \theta_1 \le f(x,y) \le \theta_2 \\ 0 & \text{otherwise} \end{cases} \qquad 4.2$$

The output values have been exchanged to ensure that the foreground has a greater value in the output image. The process is known as band thresholding, the two thresholds can be supplied manually or using an adaptation of the mode method. The two iterative methods are inappropriate.

4.1.2 Thresholding colour images

As we know, a colour image is represented using three colour channels. This suggests a simple method of thresholding the data: supply a threshold for each colour channel independently of the others and combine the results by conjunction: the effect of this is to partition the RGB colour cube into octants; the thresholded region will be the cuboid furthest from the origin: figure 4.6. The drawback of this

method is that it is not particularly flexible as the required object will not often lie in an octant of the colour cube. It is of course possible to band threshold colour images, in which case, six thresholds would be defined and the colour space would be partitioned into 27 regions.

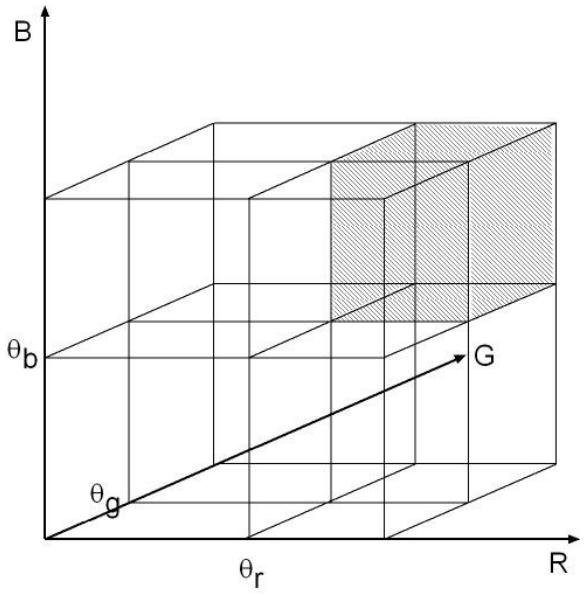


Figure 4.6: Thresholding colour data by partitioning the colour cube

Of more practical use is a method that thresholds the colour space according to each colour's distance from a user specified one; that is, the user specifies a reference colour (R_0, G_0, B_0) and the thresholding operation selects those pixels having similar colours:

$$g(x,y) = \begin{cases} 1 & D(x,y) \le D\max \\ 0 & \text{otherwise} \end{cases} \qquad 4.3$$

where $D(x,y)$ is a distance function defined in the colour space ($f(i,j)$ is the pixel value):

$$D(x,y) = \sqrt{\left(f_R(x,y) - R_0\right)^2 + \left(f_G(x,y) - G_0\right)^2 + \left(f_B(x,y) - B_0\right)^2} \qquad 4.4$$

Whilst performing this thresholding operation using RGB data will generally give satisfactory results, it may be more appropriate to use the perceptual colour space defined by the hue, saturation and value (HSV) components. Since HSV define a perceptually uniform colour space, the value of D_{max} will give the same perceptual effect, irrespective of the location of the colour in the colour space. This is not the case with the RGB colour space, the same value of D_{max} may be perceived differently according to location within the colour space.

4.1.3 Outcome of thresholding operations

In the earlier example of an image, especially the character in figure 4.1, there was a clear distinction between foreground and background, therefore there was no confusion between these regions in the thresholded images. In practice, such a state is rare. It is much more common to encounter the situation exemplified by figure 4.3 in which the greyscale ranges occupied by foreground and background overlap and there are misclassified pixels in the output: some foreground pixels have been classified as background and vice versa. The number of erroneously classified pixels may be estimated by inspecting the histogram. Two Gaussian distributions are fitted to the curve, representing the distributions of foreground and background pixels. The threshold value can be defined at the lowest value of the histogram between the two peaks, as shown in figure 4.7. Then the two shaded areas

represent the proportion of misclassified pixels. These pixels are revealed in the image as inconsistencies in the boundary between foreground and background, and isolated pixels of one class lying within the bulk of the other (isolated white or black pixels). The binary image can be processed to recognise and correct some of these errors.

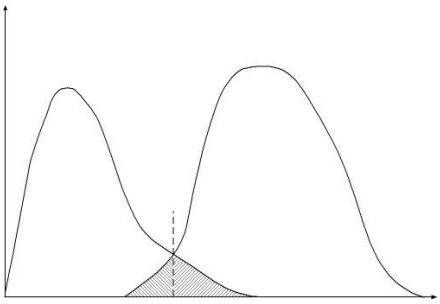


Figure 4.7: The histogram of an image containing two objects having different but overlapping grey scale distributions. The shaded region is the misclassified pixels: object 1 pixels classified as object 2 and vice versa.

4.2 Processing binary images

Binary morphology relies on three key concepts:

- A structuring element
- Fitting
- Hitting

The structuring element is a small template that is used to investigate the image. The structuring element's values, like the image it investigates, will be binary. Combinations of values making circular, square or cross-shaped templates are usually employed. The template will have an origin: it is this location that specifies the template's location in the image and therefore the position where the operation's result will be written. It is usual for the structuring element to have odd dimensions and the origin to be at the centre.

A structuring element is said to fit at an image location if *all* the image pixels that overlap the structuring element pixels of value 1 are also 1. The image values are irrelevant if the structuring element value is 0. Therefore, in figure 4.8, the structuring element fits at location A (the area with the dashed boundary) but not at location B (because of the zero at the top-left).

The structuring element is said to hit at an image location if *any* of the image pixels that overlap the 1 values in the structuring element are 1. In figure 4.9, the structuring element hits at location A but not at location B.

Given these definitions, the two major morphological operations can be defined, and their two combinations.

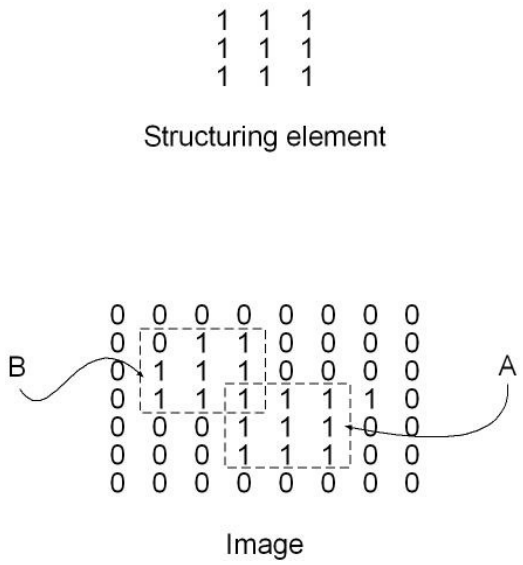


Figure 4.8: Structuring element fitting and not fitting a binary image.

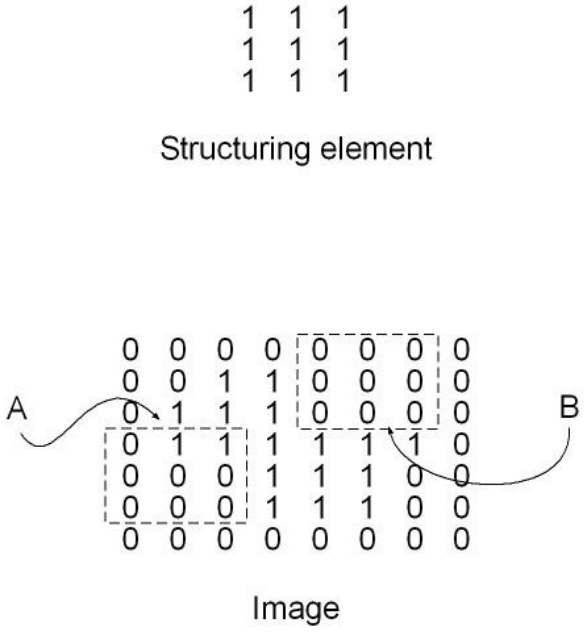


Figure 4.9: Structuring element hitting and not hitting a binary image.

4.2.1 Erosion

Three definitions of erosion will be presented: using the fitting operation, a set theoretic definition and a very informal one.

Erosion of an image *f(x,y)* by a structuring element *s(x,y)*, is denoted by $f \bullet s$ and is defined as:

$$f \bullet s = \begin{cases} 1 & if\ s\ fits\ f \\ 0 & otherwise \end{cases} \qquad 4.5$$

The origin of *s* is placed in all possible locations in the image in this computation. Just as occurred with convolutions, there will be locations where the structuring element overlaps the image boundary. In such cases we proceed as we did with convolution previously: either we process just those pixels that lie within the image, ignoring the portions of the structuring element that are outside of the image, or disregard these portions of the image entirely.

Secondly, the operation of erosion is defined formally using the set theoretic notation of equation 4.7, which uses the operation defined in equation 4.6.

$$A_x = \{c : c = a + x, \forall a \in A\} \qquad 4.6$$

$$A \bullet B = \{x : B_x \subseteq A\} \qquad 4.7$$

The first of these equations defines a displacement of the set of pixels A by an amount $\boldsymbol{x}$ ($\boldsymbol{x}$ will define a distance and direction). The second equation defines the erosion of A using the structuring element B.

Finally, we may also describe this operation informally as the following two steps:

- At each possible placement of the structuring element in the image
- Remove the pixel overlying the origin if the structuring element overlies a non-object pixel

The effect of applying this operator to a binary image is shown in figure 4.10. An image of an optic nerve head has been thresholded to give initial binary image. Four iterations of erosion have been applied. It is apparent that isolated object pixels are soon removed and the object itself shrinks at each iteration by an amount approximately equal to the size of the structuring element. The shrinkage is not only from the outside of the object but holes within the object are enlarged.

There are two obvious applications of erosion. The first is the removal of unwanted small-scale structures in an image. However, if we use erosion for this purpose, objects in the image that should be retained are also adversely affected. So, although erosion could be used for this purpose, in practice, other operators are used. Secondly, erosion may be used to identify the boundaries of objects: by subtracting a suitably eroded version of the image from the original, we identify those pixels that were removed by the erosion: isolated noise pixels plus object boundaries, figure 4.11.

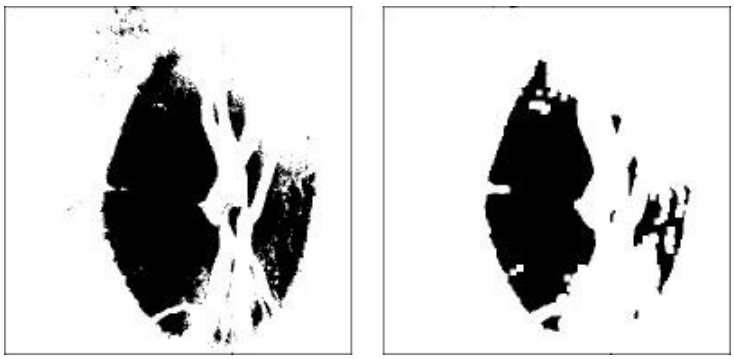


Figure 4.10: Erosion of a binary image using a square SE.

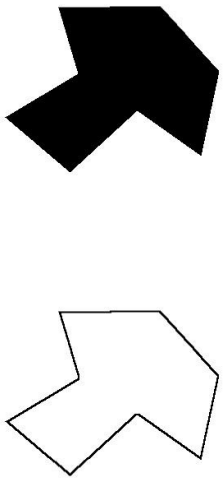


Figure 4.11: Boundary detection using erosion.

4.2.2 Dilation

The converse to erosion is dilation. Whereas erosion removes pixels from objects in an image, dilation adds pixels to the objects. Following the previous description, three definitions of dilation will be presented.

The dilation of an image, *f(x,y)*, by a structuring element *s(x,y)*, is denoted by $f \oplus s$. It is defined as:

$$f \oplus s = \begin{cases} 1 & if\ s\ hits\ f \\ 0 & otherwise \end{cases} \qquad 4.8$$

Again, this is realised by placing the structuring element's origin at all locations in the image and performing this test. Overlaps of *s* with the image boundary are dealt with in the same manner.

The formal, set theoretic definition of dilation requires the translation of equation 4.6 and a reflection operator as defined in equation 4.9. Using these, dilation is defined in equation 4.10.

$$\hat{B} = \{x : x = -b, \forall b \in B\} \qquad 4.9$$

$$A \oplus B = \{x : \hat{B}_x \cap A \neq \varnothing\} \qquad 4.10$$

And finally, an informal definition of dilation:

- At each possible placement of the structuring element in the image
- Add the pixel overlying the origin if the structuring element overlies an object pixel

What effect does this operator have on an image? Figure 4.12 shows the same source image as was used in figure 4.10, but it is now dilated four times with the same structuring element. As might be expected, at each iteration the object expands: a layer of pixels whose thickness is approximately the size of the structuring element has been added to the boundaries of the objects. Isolated noise pixels have also been expanded. Small structures on the objects are more pronounced and small holes in the object boundary have been filled in.

Figure 4.12: Dilation of a binary image using the square SE

4.2.3 Opening

Erosion and dilation both have positive effects: one removes isolated and small structures, the other fills in gaps. They also have a negative effect: the sizes of all objects are changed. Erosion and dilation operators may be combined to enhance the positive effects and suppress the negative. Opening of an image is defined as erosion followed by dilation using the same structuring element.

The erosion operator will remove noise pixels, be they isolated ones or spurs connected to an object. It will also shrink the object by removing some layers of boundary pixels and also expand any holes in the boundary, which could also be due to noise. The dilation operator will replace the boundary pixels and, to a certain extent, fill in the boundary holes. Figure 4.13 illustrates the effect of the operation, showing the original image, the intermediate result of erosion and the final result. It is apparent that a much smoother version of the object has been generated: isolated pixels have been removed and the object boundary is significantly straighter than it was originally.

Whereas further applications of erosion or dilation operators changed the image, further opening of the image with the same structuring element will produce no more changes.

4.2.4 Closing

Closing uses the other combination of the two basic operators: it is defined as dilation followed by erosion. The dilation operator will fill holes in the object and possibly result in closely adjacent objects being merged. The objects themselves will also be increased in size. The erosion operation will restore the object to its original size, but holes that were filled will remain filled, as shown in figure 4.14.

Like opening, repeated application of the closing operator, using the same structuring element, will have no further effect on the image.

Figure 4.13: Opening of a binary image:
top) the original image,
middle) the intermediate eroded image,
bottom) the final dilated image

4.3 Processing grey scale images

Although morphological operators have so far been applied exclusively to binary images, they may be modified for application to greyscale images.

In greyscale morphology, the structuring element will no longer be binary; it may take integer values, including zero and negative values. The formulation of the erosion and dilation operators must therefore be modified, and they must be understood in slightly different ways. The effects of eroding or dilating a greyscale image will be dramatically different to the equivalent binary operators.

4.3.1 Erosion and Dilation

Formally, the greyscale erosion of an image, *f(x,y)*, with a structuring element, *s(x,y)*, at location *m,n*, is defined as:

$$f \bullet s(m,n) = \min_{j,k \in S} \{f(m-j, n-k) - s(j,k)\} \qquad 4.11$$

The output value is simply the minimum difference between the image and structuring element.

Figure 4.14: Closing a binary image,
top) original,
middle) dilated image,
bottom) result

Informally, this operation may be understood by imagining the image as a landscape, with the intensity equating to height. We place the structuring element beneath the landscape and raise it as far as possible without it piercing the surface. The distance that the element is raised is the required output. Note that this distance could be negative if any of the structuring element's values is greater than the corresponding image value. The value could also exceed the maximum range of image values if the structuring element had negative values. The output of the erosion must therefore either be truncated to the correct range, or rescaled once the global minimum and maximum have been determined.

Figure 4.15 illustrates the effect of using a uniform structuring element to erode a simple greyscale image.

Once again, we may formally define the dilation of an image, $f(x,y)$, with a structuring element, $s(x,y)$, at location m,n:

$$f \oplus s(m,n) = \max_{j,k \in S} \{f(m-j, n-k) + s(j,k)\} \qquad 4.12$$

The informal explanation of this is that we invert the structuring operator and lower it from above the landscape until one element makes contact with the surface. The height of this element is the value to be output. The same comments may be made regarding the range of the output values and an example result is presented in figure 4.16.

Figure 4.15: Greyscale erosion

Figure 4.16: Greyscale dilation

4.3.2 Opening and Closing

Opening and closing of a greyscale image are defined in exactly the same ways as for a binary image, except, of course, that the erosion and dilation operators are now defined differently. The effect of the operators is also different: rather than smoothing the outlines of objects, we are now smoothing the topography of the landscape that is represented by the image. Opening and closing can be viewed as smoothing the landscape from underneath and above respectively, figures 4.17 and 4.18.

Consider the structuring element to be a sphere. Then the opening operator can be thought of as rolling the sphere over the underside of the landscape. The output of the operation will be the highest points reached by any part of the sphere. Peaks narrower than the sphere will be too small for it to enter and will therefore be reduced in amplitude and sharpness. The initial erosion removed the small details (small with respect to the structuring element) but also darkened the image. The following dilation brightened the image but could not reintroduce the details removed by erosion.

Conversely, closing can be thought of as rolling the sphere over the top of the surface and recording the lowest heights it reaches. Thus the brightest points in the image remain, but darker, smaller peaks will be removed.

Figure 4.17: Greyscale opening

Figure 4.18: Greyscale closing

4.3.3 Applications

Three particularly useful combinations of greyscale morphological operators have been defined and are discussed in this section. The section concludes with a description of two applications of these operators.

Figure 4.19: Morphological smoothing

4.3.3.1 Morphological smoothing

Recall that images were smoothed by convolution with the appropriate template. Morphological operators may also be used to smooth an image. As described above, opening an image tended to remove bright peaks in the data whilst closing the image removed dark troughs. If both operators are used, then the extrema are removed, as shown in figure 4.19.

Figure 4.20: Morphological gradient

Figure 4.21: Top-hat transformed image

4.3.3.2 Morphological gradient

The morphological gradient is computed as the difference between the opened and closed image. It has the advantage of being less sensitive to orientation than the edge enhancement operators discussed previously, which computed edge strength in the directions parallel to the image sides. Figure 4.20 illustrates the results of computing the morphological gradient of an image.

4.3.3.3 Top-hat transformation

The top-hat transformation is defined by the difference between an image and its opened version. Its strength lies in its ability to enhance detail in an image that would otherwise be obscured by shading, see figure 4.21. The dual of the top-hat transform is the difference between the closed image and the original image; this will enhance slightly different structures.

4.3.3.4 Textural segmentation

Some uniformly textured images may be rendered smooth by repeated closing with progressively larger structuring elements. Consider the textured image of figure 4.22. Starting with a small structuring element, smaller than the blobs, we may iteratively close the image, at each iteration increasing the size of the structuring element. Whilst the element is smaller than the blobs, they will not be removed. Once

the element's size equals the blob size, the blobs will be removed.

If the image contained regions of differently sized blobs, continuing the operation would eventually render the entire image a uniform shade by gradually removing each region's blobs. Maintaining a record of the image's greyscale characteristics (mean grey value would be sufficient) during the closing operations will reveal the characteristic sizes of the blobs in each textured region.

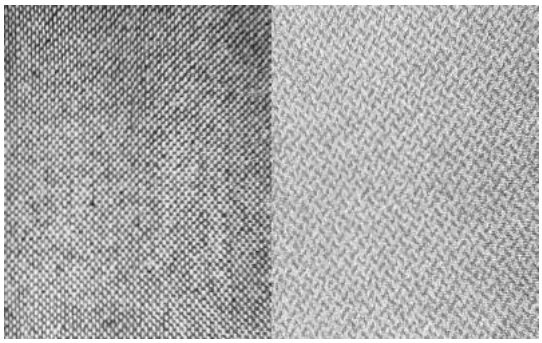


Figure 4.22: Texture segmentation

4.3.3.5 Granulometry

Granulometry is the study of determining the size distributions of objects, usually in images. In some images it is possible to isolate the particles being viewed and hence measure their dimensions. Such cases are rare. It is more common for particles to be adjacent or overlapping. Isolating individual particles is therefore impossible. By opening or closing the image (which operation is used will depend on whether the particles are brighter or darker than the background) will remove particles whose size matches the structuring element. Therefore, the difference between the original and the processed images will reveal information about the number of particles of that size. The image may therefore be characterised by iteratively opening or closing with increasingly large structuring elements. At each iteration, the difference between the original and processed images is computed. Finally, the size distribution is obtained.

4.4 Structural features

Several types of feature may be derived from binary images that are characteristic of the objects contained in them. The skeleton is a minimal representation of an object's shape. It includes just those pixels that define the structure of the object, excluding any extra detail such as brightness, colour, thickness etc. It is often computed using the distance transform. Finally, the convex hull is a minimal representation of the area covered by an object

4.4.1 Distance transform

The distance transform is used to estimate the minimum distance between each point in an object and the background. All versions of the distance transform will compute the city block distance between a point inside an object and the image's background.

One intensive method of computing the distance transform is to use a three by three pixel structuring element to repeatedly erode an object until it is completely removed. At each iteration a set of pixels will be removed from the object. The iteration number is the distance transform for these pixels, see figure 4.23.

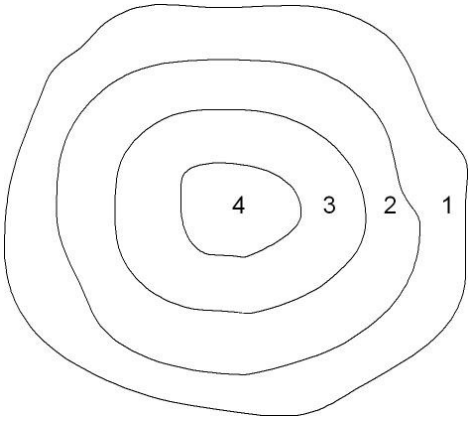


Figure 4.23: Distance transform using erosion

A slightly less intensive method uses the relationships of equation 4.13:

$$\begin{aligned} f^0(x,y) &= f(x,y) \\ f^m(x,y) &= f^0(x,y) + \min_{i,j}\left(f^{m-1}(i,j)\right) \\ i &= x \pm 1,\ j = y \pm 1 \end{aligned} \qquad 4.14$$

These simply state that the first iteration of the algorithm sets the pixels in the distance transform equal to the values of pixels in the original image (which will be zeroes and ones).

At each successive iteration of the algorithm, *m*, the distance of a pixel, *f(x,y)*, is increased by the minimum distance of its four nearest neighbours. The increments to be used will depend on the distance measure employed.

The distance transform may be useful when computing the skeleton of an image, as discussed below. Figure 4.25 illustrates its use in detecting the centre of the palm of the hand in a gesture recognition system.

Figure 4.25: Distance transform used to find the centre of a hand silhouette

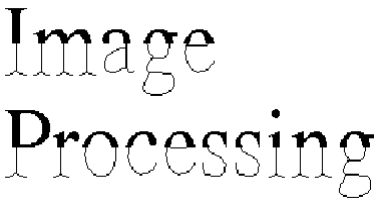


Figure 4.26: Skeletonisation

4.4.2 Skeleton

The skeleton of an object is a one pixel thick line that represents the object's shape. For example, the top halves of the characters in figure 4.26 are reduced to the skeletons of the bottom half. It is apparent that the skeleton retains the characters' shapes and topological relationships, but has lost all information relating to their thicknesses.

The skeleton must retain the following properties:

- Connected image regions must thin to connected portions of the skeleton
- The skeleton must be at least eight-connected
- The locations of ends of lines must be maintained
- The skeleton must approximate the medial line of the object
- Extraneous spurs introduced by thinning must be minimised

The medial line passes though the centre of the shape. This property is the most difficult to maintain due to the discrete nature of the data.

The skeleton is most often computed by repeated cycles of thinning operations. Each cycle consists of four applications of a thinning algorithm working from the top of the image to the bottom, the bottom to the top, left to right and right to left. These operations investigate a pixel and its eight neighbours. This ordering of the thinning is required to ensure that pixels are removed equally from all sides of the object.

During an application of the thinning algorithm all pixels are inspected in the context of their eight nearest neighbours. The pixel may be removed if:

- its removal does not result in two portions of the skeleton becoming disconnected,
- it is not the endpoint of a line and
- it has at least one background neighbour.

The first criterion ensures connected regions of an image remain connected in the skeleton. The second ensures that the endpoints of objects are maintained. Thinning from all four directions ensures the symmetry of the operation and helps to ensure that the skeleton approximates the medial axis.

Whilst this is an effective method of skeletonising an image, it is not necessarily very efficient. Other methods have been defined that perform the task more quickly, which are cited in the bibliography. One method of interest uses the information derived from the distance transform: since the skeleton is defined by points that are furthest from the background, the value of the distance transform at these locations should be maximal in some sense: it will either be a local maximum or will lie on a ridge. The adjacent points on the ridge will be the adjacent points on the skeleton.

4.4.3 Convex hull

An object can be described as convex if a line connecting any two points on the object boundary lies entirely within the object, see figure 4.27. The convex hull of an arbitrary object is the smallest convex object that contains it, see figure 4.28. The usefulness of the convex hull lies in its ability to define the area of the image that contains the object in question and to enable the discrepancies between it and the object to be identified. Both of these pieces of information are useful in the context of recognising the object.

One method of computing the convex hull relies on tracing the outline of the object. At each point on the outline, the angles defined by the tangent at this pixel and the vector between the current pixel and all other pixels on the boundary is computed, see figure 4.29. The next point on the convex hull's outline will be the point for which this angle is minimised.

Other, set theoretic, methods of determining the convex hull have been defined. The interested reader is referred to the references cited in the bibliography.

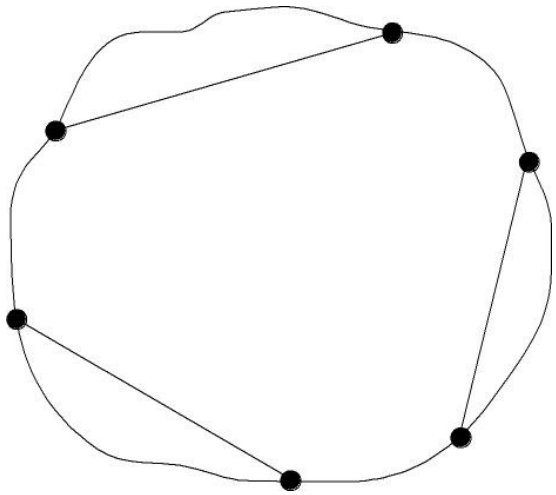


Figure 4.27: The definition of a convex object (almost)

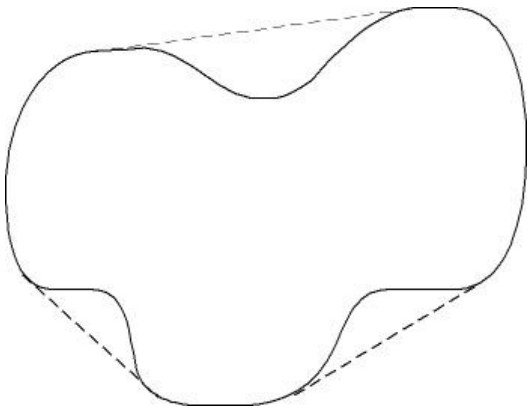


Figure 4.28: The convex hull of an arbitrary object

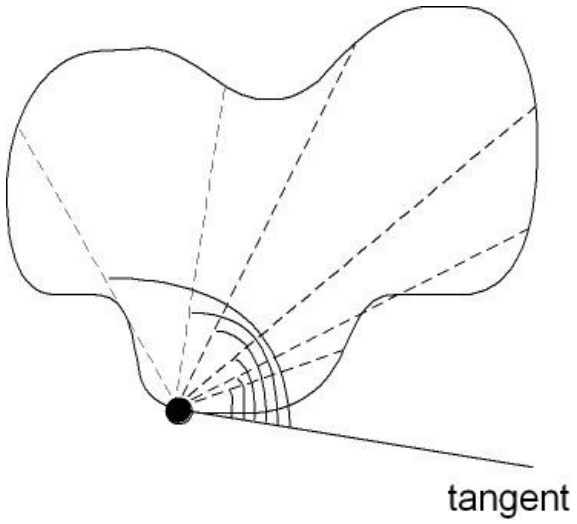


Figure 4.29: Computing the convex hull

4.5 Summary

This chapter has examined morphological image processing techniques. These are usually applied to binary images, therefore methods of reducing a greyscale image to a binary (bilevel) image were examined. The fundamental operations involved are erosion and dilation. These were defined and used to derive two further operations: opening and closing. Examples of the use of these operations have been given.

Morphological techniques are not exclusively applied to bilevel images. The modifications to the erosion and dilation operators to allow their application to greyscale images were therefore defined.

Finally, three methods of extracting structural information using morphological methods were discussed.

4.6 Bibliography

Serra (1982) was one of the originators of mathematical morphology applied to images. Dougherty (1992) also gives a useful summary. Parker (1997) discusses and evaluates various thresholding and other applications. Kalles and Morris (1993) presented a novel skeletonisation algorithm that was more efficient than any others at that time.

4.7 Exercises

1. Use one of the iterative methods to compute the optimum threshold value for separating an object from its background. Now repeat this exercise for another object. Are the thresholds the same? Can you explain why?

2. Use a thresholding function to investigate the effect of varying the threshold above and below the optimum defined by one of the iterative methods. Explain the results you observe.

3. Use a colour thresholding algorithm to identify skin coloured pixels in a head and shoulders image. How can these results be used to identify the person's face? How will the skin colour value change as a consequence of illumination and race?

4. Implement erosion and dilation algorithms and use them to improve the quality of your thresholded images from earlier exercises.

5. Under what conditions could the medial axis transform be used to plan the path of a mobile robot? (Think about the colour of the floor compared to the walls and other obstructions.)

Chapter 5
Region detection

Region detection is the process of dividing an image into separate and non-overlapping regions. All of the pixels in an image must belong to some region or another

A region can be specified by either defining the pixels that constitute it, or the pixels that bound it. In this chapter we shall examine methods that have been designed to break an image into its constituent regions using these two definitions as starting points. We shall pay particular attention to those methods that are based entirely on the pixel data: that is, methods that do not attempt to identify any objects in the image. A later chapter will examine methods that segment an image by searching for particular structures within in.

Having separated an image into regions, the next step in processing the image will be to identify those regions and the relationships between them. This is the subject of the following chapter.

5.1 Introduction

Region detection, also known as image segmentation, is the process of dividing an image into a set of non-overlapping regions that completely cover the image. A region might be

- an object,
- part of an object or
- the background

Figure 5.1 illustrates the type of results we could expect of a segmentation operator. Note that there are regions fitting into each of the three categories. This poses an interesting practical difficulty: how do we evaluate the success or failure of a segmentation function? An apocryphal story has it that a researcher was reporting the results of a simple experiment at a conference. He had sent a photograph of a simple object to a dozen of the world's leading computer vision laboratories, with the question: show the regions you would segment this image into. Not surprisingly, he received a dozen different answers. Most revealing was the comment, "I have drawn the regions you want, but they aren't the ones I see." This suggests that we cannot evaluate these functions in isolation, we must include the context of the task we are performing. Therefore, if we are designing an optical character recognition system, does the segmentation algorithm separate characters from non-characters? If we are building a vision system for a mobile robot, does the segmentation algorithm separate the flat surfaces from the rest of the data? Can we recognise the characters from the regions we have identified, and can the robot safely move onto the flat surfaces is has found?

In general, we may state that having segmented an image, every pixel in the image will have been assigned to a region, so the union of all regions is the original image; each region will be homogeneous, or self-similar in some sense and each region will be different to its neighbouring regions, or, alternatively, the union of two neighbouring regions is non-homogenous. These three criteria can be summed as:

$$\begin{aligned} &\bigcup_{i=1}^{n} R_i = P \\ &H\left(R_i\right) = True \\ &H\left(R_i \cup R_j\right) = False \end{aligned} \qquad 5.1$$

Where P defines a set of pixels that is being segmented (the image) and H some homogeneity predicate, which will return the value *True* for a region that is homogeneous.

Figure 5.1: Idealised segmentation results

A region is defined by the pixels that make it up. So we would say, "this set of pixels represents an object" in some arbitrary image. The identity of the region is of no consequence at this moment; it is simply our concern to be able to identify areas of self-similar pixels. This suggests one method of segmenting the image: accumulate all the adjacent and self-similar pixels into the regions they represent.

Alternatively, a region may be defined by the pixels that form its border, that is, the outline of the region. It is a simple matter to convert from one representation to another. A region's boundary is readily derived from the pixels constituting the region, we could erode the image using a three by three structuring element and subtract this from the original region which would leave the pixels constituting the boundary. Alternative, the area surrounded by the boundary could be filled using some form of scanline algorithm, thus identifying the pixels making up the region, figure 5.2.

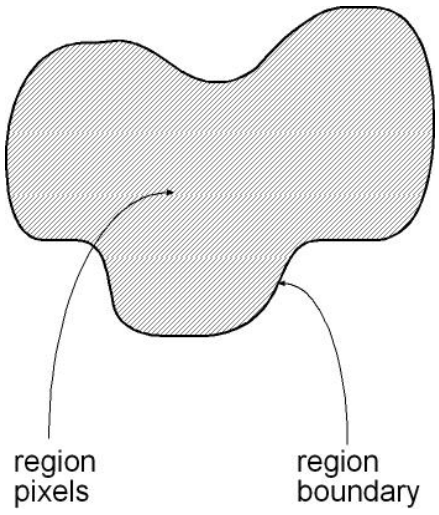


Figure 5.2: A region can be defined by contents of boundary

Region detection algorithms are also sometimes classified according to how they isolate the regions. Point based methods inspect individual pixels independently of their neighbours. If the pixel matches some global criterion, then it belongs to the region. Naturally, different criteria will be set for each region. After the initial labelling of the pixels, groups of contiguous pixels would be accumulated. Conversely, local methods will accumulate pixels into a region: starting from a single pixel we would inspect the neighbours, adding them to the region of they are similar.

In this chapter we shall discuss thresholding as an example of a point based method, and region growing, the split and merge algorithm and edge following as examples of local methods. Note that these methods divide the image into regions using only information available in the image; they are so called "bottom-up" methods. Segmentation algorithms have also been defined that search for specific objects within an image; for example, a security application might segment an image into regions resembling people and other regions. Such algorithms will always include application specific information, for example information regarding the appearance of people in the type of image being captured. They will be closely coupled to the task they solve; they will perform this task well, but no other task.

Figure 5.1 is a very idealised result image. In reality a segmentation process would not stop once an object has been found – how does it know the region corresponds to an object? Every region will be subdivided until further subdivision is unnecessary or impossible. We talked in chapter three about the scale of an edge. We have the same issue here, to what level should we segment the image: is it appropriate to identify the smallest components of a region or an agglomeration of the components? This is a problem that is addressed by scale based methods.

The final portion of this chapter will discuss methods of representing a segmented image.

5.2 Point based methods – thresholding

The methodology of thresholding was explained in the preceding chapter: what thresholding achieves and how it is realised. In this section we shall examine how these methods are applied to practical situations in which we are attempting to separate an object, objects or parts of an object from each other and the background.

5.2.1 Global thresholds

We have seen how a single threshold that is to be applied to the image as a whole may be computed. We have also seen how variable illumination may cause this single threshold to give incorrect results. If it is possible, by illuminating the scene carefully and by judicious choice of the background, we can ensure that the object and background are contrasting shades or colours. We may therefore guarantee that the image has a two-peaked (bimodal) histogram and a single threshold value will separate the object and background, as in figure 5.3. Unfortunately, we do not always have control over the scene's properties as figure 5.4 illustrates. This is the image of figure 4.3 with the threshold computed using one of the iterative methods. It is plain that the boundary is located correctly in some areas but not most. Computing thresholds locally was suggested as the solution to this problem.

Colour values can be used very effectively to isolate objects of known colour. Figure 5.5 shows how skin coloured regions are selected, the image is one extracted from a gesture recognition system where finding the hands is the first stage in being able to recognise a set of gestures. The method works by defining the colour of the skin and isolating those pixels whose colour values are similar.

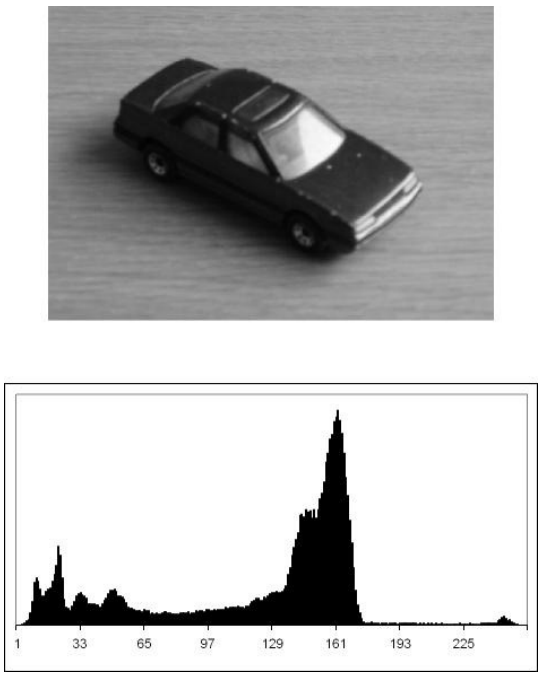


Figure 5.3: An image and its histogram. This is to illustrate that a unique threshold may not separate an object from the background.

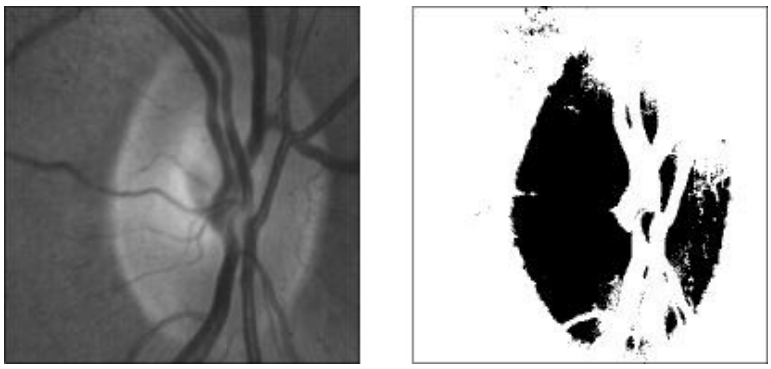


Figure 5.4: The image of figure 4.3 and its thresholded version, the threshold computed using the whole image data.

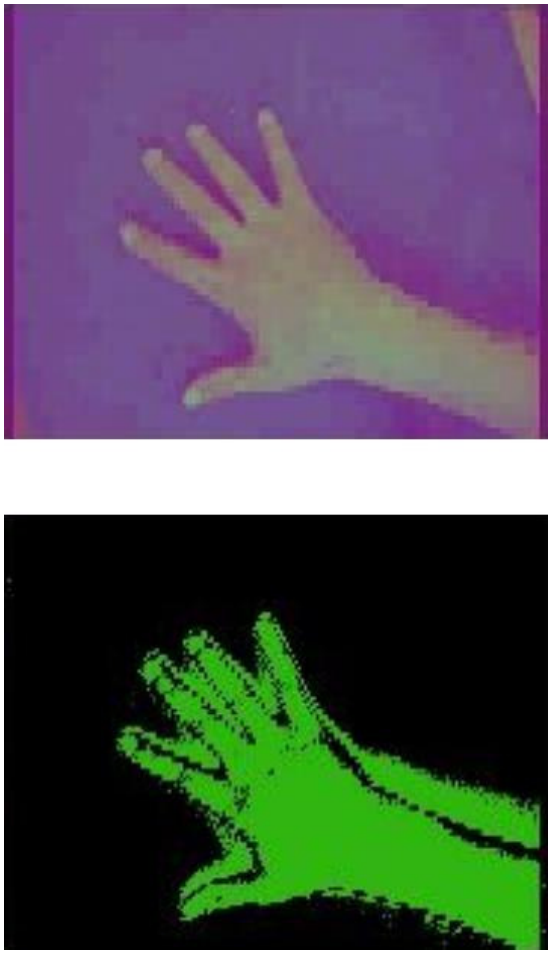


Figure 5.5: Skin colour thresholding. Using a Bayesian classifier based on the distributions of skin and background colours. The classifier has failed to identify shadow regions.

5.2.2 Local thresholds

Local thresholding methods are invoked in cases where the scene's illumination is spatially variant. They divide the image into sufficiently small regions that the illumination can be considered to be constant and therefore each sub region may be thresholded independently of its neighbours. Each sub region will contain background, object or a mixture of the two. This leads to two difficulties that must be overcome in order for this approach to be affective:

- If the sub region contains one component, we must determine which it is in order to set the threshold correctly,
- Adjacent sub regions that both contain two components may have different thresholds. This will lead to discontinuities in the object boundary at the boundary between the two sub regions.

Figure 5.6 shows a source image, divided into a number of sub regions and the histograms derived from each sub region.

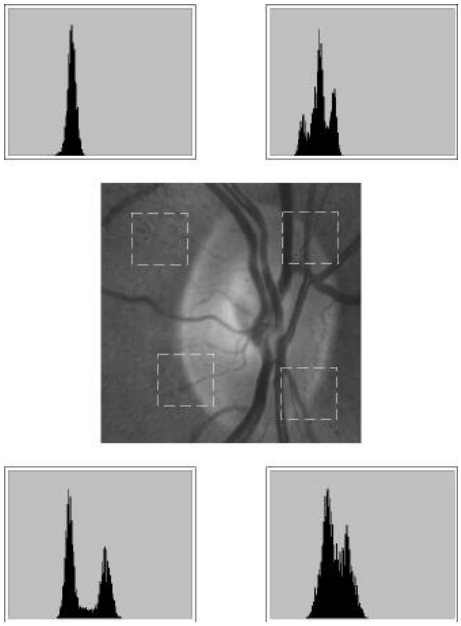


Figure 5.6: Local or adaptive thresholding. The histograms are derived from the indicated regions. (You should also note that some regions actually contain three components: optic disk, background and vessel.)

If a sub region contains one component, be it object or background, then its mean grey value will be significantly different to the image's mean. This will suggest the identity of the component in the sub region and therefore allow us to infer the appropriate threshold. We should also take notice of the suggested identities of neighbouring single component sub regions: they are likely to be the same.

Two neighbouring sub regions that contain both object and background are shown enlarged in figure 5.7a, figure 5.7b shows the histograms derived from the sub regions, the threshold values are indicated on the histograms. The final part of the figure shows the thresholded versions of the sub regions. Since the sub regions contained differing proportions of object and background, the two threshold values differed and consequently the boundary between object and background is discontinuous at the sub regions. The discontinuity is due entirely to the discontinuous nature of the threshold values, each sub region has a constant threshold value which may differ from its neighbour's. If the values merged smoothly across sub region junctions then this problem would not occur.

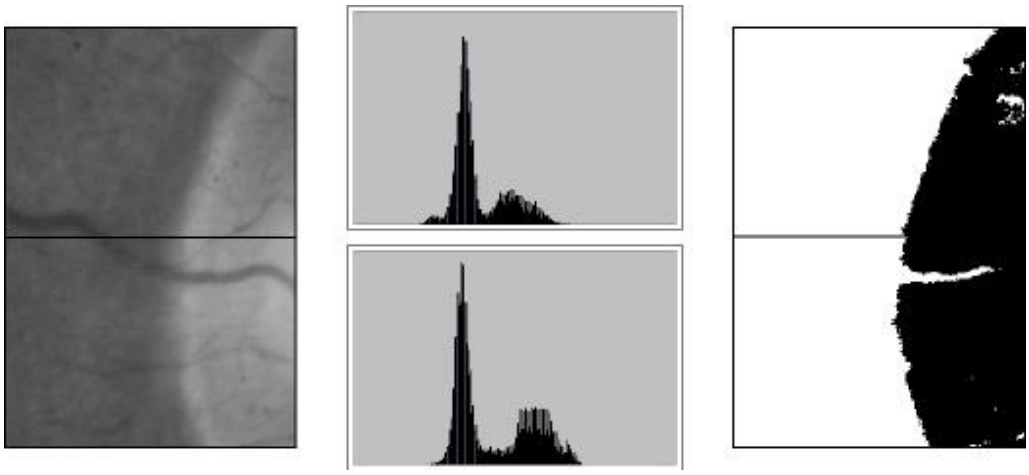


Figure 5.7: The problem with local thresholding: adjacent regions have different thresholds so the boundaries of the thresholded regions may not align.

5.3 Local methods – region growing

5.3.1 Region accumulation

Since all pixels belong to a region, we may select any pixel and it will be part of some region. Secondly, since a region is made up of adjacent pixels, this pixel's neighbours are possibly part of the same region. This is the essence of region growing algorithms.

The algorithm may be described in pseudocode as follows:

1 While there are pixels in the image that have not been assigned to a region.

2 Select an unassigned pixel and make it a new region. Initialise the region's properties.

3.0 If the properties of a neighbouring pixel are not inconsistent with the region:

3.1 Add the pixel to the region

3.2 Update the region's properties

3.3 Repeat from step 3.0 using the newly added pixel.

Until no further neighbours can b added to the region

4 Go to step 1

This recursive algorithm provides a rapid means of growing regions. Two issues must be addressed: what region properties are to be used? And how do we decide that a pixel should be part of the region (the test in line 3.0)?

Since we are dealing with individual pixels, it would seem sensible to use grey or colour values as the region's property. Therefore we would grow regions of similar shade of grey or similar colour.

The properties of the region that we will maintain will be the average grey value (or averages of the three colour channels) plus the standard deviation of the grey value (or covariance of the colour channels). We may then decide statistically whether a region's neighbours belong to the region by computing the Z score of the grey or colour difference. If we assume that the region's values are normally distributed, then a Z score less than two would lead us to conclude that the neighbour is part of the region. (The Z score is simply the difference between the region's and the neighbour's grey values normalised by the region's standard deviation.)

If we decide that the pixel does belong to the region, then the region's properties will be updated by accumulating the values into the region's mean and standard deviation.

It should also be noted that other properties could be used as a homogeneity predicate. However, any other property will involve a number of pixels and the regions that will be derived will reflect this. Texture has been used for this purpose; texture measurement is discussed in the next chapter.

5.3.2 Split and merge

The region accumulation method will grow an arbitrarily shaped region with boundaries that are accurate to the pixel level. The disadvantage with the method is the time taken to execute it. Even if the recursive step (3.3) is removed it remains a time consuming method of growing regions. The split and merge algorithm was suggested as an alternative: its execution is more rapid but the outlines it generates are less accurate. In implementing a region identification algorithm, a compromise must be made between these two properties.

As its name suggests, the split and merge algorithm has two parts, a splitting phase that divides the image in a deterministic fashion according to the results of an homogeneity predicate, and a merging phase that combines adjacent regions that are similar.

The splitting stage will apply the homogeneity predicate to a region of the image, initially the whole image is used. If the region is non-homogeneous, then it is split into quadrants and each examined by the predicate in turn (figure 5.8). The splitting will continue until either a region is homogeneous or it is too small for the homogeneity test to give sensible results.

The homogeneity predicates that will be used are texture measures that are applied to areas of the

image larger than a single pixel. The implementer will be aware of the minimum region size that each measure will require in order to yield accurate results.

Given the sample image of figure 5.9, the homogeneity predicate will determine that the whole image is non-uniform. The image is therefore split into quadrants. The predicate is applied to each quadrant in turn, some will be uniform and will not be further processed, and others will be non-uniform and will be divided into quadrants, which are, in turn, examined for homogeneity. The process terminates once all regions are uniform or too small to assess reliably.

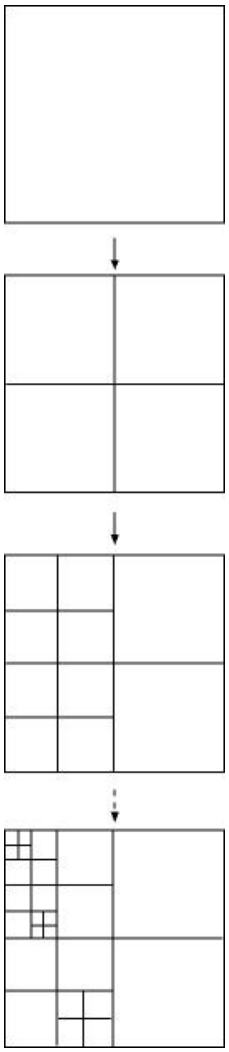


Figure 5.8: Split and merge – the splitting phase

At this stage, we can be sure that each region corresponds to a uniform area of the image. But uniform areas of the image may be represented by a number of adjacent regions, due to the deterministic nature of the splitting. The merging phase therefore inspects adjacent pairs of regions to correct this shortcoming. A pair of regions will be merged if

- They are adjacent
- They have similar greyscale or colour properties
- The boundary between them is weak

A boundary is considered to be weak if it is of low contrast, i.e. the difference in brightness across the

boundary is small in comparison with the average brightness, as shown in figure 5.10. Most authors also take account of the length of the weak boundary compared to the total boundary length: if the ratio is small, then the regions should not be merged, see figure 5.11

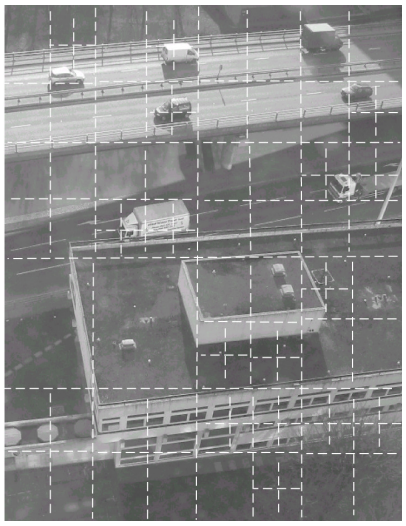


Figure 5.9: An example of splitting

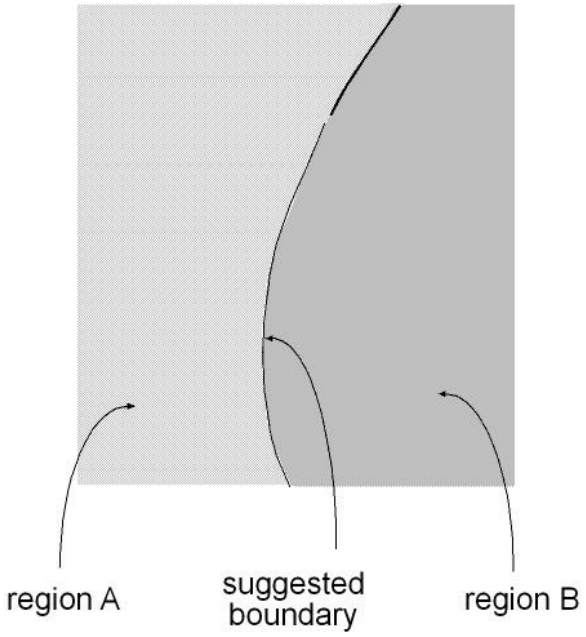


Figure 5.10: Definition of a weak boundary

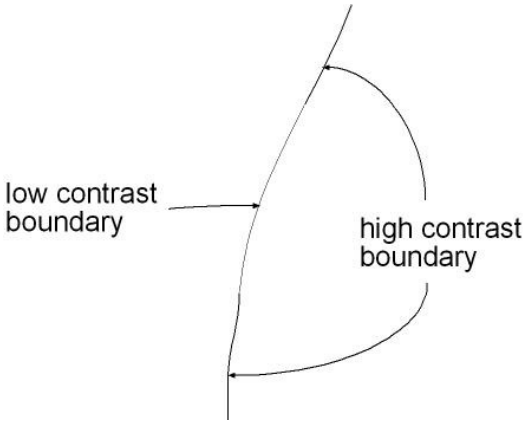


Figure 5.11: Merging adjacent regions by strengthening weak boundaries

5.4 Local methods – edge detection and following

5.4.1 Edge detection

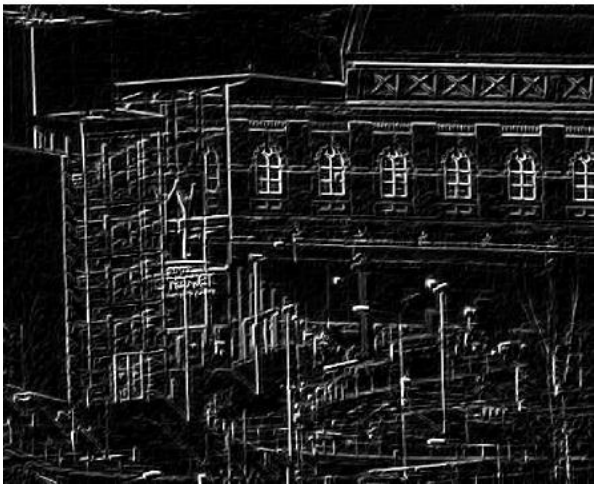
Edge detection as described in chapter three was a process that enhanced the edge structures in an image. The operator that performed the enhancement could have been one of the simple three by three pixel templates or one of the optimal detectors: the Marr-Hildreth or Canny operators were described. The result of the process is an output image whose magnitudes are proportional to the likelihood of the pixel actually being part of an edge in the image. Edge following is the process of tracking from one edge pixel to a neighbouring one and thus circumnavigating regions. In order to do this a binary image is required.

Figure 5.12 shows an image, its edge enhanced version, using the Sobel operator, and the histogram of the magnitude of the edge enhanced image. Some threshold value must be determined for this histogram. One of the iterative techniques was used to define a threshold to generate the binary image in 5.12d which is suitable for identifying the boundaries in the image.

This is a case in which the pixels are eight-connected: each pixel may be connected to any of its eight neighbours. A boundary is determined by finding one pixel on it by scanning the image in a systematic fashion, normally we would scan the image line by line starting at the top line. Thereafter, the boundary is tracked by passing from one pixel to one of its eight-neighbours until a junction or an endpoint is found, see figure 5.13. Tracking must then return to the start point and the boundary is tracked in the reverse direction, in this way we may gather the pixels that belong to boundary segments. The tracked pixels must be marked as tracked.

Having completed a boundary segment, the next segment will be tracked. Its start point is usually found by continuing the systematic search of the image from the previous first start point. Ultimately all of the unbroken segments in the image will be identified.

At this point the algorithm has defined a set of edge segments. Some of these might be so short that we may attribute them to noise in the image and delete them. The remainder must then be linked to define the outlines of regions. Two cases arise when considering how edge segments should be linked. The first, shown in figure 5.14a, is a bifurcation: the end points of three edge segments are co-located. The second, in figure 5.14b, occurs when a finite distance separates two endpoints.



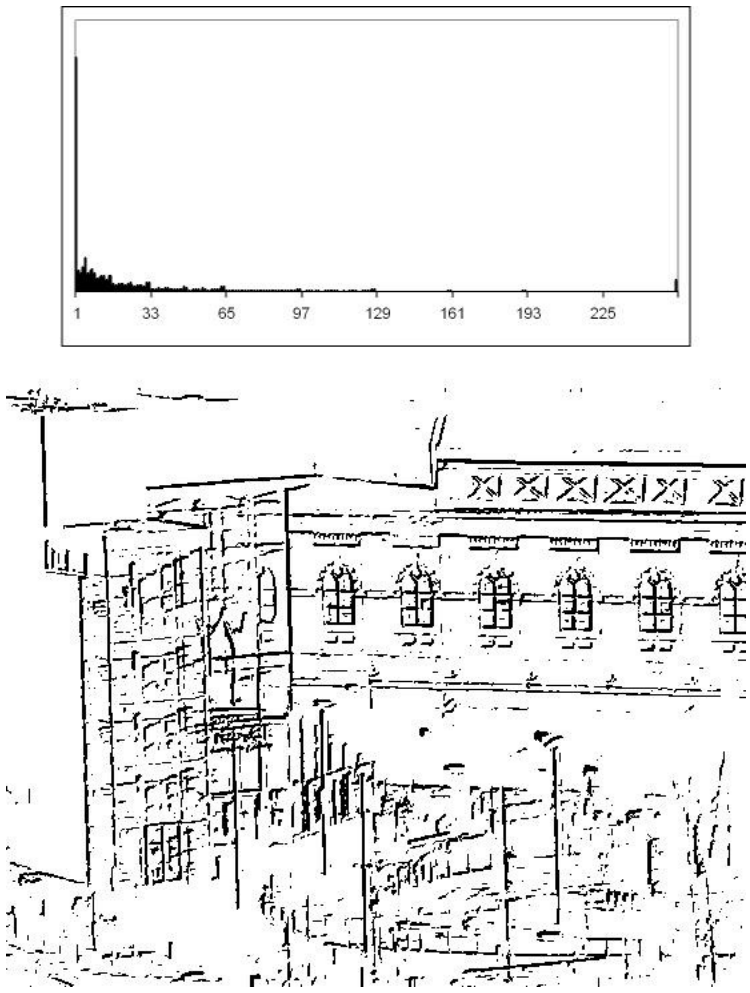


Figure 5.12: Identifying edge boundary candidates: input image, edge enhanced image, histogram of edge enhanced image, thresholded edge enhanced image

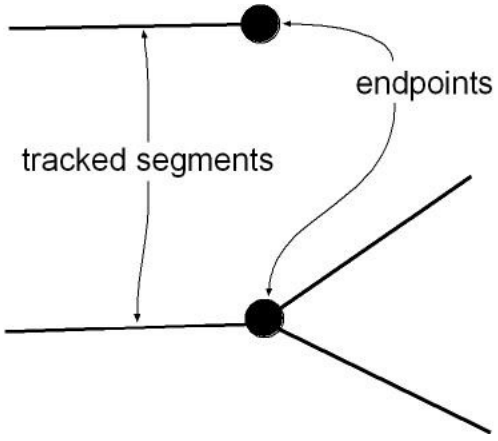


Figure 5.13: Boundary tracking

In the first case, as the diagram suggests, we would link the two segments that are most nearly collinear. In the second case, we would extend the two segments until they were linked. In this way we define a set of segments, many of which will form enclosed regions. Again, we may suggest that this

process is simplified if we have any information regarding the shape and structure of the objects being sought.

The structure of the segments may be simplified by replacing sets of edge segments by more abstract structures. This is the topic of the following section.

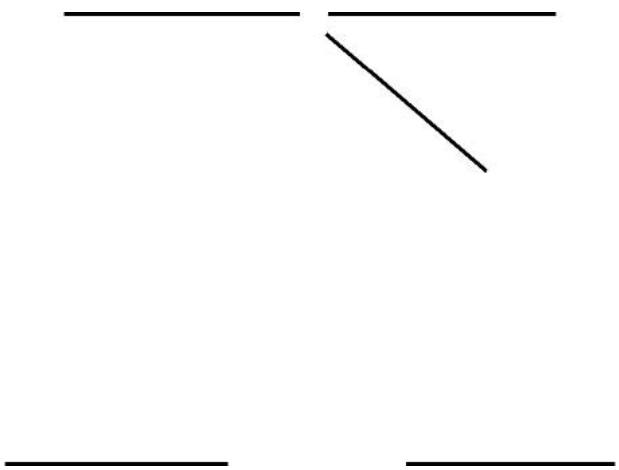


Figure 5.14: How should edge segments be joined, at a bifurcation and at a small gap?

5.4.2 Edge description

Edge segments have now been abstracted from the image. In many cases the object in the scene that produced the edge in the image has some regular structure: it may have a silhouette made of straight or regularly curved sections. A task that is sometimes performed is to replace a set of edge segments that approximate the real-world object, with a linear or curved segment that approximates the image data but is believed to be a truer representation of the real-world structure. We shall discuss two methods of replacing a set of edge segments by a linear approximation. The techniques may be adapted to detect curved structures.

Edge segments may be accumulated into a linear approximation by aggregating segments and computing the best straight line that fits the data. This may be achieved using a least squares minimisation technique. This technique not only computes the best-fit line, but will also provide a measure of the error in the fit, i.e. the total distance between the data points and the line. Accumulation will continue whilst this total error is below some threshold. Once the total exceeds the threshold, the line is terminated and a new one initiated, figure 5.15.

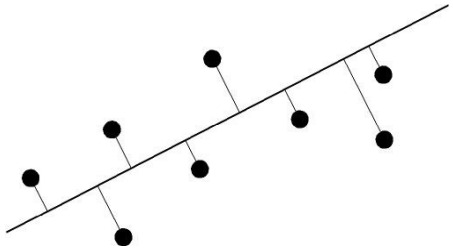


Figure 5.15: Accumulating edge segments into a linear approximation using least squares

Segments may also be accumulated using the so-called hop-along algorithm. This algorithm will function as follows:

1 The next *k* edge segments are taken.

2 A straight line is fitted between the first and last points.

2.1 The distance between the line and each point is computed. If the maximum distance is above a threshold, then the points up to the point of maximum distance become an approximated segment and the algorithm continues with the remaining points at step 2.

2.2 If the maximum error is below the threshold, this approximation is compared with the previous one, if they are collinear, they are merged.

3 The next *k* edge segments are collected and the algorithm returns to step 2.

The method is summarised in figure 5.16.

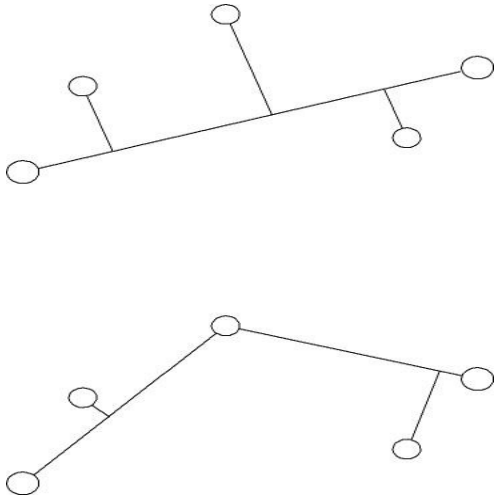


Figure 5.16: Hop-along algorithm

5.5 Scale based methods

5.5.1 Why scale?

Every object we observe has a characteristic scale or size: at either extreme of the scale range we have the universe and subatomic particles, in between are objects of varying sizes. An image may contain objects exhibiting a fairly restricted range of scales. Figure 5.17 shows a tree, the whole tree is seen at a large scale, the trunk and branches at a smaller scale and leaves (if there were any) at an even smaller scale. Images of many other objects will show similar ranges of scales.

Figure 5.17 An image to illustrate that structures exist at multiple scales

The image processing operations discussed in chapter three will process an image at the scale of individual pixels, and, as discussed in chapter two, this is related to the size of the smallest object that is to be detected in the image. Therefore, most image processing will focus on the smallest scale objects that are present in our data. This may or may not be an appropriate scale for processing the image. It is therefore sensible either to resample the image at a range of scales and process some or all of them, or to process the image using scale sensitive operators.

Images may be processed to create a pyramid representation of the original data. We may also generate a scale volume by applying a scale dependent operator to the data: two axes of the volume would correspond to the image's dimensions and the third to the scale parameter.

5.5.2 Pyramids

As figure 5.18 suggests, a pyramid representation of the image data will have at its base the original resolution data. Moving towards the apex of the pyramid, the resolution of the data will decrease, until, at the apex, we have the smallest resolution possible, which might be a single pixel or group of features. As mentioned above, two methods of generating the pyramid are available:

- image averaging and
- feature extraction.

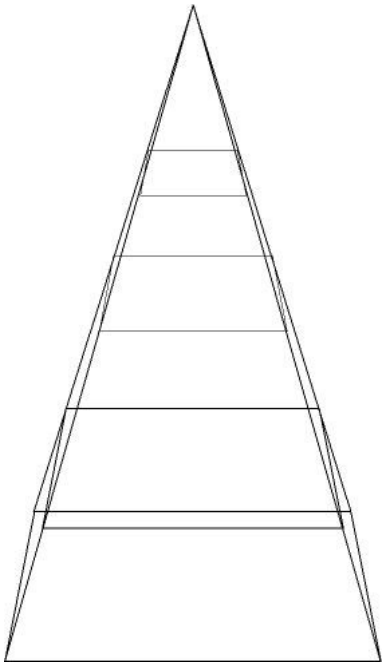


Figure 5.18: Pyramid representation of data

5.5.2.1 Image averaging

When generating a pyramid by averaging, pixels in a layer above the base layer are derived by averaging a set of pixels in the layer below. There will always be a spatial relationship between a pixel and the set of pixels that have contributed to it.

When designing a pyramid construction algorithm, three factors must be considered, see figure 5.19:

- How the set of pixels is weighted when computing the average
- How the sets of pixels tile the image
- The size of the set of pixels.
- In the simplest case, we can compute a uniformly weighted average of a set of four pixels and have no overlap between the sets. Then pixels in layer *i* of the pyramid are derived from layer *i-1* according to

$$f^{i}(x,y) = \frac{1}{4} \sum_{u=2x-1}^{u=2x} \sum_{v=2y-1}^{v=2y} f^{i-1}(u,v) \qquad 5.2$$

A slightly more complex case takes a weighted average of 16 pixels with an overlap between tiles

$$f^{i}(x,y) = \frac{1}{W} \sum_{u=2x-2}^{u=2x+1} \sum_{v=2y-2}^{v=2y+1} f^{i-1}(u,v)w(u,v) \qquad 5.3$$

where *W* represents the sum of the weighting factors *w(u,v)*, a smaller weighting would be given to the 12 boundary pixels than the central ones.

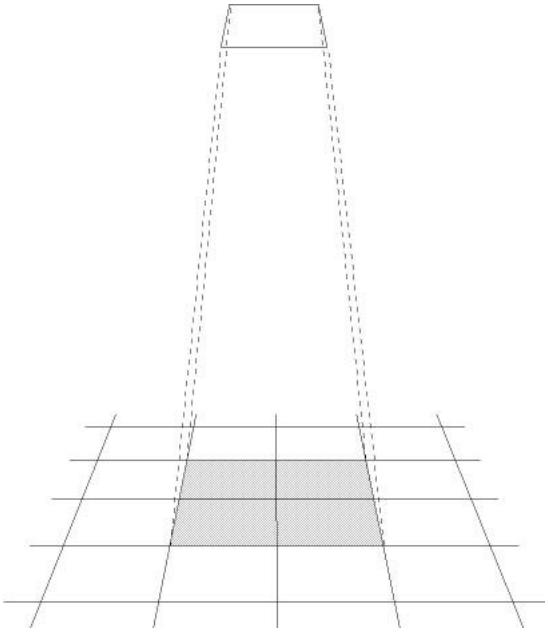


Figure 5.19: How sets of pixels are defined. Two questions must be answered: how do the sets of pixels tile the image? How are the members of the set weighted?

More complex tiling and weighting functions are possible, but few offer any great advantage.

Figure 5.20 shows the pyramid generated using the simplest scheme. Two properties of this method of representing the data are apparent.

- objects appear at a range of levels (scales) in the scheme
- each object has an appropriate level where it is most clearly visualised.

Consequently, although it is possible to select a level in the pyramid at which to search for each size of object, we cannot isolate an object using size information alone: if we select a scale suitable for an object and investigate that level of the pyramid, we shall also find information relating to objects of similar scales.

Despite this apparent disadvantage, generating a pyramid does allow more rapid processing of the data. We may select a level of the pyramid at which to start processing and find the approximate locations of object boundaries. The approximate locations will be refined as we shift attention towards the base layer of the pyramid, however, as we move down the pyramid, information from the higher levels will direct the search, it is therefore unnecessary to process the entire image.

Finally, the image pyramid requires more storage than the original data: almost twice as much.

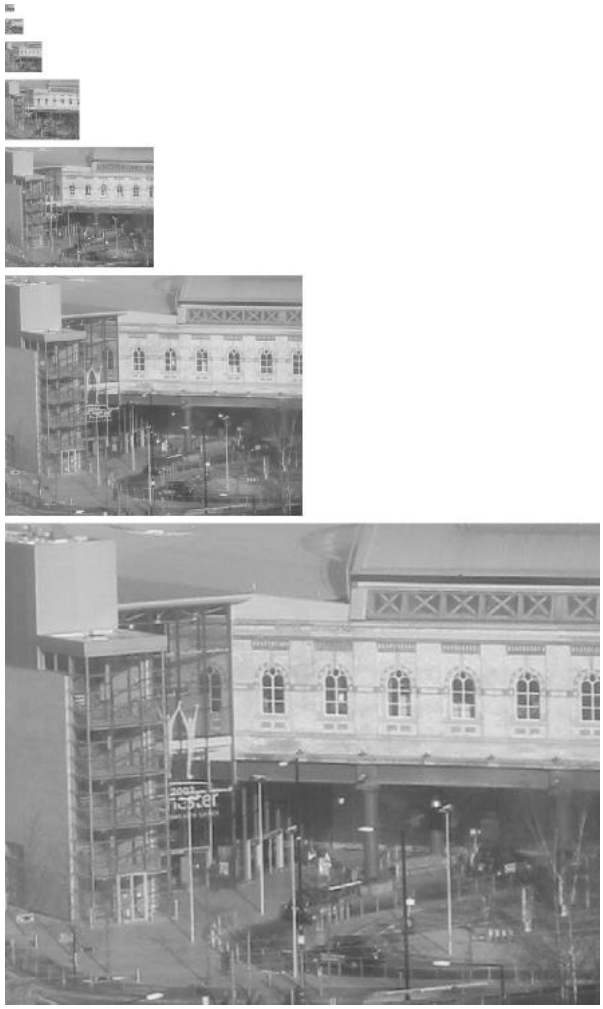


Figure 5.20: A simple image pyramid

5.5.2.2 Image smoothing

Applying a set of smoothing operators to the data may also generate a multiresolution version of an image. Instead of a pyramid with less information at each level, this will generate a volume of data as we stack the smoothed versions of the image above each other, figure 5.21.

Again it is not possible to associate a unique scale parameter to an object; each object will exist over a range of scales in the volume. This method of generating the multiresolution data offers little advantage: manipulating the information is simpler, but there is much redundant information to be stored.

5.5.3 Feature extraction

Creating the pyramid or scale-space representation is the first step in processing an image; subsequent processing stages would include processing the data to extract fundamental information, this will usually include some form of edge detection. Rather than create a multiresolution image pyramid and then perform feature extraction, it can be more efficient to perform multiresolution feature extraction, using some form of edge detector that includes a scale dependent parameter. The Marr-Hildreth and Canny operators include such a parameter in the form of the width of the smoothing function, σ.

Figure 5.21: Image scale space

By varying σ systematically, we may generate a scale volume that represents the features of the original data. As σ increases we locate progressively coarser features. This is as a consequence of smoothing the data with larger and larger smoothing kernels, see figure 5.22. Whilst σ may be varied continuously, similar values will generate similar smoothed versions of the original data. It can be shown that the sequence of σ values that yields the best scale space (best in the sense that adjacent images are most dissimilar) is given by:

$$\sigma = e^{0.37n}$$
$$n = 0,1,2,... \qquad 5.4$$

It is also instructive to view a cross section through a scale space generated in this manner shown in figure 5.23. In this figure, the scale space has been generated using the Marr-Hildreth operator with uniformly increasing values of σ. The zero crossings of the data have been located and are plotted. The plot illustrates how the zero crossings occur in pairs: one for each side of an image feature. At a sufficiently high value of σ, the zero crossings due to a feature will disappear as that feature is effectively removed by smoothing. Finally, the lines due to the zero-crossings cannot cross each other.

A similar scale-space can be obtained more efficiently by applying a simple edge-detecting operator to the image pyramid. The same comments will apply as were made at the end of section 5.5.2.1: the feature pyramid requires almost twice as much storage of the raw data, but allows features in the

original data to be localised more efficiently.

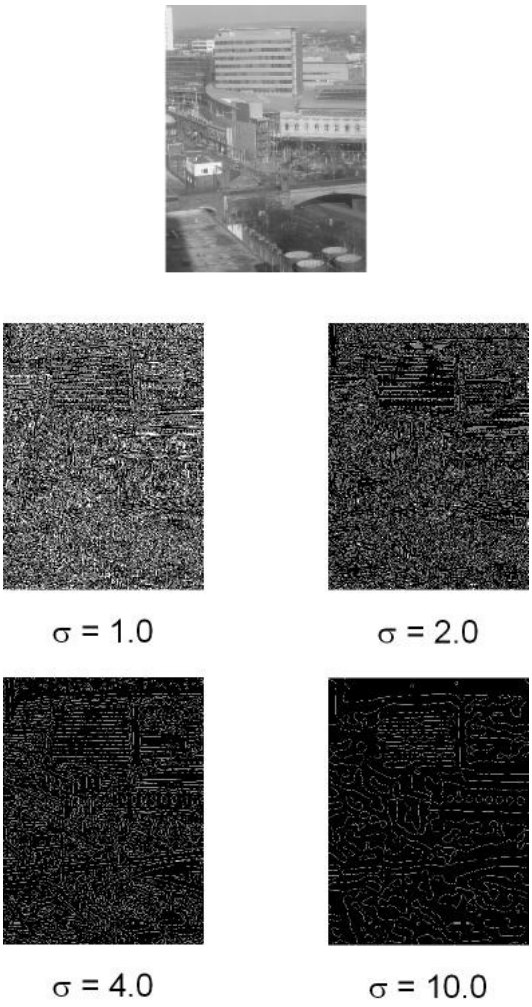


Figure 5.22: Feature scale space

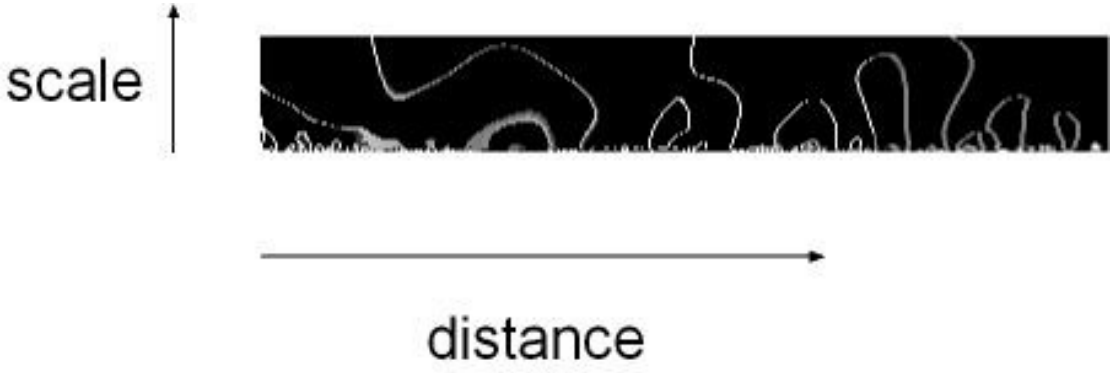


Figure 5.23: Cross section through a feature scale space

5.6 Representing regions

Region segmentation algorithms will identify regions of the image that correspond to different objects, parts of objects or the background. To be of any further use, these regions must be recorded: we must somehow be able to identify which pixels belong to which region. Three types of method have been suggested, two based directly on the image data and the third a hierarchical method.

5.6.1 Region map

The region map is considered to be the simplest but probably the most useful method of recording the regions that have been found in the image. It is simply an image of region identities. We define a numerical identity for each region which is placed in the equivalent pixel of the region map as that pixel is assigned to a region.

A variant on this scheme uses a separate image for each region that has been located. The image is a binary mask for that region: pixels with value 1 belong to the region, pixels with value 0 belong to other regions or the background.

5.6.2 Edge description

Since regions may also be defined by their outlines (boundaries), it is appropriate to use this information to delineate the regions of an image. Thus we could define the obverse to the region map: an image of the region boundaries.

Note that in many applications the exact boundary is not required, indeed in many real-time applications where speed is essential, the regions may be approximated by a bounding box, this is the smallest rectangle that completely encloses a region. Whilst the bounding boxes of adjacent regions often overlap to a considerable degree, the speed advantage that they confer more than compensates for any loss of precision.

5.6.3 Hierarchical representations

Hierarchical region representations are most frequently associated with hierarchical segmentation methods, notably split and merge. Quadtrees are the representation scheme most often used in this context.

A quadtree is a tree structure in which each node has zero or four child nodes, which are themselves quadtrees. The root node will correspond to the whole image and the leaf nodes to uniform rectangular regions. The linkage between regions (made in the merge phase of the algorithm) cannot be represented directly in this scheme, it must be a separate piece of information.

5.7 Summary

This chapter has been concerned with an examination of methods used to divide images into uniform and non-overlapping regions, which will correspond to some image structure. The following chapter will examine methods of extracting information from these regions that will be used in chapter seven to assign labels to the regions.

5.8 Bibliography

Threshold selection is a problem that is central to many autonomous systems and is therefore covered extensively, e.g. Shapiro and Stockman (2001), Sonka et al. (1999). Scale spaces were first suggested by Lindberg (1994) who has since been active in this area. Edge following and edge segment linking are topics that are covered in most computer vision texts, e.g. Castleman (1996) and Haralick and Shapiro (1992). Image pyramids were first suggested as a means of compressing image data by Burt and Adelson (1983), but have since become more widely used for machine vision applications.

5.9 Exercises

1. Think about consequences of using grey level thresholding to separate object with a view to counting and describing them and using an incorrect threshold.

2. If regions and boundaries are dual of each other, is one method more advantageous than the other? Why?

3. What will happen to the quadtree representation of an image if the image is scaled? Rotated? Translated?

4. Compute the number of pixels required to represent a simple image pyramid if the base is of size n by n pixels.

Chapter 6
Region description

Region description is the generic term for all processes that extract descriptive information from a region. This information will subsequently be used to identify the region by comparison with the information derived previously from known regions. In this chapter we shall examine the various types of descriptive information that may be extracted from a region.

6.1 Introduction

Previous chapters have discussed methods of representing images, improving their appearance and of identifying uniform regions. In common image processing systems, the next stage of the processing scheme is to extract information from these regions. For this information to be useful, it must be characteristic of the region and therefore capable of being used to assign an identity to the region. Labelling, that is actually identifying the region, will be the subject of the following chapter.

The method chosen to describe a region will be closely coupled to the problem being solved. When selecting a region description method it is therefore usual to work backwards by asking:

- What problem am I trying to solve?
- What information do I need to solve this problem?
- How do I find that information from the image?

For example, if we were asked to design a system to differentiate between potatoes and carrots, we might decide that the ratio of the maximum and minimum dimensions could be a reliable method of separating the vegetables because carrots are longer and thinner than potatoes which are usually more compact. The designer must then solve the problem of how to measure these quantities and what operations must be performed in order to extract information from the image that will enable the measurement to be made.

In this chapter we shall examine some of the methods that have been suggested for achieving this task. We shall start by examining the features that can be derived from binary images, notably measurements of area and perimeter shape. We shall discuss methods of representing the outline of an object.

6.2 Features derived from binary images

Binary images provide a good starting point for deriving simple features that can be used to describe objects. The skeleton was presented in an earlier chapter as a means of giving a compact description of an object's shape. In this section we shall firstly examine how to group the pixels that belong to a single region and secondly discuss how the area and perimeter of the objects are computed. The following section will discuss methods of describing the shape of the object.

6.2.1 Connected component analysis

A contiguous set of pixels that share some property is known as a connected component. For example, the regions of a binary image are each connected components. The process of finding these components is known as connected component analysis. Several methods have been suggested for performing this task. One of the simplest is to use the dilation operator described in chapter four, this has the effect of adding a layer of pixels to a region at each iteration through the image; it is therefore an extremely slow method of identifying components. A recursive region growing can be used as defined in section 5.3.1, this can grow regions very efficiently, especially if the tail recursion is removed.

An alternative method of identifying the pixels in a connected component uses a two pass algorithm. In the first pass we are looking for pixels that could belong to a component. Each pixel is labelled with a numerical component label as it is found, the label it is given will be dependent on the labels of any previously labelled pixels, see figure 6.1. As we are searching systematically through the image, only

those pixels in the line above and to the left of the current pixel can have been labelled. Three possibilities can occur:

- none of the adjacent pixels (pixels A, B, C and D in the figure) has been labelled: the pixel is given the next free component label;
- one or more of the adjacent pixels have been labelled but have the same label: then the pixel is given this label;
- two or more adjacent pixels have been labelled with differing labels: the current pixel is given one of the labels and the equivalence between the newly connected regions' labels is recorded.

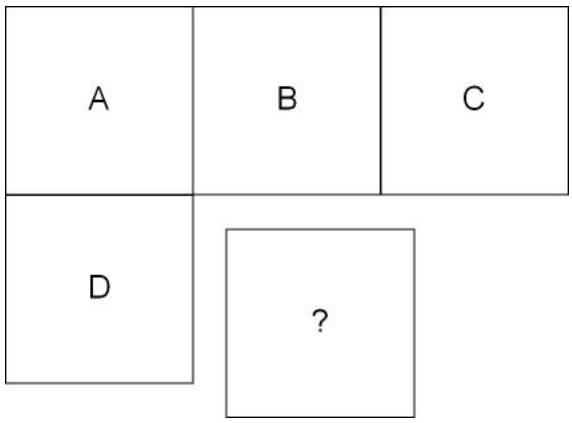


Figure 6.1: connected component labelling. How the new pixel should be labelled will depend on the labels given to pixels A to D.

After the first pass, we will have the cases illustrated in figure 6.2: some regions will have a consistent label, i.e. all pixels in the region will have the same label. Other regions will have inconsistent labels: one region with portions having different labels, we will have recorded the sets of labels that are equivalent in an equivalence table.

The second pass will simply relabel the inconsistent labels according to the equivalence table. A set of contiguous labels is not required but can be achieved by examination of the equivalence table. The status of the labelling after the second pass should be as figure 6.3.

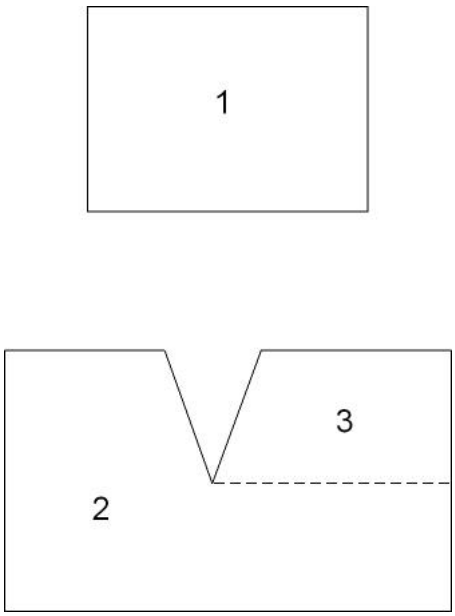


Figure 6.2: Situation after the first pass of labelling. Some regions have a single label, others have multiple labels. (This situation can appear recursively, i.e. the regions labelled 2 and 3 might also have multiple labels.) The second pass will rectify this.

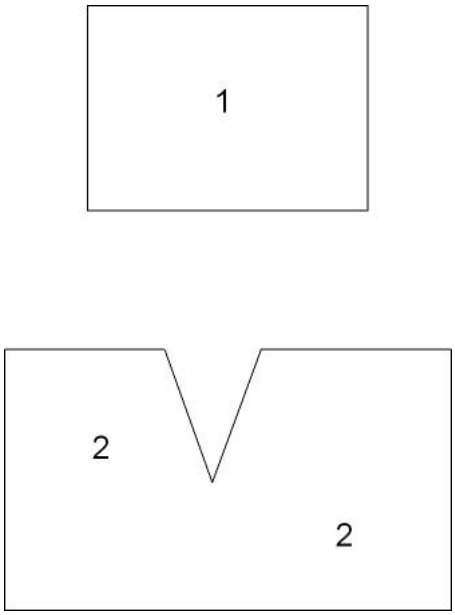


Figure 6.3: Situation after the second pass. The equivalence table was built and indicated that the following mapping of labels should be applied: 1→ 1, 2 → 2, 3 → 2

6.2.2 Area and perimeter

The area of a region is simply the number of pixels making up the region. Whilst this can be computed by counting the number of pixels as the region is grown, a simpler way is to compute the histogram of the labelled image, such as the one of figure 6.3. Figure 6.4 is the histogram of the labelled image of figure 6.3. The histogram entries are zero, apart from the entries that correspond to the labelled regions; these values are equal to the number of pixels in the region.

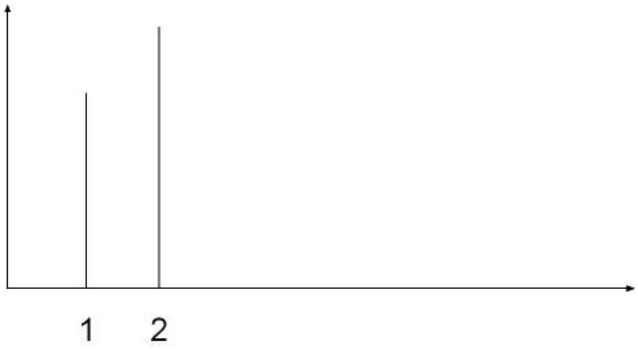


Figure 6.4: Histogram of the labelled image

The perimeter of a region is the total length of its outline, figure 6.5. A discussion of how to derive the perimeter will be postponed until we discuss chain codes below. At this point it is worth noting that a compactness measure, C, may be derived from the region's area, A, and perimeter length, p:

$$C = \frac{p^2}{A} \qquad 6.1$$

This is a dimensionless quantity that is large for non-compact regions and small for compact ones, figure 6.6 gives three examples.

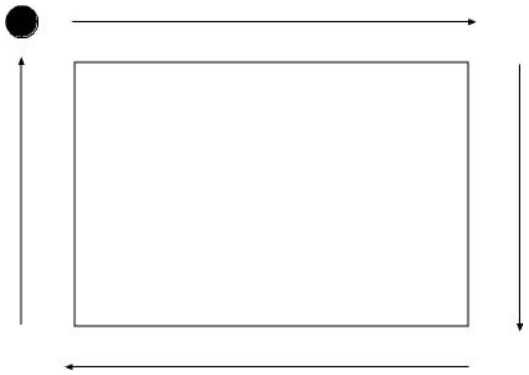


Figure 6.5: Definition of perimeter

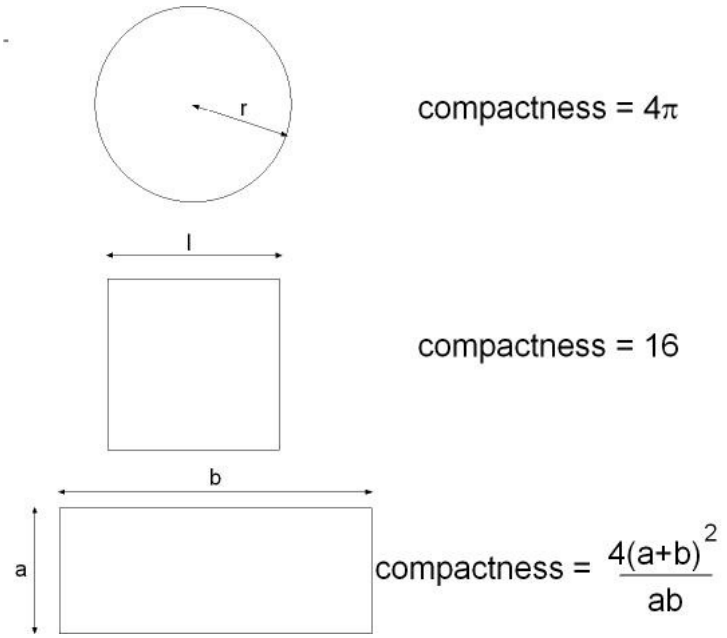


Figure 6.6: Compactness for three simple shapes

6.3 Outline description

In the previous chapter we examined methods of tracing edges, in an attempt to locate the boundaries of regions. We also examined methods by which these boundaries might be simplified. In this section we shall examine methods of extracting and describing the boundaries of regions derived from binary images. These methods will range from ones that are very closely related to the image data to ones that are much more abstract. The abstract methods are, in effect, attempts to remove the effects of image capture from the data; image capture will always introduce degradations to the data since it is an inherently noisy and approximating process. The abstract representations are an approximation to the data that we have captured, and we hope that this approximation is actually closer to the reality; this is justified since the noise is a random process and is equally likely to disturb pixels on either side of the boundary.

6.3.1 Chain codes

Chain codes were one of the earliest methods that were derived for describing the outline of a region. A chain code representation of a region will include the image co-ordinates of one point on the boundary of the region, plus a sequence of displacements that will take us from one pixel to the next connected one on the boundary and eventually return to the starting point, see figure 6.7. The displacements are coded systematically as in figure 6.8, so a displacement towards the top of the image is always represented by a code “0”, and so on. The direction in which the boundary is traversed is not important, provided the same direction is used consistently. Eight connected chain codes are most frequently used as they give a less jagged boundary.

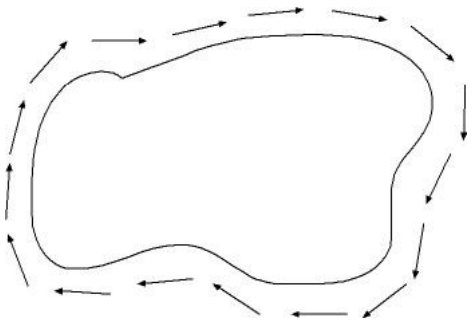


Figure 6.7: Chain code of an arbitrary shape

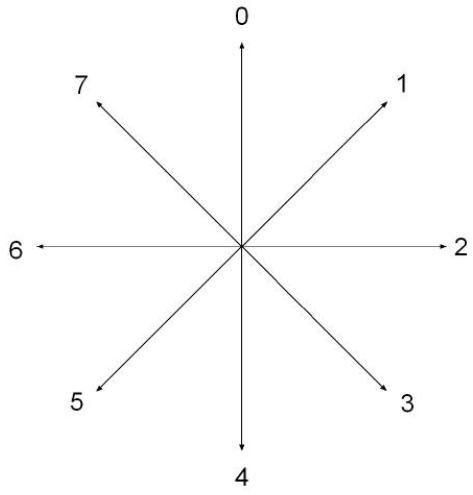


Figure 6.8: Chain codes used in 8-connectivity

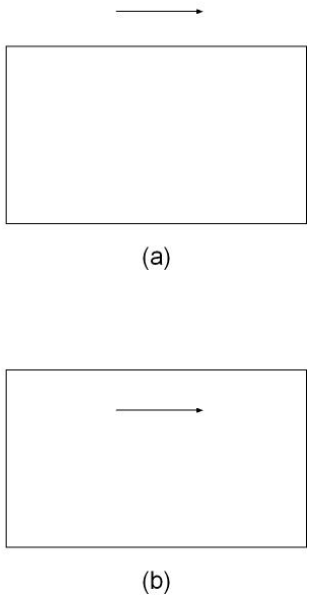


Figure 6.9: On the boundary – outside or inside the image. "On" the boundary is impossible as it's the crack between pixels

The phrase "on the boundary" must be clarified as we may use one of two alternatives. The boundary pixels could be pixels inside the region that are adjacent to at least one background pixel, see figure 6.9a, or they could be pixels in the background that are adjacent to at least one region pixel, figure 6.9b. Again, which alternative is chosen is not critical, provided it is used consistently, as measurements derived from the chain code will differ according to which method is used.

Chain codes provide a position independent means of representing the outline of an object. Wherever the object is located in the image, the chain codes defining its outline will be the same but the co-ordinates locating the object will vary. It is also possible to modify the codes such that they become orientation independent by making the chain codes' directions relative to the tangent of the boundary instead of fixed to the image's sides, as shown in figure 6.10. This is occasionally used when this property is desirable, however, it does make deriving region properties a more involved process.

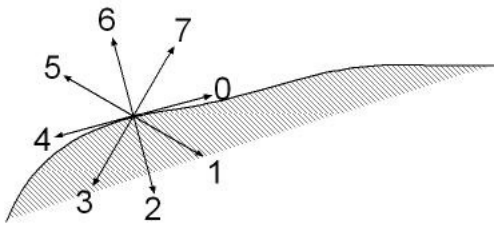


Figure 6.10: Direction independent chain codes

The perimeter and the area of a region are readily derived from the chain-coded boundary. Using eight connected codes, even numbered codes represent a distance equal to the separation between pixels. Odd numbered codes represent a distance equal to $\sqrt{2}$ times the pixel separation. The total perimeter length is therefore the number of even numbered codes plus $\sqrt{2}$ times the number of odd numbered codes.

The area of a region is estimated by integrating the contributions due to each chain code. Figure 6.11 indicates how each code's contribution to the total area is computed. Starting from an arbitrary baseline, which could be the *y* co-ordinate of the codes' starting points, and moving in a clockwise direction around the region, we have the following four cases:

- codes with a start point above the baseline and moving to the right will make a positive contribution to the area,
- codes with a start point above the baseline and moving to the left will make a negative contribution to the area,
- codes with a start point below the baseline and moving to the right will make a negative contribution to the area,
- codes with a start point below the baseline and moving to the left will make a positive contribution to the area,

The codes in figure 6.11 all fall into the first two categories: codes 1, 2 and 3 into the first and codes 5, 6 and 7 into the second. The two codes that represent displacements parallel to the y axis contribute nothing to the area.

If these codes had been below the baseline, their contributions to the area would change sign.

Finally, the magnitude of the area contributed by each codeword is the area of the quadrilateral bounded by the baseline, the displacement and the parallel lines joining the start and end points of the displacement to the baseline. The parallel lines will either be coincident (in the cases of codes 0 and 4) and so these codewords will contribute nothing, or they will be separated by a distance of one pixel. Otherwise, the area is made up of the rectangle to the start of the displacement (codes 2 and 6), or this rectangle plus or minus the triangle defined by the displacement (codes 1, 3, 5 and 7).

Using these formalisms for perimeter length and area, these quantities can be derived as the region's boundary is being traced. This is obviously an efficient and rapid method of computation.

Finally, it was stated above that the region's boundary could be defined by the pixels immediately inside or outside the region. It is also possible to define the boundary by the cracks between the pixels that span the boundary as in figure 6.12. Although this gives a more accurate representation of the boundary, the additional expense incurred means it is not often used.

Chain codes have been used as the basis of deriving higher level structures in the image. Particular groupings of codes may be recognised as more complex structures, for example linear segments may replace a set of approximately linear chain codes, or a larger scale curve might replace a set of codes that approximate the curve.

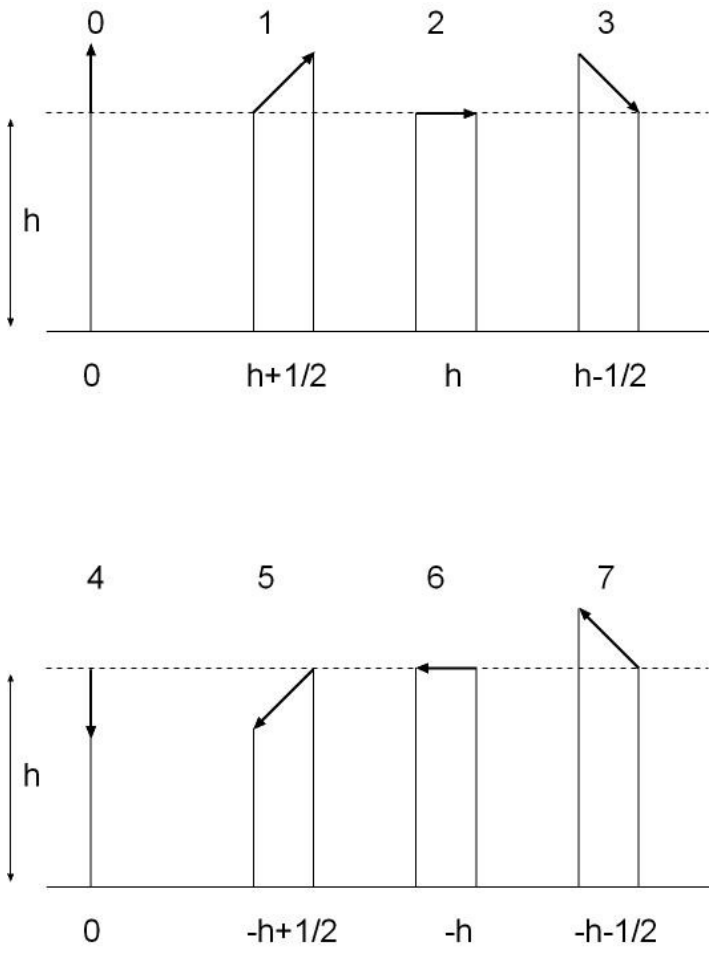


Figure 6.11: Area from chain codes

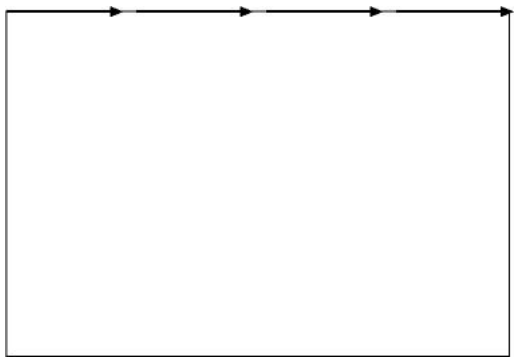


Figure 6.12: Crack coded boundary

6.3.2 Linear segments

The chain codes give a boundary representation that is accurate to the pixel level. The major disadvantage of this is that if the image is affected by noise, as it will be, then this will be at the pixel level and the chain codes will be directly affected, as shown in figure 6.13.

This difficulty can be avoided if we replace approximately linear segments of the boundary by straight line segments. The boundary of an object is thereby reduced to a polygon.

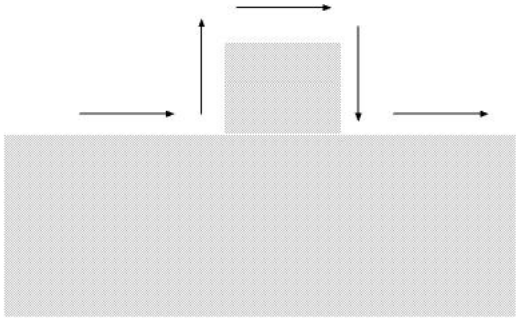


Figure 6.13: Chain codes affected by a single noise pixel

This description is achieved in two stages, starting from a list of the pixels that constitute the boundary. In the first stage, sections of the boundary are divided into linear segments. In the second stage adjacent segments are merged, if necessary.

A section of boundary is represented in figure 6.14. It has been selected by choosing an arbitrary set of connected pixels from the boundary list. This is reduced to a polyline, i.e. a set of linear segments by firstly computing the distance from each point onto the line. The maximum distance is found, if it exceeds some threshold, then a vertex is inserted to bisect the original line. The process is then repeated for the newly formed segments. The threshold will, within limits, be a function of the line length, shorter lines will require a larger threshold.

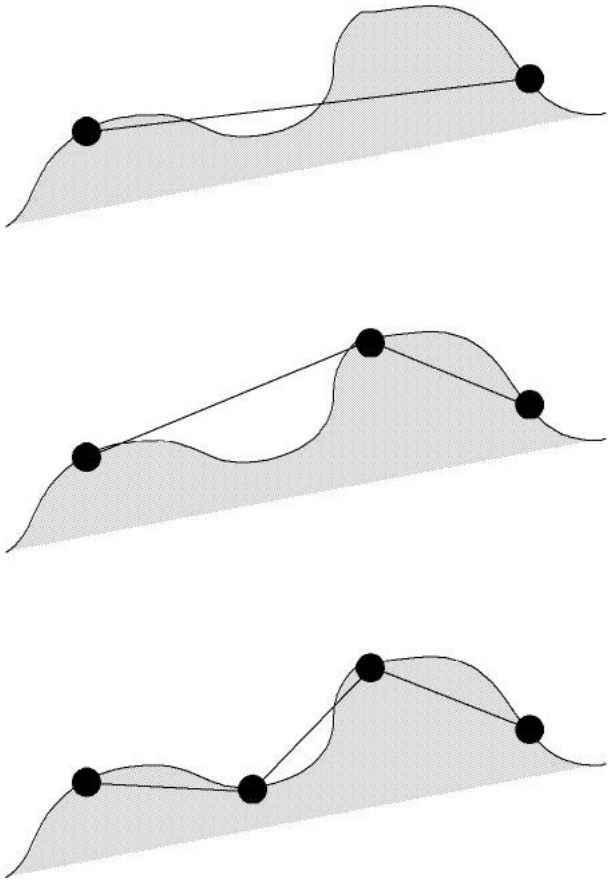


Figure 6.14: Polyline splitting

The second stage of the process will investigate adjacent line segments that were split at the boundaries

introduced in the first stage, i.e. the segments preceding and succeeding the segment of figure 6.14. If these segments are collinear, then they should be merged.

The polyline representation is ideally suited to objects that have boundary made up of linear segments. If any portion of the boundary is curved, then this representation becomes inefficient, i.e. many small segments are generated. Inserting curved sections instead of linear ones may ameliorate this situation. Conic sections have been used or the boundary can be approximated using a spline curve.

6.3.3 Spline curves

Splines were suggested as a method of approximating curves for computer aided design systems used in the motor manufacturing industry. They provide a compact representation method in that two end points and two intermediate control points represent one curve section. Adjacent curve sections will share end points to ensure continuity, see figure 6.15. Provided the control points can be identified reliably, this is an extremely compact method of representing curved sections.

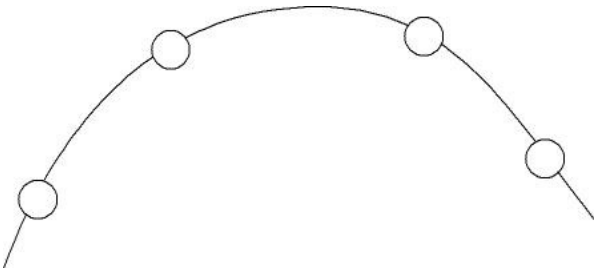


Figure 6.15: Spline curve

6.3.4 Phi-s codes

The phi-s code is a representation of an object's border that creates a plot of distances from a point to the border as a radius is rotated about the point, shown in figure 6.17. Whilst any point could be chosen as the centre of rotation, it is sensible to make a consistent choice, the region's centre of gravity is often used. Then simple shapes generate simple curves: simple geometric objects may be recognised easily from their phi-s curves by the number of maxima on it, see figure 6.18.

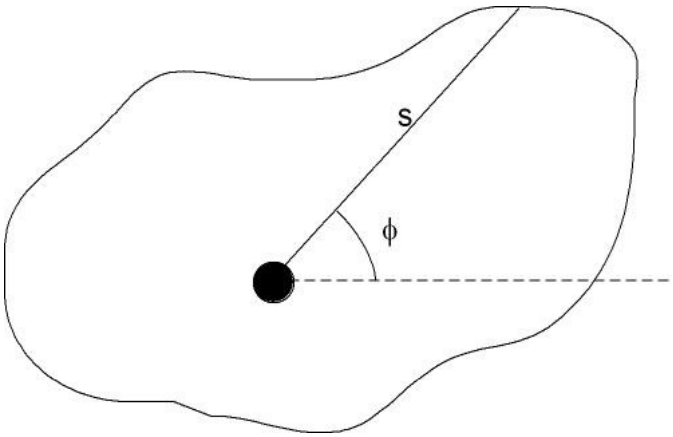


Figure 6.17: Phi-S curve, definition of parameters

Real objects will generate more complex curves, but they may still be recognisable, as in figure 6.19. Recognition will be achieved by comparison of the curve with the curve derived from known objects.

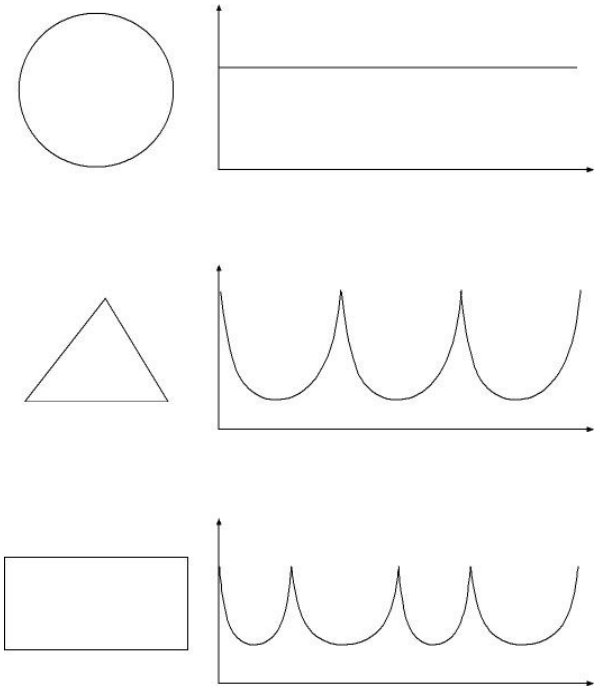


Figure 6.18: Phi-S curves of simple shapes

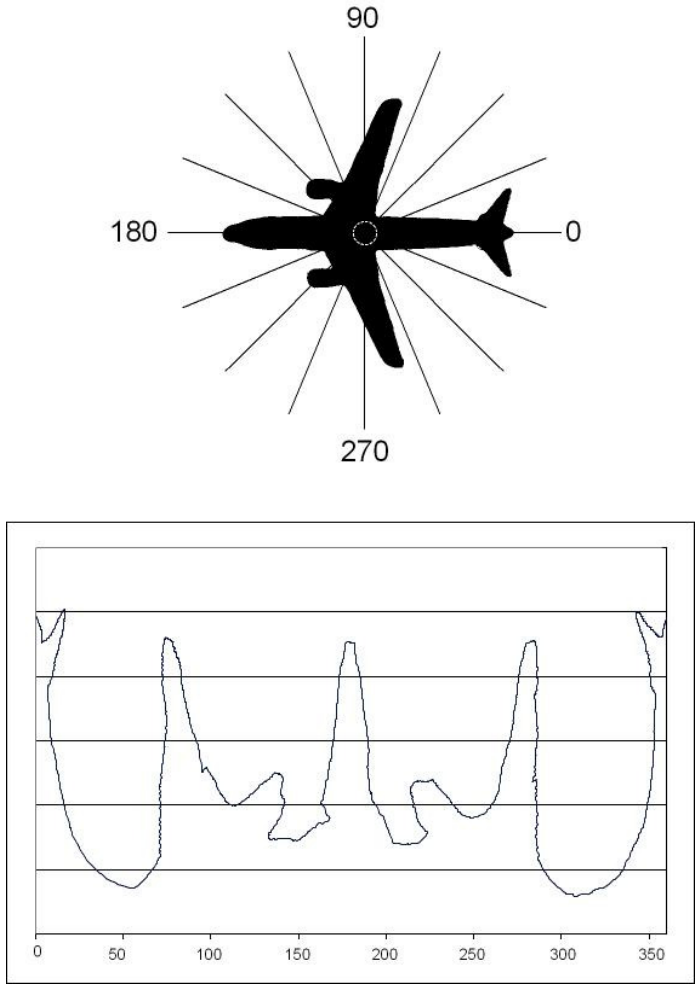


Figure 6.19: Phi-S curve of a real object

6.3.5 Boundary segments

Object boundaries can also be described as a sequence of structural building blocks that are extracted from the primitive boundary description elements. One of the first examples was used in a system for classifying chromosomes, see figure 6.22. The chromosome image was thresholded to isolate the chromosomes from the background. The outline of each chromosome was traced and the chain codes so derived translated into one of the building blocks shown.

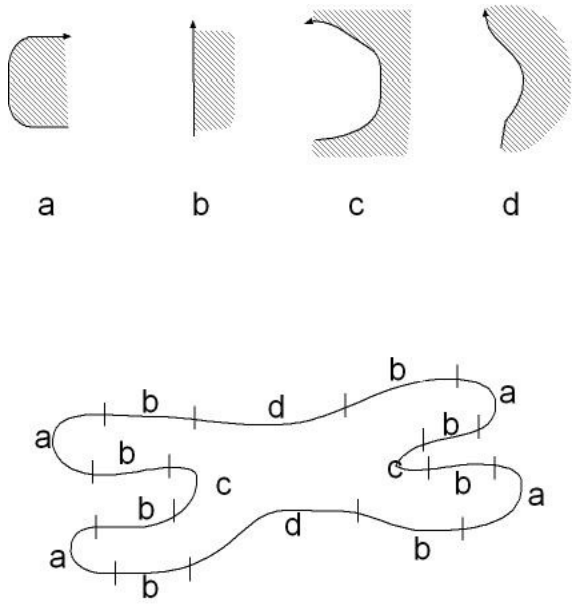


Figure 6.22: Shape primitives used to represent a chromosome outline

The strength of this approach is its immunity to scale, rotation and translation: no transformation will have any effect on the description, beyond changing its start point. Further, comparatively simple rewrite rules may be used to recognise the chromosome, as will be described in the following chapter.

6.4 Summary

This chapter has been concerned with the discussion of methods of describing regions, that is, regions' boundaries and the structure of regions. A description of a region is only of use for labelling the region, which is the subject of the following chapter.

When choosing a feature to be used for object description, four factors should be considered:

- The features should take significantly different values for objects belonging to different classes, i.e. maximum object length could separate carrots and potatoes.
- The features should take consistent values for objects of the same class, i.e. potatoes should have similar maximum lengths.
- The features derived should be uncorrelated, otherwise their discriminatory power is reduced.
- The smallest possible number of features should be used, consistent with producing correct results. A system's complexity increases dramatically as the number of features increases: the number of samples required to learn the characteristics of a problem increases exponentially, the system's performance can even be degraded by adding more features.

6.5 Bibliography

Chain codes were first investigated by Freeman (1961, 1974) but continue to receive attention. The representation of outlines using polygonal segments has been investigated by, for example, Voss and Suesse (1997) and Rosin (1997) and many others. The chromosome analysis is due to Ledley (1964) and is a very early example of computerised pattern recognition.

The phi-s curve is described by Ballard and Brown (1982)

6.6 Exercises

1. Implement the connected component labelling algorithms and compare the times taken to label an image using the two methods.

2. Compute the chain code of the shape in figure 6.33. Suggest a method by which the description could be made to be independent of the shape's size.

Chapter 7
Region labelling

Region labelling is the process of associating a label with a region of the image. Formally we may state the problem as: given an image containing one or more objects of interest and a set of labels corresponding to a set of models known to our system, how do we assign the correct labels to regions?

7.1 Introduction

The previous chapters have discussed methods of identifying regions and extracting information that was in some sense descriptive of them, whether that was the boundary of the region or statistics derived from the region itself. The present chapter deals with methods of recognising the regions, i.e. associating a label with the region by comparing the description of the region with the description derived from known samples, that is comparing the processed data with a model. Humans perform this task effortlessly, rapidly and usually correctly. However, describing how we perform this task has been surprisingly difficult and, even now, we do not have a complete description. Consequently, a wide variety of approaches to solving the problem have been suggested, all of which perform well in certain circumstances.

Four major factors make region labelling a difficult problem.

- The natural variability of the scene will cause an equal variability in the captured images. The characteristics of an object's image will depend strongly on the camera being used to capture data, the scene's illumination and other objects in the scene, plus some other, less important factors. This variability must either be captured in the descriptive model or must be removed somehow before the image is described.
- An image is a two-dimensional projection of a three-dimensional object. Any model must capture the object's 3-D properties and allow a match with a 2-D image. For example: as a car turns a corner it presents different aspects to the camera. The images are all valid examples of a car, but how do we represent these as a single model?
- Systems are often designed to recognise one object and make measurements of it. Sometimes we wish to recognise a small number of objects: car, lorry, van, bicycle or pedestrian. This can be achieved using a brute force approach. But what if the object database is too large to make this feasible? Then different matching strategies are needed, e.g. for Kanji (Chinese) character recognition, the dictionary includes several tens of thousands of characters.
- Occlusion – as the number of objects in the scene increases, so does the probability of one of them partially obscuring another. The object model ought to be robust against this problem and allow an object to be recognised given an image of part of it.

Figure 7.1 illustrates the significant stages of a region labelling system. The image is processed by the feature detector subsystem (described in chapter six). The other components of the system are:

- a database,
- a hypothesis generator and
- a hypothesis verifier.
- The database records the descriptions of the objects the system can recognise. This represents the world as far as each recognition system is concerned: the world contains only those objects known to the system. The database can contain any number of models; it is common to have systems in which one object is represented, a small number of models or many models. The strategy used to recognise these models is dependent on the number of models.
- Depending on the size of the database, a hypothesis generator might be required. For example, if we were recognising Kanji characters, the database might contain up to 44 000 entries. Clearly it is impractical to compare a candidate against each character in turn. One solution is to abstract from the database a set of possible matches using some simple technique, and

compare the candidate against each one in turn. This strategy yields acceptable results, provided the number of candidate matches is small. Conversely, if the database contained one model, a brute force approach would be suitable.

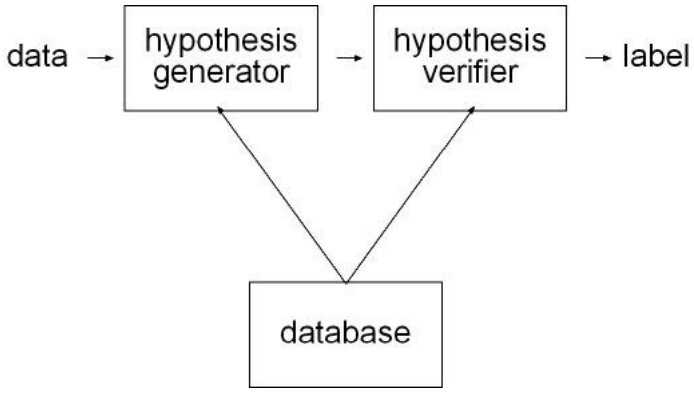


Figure 7.1: Components of an object labelling system

The hypothesis verifier will deliver the system's final answer: either the object's identity or a negative result: the object is not…. The verifier must compare the descriptions in the database (if they are few) or the descriptions suggested by the hypothesis generator against the object's description. The object will either match a model or not.

We shall examine a number of approaches to solving the recognition task. We shall start by investigating familiar template matching techniques, and look at their modification: flexible templates. We shall then examine statistical methods of pattern recognition and active shape models. Finally we shall examine syntactic methods of recognising patterns.

7.2 Object labelling

The simplest method of labelling an object is to compare it against a previously identified sample. If the object and sample resemble each other, then we have recognised it. This method is only appropriate when the object can appear in a small number of different views, for reasons that will become obvious. To overcome this difficulty, flexible templates have been defined.

7.2.1 Template matching

Template matching is the process of comparing regions of an image and a template. A measure of the similarity or dissimilarity between object and template is computed, we can therefore find the locations where the template matches a region of the image. Figure 7.2 is typical of the type of results that are obtained; it shows a scanned page of text, the template is one character abstracted from the text. The third image shows the similarity measure, and the profile of a cross section through that image. It is large where the template matches a region of the image, the magnitude of the result is proportional to the degree of similarity.

Rather than measure similarity between region, $f(i,j)$, and template, $t(i,j)$, it is simpler to first attempt to measure dissimilarity, D. How could a measure of dissimilarity be computed? We need a function that increases as the image and template become more dissimilar and is zero when they are the same. Three alternatives have been proposed:

- Compute the maximum absolute difference between template and the region of the image it overlies:

$$D(u,v) = \max_{(i,j)\in R} |f(u+i, v+j) - t(i,j)| \quad 7.1$$

- Compute the sum of absolute differences:

$$D(u,v) = \sum_{(i,j)\in R} |f(u+i, v+j) - t(i,j)| \quad 7.2$$

- Compute the sum of squared differences:

$$D(u,v) = \sum_{(i,j)\in R} (f(u+i, v+j) - t(i,j))^2 \quad 7.3$$

Whilst all options fulfil our requirements, the third is preferable as it is easiest to compute.

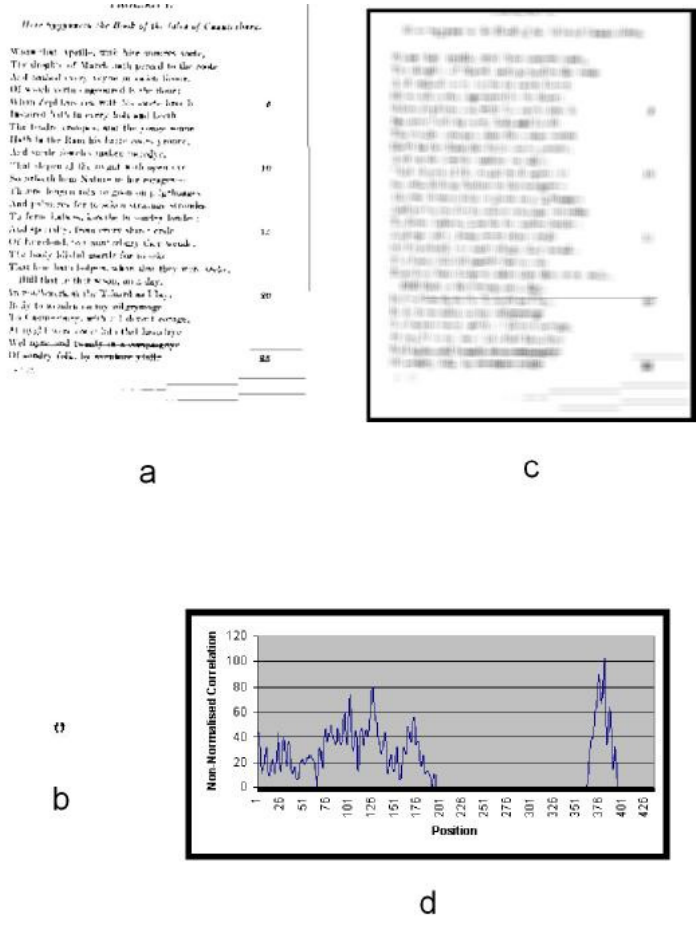


Figure 7.2: Template matching

If we expand the right hand side of equation 7.3, omitting the indices for brevity, we obtain:

$$D(u,v) = \sum_{(i,j)\in R} (f - t)^2 = \sum_{(i,j)\in R} f^2 + \sum_{(i,j)\in R} t^2 - 2\sum_{(i,j)\in R} ft \qquad 7.4$$

Now the sum of the template elements is constant and can therefore be safely ignored. The sum of image values in nearby regions will be similar and can also be ignored, at least for a first attempt at deriving a dissimilarity measurement. Therefore, if we ignore the first two terms of the expression and change the sign of the third term, we have a simple measure of similarity between a template and a region. It is the sum of the products of overlapping image and template values, which, for a symmetrical template, is the same as the convolution expression discussed in chapter three. We can therefore say that a reasonable measure of the similarity, *S*, between an image and a template is derived by computing the convolution (or cross correlation, if the template is symmetrical) between them:

$$S(u,v) = \sum_{i=-k}^{k}\sum_{j=-l}^{l} f(u+i, v+j)\,t(i,j) \qquad 7.5$$

This result is large in regions where the image and template are similar or the image values are high, and small in most other regions. It was the expression used to derive the result image of figure 7.2. It is apparent that there are also high values in the output image that cannot be attributed to this reason but are caused simply because the image values are large in that region. Normalising *S* using the sum of image values in the region can suppress these false positives. The expression that results is the cross-correlation coefficient, *C*:

$$C(u,v) = \frac{\sum_{i=-k}^{k}\sum_{j=-l}^{l} f(u+i, v+j)\,t(i,j)}{\sqrt{\sum_{i=-k}^{k}\sum_{j=-l}^{l} f^2(u+i, v+j)}} \qquad 7.6$$

Some authors also normalise using the sum of the template values, but since this factor is constant, we have omitted it. *C* will be large where the template and image are similar. Therefore, we can locate and recognise an object in the image by setting up the appropriate template, figure 7.3, and computing the normalised cross-correlation. The local maxima in this image will define regions of similarity. The implementer must decide a threshold value to differentiate true- from false-positives.

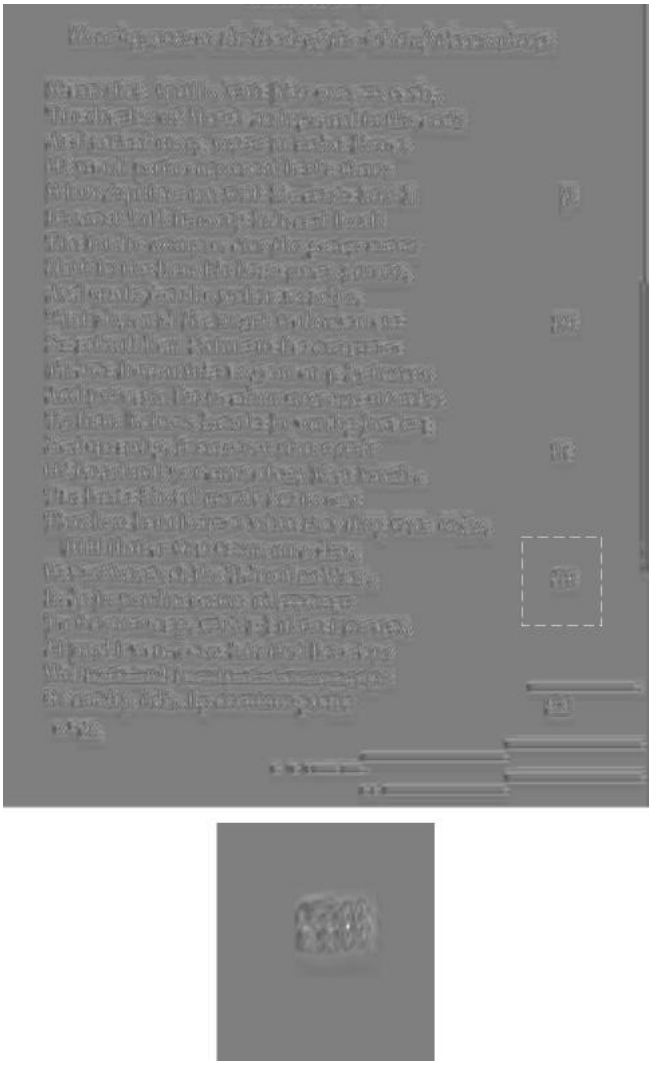


Figure 7.3: Normalised cross correlation. The lower image is an enlargement of the region indicated in the upper region.

Although cross-correlation, or template matching, is extremely effective when used correctly, we must be aware of its severe limitations.

A specific template is required for each object to be recognised. To recognise printed text using this method we would require 26 templates for lower case characters, another 26 for upper case, 10 for numerals and a small number for punctuation marks. If we change the font size, all the templates must be changed. If we rotate the text, another set of templates is required. The problem is exacerbated in real-world images, since there is infinite variability in the views of objects that we observe.

The matching fails catastrophically if the objects are partially occluded.

For these reasons, modifications to template matching algorithms are required, if they are to be deployed efficiently.

7.3 Statistical recognition

Many of the techniques discussed in chapter six generated numerical descriptions of a region. Statistical pattern recognition methods are appropriate for recognising these patterns. The data derived by these techniques are compared against the models using well-defined and mathematically rigorous statistical techniques that will yield probability values: the probability of the pattern being an instance of each of the models in the database.

The input to the recogniser will be a pattern vector, equation 7.8 (where the superscript T indicates matrix transposition). This is simply a vector formed of the measurements derived from the region. The

vector's elements must therefore be numerical values. This might be thought to be a limitation of the technique, but it is usually possible to create a numerical coding of non-numerical values.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad 7.8$$

$$\mathbf{x} = \begin{bmatrix} x_2 & x_2 & \cdots & x_n \end{bmatrix}^T$$

The vector specifies a point in pattern space. Pattern space is the volume defined by the features: it has as many dimensions as there are features and is limited by the total range of the features' values.

Provided the features have been chosen using the guidelines stated at the end of chapter six, the features derived from instances of the same class of pattern will form a tight cluster in the pattern space and each class' cluster will be clearly separated from all others. Figure 7.8 illustrates this point. It was derived during a study into the effects of illumination on apparent skin colour. Each cluster represents the skin colours derived from a sample population under different illumination conditions. Note how the clusters are clearly separated, but there is also a finite variation within the cluster. Note also that in this case the pattern space has only two dimensions, normally many more features would be used to describe a region and the pattern space would therefore have many more dimensions.

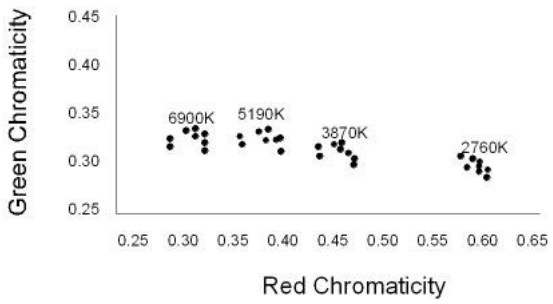


Figure 7.8: Sample pattern space. These are normalised skin chromaticities for a set of people of different ethnicities under different lighting conditions

The pattern recognition problem may be stated as the assignment of an unknown pattern **x** to one of a known set of *W* classes: ω_i.

7.3.1 Discriminant analysis

A discriminant function assesses the degree of membership of a sample to a class – how strongly does a sample belong to a class. In the case where we have *W* classes, we must define *W* discriminant functions, $d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots, d_W(\mathbf{x})$. The sample, **x**, will be assigned to the class having the largest value of its discriminant function:

$$d_i(x) > d_j(x) \quad j = 1,2,...,W \; ; \; j \neq i \qquad 7.9$$

A decision boundary that separates class ω_i from ω_j may be defined by the values of **x** for which the two discriminant functions are equal:

$$d_i(x) - d_j(x) = 0 \qquad 7.10$$

A decision function may therefore be defined the enables binary decisions to be made: does a sample belong to class *i* or class *j*?

$$d_{ij}(x) = d_i(x) - d_j(x) \qquad 7.11$$

In this formalism, a pattern may be assigned to a class depending on the value of d_{ij}, see figure 7.9.

$$if \begin{cases} d_{ij}(x) < 0 \; x \in \varpi_j \\ d_{ij}(x) = 0 \; x \, is \, unclassified \\ d_{ij}(x) > 0 \; x \in \varpi_i \end{cases} \qquad 7.12$$

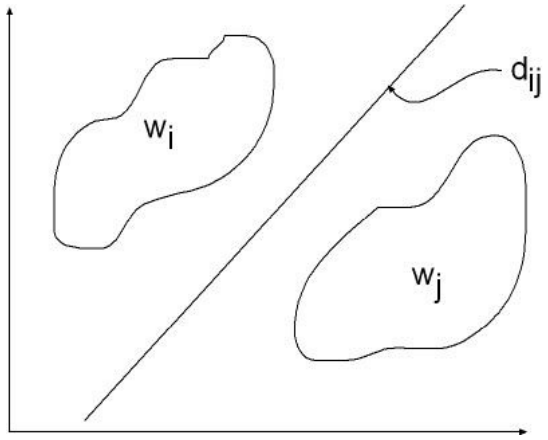


Figure 7.9: Linear discriminant analysis – two class case

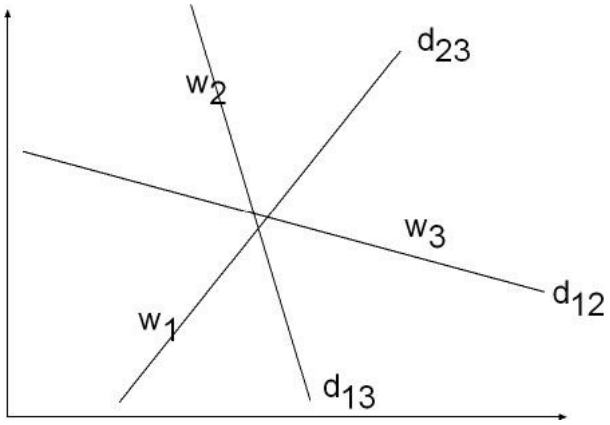


Figure 7.10: Linear discriminant analysis, multiple classes

This approach can be extended to multiple classes using a voting approach, as shown in figure 7.10. For each pair of classes we define a decision function. In the case of *W* classes, there will be *W*(*W*-1)/2 decision functions, which must be evaluated for an unlabelled sample. *W*-1 of the decisions will compare the correct classification of this sample against the other classes; the correct class will therefore receive *W*-1 votes. The other decision functions will involve two incorrect classifications and will therefore be distributed randomly. It can be shown that the incorrect classes will each receive *W*/2-1 votes, if the distribution is completely random.

How is the discriminant function defined? One simple possibility is to use the distance from the unknown sample to the centroid of the class (even though this gives a decision rule in which a sample is assigned to the class having the *smallest* discriminant function):

$$d_i(\mathbf{x}) = |\mathbf{x} - \mathbf{m}_i|$$

where 7.13

$$\mathbf{m}_i = \frac{1}{N_i} \sum_{x \in \varpi_i} \mathbf{x}_i$$

The decision function between two classes is then the perpendicular bisector of the line joining the centroids of the two classes.

7.3.2 Bayesian classification

Linear discriminant classification has a fundamental problem in that it takes no account of the distribution of values within a class. Consider the distributions of figure 7.11. The two classes represented in the diagram have unequal distributions. The minimum distance classifier takes no account of this; in fact the decision boundary that bisects the line joining the classes' centroids lies partly within one of the classes. Therefore, a minimum distance classifier would have a significant misclassification rate.

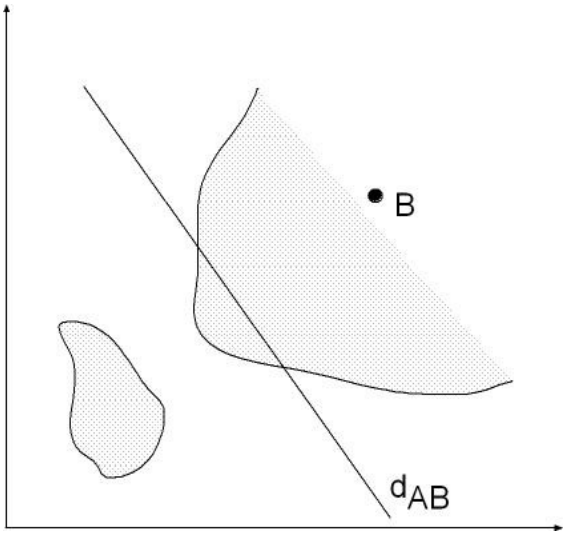


Figure 7.11: Two unequally distributed classes.

An improvement on the minimum distance classifier would be to take account of the spread of values of the distribution and compute some form of *Z* score (the distance normalised by the variance). The discriminant function would then be:

$$d_i(x) = (x - m_i)^T \Sigma_i^{-1} (x - m_i) \qquad 7.14$$

The Σ represents the covariance of the sample's distribution. If the pattern space were one dimensional, this expression would reduce to the ratio of the square of the difference between the measurement and the mean to the variance of the sample.

Provided the variance matrix, Σ, can be measured, this is a satisfactory discriminant function. If this quantity cannot be measured, then a Bayesian approach may be taken, which assumes nothing about the distributions of pattern values.

Generally, the spread of the values of one component of a pattern vector sampled from two classes might be as shown in figure 7.12, which shows a significant overlap between the values derived from the two classes. Bayes rule gives a rigorous method of assigning a sample to one class or the other with minimum error.

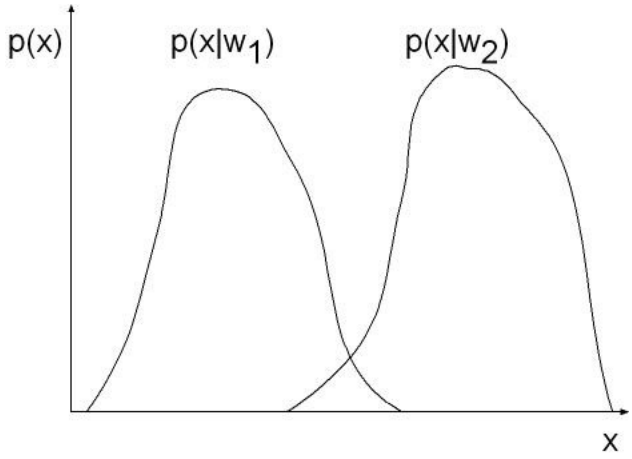


Figure 7.12: Probablity distribution of *x* from two classes. Not normalised

We define the probability of observing a member of class *j* as $P(\omega_j)$. Knowing nothing else about neither the distributions of classes nor the values of any measurements derived from a sample, we could minimise the classification error by assigning the sample to the class having the largest $P(\omega_j)$. This is not entirely satisfactory.

Suppose that we have previously measured the probabilities of observing pattern values for each class, as shown in figure 7.12. These conditional probabilities are denoted by $p(\mathbf{x}|\omega_j)$. This distribution enables us to read off the probability of a sample belonging to a class if the observed feature value is *x*. Bayes rule provides a means of combining both pieces of information: the conditional probability density, $p(\mathbf{x}|\omega_j)$, and the a priori probability $P(\omega_j)$ to give what is called the a posteriori probability:

$$p(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \qquad 7.15$$

where

$$p(x) = \sum_{j=1}^{W} p(x|\omega_j)P(\omega_j) \qquad 7.16$$

In the two-class case of figure 7.12, the a posteriori probability density functions are as shown in figure 7.13.

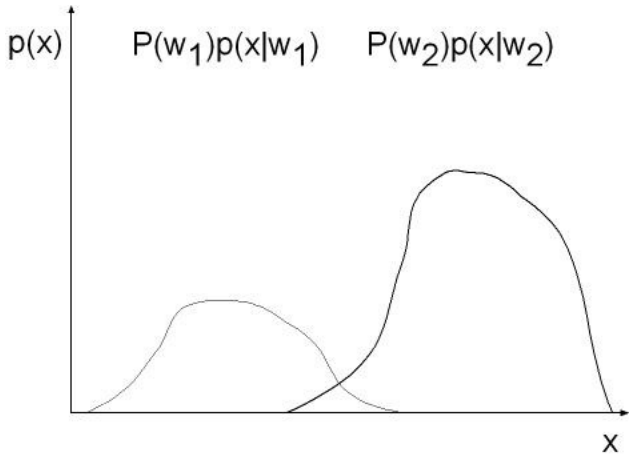


Figure 7.13: A posterior probability distribution

The sample should be assigned to the class having the largest a posteriori probability. It should also be noted that *p(x)* is common to all a posteriori measurement and could therefore be ignored. The discriminant function is therefore:

$$d_i(x) = p(x|\varpi_i)P(\varpi_i) \qquad 7.17$$

So far, nothing has been said about the form of the conditional probability density, or how we might estimate the a priori probabilities that are required in order to use Bayes rule. Determining the a priori probability is not difficult, it will often be dictated by the problem we are attempting to solve (for a system designed to identify items on a conveyor belt, the numbers of each type of item will be known in advance, therefore we know the a priori probabilities).

Determining the form of conditional probability densities is not a simple matter: they will generally be multiple-dimensioned and need not be smooth functions, few samples from each class may be available to build the distributions. Therefore, it is usual to assume an analytical form for this distribution; the Gaussian, or Normal, distribution is most often used. Since this may not be the correct distribution, the performance of the classifier will be degraded.

7.4 Syntactic recognition

Many object outlines are described structurally. The example of a chromosome was given in which the outline was represented by a sequence of elemental shapes. In a similar fashion, the outlines of characters have been represented using the appropriate shapes. Syntactic techniques are used to recognise such sequences of primitives.

A syntactic pattern recogniser requires three components:

- a set of pattern primitives: the elemental shapes used to describe the outline,
- a set of rules, a grammar, that define how the primitives are combined and
- a recogniser that processes the rules and generates a classification

The primitive elements may be considered to be words in the language generated by the grammar. The grammar, *G*, will be capable to generating every sentence in the language, *L(G)*, that is every example of a valid outline in these examples.

For recognition to occur, the rules of the grammar are applied in reverse to a sentence. If it can be shown that the sentence is consistent with the grammar, then the pattern has been recognised. If more than one class of outline is to be recognised, then multiple grammars must be specified.

The grammar is composed of a 4-tuple:

$$G = (N, \Sigma, P, S) \qquad 7.18$$

where

N is a set of nonterminal symbols, used as variables during recognition,

Σ is a set of terminal symbols, corresponding to the pattern primitives to be extracted from the image,

P is the set of rules defining the grammar, called productions and

S is a start symbol, which is effectively a class label.

Consider the grammar derived to generate and recognise chromosomes, figure 7.14, and the chromosome of figure 7.15, which is represented by the string of symbols indicated. Then by applying the rule of this grammar to replace the terminal symbols and nonterminals, we eventually apply one rule that replaces some nonterminals with the start symbol. The chromosome has thus been recognised.

Different types of chromosome could be recognised by including rules with other start symbols.

7.5 Evaluation

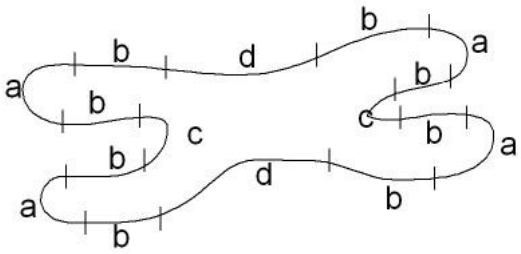
Having designed and tested a classifier, it must be evaluated. Two quantities must be specified:

- How accurately are the various classes recognised?
- Is there any pattern to the misclassification of samples?

The first of these quantities is evaluated using the classification rate: how frequently does the classifier correctly recognise a pattern. The second quantity is often described using the confusion matrix.

<submedian chromosome> → <arm pair> <arm pair>
<arm pair> → <side> <arm pair>
<arm pair> → <arm pair> <side>
<arm pair> → <arm> <right part>
<arm pair> → <left part> <arm>
<left part> → <arm> c
<right part> → c <arm>
<arm> → b <arm>
<arm> → <arm> b
<arm> → a
<side> → b <side>
<side> → <side> b
<side> → b
<side> → d

Figure 7.14: Chromosome generation and recognition grammar



d b a b c b a b d b a b c b a b
→ <side> b <arm> c <arm> <side> b <arm> c <arm>
→ <side> <arm> c <arm> <side> <arm> c <arm>
→ <side> <arm> <right part> <side> <arm> <right part>
→ <side> <arm pair> <side> <arm pair>
→ <arm pair> <arm pair>
→ <submedian chromosome>

Figure 7.15: Chromosome classification, top) a chromosome and shape primitives, bottom) the string representing the outline and the stages of its recognition

7.5.1 Classification rate

The classification rate, sometimes called the success rate, is simply the proportion of the patterns that

are assigned to their correct classes. In the ideal case all patterns would be correctly classified. In practice, due to variability of the data and due to errors in the measurements we are likely to observe, as indicated in the previous section, that there are overlaps in the distribution functions of any two classes in the pattern space. This is particularly true when trying to classify natural objects.

A classifier's success rate is estimated using test data whose identities are known. Many strategies have been proposed to estimate the success rate, a large proportion of them for cases where there is little test data. Two of the simplest techniques are probably the most obvious ones.

The simplest method of assessing a classifier is to divide our data into two sets. The first, the learning data, is used to train the classifier, that is to build the models in the database and evaluate the properties of the different classes. The second data set, the test data, is used for testing, the classifier will classify each pattern in this set and we will record the number of correct classifications. The classification or success rate is then simply the ratio of the number of correct classifications to the total number of patterns classified.

Two difficulties can arise when using this method. How can we be sure that the learning and test data are representative of the data as a whole? If there is a small amount of data, then the estimate of the success rate can be very uncertain: if we have ten patterns in the test set, then changing the classification of one could change the success rate by a significant amount.

These problems can be alleviated using the *leave one out* strategy. The classifier is trained using all the data but one sample, which is then classified. The outcome, success or failure, is recorded. This is repeated for all of the samples and the total success rate is obtained.

Both methods can only estimate the success rate. The true success rate can only be found by classifying all of the patterns, which is clearly impractical. Instead we must be aware that we have an estimate of the classification rate that is subject to some uncertainty. The question must then be asked, how big is the uncertainty in our estimate of the classification rate?

Suppose we wish to establish the proportion, p, of left handed students in a class. Suppose we interview a random sample of N students and n claim to be left-handed. We may estimate p to be:

$$\hat{p} = \frac{n}{N} \qquad 7.19$$

The true proportion can only be found if our sample is the whole class. The larger N is, the more accurate the estimated proportion will be. If we interviewed different selections of students, then the estimated proportion might differ in each case. In fact the estimates will be samples from a binomial distribution. For large values of N, more than about 25, the distribution is approximately Gaussian (Normal) and has a mean and standard deviation of:

$$\begin{aligned} \mu &= p \\ \sigma &= \sqrt{p(1-p)N} \end{aligned} \qquad 7.20$$

We must use our estimate of p in these expressions, as we do not actually know it.

Given that approximately 95% of the Gaussian distribution lies within two standard deviations of the mean, we can be reasonably certain that the true value of p is within two standard deviations of our estimate.

This analysis is only appropriate if the outcome of our classifier is accurate, presumably the students know whether they are left-handed or not. In practice, the classifier will be subject to some error. In this case, the standard deviation remains the same, but the mean of the distribution is now:

$$\mu = p(1-\varepsilon_1) + (1-p)\varepsilon_2 \qquad 7.21$$

Where ε_1 and ε_2 are the two classifier error rates, the errors caused when a left-handed person claims to be right-handed and vice versa.

7.5.2 Confusion matrix

The confusion matrix is the appropriate method of reporting the results of a multiclass classifier. It is simply an array reporting the classifications of patterns. Thus the matrix element $\boldsymbol{C}_{ij}$ reports the number of patterns of class i that were assigned to class j. The matrix is sometime presented as numbers of classifications and other times as classification probabilities.

7.6 Summary

In this chapter we have examined various methods of pattern recognition. We have taken the output of various region description algorithms from the previous chapter. These could have been numerical descriptions, symbolic descriptions coded numerically or symbolic descriptions. The numerical descriptions form a pattern vector. We hope that, given a good choice of region description, the pattern vectors associated with a class of regions will be similar, that is, they will be clustered in the same region of the pattern space. We also hope that different classes will occupy different regions of the pattern space, with no overlap. In practice, these properties are not often observed. We therefore need to be aware that when a pattern is classified, there will be some uncertainty in the classification. This uncertainty can be measured and ought to be specified as part of the classification result.

7.7 Bibliography

Introductory pattern recognition material is expanded in many texts, for example, Duda, Hart and Stork (2001), Duda and Hart (1973).

Statistical pattern recognition techniques are described by Duda et al. (2001) and by Fukunaga (1972). Syntactic, or structural, pattern recognition is described in Ledley (1964) and Gonzalez and Thomason (1978). Evaluation of classifiers is also analysed extensively, see for example Haralick and Shapiro (1992).

7.8 Exercises

1. The New Chinese Dictionary has over 43 500 entries. Each entry is a symbol representing a word or part of a word. Outline a recognition strategy that is able to recognise a character from the strokes used for drawing the characters.

2. Although the discussion of template matching indicated that most image data had so much variability that this technique could not work, it has been used as a means of recognising cars in collision avoidance systems. From your knowledge of cars, design a template that could be used to recognise the widest possible range of cars.

3. Describe the tasks that must be performed in creating a statistical pattern recognition system that is to identify different organs in an ultrasound image. The organs differ in the textural pattern they reveal.

4. Written characters can be recognised by the strokes that they are drawn with, or by the corners that are discovered when tracing the characters' outlines. Suggest a set of shape primitives and rules that could be used for recognising upper case Latin characters.

References

Ballard, D.H., Brown, C.M. (1982) *Computer Vision*, Prentice Hall, Englewood Cliffs, N.J.

Beucher, S, Meyer, F. (1992), The Morphological Approach to Segmentation: The Watershed Transformation, in *Mathematical Morphology in Image Processing*, E. Dougherty (ed) Marcel Decker, New York.

Binford, T.C. (1982), "Survey of Model-Based Image Analysis Systems", *Int J of Robotics Research*, 1, pp 18-64.

Bracewell, R.N. (2000), *The Fourier Transform and its Applications*, McGraw Hill, New York.

Brooks, R.A. (1981) "Symbolic Reasoning Among 3-D Models and 2-D Images", *Artificial Intelligence,* 17, pp 285-348.

Canny, J (1986), "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), pp 679-698.

Castleman K.R. (1996) *Digital Image Processing*, Prentice Hall, Upper Saddle River, N.J.

Cooley, J.W., Tukey J.W. (1965) "An Algorithm for the Machine Calculation of Complex Fourier Series", *Math. of Comput*. 19 pp 297-301.

Cootes, T.F., Cooper D.H., Taylor C.J., and Graham J., (1992) "Trainable method of parametric shape description", *Image and Vision Computing*, 10(5), pp 289-294.

Cootes T.F., Taylor C.J., Cooper D.H., and Graham J., (1995) "Active shape models - Their training and application", *Computer Vision and Image Understanding,* 61(1), pp 38-59.

Dickmans, E.D., Mysliwetz B.D. (1992), "Recursive 3-D road and relative ego-state recognition", IEEE Trans Pattern Anal. and Machine Intelligence, 14(2), pp 199-213.

Dougherty, E. (1992), *An Introduction to Morphological Image Processing*, SPIE Press, Bellingham, WA.

Duda R.O., Hatt, P.E. (1973) *Pattern Classification and Scene Analysis*, John Wiley and sons, New York.

Duda R.O., Hatt, P.E., Stork, D.G. (2001) *Pattern Classification*, John Wiley and sons, New York.

Erman, L.D. et al. (1980) "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, 12, pp 213-253.

Fishler M.A., Elschlager R.A. (1973) The representation and matching of pictorial structures, *IEEE Trans Computers* 22 pp67-77.

Forsyth D.A., Ponce J. (2003) *Computer Vision*, Prentice Hall, Upper Saddle River, N.J.

Freeman, A. (trans) (1878) J. Fourier, *The Analytical Theory of Heat*, Cambridge University Press, Cambridge.

Freeman, H. (1961) "On the Encoding of Arbitrary Geometric Configurations", *IEEE Trans. Elec. Computers* EC10 pp 260-268.

Freeman, H. (1974) "Computer Processing of Line Drawings", *Comput. Surveys* 6 pp 57-97.

Frendendall, G.L., Behrend, W.L. (1960), "Picture Quality – Procedures for Evaluating Subjective Effects of Interference", *Proc IRE*, 48, pp 1030-1034.

Frisby J.P. (1979) *Seeing: Illusion, Brain and Mind*, OUP, Oxford.

Fukunage, K. (1972) *Introduction to Statistical Pattern Recognition*, Academic Press, New York.

Garvey, T.D., Lowrance, J.D., Fischler, M.A., (1981) "An Intelligence Technique for Integrating Knowledge from Disparate Sources", *Proc Int Joint Conf on Artificial Intelligence*, Vancouver, pp 319-325.

Gee A., Cipolla, R. (1996), "Fast visual tracking by temporal consensus", *Image and Vision Computing*, 14(2), pp 105-114.

Gonzalez, R.C., Thomason, M.G. (1978), *Syntactic Pattern recognition: An Introduction*, Addison-Wesley, Reading, Mass.

Gonzalez R., Woods R. (1993) *Digital Image Processing*, Prentice Hall, Upper Saddle River, N.J.

Gregory R.L. (1997) *Eye and Brain*, OUP, Oxford.

Hanson, A.R., and Riseman, E.M. (1978) "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (eds) Academic Press, New York.

Haralick R., Shapiro L. (1992) *Computer and Robot Vision, volumes I and II*, Addison-Wesley, Reading, Mass.

Haralick, R.M., Shanmuhan, S.R., Dinstein, I. (1973), "Textural features for Image Classification", *IEEE Trans. Syst. Man Cyb*. SMC6(3) pp 610-621.

Hecht, E. (1998) *Optics*. Addison-Wesley, Reading, Mass.

Helmholtz, H. von. (1881) Popular Lectures on Scientific Subjects. Longmans.

Hotelling, H. (1933), "Analysis of a Complex of Statistical Variables into Principal Components", *J.*

Educ. Psychol. 24 pp 417-441, 498-520.
Horn B.K.P., Schunk B.G. (1981) "Determining Optical Flow". *Artificial Intelligence* 17 pp 185-203.
Hubel, D.H., (1988) *Eye, Brain and Vision*, Scientific Amer Library, W.H. Freeman, New York.
Huffman, D.A. (1952), "A Method for the construction of Minimum Redundancy Codes", *Proc IRE*, 40(10), pp 1098-1101.
Isard M., Blake A. (1998), "Condensation – Conditional Density Propagation for Visual Tracking", *Int J Computer Vision* 29(1), pp5-28.
Jain R., Kasturi R., Schunk B. (1995) *Machine Vision*, McGraw-Hill, New York.
Kalles, D., and Morris D.T. (1993). "An Efficient Image Skeletonisation Algorithm", *Image and Vision Computing* 11(9) pp 588-603.
Kalman, R.E. (1960) "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME--Journal of Basic Engineering, 82(D), pp 35-45.
Kass, M., Witkin, A., Terzopoulos, D. (1987), "Snakes: Active Contour Models", *Proc. First Int. Conf. Comput. Vision* London, pp 259-269.
Kittler J., Illingworth J. and Paler K. (1983) "The Magnitude Accuracy of the Template Edge Detector", *Pattern Recognition*, pp 607-613.
Ledley , R.S. (1964) "High Speed Automatic Analysis of Biomedical Pictures", *Science* 146 pp 216-223.
Levine, M.D. and Nazif, A.M., (1985) "Rule-Based Image Segmentation: A Dynamic Control Strategy Approach", *Computer Vision, Graphics and Image Processing*, 32, pp 104-126.
Lindberg, T. (1994) "Scale Space Theory: A Basic Tool for Analysing Structures at Different Scales", *J of Applied Statistics*, 21(2) pp 224-270.
Lowe D.G. (1987) "Three Dimensional Object Recognition From single Two-Dimensional Images", *Artificial Intelligence*, 31, pp355-395.
Marr D. (1982) *Vision*, Freeman, San Francisco.
Marr, D., Hildreth, E. (1980) "Theory of Edge Detection", *Proc R. Soc. Lond*, B207, pp 187-217.
Marr, D. and Poggio, T. (1979) "A Computational Theory of Human Stereo Vision", *Proc. R. Soc. Lond*. B204, pp 301-328.
Minsky, M., (1975) "A Framework for Representing Knowledge", in *The Psychology of Computer Vision*, P.H. Winston (ed) McGraw Hill, New York.
Moravec H.P., (1977) "Towards Automatic Visual Obstacle Avoidance". In *Proc. of the International Joint Conference on Artificial Intelligence*, page 584.
Myler, H.R., Weeks, A.R., (1993) *The Pocket Handbook of Image Processing Algorithms in C*, Prentice Hall, Upper Saddle River, N.J.
Newton I. (1931) Optiks, or a Treatise of the Reflections Refractions Inflections and Colour, Bell.
Parker, J.R. (1997) *Algorithms for Image Processing and Computer Vision*, John Wiley and sons, New York.
Press, W et al. (2002), *Numerical Recipes in C++: The Art of Scientific Computing*, Cambridge University Press, Cambridge.
Rosenfeldt, A., Hummel, R.A. and Zucker, S.W. (1976), "Scene Labelling by Relaxation Operations", *IEEE Trans Syst. Man and Cybern.* SMC-6, pp 420-433.
Rosin, P.L. (1997) "Techniques for Assessing Polygonal Approximations of Curves", *IEEE Trans. Pattern. Anal. Machine Intell.* 19(6) pp 659-666.
Schalkoff R. J. (1989) *Digital Image Processing and Computer Vision*, John Wiley and sons, New York.
Serra, J. (1982) *Image Analysis and Mathematical Morphology*, Academic Press, New York.
Serra, J.(ed.) (1988) *Image Analysis and Mathematical Morphology*, volume 2, Academic Press, New York.
Shafer, G. "Constructive Probability", *Synthesise*, 48, pp 1-60.
Shapiro, L.S., Stockman G.C. (2001) *Computer Vision*, Prentice Hall, Upper Saddle River, N.J.
Smith, S.M. (1996) "Flexible Filter Neighbourhood Designation", *ICPR13, Proc. 13th Int. Conf on Pattern Recognition*, Vienna, Aug 25-29. 1 pp 206-212.
Sonka, M., Hlavac, V., and Boyle, R. (1999), *Image Processing, Analysis and Machine Vision*, PWS Publishing, New York.
Spiegel, M.R. et al. (2000) *Schaum's Outline of Probability and Statistics*, McGraw Hill, New York.
Starner T., Pentland A. (1995), "Real-Time American Sign Language Recognition From Video Using Hidden Markov Models". In *Proc Int Symp Computer Vision*, Coral Gables, USA. IEEE Computer Society Press.
Stroud, J.M. (1956) "The Fine Structure of Psychological Time", In Quastler H (ed.) *Information Theory in Psychology*, Freepress, Glencoe, Ill.

Swain M.J., Ballard D.H., (1991) "Colour Indexing", *Int J. Computer Vision* 7(1) pp 11-32.
Swain M.J., Strickland M. (eds), (1991) "Promising Directions in Active Vision", University of Chicago, Technical Report CS 91-27.
Trucco E., Verri A., (1998) *Introductory techniques for 3-D Computer Vision,* Prentice-Hall, New Jersey.
Umbaugh, S.E. (1998) Computer Vision and Image Processing: a Practical Approach Using CVIPTools, Prentice Hall, Upper Saddle River, N.J.
Voss, K., Suesse, H. (1997) "Invariant Fitting of Planar Objects by Primitives", *IEEE Trans. Pattern. Anal. Machine Intell*. 19(1) pp 80-84.
Wren, C, Azerbayejani, A, Darrell, T., Pentland A. (1997) "pFinder: Real-time Tracking of the Human body", *IEEE Tran Pattern Anal. And Machine Intell.* 19 pp 780-785.
Young T. (1807) *Lectures on Natural Philosophy*, Vol II. Johnson, London.
Yuille A., Cohen D.S., Halliman P.W. (1986) Feature extraction from faces using deformable templates, *IEEE Proc CVPR*, San Diego, pp 104-9
Zahn, C.T., and Roskies, R.Z. (1072) "Fourier Descriptors for Plane Closed Curves", *IEEE Trans. Comput*. C21(3), pp 269-281.
Ziv, J. amnd Lempel, A. (1977) "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Info. Theory*, IT-23(3), pp 337-343.
Ziv, J. and Lempel, A. (1978) "Compression of Individual Sequences Via Variable Rate Coding", *IEEE Trans. Info. Theory*, IT-24(5), pp 530-536.