8

Image Processing and Analysis with Vision Systems

8.1 INTRODUCTION

There is a very large body of work associated with vision systems, image processing, and pattern recognition that addresses many different hardware- and software-related topics. This information has been accumulated since the 1950s, and with the added interest in the subject from different sectors of the industry and economy, it is growing rapidly. The enormous number of papers published every year indicates that there must be many useful techniques constantly appearing in the literature. At the same time, it also means that a lot of these techniques may be unsuitable for other applications. In this chapter, we will study and discuss some fundamental techniques for image processing and image analysis, with a few examples of routines developed for certain purposes. The chapter does not profess to be a complete survey of all possible vision routines, but only an introduction. It is recommended that the interested reader continue studying the subject through other references.

The next few sections present some fundamental definitions of terms and basic concepts that we will use throughout the chapter.

8.2 IMAGE PROCESSING VERSUS IMAGE ANALYSIS

Image processing relates to the preparation of an image for later analysis and use. Images captured by a camera or a similar technique (e.g., by a scanner) are not necessarily in a form that can be used by image analysis routines. Some may need improvement to reduce noise, others may need to be simplified, and still others may need to be enhanced, altered, segmented, filtered, etc. *Image processing* is the collection of routines and techniques that improve, simplify, enhance, or otherwise alter an image.

Image analysis is the collection of processes in which a captured image that is prepared by image processing is analyzed in order to extract information about the image and to identify objects or facts about the object or its environment.

8.3 TWO- AND THREE-DIMENSIONAL IMAGES

Although all real scenes are three dimensional, images can either be two or three dimensional. Two-dimensional images are used when the depth of the scene or its features need not be determined. As an example, consider defining the surrounding contour or the silhouette of an object. In that case, it will not be necessary to determine the depth of any point on the object. Another example is the use of a vision system for inspection of an integrated circuit board. Here, too, there is no need to know the depth relationship between different parts, and since all parts are fixed to a flat plane, no information about the surface is necessary. Thus, a two-dimensional image analysis and inspection will suffice.

Three-dimensional image processing deals with operations that require motion detection, depth measurement, remote sensing, relative positioning, and navigation. CAD/CAM-related operations also require three-dimensional image processing, as do many inspection and object recognition tasks. Other techniques, such as computed tomography (CT) scan, are also three dimensional. In computed tomography, either X-rays or ultrasonics pulses are used to get images of one slice of the object at a time, and later, all of the images are put together to create a three-dimensional image of the internal characteristics of the object.

All three-dimensional vision systems share the problem of coping with many-to-one mappings of scenes to images. To extract information from these scenes, image-processing techniques are combined with artificial intelligence techniques. When the system is working in environments with known characteristics (e.g., controlled lighting), it functions with high accuracy and speed. On the contrary, when the environment is unknown or noisy and uncontrolled (e.g., in underwater operations), the systems are not very accurate and require additional processing of the information. Thus, they operate at low speeds. In addition, a three-dimensional coordinate system has to be dealt with.

8.4 WHAT IS AN IMAGE?

An image is a representation of a real scene, either in black and white or in color, and either in print form or in a digital form. Printed images may have been reproduced either by multiple colors and gray scales (as in color print or halftone print) or by a single ink source. For example, in order to reproduce a photograph with real halftones, one has to use multiple gray inks, which, when combined, produce an image that is somewhat realistic. However, in most print applications, only one color of ink is available (such as black ink on white paper in a newspaper or copier). In that case, all gray levels must be produced by changing the ratio of black versus white areas (the size of the black dot). Imagine that a picture to be printed is divided

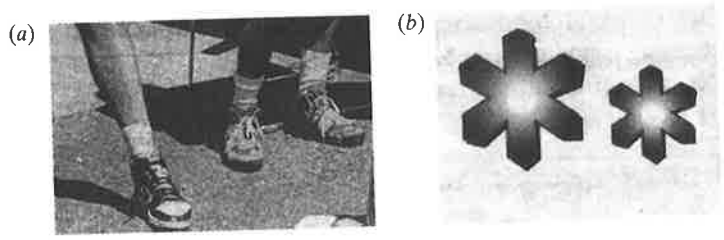


Figure 8.1 Examples of gray intensity creation in printed images. In print, only one color of ink is used, while the ratio of the black to the white area of the pixel is changed to create different gray levels.

into small sections. In each section, if the ink portion of the section is smaller compared to the white, blank area, the section will look lighter gray. (See examples in Figure 8.1.) If the black ink area is larger compared to the white area, it will look darker gray. By changing the size of the printed dot, many gray levels may be produced, and collectively, a gray-scale picture may be printed.

Unlike printed images, television and digital images are divided into small sections called *picture cells,* or *pixels* (in three-dimensional images, they are called *volume cells* or *voxels*), where the size of all pixels are the same, while the intensity of light in each pixel is varied to create the gray images. Since we deal with digital images, we will always refer to pixels of the same size with varying intensities.

8.5 ACQUISITION OF IMAGES

There are two types of vision cameras: analog and digital. Analog cameras are not very common any more, but are still around; they used to be standard at television stations. Digital cameras are much more common and are mostly similar to each other. A video camera is a digital camera with an added videotape recording section. Otherwise, the mechanism of image acquisition is the same as in other cameras that do not record an image. Whether the captured image is analog or digital, in vision systems the image is eventually digitized. In a digital form, all data are binary and are stored in a computer file or memory chip.

The following short discussion is about analog and digital cameras and how their images are captured. Although analog cameras are not common anymore, since the television sets available today are still mostly analog, understanding the way the camera works will help in understanding how the television set works. Thus, both analog and digital cameras are examined here.

8.5.1 Vidicon Camera

A vidicon camera is an analog camera that transforms an image into an analog electrical signal. The signal, a variable voltage (or current) versus time, can be stored, digitized, broadcast, or reconstructed into an image. Figure 8.2 shows a simple schematic of a vidicon camera. With the use of a lens, the scene is projected onto a screen made up of two layers: a transparent metallic film and a photoconductive mosaic that is sensitive to light. The mosaic reacts to the varying intensity of light by varying its resistance. As a result, as the image is projected onto it, the magnitude of the resistance at each location varies with the intensity of the light. An electron gun generates and sends a continuous cathode beam (a stream of electrons with a nega-

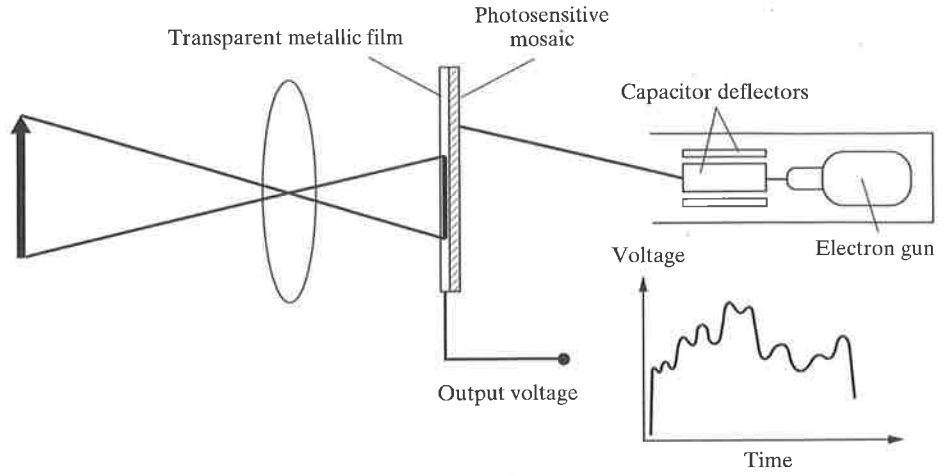


Figure 8.2 Schematic of a vidicon camera.

tive charge) through two pairs of capacitors (deflectors) that are perpendicular to each other. Depending on the charge on each pair of capacitors, the electron beam is deflected up or down, and left or right, and is projected onto the photoconductive mosaic. At each instant, as the beam of electrons hits the mosaic, the charge is conducted to the metallic film and can be measured at the output port. The voltage measured at the output is $V = IR$, where I is the current (of the beam of electrons), and R is the resistance of the mosaic at the point of interest.

Now suppose that we routinely change the charges in the two capacitors and thus deflect the beam both sideways and up and down, so as to cause it to scan the mosaic (a process called a *raster scan*). As the beam scans the image, at each instant the output is proportional to the resistance of the mosaic or proportional to the intensity of the light on the mosaic. By reading the output voltage continuously, an analog representation of the image can be obtained.

To create moving images in televisions, the image is scanned and reconstructed 30 times a second. Since human eyes possess a temporary hysteresis effect of about 1/10 second, images changing at 30 times a second are perceived as continuous and thus moving. The image is divided into two 240-line subimages, interlaced onto each other. Thus, a television image is composed of 480 image lines, changing 30 times a second. In order to return the beam to the top of the mosiac, another 45 lines are used, creating a total of 525 lines. In most other countries, 625 lines are the standard. Figure 8.3 depicts a raster scan in a vidicon camera.

If the signal is to be broadcast, it is usually frequency modulated (FM); that is, the frequency of the carrier signal is a function of the amplitude of the signal. The signal is broadcast and is received by a receiver, where it is demodulated back to the original signal, creating a variable voltage with respect to time. To re-create the image — for example, in a television set — this voltage must be converted back to an image. To do this, the voltage is fed into a cathode-ray tube (CRT) with an electron gun and similar deflecting capacitors, as in a vidicon camera. The intensity of the electron beam in the television is now proportional to the voltage of the signal, and is scanned similar to the way a camera does. In the television set, however, the beam

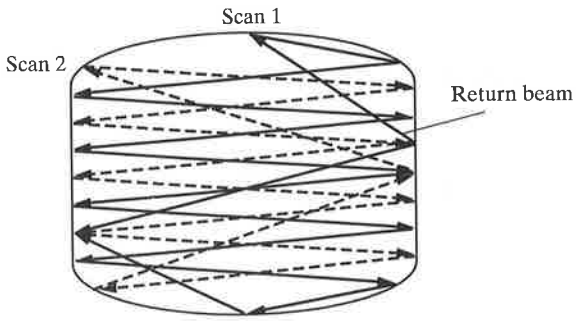


Figure 8.3 A raster scan depiction of a vidicon camera.

is projected onto a phosphorous-based material on the screen, which glows proportionally to the intensity of the beam, thus re-creating the image.

For color images, the projected image is decomposed into the three colors of red, green, and blue (RGB). The exact same process is repeated for the three images, and three simultaneous signals are produced and broadcast. In the television set, three electron guns regenerate three simultaneous images in RGB on the screen, except that the screen has three set of small dots (pixels) that react by glowing in RGB colors and are repeated over the entire screen. All color images in any system are divided into RGB images and are dealt with as three separate images.

If the signal is not to be broadcast, it either is recorded for later use, is digitized (as discussed later), or is fed into a monitor for direct viewing.

8.5.2 Digital Camera

A digital camera is based on solid-state technology. As with other cameras, a set of lenses is used to project the area of interest onto the image area of the camera. The main part of the camera is a solid-state silicon wafer image area that has hundreds of thousands of extremely small photosensitive areas called *photosites* printed on it. Each small area of the wafer is a pixel. As the image is projected onto the image area, at each pixel location of the wafer a charge is developed that is proportional to the intensity of light at that location. (Thus, a digital camera is also called a charge coupled device, or CCD camera, and a charge integrated device, or CID camera). The collection of charges, if read sequentially, would be a representation of the image pixels. (See Figure 8.4).

The wafer may have as many as 520,000 pixels in an area with dimensions of a fraction of an inch ($\frac{3}{16} \times \frac{1}{4}$). Obviously, it is impossible to have direct wire connections to all of these pixels to measure the charge in each one. To read such an enormous number of pixels, 30 times a second the charges are moved to optically isolated shift registers next to each photosite, are moved down to an output line, and then are read [1,2]. The result is that every thirtieth of a second the charges in all pixel locations are read sequentially and stored or recorded. The output is a discrete representation of the image — a voltage sampled in time — as shown in Figure 8.5(a). Figure 8.5(b) is the CCD element of a VHS camera.

Similar to CCD cameras for visible lights, long-wavelength infrared cameras yield a televisionlike image of the infrared emissions of a scene [3].

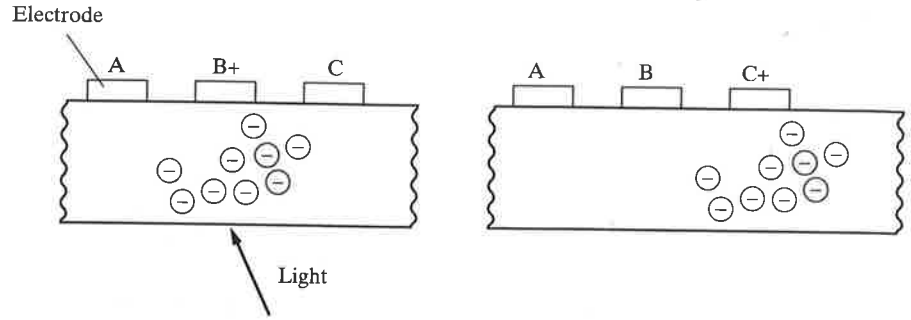
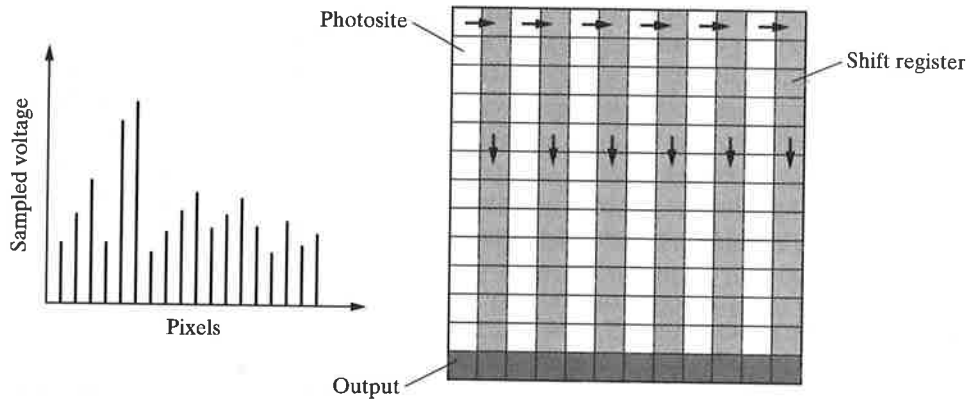


Figure 8.4 Image acquisition with a digital camera involves the development, at each pixel location, of a charge proportional to the light at the pixel. The image is then read by moving the charges to optically isolated shift registers and reading them at a known rate.



(*a*)

(*b*)

Figure 8.5 (a) Image data collection model. (b) The CCD element of a VHS camera.

8.6 DIGITAL IMAGES

The sampled voltages from the aforementioned process are first digitized through an analog-to-digital converter (ADC) and then either stored in the computer storage unit in an image format such as TIFF, JPG, Bitmap, etc., or displayed on a monitor. Since it is digitized, the stored information is a collection of 0's and 1's that represent the intensity of light at each pixel; a digitized image is nothing more than a computer file that contains the collection of these 0's and 1's, sequentially stored to represent the intensity of light at each pixel. The files can be accessed and read by a program, can be duplicated and manipulated, or can be rewritten in a different form. Vision routines generally access this information, perform some function on the data, and either display the result or store the manipulated result in a new file.

An image that has different gray levels at each pixel location is called a *gray image.* The gray values are digitized by a digitizer, yielding strings of 0's and 1's that are subsequently displayed or stored. A color image is obtained by superimposing three images of red, green, and blue hues, each with a varying intensity and each equivalent to a gray image (but in a colored state). Thus, when the image is digitized, it will similarly have strings of 0's and 1's for each hue. A binary image is an image such that each pixel is either fully light or fully dark — a 0 or a 1. To achieve a binary image, in most cases a gray image is converted by using the histogram of the image and a cut-off value called a *threshold.* A histogram determines the distribution of the different gray levels. One can pick a value that best determines a cutoff level with least distortion and use that value as a threshold to assign 0's (or "off") to all pixels whose gray levels are below the threshold value and to assign 1's (or "on") to all pixels whose gray values are above the threshold. Changing the threshold will change the binary image. The advantage of a binary image is that it requires far less memory and can be processed much faster than gray or colored images.

8.7 FREQUENCY DOMAIN VS. SPATIAL DOMAIN

Many processes that are used in image processing and analysis are based on the frequency domain or the spatial domain. In frequency-domain processing, the frequency spectrum of the image is used to alter, analyze, or process the image. In this case, the individual pixels and their contents are not used. Instead, a frequency representation of the whole image is used for the process. In spatial-domain processing, the process is applied to the individual pixels of the image. As a result, each pixel is affected directly by the process. However, the two techniques are equally important and powerful and are used for different purposes. Note that although spatial- and frequency-domain techniques are used differently, they are related. For example, suppose that a spatial filter is used to reduce noise in an image. As a result, noise level in the image will be reduced, but at the same time, the frequency spectrum of the image will also be affected, due to the reduction in noise.

The next several sections discuss some fundamental issues about frequency and spatial domains. The discussion, although general, will help us throughout the entire chapter.

8.8 FOURIER TRANSFORM AND FREQUENCY CONTENT OF A SIGNAL

As you may remember from your mathematics or other courses, any periodic signal may be decomposed into a number of sines and cosines of different amplitudes and frequencies as follows:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t. \tag{8.1}$$

If you add these sines and cosines together again, you will have reconstructed the original signal. Equation (8.1) is called a *Fourier series,* and the collection of different frequencies present in the equation is called the *frequency spectrum* or *frequency content* of the signal. Of course, although the signal is in the amplitude–time domain, the frequency spectrum is in the amplitude–frequency domain. To understand this concept better, let's look at an example.

Consider a signal in the form of a simple sine function like $f(t) = \sin(t)$. Since this signal consists of only one frequency with a constant amplitude, if we were to plot the signal in the frequency domain, it would be represented by a single line at the given frequency, as shown in Figure 8.6. Obviously, if we plot the function represented by the arrow in Figure 8.6(b) with the given frequency and amplitude, we will have reconstructed the same sine function. The plots in Figure 8.7 are similar and represent $f(t) = \Sigma_{n=1,3,\ldots 15}(1/n)\sin(nt)$. The frequencies are also plotted in the frequency–amplitude domain. Clearly, when the number of frequencies contained in $f(t)$ increases, the summation becomes closer to a square function.

Theoretically, to reconstruct a square wave from sine functions, an infinite number of sines must be added together. Since a square wave function represents a sharp change, this means that rapid changes (such as an impulse, a pulse, a square wave, or anything else similar to them or modeled by them) have a large number of frequencies. The sharper the change, the higher is the number of frequencies needed to reproduce it. Thus, any video (or other) signal that contains sharp changes (such as noise, high contrasts, or an impulse or step function) or that has

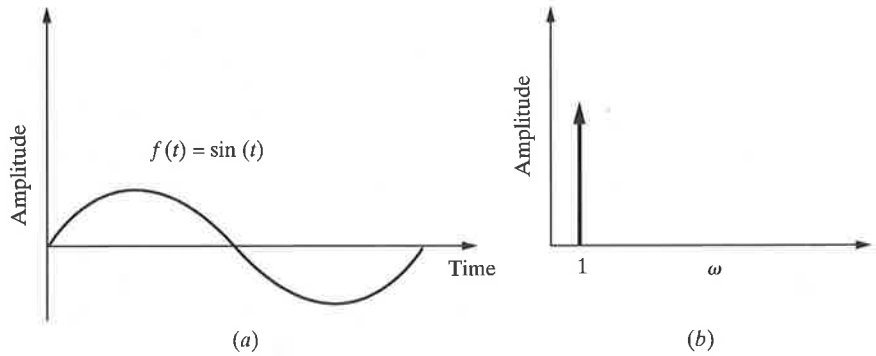


Figure 8.6 Time-domain and frequency-domain plots of a simple sine function.

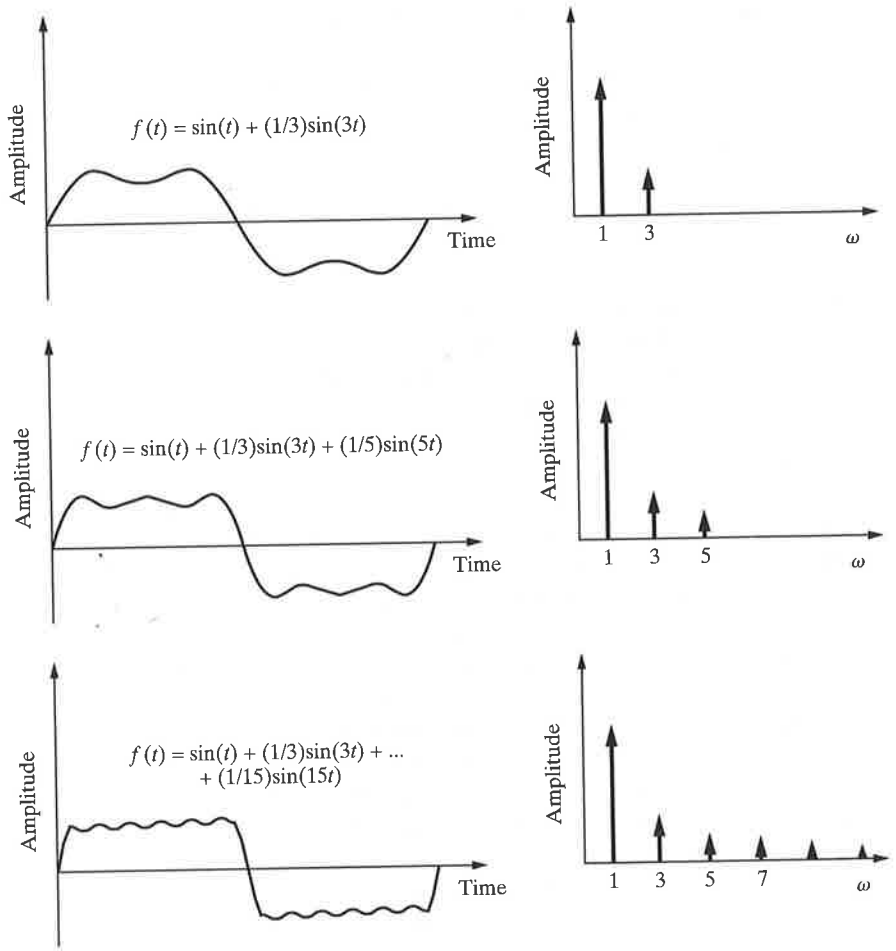


Figure 8.7 Sine functions in the time and frequency domain for a successive set of frequencies. As the number of frequencies increases, the resulting signal becomes closer to a square function.

detailed information (high-resolution signals with fast, varying changes) will have a larger number of frequencies in its frequency spectrum.

A similar analysis can be applied to nonrepeating signals as well. (The equation used is a *Fourier transform* or, sometimes, a *fast Fourier Transform,* or *FFT.*) Although we will not discuss the details of the Fourier transform in this book, suffice it to say that an approximate frequency spectrum of any signal can be found. Although, theoretically, there will be infinite frequencies in the spectrum, generally, some of the major frequencies within the spectrum will have larger amplitudes. These major frequencies, or *harmonics,* are used in identifying and labeling a signal, including recognizing voices, shapes, objects, etc.

8.9 FREQUENCY CONTENT OF AN IMAGE: NOISE, EDGES

Consider sequentially plotting the gray values of the pixels of an image (on the *y*-axis) against time or pixel location (on the *x*-axis) as the image is scanned. (See Section 8.5.) The result will be a discrete time plot of varying amplitudes showing the intensity of light at each pixel, as indicated in Figure 8.8. Let's say that we are on the ninth row and are looking at pixels 129–144. The intensity of pixel 136 is very different from the intensities of the pixels around it and may be considered to be noise. (Generally, noise is information that does not belong to the surrounding environment.) The intensities of pixels 134 and 141 are also different from the neighboring pixels and may indicate a transition between the object and the background; thus, these pixels can be construed as representing the edges of the object.

Although we are discussing a discrete (digitized) signal, it may be transformed into a large number of sines and cosines with different amplitudes and frequencies that, if added together, will reconstruct the signal. As discussed earlier, slowly changing signals (such as small changes between succeeding pixel gray values) will require few sines and cosines in order to be reconstructed, and thus have low frequency content. On the other hand, quickly varying signals (such as large differences between pixel gray levels) will require many more frequencies to be reconstructed and thus have high frequency content. Both noises and edges are instances in which one pixel value is very different from the neighboring ones. Thus, noises and edges create the larger frequencies of a typical frequency spectrum, whereas slowly varying gray level sets of pixels, representing the object, contribute to the lower frequencies of the spectrum.

However, if a high-frequency signal is passed through a low-pass filter — a filter that allows lower frequencies to go through without much attenuation in amplitude, but that severely attenuates the amplitudes of the higher frequencies in the

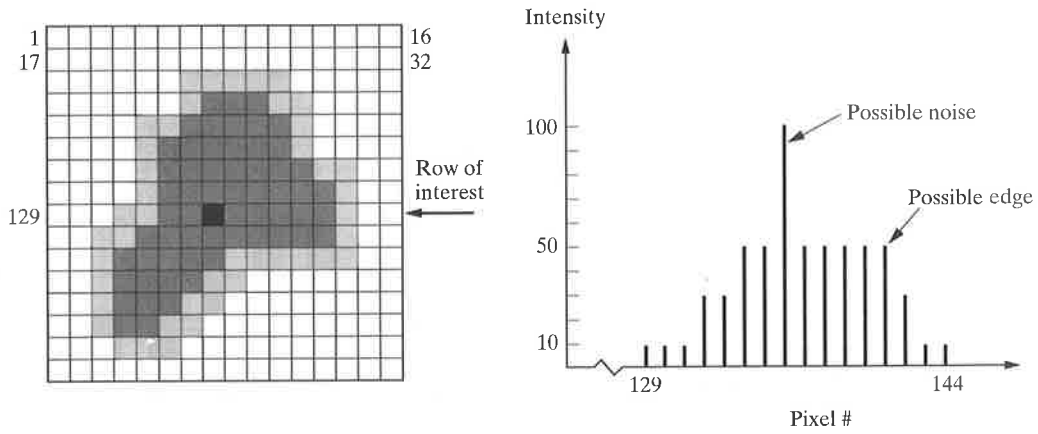
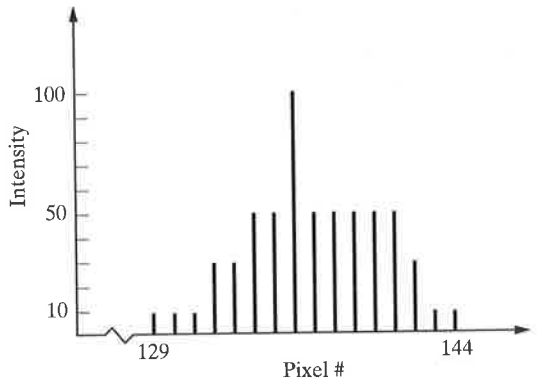


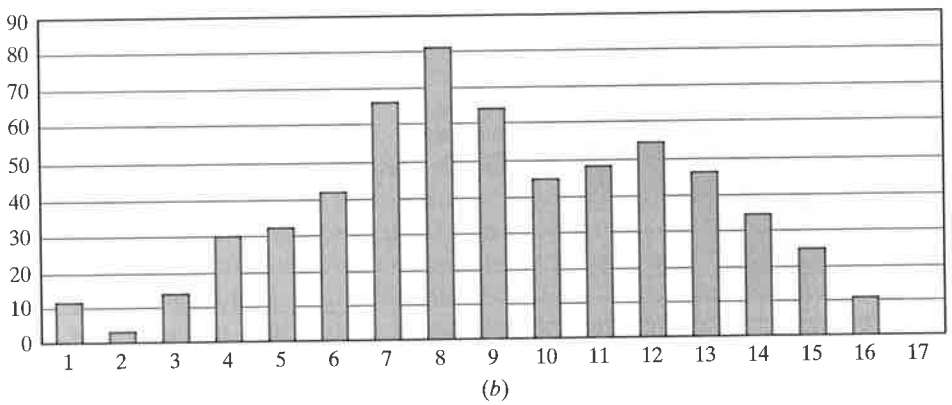
Figure 8.8 Noise and edge information in an intensity diagram of an image. The pixels with intensities that are much different from the intensities of neighboring pixels can be considered to be edges or noise.

signal — the filter will reduce the influence of all high frequencies, including the noises and edges. This means that, although a low-pass filter will reduce noises, it will also reduce the clarity of an image by attenuating the edges, thus softening the image throughout. A high-pass filter, on the other hand, will increase the apparent effect of higher frequencies by severely attenuating the low-frequency amplitudes. In such cases, noises and edges will be left alone, but slowly changing areas will disappear from the image.

To see how the Fourier transform can be applied in this case, let's look at the data of Figure 8.8 once again. The grayness level of the pixels of row 9 is repeated in Figure 8.9(a). A simple first-approximation Fourier transform of the gray values [4] was performed for the first four harmonic frequencies, and then the signal was reconstructed, as shown in Figure 8.9(b). Comparing the two graphs reveals that a digital, discrete signal can be reconstructed, even if its accuracy is dependent on the number of sines and cosines, as well as the method of integration, etc.



(a)



(b)

Figure 8.9 (a) Signal. (b) Discrete signal reconstructed from the Fourier transform of the signal in (a), using only four of the first frequencies in the spectrum.

8.10 SPATIAL-DOMAIN OPERATIONS: CONVOLUTION MASK

Spatial-domain processes access and operate on the information contained in an individual pixel. As a result, the image is directly affected by the operation. Most processes used in vision systems are in the spatial domain. One of the most popular and most common techniques in the spatial domain is the *convolution mask,* which can be adapted to many different tasks, from filtering to edge finding, to photography, and many more. We next examine the basic principles behind convolution masks, without referring to any particular type of mask. Later, we will adapt the convolution mask idea to different purposes.

Imagine that an image is composed of pixels, each with a particular gray level or color information, that collectively constitute the image. (Suppose that the gray level is not digitized into 0's and 1's, but the analog value is indicated.) As an example, let's say that the image in Figure 8.10 is part of a larger image with pixel values shown symbolically as $A, B, C, \ldots$. Let's also assume that there is a 3×3 mask that has the values indicated by $m_1, \ldots, m_9$ in its cells.

Applying the mask onto the image involves first superimposing the mask on the upper left corner of the image and taking the summation of the product of the value of each pixel and the corresponding mask value and then dividing the summation by a normalizing value. This yields

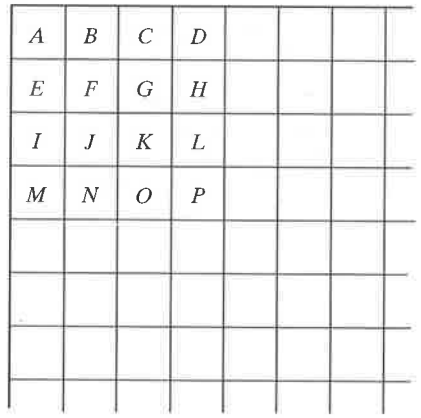
$$X = \frac{(A \times m_1 + B \times m_2 + C \times m_3 + E \times m_4 + F \times m_5 + G \times m_6 + I \times m_7 + J \times m_8 + K \times m_9)}{S}, \tag{8.2}$$

where

$$S = |m_1 + m_2 + m_3 + \ldots + m_9| \tag{8.3}$$

is the normalizing value. However, if the summation is zero, a "1" is used.

The result X of this operation will be substituted for the value of the pixel in the center of the block that was superimposed. In this case, X will replace the value



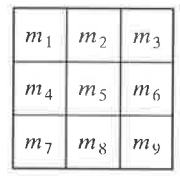


Figure 8.10 A convolution mask superimposed on an image can change the image pixel by pixel. Each step consists of superimposing the cells in the mask onto the corresponding pixels, multiplying the values in the mask's cells by the pixel values, adding the numbers, and normalizing the result, which is substituted for the pixel in the center of the area of interest. The mask is moved over pixel by pixel, and the operation is repeated until the image is completely processed.

of F ($X \rightarrow F_{\text{new}}$). Usually, the substitution takes place in a new file in order to not alter the original file. The mask is then moved one pixel to the right, and the same operation is repeated for a new value of X, which will replace G in a new file as follows:

$$X = G_{\text{new}} = \frac{(B \times m_1 + C \times m_2 + D \times m_3 + F \times m_4 + G \times m_5 + H \times m_6 + J \times m_7 + K \times m_8 + L \times m_9)}{S}.$$

Next, the mask is moved over one more pixel, and the operation is repeated until all the pixels in the row are changed. Then the operation continues with the subsequent rows until the image is completely affected. The resulting image will show characteristics that may be slightly or very severely affected by the operation, depending on the m values in the mask. The first and last rows and columns are not affected and are usually ignored. Some systems insert zeros for the first and last rows and columns.

For an image $I(R,C)$ with R rows and C columns of pixels, and for a mask $M(n,n)$ with n rows and columns in the mask, the value of the pixel $(I_{x,y})_{\text{new}}$ at the center of a block can be calculated by

$$(I_{x,y})_{\text{new}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} M_{i,j} \times I_{\left[\left(x-\left(\frac{n+1}{2}\right)+i\right),\left(y-\left(\frac{n+1}{2}\right)+j\right)\right]}}{S}, \tag{8.4}$$

where

$$S = \left|\sum_{i=1}^{n}\sum_{j=1}^{n} M_{i,j}\right| \quad \text{if the summation} \neq 0 \tag{8.5}$$

and

$$S = 1 \quad \text{if the summation} = 0.$$

Note that the normalizing or scaling factor S is arbitrary and is used to prevent saturation of the image. As a result, the user can always adjust this number to get the best image without saturation.

Example 8.1

Consider the pixels of an image, with values as shown in Figure 8.11, as well as a convolution mask with the given values. Calculate the new values of the given pixels.

Solution We will substitute zeros for the first and last columns and rows, because they are not affected by convolution. For the remaining pixels, we will superimpose the mask on the remaining cells of the image and will use Equations (8.2) and (8.3) to calculate new pixel values, as shown in Figure 8.12(a), with the result as indicated in

5	6	2	8
3	3	5	6
4	3	2	6
8	6	5	9

0	0	1
1	1	1
1	0	0

Figure 8.11 An example of a convolution mask.

2,2

5x0	6x0	2x1	8
3x1	3x1	5x1	6
4x1	3x0	2x0	6
8	6	5	9

0	0	1
1	1	1
1	0	0

2,3

5	6x0	2x0	8x1
3	3x1	5x1	6x1
4	3x1	2x0	6x0
8	6	5	9

0	0	1
1	1	1
1	0	0

3,2

5	6	2	8
3x0	3x0	5x1	6
4x1	3x1	2x1	6
8x1	6x0	5x0	9

0	0	1
1	1	1
1	0	0

3,3

5	6	2	8
3	3x0	5x0	6x1
4	3x1	2x1	6x1
8	6x1	5x0	9x0

0	0	1
1	1	1
1	0	0

(*b*)

0	0	0	0
0	3.4	5	0
0	4.4	4.6	0
0	0	0	0

(*b*)

Figure 8.12 (a) Superimposing the mask onto the cells of the image. (b) The result of the operation.

Figure 8.12(b). Superimposing the mask on the image for each remaining element, we have the following equations:

$$2,2: \quad 5(0)+6(0)+2(1)+3(1)+3(1)+5(1)+4(1)+3(0)+2(0)/5 = 3.4;$$

$$2,3: \quad 6(0)+2(0)+8(1)+3(1)+5(1)+6(1)+3(1)+2(0)+6(0)/5 = 5;$$

$$3,2: \quad 3(0)+3(0)+5(1)+4(1)+3(1)+2(1)+8(1)+6(0)+5(0)/5 = 4.4;$$

$$3,3: \quad 3(0)+5(0)+6(1)+3(1)+2(1)+6(1)+6(1)+5(0)+9(0)/5 = 4.6.$$

We next consider common routines and techniques that are used in image processing and image analysis.

8.11 SAMPLING AND QUANTIZATION

To be useful in image processing, the image must be digitized both spatially as well as in amplitude. Spatial digitization is the process that was mentioned in Section 8.5.2, wherein the intensities of light at each pixel location are read. The more pixels that are present and individually read, the better is the resolution of the camera and the image. This technique is called *sampling,* as the light intensities are sampled at equally spaced intervals. A larger sampling rate will create a larger number of pixel data and thus better resolution.

Figure 8.13 shows the same image sampled at (a) 432 × 576, (b) 108 × 144, (c) 54 × 72, and (d) 27 × 36 pixels. As the sampling rate decreases, the clarity of the image is lost. The voltage or charge read at each pixel value is an analog value and must also be digitized. Digitization of the light intensity values at each pixel location is called *quantization.* Depending on the number of bits used, the resolution of the image will change. The total number of gray level possibilities is 2^n, where n is the number of bits. For a 1-bit analog-to-digital converter (ADC), there are only two possibilities: "off" or "on," or, alternatively, 0 or 1 (called a *binary image*). For quantization with an 8-bit ADC, the maximum number of gray levels will be 256. Thus, the image will have 256 different gray levels in it. Quantization and sampling resolutions are completely independent of each other. For example, a high-resolution image may be converted into a binary image, and thus the quantization is only into two digits (0 and 1, or black and white, or "off" and "on"). Still, the same image may be quantized into 8 bits, which can yield a spectrum of 256 different shades of gray.

(a) (b) (c) (d)

Figure 8.13 Effect of different sampling rates on an image at (a) 432 × 576, (b) 108 × 144, (c) 54 × 72, and (d) 27 × 36 pixels. As the resolution decreases, the clarity of the image diminishes accordingly.

(a) (b) (c) (d)

Figure 8.14 An image quantized at 2, 4, 8, and 44 gray levels. As the quantization resolution increases, the image becomes smoother.

Figure 8.14 shows the same image quantized at (a) 2 levels, (b) 4 levels, (c) 8 levels, and (d) the original 44 levels.

The sampled light at a pixel, when quantized, will yield a string of 0's and 1's representing the light at that pixel location. The total memory required to store an image is the product of the memory needed for the total number of samples and the memory needed for each digitized sample. The larger the image size, the resolution of the image, or the number of gray levels, the larger is the memory requirement.

Example 8.2

Consider an image with 256 by 256 pixels. The total number of pixels in the image will be 256 × 256 = 65,536. If the image is binary, it will require 1 bit to record each pixel as 0 or 1. Thus, the total memory needed to record the image will be 65,536 bits, or, with 8 bits to a byte, 8,192 bytes. If each pixel were to be digitized at the rate of 8 bits for 256 shades of gray, it would require 65,536 × 8 = 524,288 bits, or 65,536 bytes. If the image were in color, it would require 65,536 bytes for each of the three colors of red, green, and blue. For a color video clip changing at the rate of 30 images per second, the memory requirement will be 65,536 × 3 × 30 = 5,898,240 bytes per second. Of course, this is only the memory requirement for recording the image pixels and does not include index information and other bookkeeping requirements.

8.12 SAMPLING THEOREM

Can you tell what the image in Figure 8.15 represents? Of course, since it is a very low resolution 16 × 16 image, it is difficult to guess what the object is. This simple illustration signifies the relationship between the sampling rate and the information obtained from it. To understand the relationship, we will discuss some fundamental issues connected with sampling.

Consider a simple sinusoidal signal with frequency f, as shown in Figure 8.16(a). Suppose that the signal is sampled at the rate of f_s. This means that the sampling circuit will read the amplitude of the signal with a frequency of f_s. The arrows in Figure 8.16(b) show the corresponding sampled amplitudes.

Now suppose that we want to use the sampled data to reconstruct the signal. (Doing this would be similar to sampling a sound source such as a CD player and then trying to reconstruct the sound signal from the sampled data through a speaker.) One possibility would be that, by chance, the same signal might be reconstructed. However, as you see in Figure 8.17, it is quite possible that, from the same data,

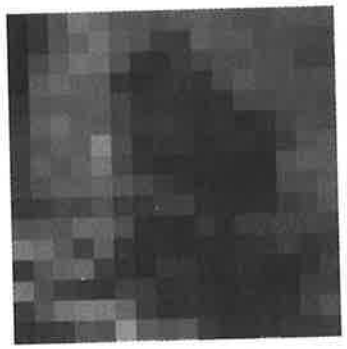
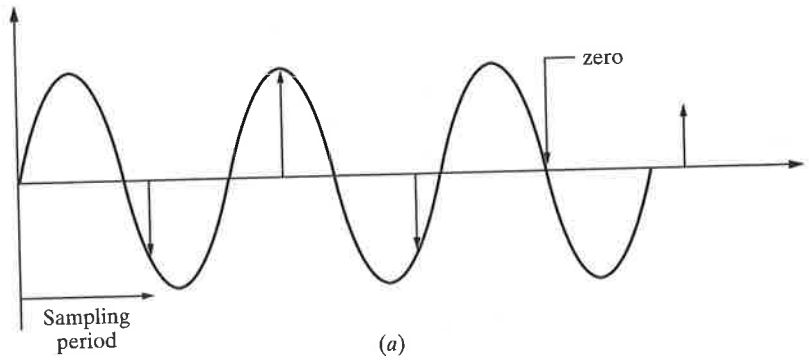


Figure 8.15 A low-resolution (16 × 16) image.



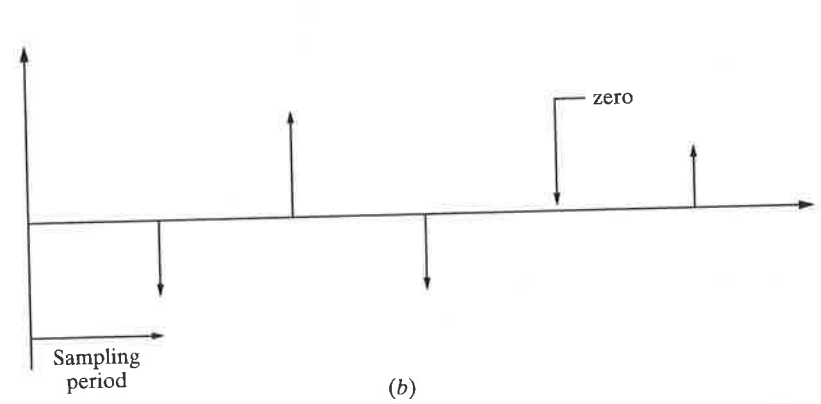


Figure 8.16 (a) Sinusoidal signal with a frequency of f. (b) Amplitudes sampled at the rate of f_s.

another signal may be reconstructed which is completely different from the original signal. Both signals are valid, and in fact, many other signals can be valid as well and might be reconstructed from the sampled data. This loss of information is called *aliasing* of the sampled data, and it can be a serious problem.

In order to prevent aliasing, according to what is referred to as the *sampling theorem,* the sampling frequency must be at least twice as large as the largest frequency present in the signal. In that case, one can reconstruct the original signal without aliasing. The highest frequency present in the signal can be determined from the frequency spectrum of the signal. Using the Fourier transform, one finds that the frequency spectrum of a signal will contain many frequencies. However, as we have seen, the higher frequencies may have smaller amplitudes. One can always pick a maximum frequency that may be of interest, while assuming that the frequencies with very low amplitudes beyond that point can be ignored without much effect in the system's total representation. In that case, the sampling rate of the signal must be at least twice as large as this maximum frequency. In practice, the sampling rate is generally chosen to be even larger, to further ensure that aliasing of the signal will not occur. Frequencies four to five times as large as the maximum frequency are common. As an example, consider a CD player. Theoretically, human ears can hear frequencies of up to about 20,000 Hz. If the CD player is to be able to reconstruct the digitized sampled music, the sampling rate of the laser sensor must be at least twice as large, namely, 40,000 Hz. In practice, CD players sample at the rate of about 44,100 Hz; at lower sampling rates, the sound may become distorted.

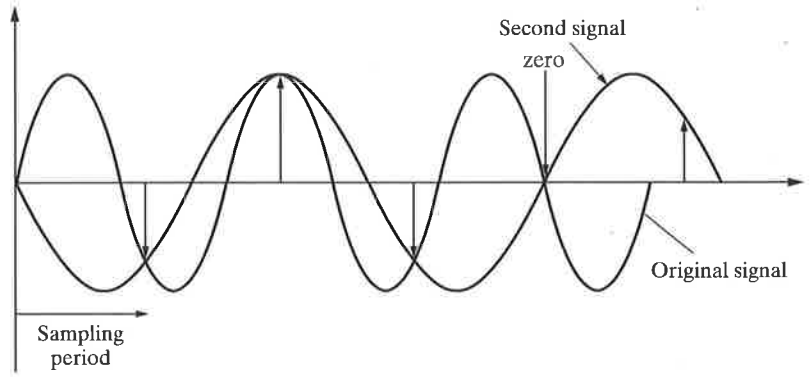


Figure 8.17 Reconstruction of signals from sampled data. More than one signal may be reconstructed from the same sampled data.

In the example of Figure 8.18, the sampling rate is lower than the higher frequencies of the signal. Although the lower frequencies of the signal are reconstructed, the signal will not have the higher frequencies of the original signal. The same may happen to any signal, including audio and video signals.

For images, too, if the sampling rate (which translates into the resolution of the image) is low, the sampled data may not have all the necessary detail, the information in the image is lost, and the image cannot be reconstructed to match the original. The image in Figure 8.15 is sampled at a very low rate, and the information in it is lost. This is why you cannot tell what the image is. However, if the sampling rate is increased, there will be a time when there will be enough information to be able to recognize the image. The still higher resolutions or sampling rates will transfer more information, and thus, increasingly more detail can be recognized. Figure 8.19 is the same image as in Figure 8.15, but at 2, 4, and 16 times higher resolutions. Now suppose that you need to recognize the difference between a bolt and a

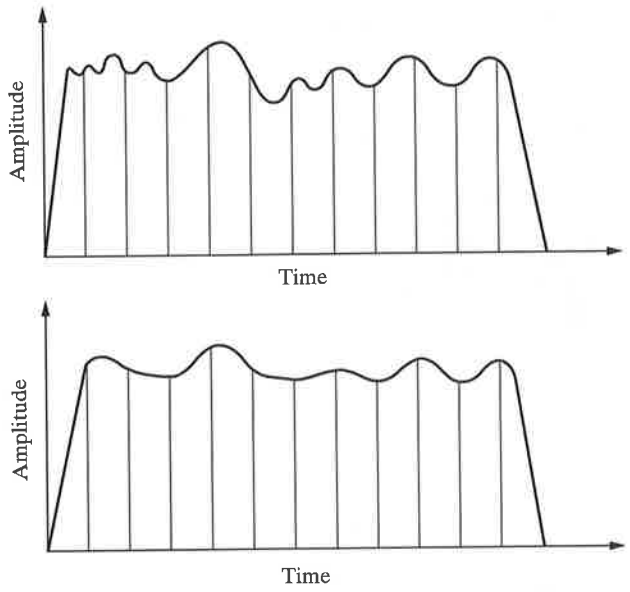


Figure 8.18 The original signal in (a) is sampled at a sampling rate that is lower than the higher frequencies of the signal. The reconstructed signal in (b) will not have the higher frequencies of the original signal.

(*a*) (*b*) (*c*)

Figure 8.19 The image of Figure 8.15, presented at higher resolutions of (a) 32 × 32, (b) 64 × 64, and (c) 256 × 256.

nut in a vision system in order to direct a robot to pick up the parts. Because the information representing a bolt is very different from that representing a nut, low-resolution images will still enable you to determine what the part is. However, in order to recognize the license plate number of a car while moving in traffic, one would need to have a high-resolution image to extract enough information about the details, such as the numbers on the license plate.

8.13 IMAGE-PROCESSING TECHNIQUES

As was mentioned earlier, image-processing techniques are used to enhance, improve, or otherwise alter an image and to prepare it for image analysis. Usually, during image processing information is not extracted from the image. The intention is to remove faults, trivial information, or information that may be important, but not useful, and to improve the image. As an example, suppose that an image was obtained while the object was moving, and as a result, the image is not clear. It would be desirable to see if the blurring in the image could be reduced or removed before the information about the object (such as its nature, shape, location, orientation, etc.) can be determined. Again, consider an image that is corrupted by direct lighting that is reflected back, or an image that is noisy because of low light. In all these cases, it is desirable to improve the image and prepare it before image analysis routines are used. Similarly, consider an image of a section of a city that is fully detailed, with streets, cars, shadows, etc. It may actually be more difficult to extract information from this image than if all unnecessary detail, except for edges, were removed.

Image processing is divided into many subprocesses, including histogram analysis, thresholding, masking, edge detection, segmentation, region growing, and modeling, among others. In the next few sections, we will study some of these processes and their application.

8.14 HISTOGRAM OF IMAGES

A *histogram* is a representation of the total number of pixels of an image at each gray level. Histogram information is used in a number of different processes, including thresholding. For example, histogram information can help in determining a cutoff point when an image is to be transformed into binary values. It can also be used to decide whether there are any prevalent gray levels in an image. For instance, suppose a systematic source of noise in an image causes many pixels to have one "noisy" gray level. Then a histogram can be used to determine what the noisy gray level is in order to attempt to remove or neutralize the noise.

Now suppose that an image has all its pixel gray levels clustered between two relatively close values, as in Figure 8.20(a). In this image, all pixel gray values are between 120 and 180 gray levels, at four-unit intervals. (The image is quantized at 16 distinct levels between 0 and 256.) Figure 8.20(c) is the histogram of the image, and clearly, all pixel gray levels are between 120 and 180, a relatively low range. As a result, the image is not very clear and details are not visible. Now suppose that we equalize the histogram such that the same 16 gray levels present in the image are spread out between 0 and 255 gray levels, at intervals of 17 units, instead of the present 120–180 gray levels at intervals of 4 units. Then, due to the equalization, the image is vastly improved, as shown in Figure 8.20(b), with its corresponding histogram in Figure 8.20(d). Notice that the number of pixels at each gray level is exactly the same in both cases, but that only the gray levels are spread out. The grayness values are given in Table 8.1.

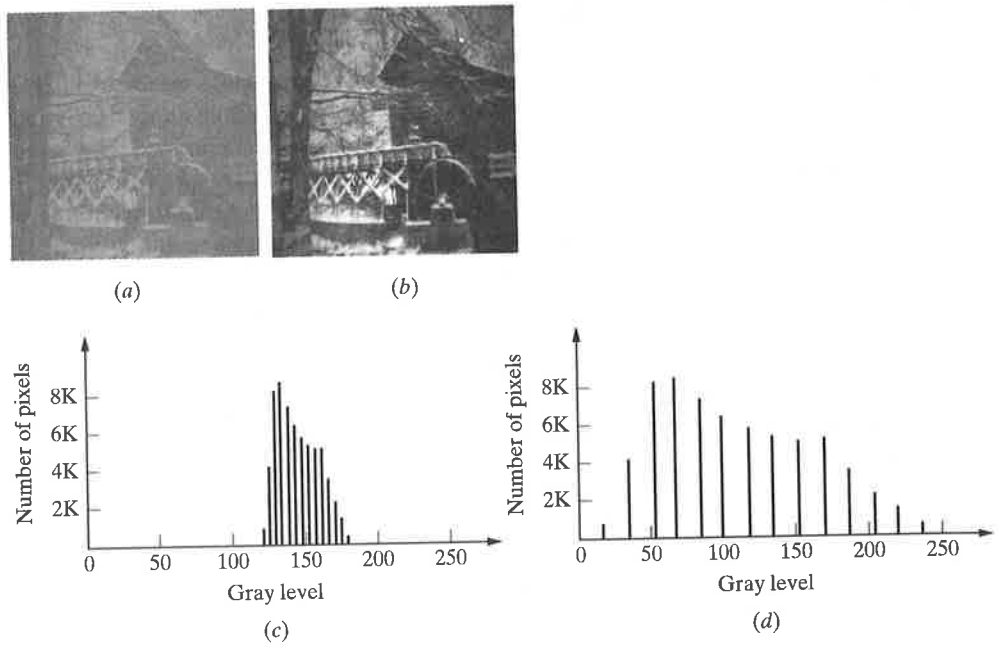


Figure 8.20 Effect of histogram equalization in improving an image.

TABLE 8.1 THE ACTUAL GRAYNESS VALUES AND NUMBER OF PIXELS FOR THE IMAGES IN FIGURES 8.20 (A) AND (B)

Levels	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
No. of Pixels	0	750	5,223	8,147	8,584	7,769	6,419	5,839	5,392	5,179	5,185	3,451	2,078	1,692	341	0
For (a)	0	17	34	51	68	85	102	119	136	153	170	187	204	221	238	256
For (b)	120	124	128	132	136	140	144	148	152	156	160	164	168	172	176	180

8.15 THRESHOLDING

Thresholding is the process of dividing an image into different portions (or levels) by picking a certain grayness level as a threshold, comparing each pixel value with the threshold, and then assigning the pixel to the different portions or levels, depending on whether the pixel's grayness level is below the threshold (off or zero, or not belonging) or above the threshold (on or 1, or belonging). Thresholding can be performed either at a single level or at multiple levels, in which the image is processed by dividing it into "layers," each with a selected threshold. To aid in choosing an appropriate threshold, many different techniques have been suggested, ranging from simple routines for binary images to sophisticated techniques for complicated images. Early routines used for a binary image had the object lighted and the background completely dark. This condition can be achieved in controlled lighting in industrial situations, but may not be possible in other environments. In binary images, the pixels are either on or off, and thus, choosing a threshold is simple and straightforward. In certain other situations, the image will have multiple gray levels, and its histogram will exhibit a bimodal distribution. In this case, the valley is chosen as the threshold value. More advanced techniques use statistical information and distribution characteristics of the image pixels to develop a thresholding value. As the thresholding value changes, so does the image. Figure 8.21 shows an original image with 256 gray levels and the result of thresholding at grayness levels of 100 and 150.

Figure 8.21 Thresholding an image with 256 gray levels at values of (b) 100 and (c) 150.

Thresholding is used in many operations, such as transforming an image into binary values, filtering operations, masking, and edge detection.

8.16 CONNECTIVITY

Sometimes we need to decide whether neighboring pixels are somehow "connected" or related to each other. Connectivity establishes whether they have the same properties, such as being of the same region, coming from the same object, having a similar texture, etc. To establish connectivity of neighboring pixels, we first have to decide upon a connectivity path. For example, we need to decide whether only pixels that are on the same column and row are connected or whether diagonally situated pixels are also accepted as being connected.

There are three fundamental connectivity paths for two-dimensional image processing and analysis: +4- or ×4-connectivity, H6 or V6 connectivity, and 8-connectivity. In three dimensions, connectivity between voxels (volume cells) can range from 6 to 26. The following terms are defined with respect to Figure 8.22.

+4-connectivity applies when a pixel p's relationship is analyzed only with respect to the four pixels immediately above, below, to the left, and to the right of p (namely, $b,g,d,$ and e).

a	b	c
d	p	e
f	g	h

Figure 8.22 Neighborhood connectivity of pixels.

×4-connectivity applies when a pixel p's relationship is analyzed only with respect to the four pixels immediately across from it diagonally on 4 sides (a, c, f, and h).

For a pixel $p(x,y)$ the relevant pixels are as follows:

- for +4-connectivity: $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, $(x, y-1)$; (8.6)
- for ×4-connectivity: $(x+1, y+1)$, $(x+1, y-1)$, $(x-1, y+1)$, $(x-1, y-1)$. (8.7)

H6-connectivity applies when a pixel p's relationship is analyzed only with respect to the six neighboring pixels on the two rows immediately above and below p (a, b, c, f, g, h).

V6-connectivity applies when a pixel p's relationship is analyzed only with respect to the six neighboring pixels on the two columns immediately to the right and to the left of p (a, d, f, c, e, h).

For a pixel $p(x,y)$, the relevant pixels are as follows:

- for H6-connectivity: $(x-1, y+1)$, $(x, y+1)$, $(x+1, y+1)$, $(x-1, y-1)$, $(x, y-1)$, $(x+1, y-1)$; (8.8)
- for V6-connectivity: $(x-1, y+1)$, $(x-1, y)$, $(x-1, y-1)$, $(x+1, y+1)$, $(x+1, y)$, $(x+1, y-1)$. (8.9)

8-connectivity applies when a pixel p's relationship is analyzed with respect to all eight pixels surrounding it (a, b, c, d, e, f, g, h).

For a general pixel $p(x,y)$, the relevant pixels are as follows:

$$(x-1, y-1),\ (x, y-1),\ (x+1, y-1),\ (x-1, y),\ (x+1, y),\ (x-1, y+1),\ (x, y+1),\ (x+1, y+1). \tag{8.10}$$

So far, we have studied some general issues and fundamental techniques that are used in image processing and analysis. Next, we will discuss particular techniques that are used for specific applications.

Example 8.3

In the image of Figure 8.23, starting with pixel (4, 3), find all succeeding pixels that can be considered as connected to each other based on +4-, ×4-, H6-, V6-, and 8-connectivity rules.

Solution Figure 8.24 shows the results of the five connectivity searches. For each search, take one pixel, find all others that are connected to it based on the rule you are working with, and then search the pixels that were found to be connected to the previous ones for additional connected pixels, until you are done. All of the remaining pixels will not be connected. We will use the same rules later for other purposes, such as region growing.

1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

Figure 8.23 The image for Example 8.3.

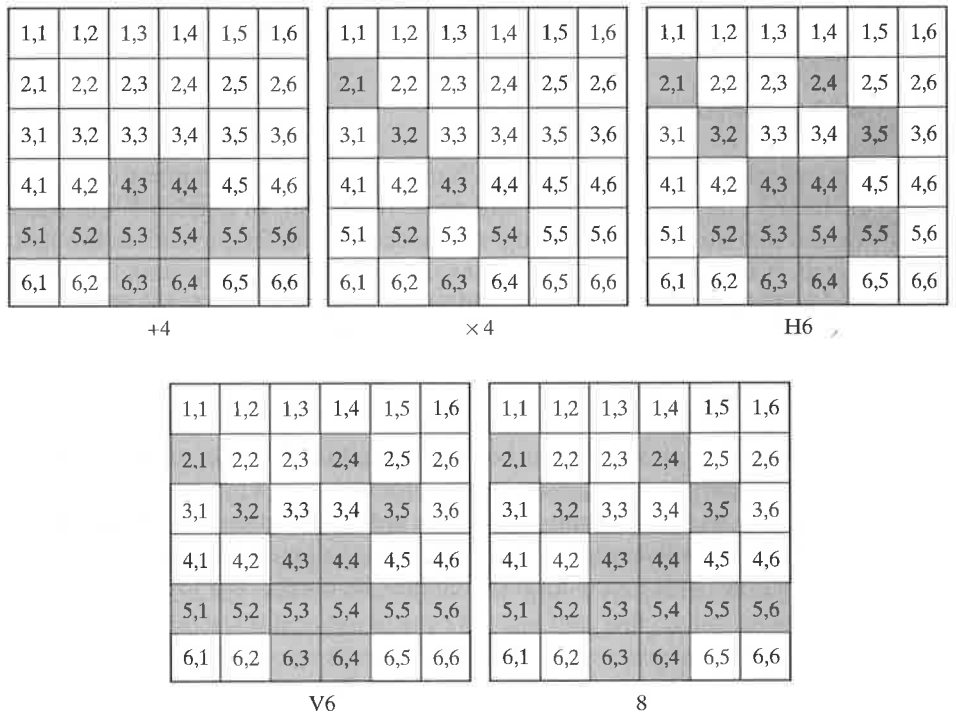


Figure 8.24 The results of the connectivity searches for Example 8.3.

8.17 NOISE REDUCTION

Like other signal-processing mediums, vision systems contain noises. Some noises are systematic and come from dirty lenses, faulty electronic components, bad mem-

ory chips, and low resolution. Others are random and are caused by environmental effects or bad lighting. The net effect is a corrupted image that needs to be preprocessed to reduce or eliminate the noise. In addition, sometimes images are not of good quality, due to both hardware and software inadequacies; thus, they have to be enhanced and improved before other analyses can be performed on them. On the hardware level, in one attempt [5], an on-chip correction scheme was devised for defective pixels in an image sensor. In this scheme, readouts from nearest neighbors were substituted for defective pixels that had been identified. However, in general, software schemes are used for most filtering operations.

Filtering techniques are divided into two categories. *Frequency-related techniques* operate on the Fourier transform of the signal, whereas *spatial-domain techniques* operate on the image at the pixel level, either locally or globally. The following is a summary of a number of different operations for reducing noise in an image.

8.17.1 Convolution Masks

As was mentioned in Section 8.10, a mask may be used for many different purposes, including filtering operations and noise reduction. In Section 8.9, it was also mentioned that noise, as well as edges, produces higher frequencies in the spectrum of a signal. It is possible to create masks that behave like a low-pass filter, such that the higher frequencies of an image are attenuated while the lower frequencies are not changed very much. Thereby, the noise is reduced.

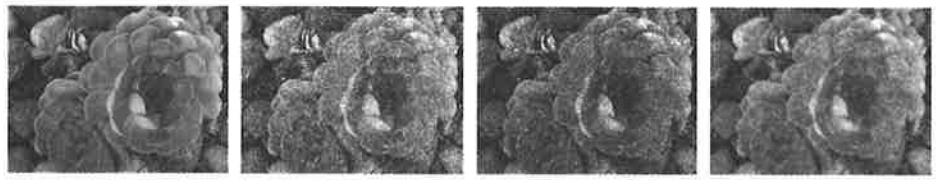
Neighborhood Averaging

Neighborhood averaging can be used to reduce noise in an image, but it also reduces the sharpness of the image. Consider the 3×3 mask shown in Figure 8.25 together with its corresponding values, as well as a portion of an imaginary image with its gray levels indicated. As you can see, all the pixels but one are at a gray value of 20. The pixel with a gray level of 100 may be considered to be noise, since it is different from the pixels around it. Applying the mask over the corner of the image, with a normalizing value of 9 (the sum of all the values in the mask), yields:

20	20	20	20				
20	100	20	20				
20	20	20	20				

1	1	1
1	1	1
1	1	1

Figure 8.25 Neighborhood-averaging mask.



(*a*) (*b*) (*c*) (*d*)

Figure 8.26 Neighborhood averaging of an image.

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

5×5

1	2	1
2	4	2
1	2	1

3×3

Figure 8.27 5×5 and 3×3 Gaussian averaging filters.

$$X = \frac{(20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 100 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1)}{9}$$

$$= 29.$$

As a result of applying the mask on the indicated corner, the pixel with the value of 100 changes to 29, and the large difference between the noisy pixel and the surrounding pixels (100 vs. 20) becomes much smaller (29 vs. 20), thus reducing the noise. With this characteristic, the mask acts as a low-pass filter. Notice that the operation will introduce new gray levels into the image (29) and thus will change its histogram. Similarly, this averaging low-pass filter will also reduce the sharpness of edges, making the resulting image softer and less focused. Figure 8.26 shows an original image (a), a corrupted image with noise (b), the image after a 3×3 averaging filter application (c), and the image after a 5×5 averaging filter application (d). As you can see, the 5×5 filter works even better than the 3×3 filter, but requires a bit more processing.

There are other averaging filters, such as the Gaussian averaging filter (also called the mild isotropic low-pass filter), which is shown in Figure 8.27. This filter will similarly improve an image, but with a slightly different result.

8.17.2 Image Averaging

In this technique, a number of images of the exact same scene are averaged together. Since the camera has to acquire multiple images of the same scene, all actions in

the scene must be halted. As a result, in addition to being time consuming, the technique is not suitable for operations that are dynamic and change rapidly. Image averaging is more effective with an increased number of images. It is fundamentally useful for noise that is random; if the noise is systematic, its effect on the image will be exactly the same for all multiple images, and as a result, averaging will not reduce the noise. If we assume that an acquired image $A(x,y)$ has random noise $N(x,y)$, then the desired image $I(x,y)$ can be found from averaging because the summation of random noises will be zero; that is,

$$A(x,y) = I(x,y) + N(x,y)$$

$$\frac{\sum_n A(x,y)}{n} = \frac{\sum_n I(x,y) + N(x,y)}{n} = \frac{\sum_n I(x,y)}{n} + \frac{\sum_n N(x,y)}{n} = I(x,y). \quad (8.11)$$

Although image averaging reduces random noise, unlike neighborhood averaging, it does not blur the image or reduce its focus.

8.17.3 Frequency Domain

When the Fourier transform of an image is calculated, the frequency spectrum might show a clear frequency for the noise, which in many cases can be selectively eliminated by proper filtering.

8.17.4 Median Filters

One of the main problems in using neighborhood averaging is that, along with removing noises, the filter will blur edges. A variation of this technique is to use a median filter, in which the value of the pixel is replaced by the median of the values of the pixels in a mask around the given pixel (the given pixel plus the eight surrounding pixels), sorted in ascending order. A median is the value such that half of the values in the set are below and half are above the median (also called the 50th percentile). Since, unlike an average, the median is independent of the value of any single pixel in the set, the median filter will be much stronger in eliminating spikelike noises without blurring the object or decreasing the overall sharpness of the image. Suppose that we apply a median filter to the image in Figure 8.25. Then sorted values, in ascending order, will be 20, 20, 20, 20, 20, 20, 20, 20, 100. The median is thus the fifth 20 in the sequence. Replacing the center pixel's value with 20 will completely eliminate the noise. Of course, noises are not always this easily removed, but the example shows how the effect of median filters can be very different from averaging. Notice that median filters do not create any new gray levels, but they do change the histogram of the image.

Median filters tend to make the image grainy, especially if applied more than once. Consider the image in Figure 8.28(a). The gray values, in ascending order, are 1, 2, 3, 4, 5, 6, 7, 8, 9. The middle value is 5, resulting in the image in (b). Observe that the image has become grainy because the pixel sets with similar values appear longer (as in 5 and 5).

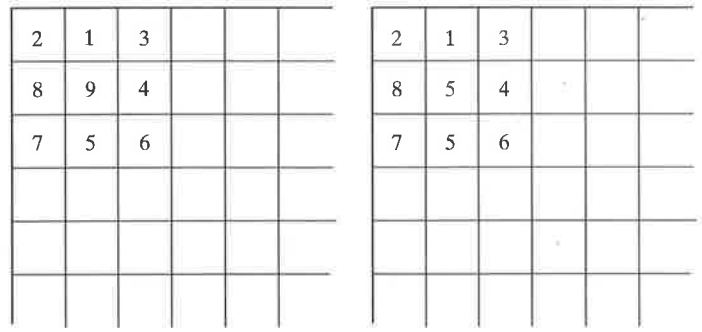


Figure 8.28 Application of a median filter.

Figure 8.29 (a) Original image. (b) The same image corrupted with a random Gaussian noise. (c) The image improved by a 3 × 3 median filter. (d) The same image improved with a 7 × 7 median filter.

Figure 8.29 shows an original image (a), the image corrupted with random Gaussian noise (b), and the image improved with a 3 × 3 median filter (c) and a 7 × 7 median filter (d).

8.18 EDGE DETECTION

Edge detection is a general name for a class of routines and techniques that operate on an image and result in a line drawing of the image. The lines represent changes in values such as cross sections of planes, intersections of planes, textures, lines, and colors, as well as differences in shading and textures. Some techniques are mathematically oriented, some are heuristic, and some are descriptive. All generally operate on the differences between the gray levels of pixels or groups of pixels through masks or thresholds. The final result is a line drawing or similar representation that requires much less memory to be stored, is much simpler to be processed, and saves in computation and storage costs. Edge detection is also necessary in subsequent processes, such as segmentation and object recognition. Without edge detection, it may be impossible to find overlapping parts, to calculate features such as a diameter and an area, or to determine parts by region growing. Different techniques of edge

detection yield slightly different results; thus, they should be chosen carefully and used wisely.

As was mentioned earlier, like noise, edges create higher frequencies in the spectrum and hence can be separated by high-pass filters. Masks can be designed to behave like a high-pass filter, reducing the amplitude of the lower frequencies while not affecting the amplitudes of the higher frequencies as much, thereby separating the noises and edges from the rest of the image. Consider the image and the mask (called a Laplacian1 filter) in Figure 8.30. The mask has negative numbers in it. Applying the mask to the image at the corner will result in

$$X = \frac{(20 \times -1 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 100 \times 4 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 20 \times -1)}{1}$$

$$= 320$$

The normalizing factor is 1, which results in the value of 100 being replaced with 320, accentuating the original difference (from 100 vs. 20 to 320 vs. 20). Since this mask accentuates differences (higher frequencies), it is a high-pass filter, which also means that the noise and edges of objects in images will be shown more effectively. As a result, the mask acts as an edge detector. Some high-pass filters can act as an image sharpener. Figure 8.31 shows some other high-pass filters.

Still other masks have the effect of differentiating an image through the use of gradients. In this case, the horizontal and vertical gradients between neighboring

20	20	20	20		
20	100	20	20		
20	20	20	20		

−1	0	−1
0	4	0
−1	0	−1

Figure 8.30 A typical high-pass edge detector mask (Laplacian1).

−1	−1	−1
−1	8	−1
−1	−1	−1

Laplacian2

0	−1	0
−1	6	−1
0	−1	0

Sharpen, low

0	−1	0
−1	5	−1
0	−1	0

Sharpen, medium

Figure 8.31 Other high-pass filters.

pixels are calculated and squared, and then the square root of the sum is found. Mathematically,

$$\nabla f = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}. \tag{8.12}$$

Equation (8.12) is equivalent to calculating the absolute value of the differences between pixel intensities. The three masks [6,7,8,9,10] shown in Figure 8.32 and called the Sobel, Roberts, and Prewitt edge detectors, effectively do the same gradient differentiation with somewhat different results and are very common. When they are applied to an image, the two pairs of masks calculate the gradients in the x and y directions, are added, and then are compared with a threshold. Figure 8.33 is an original image (a) with its edges detected by a Laplacian1 (b), Laplacian2 (c), Sobel (d), and Robert's (e) edge detectors. The result for other images may be different because the histogram of the image and the chosen thresholds have great effects on the final outcome. Some routines allow the user to change the thresholding values, and some do not. In each case, the user must decide which routine performs the best.

Other simple routines that are easy to implement and that yield continuous edges can be used for binary images. In one example [11], a search technique, dubbed left–right (L–R) in this book, is used that can quickly and efficiently detect edges in binary images of single objects which look like a blob. Imagine a binary image such as that shown in Figure 8.34. Suppose that gray pixels are "on" (or represent the object) and white pixels are "off" (or represent the background). Assume that a pointer is moving from one pixel to another, in any direction (up, down, right, or left). Anytime the pointer reaches an "on" pixel, it will turn left. Anytime it reaches an "off" pixel, it will turn right. Of course, as shown, depending on the direction of the pointer, "left" and "right" might mean different directions as well.

1	0
0	−1

0	1
−1	0

(*a*) Sobel

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

(*b*) Roberts

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

(*c*) Prewitt

Figure 8.32 The Sobel, Roberts, and Prewitt edge detectors.

(*a*) (*b*) (*c*) (*d*) (*e*)

Figure 8.33 An image (a) and its edges from Laplacian1 (b), Laplacian2 (c), Sobel (d), and Roberts (e) edge detectors.

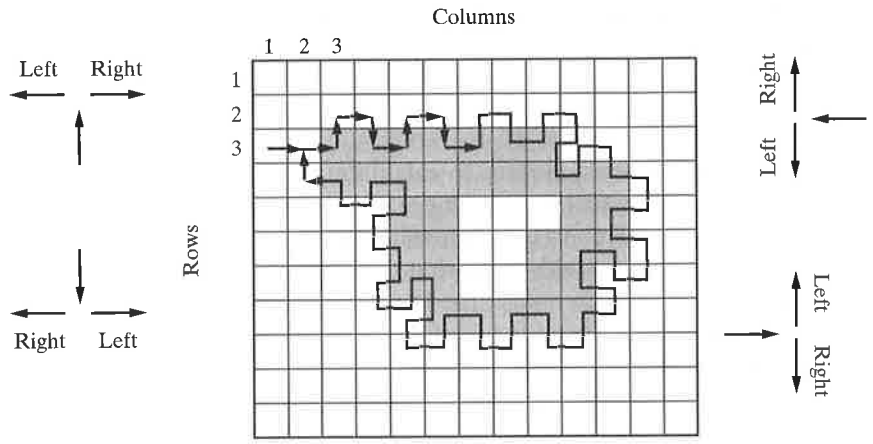


Figure 8.34 Left–right search technique for edge detection [11].

Starting at pixel 1,1, moving to 1,2, to the end of the row and then onto row 2, row 3, etc., the pointer finds the first "on" pixel at 3,3, turns left and encounters an "off" pixel, turns right twice, then left, and goes on. The process continues until the first pixel is reached. The collection of the pixels on the pointer's path is one continuous edge. Other edges can be found by continuing the process with a new pixel. In this example, the edge will be pixels 3,3; 3,4; 3,5; 3,6; , . . . ,3,9; 4,9; 4,10; 4,11; , ...

Masks may also be used to intentionally emphasize some characteristic of the image. For example, a mask may be designed to emphasize horizontal lines, vertical lines, or diagonal lines. Figure 8.35 shows three such masks. Figure 8.36 shows an

3	−6	3
3	−6	3
3	−6	3

Vertical mask

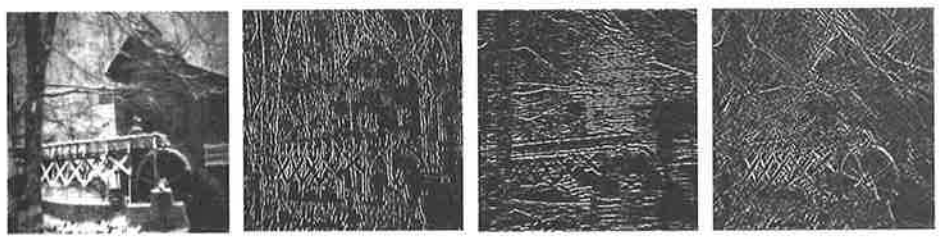
3	3	3
−6	−6	−6
3	3	3

Horizontal mask

3	3	−6
3	−6	3
−6	3	3

Diagonal mask

Figure 8.35 Masks emphasizing the vertical, horizontal, and diagonal lines of an image.



(*a*) (*b*) (*c*) (*d*)

Figure 8.36 An original image (a) with effects of vertical mask (b), horizontal mask (c), and diagonal mask (d).

original image (a), along with the effects of a vertical mask (b), a horizontal mask (c), and a diagonal mask (d).

8.19 HOUGH TRANSFORM

As you have probably noticed, in most edge detection techniques, the resulting edges are not continuous. However, in many applications, continuous edges are either necessary or preferred. For example, as we will see later, in region growing, edges that define an area or a region must be continuous and complete before a region-growing routine can detect the region and label it. In addition, it is desirable to be able to calculate the slope of detected edges in order to either complete a broken line or to detect objects. The Hough transform [12] is a technique used to determine the geometric relationship between different pixels on a line, including the slope of the line. For example, one can determine whether a cluster of points is on a straight line. This kind of determination aids in further developing an image in preparation for object recognition, since it transforms individual pixels into recognizable forms. The Hough transform is based on transforming the image space into an (r,θ) space, which in turn is based on the fact that an infinite number of straight lines go through any point in a plane. The normal to any one of these lines through the origin will have an angle of θ with respect to the x-axis and will be at a distance of r from the origin. The transformation into the (r,θ) plane (also called the Hough plane) showing these values is the Hough transform. Alternatively, a line in the xy-plane, with slope m and intercept c, can be transformed into a Hough plane of m,c, with x and y as its slope and intercept. Thus, a line in the xy-plane with a particular slope and intercept will transform into a point in the Hough plane. All lines through a point will transform into a single line in the Hough plane. If a group of points are collinear, their Hough transforms will all intersect. By examining these properties in a particular case, it can be determined whether a cluster of pixels is on a straight line. Hough transforms can also be used to determine the angle or orientation of a line, whereupon the orientation of an object in a plane can be determined by calculating the orientation of a particular line in the object.

To make this clearer, let's consider a plane xy (Figure 8.37) with a line in it. The line can be described by its slope (m) and intercept (c) as

$$y = mx + c. \tag{8.13}$$

Equation (8.13) can also be written in terms of m and c as

$$c = -xm + y, \tag{8.14}$$

where, in the mc-plane, the x and y will be the slope and the intercept.

Clearly, the line given by Equation (8.13) with m and c will be shown as a single point A in the mc- (Hough) plane. This is because all points on the line have the same slope and intercept m,c, and all produce the same location in that plane. Whether the line is drawn with this equation or in polar coordinates with (r,θ), the result is the same. Thus, a line (and all the points on it) are represented by a point in the Hough plane.

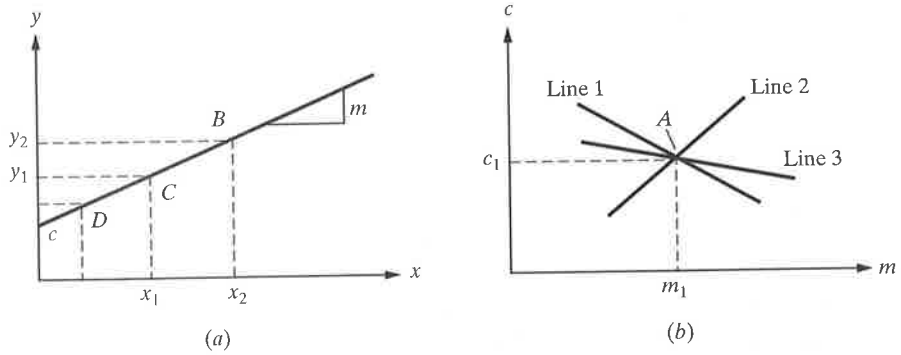


Figure 8.37 Hough transform.

Now consider the Hough plane in Figure 8.37 (b). Two lines intersect at point A. From Equation (8.14), Line 1 has a slope and intercept of x_1 and y_1, which can be shown as point C in the xy-plane. Similarly, Line 2 can be shown as point B in the xy-plane. Thus, each line in the mc-plane transforms into a point in xy-plane. This means that if two lines intersect in the mc-plane, they will form a line in the xy-plane. If a third line intersects at the same point, it will have to be on the same line in xy-plane. As a result, if a number of points in the xy-plane are on the same line, they correspond to intersecting lines in the mc-plane. This property can be used to determine whether a number of points are all on the same line. If one takes five points in the xy-plane, for example, so long as they all are on the same line, their corresponding lines will all intersect at the same point in the mc-plane. Of course, using the slope and intercept of the line, one can add points to the line to make it continuous or to close a shape.

Example 8.4

Following are the coordinates of five points:

y	x
3	1
2	2
1.5	3
1	4
0	5

Using the Hough transform, determine which points are on the same line. Find the slope and intercept of the line.

Solution Of course, *any* two points are on a line. So we will look for at least three points that will be on the same line. Looking at the graph of the points, we see that it is very easy — even trivial — to answer the questions. However, in computer vision, since the computer does not have the intelligence to understand an image, it must be calculated. Imagine having thousands of points in a computer file representing an image. It is impossible, for either a computer or a human being, to tell which points are on the same line and which are not. We will use a Hough transform to determine which points fall on the same line. The following table summarizes the lines formed in the mc-plane that correspond to the points shown in the xy-plane:

y	x	xy	mc
3	1	$3 = m1 + c$	$c = -1m + 3$
2	2	$2 = m2 + c$	$c = -2m + 2$
1.5	3	$1.5 = m3 + c$	$c = -3m + 1.5$
1	4	$1 = m4 + c$	$c = -4m + 1$
0	5	$0 = m5 + c$	$c = -5m + 0$

Figure 8.38 shows the five corresponding lines drawn in the mc-plane. Three of the lines are intersecting, while the other two are not. The latter lines correspond to points 1,3 and 5,0. The slope and intercept of the line representing the points that are on the same line are -0.5 and 3, respectively. Once again, these lines intersect at other points as well, indicating that any two points form a line. This shows how the Hough transform can be cluttered with numerous intersecting lines. Determining which lines are intersecting is the main issue in Hough transform analysis.

A similar analogy may be made for circles and points instead of lines and points. All points on a circle correspond to intersecting circles in the Hough plane. (See [7] for more information.)

The Hough transform has many desirable features. For example, because each point in the image is treated independently, all points can be processed simultaneously with parallel processing methods. This makes the Hough transform a suitable candidate for real-time processing. It is also insensitive to random noise, since individual points do not greatly contribute to the final count of the part itself. However, the Hough transform is computationally intensive. To reduce the number of calculations needed to determine whether lines are actually intersecting

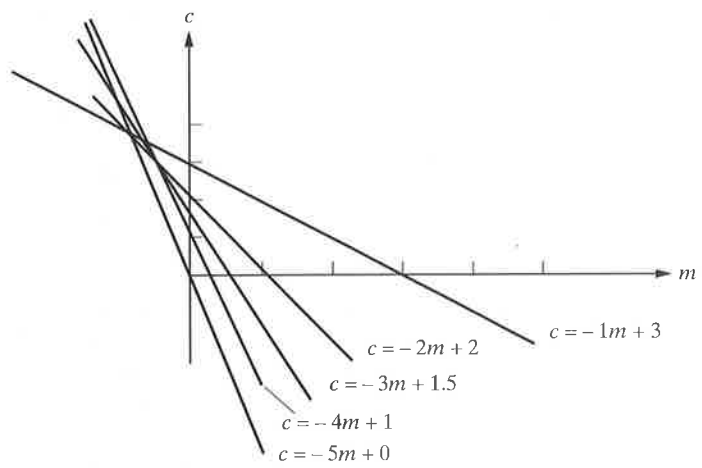


Figure 8.38 Hough transform for Example 8.4.

with each other at the same location, one must define a "circle" (or other boundary) within which, if the lines are intersecting with each other, they are assumed to be intersecting in general. Many variations of the Hough transform have been devised to increase its efficiency and utility for different tasks, including object recognition [13].

8.20 SEGMENTATION

Segmentation is a generic name for a number of different techniques that divide the image into segments of its constituents. The purpose of segmentation is to separate the information contained in the image into smaller entities that can be used for other purposes. For example, an image can be segmented by the edges in the scene, by small areas, etc. Each of these entities can then be used for further processing, representation, or identification. Segmentation includes, but is not limited to, edge detection, region growing, and texture analysis.

The early segmentation routines were all based on edge detection of simple geographic models such as polyhedrons. In the three-dimensional analysis of objects, models such as cylinders, cones, spheres, and cubes were used as well. Although these shapes and figures do not necessarily match those of any real objects, they provided a means for early developmental work, which evolved into more sophisticated routines and techniques. They also provided a means of developing routines that could process complicated shapes and recognize objects. As an example, with very little processing power the routines could model a tree as a cone mounted on a cylinder and could match the resulting figure with a model of a tree. The tree could thus be expressed with only a few pieces of information, such as the diameters of the cone and cylinder and their heights. Representing *all* the information pertaining to a tree could be staggering in comparison.

8.21 SEGMENTATION BY REGION GROWING AND REGION SPLITTING

Region growing and image splitting are techniques of segmentation, as are edge detection routines. Through these techniques, an attempt is made to separate the different parts of an image into segments or components with similar characteristics that can be used in further analysis, such as object detection. While an edge detector will find the separation lines of textures, colors, planes, and gray levels, segmentation by regions will naturally result in complete and closed boundaries. (For a survey of other segmentation techniques, see [14].)

Two approaches are used for region segmentation. One is to grow regions by similar attributes, such as gray-level ranges or other similarities, and then try to relate the regions by their average similarities or spatial relationships. The other is region splitting, which will split images into smaller areas by using their finer differences.

One technique of region splitting is *thresholding*. The image is split into closed areas of neighboring pixels by comparing them with thresholding value or range. Any pixel that falls below a threshold (or between a range of values) will belong to

a given region; otherwise, it belongs to another region. Thresholding will split the image into a series of regions or clusters of pixels that have common or similar attributes. Generally, although it is a simple technique, it is not very effective, because choosing an appropriate threshold is difficult. The results are also highly dependent on the threshold value and will change accordingly when the thresholds change. Still, thresholding is a useful technique under certain conditions, such as backlighting, and for images with relatively uniform regions.

In region growing, first nuclei regions are formed on the basis of some specific selection law. (Nuclei regions are the small clusters of pixels that are formed at the beginning of segmentation. They are usually small and act as a nucleus for subsequent growing and merging, as crystals do in alloys.) The result is a large number of little regions. Successively, these regions are combined into larger regions on the basis of some other attributes or rules. Although these rules will merge many smaller regions together to create a smoother set of regions, they may unnecessarily merge certain features that should not be merged, such as holes, smaller but distinct areas, or distinct areas with similar intensities.

A simple search technique for growing regions for a binary image (or, with the application of thresholding, for gray images as well), uses a bookkeeping approach to find all pixels that belong to the same region [15]. Figure 8.39 shows a binary image. Each pixel is referred to by a pair of index numbers. Assume that a pointer starts at the top and will search for a nucleus to start a region. As soon as a nucleus that does not already belong to another region is found, the program assigns a region number to it. All pixels connected to that nucleus will receive the same region number and are placed in a stack. The search continues with all the pixels in the stack until the stack is emptied. The pointer will then continue searching for a new nucleus and a new region number.

It is important to decide what form of connectivity is to be used in growing regions, as the form determines the final outcome. As was discussed in Section 8.16, +4-, ×4-, H6-, V6-, and 8-connectivity can be used for region growing. In Figure 8.39, the first nucleus is found at cell 2*d*. Suppose we have chosen +4-connectivity.

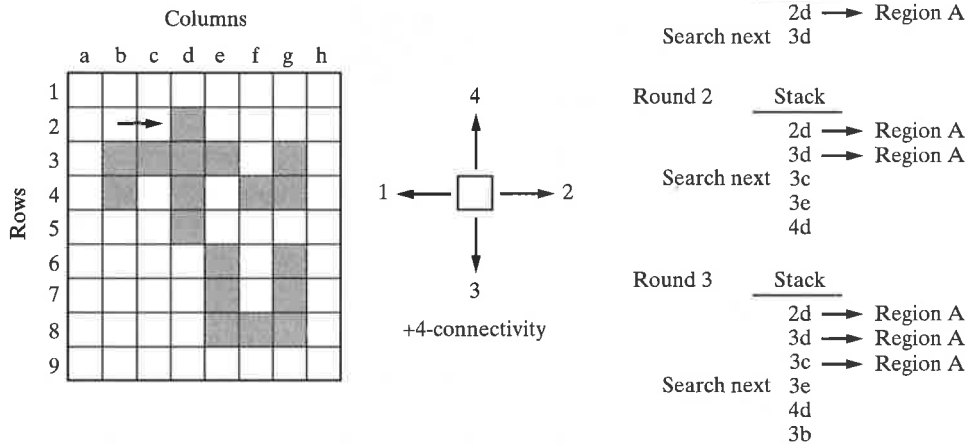


Figure 8.39 Region growing based on a search technique.

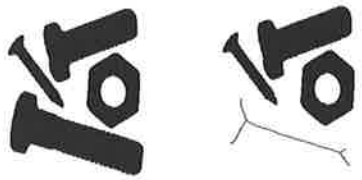
Then the program will check the four corresponding pixels around the nucleus to determine their connectivity. If there is an "on" pixel, the index numbers of its location are placed in a stack, the cell is given the region number (A), and the pointer is moved down in the stack to the next cell, 3*d*. At this location, the connectivity of pixels around the cell is checked again, the "on" pixel index numbers are placed in the search stack, the cell is given the A region designation, and the process is repeated for the next index number in the stack, 3*c*. The process continues until the stack is empty.

Notice that, on the basis of +4-connectivity, as the pointer gets to pixels 4*f* and 6*e*, it will not assign them to the same region. On the basis of ×4-, H6-, V6-, or 8-connectivity, both regions would be part of region A. Otherwise, the pointer continues until new nuclei are found for the next regions — say, region B and region C.

This kind of search technique is nothing more than a bookkeeping instrument to make sure that the computer program can find all connected pixels in the region without missing any. Otherwise, it is a simple search technique.

8.22 BINARY MORPHOLOGY OPERATIONS

Morphology operations refer to a family of operations that are performed on the shape (and thus the morphology) of subjects in an image. They include many different operations, both for binary and gray images, such as thickening, dilation, erosion, skeletonization, opening, closing, and filling. These operations are performed on an image in order to aid in its analysis, as well as to reduce the "extra" information that may be present in the image. For example, consider the binary image in Figure 8.40 and the stick figure representing one of the bolts. As we will see later, a moment equation may be used to calculate the orientation of the bolts. However, the same equation can also be applied to the stick figure of the bolt, but with much less effort. As a result, it would be desirable to convert the bolt to its stick figure or skeleton. In the sections that follow we will discuss a few of these operations.



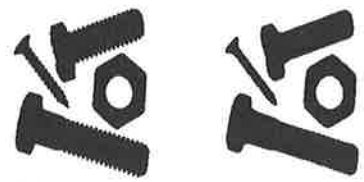
(*a*) (*b*)

Figure 8.40 The binary image of a bolt and its stick (skeleton) representation.

8.22.1 Thickening Operation

The thickening operation fills the small holes and cracks on the boundary of an object and can be used to smooth the boundary. In the example of Figure 8.40 (a), a thickening operation will reduce the appearance of the threads of the bolts.

This will become important when we try to apply other operations, such as skeletonization, to the object. The initial thickening will prevent the creation of whiskers caused by the threads, as we will see later. Figure 8.41 shows the effect of three rounds of thickening operations on the threads of the bolts.

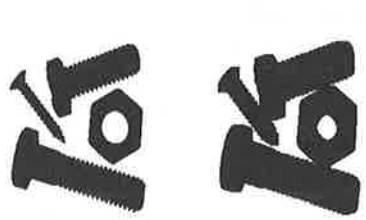


(*a*) (*b*)

Figure 8.41 The threads of the bolts are removed by a triple application of a thickening operation, resulting in smooth edges.

8.22.2 Dilation

In dilation, the background pixels that are 8-connected to the foreground (object) are changed to foreground pixels. As a result, a layer is effectively added to the object every time the process is implemented. Because dilation is performed on pixels that are 8-connected to the object, repeated dilations can change the shape of the object. Figure 8.42(b) is the result of four dilation operations on the objects in Figure 8.42(a). As can be seen, the objects have bled into one piece. With additional applications of dilation, the objects, as well as the disappearing hole, can become one solid piece and hence cannot be recognized as distinct objects anymore.



(*a*) (*b*)

Figure 8.42 Effect of dilation operations. Here, the objects in (a) were subjected to four rounds of dilation (b).

8.22.3 Erosion

In erosion, foreground pixels that are 8-connected to a background pixel are eliminated. This effectively eats away a layer of the foreground (the object) each time the operation is performed. Figure 8.43(b) shows the effect of three repetitions of the erosion operation on the binary image in Figure 8.43(a). Since erosion removes one pixel from around the object, the object becomes increasingly thinner with each pass. However, erosion disregards all other requirements of shape representation. It will remove one pixel from the perimeter (and holes) of the object even if the shape of the object is eventually lost, as in (c) with seven repetitions, where one bolt is completely lost and the nut will soon disappear. The final result of too many erosions will be the loss of the object. That is to say, if the reversing operation of dila-

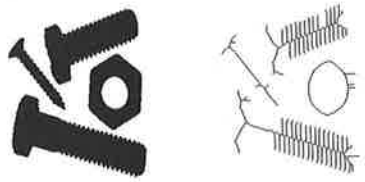
(a) (b) (c)

Figure 8.43 Effect of erosion on objects (a) with (b) 3 and (c) 7 repetitions.

tion, which adds one pixel to the perimeter of the object with each pass, is used, the dilated object may not resemble the original object at all. In fact, if the object is totally eroded to one pixel, dilation will result in a square or a circle. As a result, erosion can irreparably damage the image. However, it can also be successfully used to eliminate unwanted objects in an image. For example, if one is interested in identifying the largest object in an image, successive erosions will eliminate all other objects before the largest is eliminated. Thus, the object of interest can be identified.

8.22.4 Skeletonization

A *skeleton* is a stick representative of an object in which all thicknesses have been reduced to one pixel at any location. Skeletonization is a variation of erosion. Whereas in erosion the thickness of an object may go to zero and the object may be totally lost, in skeletonization, as soon as the thickness of the object becomes one pixel, the operation at that location stops. Also, although in erosion the number of repetitions are chosen by the user, in skeletonization the process automatically continues until all thicknesses are one pixel thick. (The program stops when no new changes are made as a result of the operation.) The final result of skeletonization is a stick figure (skeleton) of the object, which is a good representation of it—indeed, sometimes much better than the edges. Figure 8.44(b) shows the skeleton of the original objects in Figure 8.44(a). The whiskers are created because the objects were not smoothed by thickening. As a result, all threads are reduced to one pixel, creating the whiskers. Figure 8.45 shows the same objects, thickened to eliminate the threads, resulting in a clean skeleton. Figure 8.45(c) is the result of dilating the skeleton seven times. As can be seen, the dilated objects are not the same



(a) (b)

Figure 8.44 The effect of skeletonization on an image without thickening. The threads of the bolts have resulted in the creation of whiskers.

(a) (b) (c)

Figure 8.45 The skeleton of the objects in (a) after the application of thickening results in a clean skeleton (b). Part (c) is the dilated image of the skeletons.

as the original ones. Notice how the smaller screw appears to be as big as the bigger bolts.

Although dilating a skeleton will also result in a shape different from that of the original object, skeletons are useful in object recognition, since they are generally a better representation of an object than other representations. When a stick representation of an object is found, it can be compared with the available a priori knowledge of the object for matching.

8.22.5 Open Operation

Opening is erosion followed by dilation and causes a limited smoothing of convex parts of the object. Opening can be used as an intermediate operation before skeletonization.

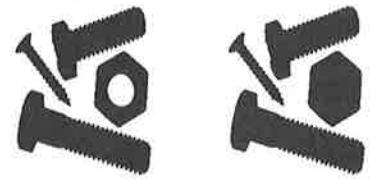
8.22.6 Close Operation

Closing is dilation followed by erosion and causes a limited smoothing of convex parts of the object. Like opening, closing can be used as an intermediate operation before skeletonization.

8.22.7 Fill Operation

The fill operation fills the holes in the foreground (object). In Figure 8.46, the hole in the nut is filled with foreground pixels until it is eliminated.

Information on other operations may be found in vision systems manufacturers' references. Different companies include other operations to make their software unique.



(a) (b)

Figure 8.46 As a result of a fill operation, the hole in the nut is filled with foreground pixels and is thus eliminated.

8.23 GRAY MORPHOLOGY OPERATIONS

Gray morphology operations are similar to binary morphology operations, except that they operate on a gray image. Usually, a 3×3 mask is used to apply the operations, where each cell in the mask may be either 0 or 1. Imagine that a gray image is a multilayered three-dimensional image in which the light areas are peaks and the dark areas are valleys. The mask is applied to the image by moving it from pixel to pixel. Wherever the mask matches the gray values in the image, no changes are made. If the gray values of the pixels do not match the mask, they will be changed according to the selected operation, as described in the sections that follow.

8.23.1 Erosion

In this case, each pixel is replaced by the value of the darkest pixel in its 3×3 neighborhood, known as a *min operator* and effectively erodes the object. Of course, the result is dependent on which cells in the mask are 0 or 1. Gray morphology erosion removes light bridges between dark objects.

8.23.2 Dilation

In this case, each pixel is replaced by the value of the lightest pixel in its 3×3 neighborhood, known as a *max operator* and effectively dilates the object. Of course, the result is dependent on which cells in the mask are 0 or 1. Gray morphology dilation removes dark bridges between light objects.

8.24 IMAGE ANALYSIS

Image analysis is a collection of operations and techniques that are used to extract information from images. Among these operations and techniques are object recognition; feature extraction; analysis of the position, size, orientation, and other properties of objects in images; and extraction of depth information. Some techniques may be used for multiple purposes, as we will see later. For example, moment equations may be used for object recognition, as well as to calculate the position and orientation of an object.

Generally, it is assumed that image-processing routines have already been applied to the image or that they are available for further use, when needed, to improve and prepare the image for analysis. Image analysis routines and techniques may be used on both binary and gray images. Some of these techniques are discussed next.

8.25 OBJECT RECOGNITION BY FEATURES

Objects in an image may be recognized by their features, which may include, but are not limited to, gray-level histograms, morphological features such as area, perimeter, number of holes, etc., eccentricity, cord length, and moments. In many cases, the information extracted is compared with a priori information about the object, which may be in a lookup table. For example, suppose that two objects are present in the image, one with two holes and one with one hole. By using previously discussed routines, it is possible to determine how many holes each part has, and by comparing the two parts (let's say they are assigned regions 1 and 2) with information about them in a lookup table, it is possible to determine what each of the two parts are. As another example, suppose that a moment analysis of a known part is performed at different angles, that the moment of the part relative to an axis is calculated for these angles, and that the resulting data are collected in a lookup table. Later, when the moment of the part in the image is calculated relative to the same axis and is compared with the information in the lookup table, the angle of the part in the image can be estimated.

We next discuss a few techniques and different features that may be used for object recognition.

8.25.1 Basic Features Used for Object Identification

The following morphological features may be used for object recognition and identification:

a. The average, maximum, or minimum gray levels may be used to identify different parts or objects in an image. As an example, assume that the image is divided into three parts, each with a different color or texture that will create different gray levels in the image. If the average, maximum, or minimum gray levels of the objects are found, say, through histogram mapping, the objects can be recognized by comparing them with this information. In other cases, even the presence of one particular gray level may be enough to recognize a part.

b. The perimeter, area, and diameter of an object, as well as the number of holes it has and other morphological characteristics, may be used to identify the object. The perimeter of an object may be found by first applying an edge detection routine and then counting the number of pixels on the perimeter. The left-right search technique of Section 8.18 can also be used to calculate the perimeter of the object by counting the number of pixels that are on the object's path in an accumulator. The area of the object can be calculated by region-growing techniques. Moment equations can also be used, as will be discussed later. The diameter of a noncircular object is defined as the maximum distance between any two points on any line that crosses the identified area of the object.

c. An object's aspect ratio is the ratio of the width to the length of a rectangle enclosed about the object, as shown in Figure 8.47. All aspect ratios, except for

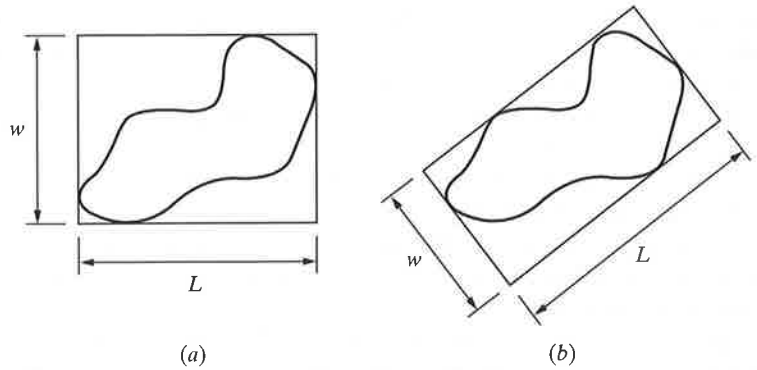


Figure 8.47 Aspect ratio of an object, with minimum aspect ratio shown in part (b).

the minimum aspect ratio, are sensitive to orientation. Thus, the minimum aspect ratio is generally used to identify objects.

d. Thinness is defined as one of the following two ratios:

$$1. \quad \text{Thinness} = \frac{(\text{perimeter})^2}{\text{area}}. \tag{8.15}$$

$$2. \quad \text{Thinness} = \frac{\text{diameter}}{\text{area}}. \tag{8.16}$$

e. Moments are discussed in the next section, due to their special importance.

8.25.2 Moments

Imagine an object within a binary image. The object is represented by pixels that are turned on, and the background is represented by pixels that are turned off. This effect can be achieved either by backlighting or by rendering the image in binary form.

Now consider the general moment equation

$$M_{a,b} = \sum_{x,y} x^a y^b, \tag{8.17}$$

where $M_{a,b}$ is the moment of the object within the image with indices a and b and x and y are the coordinates of each pixel that is turned on within the image, raised to powers of a and b, as in Figure 8.48. In such a case, a routine based on Equation (8.17) will first determine whether each pixel belongs to the object (is turned on) and, if so, will then raise the coordinates of the location of the pixel to the given values of a and b. The summation of this operation over the whole image will be the particular moment of the object with a and b values. $M_{0,0}$ is the moment of the object with $a = 0$ and $b = 0$. This means that all x and y values are raised to a power of 0. $M_{0,2}$ means that all x values are raised to the power of 0, all y values are raised to power of 2, etc. All combinations of values between 0 and 3 are common.

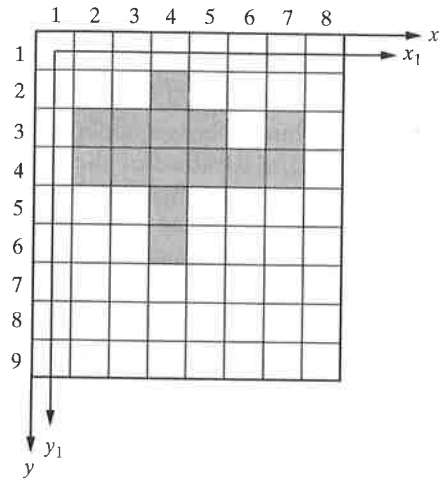


Figure 8.48 Calculation of the moment of an image. For each pixel that belongs to the object, the coordinates of the pixel are raised to the powers indicated by the moment's indices. The summation of the values thus calculated will be the particular moment of the image.

Distances x and y are measured either from a fictitious coordinate frame located at the edge of the image (x,y) or are measured from a coordinate frame formed by the first row and column of the image. Since the distances are measured by counting the number of pixels, the use of the first row and column as the coordinate frame is more logical. Note, however, that in this case all distances should be measured to the centerline of the pixel row or column. As an example, the first "on"-pixel in the second row is $I_{2,4}$. The x distance of the pixel from the x_1y_1 coordinate frame will be 3, whereas the same distance from the xy coordinate is 4 (or, more accurately, 3.5). As long as the same distances are used consistently, the choice is not important.

Based on the preceding discussion, since all numbers raised to the power of 0 are equal to 1, then all x^0's and y^0's are equal to 1. As a result, the moment $M_{0,0}$ is the summation of as many 1's as there are "on"-pixels, yielding the total number of "on"-pixels, which is the area of the object. In other words, the moment $M_{0,0}$ is the same as the area of the object. This moment can be used to determine the nature of an object and to distinguish it from other objects that have a different area. Obviously, the moment $M_{0,0}$ can also be used to calculate the area of an object within an image.

Similarly, $M_{0,1}$ is $\Sigma x^0 y^1$, or the summation of $1 \times y$'s, which is the same as the summation of each pixel area multiplied by its distance from the x-axis. This is similar to the first moment of the area relative to the x-axis. Then the location of the center of the area relative to the x-axis can be calculated by

$$\bar{y} = \frac{\sum y}{\text{area}} = \frac{M_{0,1}}{M_{0,0}}. \tag{8.18}$$

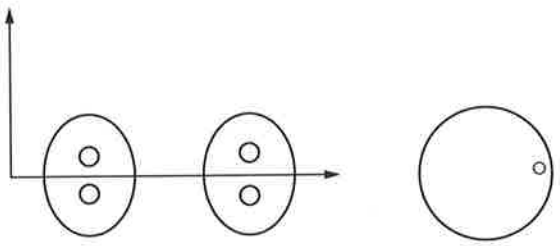
So, simply by dividing the two moments, you can calculate the $\bar{y}$ coordinate of the center of the area of the object. Similarly, the location of the center of the area relative to the y-axis will be

$$\bar{x} = \frac{\sum x}{\text{area}} = \frac{M_{1,0}}{M_{0,0}}. \tag{8.19}$$

This way, an object may be located within an image, independently of its orientation. (The orientation will not change the location of the center of an area.) Of course, this information can be used to locate an object, say, for grabbing by a robot.

Now consider $M_{0,2}$ and $M_{2,0}$. $M_{0,2}$ is $\Sigma x^0 y^2$ and represents the second moment of the area relative to the x-axis. Similarly, $M_{2,0}$ is the second moment of the area relative to the y-axis. As you can imagine, the moment of inertia of an object such as the one in Figure 8.48 will vary significantly as the object rotates about its center. Suppose that one would calculate the moments of the area about, say, the x-axis, at different orientations. Since each orientation creates a unique value, a lookup table that contains these values can later be used to identify the orientation of the object. Thus, if a lookup table containing the values of the moments of inertia of the known object at different orientations is prepared, the subsequent orientation of the object can be estimated by comparing its second moment with the values in the table. Of course, if the object translates within an image, its moments of inertia will also change. However, if the coordinates of the center of the area of the object are known, then, with a simple application of the parallel axes theorem, the second moments about the center of the area can be calculated independently of their location. As a result, with the use of the moment equations, an object, its location, and its orientation can be identified. In addition to identifying the part, the information can be used in conjunction with a robot controller to direct the robot to pick up the part or operate on it.

Other moments can be used similarly. For example, $M_{1,1}$ represents the product of inertia of the area and can also be used to identify an object. Higher order moments such as $M_{0,3}$,$M_{3,0}$,$M_{1,2}$, etc., can also be used to identify objects and their orientations. Imagine two objects that are relatively similar in shape, as in Figure 8.49(a). It is possible that the second moments, areas, perimeters, or other morphological characteristics of the objects may be similar or close to each other, such that they may not be useful in identifying the object. In this case, a small difference between the two objects may be exaggerated through higher order moments, making identification of the object possible. The same is true for an object with a small asymmetry (Figure 8.49(b)). The orientation of the object may be found by higher order moments.



(a) (b)

Figure 8.49 Small differences between objects or a small asymmetry in an object may be detected by means of higher order moments.

A *moment invariant* is a measure of an object based on its different moments and is independent of the location and orientation of the object, as well as of the scale factor used to represent the object. There are seven different moment invariants, of which one is

$$MI_1 = \frac{M_{0,0}M_{2,0} - M_{1,0}^2 + M_{0,0}M_{0,2} - M_{0,1}^2}{M_{0,0}^3}. \tag{8.20}$$

(See [6] for the other six moment invariants.)

Example 8.5

For the simple object in the low-resolution image of Figure 8.50, calculate the area, center of the area, and second moments of inertia of the object relative to the x_1 and y_1-axes.

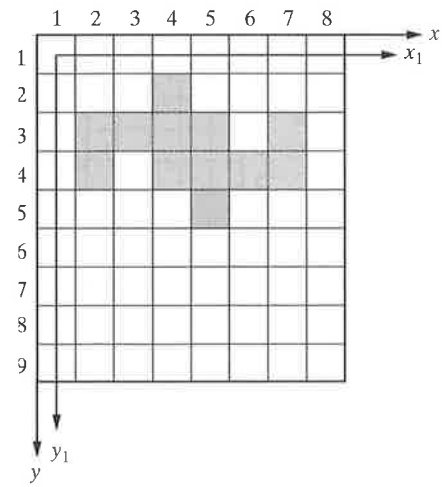


Figure 8.50 Image used for Example 8.5.

Solution Measuring the distances of each pixel from the x_1- and y_1-axes and substituting the measurements into the moment equations yields the following results:

$$M_{0,0} = \sum x^0 y^0 = 12(1) = 12;$$

$$M_{1,0} = \sum x^1 y^0 = \sum x = 2(1) + 1(2) + 3(3) + 3(4) + 1(5) + 2(6) = 42;$$

$$M_{0,1} = \sum x^0 y^1 = \sum y = 1(1) + 5(2) + 5(3) + 1(4) = 30;$$

$$\bar{x} = \frac{M_{1,0}}{M_{0,0}} = \frac{42}{12} = 3.5 \quad \text{and} \quad \bar{y} = \frac{M_{0,1}}{M_{0,0}} = \frac{30}{12} = 2.5;$$

$$M_{2,0} = \sum x^2 y^0 = \sum x^2 = 2(1)^2 + 1(2)^2 + 3(3)^2 + 3(4)^2 + 1(5)^2 + 2(6)^2 = 178;$$

$$M_{0,2} = \sum x^0 y^2 = \sum y^2 = 1(1)^2 + 5(2)^2 + 5(3)^2 + 1(4)^2 = 82.$$

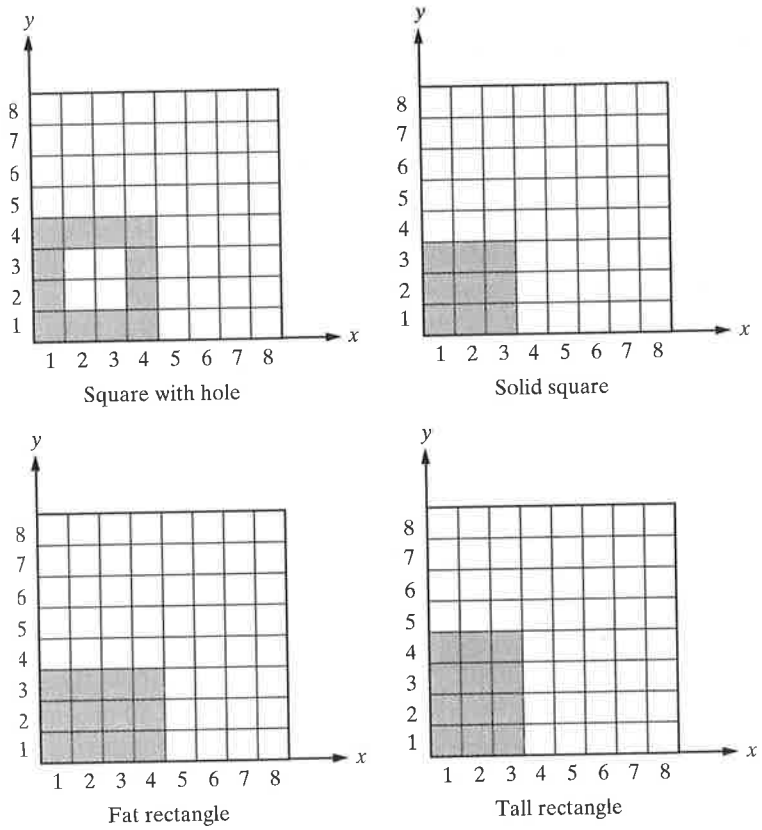


Figure 8.51 Image used for Example 8.6.

The same procedure may be used for an image with much higher resolution; there will just be many more pixels to deal with. However, a computer program can handle as many pixels as necessary without difficulty.

Example 8.6

In a certain application, a vision system looks at an 8 × 8 binary image of rectangles and squares. The squares are either 3 × 3-pixel solids, or 4 × 4 hollows, while the rectangles are 3 × 4 solids. Through guides, jigs, and brackets, we can be certain that the objects are always parallel to the reference axes, as shown in Figure 8.51, and that the lower left corner of the objects is always at the pixel 1,1. We want to use the moment equations only to distinguish the parts from each other. Find one set of lowest values a and b in the moment equation that would be able to do so, with corresponding values for each part. Use the absolute coordinates of each pixel for distances from the corresponding axes.

Solution Using the moment equations, we calculate the different moments for all four until we find one set that is unique for each object:

Square with hole	Solid square	Fat rectangle	Tall rectangle
$M_{0,0} = 12$	$M_{0,0} = 9$	$M_{0,0} = 12$	$M_{0,0} = 12$
$M_{0,1} = 30$	$M_{0,1} = 18$	$M_{0,1} = 24$	$M_{0,1} = 30$
$M_{1,0} = 30$	$M_{1,0} = 18$	$M_{1,0} = 30$	$M_{1,0} = 24$
$M_{1,1} = 75$	$M_{1,1} = 36$	$M_{1,1} = 60$	$M_{1,1} = 60$
$M_{0,2} = 94$	$M_{0,2} = 42$	$M_{0,2} = 56$	$M_{0,2} = 90$

The lowest set of moment indices that yields a unique solution for each object is $M_{0,2}$. Of course, $M_{2,0}$ would result in similar numbers.

Example 8.7

For the image of the screw in Figure 8.52, calculate the area, $\bar{x}$, $\bar{y}$, $M_{0,2}$, $M_{2,0}$, $M_{1,1}$, $M_{2,0}$ @ $\bar{x}$, $M_{0,2}$ @ $\bar{y}$, and the moment invariant. $M_{0,2}$ @ $\bar{y}$ means $M_{0,2}$ about centroidal $\bar{y}$-axis. Similarly, $M_{2,0}$ @ $\bar{x}$ is the moment of inertia about centroidal $\bar{x}$-axis.

Figure 8.52 Image used for Example 8.7.

Solution A macro called moments.macro was written for the Optimas™ 6.2 vision software to calculate the moments. In this program, distances used for moments are all in terms of the number of pixels and not in units of length. The values were calculated for five separate cases: horizontal, 30°, 45°, 60°, and vertical. Small variations in the results are due to rotations. Every time a part of an image is rotated, since every point in it must be converted with a sine or cosine function, the image changes slightly. Otherwise, the results are consistent. For example, as the part is rotated in place, the location of its center of area does not change. Also, the moment invariant is constant, and using information about the moment of inertia about the centroid, we can estimate the orientation of the part. This information can now be used to identify the object or to direct a robot controller to send the robot arm, with the proper orientation, to the location to pick up the part.

	Horizontal	30°	45°	60°	**Vertical**
Area	3713	3747	3772	3724	3713
$\bar{x}$	127	123	121	118	113
$\bar{y}$	102	105	106	106	104
$M_{0,2}$	38.8 E6	43.6 E6	46.4 E6	47.6 E6	47.8 E6
$M_{2,0}$	67.6 E6	62.6 E6	59 E6	53.9 E6	47.8 E6
$M_{1,1}$	48.1 E6	51.8 E6	52 E6	49.75 E6	43.75 E6
Moment Invariant	7.48	7.5	7.4	7.3	7.48
$M_{2,0}$ @ $\bar{x}$	7.5 E6	5.7 E6	3.94 E6	2.07 E6	0.264 E6
$M_{0,2}$ @ $\bar{y}$	0.264 E6	2.09 E6	3.77 E6	5.7 E6	7.5 E6

Although the moments.macro program cannot directly be used with other software, we list it next to show how simply a program can be developed to do similar moment op-

erations. The Excel part of the program is nothing more than a simple set of Excel equations that operate on the coordinates of all pixels and, later, are summed up. The following is a listing of the program:

```
/*MOMENTS.MAC PROGRAM Written by Saeed Niku, Copyright 1998

This macro checks an active image within the Optimas vision system and records the coordinates
of all pixels above the given threshold. It subsequently writes the coordinates into an Excel
worksheet, which determines the moments. Moments.mac will then read back and display the
data. The DDE commands communicate the data between Excel and the Optimas macro. If the
number of coordinates is more than 20,000 pixels, you must change the DDEPoke command
below. */

BinaryArray = GetPixelRect (ConvertCalibToPixels(ROI));
INTEGER NewArray[,]; Real MyArea; Real XBar; Real YBar;
Real Mymoment02; Real Mymoment20; Real Mymoment11; Real VariantM;
Real MyMXBar; Real MyMYBar;

For(XCoordinate = 0; XCoordinate < = (VectorLength(BinaryArray[0,])-1); XCoordinate ++)
{
      For(YCoordinate = 0; YCoordinate < = (VectorLength(BinaryArray[,0])-1); YCoordinate ++)
      {
            If (BinaryArray[YCoordinate,XCoordinate] > 100)
            {
                  NewArray ::= XCoordinate : YCoordinate;
            }
      }
}
hChanSheet1 = DDEInitiate ("Excel,""Sheet1");
DDEPoke(hChanSheet1,"R1C1:R20000C2,"NewArray);
DDETerminate(hChanSheet1);

Show("Please Enter to Show Values");
hChanSheet1 = DDEInitiate ("Excel,""Sheet1");
DDERequest(hChanSheet1,"R1C14,"MyArea);
DDERequest(hChanSheet1,"R2C14,"YBar);
DDERequest(hChanSheet1,"R3C14,"XBar);
DDERequest(hChanSheet1,"R4C14,"Mymoment02);
DDERequest(hChanSheet1,"R5C14,"Mymoment20);
DDERequest(hChanSheet1,"R6C14,"Mymoment11);
DDETerminate(hChanSheet1);

VariantM=(MyArea*Mymoment20*1000000.0-XBar*XBar
                  +MyArea*Mymoment02*1000000.0-YBar*YBar)
                  /(MyArea*MyArea*MyArea);

MyMYBar=(Mymoment20*1000000.0-MyArea*XBar*XBar)/1000000.0;
MyMXBar=(Mymoment02*1000000.0-MyArea*YBar*YBar)/1000000.0;

MacroMessage("Area=,"MyArea,"\n,""XBar=,"XBar,"\n,"
```



```
                  "YBar=,"YBar,"\n,""Moment02=,"Mymoment02," x10^6,"
                  "\n,""Moment20=,"Mymoment20," x10^6,""\n,""Moment11="
                  ,Mymoment11," x10^6,""\n,""Invariant 1="
                  ,VariantM);
MacroMessage("Moment20@Xbar=,"MyMXBar," x10^6,""\n,"
                  "Moment02@Ybar=,"MyMYBar," x10^6");
```

8.25.3 Template Matching

Another technique for object recognition is *model,* or *template, matching.* If a suitable line drawing of a scene is found, the topological or structural elements, such as the total number of lines (sides), vertices, and interconnections can be matched to a model. Coordinate transformations (e.g., rotation, translation, and scaling) can be performed to eliminate the differences between the model and the object resulting from differences in position, orientation, or depth between them. This technique is limited by the fact that a priori knowledge of the models is needed for matching. Thus, if the object is different from the models, they will not match, and the object will not be recognized. Another major limitation is that if one object is occluded by other objects, it will not match a model.

8.25.4 Discrete Fourier Descriptors

In a manner similar to a Fourier transform that is calculated for an analog signal, a discrete fourier transform (DFT) of a set of discrete points (such as pixels) can be calculated. This means that if the contour of an object within an image is found (such as in edge detection), the discrete pixels of the contour can also be used for DFT calculations. The result of a DFT calculation is a set of frequencies and amplitudes in the frequency domain that describe the spatial relationship of the points in question [16].

To calculate the DFT of a set of points in a plane, assume that the plane is the complex-number plane, such that each point is described by the relationship $x + iy$. If the contour surrounding the set of points is completely traced around, starting from any pixel, and the locations of the points are measured, the information can be used to calculate the corresponding frequency spectrum of the set. These frequencies can then be matched with the frequencies found for possible objects in a lookup table in order to determine the nature of the object. In one unpublished experiment, matching eight frequencies yielded enough information about the nature of the object (an airplane), and matching 16 frequencies could determine the type of airplane from a large class of planes. An advantage of this technique is that the Fourier transform can be normalized for size, position, and orientation very simply. A disadvantage of the technique is that it requires a complete contour of the object. Of course, other techniques, such as the Hough transform, can be used to complete broken contours of objects.

8.25.5 Computed Tomography (CT)

Tomography is a technique of determining the distribution of material density in the part being examined. In computed tomography (CT), a three-dimensional

image of the density distribution of the object is reconstructed from a large number of two-dimensional images of the material density taken by different scanning techniques, such as X-rays or ultrasonics. In computed tomography, it is assumed that the part consists of a sequence of overlaying slices. Images of the density distribution of each slice are taken repeatedly around the object. Although partial coverage of the part has been used as well, a complete coverage of 360° is preferred. The data are stored in a computer and subsequently are reduced to a three-dimensional image of the part's density distribution, which is shown on a CRT. Tomography is the only efficient technique for mapping the internals of objects.

Although this technique is completely different from the other techniques mentioned, it is a viable one for object recognition. In many situations, either alone or in conjunction with other techniques, CT may be the only way to recognize an object or differentiate it from other, similar objects. Specifically, in medical situations, a CT scan can be used in conjunction with medical robots, with the three-dimensional mapping of the internal organs of the human body used to direct the robot as it performs surgical operations.

8.26 DEPTH MEASUREMENT WITH VISION SYSTEMS

Depth information is extracted from a scene by means of two basic techniques. One is the use of range finders in conjunction with a vision system and image-processing techniques. In this combination the scenes are analyzed in relation to the information gathered by the range finders about the distances of different portions of an environment or the location of particular objects or sections of the object in that environment. Second is the use of binocular or stereo vision. In this technique, as in humans and animals, two cameras look at a scene simultaneously, or one camera takes an image of a scene, moves a certain distance, takes another image, and continues for multiple images. As long as the scene does not change during this operation, the results will be the same as that obtained with the use of multiple cameras. Since the locations of the two cameras in relation to any particular point in the scene are slightly different, the two cameras will develop slightly different images. By analyzing and measuring the differences between the two scenes, depth information can be extracted.

8.26.1 Scene Analysis vs. Mapping

Scene analysis refers to the analysis of complete scenes of images developed by a camera or another, similar device. In other words, the image is a complete replica of the scene, within the limits of resolution of the device. In this case, more processing is generally required to extract information from the image, but more information can be extracted. For instance, in order to identify an object within a scene, the image may have to be filtered and enhanced, as well as segmented by edge detection or thresholding. Then the part is isolated by region growing and identified by extracting its features and comparing them with information in a template or a lookup table. By contrast, *mapping* refers to drawing the surface topology of a scene or object where the image consists of a set of discrete distance measurements, usually at low resolutions. The final image is a collection of lines that are linked with the relative positions of points on the object at discrete locations. Since the image is already sliced, less processing is required in the analysis of mapped images, but less information can be extracted from the scene as well. Each technique has its own merits, benefits, and limitations and is used for different purposes, including navigation.

8.26.2 Range Detection and Depth Analysis

Range measurement and depth analysis are performed using many different techniques, such as active ranging [22], stereo imaging, scene analysis, or specialized lighting. Humans employ a combination of techniques to extract information about the depth and positional relationship between different elements of an image. Even in a two-dimensional image, humans can extract useful information from details such as the changing size of similar elements, vanishing lines, shadows, and the changing intensity of textures and shades. Since many artificial-intelligence techniques are based on, and in fact are studied to gain an understanding of, the way humans do things, a number of depth measurement techniques are designed after similar human operations [17].

8.26.3 Stereo Imaging

An image is the projection of a scene onto the image plane through an ideal lens. Thus, every point in the image will correspond to a certain point in the scene. However, the distance of the point from the plane is lost in this projection and cannot be retrieved simply from the single scene. If two images of the same scene are formed, then the relative depths of different points from the image plane can be extracted by comparing the two images; the differences represent the spatial relationship between different points [18,19]. Humans do the same automatically by combining the two images and forming a three-dimensional one [20,21]. The stereo image used for depth measurement is actually considered to be a 2.5-dimensional image; many more images are required to form a true three-dimensional image.

Depth measurement using stereo images requires two operations:

1. A determination of the point pairs in the two images that correspond to the same point in the scene. This is called the *correspondence* or *disparity* of the point pair. It is a difficult operation to carry out, since some points in one image may not be visible in another, or because sizes and spatial relationships may be different in the two images due to perspective distortion.
2. A determination of the depth or location of the point on the object or in the scene by triangulation or other techniques.

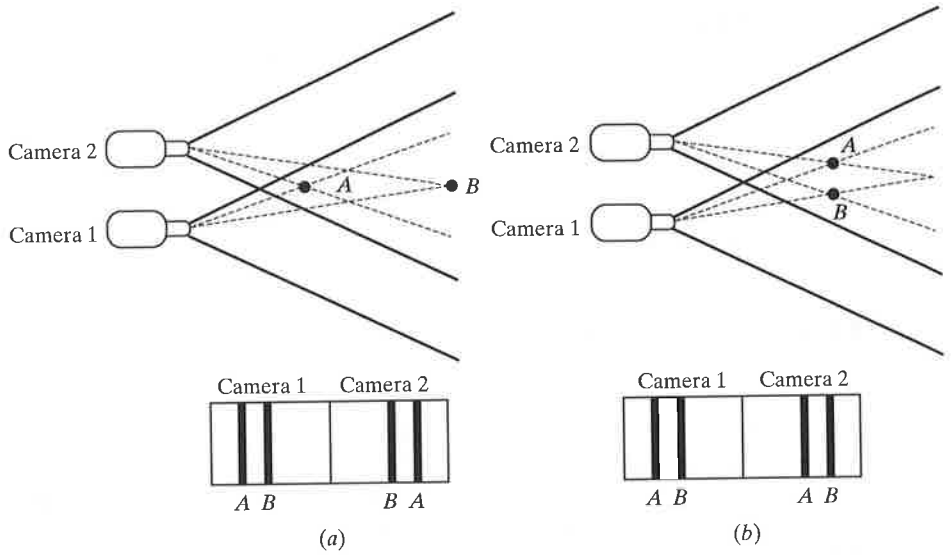


Figure 8.53 Correspondence problem in stereo imaging.

Generally, if the two cameras (or the relative locations of a single camera used twice to get two images of a static scene) are accurately calibrated, triangulation is relatively simple as long as enough corresponding points have been found.

Correspondence points can be determined by matching specific features, such as corners or small segments, from the two images. Depending on their locations, correspondence points can create matching problems. Consider the two marks *A* and *B* in Figure 8.53. In each case, the two cameras will see the marks as shown in (a) and (b). Although the locations of the marks are different, the cameras will see them similarly. As a result, the marks may be located wrongly.

The accuracy of depth measurement in stereo imaging depends on the angle between the two images and thus the disparity between them. However, larger disparities require more searching over larger areas. To improve the accuracy and reduce computation time, multiple images of the same scene can be used [18]. A similar technique was employed in the Stanford Cart, wherein the navigation system would use a camera mounted on a shaft to take multiple images of the scene in order to calculate distances and find obstacles [23].

8.26.4 Scene Analysis with Shading and Sizes

Humans use the details contained in a scene to extract information about the locations of objects, their sizes, and their orientation. One of these details is the shading on different surfaces. Although the smoothly changing intensity of shades on surfaces is a source of difficulty in some other operations, such as segmentation, it can be indirectly used in extracting information about the depth and shape of objects. Shading is the relationship between the orientation of the object and the reflected

light. If this relationship is known, it can be used to derive information about the object's location and orientation. Depth measurement using shading requires a priori knowledge of the reflectance properties of the object and exact knowledge of the light source. As a result, its utility is limited and difficult.

Another source of information to be used for depth analysis is the texture gradient, or the changes caused in textures as a result of changes in depth. These variations are due to changes in the texture itself, which is assumed to be constant, changes in the depth or distance (scaling gradient), or changes in the orientation of the plane (called the foreshortening gradient). An example is the perceived change in the size of bricks on a wall. By calculating the gradient of the brick sizes on the wall, a depth can be estimated.

8.27 SPECIALIZED LIGHTING

Another possibility for depth measurement is to utilize special lighting techniques that will yield specific results, which can then be used to extract depth information. Most of these techniques have been designed for industrial applications where specialized lighting is possible and the environment is controlled. The theory behind the technique is that if a strip of light is projected over a flat surface, it will generate a straight line in relation to the relative positions and orientations of the plane and light source. However, if the plane is not flat and an observer looks at the light strip in a plane other than the plane of the light, a curved or broken line will be observed. (See Figure 8.54.) By analyzing the reflected light, we can extract information about the shape of the object, its location, and its orientation. In certain systems, two strips of light are used such that in the absence of any object on the table, the two strips will intersect exactly on the surface. When an object is present, the two strips

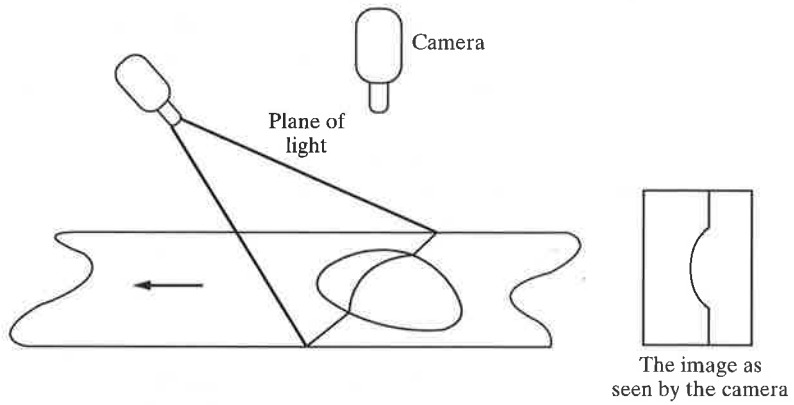


Figure 8.54 Application of a strip of light in depth measurement. A plane of light strikes the object. The camera, located at a different angle than that of the plane of light, will see the reflection of the light plane on the object as a curved line. The curvature of the line is used to calculate depth.

of light develop two reflections. The reflections are picked up by a camera, and a routine calculates the object's features and reports them. A commercial system based on this technique was developed by GM and is called CONSIGHT™. A disadvantage of the technique is that only information about the points that are lit can be extracted. Thus, in order to have information about the complete image, it is necessary to scan the entire object or scene.

8.28 IMAGE DATA COMPRESSION

Electronic images contain large amounts of information and thus require data transmission lines with a large bandwidth capacity. The requirements for the temporal and spatial resolution of an image, the number of images per second, and the number of gray levels (or colors for color images) are determined by the required quality of the images. Recent data transmission and storage techniques have significantly improved image transmission capabilities, including transmission over the Internet.

Although there are many different techniques of data compression, only some of them relate directly to vision systems. (The subject of data transmission in general is beyond the scope of this book and will not be discussed here.) Image data compression techniques are divided into intraframe (within-frame) and interframe (between-frame) methods.

8.28.1 Intraframe Spatial Domain Techniques

Pulse code modulation (PCM) is a popular technique of data transmission in which an analog signal is sampled, usually at the Nyquist rate (a rate that will prevent aliasing), and quantized. The quantizer will have N levels, where N is a power of 2. If N is 8, then 2^8 will yield a quantizer with 256 different gray levels (an eight-bit image quantizer). This arrangement is very common for television-type images and vision systems. Certain other applications (e.g., space and medical applications) use higher resolutions, such as 2^{10} or 2^{12}.

In a technique called *pusedorandom quantization dithering* [24], random noise is added to pixels' gray values to reduce the number of bits necessary to represent the image with the same quality as the original. If the number of bits in a quantizer is reduced without any dithering, contouring will take place. Due to the smaller number of available gray levels in the image, there will be contours of gray levels in the image that make it look like a topological map. (See Section 8.11 and Figure 8.14.) These contours can be broken up by adding a small amount of broadband psuedorandom, uniformly distributed noise, called dither, to the signal prior to sampling. The dither causes the pixel to oscillate about the original quantization level, removing the contours. In other words, the contours are forced to randomly make small oscillations about their average value. The proper amount of noise will enable the system to have the same resolution while the number of bits is reduced significantly.

Another technique of data compression is to use halftoning. In this technique, a pixel is effectively broken up into a number of pixels by increasing the number of samples per pixel. Instead, every sample is quantized by a simple one-bit binary quantizer into a black or a white pixel. Since the human eye will average the groups of pixels, the image will still look gray and not binary.

Predictive coding refers to a class of techniques that are based on the theory that, in highly repetitive images, only the new information (innovations) need be sampled, quantized, and transmitted. In these types of images, many pixels remain without change for many images. Thus, data transmission can be significantly reduced if only the changes between successive images are transmitted.

A predictor is used to predict an optimum value for each pixel, based on the information obtained from the previous images. The *innovation* is the difference between the actual value of the pixel and the predicted value. The predicted value is transmitted by the system to update the previous image. If, in an image, many pixels remain the same, innovations are few and transmission is reduced. In many situations, the images are highly repetitive, elements do not move fast, and large portions of any image, such as the background, do not change. Thus, predictive coding can be used effectively.

In an attempt to reduce the amount of data transmission by the *Voyager 2* spacecraft, its computers were reprogrammed to use a differential coding technique while the vehicle was in space. At the beginning of its journey into space, *Voyager*'s system was designed to transmit information about every pixel at a 256-gray-level scale. It took 5,120,000 bits to transmit one image, not including error detection and correction codes, which were about the same length. Beginning with the Uranus flyby, the system was reprogrammed to send only the difference between successive pixels, rather than the absolute brightness of the pixels. Thus, if there were no differences between successive pixels, no information would be transmitted. In scenes such as those in space, where the background is essentially black, many pixels are similar to their neighbors; hence, data transmission was reduced by about 60% [25]. Other examples of fixed background information include TV news sets, theatrical sets, and industrial images.

In *constant-area quantization,* or CAQ [26,27], data transmission is reduced by transmitting fewer pulses at lower resolution in low-contrast areas compared with high-contrast areas. This practice, in effect, takes advantage of the fact that higher contrast areas have higher frequency content and require more information transmission than lower contrast areas.

8.28.2 Interframe Coding Techniques

These methods take advantage of the redundant information that exists between successive images. The difference between interframe coding techniques and the intraframe methods is that, rather than using the information within one image, a number of different images are used to reduce the amount of information transmitted.

A simple technique to achieve this aim is to use a frame memory at the receiver. The frame memory will hold an image and will continually show it at the display. When information about any pixel is changed, the corresponding location in the frame

memory is updated. Thus, the rate of transmission is significantly reduced. The disadvantage of this technique is that there is flickering in the presence of rapidly moving elements.

8.29 REAL-TIME IMAGE PROCESSING

In many of the techniques considered so far, the image is digitized and stored before processing. In other situations, although the image is not stored, the processing routines require long computational times before they are finished. This means that, in general, there is a long lapse between the time an image is taken and the time a result is obtained. This may be acceptable in situations in which the decisions do not affect the process. However, in other situations, there is need for real-time processing such that the results are available in real time or in a short enough time to be considered real time. Two different approaches are considered for real-time processing. One is to design dedicated hardware such that the processing is fast enough to occur in real time [28]. The other is to try to increase the efficiency of both the hardware (or parts of it) and the software and thereby reduce processing and computational requirements such that the required times become shorter and closer to real times. In many situations, although a process does not truly happen in real time, compared with the speed of changes in the system and the decision time, the processing time is fast enough to be considered real time.

8.30 HEURISTICS

Heuristics are a collection of rules of thumb that are developed for semiintelligent systems in order to enable them to make decisions based on the current situation. Heuristics are used in conjunction with mobile robots, but have applications in many fields.

Consider a mobile robot that is supposed to navigate through a maze. Imagine that the robot starts at a point and is equipped with a sensor which alerts its controller that the robot has reached an obstacle such as a wall. At this point, the controller has to decide what to do next. Let's say that the first rule is that, when encountering an obstacle, the robot should turn left. As the robot continues, if it reaches another wall, it will turn left again and continue. Suppose that after three left turns the robot reaches the starting point. In this case, should it continue to turn left? Obviously, doing so will result in a never-ending loop. The second rule may be to turn right if the first point is encountered. Now imagine that after a left turn, the robot gets to a dead end. Then what? A third rule may be to trace back the path until an alternative route can be found. As you see, there are many different situations the robot may encounter. Each one of these situations must be considered by the designer, and a decision must be provided. The collection of these rules will be the heuristics rule base for the controller to "intelligently" decide how

to control the motions of the robot. However, it is important to realize that this intelligence is not a true intelligence, since the controller is not really making decisions, but merely selects from a set of decisions that have already been made. If a new situation is encountered that is not in the rule base, the controller will not know how to respond [29].

8.31 APPLICATIONS OF VISION SYSTEMS

Vision systems may be used in many different applications, sometimes in conjunction with robotic operations and robots. Vision systems are commonly used for operations that require information from the work environment and that include inspection, navigation, the identification of parts, assembly operations, and communication.

Suppose that in an automatic manufacturing setting, a circuit board is to be manufactured. One important part of this operation is the inspection of the board at different stages — before and after certain operations. A very common application for vision systems is to set up a cell wherein an image of the part to be inspected is taken. Subsequently, image-processing routines are used to modify, improve, and alter the image. Then the processed image is compared with an image from the memory. If there is a match, the part is accepted. Otherwise, the part either is rejected or is repaired. This image-processing-and-analysis operation generally is made up of the processes that were mentioned earlier. Most commercial vision systems have embedded routines that can be called from a macro, making it very easy to set up such a system.

In navigation, a scene is usually analyzed in order to find acceptable pathways, obstacles, and other elements that confront a robot. In some operations, the vision system sends its information to an operator, who controls the motions from a distance. This configuration is very common in telerobotics, as well as in space applications [30]. In some medical applications, too, the surgeon guides the device, be it a surgical robot or a small investigative, exploratory device that produces an angiogram through its operations. Autonomous navigation requires the integration of depth measurement with the vision system, either by stereo vision analysis or by range finders. It also requires heuristic rules of behavior for the robotic device to navigate around an environment.

In another application [31,32], an inexpensive laser diode was mounted next to a camera. The projected laser light was captured by the camera and was used to measure the depth of a scene, as well as to calibrate the camera. In both cases, due to the brightness of the laser light and bleeding effects, the image contained a large, bright circular spot. To identify the spot and separate it from the rest of the scene, histogram and thresholding operations were used. Subsequently, the circle was identified and then skeletonized until only its center remained. The location of the pixel representing the center of the circle was then used in a triangulation method to calculate the depth of the image or to calibrate the camera.

These simple examples are all related to what we have discussed. Although many other routines are available, fundamental knowledge about vision systems enables one to proceed with an application and adapt to it what vision systems have to offer.

8.32 DESIGN PROJECT

There are many inexpensive digital cameras on the market that can be used to create a simple vision system. These cameras are simple, small, and lightweight and provide a simple image that can be captured by computers and be used to develop a vision system. In fact, many cameras come with software to capture and digitize an image. Standard VHS video cameras can also be used in conjunction with products such as Snappy™, which allows you to digitize and save images in your computer. You may also use a standard digital camera to capture an image and to download the image into your computer. In this case, although you can capture an image for later analysis, due to the additional steps involved in downloading the image from the camera to the computer, the image is not available for immediate use.

There are many simple programs, such as Adobe Photoshop™, that have many routines similar to those we have discussed in this chapter. Additional routines may be developed using common computer languages, such as C. The final product will be a simple vision system with some vision capability that can be used to perform vision-related tasks. This development may be done independently or in conjunction with a three-axis robot and can include routines for identifying and picking up parts, developing mobile robots, and creating other, similar devices.

All images shown in this chapter were captured and processed by the vision systems including MVS909™ and Optimas™ 6.2 vision systems, in the Mechanical Engineering Robotics laboratory at Cal Poly, San Luis Obispo, CA. You may also develop your own simple vision system using other programming languages and systems that can handle an image file. Among these systems are Excel™, LabView™, and other development systems.

8.33 SUMMARY

In this chapter, we studied the fundamentals of capturing an image, image processing to modify, alter, improve, or enhance an image, and image analysis through which data can be extracted from an image for subsequent application. Vision systems may be used for a variety of applications, including manufacturing, surveillance, navigation, and robotics. Vision systems are flexible, inexpensive, powerful tools that can be used with ease.

There are countless different routines that can be used for variety of purposes. Most of these routines are created for specific operations and applications. However, certain fundamental techniques, such as convolution masks, can be applied to many classes of routines. We have concentrated on these techniques, which enable you to adopt, develop, and use other routines and techniques for other applications. The advances in technology have created tremendous opportunities for vision system and vision analysis. There is no doubt that the trend will continue into the future.

REFERENCES

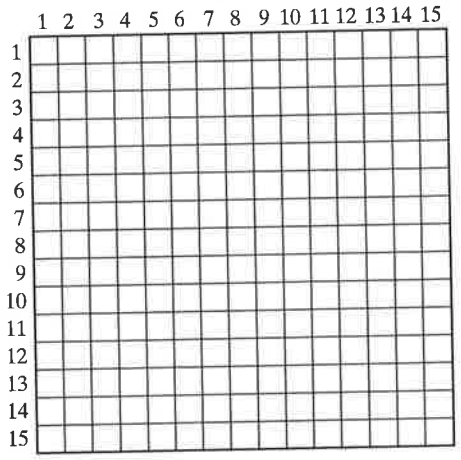
1. Madonick, N., "Improved CCDs for Industrial Video," *Machine Design,* April 1982, pp. 167–172.
2. Wilson, A., "Solid-State Camera Design and Application," *Machine Design,* April 1984, pp. 38–46.
3. "A 640 × 486 Long-Wavelength Infrared Camera," *NASA Tech Briefs,* June 1999, pp. 44–47.
4. Meagher, J., *Fourier.xle Program,* Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo, CA, 1999.
5. Doudoumopoulos, Roger, "On-Chip Correction for Defective Pixels in an Image Sensor," *NASA Tech Briefs,* May 2000, p. 34.
6. Gonzalez, R. C., Richard Woods, *Digital Image Processing,* Addison-Wesley, 1992.
7. Low, Adrian, *Introductory Computer Vision and Image Processing,* McGraw-Hill, 1991.
8. Horn, B.K.P., *Robot Vision,* McGraw-Hill, 1986.
9. Hildreth, Ellen, "Edge Detection for Computer Vision System," *Mechanical Engineering,* August 1982, pp. 48–53.
10. Olson, Clark, "Image Smoothing and Edge Detection Guided by Stereoscopy," *NASA Tech Briefs,* September 1999, pp. 68–69.
11. Groover, M. P., et al., *Industrial Robotics, Technology, Programming, and Applications,* McGraw-Hill, 1986, p. 177.
12. Hough, P. V. C., *A Method and Means for Recognizing Complex Patterns,* U.S. Patent 3,069,654, 1962.
13. Illingworth, J., J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing,* Vol. 44, 1988, pp. 87–116.
14. Kanade, T., "Survey; Region Segmentation: Signal vs. Semantics," *Computer Graphics and Image Processing,* Vol. 13, 1980, pp. 279–297.
15. Snyder, Wesley, *Industrial Robots: Computer Interfacing and Control,* Prentice Hall, 1985.
16. Gonzalez, Rafael, P. Wintz, *Digital Image Processing,* 2d ed., Addison-Wesley, Reading, MA, 1987.
17. Liou, S. P., R. C. Jain, "Road Following Using Vanishing Points," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1986, pp. 41–46.
18. Nevatia, R., *Machine Perception,* Prentice-Hall, 1982.
19. Fu, K. S., Gonzalez, R. C., Lee, C. S. G., *Robotics; Control, Sensing, Vision, and Intelligence,* McGraw-Hill, 1987.
20. Marr, D., T. Poggio, "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society, London,* B204, 1979, pp. 301–328.
21. Marr, D., *Vision,* Freeman and Co., 1982.
22. Pipitone, Frank, T. G. Marshall, "A Wide-field Scanning Triangulation Rangefinder for

Machine Vision," *International Journal of Robotics Research,* Vol. 2, No. 1, Spring 1983, pp. 39–49.

23. Moravec, H. P., "Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover," *Stanford Artificial Intelligence Laboratory Memo,* AIM-340, Sep. 1980.
24. Thompson, J. E., "A 36-Mbit/s Television Coder Employing Psuedorandom Quantization," *IEEE Transactions on Communication Technology,* COM-19, No. 6, December 1971, pp. 872–879.
25. Goldstein, Gina, "Engineering the Ultimate Image, The *Voyager 2* Mission," *Mechanical Engineering,* December 1989, pp. 30–36.
26. Pearson, J. J., R. M. Simonds, "Adaptive, Hybrid, and Multi-Threshold CAQ Algorithms," *Proceedings of SPIE Conference on Advanced Image Transmission Technology,* Vol. 87, August 1976, pp. 19–23.
27. Arnold, J. F., M. C. Cavenor, "Improvements to the CAQ Bandwidth Compression Scheme," *IEEE Transactions on Communications,* COM-29, No. 12, December 1981, pp. 1818–1822.
28. McHugh, Peter, "Vision and Manufacturing," *NASA Tech Briefs,* June 1999, pp. 36, 37.
29. Chattergy, R., "Some Heuristics for the Navigation of a Robot," *International Journal of Robotics Research,* Vol. 4, No. 1, Spring 1985, pp. 59–66.
30. Ashley, Steven, associate editor, "Roving Other Worlds by Remote," *Mechanical Engineering,* July 1997, pp. 74–76.
31. Niku, S. B., "Active Distance Measurement and Mapping Using Non Stereo Vision Systems," *Proceedings of Automation '94 Conference,* July 1994, Taipei, Taiwan, R.O.C., Vol. 5, pp. 147–150.
32. Niku, S. B., "Camera Calibration and Resetting with Laser Light," *Proceedings of the Third International Conference on Mechatronics and Machine Vision in Practice,* September 1996, Guimarães, Portugal, Vol. 2, pp. 223–226.

PROBLEMS

If you do not have access to an image, simulate the image by creating a file called $I_{m,n}$, where m and n are the row and column indices of the image. Then, using the following image matrix, create an image by substituting 0's and 1's or gray-level numbers in the file:



In a binary image, the 0's represent "off," dark, or background pixels, while 1's represent "on," light, or object pixels. In gray images, each pixel is represented by a corresponding grayness value. A computer routine can then be written to access this file for image data. The result of each operation can be written to a new file, such as $R_{m,n}$, where R represents the result of the operation and m and n are the row and column indices, respectively, of the resulting file.

Alternatively, you may use your own graphics system or any commercially available graphics language to create, access, and represent an image.

1. Write a computer program for the application of a 3×3 averaging convolution mask onto a 15×15 image.
2. Write a computer program for the application of a 5×5 averaging convolution mask onto a 15×15 image.
3. Write a computer program for the application of a 3×3 high-pass convolution mask onto a 15×15 image for edge detection.
4. Write a computer program for the application of an $n \times n$ convolution mask onto a $k \times k$ image. Write the routine such that the user can choose the size of the mask and the values of each mask cell individually.
5. Write a computer program that will perform the left–right search routine for a 15×15 image.
6. Using the left-right search technique, find the outer edge of the object shown in Figure P.8.6.

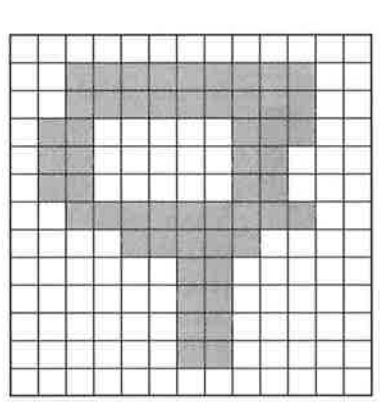
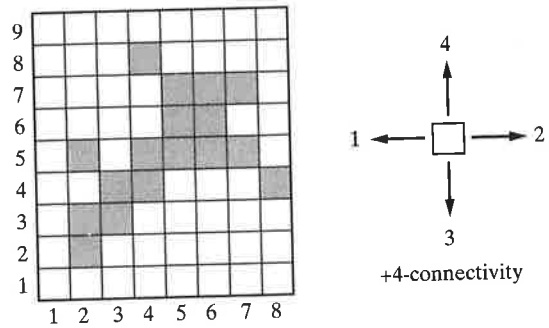


Figure P.8.6

7. Write a computer program that will perform a region-growing operation based on +4-connectivity. The routine should start at the 1,1 corner pixel, search for a nucleus, grow a region with a chosen index number, and, after finishing that region, continue searching for another nucleus until all object pixels have been checked.
8. Using +4-connectivity logic and starting from the pixel 1,1, write the sequence of pixels, in the correct order, that will be detected by a region-growing routine for the image shown in Figure P.8.8.



+4-connectivity

Figure P.8.8.

9. Write a computer program in which different moments of an object in an image can be calculated. The program should query you for moment indices. The results may be reported to you in a new file or may be stored in memory.
10. For the 10×10 binary image of the key shown in Figure P.8.10, calculate the following:
    - Perimeter, based on the left–right search technique.
    - Thinness, based on $\frac{P^2}{\text{Area}}$.
    - Center of gravity.
    - Moment $M_{0,1}$ about the origin (pixel 1,1) and about the lowest pixel of a rectangular box around the key.

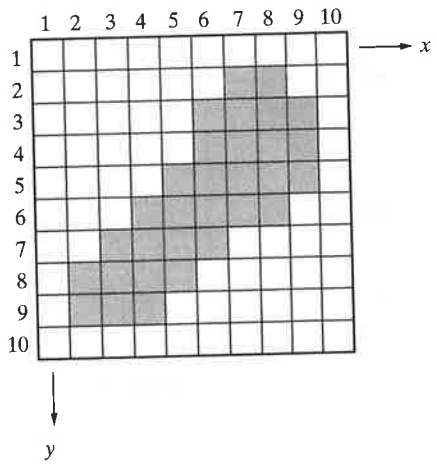


Figure P.8.10.

9

Fuzzy Logic Control

9.1 INTRODUCTION

Consider the following statement: Tuesday, October 26 was supposed to be a very warm day in San Luis Obispo, and in fact it turned out to be pretty hot. When the robotics lab was opened in the morning, we found out that the steam line had leaked into the room, and much heat and humidity had been released into the environment. When the hydraulic power unit for the robots was turned on, it added even more heat to the lab, raising the temperature even further. Eventually, it got so hot that we had to bring in large fans to cool down the lab a bit to make it a little more comfortable for students.

This true statement is a very good example of what fuzzy logic is about. Let's look at the statement again, noticing the italicized words:

Tuesday, October 26 was supposed to be a *very warm* day in San Luis Obispo, and in fact it turned out to be *pretty hot.* When the robotics lab was opened in the morning, we found out that the steam line had leaked into the room, and *much* heat and humidity had been released into the environment. When the hydraulic power unit for the robots was turned on, it added even more heat to the lab, raising the temperature even *further.* Eventually, it got *so hot* that we had to bring in *large* fans to cool down the lab *a bit* to make it *a little more* comfortable for students.

As you see, in this statement, there are a number of "descriptors" used to state certain conditions that are not very clear. For example, when we state that the day was supposed to be very warm, what do you think it was supposed to be? 85°F? Or maybe 100°F? In fact, if you live in San Luis Obispo, even 80°F may be a warm day. Then, as you can see, this description of the temperature is in fact fuzzy. It is not very clear what the temperature may be. As you go on, the statement continues to be fuzzy. We also don't know exactly what is meant by so hot, or a bit, or large fans. How large? How much cooler did the temperature get when we turned on the fans? Now I suggest that you read the paragraph you are reading once again and see how