CHAPTER 9

Image Processing and Analysis with Vision Systems

9.1 Introduction

A very large body of work is associated with vision systems, image processing, and pattern recognition that addresses many different hardware and software related topics on these subjects. 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 rapidly growing. The enormous number of papers published every year indicates that many useful techniques constantly appear in the literature. At the same time, it also means that many of these techniques may be suitable for certain applications only. 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. This chapter does not intend to be a complete survey of all possible vision routines, but only an introduction. If interested, it is recommended that you continue studying the subject through other references.

9.2 Basic Concepts

The following sections include some fundamental definitions of terms and basic concepts that we will use throughout the chapter.

9.2.1 Image Processing versus Image Analysis

Image processing relates to the preparation of an image for later analysis and use. Images, as captured by a camera or other similar techniques (such as a scanner), are not necessarily in a form that can be used by image analysis routines. Some may need improvement to

reduce noise; some may need to be simplified; others may need to be enhanced, altered, segmented, filtered, and so on. Image processing is the collection of routines and techniques that improve, simplify, enhance, and otherwise alter an image.

Image analysis is the collection of processes by which a captured and processed image is analyzed to extract information about the content and to identify objects or other related facts about the objects within the image or the environment.

9.2.2 Two- and Three-Dimensional Image Types

Although all scenes are three-dimensional, images can either be two-dimensional (lacking depth information) or three-dimensional (containing depth information). Most images with which we normally deal, obtained by cameras, are two-dimensional. However, other systems such as Computed Tomography (CT) and CAT-scans create three-dimensional images that contain depth information. Therefore, these images can be rotated about different axes in order to better visualize the depth information. A two-dimensional image is extremely useful for many applications even though it has no depth information. This includes feature extraction, inspection, navigation, parts handling, and many more.

Three-dimensional images are used with applications 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 applications. For three-dimensional images, either X-rays or ultrasonics are used to get images of one slice of the object at a time; later, all images are put together to create a three-dimensional image representation of the internal characteristics of the object.

All three-dimensional vision systems share the problem of coping with many-to-one mapping 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., underwater operations), the systems are not very accurate and require additional processing of the information, and therefore, operate at lower speeds.

9.2.3 The Nature of an Image

An image is a representation of a real scene, either in black and white (B/W) or in color, and either in print or in digital form. Printed images may be reproduced either by multiple colors and gray scales, such as CMYK in color print or halftone black and white print, or by a single ink source. For example, to reproduce a photograph with real halftones, we use multiple gray inks, which when combined, produce a somewhat realistic image. However, in most print applications, only one ink color is available (such as black ink on white paper in a newspaper or copier). In this 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 into small sections. In each section, if the ink portion of the section is smaller than the white, the section will look a lighter gray

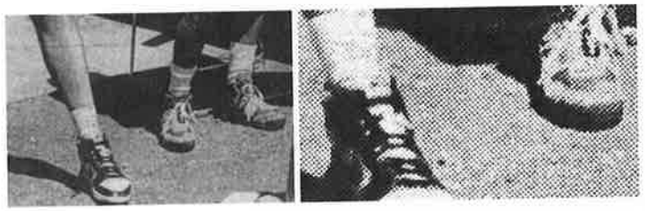


Figure 9.1 Examples of how gray intensities are created in printed images. In print, only one color ink is used, while the ratio of black to white area of the pixel is changed to create different gray levels.

(Figure 9.1). If the black ink area is larger than the white area, it will look a 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.

Similar to printed images, electronic and digital images are also divided into small sections called picture cells, or *pixels* (in three-dimensional images, it is called volume cell or *voxel*), where the size of all pixels is the same. To capture an image, the intensity of each pixel is measured and recorded; similarly, to recreate an image, the intensity of light at each pixel location is varied. Therefore, an image file is the collection of the data representing the light intensities of a large number of pixels. This file can be recreated, processed, modified, or analyzed. A color image is essentially the same, except that the original image is separated into three images of red, green, and blue before capturing and digitization. When the three colors with different intensities at each pixel location are superimposed, color images are recreated.

9.2.4 Acquisition of Images

There are two types of vision cameras: analog and digital. Analog cameras are no longer common but are still around and used to be the standard camera at television stations. Digital cameras are the current standard and are practically all the same. A digital moving-picture camera is similar to a digital still-camera plus a recording section; otherwise, the mechanism of image acquisition is the same. Whether the captured image is analog or digital, in vision systems, the image is eventually digitized. In digital systems, all data is in binary form and is stored in a computer file or memory device. Therefore, we ultimately deal with a file of numbers 0 and 1, from which we extract information and make decisions.

Appendix B presents a short discussion about analog and digital cameras and how the image is captured. The final outcome of these systems is a file that contains sequential pixel-location and pixel-intensity data that we use in our discussions. Refer to Appendix B for understanding the fundamental issues about image acquisition. For more information about details of these systems, refer to digital data acquisition references.

9.2.5 Digital Images

The light intensities at each pixel location are measured and converted to digital form regardless of the type of camera or image acquisition system. The data is either stored in

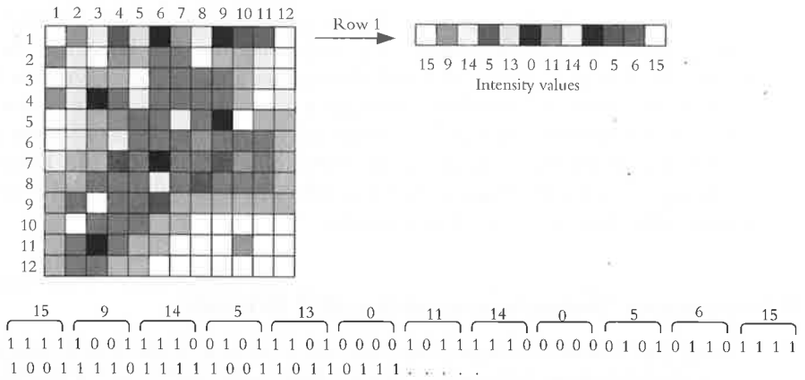


Figure 9.2 An image and the binary representation of its first row using 4 bits per pixel.

memory, in a file, or in recording devices with an image format such as TIFF, JPG, Bitmap, and so on, or is displayed on a monitor. Since it is digitized, the stored information is a collection of 0s and 1s 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 0s and 1s, sequentially stored to represent the intensity of light at each pixel. However, these files can be accessed and read by a program, duplicated and manipulated, or rewritten in a different form. Vision routines generally access this information and perform some function on the data and either display the result or store the manipulated result in a new file. A fundamental issue is to be able to extract information or manipulate this collection of 0 and 1 values in a meaningful way.

To understand this better, consider the simple low resolution image in Figure 9.2. Each pixel is referred to by row and column numbers. Assuming the system is digitized with only 4 bits (we will discuss this further shortly), there will be up to $2^4 = 16$ distinct light intensities possible. The sequence of 0 and 1 numbers representing the first row of the image will look as shown in the figure (all with only 4 bits per pixel). Different file formats list these numbers differently. In a simple Portable Gray Map (PGM) format, the intensities are listed sequentially as shown. A header at the beginning of the file will indicate the number of pixels in each row and column (in this case, it is 12×12). The program knows that every 4 bits is 1 pixel. Therefore, it can access each pixel intensity directly. However, as you notice, the file is reduced to a string of 0 and 1 values, to which, image processing routines are applied, and from which, information is extracted. Now imagine the size of the string of 0 and 1 values that represents a large image (sometimes in mega-pixel range) at up to 24 bits per pixel, for three primary colors.

An image with different gray levels at each pixel location is called a gray image. A color image results by superimposing three images of red, green, and blue hues (RGB), each with a varying intensity and each equivalent to a gray image (but in one of the three hues). Therefore, when the image is digitized, it will similarly have strings of 0s and 1s for each hue (an alternative way is to assign a number to each color, all declared in a header at the beginning of the image file. Then the number representing the pixel represents the color reference and intensity). A binary image is an image where each pixel is either fully light or fully dark, either a 0 or a 1. To achieve a binary image, in most cases, a gray image is

converted using the histogram of the image and a cut-off value called a threshold. A histogram of the pixel gray levels will determine the distribution of the different gray levels. We can pick a value that best determines a cut-off level with the least distortion and use this value as a threshold to assign a 0 (or off) to all pixels whose gray levels are below the threshold value and to assign a 1 (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 it can be processed much faster than gray or colored images.

9.2.6 Frequency Domain versus Spatial Domain

Many processes used in image processing and analysis are either based on frequency domain or on 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. Both techniques are equally important and powerful and are used for different purposes. It should be noted here that although spatial and frequency domain techniques are used differently, they are both related. For example, suppose a spatial filter is used to reduce noise in an image. As a result of this filter, 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 this reduction in noise.

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

9.3 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 collection of sines and cosines of different amplitudes and frequencies, called Fourier series, as follows:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t \qquad (9.1)$$

When these sines and cosines are added together, the original signal is reconstructed. This conversion to frequency domain is called Fourier series, and the collection of different frequencies present in the equation is called *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 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 only of one frequency and a constant amplitude, the frequency spectrum representing it consists of a single value at the given frequency, as shown in Figure 9.3. Obviously, if we plot the function represented by the arrow in Figure 9.3(b) with the given frequency and amplitude, we will have the same sine function reconstructed. The plots in Figure 9.4 are similar and represent $f(t) = \sum_{n=1,3\cdots 15} \frac{1}{n} \sin(nt)$. The frequencies are

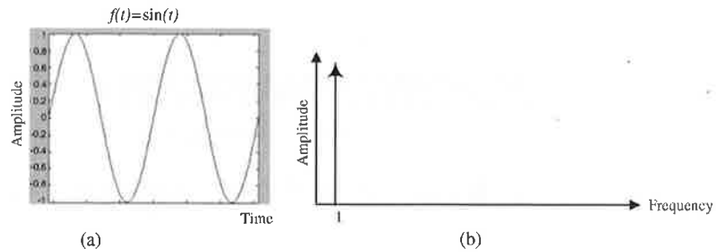


Figure 9.3 Time domain and frequency domain plots of a simple sine function.

also plotted in the frequency-amplitude domain. As you can see, when the number of frequencies contained in $f(t)$ increases, the summation gets closer to a square function.

Figure 9.5(a) shows a signal from a sensor and its frequency content. Although the signal is not a true sine function, the dominant frequency is 0.75 Hz. However, due to these discrepancies and the variations in the signal, the frequency spectrum contains many

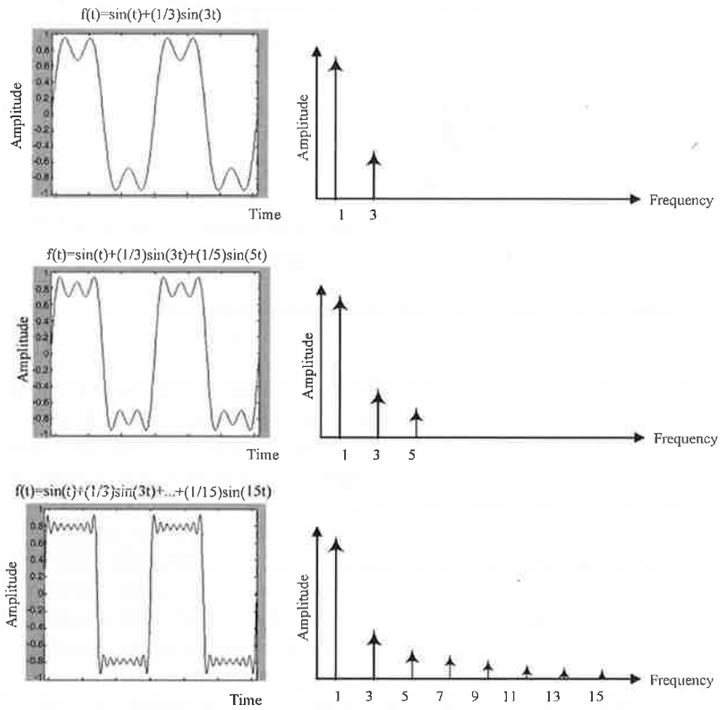


Figure 9.4 Sine functions in time and frequency domains for a successive set of frequencies. As the number of frequencies increases, the resulting signal gets closer to a square function.

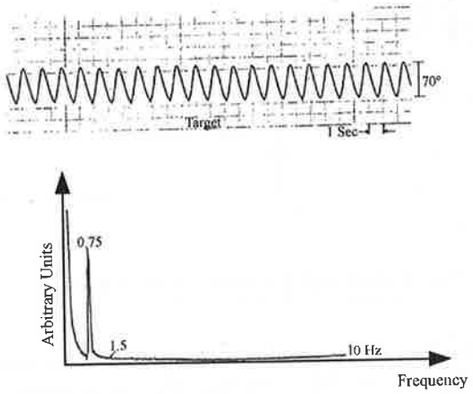


Figure 9.5 Two signals and their frequency spectrums.

other frequencies. Figure 9.5(b) shows a signal with more frequent variations and its frequency spectrum. Clearly, many more sine and cosine functions must be added in order to reconstruct this signal; therefore, the spectrum contains many more frequencies.

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, square wave, or other similar functions) decompose into a large number of frequencies. The sharper the change, the higher the number of frequencies needed to reconstruct it. Therefore, any video or other signal that contains sharp changes (noise, edges, high contrasts, impulse, step function) or has 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 made on nonrepeating signals too (called Fourier Transform, and particularly, *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 there will be some major frequencies within the spectrum with larger amplitudes called *harmonics*. These major frequencies or harmonics are used in identifying and labeling a signal, including recognizing voices, shapes, objects, and the like.

9.4 Frequency Content of an Image; Noise, Edges

Figure 9.6 shows a low resolution artificial image and a graph of its pixel intensities versus their positions. A representation such as this may be obtained when an image is scanned by an analog camera or a frame grabber is used with a digital system to sample and hold the data (see Appendix B). The graph is a discrete representation of varying amplitudes showing the intensity of light at each pixel (or versus time). Let's say we are on the 9th row and are looking at pixel numbers 129–144. The intensity of pixel number 136 is very different from the ones around it and may be considered noise, which is generally 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, and therefore, can be construed as an edge of the object.

Although this is a discrete (digitized) signal, as discussed earlier, it may be transformed into a large number of sines and cosines with different amplitudes and frequencies which, if added, will reconstruct the signal. As discussed earlier, portions of the signal that change slowly, such as small changes between succeeding pixel gray values, will require fewer sines and cosines to be reconstructed, and consequently, contribute more low frequencies to the spectrum. On the other hand, parts of the signal that vary quickly or significantly, such as large differences between pixel gray levels, require a large number of higher frequencies to be reconstructed and, as a result, contribute more high frequencies to the spectrum. Both noises and edges are among cases where one pixel value is substantially different from the neighboring pixels. Therefore, noises and edges contribute to the higher 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.

If a high frequency signal is passed through a low-pass filter—a filter that allows lower frequencies through without much attenuation in amplitude, but which severely

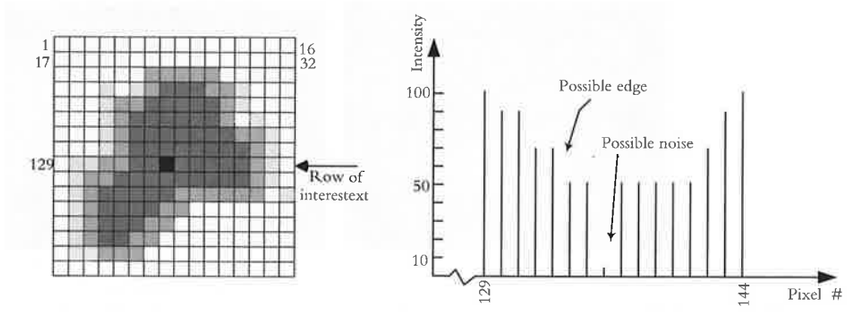


Figure 9.6 Noise and edge information in an intensity diagram of an image. The pixels with intensities much different from the neighboring pixels can be considered edges or noise.

attenuates the amplitudes of the higher frequencies in the signal—it will reduce the influence of all high frequencies, including the noises and edges. This means that although a low-pass filter reduces noise, it also reduces the clarity of an image by attenuating the edges and 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, noise and edges will be left alone, but slowly changing areas will disappear from the image. The application of different methods for noise reduction and edge detection will be discussed further in later sections of this chapter.

9.5 Resolution and Quantization

Two measures significantly affect the usefulness of an image and the data contained within it. The first one is resolution, which is affected by how often a signal is measured and read or sampled. Higher numbers of samples at equally spaced periodic times result in higher resolution, and therefore, more data. The resolution of an analog signal is a function of *sampling rate*. The resolution of a digital system is a function of how many pixels are present. Fundamentally, these two are the same measure; reading the light intensity of the image at more pixel locations is the same as sampling more often. Figure 9.7 shows an image sampled at (a) 432×576, (b) 108×144, (c) 54×72, and (d) 27×36 pixels. The clarity of the image is lost when the sampling rate decreases.

The second issue is how accurately the value of the signal at any given point is converted to digital form. This is called *quantization*—a function of how many bits are

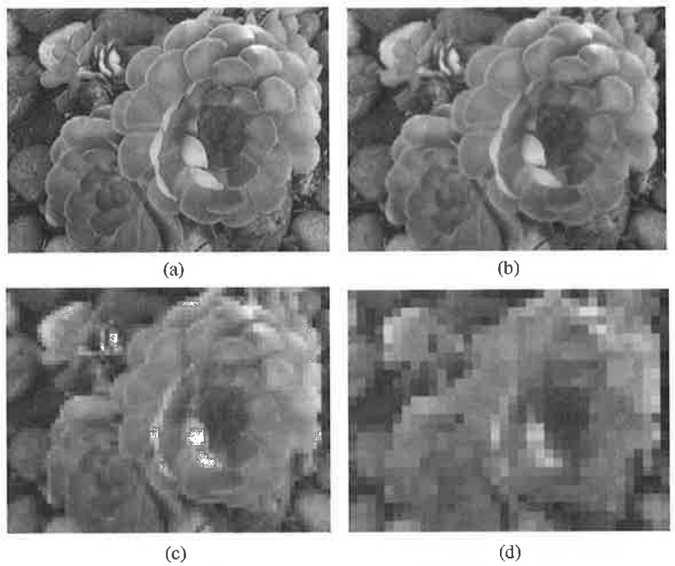


Figure 9.7 Effect of different sampling rates on an image at (a) 432 × 576, (b) 108 × 144, (c) 54 × 72, and (d) 27 ×36 pixels. As the resolution decreases, the clarity of the image decreases accordingly.

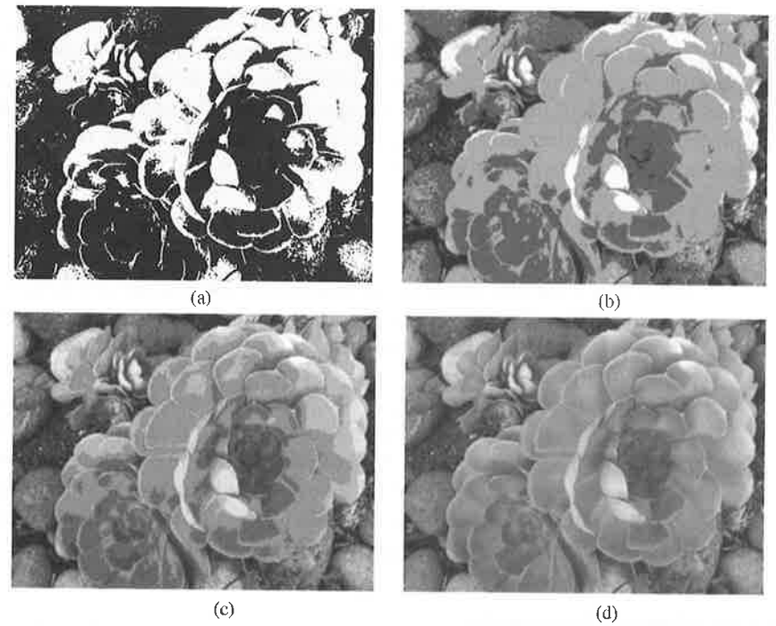


Figure 9.8 An image at different quantization levels of 2, 4, 8, and 44 gray levels. As the quantization resolution increases, the image becomes smoother.

used to represent the digitized magnitude of the sampled signal. Depending on the number of bits used for quantization, the grayness variations 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, on or off, or 0 or 1 (called a binary image). For quantization with an 8-bit ADC, the maximum number of gray levels will be 256. Therefore, the image will have 256 different gray levels (0–255).

Quantization and resolution are completely independent of each other. For example, a high resolution image may be converted into a binary image, where there are only on and off pixels (0 and 1, or dark and light), or the same image may be quantized into 8 bits, which can yield a spectrum of 256 different shades of gray. Figure 9.8 shows the same image quantized at (a) 2 levels, (b) 4 levels, (c) 8 levels, and (d) the original at 44 levels.

Both the resolution and quantization must be sufficiently high in order to provide adequate information for a specific task. A low-resolution image may not be adequate for recognition of parts with high detail, but enough for distinguishing between a bolt and a nut. Low bit-count quantization may be enough for many applications where binary images are adequate, but not in others where different objects must be distinguished from each other. For example, a high-resolution image is necessary for reading the license plate of a car or recognition of faces with a security camera. However, because the license plate consists of primarily dark letters on a light background, even a binary image (only one bit per pixel) may be sufficient. A similar image must be quantized at a higher bit-count in order to allow face recognition. When choosing a camera, both these values must be considered.

The sampled light at a pixel, when quantized, yields a string of 0s and 1s 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 (pixels) and the memory needed for each digitized sample. A larger image with higher resolution (total number of pixels) and a higher number of gray levels requires a larger memory size. The total memory requirement is a function of both values.

Example 9.1

Consider an image with 256 by 256 pixels. The total number of pixels in the image is $256 \times 256 = 65{,}536$. If the image is binary, it will require one bit to record each pixel as 0 or 1. Therefore, the total memory needed to record the image will be 65,536 bits, or with 8 bits to a byte, it will require 8192 bytes. If each pixel were to be digitized at the rate of 8 bits for 256 shades of gray, it would require $65{,}536 \times 8 = 524{,}288$ bits, or 65,536 bytes. For a video clip, changing at the rate of 30 images per second, the memory requirement will be $65{,}536 \times 30 = 1{,}966{,}080$ 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. The actual memory requirement may be less depending on the format in which the image is saved. ■

9.6 Sampling Theorem

Can you tell from Figure 9.9 what the image represents? Of course, since this is a very low-resolution 16×16 image, it is difficult to guess what the object is. This simple illustration signifies the relationship between sampling rate and the information obtained from it. To understand this, we will discuss some fundamental issues about sampling.

Consider a simple sinusoidal signal with frequency f as shown in Figure 9.10. Suppose the signal is sampled at the rate of f_s. The arrows in 9.10(b) show the corresponding sampled amplitudes.

Now suppose we want to use the sampled data to reconstruct the signal. This would be similar to sampling a sound source such as a CD and 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 can see in Figure 9.11, it is very

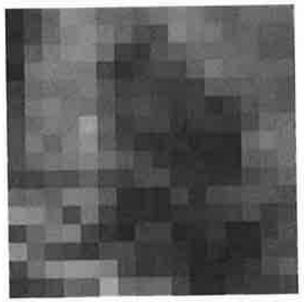


Figure 9.9 A low-resolution (16×16) image.

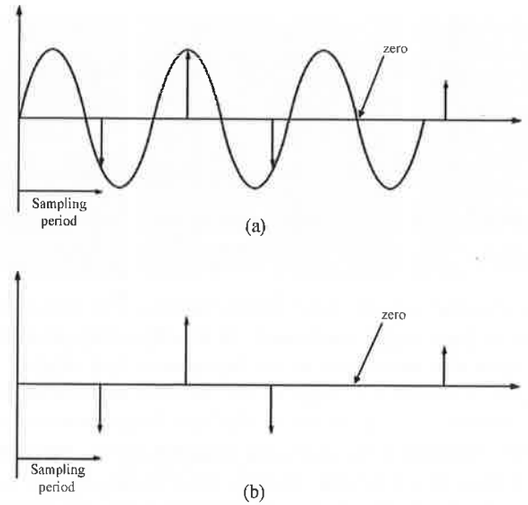


Figure 9.10 (a) Sinusoidal signal with a frequency of f, (b) sampled amplitudes at the rate of f_s.

possible that another signal may be reconstructed from the same data that is completely different from the original signal but yields the same sampled data. Both are valid, and many other signals can be valid and might be reconstructed from this sampled data too. This loss of information is called *aliasing* of the sampled data, and it can be a very serious problem.

In order to prevent aliasing, according to what is called *sampling theorem*, the sampling frequency must be at least twice as large as the largest frequency present in the signal. In that case, we can reconstruct the original signal without aliasing. The highest frequency present in the signal can be determined from the frequency spectrum of the signal. If a signal's frequency spectrum is found using the Fourier transform, it will contain many frequencies. However, as we have seen, the higher frequencies have smaller amplitudes. We 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

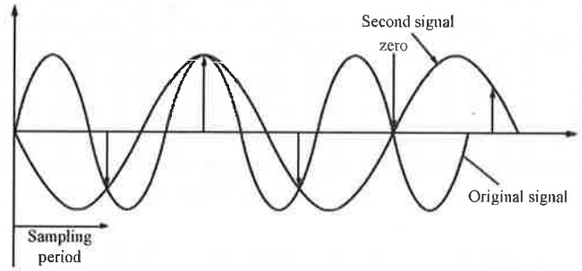


Figure 9.11 Reconstruction of signals from the sampled data. More than one signal may be reconstructed from the same sampled data.

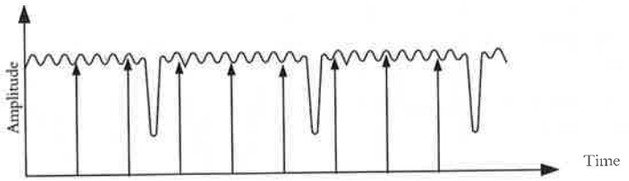
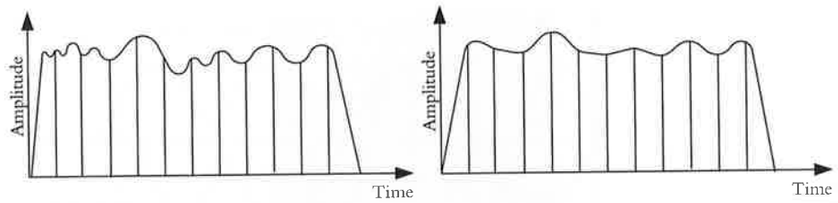


Figure 9.12 An inappropriate sampling rate may completely miss important data within a signal.

much effect in the system's total representation. The sampling rate of the signal must be at least twice as large as this frequency. In practice, the sampling rate is generally chosen to be larger than this minimum to further ensure that aliasing of the signal will not occur. Frequencies 4–5 times as large as the desired maximum frequency are common. For example, human ears can theoretically hear frequencies up to about 20,000 Hz. If a CD player is 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. In reality, if a signal changes more quickly than the sampling rate, the details of the change will be missed, and therefore, the sampled data will be inadequate. For example, it has been shown that the resulting vibration from a rotating gear with a broken tooth is distinctly different from a regular gear (Figure 9.12). However, at a low sampling rate, the sampled data may be completely void of this important information. Similarly, in Figure 9.13, the sampling rate is lower than the higher frequencies of the signal. As shown, although the lower frequencies of the signal are reconstructed, the signal will not have the higher frequencies of the original signal. The same is true with sound and image signals. If a sound signal is sampled at a low rate, the high frequency information will be lacking and the reconstructed sound will lack high frequency sounds. The output of the system, even if the best speakers are used, will be distorted and different from the real signal.

For images too, if the sampling rate is low, translating into a low resolution image, the sampled data may not have all the necessary detail; the information in the image is lost, and the image cannot be reconstructed like the original image. Figure 9.9 is sampled at a very low rate, and the information in it is lost. This is why you cannot decipher the



(a) (b)

Figure 9.13 The original signal in (a) is sampled at a sampling rate 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 9.14 The image of Figure 9.9, presented at higher resolutions of (a) 32 × 32, (b) 64 × 64, (c) 256 × 256.

image. However, when the sampling rate is increased, there will be a time when there is enough information to recognize the image. The still higher resolutions or sampling rates will transfer more information, and therefore, increasingly more detail can be recognized. Figure 9.14 is the same image as in Figure 9.9, but at 2, 4, and 16 times higher resolutions. Now suppose you need to recognize the difference between a bolt and a nut in a vision system in order to direct a robot to pick up the parts. Because the information representing a bolt and a nut is very different, a low-resolution image still enables you to determine what the part is. However, in order to recognize the license plate number of a car while moving in traffic, a high-resolution image is needed to extract enough information about the details such as the numbers on the license plate.

9.7 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 an image. Instead, the intention is to remove faults, trivial information, or information that may be important but not useful to improve the image. As an example, suppose 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.) could be determined. Also consider an image corrupted by reflections due to direct lighting 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 the image of a section of a city fully detailed with streets, cars, shadows, and the like. 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 sections, including histogram analysis, thresholding, masking, edge detection, segmentation, region growing, modeling, and many more. In the next sections, we will study some of these techniques and their applications.

9.8 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 for converting the image into binary form. It can also be used to decide if 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. A histogram can be used to determine the noisy gray level in order to attempt to remove or neutralize the noise. The same may be used to separate an object from the background so long as they have distinctly different colors or gray values.

Figure 9.15(a) shows a low-contrast image that has all its pixel gray levels clustered between two relatively close values. In this image, all pixel gray values are between 120 to 180 gray levels, at intervals of 4 (the image is quantized at 16 distinct levels between 0 to 256). Figure 9.15(c) shows the histogram of this image and, as you see, all pixel gray levels are between 120 to 180, a relatively low range. As a result, the image is not very clear and details are not visible. Now suppose we equalize the histogram such that the same 16 gray levels present in the image are spread out between 0 to 255 gray levels at intervals of 17, instead of the present 120–180 at intervals of 4. As a result of this histogram equalization, the image is vastly improved, as shown in Figure 9.15(b), with its corresponding histogram in (d). Notice that the number of pixels at each gray level are exactly the same in both cases, but the gray levels are spread out. The grayness values are given in Table 9.1.

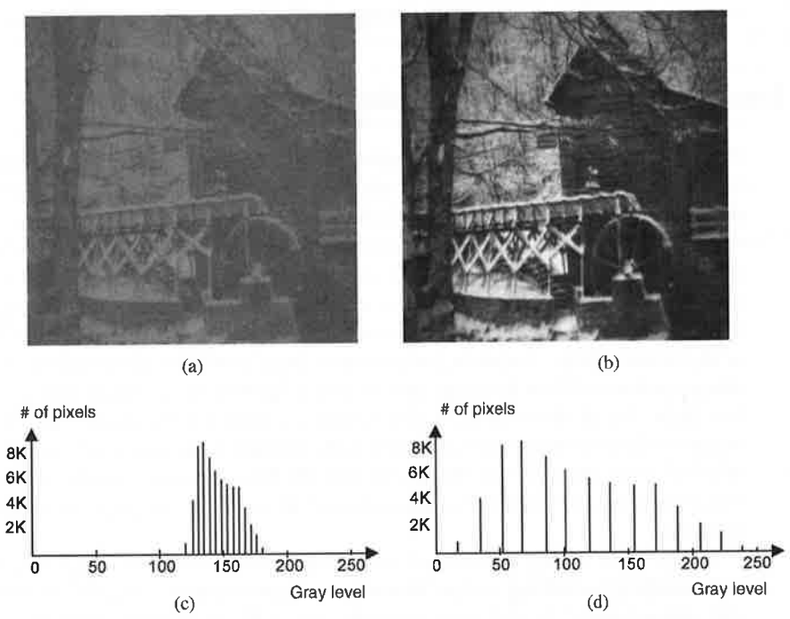


Figure 9.15 Effect of histogram equalization in improving an image.

Table 9.1 *The Actual Grayness Values and # of Pixels for Images in Figure 9.15(a) and 9.15(b).*

Levels	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of Pixels	0	750	5223	8147	8584	7769	6419	5839	5392	5179	5185	3451	2078	1692	341	0
For (b)	0	17	34	51	68	85	102	119	136	153	170	187	204	221	238	256
For (a)	120	124	128	132	136	140	144	148	152	156	160	164	168	172	176	180

Example 9.2

Assume the histogram of an image is spread between 100 and 150 out of the maximum grayness level of 255. What is the effect of multiplying the range by 1.5 or by 2? What is the effect of adding 50 to all gray values?

Solution: The two operations mentioned here are common in formatting images and in many vision systems. When all gray values are increased by the same amount, the image becomes brighter but the contrast does not change. So long as the added value does not increase the greyness level of any pixel beyond the 255 level, no information is lost and the original image may be regained by decreasing all pixel values by the same amount.

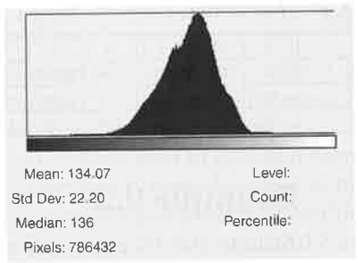
If the pixel greyness levels are multiplied by a number, so long as the maximum available gray levels are not exceeded, the histogram range is extended and contrast is increased. In this example, since the range is between 100 and 150, equalizing the histogram by 1.5 increases the range to 150 and 225. However, increasing the value to 2 will extend the histogram to 200 and 300, therefore saturating the image beyond 255 and changing its nature. Unless the original image is saved, dividing the pixel values by 2 will yield an image with a histogram between 100 and 127.

Figure 9.16(a) shows an original image that was later altered by an image-formatting routine for increased brightness (Figure 9.16(c)) and increased contrast (Figure 9.16(e)). As is evident in the histograms (b) and (d), when an image is brightened, its histogram distribution simply shifts, in this case by 30 points. When the contrast of the image is increased, in this case by 50%, the distribution of pixel gray levels is expanded, although the relationship remains the same. However, unlike the previous example, the distribution of gray levels is different because new gray values are introduced. ■

9.9 Thresholding

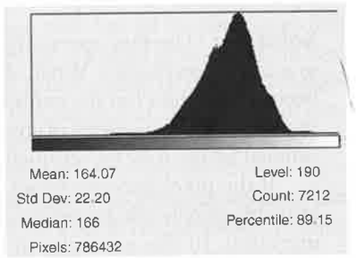
Thresholding is the process of dividing an image into different portions, or levels, by picking a certain greyness level as a threshold, comparing each pixel with the threshold value, and assigning the pixel to the different portions (or levels) of interest depending on whether the pixel's greyness level is below the threshold (off, zero, or not belonging) or above the threshold (on, 1, or belonging). Thresholding can either be performed at a single level or with multiple thresholding values where the image is processed by dividing the image into layers, each layer with a selected threshold. To aid in choosing an appropriate threshold, many different techniques have been suggested. These techniques range from simple routines for binary images to sophisticated techniques for complicated images. Early routines were used for a binary image where the object was bright and the background was completely dark. This condition can be achieved in controlled lighting in industrial situations but may not be available in other environments. In binary images, the pixels are either on or off; therefore, choosing a threshold is simple and

(a)

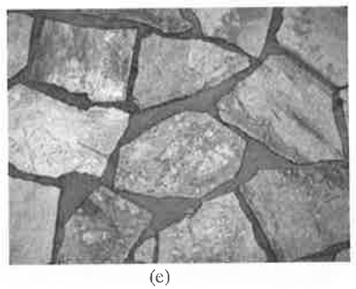


(b)

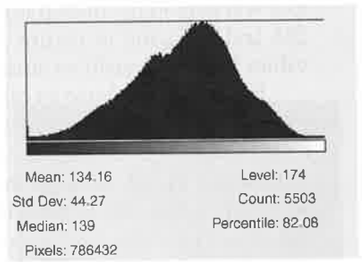
(c)



(d)



(e)

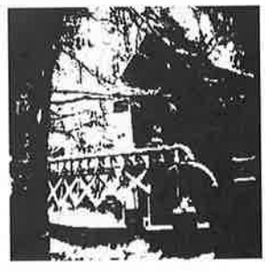


(f)

Figure 9.16 Increasing the contrast in an image expands the histogram to include new gray values.

(a)

(b)



(c)

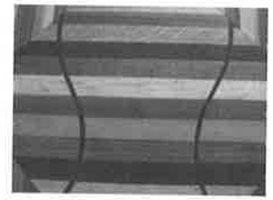
Figure 9.17 Thresholding an image with 256 gray levels at two different values of (b) 100 and (c) 150.

straightforward. In other situations, the histogram may be a multimodal distribution. In this case, the valley(s) are chosen as the threshold value. More advanced techniques use statistical information and distribution characteristics of the image pixels to develop a thresholding value. For example, the lowest value between two peaks, the midpoint between two peaks, the average of two peaks, and many other scenarios may be used. As the thresholding value changes, so does the image. Figure 9.17(a) shows an original image with 256 gray levels and the result of thresholding at greyness levels of (b) 100 and (c) 150.

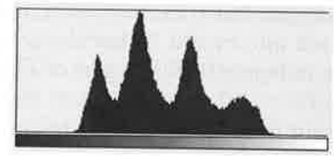
Thresholding is used in many operations such as converting an image into binary form, filtering operations, masking, and edge detection.

Example 9.3

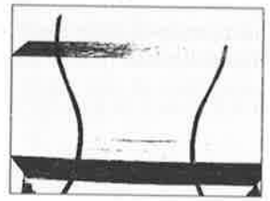
Figure 9.18(a) shows an image of a cutting board and its histogram. Due to the nature of this image, there are four peaks in the histogram. Figures 9.18(c), (e), and (g) show



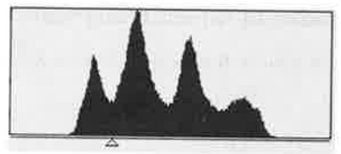
(a)



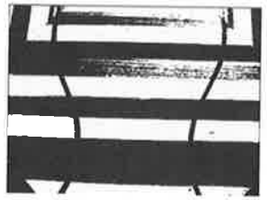
(b)



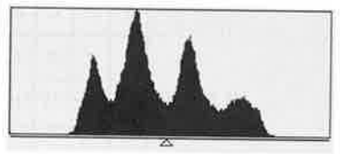
(c)



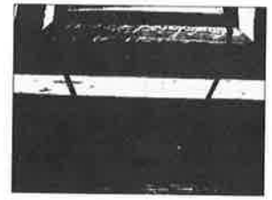
(d)



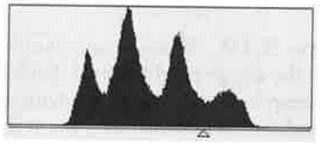
(e)



(f)



(g)



(h)

Figure 9.18 Images and histograms for Example 9.3.

the effect of thresholding at different levels. In fact, in this case, different types of wood can be identified and separated from each other due to their colors. ■

9.10 Spatial Domain Operations: Convolution Mask

Spatial domain processes access and operate on the individual pixel information. As a result, the image is directly affected by the operation. Many processes used in vision systems are in spatial domain. One of the most popular and most common techniques in this domain is convolution, which can be adapted to many different activities such as filters, edge finders, morphology, and many more. Many processes in commercial vision systems and photography software are based on convolution, too. The following is a discussion of basic principles behind convolution. Later, we will apply the convolution idea to different purposes.

Imagine an image is composed of pixels, each with a particular gray level or color information that collectively constitute the image (in this example, the gray level is not digitized into 0s and 1s, but the actual value is indicated). As an example, let's say the image in Figure 9.19(a) is part of a larger image with pixel values shown symbolically as *A, B, C*. . . . Let's also assume there is a 3 × 3 kernel or mask, as shown, which has values in its cells as indicated by m_1 through m_9.

Applying the mask onto the image involves superimposing (convolving) the mask, first on the upper left corner of the image and taking the summation of the product of the value of each pixel multiplied by the corresponding mask value and dividing the summation by a normalizing value. This will yield (please follow carefully):

$$R = (A \times m_1 + B \times m_2 + C \times m_3 + E \times m_4 + F \times m_5 + G \times m_6 + I \times m_7 + J \times m_8 + K \times m_9)/S \tag{9.2}$$

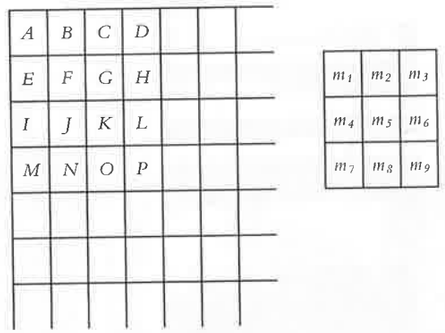


Figure 9.19 When a convolution mask (kernel) is superimposed on an image, it can change the image pixel by pixel. Each step consists of superimposing the cells in the mask onto the corresponding pixels, multiplying the values in the mask's cells by the pixel values, adding the numbers, and normalizing the result. The result 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.

where S is the normalizing value. This is usually the summation of the values in the mask, or

$$S = |m_1 + m_2 + m_3 + \ldots + m_9| \tag{9.3}$$

If the summation is zero, substitute $S = 1$ or choose the largest number. The result R is substituted for the value of the pixel in the center of the block that was superimposed. In this case, R will replace the pixel value of F. Usually, the substitution takes place into a new file in order to not alter the original file($R \rightarrow F_{new}$).

The mask is then moved one pixel to the right and the same is repeated for a new R which will replace G as follows:

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

The result is, once again, substituted for G in a new file. The mask is then moved over one more pixel, and the operation is repeated until all the pixels in the row are changed. Then the operation continues in a raster scan fashion (see Appendix B) with the following rows until the image is completely affected. The resulting image will show characteristics that may be slightly or very severely affected by the operation, all depending on the m values in the mask. The first and last rows and columns are not affected by this operation, and therefore, are usually ignored. Some systems insert zeros for the first and last rows and columns or retain the original values. Another alternative is to copy the first and last rows and columns into an additional layer of rows and columns around the image in order to calculate new values for these pixels.

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 as shown in Figure 9.20, the value for the pixel $(I_{x,y})_{new}$ as the center of a block can be calculated by:

$$(I_{x,y})_{new} = \frac{1}{S}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} M_{i,j} \times I\left[\left(x - \left(\frac{n+1}{2}\right) + i\right), \left(y - \left(\frac{n+1}{2}\right) + j\right)\right]\right) \tag{9.4}$$

$$S = \left|\sum_{i=1}^{n}\sum_{j=1}^{n} M_{i,j}\right| \quad \text{if } S \neq 0$$
$$S = 1 \text{ or largest number} \quad \text{if } S = 0 \tag{9.5}$$

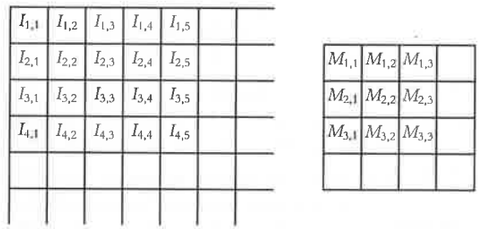


Figure 9.20 The representation of an image and a mask.

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 9.4

Consider the pixels of an image, with values as shown in Figure 9.21, as well as a convolution mask with the given values. Calculate the new values for the given pixels.

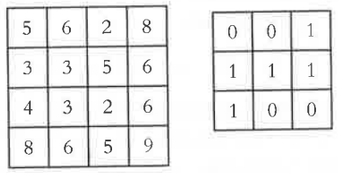
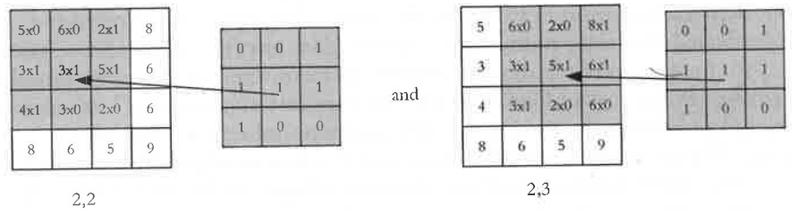


Figure 9.21 An example of a convolution mask.



(a)

0	0	0	0
0	3	5	0
0	4	5	0
0	0	0	0

(b)

Figure 9.22 (a) Convolving the mask onto the cells of the image; (b) the result of the operation.

Solution: We substitute zeros for the first and last columns and rows because they are not affected by this process. For the remaining pixels, we superimpose the mask on the remaining cells of the image and use Equations (9.2) and (9.3) to calculate new pixel values, as shown in Figure 9.22(a), with the result shown in Figure 9.22(b). Superimposing the mask on the image as shown for each remaining element, we get:

2,2: $[5(0) + 6(0) + 2(1) + 3(1) + 3(1) + 5(1) + 4(1) + 3(0) + 2(0)]/5 = 3.4$
2,3: $[6(0) + 2(0) + 8(1) + 3(1) + 5(1) + 6(1) + 3(1) + 2(0) + 6(0)]/5 = 5$
3,2: $[3(0) + 3(0) + 5(1) + 4(1) + 3(1) + 2(1) + 8(1) + 6(0) + 5(0)]/5 = 4.4$
3,3: $[3(0) + 5(0) + 6(1) + 3(1) + 2(1) + 6(1) + 6(1) + 5(0) + 9(0)]/5 = 4.6$

In reality, greyness levels are integers, and therefore, all numbers are rounded to whole numbers. ■

Example 9.5

Apply the 7×7 mask shown to the image of Figure 9.23.

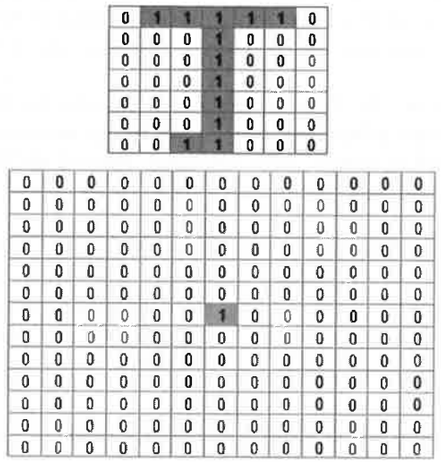


Figure 9.23 The image and mask for Example 9.5.

Solution: Applying the mask to the image results in Figure 9.24. As you can see, the on-cells in the mask have convolved into the same shape within the image, albeit an upside down and mirror image, when applied to a single on-pixel in the image. This, in fact, demonstrates the real meaning of the convolution mask. Any set of numbers used in the mask will convolve into the image and will affect it accordingly. Therefore, the choice of numbers in the mask can have a significant effect on the image. Please also notice how the first and last three rows and columns are unaffected by the 7×7 mask.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9.24 The result of convolving the mask on the image of Example 9.5. ■

9.11 Connectivity

In a number of instances, we need to decide whether or not neighboring pixels are somehow "connected" or related to each other. This connectivity establishes whether they are of the same properties, such as being of the same region or object, similar textures or colors, and so on. To establish this connectivity of neighboring pixels, we first have to decide a connectivity path. For example, we need to decide whether only pixels on the same column and row are connected, or diagonally situated pixels are also accepted as connected.

There are three fundamental connectivity paths for 2D image processing and analysis: +4 or ×4-connectivity, H6 or V6 connectivity, and 8-connectivity. In 3D, connectivity between voxels (volume cells) can range from 6 to 26. Refer to Figure 9.25 for the following definitions:

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

×4-connectivity—a pixel p's relationship is only analyzed with respect to the 4 pixels diagonally across from it on 4 sides (a,c,f,h). For pixel $p(x,y)$, these are defined as:

$$\text{for } +4\text{-connectivity } (x+1,y),(x-1,y),(x,y+1),(x,y-1) \tag{9.6}$$

$$\text{for } \times 4\text{-connectivity } (x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1) \tag{9.7}$$

H6-connectivity—a pixel p's relationship is only analyzed with respect to the 6 neighboring pixels on two rows immediately above and below the pixel (a,b,c,f,g,h).

a	b	c
d	p	e
f	g	h

Figure 9.25 Neighborhood connectivity of pixels.

V6-connectivity—a pixel p's relationship is only analyzed with respect to the 6 neighboring pixels on two columns immediately to the right and to the left of the pixel (a,d,f,c,e,h). For pixel $p(x,y)$, these are defined as:

for H6-connectivity

$$(x-1,y+1),(x,y+1),(x+1,y+1),(x-1,y-1),(x,y-1),(x+1,y-1) \tag{9.8}$$

for V6-connectivity

$$(x-1,y+1),(x-1,y),(x-1,y-1),(x+1,y+1),(x+1,y),(x+1,y-1) \tag{9.9}$$

8-connectivity—a pixel p's relationship is analyzed with respect to all 8 pixels surrounding it (a,b,c,d,e,f,g,h). For pixel $p(x,y)$, this is defined as:

$$(x-1,y-1),(x,y-1),(x+1,y-1),(x-1,y),(x+1,y),(x-1,y+1),(x,y+1),(x+1,y+1) \tag{9.10}$$

Example 9.6

In Figure 9.26, starting with pixel (4,3), find all succeeding pixels that can be considered connected to each other based on +4, ×4, H6, V6, and 8-connectivity rules.

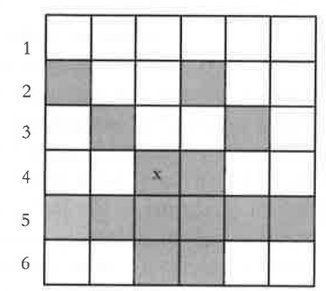


Figure 9.26 The image for Example 9.6.

Solution: Figure 9.27 shows the results of the connectivity search. Follow each one. You must take one pixel, find all others connected to it based on the applicable connectivity rule, and search the pixels found to be connected to the previous ones for additional connected pixels, until done. The remaining pixels are not connected. We will use the same rules later for other purposes such as region growing. The H6, V6, and 8-connectivity search is left for you to do as an exercise.

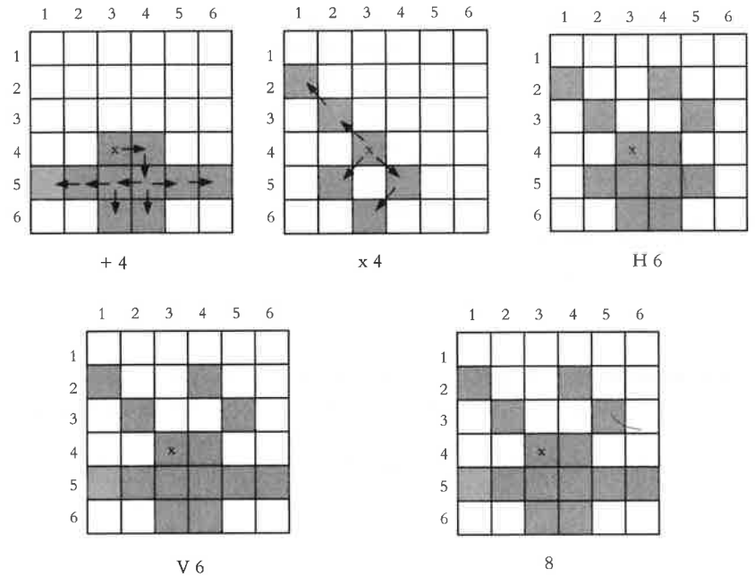


Figure 9.27 The results of the connectivity searches for Example 9.6. ■

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

9.12 Noise Reduction

Similar to other signal-processing mediums, vision systems contain noise. Some noise is systematic and comes from dirty lenses, faulty electronic components, bad memory 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, some images have low quality due to hardware and software inadequacies, and therefore, have to be enhanced and improved before other analyses can be performed on them. At the hardware level, in one attempt,[1] an on-chip correction scheme was devised for defective pixels in an image sensor. In this scheme, readouts from the nearest neighbors were substituted for identified defective pixels. However, in general, software schemes are used for most filtering operations.

Filtering techniques are divided into two categories of frequency domain and spatial domain. 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.

9.12.1 Neighborhood Averaging with Convolution Masks

As discussed in section 9.10, a mask may be used for many different purposes, including filtering operations and noise reduction. In section 9.4, it was also discussed that noise,

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

1	1	1
1	1	1
1	1	1

Figure 9.28 Neighborhood averaging mask.

along with edges, creates higher frequencies in the spectrum. It is possible to create masks that behave like low-pass filters, such that the higher frequencies of an image are attenuated, while the lower frequencies are not changed much, and thereby, reduce the noise.

Neighborhood averaging with a convolution mask can be used to reduce the noise in images, but it also reduces the sharpness of an image. Consider the 3×3 mask in Figure 9.28 with its corresponding values, as well as a portion of an imaginary image, with its gray levels shown.

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 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 (summation of all values in the mask) will yield:

$$R = (20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 100 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1 + 20 \times 1)/9 = 29$$

As a result of applying the mask on that corner, the pixel with the 100 value will change to 29. Consequently, the large difference between the noisy pixel and the surrounding pixels (100 versus 20) becomes much smaller (29 versus 20), thus reducing the noise. If the mask is applied to the set of pixels in columns 3, 4, 5, the average will be 20; therefore, the operation has no effect on the set. The difference between pixels remains low. With this characteristic, this mask acts as a low-pass filter because it attenuates the sharp differences between neighboring pixels but has little effect on pixels whose intensities are similar. Notice that this routine will introduce new gray levels in the image (29), and therefore, will change the histogram of the image. Similarly, this averaging low-pass filter will also reduce the sharpness of edges, making the resulting image softer and less focused. Figure 9.29 shows (a) an original image, (b) an image corrupted with noise, (c) the image after a 3×3 averaging filter application, and (d) the image after a 5×5 averaging filter application. 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 Gaussian (also called Mild Isotropic Low-Pass), shown in Figure 9.30. This filter will similarly improve the image, but with a slightly different result.

9.12.2 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 completely stop. As a result, in addition to being time consuming, this technique is

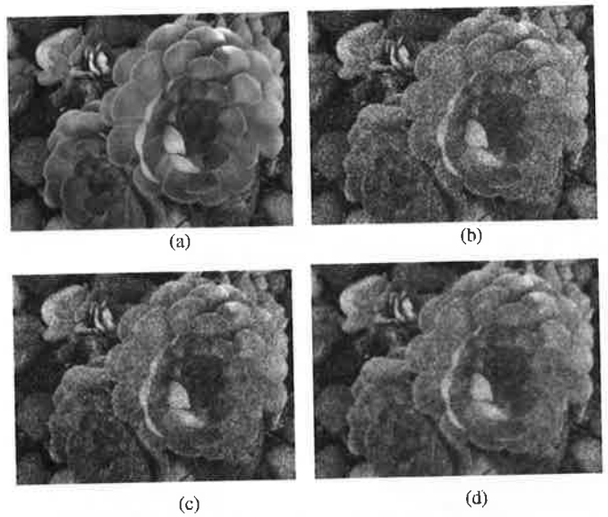


Figure 9.29 Neighborhood averaging of an image.

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 9.30 5×5 and 3×3 Gaussian averaging filters.

not suitable for operations that are dynamic and change rapidly. Image averaging is more effective at increased numbers of images and is fundamentally useful for random noise. 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 this averaging because the summation of random noises will be zero, or:

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

$$\frac{\sum_n A(x,y)}{n} = \frac{\sum_n I(x,y) + N(x,y)}{n} = \frac{\sum_n I(x,y)}{n} + \frac{\sum_n N(x,y)}{n} = I(x,y) \qquad (9.11)$$

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

9.12.3 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.

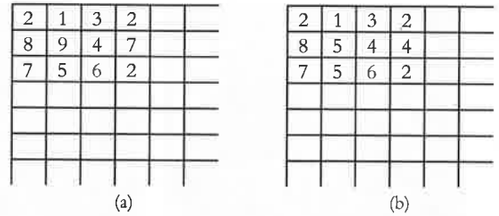


Figure 9.31 Application of a median filter.

9.12.4 Median Filters

One of the main problems in using neighborhood averaging is that along with removing noise, the filter will also blur the edges and reduce the sharpness of the image. A variation to 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 pixel (the pixel plus the 8 surrounding pixels), sorted in ascending order. A median is the value where half of the values in the set are below and half are above the median (also called 50th percentile). Since, unlike an average, the median's final value is independent of the value of any single pixel in the set, the median filter will be much stronger in eliminating spike-like noises without blurring the object or decreasing the sharpness of the image.

Suppose we apply a median filter to the image in Figure 9.28. The sorted values in ascending order will be 20, 20, 20, 20, 20, 20, 20, 20, 100. The median is 20 (the fifth one from the left). Replacing the center pixel's value with 20 will completely eliminate the noise. Of course, noise is not always this easily removed. But this 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 9.31(a). The values in ascending order for the left-most corner are 1, 2, 3, 4, 5, 6, 7, 8, 9. The middle value is 5, resulting in the image in (b). The values for the second set of 9 pixels are 1, 2, 2, 3, 4, 5, 6, 7, 9, and the median is 4. As you can see, the image has become grainy because the pixel sets with similar values appear longer (as in 5 and 5 or 4 and 4).

Figure 9.32 shows (a) an original image, (b) the image corrupted with random Gaussian noise, (c) the image improved with a 3×3 median filter, and (d) a 7×7 median filter. Generally, larger size median filters are more effective.

9.13 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, textures, lines, and colors, differences in light intensities between parts and backgrounds or features such as holes and protrusions, as well as differences in shading and textures. Some techniques are mathematically oriented, some are heuristic, and some are descriptive techniques. They generally operate on the differences between the gray levels of pixels or groups of pixels through masks or

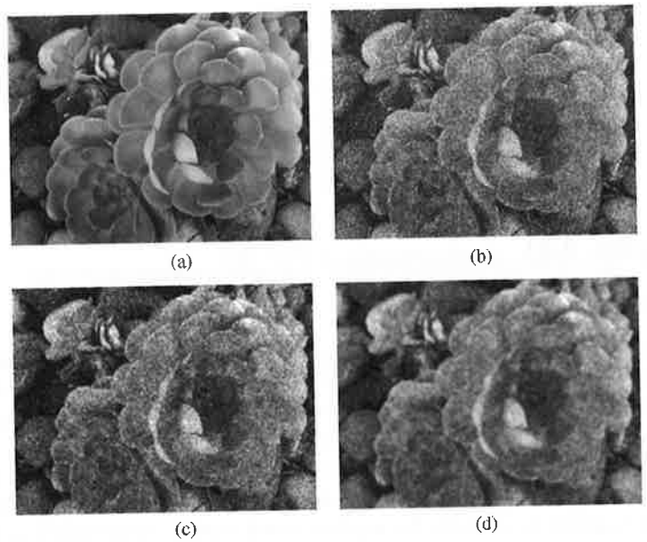


Figure 9.32 (a) is the original image, (b) is the same image corrupted with a random Gaussian noise, (c) is the image improved by a 3×3 median filter, and (d) is the same image improved with a 7×7 median filter.

thresholds. The final result is a line drawing or similar representation that requires much less memory, can be processed more easily, and saves in computational 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, calculate features such as diameter and area, or determine parts by region growing. Different techniques of edge detection yield slightly different results, and therefore, should be chosen carefully and used wisely.

Except in binary images, edges are generally not ideal. This means that instead of a clear distinction between two neighboring pixels' gray levels, the edge is spread over a number of pixels, as shown in Figure 9.33. A simple comparison between two pixels may be inadequate for edge detection. The first and second derivatives of the graph are also shown. It is possible to assume that the edge is at the peaks of the first derivative or at the zero crossing of the second derivative and to use these values to detect the edges. The problem is exacerbated when the image is noisy; therefore, the derivatives have excessive numbers of peaks or zero crossings.

Generally, the edges are at regions of rapid intensity change. Referring to Figure 9.34, the magnitude and direction of the gradient of image intensity can be calculated as:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

$$(\nabla I)_{magnitude} = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{9.12}$$

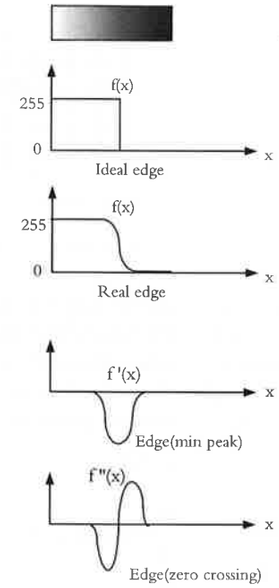


Figure 9.33 Edge detection with first and second derivatives.

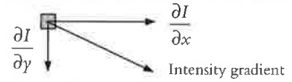


Figure 9.34 Gradient of image intensity.

$$(\nabla I)_{direction} = \tan^{-1}\frac{(\partial I/\partial y)}{(\partial I/\partial x)} \tag{9.13}$$

Similarly, the second gradient of the intensity, called *Laplacian,* is shown as Equation (9.14). The magnitude and orientation of the second gradient can be calculated in a similar fashion.

$$\nabla^2 I = \left(\frac{\partial^2 I}{\partial x^2}, \frac{\partial^2 I}{\partial y^2}\right) \tag{9.14}$$

Digital Implementation Since images are discrete, a finite difference approach may be taken in order to calculate the gradients. For a one-dimensional system, the finite difference between successive elements is:

$$f'(x) = \lim_{dx \to 0} \frac{f(x + dx) - f(x)}{dx} \tag{9.15}$$

In an image, dx is 1 pixel wide. Therefore, the finite difference for an image can be simplified to $F'(x) = F(x + 1) - F(x)$ and be implemented by kernel $[-1 \quad 1]$. For a

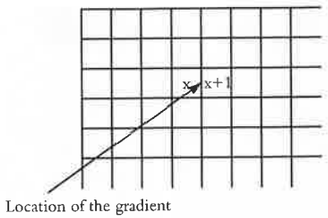


Figure 9.35 Intensity gradient between successive pixels.

two dimensional system, the same is applied in both x and y directions. Referring to Figure 9.35, notice that when the finite difference is calculated, it does not relate to the center of the pixel of interest; rather, there are two midpoints between successive pixels to which the gradients apply. To remedy this, the finite difference can be calculated between the pixels before and after the point of interest and averaged using the modified kernel (mask) $\frac{1}{2}[-1 \quad 0 \quad 1]$, yielding:

$$\begin{aligned} \frac{dF}{dx} &\approx F(x+1) - F(x-1) \\ \frac{dF}{dy} &\approx F(y+1) - F(y-1) \end{aligned} \tag{9.16}$$

Similarly, the second derivative of the image intensities can be calculated with finite difference as:

$$\begin{aligned} F''(x) = \frac{\partial^2 F}{\partial x^2} &= [F(x+1) - F(x)]' \\ &= [F(x+1) - F(x)] - [F(x) - F(x-1)] \\ &= F(x-1) - 2F(x) + F(x+1) \end{aligned} \tag{9.17}$$

which can be implemented by a kernel $[1 \quad -2 \quad 1]$. Therefore, the approximate magnitude of the Laplacian for a two-dimensional image can be calculated by applying the following kernel (mask):

$$\begin{aligned} \text{Laplacian}(0°, 90°) &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ \text{Laplacian}(45°) &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{aligned} \tag{9.18}$$

As we will soon see, this is a common way of detecting edges. In fact, many other common masks used for edge detection are variations of the gradient scheme.

As discussed earlier, like noise, edges are high frequency, and therefore, 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 noises and edges from the rest of the

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

-1	0	-1
0	4	0
-1	0	-1

Figure 9.36 The Laplacian-1 high-pass edge detector mask.

image. Consider the image and the Laplacian kernel (mask) in Figure 9.36. As you see, this mask has negative numbers. Applying the mask to the image at the corner yields:

$$\begin{aligned} \text{R} = (&20 \times -1 + 20 \times 0 + 20 \times -1 + 20 \times 0 + 100 \times 4 + 20 \times 0 + 20 \times -1 \\ &+ 20 \times 0 + 20 \times -1)/1 = 320 \end{aligned}$$

The normalizing factor is 1, which results in the value of 100 replaced with 320, comparatively accentuating the original difference (from 100 versus 20 to 320 versus 20) while applying the mask to the set of pixels in columns 3, 4, 5 yields zero, indicating that the difference between pixels is not changed. Since this mask accentuates large intensity variations (higher frequencies) while ignoring similar intensities (lower frequencies), it is a high-pass filter. This also means that the noise and edges of objects in images will be shown more effectively. As a result, this mask acts as an edge detector. Some high-pass filters act as an image sharpener. Figure 9.37 shows some other high-pass filters.

The following three masks[2–6]—Sobel operator, Roberts edge, and Prewitt—shown in Figure 9.38 effectively do the same gradient differentiation with somewhat different results and are very common. When applied to an image, the two pairs of masks calculate the gradients in the x and y directions, which are added and compared to a threshold. Notice how these follow the gradient equations developed earlier.

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian-2

0	-1	0
-1	6	-1
0	-1	0

Sharpen, Low

0	-1	0
-1	5	-1
0	-1	0

Sharpen, Medium

Figure 9.37 Other high-pass filters.

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 9.38 The Sobel, Roberts, and Prewitt edge detectors.

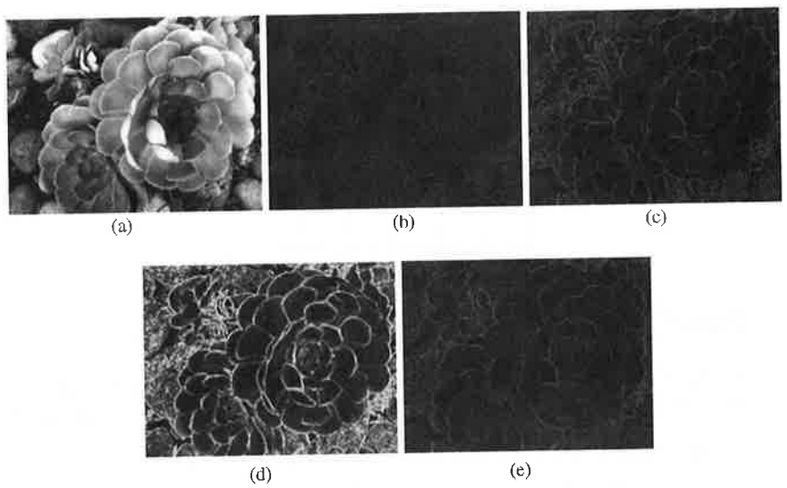


Figure 9.39 An image and its edges from Laplacian-1 (b), Laplacian-2 (c), Sobel operator (d), and Robert's edge (e).

Figure 9.39 is an original image (a) with its edges detected by a Laplacian-1 (b), Laplacian-2 (c), Sobel operator (d), and Robert's edge (e).

You must realize that although in this example the results are as shown, the result for other images may be different. This is 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 methods can be used for binary images that are simple to implement and yield continuous edges. In one example,[7] a search technique, dubbed Left-Right (L-R) in this book, is used to quickly and efficiently detect edges in binary images of single objects that look like a blob. Imagine a binary image as shown in Figure 9.40. Let's assume that gray pixels are "on" (or the object) and white pixels are "off" (background).

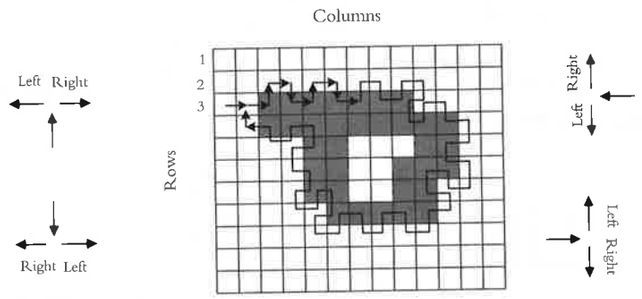


Figure 9.40 Left-Right search technique for edge detection.[7]

Table 9.2 *Possible Left-Right Schemes Based on the Direction of Search.*

If Vpresent-Vprevious>0	left	Unext=Upresent−1
	right	Unext=Upresent+1
if Vpresent-Vprevious<0	right	Unext=Upresent−1
	left	Unext=Upresent+1
If Upresent-Uprevious<0	right	Vnext=Vpresent+1
	left	Vnext=Vpresent−1
If Upresent-Uprevious>0	right	Vnext=Vpresent−1
	left	Vnext=Vpresent+1

Assume a pointer is moving from one pixel to another, in any direction (up, down, right, left). Any time the pointer reaches an "on" pixel, it will turn left. Any time it reaches an "off" pixel, it will turn right. Of course, as shown, depending on the direction of the pointer, the left and right might mean different directions. Starting at pixel 1,1, moving to 1,2, to the end, then row 2, and then row 3, the pointer will find the first "on" pixel at 3,3, will turn left, and encounter an "off" pixel, turn right twice, then left, and will go 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 . . . .

Table 9.2 shows how a simple computer program can be developed to do the search. U and V are pixel coordinates.

Masks may also be used for the intentional emphasis of some characteristic of the image. For example, a mask may be designed to emphasize horizontal lines, vertical lines, or diagonal lines. Figure 9.41 shows three such masks. Figure 9.42 shows an original image (a), along with the effects of a vertical mask (b), a horizontal mask (c), and a diagonal mask (d).

3	-6	3
3	-6	3
3	-6	3

Vertical emphasis mask

3	3	3
-6	-6	-6
3	3	3

Horizontal emphasis mask

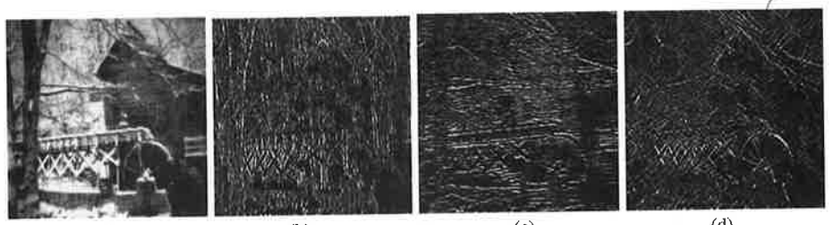
3	3	-6
3	-6	3
-6	3	3

Diagonal emphasis mask

Figure 9.41 These masks emphasize the vertical, horizontal, and diagonal lines of an image.

9.14 Sharpening an Image

Image sharpening can be accomplished in many different ways. The simplest is to apply a relatively high-pass filter to the image that increases the sharpness of the image by eliminating some of the lower frequencies from the edges. However, in sharpening operations, noise is increased too, and therefore, as the level of sharpening increases, so does the noise level. Figure 9.37, partially repeated here, shows two simple sharpening masks.



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

Figure 9.42 An original image (a) with effects of vertical emphasis mask (b), horizontal emphasis mask (c), and diagonal emphasis mask (d).

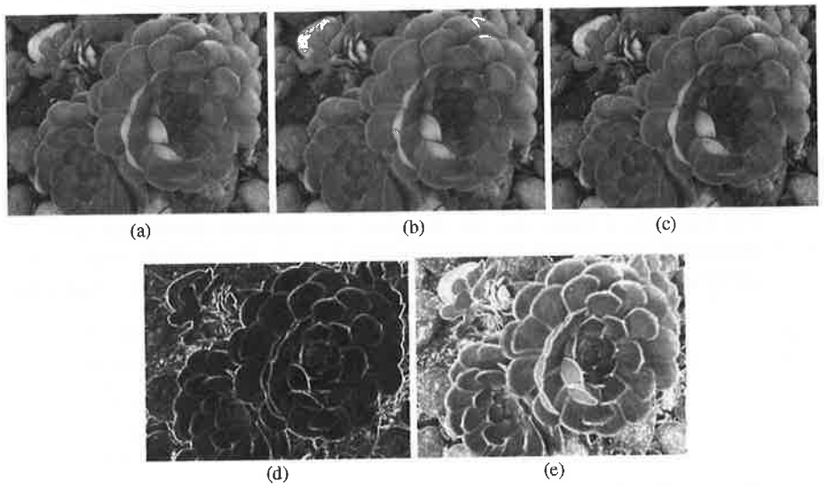
0	-1	0
-1	6	-1
0	-1	0

Sharpen, Low

0	-1	0
-1	5	-1
0	-1	0

Sharpen, Medium

Figure 9.37 Repeated



(a) (b) (c)

(d) (e)

Figure 9.43 The original image (a), after an averaging mask was applied to it (b), the result of sharpening with a low sharpening mask (c), Sobel edge (d), the result of adding the Sobel edge to the original image.

Figure 9.43 shows a more sophisticated method to sharpen images. In this case, a 3×3 mask was applied to the original image to decrease noise (b), followed by a low sharpening mask (c), followed by a Sobel edge detector (d). The result was added to the original image (e). As you can see, the image shows more detail and is somewhat sharpened, but there is also more noise present.

9.15 Hough Transform

As you have probably noticed, in most edge detection techniques, the resulting edges are not continuous. However, there are many applications where continuous edges are either necessary or preferred. For example, as we will see later, in region growing, edges that define an area or region must be continuous and complete before a region growing routine can detect and label it. Additionally, 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. *Hough transform*[8] is a technique used to determine the geometric relationship between different pixels on a line, including the slope of the line. For example, we can determine whether a cluster of points is on a straight line or not. This also aids in the further development of an image in preparation for object recognition since it relates individual pixels into recognizable forms.

Hough transform is based on transforming the image space (x,y) into either (r,θ) or (m,c) space. The normal from the origin to any line will have an angle of θ with respect to the x-axis and a distance of r from the origin. The transformation into the r,θ-plane (also called Hough plane) showing these values is called Hough transform (Figure 9.44(a)). Note that since all the points constituting the line in x,y-plane have the same r,θ values, they are all represented by the same point A in r,θ-plane. Therefore, all points on a straight line are represented by a single point in the Hough plane.

Similarly, a line in the x,y-plane with a slope m and intercept c can be transformed into a Hough plane of m,c with x and y as its slope and intercept (Figure 9.44(b)). Therefore, a line in the x,y-plane with a particular slope and intercept will transform into a point in the Hough plane. Since all points on this line have the same m and c, they are all represented by the same point in the Hough plane.

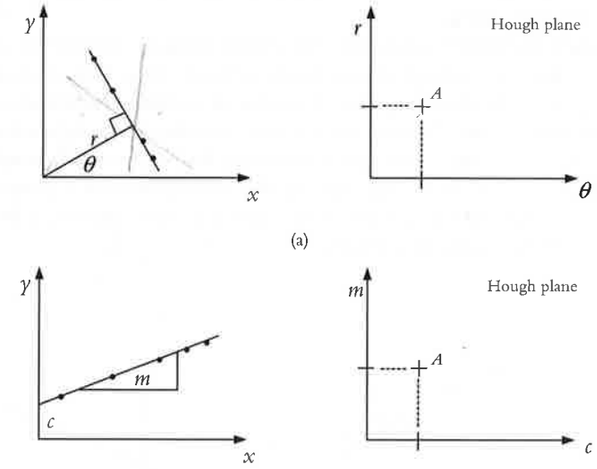


Figure 9.44 The Hough transform from x,y-plane into r,θ-plane or m,c-plane.

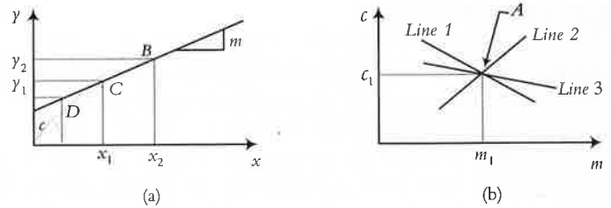


Figure 9.45 Hough transform.

Now consider the line in Figure 9.45(a), described by its slope m and intercept c as:

$$y = mx + c \tag{9.19}$$

Equation (9.19) can also be written in terms of m and c as variables as:

$$c = -xm + y \tag{9.20}$$

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

As discussed earlier, the line of Equation (9.19) with m and c converts to a single point A in the m,c-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.

The opposite is also true. As shown in Figure 9.46, an infinite number of lines may go through a point in the x,y-plane, all intersecting at the same location. Although these lines have different slopes m and intercepts c, they all share the same point x,y which become the slope and intercept in the Hough plane. Therefore, the same x and y values represent all these lines, and consequently, a point in the x,y-plane is represented by a line in the Hough plane.

Hough transform converts the pixels (edges) within an image into lines in the Hough plane. If a group of points are colinear, their Hough transforms will all intersect at one point. By checking this, it can be determined whether a cluster of pixels is on a straight line or not. Hough transforms can also be used in determining the angle or orientation of a line. This application has found use in determining the orientation of an object in a plane by calculating the orientation of a particular line in the object. Since the intercept and slope of the line are now known, a broken line can easily be completed by additional points.

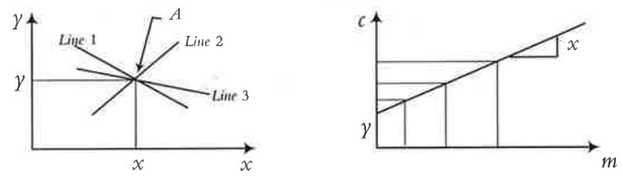


Figure 9.46 Transformation of a point in x,y-plane into a line in the Hough plane.

Example 9.7

The x and y coordinates of 5 points are given as (1,3), (2,2), (3,1.5), (4,1), and (5,0). 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 form a line. So, we will look for at least three points that will be on the same line. Clearly, looking at the graph of the points, it is very easy to answer the questions, a trivial matter. 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, whether for a computer or a human, to tell which points are on the same line and which ones are not. We will perform a Hough transform to determine which points fall on the same line. The following table summarizes the lines formed in the m,c-plane that correspond to the points in the x,y plane:

y	x	x,y	m,c
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 9.47 shows the five corresponding lines drawn in the m,c-plane. As you see, three different lines intersect at two different places, while other intersections are just between two lines. These correspond to points (1,3), (3,1.5), (5,0) and to (2,2), (3,1.5), and (4,1). The slope and intercept of the first line are -0.75 and 3.75. The slope and intercept for the second line are -0.5 and 3 respectively. This shows how the Hough transform can be cluttered with an exceeding number of intersecting lines. Determining which lines are intersecting is the main issue in Hough transform analysis.

The equations representing these lines are $y = -0.75x + 3.75$ and $y = -0.5x + 3$. Using these equations, additional points lying on these lines can be assigned to the group, therefore completing broken lines.

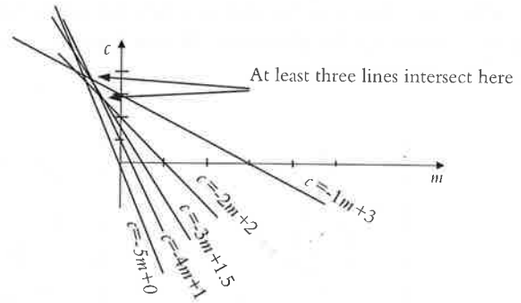


Figure 9.47 Hough transform for Example 9.7.

■

Coincidentally, the same analogy may be made for circles and points instead of lines and points. All points on a circle will correspond to intersecting circles in the Hough plane and vice versa. For more information, refer to Reference 3.

The Hough transform has many desirable features. For example, since 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, Hough transform is computationally intensive. To reduce the number of calculations needed to determine whether lines are actually intersecting with each other at the same point, we may use a circle within which, if the lines approximately intersect with each other, they are assumed to be intersecting. Many variations to the Hough transform have been devised to increase its efficiency and utility for different tasks, including object recognition.[9]

9.16 Segmentation

Segmentation is a generic name for a number of different techniques that divide the image into segments or constituents. The purpose 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, or by division into small areas (blobs), and so on. 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 three-dimensional analysis of objects, models such as cylinders, cones, spheres, and cubes were used as well. Although these shapes and figures did not necessarily match any real objects, they provided a means for early developmental work that evolved into more sophisticated routines and techniques. They also provided a means to develop schemes that could process complicated shapes and recognize objects. As an example, the routines could model a tree as a cone or sphere mounted on a cylinder (Figure 9.48) and could match it with a model of a tree, requiring very little processing power; the tree could be expressed with only a few pieces of information such as the diameters of the cone and cylinder and their heights, while representing all the information pertaining to a tree could be enormous in comparison.

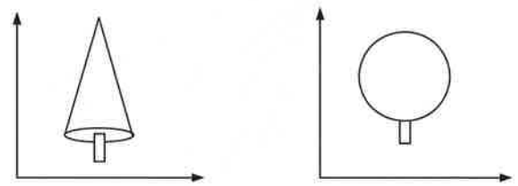


Figure 9.48 Representation of objects such as a tree with models such as a cone or sphere mounted on a cylinder can reduce processing requirements.

9.17 Segmentation by Region Growing and Region Splitting

In addition to edge detection routines, region growing and image splitting are other common techniques of segmentation. 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 in object detection. Edges found by an edge detector are lines of textures, colors, planes, and gray levels, and therefore, may or may not be continuous. However, segmentation by regions naturally results in complete and closed boundaries. For a survey of other segmentation techniques, see Reference [10].

Two approaches are used for region segmentation. One is to grow regions by similar attributes such as a range of gray levels or other similarities. The other is region splitting which will split images into smaller areas using their finer differences.

One technique of region splitting is thresholding. The image is split into closed areas of neighboring pixels by comparing them to a thresholding value or range. Any pixel that falls below a threshold (or between a range of values) will belong to a region, and otherwise, to another. This will split the image into a series of regions or clusters of pixels that have common or similar attributes. Generally, although this is a very simple technique, it is not very effective since 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, it is a useful technique under certain conditions such as silhouettes and for images with relatively uniform regions.

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

The following is a simple search technique for growing regions for a binary image (or with the application of thresholding, for gray images as well) that uses a bookkeeping approach to find all pixels that belong to the same region.[11] Figure 9.49 shows a binary image. Each pixel is referred to by a pair of index numbers. Assume a pointer starts at the top and searches for a nucleus to start a region. As soon as a nucleus is found (which does not already belong to another region), the program assigns a region number to it. All pixels connected to it 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 this will change the final outcome. As discussed in section 9.11, +4-, ×4-, H6-, V6-, and 8-connectivity can be used for region growing. In Figure 9.49, the first nucleus is found at pixel 2d. Suppose we have chosen the +4-connectivity. The program will check the four corresponding pixels around the nucleus to determine connectivity. If there is an "on" pixel, its location index numbers will be placed in a stack, the cell is given the region number (n), and the pointer is moved down in the stack to the next cell, 3d. 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 region-n designation, and the

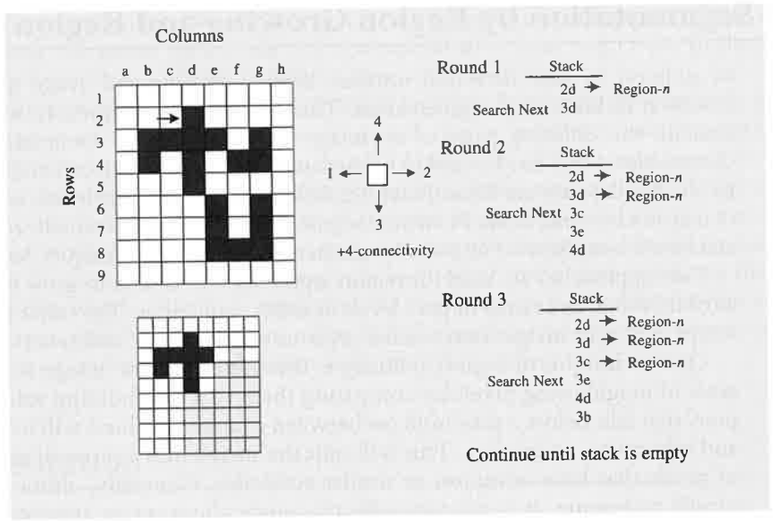


Figure 9.49 Region growing based on a search technique. With +4-conectivity search, region-*n* is as shown.

process is repeated for the next index number on the stack, 3c. The process continues until the stack is empty.

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

Example 9.8

Using ×4-, H6-, V6-, and 8-connectivity, determine the first region that results from a search of Figure 9.49.

Solution: The result is shown in Figure 9.50 for the ×4-connectivity. Please follow the routine for other connectivity rules.

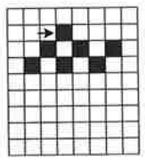


Figure 9.50 The result of a search for ×4-connectivity for Example 9.8. ■

There are many other segmentation schemes that apply to different situations. For example, in one technique, the following is done:

- Assign the image to *k* clusters.
- Calculate the mean of each cluster.

- If the mean of a token area (or pixel) is closer to its cluster's mean, keep it.
- If the mean of a token area (or pixel) is closer to another cluster's mean, reassign it to the other cluster.
- Continue until no changes are made.

Once again, as you notice, a bookkeeping and comparison routine is applied to the image in order to segment it based on a desired characteristic, in this case the mean of each area. Other schemes can be found in other references.[2,12–14]

9.18 Binary Morphology Operations

Morphology operations refer to a family of operations performed on the shape (therefore, 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 image analysis as well as for reducing the "extra" information that may be present in the image. For example, consider the binary image in Figure 9.51(a) and the stick figure representing one of the bolts in (b). As we will see later, a moment equation may be used to calculate the orientation of the bolts. However, the same moment calculation can also be performed on 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 following sections, we will discuss a few of these operations.

Morphology operations are based on the set theory. For example, in Figure 9.52, the union between the two lines creates the parallelogram (apply the first line to the second line), while the union between the two smaller circles is the larger circle. In this case, the radius of the first circle is added to the second one, enlarging it. This is called *dilation*,

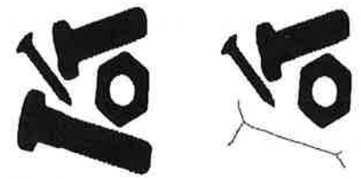


Figure 9.51 The binary image of a bolt and its stick (skeleton) representation.

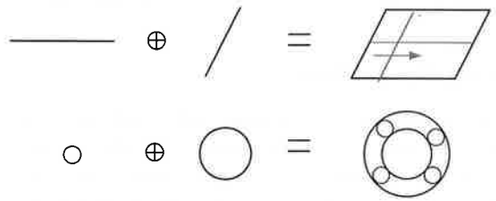


Figure 9.52 The union between two geometries creates *dilation*.

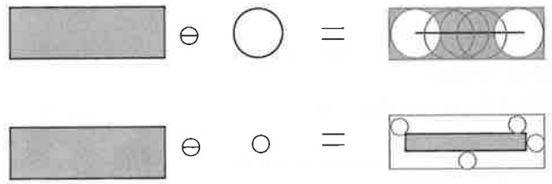


Figure 9.53 The subtraction of two geometries creates *erosion*.

shown with the symbol ⊕, and as we will soon discuss, it can be as little as one pixel added to the perimeter of a part in an image.

Similarly, in Figure 9.53, subtracting the second set from the first results in an eroded shape, therefore *erosion*, shown with the symbol ⊖, which as will be discussed later, may be as small as one pixel around the object. Similar combinations of dilation and erosion create other effects as follows.

Example 9.9

Figure 9.54 shows the effect of a union operation between two shapes. As shown, this union reduces the appearance of the peaks and valleys in the original shape. This is used for smoothing the jagged edges of shapes such as a bolt or gear.

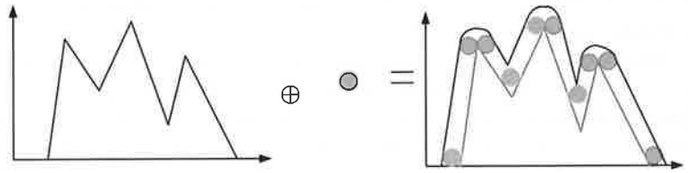


Figure 9.54 The result of the union of the two shapes reduces the appearance of the peaks and valley. ■

Example 9.10

Figure 9.55 shows the image of a plate with a small protrusion in it. In order to locate the protrusion, we may subtract a circle from the plate with a diameter slightly larger than the protrusion, add the circle to the result, and subtract the result from the original image. The remaining object is only the protrusion. ■

The following operations are all based on the previously mentioned operations.

9.18.1 Thickening Operation

A 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 shown in Figure 9.56, the thickening operation reduced the appearance of the threads of the bolts. This is a very useful operation 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,

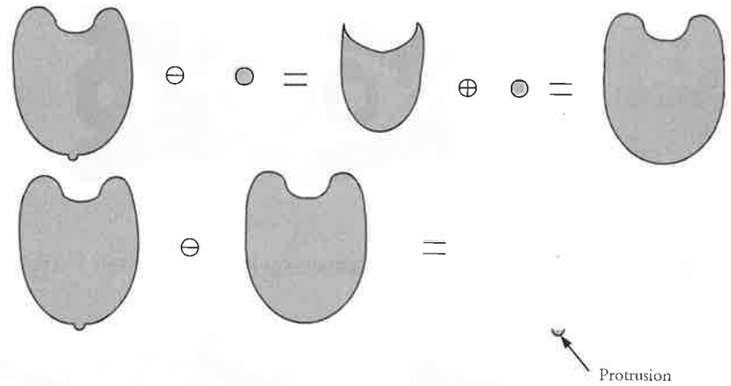


Figure 9.55 The application of union and subtraction operations for locating the protrusion.

Figure 9.56 The threads of the bolts are removed by a triple application of a thickening operation, resulting in smooth edges.

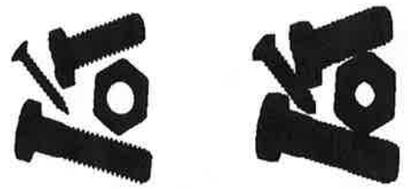
as we will see later. Figure 9.56 shows the effect of three rounds of thickening operations on the threads of the bolts.

9.18.2 Dilation

In dilation, the background pixels that are 8-connected to the foreground (object) are changed to foreground. As a result, effectively, a layer is added to the object every time the process is implemented. Due to the fact that dilation is performed on pixels that are 8-connected to the object, repeated dilations can change the shape of the object. Figure 9.57 (b) is the result of five dilation operations on the objects in (a). As you can see, due to this dilation, the four objects have bled into one piece. With additional applications of dilation, the four objects, as well as the disappearing hole, can become one solid piece, which can no longer be recognized.

9.18.3 Erosion

In this operation, 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 it is performed. Figure 9.58(b) shows the effect of 3 repetitions of the erosion operation on



(a) (b)

Figure 9.57 Effect of dilation operations. Here, the objects in (a) were subjected to five rounds of dilation (b).

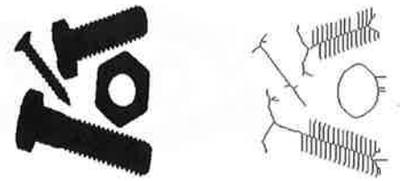
(a) (b) (c)

Figure 9.58 Effect of erosion operation on objects with (b) 3 and (c) 7 repetitions.

the binary image in (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 7 repetitions, where one bolt is completely lost and the nut will soon disappear. Erosion can eventually remove all objects. This means that if the reversing operation of dilation, which will add 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 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 we want to identify the largest object in an image, successive erosions will eliminate all other smaller objects before the largest object is eliminated. Therefore, the object of interest can be identified.

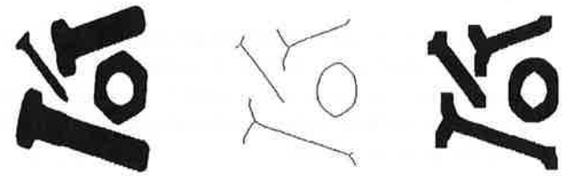
9.18.4 Skeletonization

A skeleton is a stick representative of an object where all thicknesses have been reduced to one pixel at any location. Skeletonization is a variation of erosion. Whereas in erosion, the thickness 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. Although in erosion the number of repetitions are chosen by the user, in skeletonization the process automatically continues until all thicknesses are 1 pixel (the program stops when no new changes are made as a result of the operation). The final result of



(a) (b)

Figure 9.59 The effect of skeletonization on an image without thickening. The threads of the bolts have resulted in whiskers.



(a) (b) (c)

Figure 9.60 The skeleton of the objects in (a) after the application of thickening operation results in a clean skeleton (b). Part (c) is the dilated image of the skeletons.

skeletonization is a stick figure (skeleton) of the object, which is a good representation of the object, sometimes much better than the edges. Figure 9.59(b) shows the skeleton of the original objects in (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 9.60 shows the same objects that are thickened to eliminate the threads, resulting in a clean skeleton. Figure 9.60(c) is the result of dilating the skeleton 7 times. As can be seen, the dilated objects are not the same as the original objects. Notice how the smaller screw appears as big as the bigger bolts.

Although dilation of a skeleton will also result in a shape different from the original object, skeletons are very useful in object recognition since they are generally a better representation of an object than others. The stick representation of an object can be compared to the available *a priori* knowledge of the object for matching.

9.18.5 Open Operation

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

9.18.6 Close Operation

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

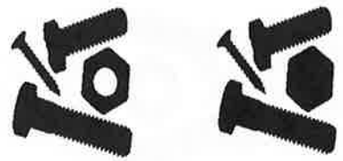


Figure 9.61 As a result of a fill operation, the hole in the nut is filled with foreground pixels, thus eliminating the hole.

9.18.7 Fill Operation

Fill operation fills the holes in the foreground (object). In Figure 9.61, the hole in the nut is filled with foreground pixels until it is eliminated.

For information on other operations, refer to vision systems manufacturers' references. Different companies include other operations to make their software unique. These operations can be used as available.

9.19 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 a gray image is a multilayer, three-dimensional image, where the light areas are peaks and the dark areas are valleys. The mask will be applied to the image by moving it from pixel to pixel. Where the mask matches the gray values in the image, there are no changes 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 following sections.

9.19.1 Erosion

In this case, each pixel will be replaced by the value of the darkest pixel in its 3 × 3 neighborhood, known as a Min Operator, effectively eroding the object. Of course, the result is dependent on which cells in the mask are 0 or 1. It removes light bridges between dark objects.

9.19.2 Dilation

In this case, each pixel will be replaced by the value of the lightest pixel in its 3 × 3 neighborhood, known as a Max Operator, effectively dilating the object. Of course, the result is dependent on which cells in the mask are 0 or 1. It removes dark bridges between light objects.

9.20 Image Analysis

Image analysis is a collection of operations and techniques used to extract information from images. This includes object recognition, feature extraction, analysis of 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 calculation of position and orientation of objects.

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. In the following sections, some of these techniques are discussed.

9.21 Object Recognition by Features

Objects in an image may be recognized by their features. These features may include, but are not limited to: gray level histograms; morphological features such as area, perimeter, number of holes, and others; eccentricity; cord length; and moments. In many cases, the information extracted is compared to *a priori* information about the object, which may be in a look-up table. For example, suppose two objects are present in the image, one with two holes and one with one hole. 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) to a look-up table, it is possible to determine what each of the two parts are. In another example, assume a moment analysis is performed for a known object and the moment, relative to an axis, is calculated at many angles and the data is stored in a look-up table. Later, when the moment of the part in the image is calculated relative to the same axis and is compared to the look-up table, the angle of the part in the image can be estimated.

The following is a discussion of a few techniques and different features that may be used for object recognition.

9.21.1 Basic Features Used for Object Identification

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

(a) Gray Levels: Average, maximum, or minimum gray levels may be used to identify different parts or objects in an image. As an example, assume there are three parts in an image, each one with a different color or texture. The colors and textures will create different gray levels in the image. If the average, maximum, or minimum gray levels of the objects are found (e.g., through histograms mapping), the objects can be recognized by comparison of this information. In other cases, even the presence of one particular gray level may be enough to recognize a part.

(b) Perimeter, area, diameter, number of holes, and other similar morphological characteristics may be used for object identification. The perimeter of an object may be found by first applying an edge detection routine, and then, by counting the number of pixels on the perimeter. The Left-Right search technique of section 9.13 can also be used to calculate the perimeter by counting the pixels that are on the path in an accumulator. Area can be calculated by region growing techniques. Moment equations can also be used, as will be discussed later. Diameter for noncircular objects

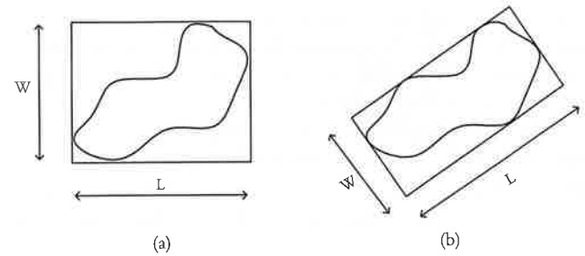


Figure 9.62 (a) Aspect ratio of an object, (b) minimum aspect ratio.

is the maximum distance between any two points on any line that crosses the identified area of the object.

(c) Aspect Ratio: Aspect ratio is the width to length ratio of an enclosing rectangle about the object, as shown in Figure 9.62. All aspect ratios are sensitive to orientation, except the minimum aspect ratio. Therefore, the minimum aspect ratio is usually used to identify objects.

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

$$1.\ \text{Thinness} = \frac{(\text{perimeter})^2}{\text{area}} \tag{9.21}$$

$$2.\ \text{Thinness} = \frac{\text{diameter}}{\text{area}} \tag{9.22}$$

(e) Moments: Due to their importance, moments are discussed in the next section.

9.21.2 Moments

The moment of an object within an image is:

$$M_{a,b} = \sum_{x,y} x^a y^b I_{x,y} \tag{9.23}$$

where $M_{a,b}$ is the moment of the object with a and b indices, x *and* y are the coordinates of each pixel raised to the power of a and b, and $I_{x,y}$ is the intensity of the pixel, as in Figure 9.63. If the image is binary, the intensities are either 1 (or on) for the object and 0 (off) for the background, and therefore, only the pixels that are turned on are considered. In gray images, the intensities may vary greatly, and consequently, the value of the moment may be exceedingly influenced by gray values. As a result, although it is mathematically possible to apply moment equations to a gray image, it is not practical or useful unless other rules are engaged, too. For binary images, $I_{x,y}$ is either 0 or 1; therefore, considering only the on-pixels in the image, Equation (9.23) simplifies to:

$$M_{a,b} = \sum_{x,y} x^a y^b \tag{9.24}$$

To calculate the moments, first determine whether or not each pixel belongs to the object (is turned on); if so, raise the coordinates of the location of the pixel to the given

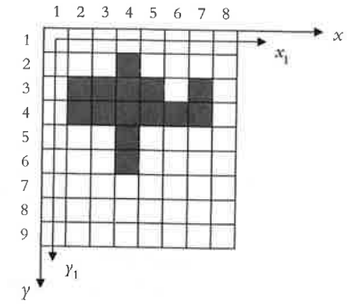


Figure 9.63 Calculation of the moment of an image. For each pixel that belongs to the object, the coordinates of the pixel are raised to the powers indicated by the moment's indices. The summation of the values thus calculated will be the particular moment of the image.

values of a and b. The summation of this operation over the entire image is the particular moment of the object with a and b indices.

$M_{0,0}$ is the moment of the object with $a = 0$ and $b = 0$. This means the x and y coordinate values of all on-pixels are raised to a power of 0. $M_{0,2}$ means all x values are raised to the power of 0 and all y values are raised to the power of 2, and so on. All combinations of values between 0 and 3 are common.

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. However, note 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 on the second row is pixel (2,4). The x distance of the pixel from the x_1-y_1 coordinate frame will be 3, whereas the same coordinate from the x-y coordinate is 4 (or more accurately, 3.5 pixels). As long as the same distances are used consistently, the choice is not important.

Based on the above, since all numbers raised to the power of 0 are equal to 1, all x^0 and y^0s are equal to 1. Therefore, the $M_{0,0}$ moment is the summation of all on-pixels, which is the area of the object. This moment can be used to determine the nature of an object and to distinguish it from others that have a different area. Obviously, the $M_{0,0}$ moment can also be used to calculate the area of an object within an image.

Similarly, $M_{0,1}$ is $\sum x^0 y^1$ for all on-pixels, or the summation of $1 \times y$ values, which is the summation of the y-coordinates of all on-pixels from the x-axis. This is similar to the first moment of the area relative to the x-axis. Therefore, the location of the center of the area relative to the x-axis can be calculated by:

$$\bar{y} = \frac{\sum y}{area} = \frac{M_{0,1}}{M_{0,0}} \tag{9.25}$$

So, by simply dividing the two moments, you may 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 is:

$$\bar{x} = \frac{\sum x}{area} = \frac{M_{1,0}}{M_{0,0}} \tag{9.26}$$

This way, an object may be located within an image regardless 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.

$M_{0,2}$ is $\sum x^0 y^2$ 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 9.63 varies significantly as the object rotates about its center. Suppose we calculate the moments of the area about an axis, say, the x-axis, at different orientations. Since each orientation creates a unique value, a look-up table that contains these values can later be used to identify the orientation of the object. Therefore, if a look-up table is prepared containing the values of the moments of inertia of the known object at different orientations, the subsequent orientation of the object can be estimated by comparing its second moment to the values in the look-up table. Of course, if the object translates within an image, its moments of inertia will also change, rendering it impossible to determine the orientation except in known locations. However, with a simple application of the parallel axes theorem, the second moments about the center of the area can be calculated. Since this measure is independent of the location, it can be used to determine the orientation of the object regardless of its location. Therefore, using the moment equations, the object, its location, and its orientation can be identified. In addition to identification of the part, the information can be used in conjunction with a robot controller to direct the robot to pick up the part and/or operate on it.

Other moments can also be used similarly. For example, $M_{1,1}$ represents the product of inertia of the area and can also be used for object identification. Higher-order moments such as $M_{0,3}, M_{3,0}, M_{1,2}$, and so on can also be used to identify objects and their orientation. Imagine two objects relatively similar in shape, as in Figure 9.64(a). It is possible that the second moments, areas, perimeters, or other morphological characteristics of the two objects may be similar or close to each other such that they may not be useful in object identification. In this case, a small difference between the two objects may be exaggerated through higher-order moments, making object identification possible. The same is true for an object with a small asymmetry (Figure 9.64(b)). The orientation of the object may be found by higher-order moments.

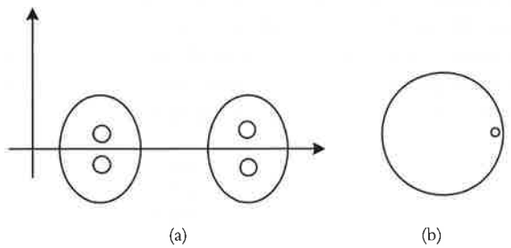


Figure 9.64 Small differences between objects or small asymmetry in an object may be detected using higher-order moments.

A moment invariant is a measure of an object based on its different moments, and is independent of its location, orientation, and scale factor. Therefore, the moment invariants may be used for object recognition and parts identification without regard to camera set-up, location, or orientation. There are seven different moment invariants, such as:

$$MI_1 = \frac{M_{0,0}M_{2,0} - M_{1,0}^2 + M_{0,0}M_{0,2} - M_{0,1}^2}{M_{0,0}^3} \tag{9.27}$$

Refer to Reference [2] for the other six moment invariants.

Example 9.11

For the simple object in a low-resolution image of Figure 9.65, calculate the area, center of the area, and second moments of area of the object relative to the x_1, y_1-axes.

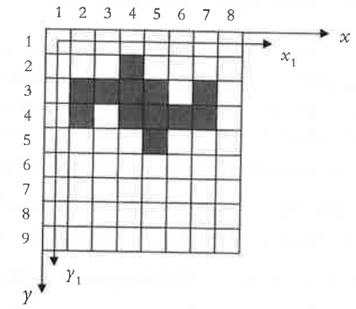


Figure 9.65 Image used for Example 9.11.

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

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

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

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

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

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

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

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 9.12

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 hollow, while the rectangles are 3×4 solids. Through guides, jigs, and brackets, we can be certain the objects are always parallel to the reference axes, as shown in Figure 9.66, and that

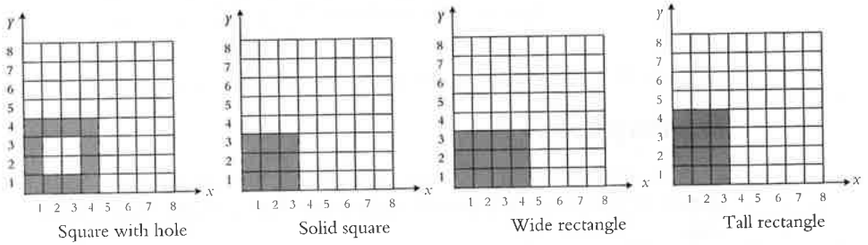


Figure 9.66 Image used for Example 9.12.

the lower left corners of the objects are always at pixel 1,1. We want to only use the moment equations 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. For this example, use the absolute coordinates of each pixel for distances from the corresponding axes.

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

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

As shown, 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 9.13

For the image of the screw in Figure 9.67, 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.

Solution: A macro program 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 rotation operations. Every time a part of an image is rotated, since every point in the image must be converted with a sine or cosine function, it changes slightly. Otherwise, as you can see, the results are consistent. For example, as

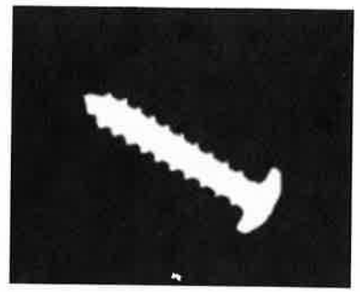


Figure 9.67 Image used for Example 9.13.

the part is rotated in place, the location of its center of area does not change. You also can see that the moment invariant is constant and, from the moment of inertia information about the centroid, the orientation can be estimated. This information can now be used to identify the object or to direct a robot controller to send the robot arm to the location, with proper orientation, to pick up the part.

	Horizontal	30°	45°	60°	Vertical
Area	3713	3747	3772	3724	3713
x-bar	127	123	121	118	113
y-bar	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

■

The following is the listing of the moments.macro program, written for Optimas. Although this program cannot directly be used with other software, it is listed here to show how simply a program can be developed to do similar operations. 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.

```
/*MOMENTS.MAC PROGRAM Written by Saeed Niku, Copyright 1998
This macro checks an active image within Optimas vision system. It 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 the data and display it. The DDE commands communicate the data between
Excel and Optimas macro program. If your number of coordinates is more than 20,000
pixels, you must change the 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");
```

9.21.3 Template Matching

Another technique for object recognition is model or template matching. If a suitable line drawing of the scene is found, the topological or structural elements such as total number of lines (sides), vertices, and interconnections can be matched to a model. Coordinate transformations such as rotation, translation, and scaling can be performed to eliminate

the differences between the model and the object resulted from position, orientation, or depth differences between them. This technique is limited by the fact that *a priori* knowledge of the object models is needed for matching. Therefore, 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.

9.21.4 Discrete Fourier Descriptors

Similar to a Fourier transform calculated for an analog signal, a Discrete Fourier Transform (DFT) of a set of discrete points (such as pixels) can also 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 DFT calculation is a set of frequencies and amplitudes in frequency domain that describe the spatial relationship of the points in question.[14]

To calculate the DFT of a set of points in a plane, assume the plane is a real-imaginary plane, such that each point is described by an $x + iy$ relationship. If the contour 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. Matching these frequencies with the frequencies found for possible objects in a look-up table may be used to determine the nature of the object. In one unpublished experiment, matching 8 frequencies yielded enough information about the nature of the object (an airplane). Matching 16 frequencies could determine the type of an airplane from a large class of planes. An advantage of this technique is that the Fourier transform can very simply be normalized for size, position, and orientation. 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 the object.

9.21.5 Computed Tomography (CT)

Tomography is a technique of determining the distribution of material density in the examined part. In computed tomography (CT), a three-dimensional image of the object's density distribution is reconstructed from a large number of two-dimensional images of the 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 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 is stored in a computer, and subsequently, is reduced to a three-dimensional image of the part's density distribution that is shown on a monitor.

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

9.22 Depth Measurement with Vision Systems

Extracting depth information from a scene is performed using two basic techniques. One is the use of range finders in conjunction with a vision system and image-processing techniques. In this combination, the scene is analyzed in relation to the information gathered by range finders about the distances of different portions of an environment or the location of particular objects or sections of the object. Second is the use of binocular or stereo vision similar to humans and animals. In this technique, either simultaneous images from multiple cameras or multiple images from one camera that moves on a track are used to extract depth information. As long as the scene does not change during this operation, the results will be the same as the use of multiple cameras. Since the location of the multiple (usually two) cameras in relation to any particular point in the scene is slightly different, each camera develops a slightly different image. By analyzing and measuring the differences between the two scenes, depth information can be extracted.

9.22.1 Scene Analysis versus Mapping

Scene analysis refers to the analysis of images developed by a camera or other similar devices in which a complete scene is analyzed. In other words, the image is a complete replica of the scene within the resolution limit of the device where all the details of the scene are included in the image. 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, segmented by edge detection or thresholding, the part isolated by region growing, and then identified by extracting its features and comparing them to a template or look-up table. On the other hand, 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 relate to the relative position of points on the object at discrete locations. Since the image is already sliced, less processing is required in 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.

9.22.2 Range Detection and Depth Analysis

Range measurement and depth analysis is performed using many different techniques such as active ranging,[20] stereo imaging, scene analysis, or specialized lighting. Humans use 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 using details such as the changing size of similar elements, vanishing lines, shadows, and changing intensity of textures and shades. Since many artificial intelligence techniques are based on, and studied for, understanding of the way humans do things, a number of depth measurement techniques are designed after similar human operations.[15]

9.22.3 Stereo Imaging

An image is the projection of a scene into the image plane through an ideal lens. Therefore, every point in the image will correspond to a certain point in the scene.

However, the depth information of the point is lost in this projection and cannot simply be retrieved from a single image. If two images of the same scene are taken, the relative depth of different points from the image plane can be extracted by comparing the two images; the differences represent the spatial relationship between different points.[16,17] Humans do the same, automatically, by combining the two images and forming a three-dimensional image.[18,19,21] The stereo image used for depth measurement is considered 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. To determine the point-pairs in the two images that correspond to the same point in the scene. This is called correspondence or disparity of the point-pair. This is a difficult operation since some points in one image may not be visible in another, or since due to perspective distortion, sizes and spatial relationships may be different in the two images.
2. To determine the depth or location of the point on the object or in the scene by triangulation or other techniques.

Generally, if the two cameras (or the relative locations of a single camera used twice to get two images of a nonmoving, 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 9.68. In each case, the two cameras will see the marks as shown in (a) and (b). Although the locations of the two marks are different, the cameras will see them similarly. As a result, the marks may be located wrongly (unless additional information such as vanishing lengths are also considered).

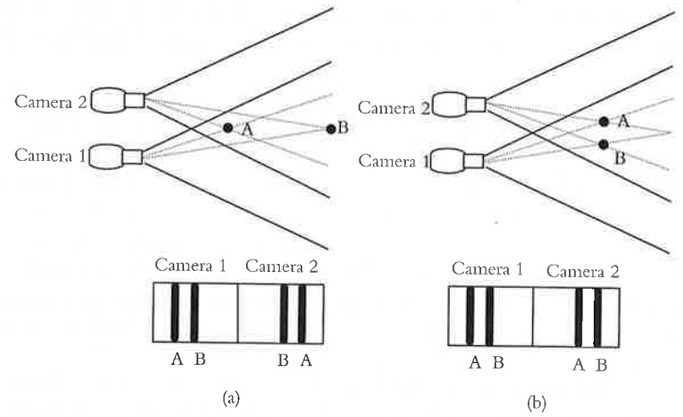


Figure 9.68 Correspondence problem in stereo imaging.

The accuracy of depth measurement in stereo imaging is dependent on the angle between the two images, and therefore, the disparity. 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 used in the Stanford Cart, where 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]

9.22.4 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 orientations. One detail 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 shades 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.

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

9.23 Specialized Lighting

Another possibility for depth measurement is utilizing special lighting techniques that yield a specific result. The specialized result can be used for extracting depth information. Most of these techniques are designed for industrial applications, where specialized lighting is possible and the environment is controlled. The following is the theory behind one technique.

If a strip (narrow plane) of light is projected over a flat surface, it will generate a straight line in relation with 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 light, a curved or broken line will be observed (Figure 9.69). By analyzing the reflected light, we can extract information about the shape of the object, its location, and orientation. In certain systems, two strips of light are used, such that in the absence of any object on the table, the two strips intersect exactly on the surface. When an object is present, the two strips of light develop two reflections. The reflections are picked up by a camera and depth information is calculated and reported. A commercial system based on this technique is called CONSIGHT™.

A disadvantage of this technique is that only information about the points that are lit can be extracted. Therefore, in order to have information about the complete image, it is necessary to scan the entire object or scene.

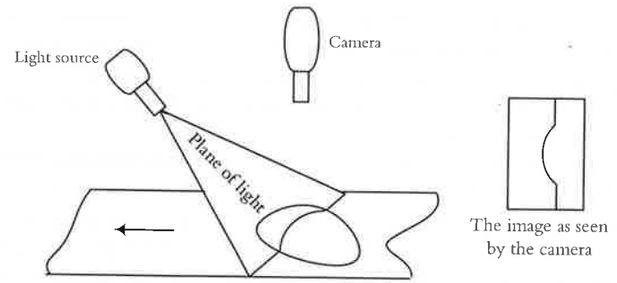


Figure 9.69 Application of specialized lighting in depth measurement. A plane of light strikes the object. The camera, located at a different angle than the plane of light, will see the reflection of the light plane on the object as a curved line. The curvature of the line is used to calculate depth.

9.24 Image Data Compression

Electronic images contain large amounts of information, and therefore, require data transmission lines with large bandwidth capacity. The requirements for spatial resolution, number of images per second, and 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 capability, including transmission over the Internet.

The following are some techniques that accomplish this task. Although there are many different techniques of data compression, only some of them directly relate 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 frames) methods.

9.24.1 Intraframe Spatial Domain Techniques

Pulse Code Modulation (PCM) is a popular technique of data transmission in which the 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 8-bit image quantizer). Certain applications (space and medical) use higher resolutions such as 2^{10} or 2^{12}.

In a technique called Pseudorandom Quantization Dithering,[22] random noise is added to the pixels' gray values in order to maintain the same quality while reducing the number of bits. This is done to prevent contouring, which happens when the number of bits of a quantizer is reduced (see section 9.5 and Figure 9.8). These contours can be broken up by adding a small amount of broadband pseudorandom, 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. A proper amount of noise will enable the system to have the same apparent resolution while the number of bits is reduced significantly.

Another technique of data compression is halftoning. In halftoning, 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 1-bit binary quantizer into a black or 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 which 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 in multiple images (e.g., TV news sets). Therefore, the data transmission can be significantly reduced if only the changes between successive images are transmitted. To do this, 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. This value is transmitted by the system to update the previous image. If in an image many pixels remain the same, the innovations are few and transmission is reduced.

Example 9.14

In a similar attempt to reduce the amount of data transmission in space by the Voyager 2 Spacecraft, its computers were reprogrammed, while in space, to use a differential coding technique. At the beginning of its space travel, the Voyager's system was designed to transmit information about every pixel, at a 256 gray level scale. This took 5,120,000 bits to transmit one single image, not including error detection and correction codes, which were about the same length. Beginning with the Uranus fly-by, the system was reprogrammed to only send the difference between successive pixels rather than the absolute brightness of the pixels. Consequently, if there were no differences between successive pixels, no information would be transmitted. In scenes such as in space, where the background is essentially black, there are many pixels that are similar to their neighbors. This reduced data transmission by about 60%.[23] Other examples of fixed background information include theatrical sets and industrial images. ■

In Constant Area Quantization (CAQ),[24,25] data transmission is reduced by transmitting fewer pulses at lower resolution in low contrast areas compared to high contrast areas. This, in effect, is taking advantage of the fact that higher contrast areas have higher frequency content and require more information transmission than the lower contrast areas.

9.24.2 Interframe Coding

These methods take advantage of the redundant information that exists between successive images. The difference between these 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 to be transmitted.

A simple technique to achieve this is to use a frame memory at the receiver. The frame memory will hold an image and continually show it at the display. When information about any pixel is changed, the corresponding location in the frame memory is updated. As a result, the rate of transmission is significantly reduced. The disadvantage of this technique is that in the presence of rapidly moving elements, flickering may happen.

9.24.3 Compression Techniques

Two general methods are used for data compression. In one method (such as in zip archives) called *lossless compression*, codes are assigned to repetitive words, phrases, or values in order to reduce the size of the data file. As the name implies, in these methods no data is lost, and consequently, the original file may be reconstructed without any change or loss. However, the level of savings or compression in chromatic or achromatic gray images is not large because in these images pixels rarely have repeating patterns. These methods, however, are more useful in particular situations. For example, in a binary file, large areas (blobs) have similar values. If the data is presented line by line (such as in a facsimile file scanned line by line), the data may be compressed by coding the length of on-off sets of pixels rather than the value of each individual pixel. Therefore, many pixels with similar values can be represented by only specifying the starting point of each section and its length.

The second category covers methods that compress image data by reducing the information, and therefore, are called *lossy compression*, including the popular JPEG (Joint Photographers Expert Group) compression.[13] Although we will not discuss the elaborate sequence of steps taken in order to compress the data, it should be mentioned that a lot of information is lost during this process, although a much smaller file is generated when an image is saved or converted to JPEG format. However, unless the original detailed data is needed for other purposes or a picture must be zoomed in to extract information, the human eye may not recognize the difference as much.

9.25 Color Images

White light can be decomposed into a rainbow of colors that span the range of 400 to 700 nm wavelengths. Although it is rather difficult to subscribe an exact value to any particular hue, the primary colors of light are thought to be red, green, and blue (RGB). Theoretically, all other hues and color intensities can be recreated by mixing varying levels of the primary color lights, although in reality, the recreations are not truly accurate. However, in images, most colors can be recreated using RGB.

To capture color images, filters are used to separate the light into these three subimages and each one is captured, sampled, and quantized individually, therefore creating three image files. To recreate color images, the screen is composed of three sets of pixels, interlaced sequentially (RGBRGB . . .). Each set of pixels is recreated individually, but simultaneously. Due to the limited spatial resolution of our eyes, we tend to mix the three images together and perceive color images. However, as far as image processing is concerned, a color image is in fact a set of three images, each representing the intensities of the three primary colors of the original image.

To convert a color (chromatic) image into a black and white (achromatic) image, the intensities of the individual colored files must be converted into gray values. One method

to do this is to take the average values of the three files for the same pixel location and to use that as a substitute for the gray value. Therefore, the histogram of a gray image shows the same exact values for all three channels of RGB. For more information about the image processing of colored images please refer to other resources such as References 2 and 13.

9.26 Heuristics

Heuristics is a collection of rules of thumb developed for semi-intelligent systems in order to enable them to select a predetermined decision from a list based on the current situation. Heuristics is used in conjunction with mobile robots, but has applications in many fields.

Consider a mobile robot that is supposed to navigate through a maze. Imagine the robot starts at a point and is equipped with a sensor that 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 the first rule is that when encountering an obstacle, the robot should turn left. As the robot continues, it may reach another wall, 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, this 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 alternate route can be found. As you can 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 is 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 is not true intelligence since the controller is not really making decisions but merely selecting from a set of decisions that have already been made. If a new situation is encountered that is not in the rules base, the controller will not know how to respond.[26]

9.27 Applications of Vision Systems

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

Suppose that in an automatic manufacturing setting, a circuit board is to be manufactured. One important part in this operation is the inspection of the board at different states before and after certain operations. A common method is to set up a cell where an image of the part is taken and subsequently modified, improved, and altered. The processed image is compared to a look-up image. If there is a match, the part is accepted. Otherwise, either the part is rejected or is repaired. These image-processing and analysis operations are generally made up of the processes discussed earlier. Most commercial vision systems have embedded routines that can be called from a macro program, making it very easy to set up a system.

Vision systems have been used for many applications, for example, for locating radioactive pucks,[27] random bin picking,[28] creating an automated brake inspection system,[29] measuring robotic motions and external objects,[30] food inspection such as texture of cookies and consistency of packaging,[31] creating adaptive behavior for mobile robots,[32] analyzing the health of agricultural crops,[33] and many others.

In navigation, the scene is usually analyzed for finding acceptable pathways, obstacles, and other elements that confront the robot.[34] In some operations, the vision system sends its information to an operator, who controls the motions from a distance. This is very common in telerobotics as well as in space applications.[35] In some medical applications, the surgeon guides the device, whether a surgical robot or a small investigative, exploratory device such as an angiogram, through its operations.[36] 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,[37,38] an inexpensive laser diode was mounted next to a camera. The projected laser light was captured by the camera and was used to both 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 dot and separate it from the rest of the scene, a histogram and thresholding operation was used. Subsequently, the circle was identified and skeletonized until only the center of the circle 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, the fundamental knowledge about vision systems enables you to proceed with an application and adapt to your application what vision systems have to offer.

9.28 Design Project

There are many inexpensive digital cameras on the market that can be used to create a simple vision system. They 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 the software to capture and digitize an image. Standard still and video cameras can also be used for capturing images. In this case, although you can capture an image for later analysis, due to the additional steps of downloading the image from the camera to the computer, the image is not available for immediate use.

Additionally, many simple programs such as Adobe's Photoshop™ have many routines similar to what we have discussed in this chapter. Additional routines may be developed using common computer languages such as C. Many other routines may be downloaded from the public domain. The final product will be a simple vision system that can be used to perform vision tasks. This may be done independently, or in conjunction with your 3-axis robot, and can include routines for parts identification and pick up, the development of mobile robots, and many other similar devices.

Most images shown in this chapter were captured and processed by standard digital cameras and the vision systems in the Mechanical Engineering Robotics laboratory at Cal Poly, including MVS909™ and Optimas™ 6.2 vision systems and Photoshop™.

You may also develop your own simple vision system using other programming languages and systems that can handle an image file. This includes Excel™, LabView™, and other development systems.

Summary

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

There are countless routines that can be used for a variety of different purposes. Most of these types of 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 mostly concentrated on these types of techniques, which enable you to adopt, develop, and use other routines and techniques for other applications. The advances in technology have also created tremendous opportunities in this area. There is no doubt that this trend will continue in the future as well.

References

1. Doudoumopoulos, Roger, "On-Chip Correction for Defective Pixels in an Image Sensor," NASA Tech Briefs, May 2000, p. 34.
2. Gonzalez, R. C., Richard Woods, "Digital Image Processing," Prentice Hall, New Jersey, 2002.
3. Low, Adrian, "Introductory Computer Vision and Image Processing," McGraw-Hill, 1991.
4. Horn, B. K. P., "Robot Vision," McGraw-Hill, 1986.
5. Hildreth, Ellen, "Edge Detection for Computer Vision System," *Mechanical Engineering*, August 1982, pp. 48–53.
6. Olson, Clark, "Image Smoothing and Edge Detection Guided by Stereoscopy," NASA Tech Briefs, September 1999, pp. 68–69.
7. Groover, M. P., et al. "Industrial Robotics, Technology, Programming, and Applications," McGraw-Hill, 1986, p. 177.
8. Hough, P. V. C., *A Method and Means for Recognizing Complex Patterns*, U.S. Patent 3,069,654, 1962.
9. Illingworth, J., J. Kittler, "A Survey of the Hough Transform," Computer Vision, Graphics, and Image Processing, Vol. 44, 1988, pp. 87–116.
10. Kanade, T., "Survey; Region Segmentation: Signal vs. Semantics," Computer Graphics and Image Processing, Vol. 13, 1980, pp. 279–297.
11. Snyder, Wesley, "Industrial Robots: Computer Interfacing and Control," Prentice Hall, 1985.
12. Haralick, Robert M., L. G. Shapiro, "Computer and Robot Vision," Volume I, Addison Wesley, MA, 1992.
13. Russ, John C., J. C. Russ, "Introduction to Image Processing and Analysis," CRC Press, 2008.
14. Gonzalez, Rafael, P. Wintz, "Digital Image Processing," 2nd Edition, Addison-Wesley, Reading, Mass., 1987.
15. 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.
16. Nevatia, R., "Machine Perception," Prentice Hall, New Jersey, 1982.

17. Fu, K. S., Gonzalez, R. C., Lee, C. S. G., "Robotics; Control, Sensing, Vision, and Intelligence," McGraw-Hill, 1987.
18. Marr, D., T. Poggio, "A Computational Theory of Human Stereo Vision," Proceedings of the Royal Society, London, B204, 1979, pp. 301–328.
19. Marr, D., "Vision," Freeman and Co., 1982.
20. Pipitone, Frank, T. G. Marshall, "A Wide-field Scanning Triangulation Rangefinder for Machine Vision," The International Journal of Robotics Research, Vol. 2, No. 1, Spring 1983, pp. 39–49.
21. Moravec, H. P., "Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover," Stanford Artificial Intelligence Laboratory Memo, AIM-340, September 1980.
22. 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.
23. Goldstein, Gina, "Engineering the Ultimate Image, The Voyager 2 Mission," *Mechanical Engineering*, December 1989, pp. 30–36.
24. 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.
25. 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.
26. Chattergy, R., "Some Heuristics for the Navigation of a Robot," The International Journal of Robotics Research, Vol. 4, No. 1, Spring 1985, pp. 59–66.
27. Wilson, Andrew, Editor, "Robot Vision System Locates Radioactive Pucks," Vision Systems Design, May 2002, pp. 7–8.
28. "Using Vision to Enable Robotic Random Bin Picking," Imaging Technology, June 2008, pp. 84–86.
29. "Creating an Automated Brake Inspection System with Machine Vision," Imaging Technology, June 2008, pp. 88–90.
30. "Vision System Measures Motions of Robots and External Objects," NASA Tech Briefs, November 2008, pp. 24–26.
31. Thilmany, Jean, "Accessible Vision," *Mechanical Engineering*, July 2009, pp. 42–45.
32. "Adaptive Behavior for Mobile Robots," NASA Tech Briefs, August 2009, pp. 52–53.
33. "Imaging System Analyzes Crop Health," Defense Tech Briefs, August 2009, pp. 32– 33.
34. "Vision-Based Maneuvering and Manipulation by a Mobile Robot," NASA Tech Briefs, March 2002, pp. 59–60.
35. Ashley, Steven, Associate Editor, "Roving Other Worlds by Remote," *Mechanical Engineering*, July 1997 pp. 74–76.
36. Hallett, Joe,Contributing Editor, "3-D Imaging Guides Surgical Operations," Vision Systems Design, May 2001, pp. 25–29.
37. 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.
38. Niku, S. B., "Camera Calibration and Resetting with Laser Light," proceedings of the 3rd International Conference on Mechatronics and Machine Vision in Practice, September 1996, Guimaraez, Portugal, Vol. 2, pp. 223–226.

Problems

Note: 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 numbers 0 and 1 or gray-level numbers in the file. In a binary image 0 represents off, dark or

background pixel, while 1 represents on, light, or object pixels. In gray images, each pixel is represented by a corresponding greyness level 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 result of the operation, and m and n are the row and column indices of the resulted file.

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

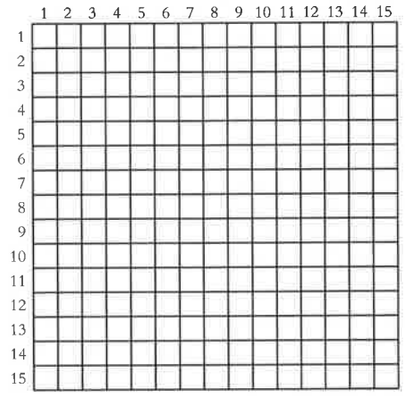


Figure 9.70 A blank image grid.

9.1. Calculate the necessary memory requirement for a still color image from a camera with 10 megapixels at:

- 8-bits per pixel (256 levels)
- 16-bits per pixel (65,536 levels)

9.2. Consider the pixels of an image, with values as shown in Figure P.9.2 , as well as a convolution mask with the given values. Calculate the new values for the given pixels.

4	7	1	8
3	2	6	6
5	3	2	3
8	6	5	9

1	0	1
1	1	1
1	0	1

Figure P.9.2

9.3. Consider the pixels of an image, with values as shown in Figure P.9.3, as well as a convolution mask with the given values. Calculate the new values for the given pixels. Substitute 0 for negative gray levels. What conclusion can you make from the result?

5	5	5	7	7	8
5	5	5	8	8	9
10	10	10	10	10	10
8	8	8	9	9	8
8	8	8	9	9	10

1	0	1
0	-4	0
1	0	1

Figure P.9.3

9.4. Repeat Problem 9.3, but substitute the absolute value of negative gray levels. What conclusion can you make from the result?

9.5. Repeat Problem 9.3, but apply the mask of Figure P.9.5 and compare your results with Problem 9.3. Which one is better?

-1	0	-1
0	4	0
-1	0	-1

Figure P.9.5

9.6. Repeat Problem 9.3, but apply the mask of Figure P.9.6 and compare your results with Problem 9.3. Which one is better?

0	1	0
1	-4	1
0	1	0

Figure P.9.6

9.7. Repeat Problem 9.3, but apply the mask of Figure P.9.7 and compare your results with Problem 9.3. Which one is better?

1	1	1
1	-8	1
1	1	1

Figure P.9.7

9.8. An image is represented by values shown below.

(a) Find the value of pixel 2c when mask 1 is applied.

(b) Find the value of pixel 3b when mask 2 is applied.

(c) Find the values of pixels 2b and 3c when a 3×3 median filter is applied.

(d) Find the area of the major object that results when a threshold of 4.5 is applied based on a +4-connectivity (start at the first on-pixel).

	a	b	c	d	e
1	4	9	6	2	6
2	4	3	7	4	6
3	6	2	6	5	3
4	1	6	5	9	2
5	2	8	4	4	7

1	1	1
1	1	1
1	1	1

Mask 1

0	1	0
1	-4	1
0	1	0

Mask 2

Figure P.9.8

9.9. An image is represented by values shown below.

(a) Find the value of pixel 3b when mask 1 is applied.

(b) Find the values of pixels 2b, 2c, 2d when mask 2 is applied.

(c) Find the value of pixel 3c when a 5 × 5median filter is applied.

(d) Find the area of the major object that results when a threshold of 4.5 is applied based on a ×4-connectivity (start at the first on-pixel).

	a	b	c	d	e
1	8	9	6	2	5
2	4	6	2	4	6
3	6	7	5	6	5
4	1	10	5	9	4
5	2	8	4	3	2

0	1	0
1	4	1
0	1	0

Mask 1

1	1	1
-2	-2	-2
1	1	1

Mask 2

Figure P.9.9

9.10. Write a computer program for the application of a 3×3 averaging convolution mask unto a 15×15 image. Refer to the note on page 415 for more information.

9.11. Write a computer program for the application of a 5×5 averaging convolution mask unto a 15×15 image. Refer to the note on page 415 for more information.

9.12. Write a computer program for the application of a 3×3 high-pass convolution mask unto a 15×15 image for edge detection. Refer to the note on page 415 for more information.

9.13. Write a computer program for the application of an n×n convolution mask unto a k×k image. Refer to the note on page 415 for more information. You should write the routine such that the user can choose the size of the mask and the values of each mask cell individually.

9.14. Write a computer program that will perform the Left-Right search routine for a 15×15 image. Refer to the note on page 415 for more information.

9.15. Using the Left-Right search technique, find the outer edge of the object in Figure P.9.15:

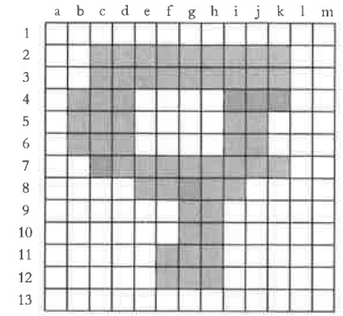


Figure P.9.15

9.16. The x and y coordinates of 5 points are given as (2.5, 0), (4,2), (5,4), (7,6), and (8.5,8). Using the Hough transform, determine which of these points form a line and find its slope and intercept.

9.17. 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, must continue searching for another nuclei until all object pixels have been checked. Refer to the note on page 415 for more information.

9.18. 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, must continue searching for other nuclei until all object pixels have been checked. Please refer to the note on page 415 for more information.

9.19. Using +4 connectivity logic and starting from 1,a pixel, write the sequence of pixels in correct order that will be detected by a region growing routine for the object in Figure P.9.19:

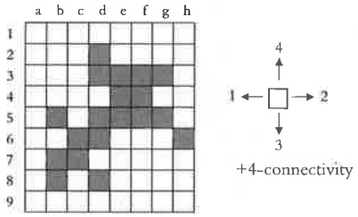


Figure P.9.19

9.20. Using ×4 connectivity logic and starting from 1,a pixel, write the sequence of pixels in correct order that will be detected by a region growing routine for the object in Figure P.9.20:

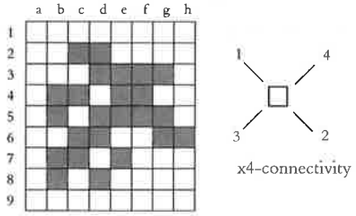


Figure P.9.20

9.21. Find the union between the two objects in Figure P.9.21.

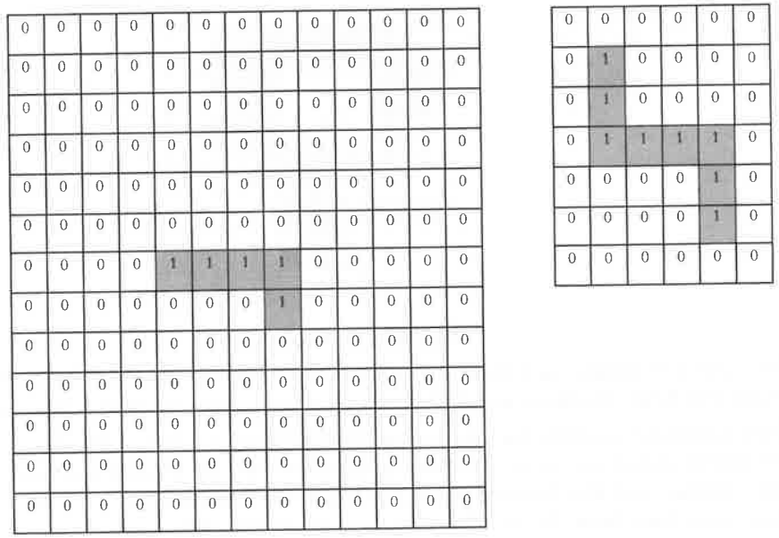


Figure P.9.21

9.22. Apply a single-pixel erosion based on 8-connectivity on the image of Figure P.9.22.

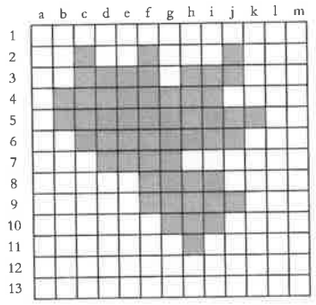


Figure P.9.22

9.23. Apply a one-pixel dilation to the result of Problem 9.22 and compare your result to Figure P.9.22.

9.24. Apply an open operation to Figure P.9.24.

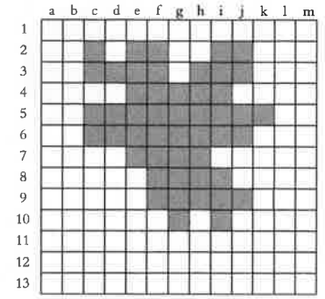


Figure P.9.24

9.25. Apply a close operation to Figure P.9.24.

9.26. Apply a skeletonization operation to Figure P.9.26 .

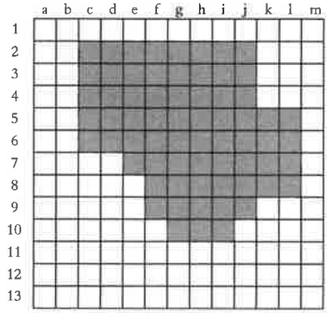


Figure P.9.26

9.27. Write a computer program in which different moments of an object in an image can be calculated. The program should ask you for moment indices. The results may be reported to you in a new file, or may be stored in memory. Refer to the note on page 415 for more information.

9.28. Calculate the $M_{0,2}$ moment for the result of Problem 9.8(d) based on +4-connectivity.

9.29. For the binary image of a key in Figure P.9.29, calculate the following:

- Perimeter, based on the Left-Right search technique.
- Thinness, based on $P^2/Area$.
- 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 (2,2).

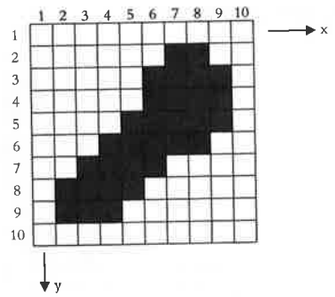


Figure P.9.29

9.30. Using moment equations, calculate $M_{0,2}$ and $M_{2,0}$ about the centroidal axes of the part in Figure P.9.30.

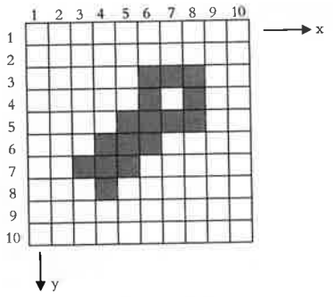


Figure P.9.30

9.31. Using moment equations, calculate $M_{0,2}$ and $M_{2,0}$ about the centroidal axes of the part in Figure P.9.31.

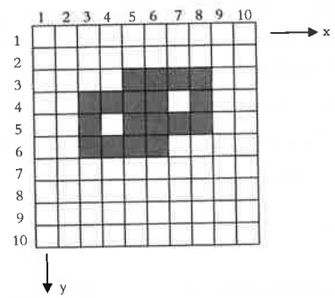


Figure P.9.31

CHAPTER 10

Fuzzy Logic Control

10.1 Introduction

"Tuesday, October 26, 1993 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 underlined words:

"Tuesday, October 26, 1993 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 can see, a number of "descriptors" are used in this statement to describe 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? If you live in San Luis Obispo, even 80°F may be too warm. Then, as you can see, this description of the temperature is, in fact, fuzzy. It is not very clear what the temperature may have been. And 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, read the paragraph once again and see how many other fuzzy descriptions are used in addition to the ones we are discussing.