(b) Suppose we have an image of 5 cm × 5 cm, and we wish to sample it with uniform sampling on a Cartesian grid at 512 × 512 points. What is a reasonable size of r for the aperture? What is the effect of choosing a very small r? a very large r?

7.15. In television broadcasting, a horizontal line time is the time interval during which the scanner traces one horizontal line and retraces to the beginning of the next horizontal line.

(a) What is the approximate horizontal line time in an NTSC television system? Assume that the time it takes the scanner to retrace from the end of one field to the beginning of the next is negligible.

(b) Approximately 16% of the horizontal line time is used to allow the retrace from the end of one horizontal line to the beginning of the next horizontal line. The picture intensity information is blanked out during this period, which is called a *horizontal blanking period.* What is the approximate time used as horizontal blanking periods during the period the scanner scans one complete field in an NTSC television system?

7.16. In NTSC television broadcasting, the frame rate used is 30 frames/sec. Each frame consists of 525 horizontal lines which are divided into two fields, odd and even. The field rate used is, therefore, 60 fields/sec. The spacing in time between any two consecutive fields is the same. The 2:1 interlace is used to give a vertical resolution of 525 lines/frame at the rate of 30 frames/sec, but with a flicker frequency of 60 cycles/sec to reduce the perception of flicker. The 2:1 interlace, however, causes some artifacts. By considering a solid circle moving in the horizontal and vertical directions, discuss what distortion in the circle may be visible. Assume that all the lines in a given field are displayed at the same time.

8

Image Enhancement

8.0 INTRODUCTION

Image enhancement is the processing of images to improve their appearance to human viewers or to enhance other image processing systems' performance. Methods and objectives vary with the application. When images are enhanced for human viewers, as in television, the objective may be to improve perceptual aspects: image quality, intelligibility, or visual appearance. In other applications, such as object identification by machine, an image may be preprocessed to aid machine performance. Because the objective of image enhancement is dependent on the application context, and the criteria for enhancement are often subjective or too complex to be easily converted to useful objective measures, image enhancement algorithms tend to be simple, qualitative, and ad hoc. In addition, in any given application, an image enhancement algorithm that performs well for one class of images may not perform as well for other classes.

Image enhancement is closely related to image restoration, which will be discussed in Chapter 9. When an image is degraded, restoration of the original image often results in enhancement. There are, however, some important differences between restoration and enhancement. In image restoration, an ideal image has been degraded, and the objective is to make the processed image resemble the original image as much as possible. In image enhancement, the objective is to make the processed image better in some sense than the unprocessed image. In this case, the ideal image depends on the problem context and often is not well defined. To illustrate this difference, note that an original, undegraded image cannot be further restored but can be enhanced by increasing sharpness through highpass filtering.

Image enhancement is desirable in a number of contexts. In one important class of problems, an image is enhanced by modifying its contrast and/or dynamic

range. For example, a typical image, even if undegraded, will often appear better when its edges are sharpened. Also, if an image with a large dynamic range is recorded on a medium with a small dynamic range, such as film or paper, the contrast and therefore the details of the image are reduced, particularly in the very bright and dark regions. Contrast in an image taken from an airplane is reduced when the scenery is covered by cloud or mist. Increasing the local contrast and reducing the overall dynamic range can significantly enhance the quality of such an image.

In another class of enhancement problems, a degraded image may be enhanced by reducing the degradation. Examples of image degradation are blurring, random background noise, speckle noise, and quantization noise. This area of image enhancement overlaps with image restoration. An algorithm that is simple and ad hoc, and does not attempt to exploit the characteristics of the signal and degradation, is generally considered an enhancement algorithm. An algorithm that is more mathematical and complex, and exploits the characteristics of the signal and degradation with an explicit error criterion that attempts to compare the processed image with the original undegraded image, is generally regarded as a restoration algorithm. This distinction is admittedly somewhat vague and arbitrary. Some arbitrary decisions have been necessary in dividing certain topics between this chapter and the next chapter, which deals with the image restoration problem.

It is well known that the contours or edges in an object contain very important information that may be used in image understanding applications. The first step in such an application may be to preprocess an image into an edge map that consists of only edges. Since more accurate detection of edges in an image can enhance the performance of an image understanding system that exploits such information, converting an image to its corresponding edge map may be viewed as an enhancement process.*

Another important class of image enhancement problems is the display of 2-D data that may or may not represent the intensities of an actual image. A low-resolution image of 128 × 128 pixels may be made more visually pleasant to a human observer by interpolating it to generate a larger image, say 256 × 256 pixels. In 2-D spectral estimation, the spectral estimates have traditionally been displayed as contour plots. Although such 2-D data are not images in the conventional sense, they can be presented as images. We can display them as black-and-white images, or we can enhance them with color so that their appearance may be improved and information conveyed more clearly. In other applications, such as infrared radar imaging, range information as well as image intensities may be available. By displaying the range information with color, relative distances of objects in an image can be highlighted. Even good-quality images may be enhanced by certain types of distortion. For example, when an object in an image is displayed with false color, the object may stand out more clearly to a human viewer.

*Edge detection is useful in a variety of image processing applications including image enhancement, restoration, coding, and understanding. We have chosen to discuss the topic in this chapter.

In this chapter, we study methods of solving the image enhancement problems discussed above. In Section 8.1, we discuss modification of the contrast and dynamic range. In Section 8.2, we discuss noise smoothing. In Section 8.3, the problem of detecting edges of an image is discussed. In Section 8.4, we discuss the problem of image interpolation and motion estimation which can be used for image interpolation. In Section 8.5, we discuss enhancement of images by means of pseudocolor and false color.

Throughout this chapter and Chapters 9 and 10, the performance of various algorithms is illustrated using examples. These examples are included only for illustrative purposes and should not be used for comparing the performance of different algorithms. The performance of an image processing algorithm depends on many factors, such as the objective of the processing and the type of image used. One or two examples do not adequately demonstrate the performance of an algorithm. Unless specified otherwise, all images used are quantized at 8 bits/pixel for monochrome images and at 24 bits/pixel (8 bits/pixel for each of the red, green, and blue components) for color images.

8.1 CONTRAST AND DYNAMIC RANGE MODIFICATION

8.1.1 Gray Scale Modification

Gray scale modification is a simple and effective way of modifying an image's dynamic range (the range of image intensities) or contrast. In this method, the gray scale or intensity level of an input image $f(n_1, n_2)$ is modified according to a specific transformation. The transformation $g = T[f]$ that relates an input intensity f to an output intensity g is often represented by a plot or a table. Consider a simple illustration of this method. Figure 8.1(a) shows an image of 4 × 4 pixels, with each pixel represented by three bits, so that there are eight levels; that is, $f = 0$ (darkest level), 1, 2, . . . , 7 (brightest level). The transformation that relates the input intensity to the output intensity is shown as a plot and as a table in Figure 8.1(b). For each pixel in the input image, the corresponding output intensity is obtained from the plot or the table in Figure 8.1(b). The result is shown in Figure 8.1(c). By properly choosing the specific transformation, contrast or dynamic range can be modified.

The specific transformation desired depends on the application. In some applications, physical considerations determine the transformation selected. For example, when a display system has nonlinear characteristics, the objective of the modification may be to compensate for the nonlinearities. In such a case, the most suitable transformation can be determined from the nonlinearity of the display system.

A good transformation in typical applications can be identified by computing the histogram of the input image and studying its characteristics. The *histogram* of an image, denoted by $p(f)$, represents the number of pixels that have a specific intensity f as a function of f. For example, the 4 × 4-pixel image shown in Figure 8.1(a) has the histogram shown in Figure 8.2(a). The histogram displays some

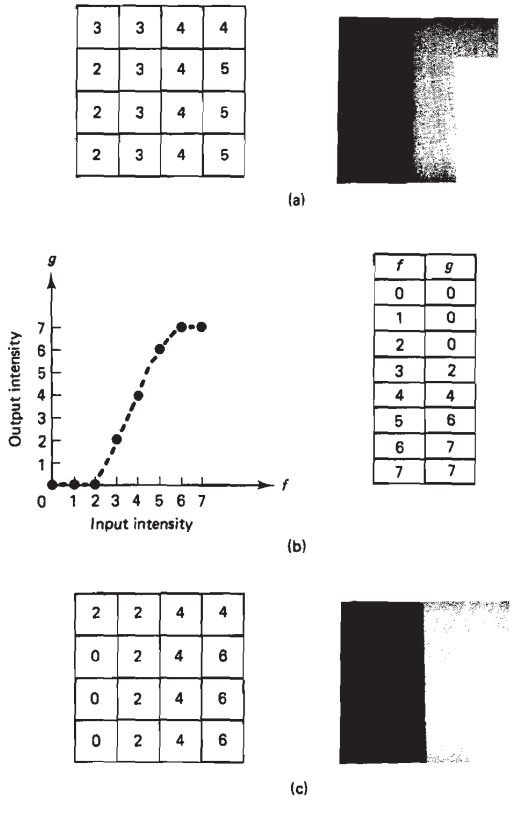


Figure 8.1 Example of gray scale modification. (a) Image of 4 × 4 pixels, with each pixel represented by 3 bits; (b) gray scale transformation function; (c) result of modifying the image in (a) using the gray scale transformation function in (b).

important image features that help determine which particular gray scale transformation is desirable. In Figure 8.2(a), the image's intensities are clustered in a small region, and the available dynamic range is not very well utilized. In such a case, a transformation of the type shown in Figure 8.1(b) would increase the overall dynamic range, and the resulting image would appear to have greater contrast. This is evidenced by Figure 8.2(b), which is the histogram of the processed image shown in Figure 8.1(c).

Because computing the histogram of an image and modifying its gray scale for a given gray scale transformation requires little computation, the desirable gray scale transformation can be determined by an experienced human operator in real

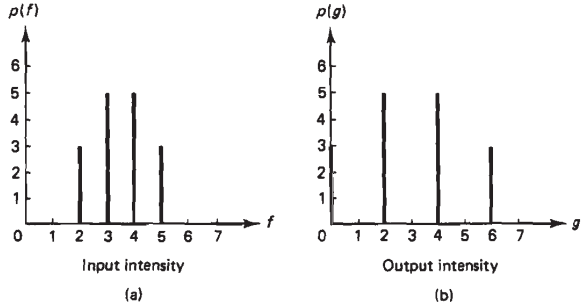


Figure 8.2 Histogram of 4 × 4-pixel image (a) in Figure 8.1(a); (b) in Figure 8.1(c).

time. On the basis of the initial histogram computation, the operator chooses a gray scale transformation to produce a processed image. By looking at the processed image and its histogram, the operator can choose another gray scale transformation, obtaining a new processed image. These steps can be repeated until the output image satisfies the operator.

In such circumstances as when there are too many images for individual attention by a human operator, the gray scale transformation must be chosen automatically. A method known as *histogram modification* is useful in this case. In this method, the gray scale transformation that produces a desired histogram is chosen for each individual image. The desired histogram of the output image, denoted by $p_d(g)$, that is useful for typical images has a maximum around the middle of the dynamic range and decreases slowly as the intensity increases or decreases. For a given image, we wish to determine the transformation function so that the resulting output image has a histogram similar to $p_d(g)$. This problem can be phrased in terms of a problem in elementary probability theory. Specifically, the histograms $p(f)$ and $p_d(g)$ can be viewed as scaled probability density functions of random variables f and g, respectively. For example, $p(3)/16$ in Figure 8.2(a) is the probability that a randomly chosen pixel in the 4 × 4-pixel image in Figure 8.1(a) will have an intensity level of 3. We wish to find a transformation $g = T[f]$ with the constraint that $T[f]$ must be a monotonically nondecreasing function of f such that $p(g)$ is equal to or close to $p_d(g)$. One approach to solving this probability problem is to obtain the probability distribution functions $P(f)$ and $P_d(g)$ by integrating the probability density functions $p(f)$ and $p_d(g)$ and then choosing the transformation function such that $P(f)$ will be equal to or close to $P_d(g)$ at $g = T[f]$. Imposing the constraint that $T[f]$ must be a monotonically nondecreasing function ensures that a pixel with a higher intensity than another pixel will not become a pixel with a lower intensity in the output image.

Applying this approach to the histogram modification problem which involves

discrete variables f and g, we first compute the cumulative histograms $P(f)$ and $P_d(g)$ from $p(f)$ and $p_d(g)$ by

$$P(f) = \sum_{k=0}^{f} p(k) = P(f-1) + p(f) \tag{8.1a}$$

$$P_d(g) = \sum_{k=0}^{g} p_d(k) = P_d(g-1) + p_d(g). \tag{8.1b}$$

An example of the cumulative histograms is shown in Figure 8.3. Figures 8.3(a) and (b) show an example of $p(f)$ and $p_d(g)$, and Figures 8.3(c) and (d) show $P(f)$ and $P_d(g)$ obtained by using (8.1). From $P(f)$ and $P_d(g)$, the gray scale transformation $g = T[f]$ can be obtained by choosing g for each f such that $P_d(g)$ will be closest to $P(f)$. The gray scale transformation function obtained from Figure 8.3 is shown in Figure 8.4(a), and the histogram of the image obtained by using this transformation function is shown in Figure 8.4(b). If the desired histogram $p_d(g)$ remains the same for different input images, $P_d(g)$ needs to be computed only once from $p_d(g)$.

In the example we considered above, note that the histogram of the processed image is not the same as the given desired histogram. This is in general the case when f and g are discrete variables and we require that all the pixels with the same input intensity be mapped to the same output intensity. Note also that the desired cumulative histogram $P_d(g)$ is close to a straight line. In the special case of the

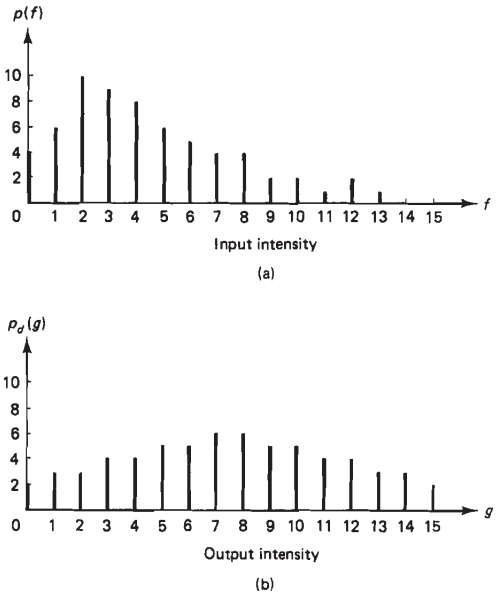
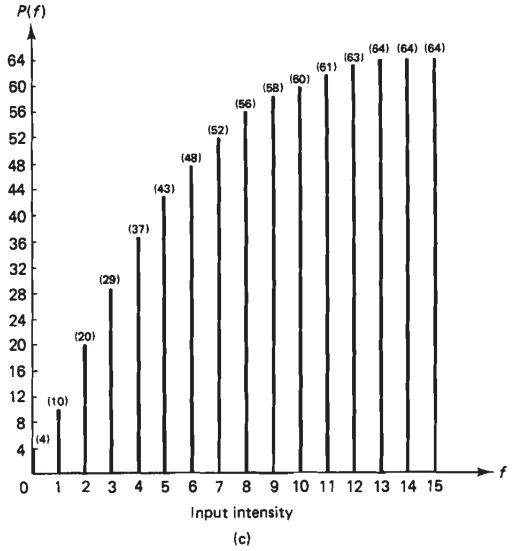


Figure 8.3 Histograms and cumulative histograms. (a) Histogram of an 8 × 8-pixel image; (b) desired histogram; (c) cumulative histogram derived from (a); (d) cumulative histogram derived from (b).



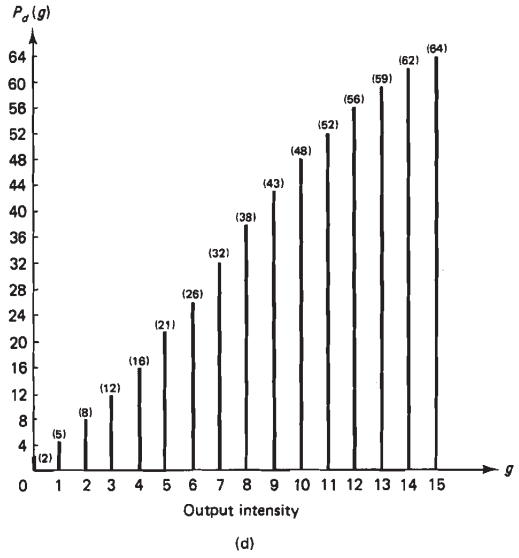


Figure 8.3 (continued)

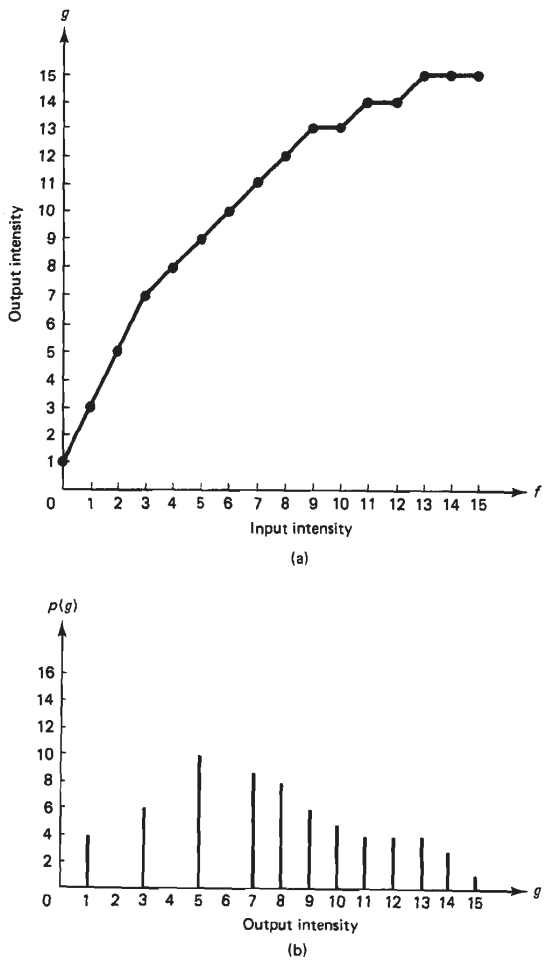


Figure 8.4 (a) Gray scale transformation function that approximately transforms the histogram in Figure 8.3(a) to the desired histogram in Figure 8.3(b); (b) histogram of gray-scale-transformed image obtained by applying the transformation function in (a) to an image with the histogram shown in Figure 8.3(a).

histogram modification known as *histogram equalization*, the desired histogram is assumed constant. In this case, the desired cumulative histogram would be exactly a straight line. Images processed by histogram equalization typically have more contrast than unprocessed images, but they tend to appear somewhat unnatural.

Even though gray scale modification is conceptually and computationally simple, it can often provide significant improvement in image quality or intelligibility to the human observer, and is therefore used routinely in many image processing applications. This is illustrated in the following two examples. Figure 8.5(a) shows an original image of 512 × 512 pixels, with each pixel represented by eight bits. Figure 8.5(b) shows the histogram of the image in Figure 8.5(a). The histogram clearly shows that a large number of the image's pixels are concentrated in the lower intensity levels of the dynamic range, suggesting that the image will appear very dark with a loss of contrast in the dark regions. By increasing the contrast in the dark regions, the details can be made more visible. This can be accomplished by using the transformation function shown in Figure 8.5(c). The processed image using the function in Figure 8.5(c) is shown in Figure 8.5(d), and the histogram of the processed image is shown in Figure 8.5(e). Another example is shown in Figure 8.6. The unprocessed image is shown in Figure 8.6(a) and the image processed by gray scale modification is shown in Figure 8.6(b).

The histogram modification method discussed above can also be applied to color images. To improve the image contrast with only a relatively small effect on the hue or saturation, we can transform RGB images $f_R(n_1, n_2)$, $f_G(n_1, n_2)$, and $f_B(n_1, n_2)$ to YIQ images $f_Y(n_1, n_2)$, $f_I(n_1, n_2)$, and $f_Q(n_1, n_2)$ by using the transformation in (7.8). Gray scale modification can be applied to only the Y image $f_Y(n_1, n_2)$, and the result can be combined with the unprocessed $f_I(n_1, n_2)$ and $f_Q(n_1, n_2)$. Again using the transformation in (7.8), the processed RGB images $g_R(n_1, n_2)$, $g_G(n_1, n_2)$, and $g_B(n_1, n_2)$ can be obtained. Figure 8.7(a) (see color insert) shows an original color image of 512 × 512 pixels, and Figure 8.7(b) shows the image processed by the gray scale transformation discussed above.

8.1.2 Highpass Filtering and Unsharp Masking

Highpass filtering emphasizes the high-frequency components of a signal while reducing the low-frequency components. Because edges or fine details of an image are the primary contributors to the high-frequency components of an image, highpass filtering often increases the local contrast and sharpens the image.

Unsharp masking, which has been known to photographic artists for a long time, is closely related to highpass filtering. In unsharp masking, the original image is blurred (unsharpened) and a fraction of the unsharp image is subtracted from, or masks, the original. The subtraction is performed by adding the negative of the unsharp image to the original. The image processed by unsharp masking can be expressed as

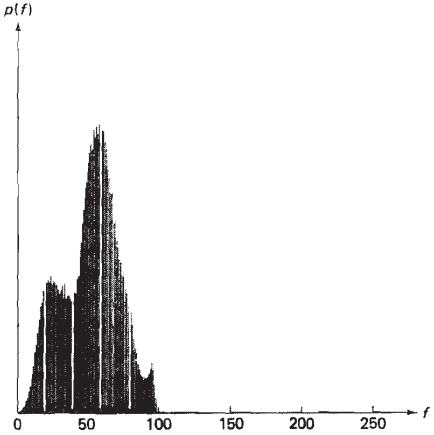
$$g(n_1, n_2) = af(n_1, n_2) - bf_L(n_1, n_2) \tag{8.2}$$

where $f(n_1, n_2)$ is the original image, $f_L(n_1, n_2)$ is the lowpass filtered or unsharp image, a and b are positive scalars with $a > b$, and $g(n_1, n_2)$ is the processed image. Rewriting $f(n_1, n_2)$ as a sum of the lowpass filtered image $f_L(n_1, n_2)$ and the highpass filtered image $f_H(n_1, n_2)$, we can write (8.2) as

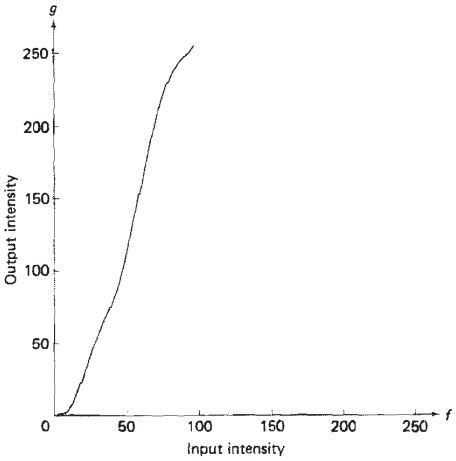
$$g(n_1, n_2) = (a - b) f_L(n_1, n_2) + af_H(n_1, n_2). \tag{8.3}$$

From (8.3), it is clear that high-frequency components are emphasized over low-

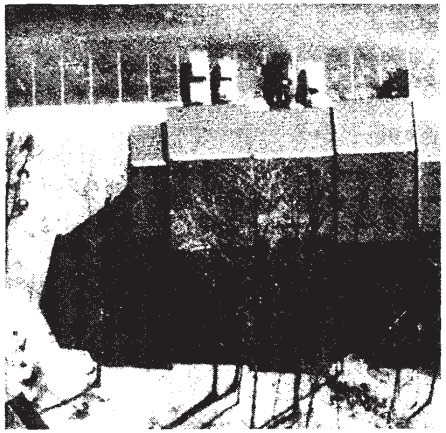
(a)



(b)

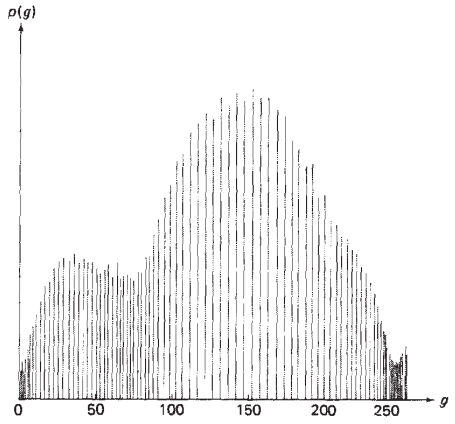


(c)



(d)

Figure 8.5 Example of gray scale modification. (a) Original image of 512 × 512 pixels; (b) histogram of the image in (a); (c) transformation function used in the gray scale modification; (d) processed image; (e) histogram of the processed image in (d).



(e)

Figure 8.5 (continued)

(a)

(b)

Figure 8.6 Example of gray scale modification. (a) Original image of 512 × 512 pixels; (b) processed image.

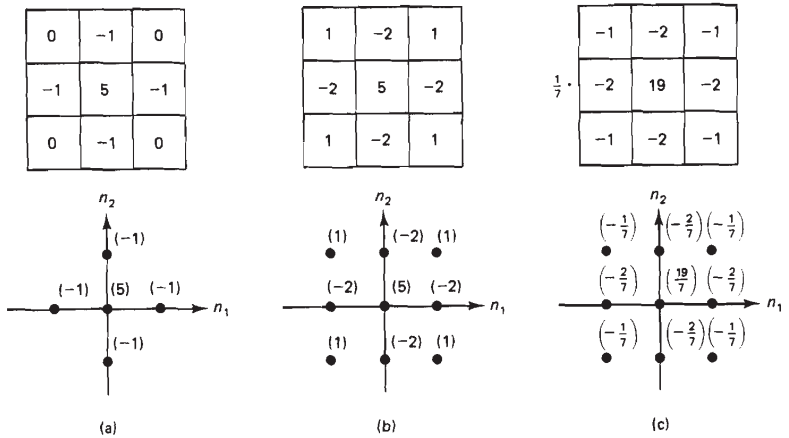


Figure 8.8 Impulse responses of highpass filters useful for image enhancement. Two representations are shown for each filter.

frequency components and that unsharp masking is some form of highpass filtering.*

Some typical examples of the impulse response of a highpass filter used for contrast enhancement are shown in Figure 8.8. One characteristic of all the filters in Figure 8.8 is that the sum of all amplitudes of each impulse response is one, so that the filter frequency response $H(\omega_1, \omega_2)$ is one at $\omega_1 = \omega_2 = 0$ and passes the DC component unaltered. This characteristic has the effect of preserving the average intensity of the original image in the processed image. It should be noted that this characteristic does not itself guarantee that the intensity of the processed image will remain in the range between 0 and 255. If intensities of some pixels in the processed image lie outside this range, they can be clipped to 0 and 255, or the image can be rescaled so that the intensities of all pixels in the processed image lie in the range between 0 and 255.

Figure 8.9 illustrates the performance of highpass filtering. Figure 8.9(a) shows an original image of 256 × 256 pixels, and Figure 8.9(b) shows the result of highpass filtering using the filter in Figure 8.8(a). Although the original image is not degraded, some highpass filtering increases the local contrast and thus gives it a sharper visual appearance. However, because a highpass filter emphasizes high-frequency components, and background noise typically has significant high-frequency components, highpass filtering tends to increase the background noise

*Unsharp masking is performed by a photographic artist in the domain of film density, which can be approximated by log intensity. Unsharp masking, therefore, is highpass filtering in the log intensity domain, which is related to homomorphic processing discussed in Section 8.1.3.

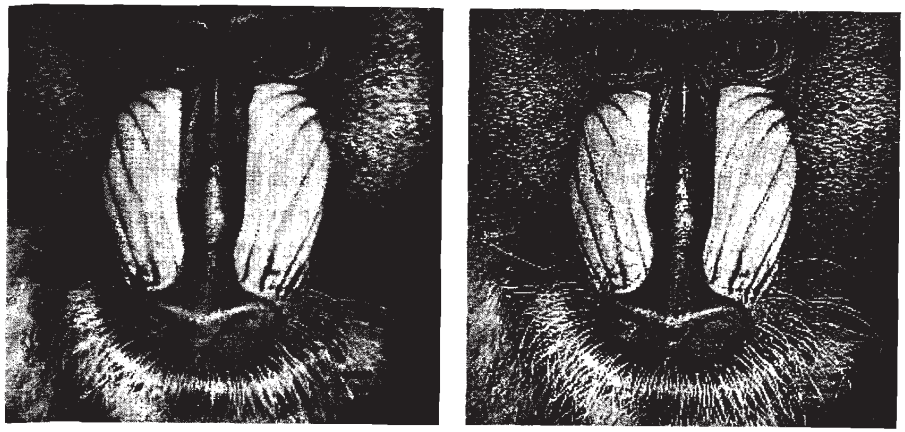


Figure 8.9 Example of highpass filtering. (a) Original image of 256 × 256 pixels; (b) highpass filtered image.

power. A comparison of the background regions of Figures 8.9(a) and 8.9(b) shows that the highpass filtered image appears more noisy than the unprocessed image. The accentuation of background noise is a limitation of any algorithm that attempts to increase the local contrast and sharpen the visual appearance of an image.

8.1.3 Homomorphic Processing

When an image with a large dynamic range, for instance, a natural scene on a sunny day, is recorded on a medium with a small dynamic range, such as film or paper, the image's contrast is significantly reduced, particularly in the dark and bright regions. One approach to enhancing the image is to reduce its dynamic range and increase its local contrast prior to recording it on a medium with a small dynamic range.

One method developed to reduce the dynamic range and increase the local contrast is based on applying a homomorphic system for multiplication to an image formation model. An image is typically formed by recording the light reflected from an object that has been illuminated by some light source. Based on this observation, one simple model of an image is

$$f(n_1, n_2) = i(n_1, n_2)r(n_1, n_2) \tag{8.4}$$

where $i(n_1, n_2)$ represents the illumination and $r(n_1, n_2)$ represents the reflectance. In developing a homomorphic system for image enhancement, the illumination component $i(n_1, n_2)$ is assumed to be the primary contributor to the dynamic range

of an image and is assumed to vary slowly, while the reflectance component $r(n_1, n_2)$ that represents the details of an object is assumed to be the primary contributor to local contrast and is assumed to vary rapidly. To reduce the dynamic range and increase the local contrast, then, we need to reduce $i(n_1, n_2)$ and increase $r(n_1, n_2)$. To separate $i(n_1, n_2)$ from $r(n_1, n_2)$ in (8.4), a logarithmic operation is applied to (8.4) and the result is

$$\log f(n_1, n_2) = \log i(n_1, n_2) + \log r(n_1, n_2). \tag{8.5}$$

If we assume that $\log i(n_1, n_2)$ remains slowly varying and $\log r(n_1, n_2)$ remains rapidly varying, lowpass filtering $\log f(n_1, n_2)$ will result in $\log i(n_1, n_2)$ and highpass filtering $\log f(n_1, n_2)$ will result in $\log r(n_1, n_2)$. Once $\log i(n_1, n_2)$ and $\log r(n_1, n_2)$ have been separated at least approximately, $\log i(n_1, n_2)$ is attenuated to reduce the dynamic range while $\log r(n_1, n_2)$ is emphasized to increase the local contrast. The processed $\log i(n_1, n_2)$ and $\log r(n_1, n_2)$ are then combined and the result is exponentiated to get back to the image intensity domain. This is shown in Figure 8.10(a). The system in Figure 8.10(a) can be simplified by replacing the system inside the dotted line in the figure with the corresponding highpass filter. The simplified system is shown in Figure 8.10(b). An example illustrating the performance of the system in Figure 8.10(b) is shown in Figure 8.11. Figure 8.11(a)

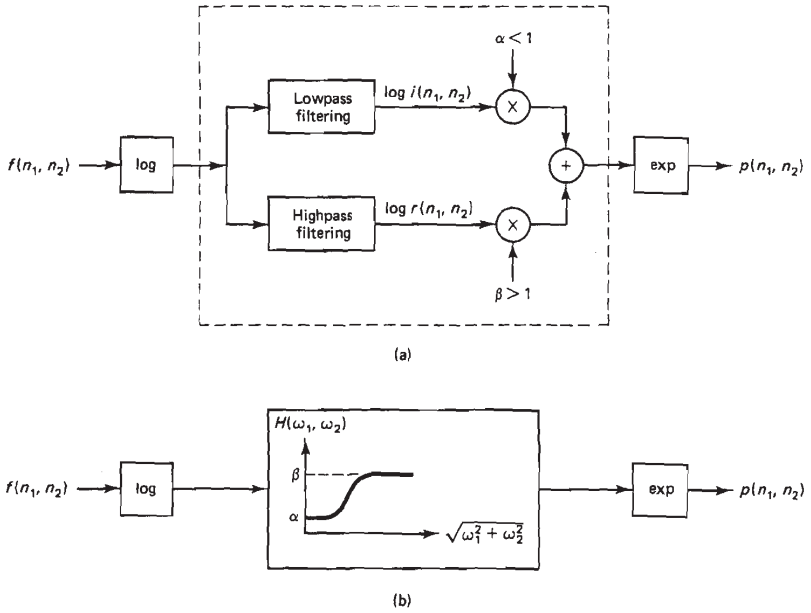


Figure 8.10 Homomorphic system for image enhancement. (a) Homomorphic system for contrast enhancement and dynamic range modification; (b) simplification of system in (a).

shows an original image of 256 × 256 pixels and Figure 8.11(b) shows the processed image using the system in Figure 8.10(b).

A system like that in Figure 8.10, which performs a logarithmic operation followed by a linear operation followed by an exponentiation operation, is called a *homomorphic system for multiplication.* This is the origin of the terms *homomorphic processing* and *homomorphic filtering.* The logarithmic operation transforms multiplicative components to additive components. The linear operation performs the separation by exploiting characteristics of the resulting additive components and processes the separated components. The exponentiation operation brings the processed signal back to the original signal domain.

Although the system in Figure 8.10 was developed from a model of image formation and a homomorphic system, the system can be viewed simply as highpass filtering in the log intensity domain. Performing highpass filtering in the log intensity domain is also reasonable in light of human visual system. As discussed in Section 7.2.2, the image intensity appears to be modified at the peripheral level of a human visual system by some form of nonlinearity such as a logarithmic operation. Thus, the log intensity domain is, in a sense, more central to the human visual system than is the intensity domain.

8.1.4 Adaptive Modification of Local Contrast and Local Luminance Mean

In some applications, it is desirable to modify the local contrast and local luminance mean as the local characteristics of an image vary. In such applications, it is reasonable to process an image adaptively.

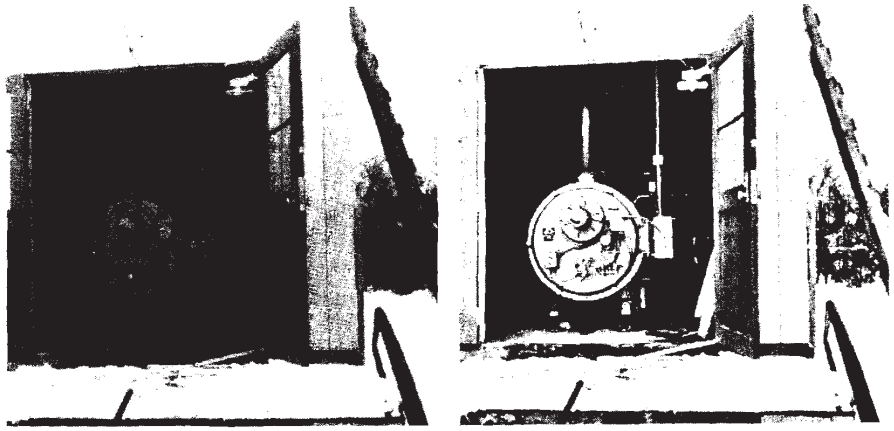


Figure 8.11 Example of homomorphic processing for image enhancement. (a) Original image of 256 × 256 pixels; (b) processed image by homomorphic system for multiplication. After [Oppenheim et al.].

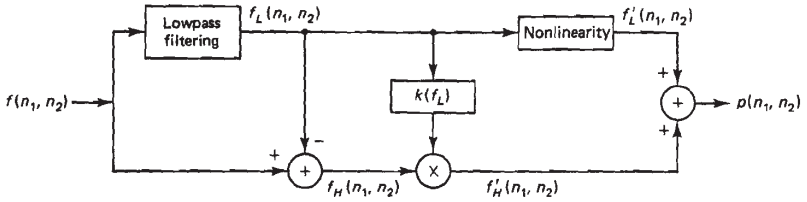


Figure 8.12 System for the modification of local contrast and local luminance mean as a function of luminance mean.

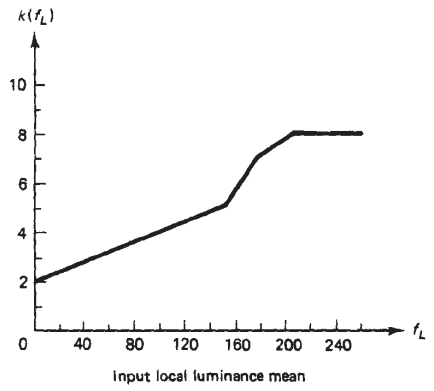
One application in which adaptive modification of the local contrast and local luminance mean is used is enhancement of an image taken from an airplane through varying amounts of cloud cover. According to one simple model of image degradation due to cloud cover, regions of an image covered by cloud have increased local luminance mean due to direct reflection of the sunlight by cloud cover and decreased local contrast due to attenuation of the signal from the ground when it passes through the cloud cover. One approach to enhancing the image, then, is to increase the local contrast and decrease the local luminance mean whenever cloud cover is detected. One way to detect the presence of cloud cover is by measuring the local luminance mean. When the local luminance mean is high, it is likely that cloud cover is present.

One system developed to reduce the effect of the cloud cover is shown in Figure 8.12. This system modifies the local contrast and the local luminance mean. In the figure, $f(n_1, n_2)$ denotes the unprocessed image. The sequence $f_L(n_1, n_2)$ which denotes the local luminance mean of $f(n_1, n_2)$ is obtained by lowpass filtering $f(n_1, n_2)$. The sequence $f_H(n_1, n_2)$, which denotes the local contrast, is obtained by subtracting $f_L(n_1, n_2)$ from $f(n_1, n_2)$. The local contrast is modified by multiplying $f_H(n_1, n_2)$ with $k(f_L)$, a scalar that is a function of $f_L(n_1, n_2)$. The modified contrast is denoted by $f'_H(n_1, n_2)$. If $k(f_L)$ is greater than one, the local contrast is increased, while $k(f_L)$ less than one represents local contrast decrease. The local luminance mean is modified by a point nonlinearity, and the modified local luminance mean is denoted by $f'_L(n_1, n_2)$. The modified local contrast and local luminance mean are then combined to obtain the processed image, $p(n_1, n_2)$. To increase the local contrast and decrease the local luminance mean when the local luminance mean is high, we choose a larger $k(f_L)$ for a larger f_L, and we choose the nonlinearity, taking into account the local luminance mean change and the contrast increase. Figure 8.13 shows the result of applying the system in Figure 8.12 to enhance an image taken from an airplane through varying amounts of cloud cover. Figure 8.13(a) shows the original image of 256 × 256 pixels. Figure 8.13(b) shows the processed image. The function $k(f_L)$ and the nonlinearity used are shown in Figures 8.13(c) and 8.13(d). The lowpass filtering operation was performed by using an FIR filter whose impulse response is an 8 × 8-point rectangular window.

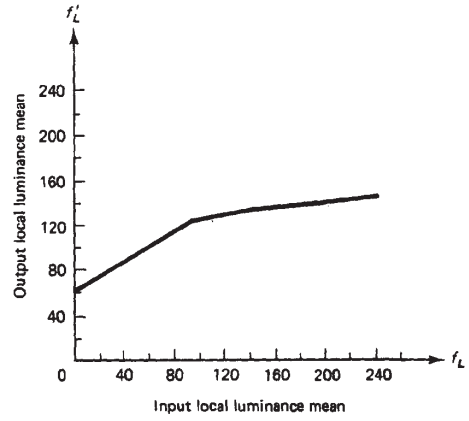
The system in Figure 8.12 can be viewed as a special case of a *two-channel process*. In the two-channel process, the image to be processed is divided into

(a)

(b)



(c)



(d)

Figure 8.13 Example of image enhancement by adaptive filtering. (a) Image of 256 × 256 pixels taken from an airplane through varying amounts of cloud cover; (b) result of processing the image in (a) with the system in Figure 8.12; (c) function $k(f_L)$ used in the processing; (d) nonlinearity used in the processing. After [Peli and Lim].

two components, the local luminance mean and the local contrast. The two components are modified separately and then the results are combined. In the system in Figure 8.12, the local luminance mean is modified by a nonlinearity, and the local contrast is modified by the multiplication factor $k(f_L)$. As Chapters 9 and 10 show, a two-channel process is also useful in image restoration and coding.

The notion of adapting an image enhancement system to changing local characteristics is generally a very useful idea that can be applied in a number of different contexts. For example, gray scale transformation and highpass filtering, discussed earlier, can be modified so that they adapt to some varying local characteristics. Even though an adaptive system often requires considerably more computations than a nonadaptive system, the adaptive system's performance is generally considerably better. It is worthwhile to explore adaptive systems in solving an image enhancement problem that requires high performance. Adaptive processing of images is also very useful in image restoration and coding, and is discussed further in Chapters 9 and 10.

8.2 NOISE SMOOTHING

In addition to enhancement of images by contrast and dynamic range modification, images can also be enhanced by reducing degradations that may be present. This area of image enhancement overlaps with image restoration. In this section, we discuss very simple algorithms that attempt to reduce random noise and salt-and-pepper type of noise. Algorithms that are more mathematical and complex and algorithms that address other types of degradation will be discussed in Chapter 9.

8.2.1 Lowpass Filtering

The energy of a typical image is primarily concentrated in its low-frequency components. This is due to the high spatial correlation among neighboring pixels. The energy of such forms of image degradation as wideband random noise is typically more spread out over the frequency domain. By reducing the high-frequency components while preserving the low-frequency components, lowpass filtering reduces a large amount of noise at the expense of reducing a small amount of signal.

Lowpass filtering can also be used together with highpass filtering in processing an image prior to its degradation by noise. In applications such as image coding, an original undegraded image is available for processing prior to its degradation by noise, for instance, quantization noise. In such applications, the undegraded image can be highpass filtered prior to its degradation and then lowpass filtered after degradation. This may result in some improvement in the quality or intelligibility of the resulting image. For example, when the degradation is due to wideband random noise, the effective SNR (signal-to-noise ratio) of the degraded image is much lower in the high-frequency components than in the low-frequency components, due to the lowpass character of a typical image. Highpass filtering prior to the degradation significantly improves the SNR in the high-frequency components at the expense of small SNR decrease in the low-frequency components.

Examples of impulse responses of lowpass filters typically used for image enhancement are shown in Figure 8.14. To illustrate the performance of lowpass filtering for image enhancement, two examples are considered. Figure 8.15(a) shows an original noise-free image of 256 × 256 pixels, and Figure 8.15(b) shows an image degraded by wideband Gaussian random noise at an SNR of 15 dB. The SNR is defined as 10 $\log_{10}$ (image variance/noise variance). Figure 8.15(c) shows the result of lowpass filtering the degraded image. The lowpass filter used is shown in Figure 8.14(c). In Figure 8.15, lowpass filtering clearly reduces the additive noise, but at the same time it blurs the image. Blurring is a primary limitation of lowpass filtering. Figure 8.16(a) shows an original image of 256 × 256 pixels with 8 bits/pixel. Figure 8.16(b) shows the image coded by a PCM system with Roberts's pseudonoise technique at 2 bits/pixel. Roberts's pseudonoise technique is discussed in Chapter 10. Figure 8.16(c) shows the result of highpass filtering before coding and lowpass filtering after coding. The highpass and lowpass filters used in this example are those in Figure 8.8(c) and Figure 8.14(c), respectively.

8.2.2 Median Filtering

Median filtering is a nonlinear process useful in reducing impulsive, or salt-and-pepper noise. It is also useful in preserving edges in an image while reducing random noise. Impulsive or salt-and-pepper noise can occur due to a random bit error in a communication channel. In a median filter, a window slides along the image, and the median intensity value of the pixels within the window becomes the output intensity of the pixel being processed. For example, suppose the pixel values within a window are 5, 6, 55, 10, and 15, and the pixel being processed has

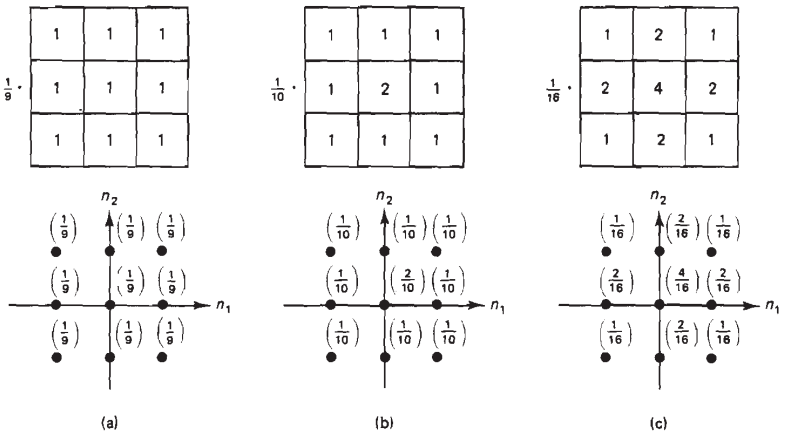
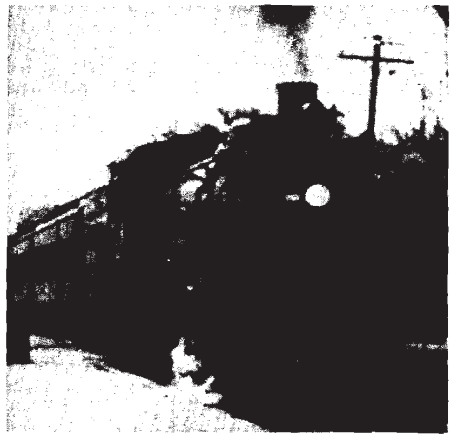


Figure 8.14 Impulse responses of lowpass filters useful for image enhancement.

(a)

(b)



(c)

Figure 8.15 Example of noise reduction by lowpass filtering. (a) Original image of 256 × 256 pixels; (b) original image degraded by wideband Gaussian random noise at SNR of 15 dB; (c) result of processing the image in (b) with a lowpass filter.

a value of 55. The output of the median filter at the current pixel location is 10, which is the median of the five values.

Like lowpass filtering, median filtering smoothes the image and is thus useful in reducing noise. Unlike lowpass filtering, median filtering can preserve discontinuities in a step function and can smooth a few pixels whose values differ signif-

(a)

(b)

(c)

Figure 8.16 Application of lowpass filtering in image coding. (a) Original image of 256 × 256 pixels; (b) image in (a) coded by a PCM system with Roberts's pseudonoise technique at 2 bits/pixel; (c) result of highpass filtering the image in (a) before coding and lowpass filtering after coding with a PCM system with Roberts's pseudonoise technique at 2 bits/pixel.

icantly from their surroundings without affecting the other pixels. Figure 8.17(a) shows a 1-D step sequence degraded by a small amount of random noise. Figure 8.17(b) shows the result after filtering with a lowpass filter whose impulse response is a 5-point rectangular window. Figure 8.17(c) shows the result after filtering with a 5-point median filter. It is clear from the figure that the step discontinuity

is better preserved by the median filter. Figure 8.18(a) shows a 1-D sequence with two values that are significantly different from the surrounding points. Figures 8.18(b) and (c) show the result of a lowpass filter and a median filter, respectively. The filters used in Figure 8.18 are the same as those used in Figure 8.17. If the two impulsive values are due to noise, the result of using a median filter will be to reduce the noise. If the two values are part of the signal, however, using the median filter will distort the signal.

An important parameter in using a median filter is the size of the window. Figure 8.19 illustrates the result of median filtering the signal in Figure 8.18(a) as a function of the window size. If the window size is less than 5, the two pixels with impulsive values will not be significantly affected. For a larger window, they will be. Thus, the choice of the window size depends on the context. Because it is difficult to choose the optimum window size in advance, it may be useful to try several median filters of different window sizes and choose the best of the resulting images.

In the above, we discussed 1-D median filtering. The task involved in performing a median filtering operation extends straightforwardly from the 1-D case to the 2-D case. However, not all properties of a 1-D median filter apply to a

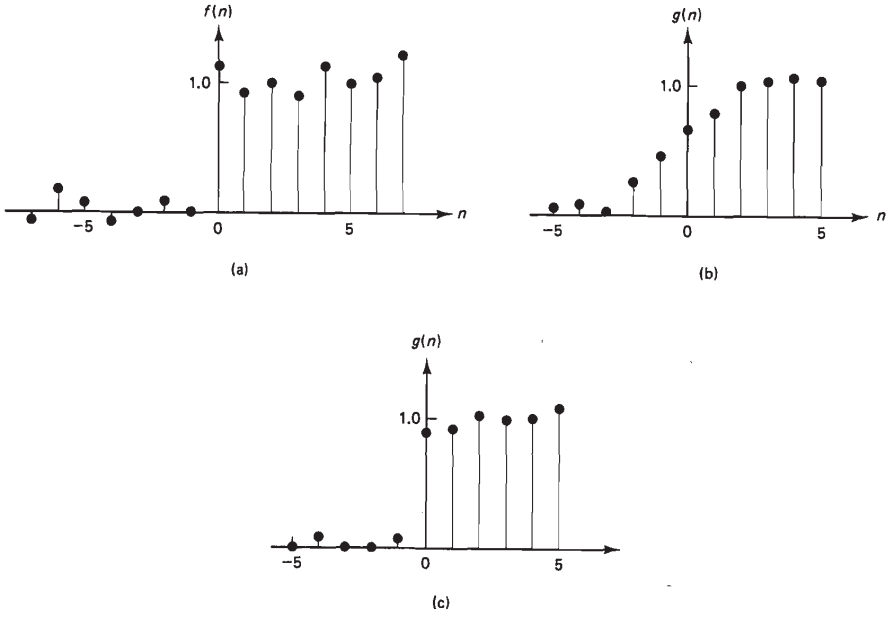


Figure 8.17 Illustration of median filter's tendency to preserve step discontinuities. (a) One-dimensional step sequence degraded by random noise; (b) result of lowpass filtering the sequence in (a) with a 5-point rectangular impulse response; (c) result of applying a 5-point median filter to the sequence in (a).

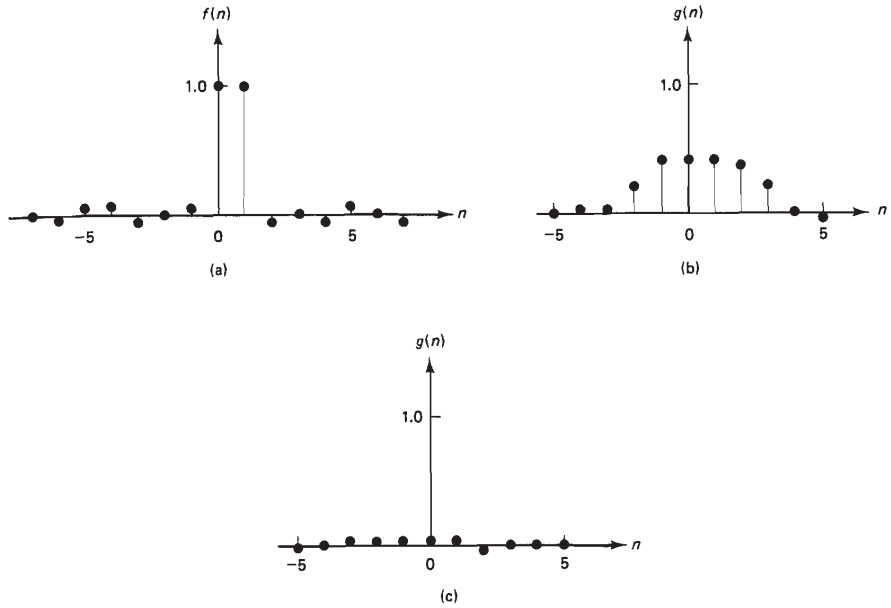


Figure 8.18 Illustration of a median filter's capability to remove impulsive values. (a) One-dimensional sequence with two consecutive samples significantly different from surrounding samples; (b) result of lowpass filtering the sequence in (a) with a 5-point rectangular impulse response; (c) result of applying a 5-point median filter to the sequence in (a).

2-D median filter. For example, median filtering a 1-D unit step sequence $u(n)$ preserves the step discontinuity and does not affect the signal $u(n)$ at all. Suppose we filter a 2-D step sequence $u(n_1, n_2)$ with a 2-D $N \times N$-point median filter. Figure 8.20(a) shows $u(n_1, n_2)$ and Figure 8.20(b) shows the result of filtering $u(n_1, n_2)$ with a 2-D 5×5-point median filter. From Figure 8.20(b), the intensity discontinuities which can be viewed as 1-D steps (for large n_1 at $n_2 = 0$ and large n_2 at $n_1 = 0$) are not affected. However, the discontinuities which are truly 2-D steps ($n_1 = n_2 = 0$) are seriously distorted. One method that tends to preserve 2-D step discontinuities well is to filter a 2-D signal along the horizontal direction with a 1-D median filter and then filter the result along the vertical direction with another 1-D median filter. This method is called *separable median filtering*, and is often used in 2-D median filtering applications. When a separable median filter is applied to $u(n_1, n_2)$, the signal $u(n_1, n_2)$ is not affected.

A median filter is a nonlinear system, and therefore many theoretical results on linear systems are not applicable. For example, the result of separable median filtering depends on the order in which the 1-D horizontal and vertical median filters are applied. Despite this difficulty, some theoretical results have been developed [Gallagher and Wise; Nodes and Gallagher; Arce and McLoughlin] on

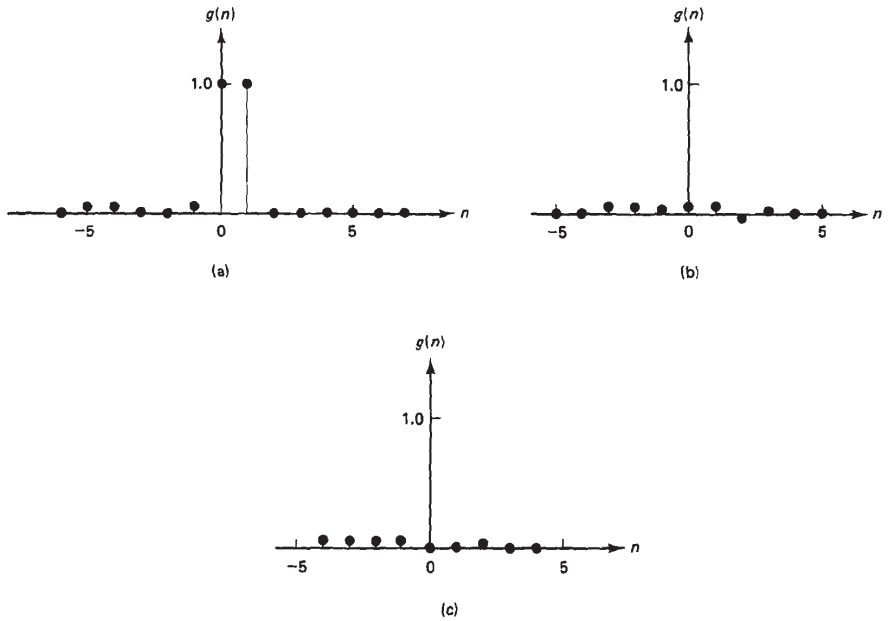


Figure 8.19 Results of applying a median filter to the sequence in Figure 8.18(a) as a function of window size. This illustrates that removal of impulsive values by a median filter depends on the window size. (a) Window size = 3; (b) window size = 5; (c) window size = 7.

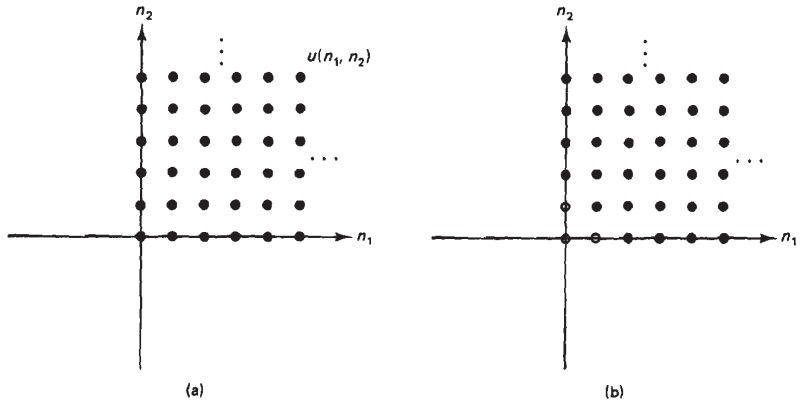


Figure 8.20 Illustration that a 2-D $N \times N$-point median filter distorts 2-D step discontinuities. (a) Unit step sequence $u(n_1, n_2)$; (b) result of filtering $u(n_1, n_2)$ with a 5×5-point median filter.

median filtering. One result states that repeated application of a 1-D median filter to a 1-D sequence eventually leads to a signal called a *root signal*, which is invariant under further applications of the 1-D median filter.

Two examples are given to illustrate the performance of a median filter. In the first, the original image of 512×512 pixels shown in Figure 8.21(a), is degraded

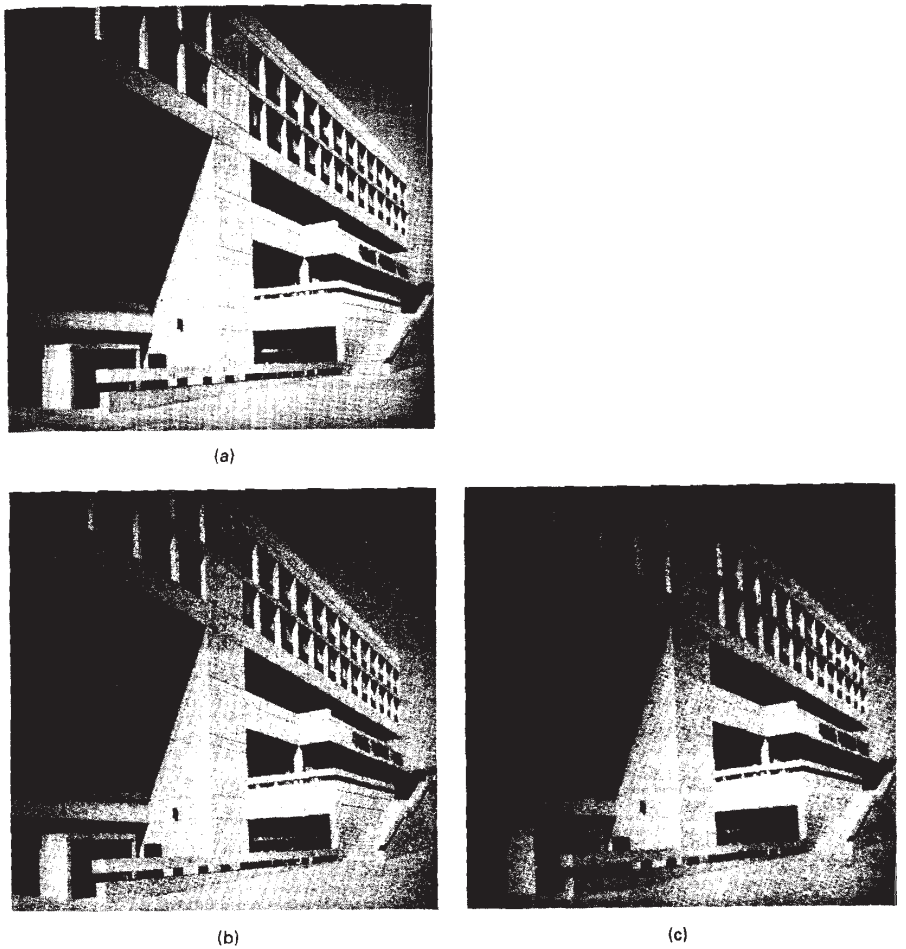


Figure 8.21 Example of wideband random noise reduction by median filtering. (a) Original image of 512×512 pixels; (b) image degraded by wideband Gaussian random noise at SNR of 7 dB; (c) processed image by a separable median filter with window size of 3 for both the horizontal and vertical 1-D median filter.

by wideband Gaussian random noise at an SNR of 7 dB. The degraded image is shown in Figure 8.21(b). Figure 8.21(c) shows the image processed by a separable median filter with a window size of 3 for both the horizontal and vertical 1-D median filters. Although the very sharp edges are not blurred, median filtering blurs the image significantly. In the second example, the original image from Figure 8.21(a) is degraded by salt-and-pepper noise. The degraded image is shown in Figure 8.22(a) and the image processed by the same separable median filter used in Figure 8.21 is shown in Figure 8.22(b). This example shows that median filtering is quite effective in removing salt-and-pepper noise.

8.2.3 Out-Range Pixel Smoothing

Like median filtering, *out-range pixel smoothing* is a nonlinear operation and is useful in reducing salt-and-pepper noise. In this method, a window slides along the image, and the average of the pixel values, excluding the pixel being processed, is obtained. If the difference between the average and the value of the pixel processed is above some threshold, then the current pixel value is replaced by the average. Otherwise, the value is not affected. Because it is difficult to determine the best parameter values in advance, it may be useful to process an image using several different threshold values and window sizes and select the best result.

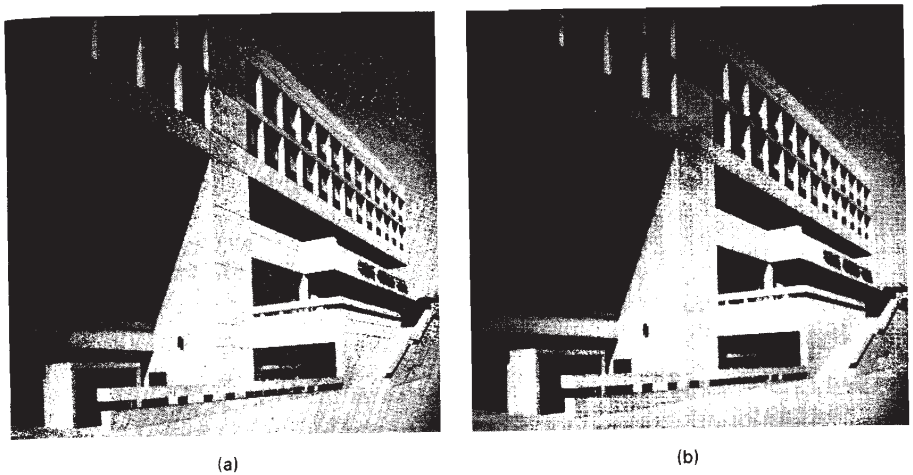
Figure 8.23 illustrates the performance of out-range pixel smoothing. The image in Figure 8.23 is the result after processing the image in Figure 8.22(a) using out-range pixel smoothing with a threshold value of 50 and a 3 × 3-point window.

8.3 EDGE DETECTION

An *edge* in an image is a boundary or contour at which a significant change occurs in some physical aspect of an image, such as the surface reflectance, illumination, or the distances of the visible surfaces from the viewer. Changes in physical aspects manifest themselves in a variety of ways, including changes in intensity, color, and texture. In our discussion, we are concerned only with the changes in image intensity.

Detecting edges is very useful in a number of contexts. For example, in a typical image understanding task such as object identification, an essential step is to segment an image into different regions corresponding to different objects in the scene. Edge detection is often the first step in image segmentation. As another example, one approach to the development of a low bit-rate image coding system is to code only the detected edges. It is well known that an image that consists of only edges is highly intelligible.

The significance of a physical change in an image depends on the application; an intensity change that would be classified as an edge in some applications might not be considered an edge in other applications. In an object identification system, an object's boundaries may be sufficient for identification, and contours that represent additional details within the object may not be considered edges. An edge cannot be defined, then, outside of the context of an application. Nevertheless, edge detection algorithms that detect edges that are useful in a broad set of applications have been developed. In this section, we discuss some of the more representative edge detection algorithms.



(a) (b)

Figure 8.22 Example of salt-and-pepper noise reduction by median filtering. (a) Image in Figure 8.21(a) degraded by salt-and-pepper noise; (b) processed image by the same separable median filter used in Figure 8.21.

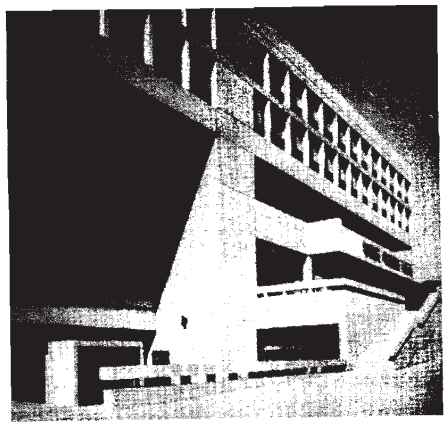


Figure 8.23 Example of salt-and-pepper noise reduction by outrange pixel smoothing. Image in Figure 8.22(a) processed by outrange pixel smoothing with threshold value of 50 and window size of 3 × 3.

8.3.1 Gradient-Based Methods

Consider an analog* function $f(x)$ which represents a typical 1-D edge, as shown in Figure 8.24(a). In typical problems, it is reasonable to consider the value x_0 in the figure an edge point. One way to determine x_0 is to compute the first derivative $f'(x)$ or the second derivative $f''(x)$. Figures 8.24(b) and (c) show $f'(x)$ and $f''(x)$. From the figure, the value x_0 can be determined by looking for the local extremum (maximum or minimum) of $f'(x)$ or by looking for a zero crossing of $f''(x)$ where $f''(x)$ changes its sign. In this section, we discuss methods that exploit the characteristics of $f'(x)$. In the next section, we discuss methods that exploit the characteristics of $f''(x)$.

In addition to determining the possible edge point x_0, $f'(x)$ can also be used in estimating the strength and direction of the edge. If $|f'(x)|$ is very large, $f(x)$ is changing very rapidly and a rapid change in intensity is indicated. If $f'(x)$ is positive, $f(x)$ is increasing. Based on the above observations, one approach to detecting edges is to use the system shown in Figure 8.25. In the system, first $|f'(x)|$ is computed from $f(x)$. If $|f'(x)|$ is greater than some threshold, it is a candidate to be an edge. If all values of x such that $|f'(x)|$ is greater than a certain threshold are detected to be edges, an edge will appear as a line rather than a point. To avoid this problem, we further require $|f'(x)|$ to have a local maximum at the edge points. It may be desirable to determine whether $f(x)$ is increasing or decreasing at $x = x_0$. The necessary information is contained in $f'(x)$ at $x = x_0$. The choice of the threshold depends on the application. As the threshold increases, only the values of x where $f(x)$ changes rapidly will be registered as candidate edges. Since it is difficult to choose the threshold optimally, some trial and error is usually involved. It is also possible to choose the threshold adaptively. The system in Figure 8.25 is based on the particular type of edge shown in Figure 8.24(a), but it is generally applicable to detecting various other types of edges.

The generalization of $f'(x)$ to a 2-D function $f(x, y)$ is the gradient $\nabla f(x, y)$ given by

$$\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x} \hat{i}_x + \frac{\partial f(x, y)}{\partial y} \hat{i}_y \tag{8.6}$$

where $\hat{i}_x$ is the unit vector in the x-direction and $\hat{i}_y$ is the unit vector in the y-direction. A generalization of the edge detection system in Figure 8.25 based on $\nabla f(x, y)$ is shown in Figure 8.26. The magnitude of $\nabla f(x, y)$ is first computed and is then compared with a threshold to determine candidate edge points. If all values of (x, y) such that $|\nabla f(x, y)|$ is greater than a certain threshold are detected to be edges, the edges will appear as strips rather than lines. The process of determining an edge line from a strip of candidate edge points is called *edge thinning*. In one simple edge thinning algorithm, the edge points are selected by

*Sometimes, it is more convenient to develop results in the analog domain. In such instances, we will begin the development of results in the analog domain and then discretize the results at some later point in the development.

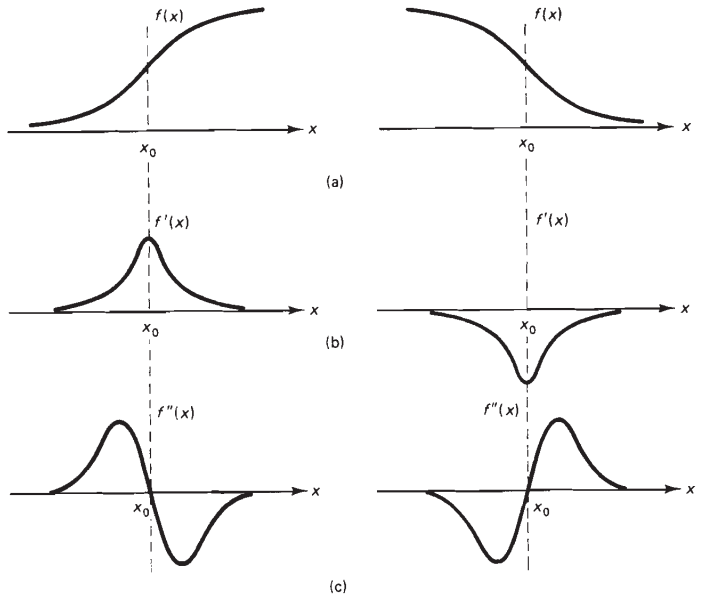


Figure 8.24 (a) $f(x)$, (b) $f'(x)$, and (c) $f''(x)$ for a typical 1-D edge.

checking if $|\nabla f(x, y)|$ is a local maximum in at least one direction. The property that $|\nabla f(x, y)|$ achieves its local maximum in at least one direction is usually checked along a few specified directions. In most cases, it is sufficient to check for local maxima in only the horizontal and vertical directions. If $|\nabla f(x, y)|$ is a local maximum along any one of the specified directions at a potential edge point, the potential edge point is considered to be an edge point. One difficulty with this simple edge thinning algorithm is that it creates a number of minor false edge lines in the vicinity of strong edge lines. One simple method to remove most of these minor false edge lines is to impose the following additional constraints:

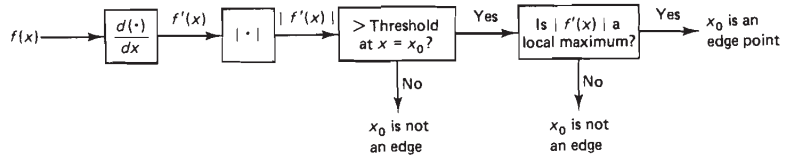


Figure 8.25 System for 1-D edge detection.

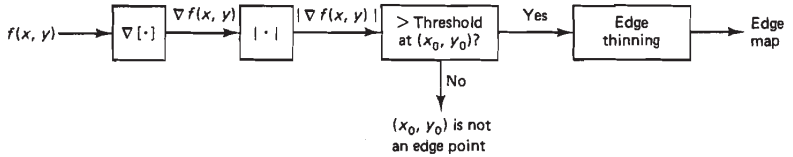


Figure 8.26 System for 2-D edge detection.

(a) If $|\nabla f(x, y)|$ has a local maximum at (x_0, y_0) in the horizontal direction but not in the vertical direction, (x_0, y_0) is an edge point when $\left|\frac{\partial f(x, y)}{\partial x}\right|_{x=x_0, y=y_0} > k \left|\frac{\partial f(x, y)}{\partial y}\right|_{x=x_0, y=y_0}$ with k typically chosen around 2.

(b) If $|\nabla f(x, y)|$ has a local maximum at (x_0, y_0) in the vertical direction but not in the horizontal direction, (x_0, y_0) is an edge point when $\left|\frac{\partial f(x, y)}{\partial y}\right|_{x=x_0, y=y_0} > k \left|\frac{\partial f(x, y)}{\partial x}\right|_{x=x_0, y=y_0}$ with k typically chosen around 2.

When $|\nabla f(x, y)|$ has a local maximum at (x_0, y_0) in the horizontal direction but not in the vertical direction, Condition (a) requires that the rate of intensity change along the horizontal direction is significantly larger than that along the vertical direction. Condition (b) is the same as Condition (a) with the roles of x and y reversed. Why these additional constraints remove most of the minor false edges in the vicinity of major edges is discussed in Problem 8.17.

An edge detection system that is based on a function such as $|\nabla f(x, y)|$ is called a *nondirectional edge detector*, since such functions do not have a bias toward any particular direction. If an edge detection system is based on a function that has a bias toward one particular direction, it is called a *directional edge detector*. If we use $|\partial f(x, y)/\partial x|$ instead of $|\nabla f(x, y)|$ in the system in Figure 8.26, for example, the system will detect edges in the vertical direction, but will not respond to edges in the horizontal direction.

For a 2-D sequence $f(n_1, n_2)$, the partial derivatives $\partial f(x, y)/\partial x$ and $\partial f(x, y)/\partial y$ can be replaced by some form of difference. For example, $\partial f(x, y)/\partial x$ may be replaced by

$$\frac{\partial f(x, y)}{\partial x} \leftrightarrow [f(n_1, n_2) - f(n_1 - 1, n_2)]/T, \tag{8.7a}$$

$$[f(n_1 + 1, n_2) - f(n_1, n_2)]/T, \tag{8.7b}$$

or

$$[f(n_1 + 1, n_2) - f(n_1 - 1, n_2)]/(2T). \tag{8.7c}$$

Since the computed derivatives are compared later with a threshold and the threshold can be adjusted, the scaling factors $1/T$ and $1/2T$ can be omitted. Typically the expressions in (8.7) are averaged over several samples to improve the reliability

and continuity of the estimate of $\partial f(x, y)/\partial x$. Examples of "improved" estimates of $\partial f(x, y)/\partial x$ are

$$\frac{\partial f(x, y)}{\partial x} \leftrightarrow [f(n_1 + 1, n_2 + 1) - f(n_1 - 1, n_2 + 1)] + [f(n_1 + 1, n_2) - f(n_1 - 1, n_2)] + [f(n_1 + 1, n_2 - 1) - f(n_1 - 1, n_2 - 1)] \tag{8.8a}$$

or

$$[f(n_1 + 1, n_2 + 1) - f(n_1 - 1, n_2 + 1)] + 2[f(n_1 + 1, n_2) - f(n_1 - 1, n_2)] + [f(n_1 + 1, n_2 - 1) - f(n_1 - 1, n_2 - 1)]. \tag{8.8b}$$

The unnecessary scaling factors have been omitted in (8.8).

The differencing operation in (8.7) and (8.8) can be viewed as the convolution of $f(n_1, n_2)$ with the impulse response of a filter $h(n_1, n_2)$. Examples of impulse responses that can be used in developing directional edge detectors are shown in Figure 8.27. The filters $h(n_1, n_2)$ in Figures 8.27(a) and (b) detect edges in the vertical and horizontal directions and can be viewed as approximating $\partial f(x, y)/\partial x$ and $\partial f(x, y)/\partial y$ respectively. The filters $h(n_1, n_2)$ in Figures 8.27(c) and (d) detect edges along the two diagonal directions. The gradient $\nabla f(x, y)$ in (8.6) can also be expressed in terms of the first-order partial derivatives in a rotated coordinate system. When the rotation is 45 degrees, the directions of partial derivatives are along the two diagonal directions.

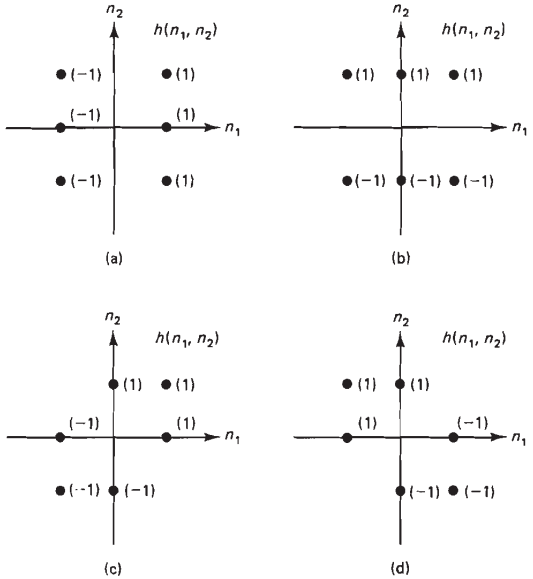


Figure 8.27 Impulse responses of filters that can be used for directional edge detection. (a) Vertical edge detection; (b) horizontal edge detection; (c) and (d) diagonal edge detection.

Nondirectional edge detectors can be developed by discrete approximation of $|\nabla f(x, y)|$ in the system in Figure 8.26. From (8.6),

$$|\nabla f(x, y)| = \sqrt{\left(\frac{\partial f(x, y)}{\partial x}\right)^2 + \left(\frac{\partial f(x, y)}{\partial y}\right)^2} \tag{8.9}$$

From (8.9), nondirectional edge detectors can be developed by nonlinear combination of the terms used in the development of directional edge detectors. An example of discrete approximation of (8.9) that can be used for nondirectional edge detectors is given by

$$|\nabla f(x, y)| \rightarrow \sqrt{(f_x(n_1, n_2))^2 + (f_y(n_1, n_2))^2} \tag{8.10}$$

where

$$f_x(n_1, n_2) = f(n_1, n_2) * h_x(n_1, n_2)$$

$$f_y(n_1, n_2) = f(n_1, n_2) * h_y(n_1, n_2)$$

and $h_x(n_1, n_2)$ and $h_y(n_1, n_2)$ are shown in Figure 8.28. The method developed by Sobel [Duda and Hart] is based on (8.10) with $h_x(n_1, n_2)$ and $h_y(n_1, n_2)$ in Figure 8.28. Another example is the method developed by [Roberts], which is based on (8.10) with $h_x(n_1, n_2)$ and $h_y(n_1, n_2)$ shown in Figure 8.29. Depending on exactly how $|\nabla f(x, y)|$ is approximated in the discrete domain, many other variations can be developed.

Figure 8.30 shows the result of edge detection using a directional edge detector. Figure 8.30(a) shows an image of 512 × 512 pixels. Figure 8.30(b) and (c) show the results of a vertical edge detector and a horizontal edge detector, respectively, applied to the image in Figure 8.30(a). The vertical and horizontal edge detectors are based on $h(n_1, n_2)$ in Figures 8.27(a) and (b). Figures 8.31(a) and (b) show the results of applying the Sobel edge detector and Roberts's edge detector to the image in Figure 8.30(a). Both belong to the class of nondirectional edge detectors, and the specific method of determining the threshold value and checking the local maximum property of an edge used is the same as that used in Figure 8.30.

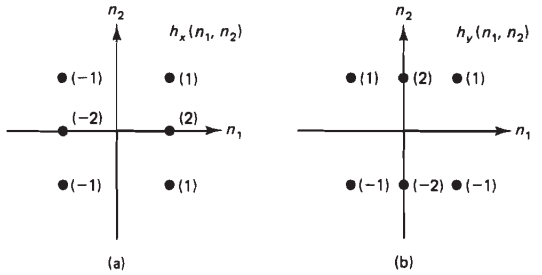


Figure 8.28 Approximation of (a) $\partial f(x, y)/\partial x$ with $f(n_1, n_2) * h_x(n_1, n_2)$; (b) $\partial f(x, y)/\partial y$ with $f(n_1, n_2) * h_y(n_1, n_2)$. Sobel's edge detection method is based on comparison of $\sqrt{(f(n_1, n_2) * h_x(n_1, n_2))^2 + (f(n_1, n_2) * h_y(n_1, n_2))^2}$ with a threshold.

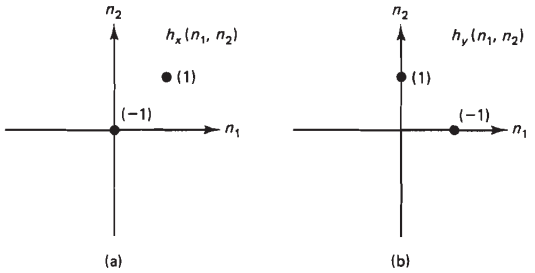


Figure 8.29 Impulse responses of filters used in Roberts's edge detection method. The method is based on comparison of

$$\sqrt{(f(n_1, n_2) * h_x(n_1, n_2))^2 + (f(n_1, n_2) * h_y(n_1, n_2))^2}$$

with a threshold.

There are many variations of the edge detection methods discussed in this section. For example, we could use a different nonlinear combination of $\partial f(x, y)/\partial x$ and $\partial f(x, y)/\partial y$ instead of

$$\sqrt{(\partial f(x, y)/\partial x)^2 + (\partial f(x, y)/\partial y)^2}$$

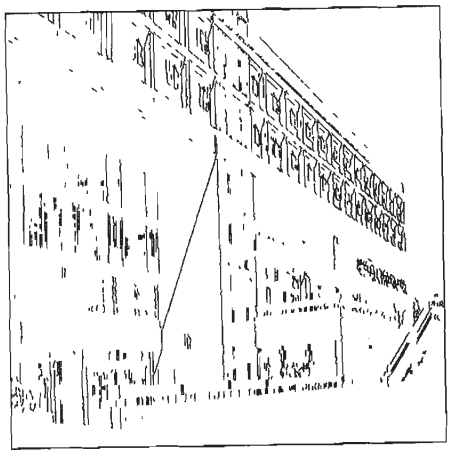
in the system in Figure 8.26. Many different methods can also be developed for edge thinning.

The edge detection methods discussed in this section can be improved in various ways. Methods based on computing some form of gradient or differencing are typically sensitive to noise. A certain number of isolated edge points which appear randomly distributed throughout the edge maps in Figure 8.31 are most likely the result of some background noise or very fine image details. Some noise smoothing using the methods discussed in Section 8.2 or more sophisticated noise reduction methods that we will discuss in Chapter 9 may be desirable prior to applying an edge detection algorithm. Isolated random edge points may also be removed by some simple processing of the edge maps. Gradient-based edge detection methods also cause some discontinuities in the detected edge contours, as can be seen from the edge maps in Figure 8.31. Methods that impose continuity constraints in the detected edge contours can also be developed [Roberts].

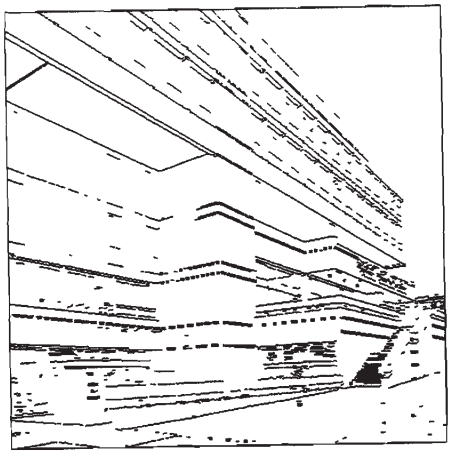
8.3.2 Laplacian-Based Methods

The objective of an edge detection algorithm is to locate the regions where the intensity is changing rapidly. In the case of a 1-D function $f(x)$, searching for regions of rapidly changing intensity corresponds to searching for regions where $f'(x)$ is large. For gradient-based methods, $f'(x)$ is considered large when its magnitude $|f'(x)|$ is greater than a threshold. Another possible way is to assume that $f'(x)$ is large whenever it reaches a local extremum, that is, whenever the second derivative $f''(x)$ has a zero crossing. This is illustrated in Figure 8.24. Declaring zero-crossing points as edges results in a large number of points being declared to be edge points. Since there is no check on the magnitude of $f'(x)$,

(a)

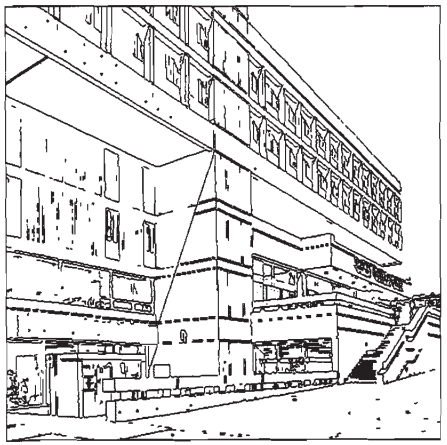


(b)

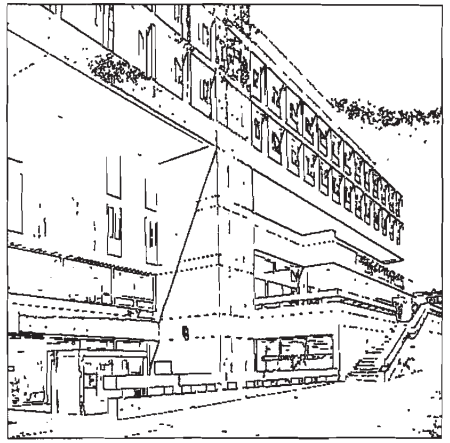


(c)

Figure 8.30 Edge maps obtained by directional edge detectors. (a) Image of 512 × 512 pixels; (b) result of applying a vertical edge detector; (c) result of applying a horizontal edge detector.



(a)



(b)

Figure 8.31 Result of applying (a) Sobel edge detector and (b) Roberts's edge detector to the image in Figure 8.30(a).

any small ripple in $f(x)$ is enough to generate an edge point. Due to this sensitivity to noise, the application of a noise reduction system prior to edge detection is very desirable in processing images with background noise.

A generalization of $\partial^2 f(x)/\partial x^2$ to a 2-D function $f(x, y)$ for the purpose of edge detection (see Problem 8.19) is the Laplacian $\nabla^2 f(x, y)$ given by

$$\nabla^2 f(x, y) = \nabla(\nabla f(x, y)) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}. \tag{8.11}$$

For a 2-D sequence $f(n_1, n_2)$, the partial second derivatives $\partial^2 f(x, y)/\partial x^2$ and $\partial^2 f(x, y)/\partial y^2$ can be replaced by some form of second-order differences. Second-order differences can be represented by convolution of $f(n_1, n_2)$ with the impulse response of a filter $h(n_1, n_2)$. Examples of $h(n_1, n_2)$ that may be used are shown in Figure 8.32. To illustrate that $f(n_1, n_2) * h(n_1, n_2)$ may be viewed as a discrete approximation of $\nabla^2 f(x, y)$, let us consider $h(n_1, n_2)$ in Figure 8.32(a). Suppose we approximate $\partial f(x, y)/\partial x$ by

$$\frac{\partial f(x, y)}{\partial x} \rightarrow f_x(n_1, n_2) = f(n_1 + 1, n_2) - f(n_1, n_2). \tag{8.12}$$

We again omitted the scaling factor, since it does not affect zero-crossing points. Since the forward difference is used in (8.12), we can use the backward difference in approximating $\partial^2 f(x, y)/\partial x^2$:

$$\frac{\partial^2 f(x)}{\partial x^2} \rightarrow f_{xx}(n_1, n_2) = f_x(n_1, n_2) - f_x(n_1 - 1, n_2). \tag{8.13}$$

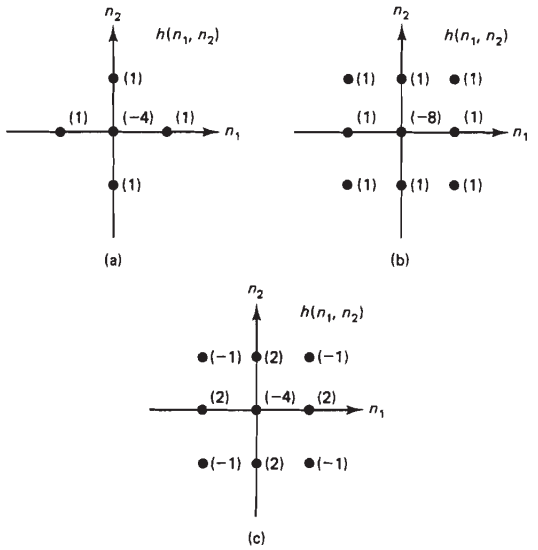


Figure 8.32 Examples of $h(n_1, n_2)$ that may be used in approximating $\nabla^2 f(x, y)$ with $f(n_1, n_2) * h(n_1, n_2)$.

From (8.12) and (8.13),

$$\frac{\partial^2 f(x, y)}{\partial x^2} \rightarrow f_{xx}(n_1, n_2) = f(n_1 + 1, n_2) - 2f(n_1, n_2) + f(n_1 - 1, n_2). \tag{8.14}$$

From (8.11) and (8.14), and approximating $\partial^2 f(x, y)/\partial y^2$ in a similar manner, we obtain

$$\begin{aligned}\nabla^2 f(x, y) \rightarrow \nabla^2 f(n_1, n_2) &= f_{xx}(n_1, n_2) + f_{yy}(n_1, n_2) \\ &= f(n_1 + 1, n_2) + f(n_1 - 1, n_2) + f(n_1, n_2 + 1) \\ &\quad + f(n_1, n_2 - 1) - 4f(n_1, n_2).\end{aligned} \tag{8.15}$$

The resulting $\nabla^2 f(n_1, n_2)$ is $f(n_1, n_2) * h(n_1, n_2)$ with $h(n_1, n_2)$ in Figure 8.32(a). Depending on how the second-order derivatives are approximated, it is possible to derive many other impulse responses $h(n_1, n_2)$, including those shown in Figures 8.32(b) and (c).

Figure 8.33 shows an example where edges were detected by looking for zero-crossing points of $\nabla^2 f(n_1, n_2)$. Figure 8.33(a) shows an image of 512 × 512 pixels. Figure 8.33(b) shows the zero-crossing points of $\nabla^2 f(n_1, n_2)$, obtained from (8.15) and using the image in Figure 8.33(a) as $f(n_1, n_2)$. Since zero-crossing contours are boundaries between regions, they tend to be continuous lines. As a result, edge thinning necessary in gradient-based methods is not needed in Laplacian-based methods. In addition, algorithms that force continuity in edge contours are not as useful in Laplacian-based methods as in gradient-based methods. As is

clear from Figure 8.33(b), however, choosing all zero-crossing points as edges tends to generate a large number of edge points.

The Laplacian-based methods discussed above generate many "false" edge contours, which typically appear in regions where the local variance of the image is small. As a special case, consider a uniform background region so that $f(n_1, n_2)$ is constant. Since $\nabla^2 f(n_1, n_2)$ is zero and we detect edges from zero-crossing points of $\nabla^2 f(n_1, n_2)$, any small perturbation of $f(n_1, n_2)$ is likely to cause false edge contours. One method to remove many of these false contours is to require that the local variance is sufficiently large at an edge point, as shown in Figure 8.34. The local variance $\sigma_f^2(n_1, n_2)$ can be estimated by

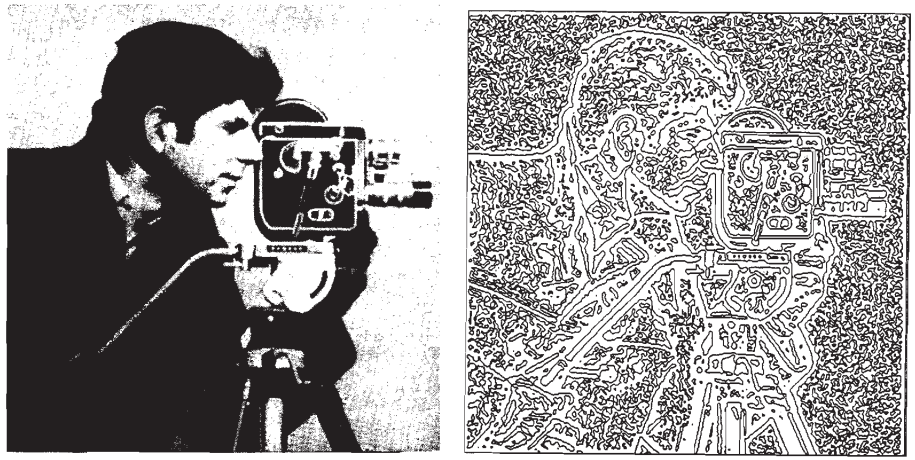
$$\sigma_f^2(n_1, n_2) = \frac{1}{(2M + 1)^2} \sum_{k_1 = n_1 - M}^{n_1 + M} \sum_{k_2 = n_2 - M}^{n_2 + M} [f(k_1, k_2) - m_f(k_1, k_2)]^2 \tag{8.16a}$$

where

$$m_f(n_1, n_2) = \frac{1}{(2M + 1)^2} \sum_{k_1 = n_1 - M}^{n_1 + M} \sum_{k_2 = n_2 - M}^{n_2 + M} f(k_1, k_2) \tag{8.16b}$$

with M typically chosen around 2. Since $\sigma_f^2(n_1, n_2)$ is compared with a threshold, the scaling factor $1/(2M + 1)^2$ in (8.16a) can be eliminated. In addition, the local variance σ_f^2 needs to be computed only for (n_1, n_2) which are zero-crossing points of $\nabla^2 f(n_1, n_2)$. Figure 8.35 shows the result of applying the system in Figure 8.34 to the image in Figure 8.33(a). Comparison of Figures 8.33(b) and 8.35 shows considerable reduction in the "false" edge contours.

The system in Figure 8.34 can be interpreted as a gradient-based method.



(a) (b)

Figure 8.33 Edge map obtained by a Laplacian-based edge detector. (a) Image of 512 × 512 pixels; (b) result of convolving the image in (a) with $h(n_1, n_2)$ in Figure 8.32(a) and then finding zero-crossing points.

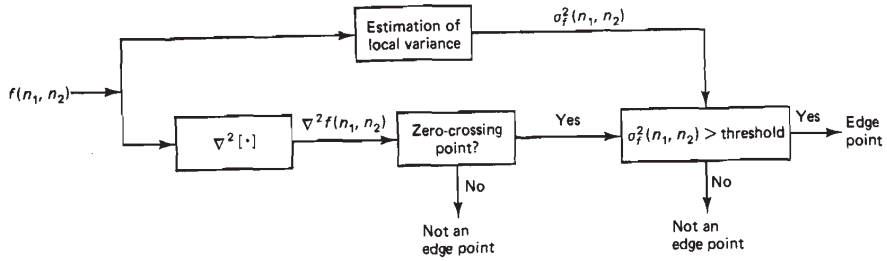


Figure 8.34 Laplacian-based edge detection system that does not produce many false edge contours.

The local variance $\sigma_f^2(n_1, n_2)$ is closely related to the gradient magnitude. Comparing $\sigma_f^2(n_1, n_2)$ with a threshold is similar to comparing the gradient magnitude with a threshold. Requiring that $\nabla^2 f(n_1, n_2)$ crosses zero at an edge can be interpreted as edge thinning. With this interpretation, we can implement the system in Figure 8.34 by computing $\sigma_f^2(n_1, n_2)$ first and then by detecting the zero-crossing points of $\nabla^2 f(n_1, n_2)$ only at those points where $\sigma_f^2(n_1, n_2)$ is above the chosen threshold.

8.3.3 Edge Detection by Marr and Hildreth's Method

In the previous two sections, we discussed edge detection algorithms that produce one edge map from an input image. Marr and Hildreth [Marr and Hildreth; Marr] observed that significant intensity changes occur at different scales (resolution) in an image. For example, blurry shadow regions and sharply focused fine-detail

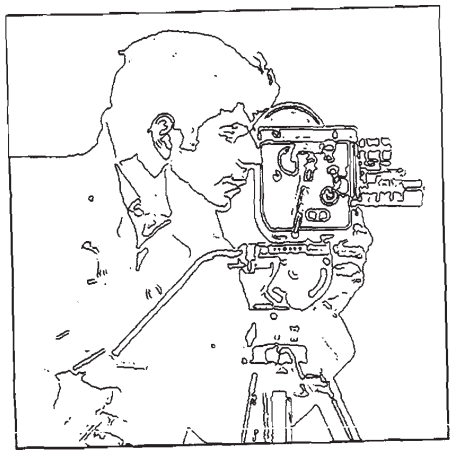


Figure 8.35 Edge map obtained by applying the system in Figure 8.34 to the image in Figure 8.33(a).

regions may be present in the same image. "Optimal" detection of significant intensity changes, therefore, generally requires the use of operators that respond to several different scales. Marr and Hildreth suggested that the original image be band-limited at several different cutoff frequencies and that an edge detection algorithm be applied to each of the images. The resulting edge maps have edges corresponding to different scales.

Marr and Hildreth argue that edge maps of different scales contain important information about physically significant parameters. The visual world is made of elements such as contours, scratches, and shadows, which are highly localized at their own scale. This localization is also reflected in such physically important changes as reflectance change and illumination change. If the same edge is present in a set of edge maps of different scale, it represents the presence of an image intensity change due to a single physical phenomenon. If an edge is present in only one edge map, one reason may be that two independent physical phenomena are operating to produce intensity changes in the same region of the image.

To bandlimit an image at different cutoff frequencies, the impulse response $h(x, y)$ and frequency response $H(\Omega_x, \Omega_y)$ of the lowpass filter proposed [Marr and Hildreth; Canny] is Gaussian-shaped and is given by

$$h(x, y) = e^{-(x^2+y^2)/(2\pi\sigma^2)} \tag{8.17a}$$

$$H(\Omega_x, \Omega_y) = 2\pi^2\sigma^2 e^{-\pi\sigma^2(\Omega_x^2+\Omega_y^2)/2} \tag{8.17b}$$

where σ determines the cutoff frequency with larger σ corresponding to lower cutoff frequency. The choice of Gaussian shape is motivated by the fact that it is smooth and localized in both the spatial and frequency domains. A smooth $h(x, y)$ is less likely to introduce any changes that are not present in the original shape. A more localized $h(x, y)$ is less likely to shift the location of edges.

From the smoothed images, edges can be detected by using the edge detection algorithms discussed in the previous two sections. Depending on which method is used, the lowpass filtering operation in (8.17) and the partial derivative operation used for edge detection may be combined. For example, noting that $\nabla^2[\cdot]$ and convolution $*$ are linear, we obtain

$$\begin{aligned} \nabla^2(f(x, y) * h(x, y)) &= f(x, y) * [\nabla^2 h(x, y)] \\ &= f(x, y) * \left[\frac{\partial^2 h(x, y)}{\partial x^2} + \frac{\partial^2 h(x, y)}{\partial y^2}\right]. \end{aligned} \tag{8.18}$$

For the Gaussian function $h(x, y)$ in (8.17), $\nabla^2 h(x, y)$ and its Fourier transform are given by

$$\nabla^2 h(x, y) = \frac{e^{-(x^2+y^2)/(2\pi\sigma^2)}}{(\pi\sigma^2)^2}(x^2 + y^2 - 2\pi\sigma^2) \tag{8.19a}$$

$$F[\nabla^2 h(x, y)] = -2\pi^2\sigma^2 e^{-\pi\sigma^2(\Omega_x^2+\Omega_y^2)/2}(\Omega_x^2 + \Omega_y^2). \tag{8.19b}$$

Marr and Hildreth chose, for simplicity, to detect edges by looking for zero-crossing points of $\nabla^2 f(x, y)$. Bandlimiting $f(x, y)$ tends to reduce noise, thus reducing the noise sensitivity problem associated with detecting zero-crossing points. The func-

tions $\nabla^2 h(x, y)$ and $-F[\nabla^2 h(x, y)]$ in (8.19) are sketched in Figure 8.36. Clearly, computing $f(x, y) * \nabla^2 h(x, y)$ is equivalent to bandpass filtering $f(x, y)$ where σ^2 in (8.19) is a parameter that controls the bandwidth of the bandpass filter. For a sequence $f(n_1, n_2)$, one approach is to simply replace x and y in (8.19) with n_1 and n_2.

Figure 8.37 shows an example of the approach under discussion. Figures 8.37(a), (b), and (c) show three images obtained by blurring the original image in Figure 8.33(a) with $h(n_1, n_2)$ obtained by replacing x and y of $h(x, y)$ in (8.17) with n_1 and n_2 with $\sigma^2 = 4$, 16, and 36, respectively. Figures 8.37(d), (e), and (f) show the images obtained by detecting zero crossings of $f(n_1, n_2) * \nabla^2 h(x, y)|_{x=n_1, y=n_2}$ with $\nabla^2 h(x, y)$ given by (8.19a) for $\sigma^2 = 4$, 16, and 36, respectively. Marr and Hildreth used the edge maps of different scales, such as those in Figures 8.37(d), (e), and (f) for object representation in their image understanding work.

8.3.4 Edge Detection Based on Signal Modeling

The edge detection algorithms discussed above are general methods, in that they are developed independent of an application context. An alternative approach is to develop an edge detection algorithm specific to a particular application problem. If we know the shape of an edge, for example, this information can be incorporated in the development of an edge detection algorithm. To illustrate how an edge detection algorithm specific to an application problem may be developed, we consider the problem of detecting boundaries of coronary arteries from an angiogram [Abrams].

The coronary arteries are the blood vessels which encircle the heart and supply blood to the heart muscle. Narrowing of the coronary arteries prevents adequate blood supply from reaching the heart, causing pain and damage to the heart muscle. Such damage is called coronary disease. To determine the severity of coronary disease, a coronary angiogram is used. An angiogram is an X ray picture of arteries taken after a contrast agent, typically iodine, has been injected into the vessels. The contrast agent is injected directly into the arteries through a catheter in order to achieve high concentrations. An example of a coronary angiogram is shown in Figure 8.38. Different observers making conventional visual evaluations of an angiogram will give widely varying evaluations of the severity of the disease.

The most commonly used measure of an obstruction is percentage of stenosis, which is defined as the maximum percentage of arterial narrowing within a specified length of the vessel. One approach to estimating the percentage of stenosis begins with determining the vessel boundaries from an angiogram. We will be concerned with the problem of detecting the vessel boundaries.

One reasonable model of an angiogram $f(n_1, n_2)$ is given by

$$f(n_1, n_2) = (v(n_1, n_2) + p(n_1, n_2)) * g(n_1, n_2) + w(n_1, n_2) \qquad (8.20)$$

where $v(n_1, n_2)$ denotes the vessel, $p(n_1, n_2)$ denotes the background, $g(n_1, n_2)$ denotes blurring, and $w(n_1, n_2)$ denotes the background noise. The vessel function $v(n_1, n_2)$ is derived from a generalized cone model of a 3-D vessel which is con-

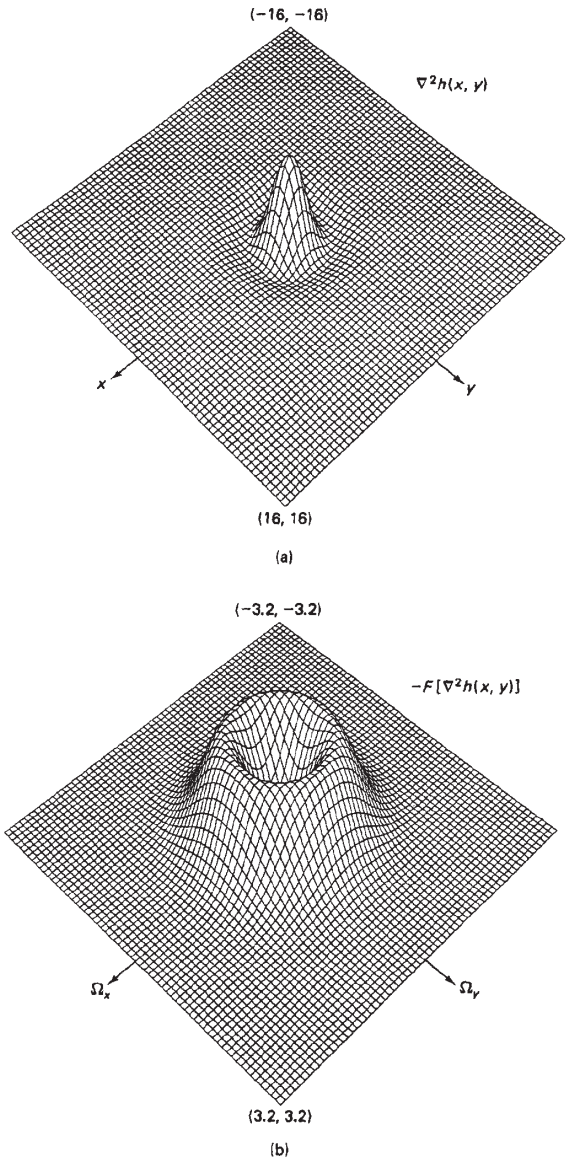


Figure 8.36 Sketch of (a) $\nabla^2 h(x, y)$ and (b) $-F[\nabla^2 h(x, y)]$ in Equation (8.19) for $\sigma^2 = 1$.

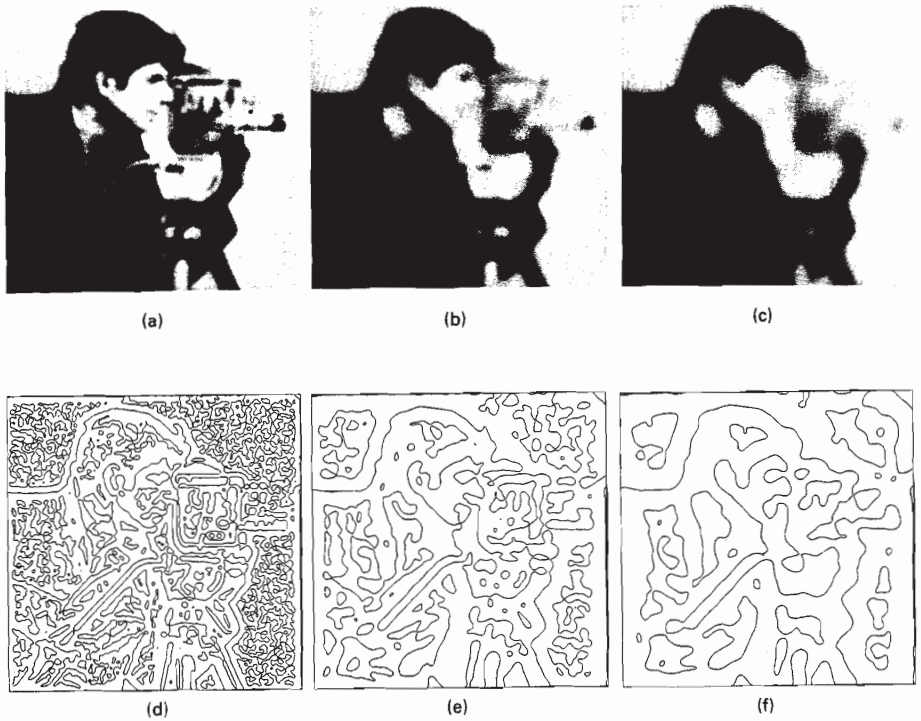


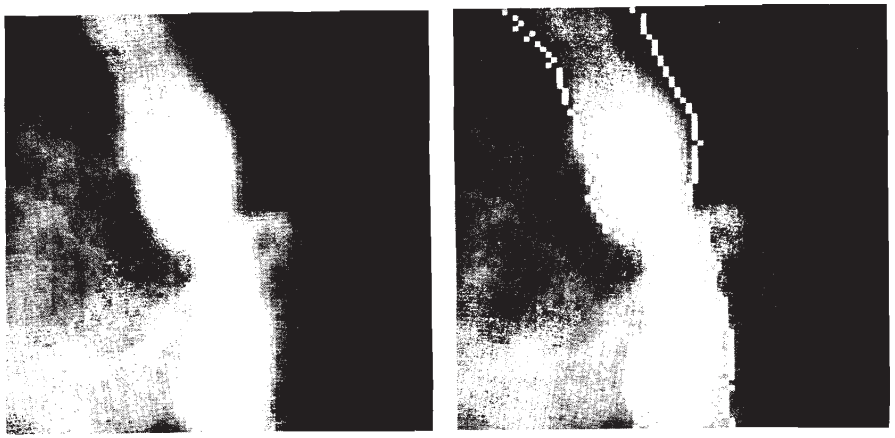
Figure 8.37 Edge maps obtained from lowpass filtered image. Blurred image with (a) $\sigma^2 = 4$; (b) $\sigma^2 = 16$; (c) $\sigma^2 = 36$. Result of applying Laplacian-based algorithm to the blurred image; (d) $\sigma^2 = 4$; (e) $\sigma^2 = 16$; (f) $\sigma^2 = 36$.

tinuous and has elliptical cross sections. The elliptical shape is chosen because of the small number of parameters involved in its characterization and because of some empirical evidence that it leads to a good estimate of percentage of stenosis. The 1-D cross section of $v(n_1, n_2)$, which consists of one blood vessel, is totally specified by three parameters, two representing the blood vessel boundaries and one related to the x-ray attenuation coefficient of iodine. The continuity of the vessel is guaranteed by fitting a cubic spline function to the vessel boundaries. The background $p(n_1, n_2)$ is modeled by a 2-D low-order polynomial. Low-order polynomials are very smooth functions, and their choice is motivated by the observation that objects in the background, such as tissue and bone, are much bigger than the blood vessels. The blurring function $g(n_1, n_2)$ is modeled by a known 2-D Gaussian function that takes into account the blurring introduced at various stages of the imaging process. The noise $w(n_1, n_2)$ is random background noise and is assumed to be white. The parameters in the model of $f(n_1, n_2)$ are the

Figure 8.38 Coronary angiogram.

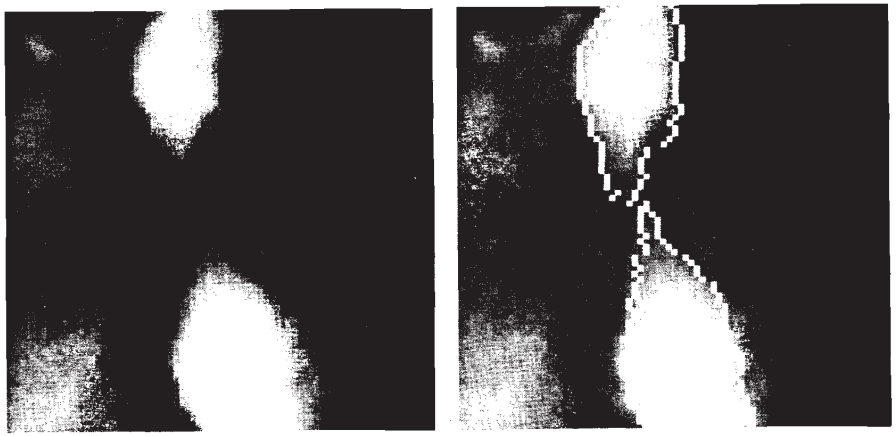
vessel parameters, the polynomial coefficients of $p(n_1, n_2)$, and the noise variance.

The vessels, tissues, bones, and the radiographic imaging process are much more complicated than suggested by the simple model presented above. Nevertheless, the model has been empirically observed to lead to good estimates of the vessel boundaries and corresponding percentage of stenosis. The model parameters may be estimated by a variety of different procedures. One possibility is the maximum likelihood (ML) parameter estimation method discussed in Section 6.1.5. In the ML method, the unknown parameters denoted by $\boldsymbol{\theta}$ are estimated by maximizing the probability density function $p_{f(n_1,n_2)|\boldsymbol{\theta}}(f_0(n_1, n_2)|\boldsymbol{\theta}_0)$ where $f(n_1, n_2)$ is the angiogram observation and $\boldsymbol{\theta}$ represents all the unknown parameters to be estimated. The ML method applied to vessel boundary detection is a nonlinear problem, but has been solved approximately [Pappas and Lim]. Figures 8.39 and 8.40 illustrate the results of applying the ML parameter estimation method to the detection of the blood vessels using the 1-D version of the 2-D model in (8.20). In the 1-D version, $f(n_1, n_2)$ in (8.20) is considered a 1-D sequence with variable n_1 for each n_2. Computations simplify considerably when the 1-D model is used. Figure 8.39(a) shows the original angiogram of 80 $\times$ 80 pixels, and Figure 8.39(b) shows the detected vessel boundaries superimposed on the original image. Figure 8.40 is another example. Developing an edge detection algorithm specific to an application problem is considerably more complicated than applying the general edge detection algorithms discussed in previous sections. However, it has the potential of leading to much more accurate edge detection.



(a) (b)

Figure 8.39 Example of vessel boundary detection from an angiogram by signal modeling. (a) Angiogram of 80 × 80 pixels; (b) vessel boundary detection.



(a) (b)

Figure 8.40 Another example of vessel boundary detection from an angiogram by signal modeling. (a) Angiogram of 80 × 80 pixels; (b) vessel boundary detection.

8.4 IMAGE INTERPOLATION AND MOTION ESTIMATION

In signal interpolation, we reconstruct a continuous signal from samples. Image interpolation has many applications. It can be used in changing the size of a digital image to improve its appearance when viewed on a display device. Consider a digital image of 64 × 64 pixels. If the display device performs zero-order hold, each individual pixel will be visible and the image will appear blocky. If the image size is increased through interpolation and resampling prior to its display, the resulting image will appear smoother and more visually pleasant. A sequence of image frames can also be interpolated along the temporal dimension. A 24 frames/sec motion picture can be converted to a 60 fields/sec NTSC signal for TV through interpolation. Temporal interpolation can also be used to improve the appearance of a slow-motion video.

Interpolation can also be used in other applications such as image coding. For example, a simple approach to bit rate reduction would be to discard some pixels or some frames and recreate them from the coded pixels and frames.

8.4.1 Spatial Interpolation

Consider a 2-D sequence $f(n_1, n_2)$ obtained by sampling an analog signal $f_c(x, y)$ through an ideal A/D converter:

$$f(n_1, n_2) = f_c(x, y)|_{x = n_1 T_1, y = n_2 T_2}. \tag{8.21}$$

If $f_c(x, y)$ is bandlimited and the sampling frequencies $1/T_1$ and $1/T_2$ are higher than the Nyquist rate, from Section 1.5 $f_c(x, y)$ can be reconstructed from $f(n_1, n_2)$ with an ideal D/A converter by

$$f_c(x, y) = \sum_{n_1 = -\infty}^{\infty} \sum_{n_2 = -\infty}^{\infty} f(n_1, n_2) h(x - n_1 T_1, y - n_2 T_2) \tag{8.22}$$

where $h(x, y)$ is the impulse response of an ideal separable analog lowpass filter given by

$$h(x, y) = \frac{\sin \frac{\pi}{T_1} x \sin \frac{\pi}{T_2} y}{\frac{\pi}{T_1} x \quad \frac{\pi}{T_2} y}. \tag{8.23}$$

There are several difficulties in using (8.22) and (8.23) for image interpolation. The analog image $f_c(x, y)$, even with an antialiasing filter, is not truly bandlimited, so aliasing occurs when $f_c(x, y)$ is sampled. In addition, $h(x, y)$ in (8.23) is an infinite-extent function, so evaluation of $f_c(x, y)$ using (8.22) cannot be carried out in practice. To approximate the interpolation by (8.22) and (8.23), one approach is to use a lowpass filter $h(x, y)$ that is spatially limited. For a spatially limited

$h(x, y)$, the summation in (8.22) has a finite number of nonzero terms. If $h(x, y)$ is a rectangular window function given by

$$h(x, y) = 1, \qquad -\frac{T_1}{2} \le x \le \frac{T_1}{2}, \qquad -\frac{T_2}{2} \le y \le \frac{T_2}{2} \tag{8.24}$$

then it is called *zero-order interpolation*. In zero-order interpolation, $\hat{f}_c(x, y)$ is chosen as $f(n_1, n_2)$ at the pixel closest to (x, y). Other examples of $h(x, y)$ which are more commonly used are functions of smoother shape, such as the spatially limited Gaussian function or the windowed ideal lowpass filter.

Another simple method widely used in practice is *bilinear interpolation*. In this method, $f_c(x, y)$ is evaluated by a linear combination of $f(n_1, n_2)$ at the four closest pixels. Suppose we wish to evaluate $f(x, y)$ for $n_1T_1 \le x \le (n_1 + 1)T_1$ and $n_2T_2 \le y \le (n_2 + 1)T_2$, as shown in Figure 8.41. The interpolated $\hat{f}_c(x, y)$ in the bilinear interpolation method is

$$\begin{aligned}\hat{f}_c(x, y) = {} & (1 - \Delta_x)(1 - \Delta_y)f(n_1, n_2) + (1 - \Delta_x)\Delta_y f(n_1, n_2 + 1) \\ & + \Delta_x(1 - \Delta_y)f(n_1 + 1, n_2) + \Delta_x\Delta_y f(n_1 + 1, n_2 + 1)\end{aligned} \tag{8.25a}$$

where

$$\Delta_x = (x - n_1T_1)/T_1 \tag{8.25b}$$

and

$$\Delta_y = (y - n_2T_2)/T_2. \tag{8.25c}$$

Another method is *polynomial interpolation*. Consider a local spatial region, say 3 × 3 or 5 × 5 pixels, over which $f(x, y)$ is approximated by a polynomial. The interpolated image $\hat{f}_c(x, y)$ is

$$\hat{f}_c(x, y) = \sum_{i=1}^{N} S_i\phi_i(x, y) \tag{8.26}$$

where $\phi_i(x, y)$ is a term in a polynomial. An example of $\phi_i(x, y)$ when $N = 6$ is

$$\phi_i(x, y) = 1, x, y, x^2, y^2, xy. \tag{8.27}$$

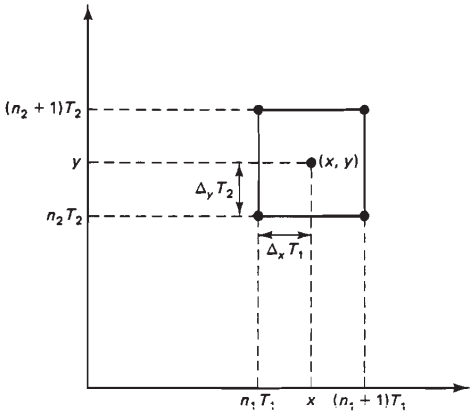


Figure 8.41 Region where $f_c(x, y)$ is interpolated from the four neighboring pixels $f_c(n_1T_1, n_2T_2)$, $f_c((n_1 + 1)T_1, n_2T_2)$, $f_c(n_1T_1, (n_2 + 1)T_2)$, and $f_c((n_1 + 1)T_1, (n_2 + 1)T_2)$.

The coefficients S_i can be determined by minimizing

$$\text{Error} = \sum_{(n_1, n_2) \in \psi} \sum \left[f(x, y) - \sum_{i=1}^{N} S_i\phi_i(x, y) \right]^2 \Bigg|_{x = n_1T_1, y = n_2T_2} \tag{8.28}$$

where ψ denotes the pixels over which $f(x, y)$ is approximated. Solving (8.28) is a simple linear problem, since the $\phi_i(x, y)$ are fixed. Advantages of polynomial interpolation include the smoothness of $\hat{f}_c(x, y)$ and the simplicity of evaluating $\partial\hat{f}_c(x, y)/\partial x$ and $\partial\hat{f}_c(x, y)/\partial y$, partial derivatives which are used in such applications as edge detection and motion estimation. In addition, by fitting a polynomial with fewer coefficients than the number of pixels in the region ψ in (8.28), some noise smoothing can be accomplished. Noise smoothing is particularly useful in applications where the partial derivatives $\partial\hat{f}_c(x, y)/\partial x$ and $\partial\hat{f}_c(x, y)/\partial y$ are used.

Spatial interpolation schemes can also be developed using motion estimation algorithms discussed in the next section. One example, where an image frame that consists of two image fields is constructed from a single image field, is discussed in Section 8.4.4.

Figure 8.42 shows an example of image interpolation. Figure 8.42(a) shows an image of 256 × 256 pixels interpolated by zero-order hold from an original image of 64 × 64 pixels. Figure 8.42(b) shows an image of 256 × 256 pixels obtained by bilinear interpolation of the same original image of 64 × 64 pixels.

8.4.2 Motion Estimation

New image frames can be created from existing ones through temporal interpolation. Unlike spatial interpolation, temporal interpolation requires a large amount

(a) (b)

Figure 8.42 Example of spatial interpolation. (a) Image of 256 × 256 pixels interpolated by zero-order hold from an original image of 64 × 64 pixels; (b) image of 256 × 256 pixels obtained by bilinear interpolation of the same original image used in (a).

of storage. Therefore a new frame is usually created from two adjacent frames, one in the past and one in the future relative to the frame being created.

The simplest method, often used in practice, is the zero-order hold method, in which a new frame is created by repeating the existing frame which is closest in time. In transforming a 24 frames/sec motion picture to a 60 fields/sec NTSC signal, three consecutive fields are created from a single motion picture frame, and the next two consecutive fields are created from the next motion picture frame. This process is repeated for the entire motion picture. This is known as the 3:2 pull-down method. For most scenes without large global motion, the results are quite good. If there is large global motion, however, zero-order hold temporal interpolation can cause noticeable jerkiness of motion. One method to improve the performance of temporal interpolation is through motion compensation.

A motion picture or television broadcast is a sequence of still frames that are displayed in rapid succession. The frame rate necessary to achieve proper motion rendition is usually high enough to ensure a great deal of temporal redundancy among adjacent frames. Much of the variation in intensity from one frame to the next is due to object motion. The process of determining the movement of objects within a sequence of image frames is known as *motion estimation*. Processing images accounting for the presence of motion is called *motion-compensated image processing*.

Motion-compensated image processing has a variety of applications. One application is image interpolation. By estimating motion parameters, we can create a new frame between two adjacent existing frames. The application of motion-compensated processing to image interpolation is discussed in the next section. Another application is image restoration. If we can estimate the motion parameters and identify regions in different frames where image intensities are expected to be the same or similar, temporal filtering can be performed in those regions. Application to image restoration is discussed in Chapter 9. Motion-compensated image processing can also be applied to image coding. By predicting the intensity of the current frame from the previous frames, we can limit our coding to the difference in intensities between the current frame and the predicted current frame. In addition, we may be able to discard some frames and reconstruct the discarded frames through interpolation from the coded frames. Application to image coding is discussed in Chapter 10.

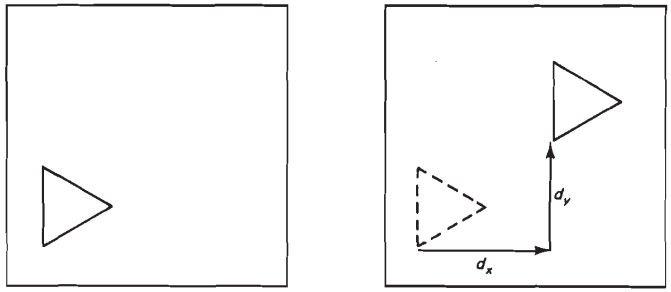
The motion estimation problem we consider here is the translational motion of objects. Let $f(x, y, t_{-1})$ and $f(x, y, t_0)$ denote the image intensity at times t_{-1} and t_0, respectively. We will refer to $f(x, y, t_{-1})$ and $f(x, y, t_0)$ as the past and current frame. We assume that

$$f(x, y, t_0) = f(x - d_x, y - d_y, t_{-1}) \tag{8.29}$$

where d_x and d_y are the horizontal and vertical displacement between t_{-1} and t_0. An example of $f(x, y, t_{-1})$ and $f(x, y, t_0)$ which satisfy (8.29) is shown in Figure 8.43. If we assume uniform motion between t_{-1} and t_0,

$$f(x, y, t) = f(x - v_x(t - t_{-1}), y - v_y(t - t_{-1}), t_{-1}), \quad t_{-1} \le t \le t_0 \tag{8.30}$$

where v_x and v_y are the uniform horizontal and vertical velocities.



(a) $f(x, y, t_{-1})$ (b) $f(x, y, t_0)$

Figure 8.43 Image translated with displacement of (d_x, d_y). (a) $f(x, y, t_{-1})$; (b) $f(x, y, t_0)$.

A direct consequence of (8.30) is a differential equation which relates v_x and v_y to $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$, which is valid in the spatio-temporal region over which uniform translational motion is assumed. To derive the relationship, let $f(x, y, t_{-1})$ be denoted by $s(x, y)$:

$$s(x, y) = f(x, y, t_{-1}). \tag{8.31}$$

From (8.30) and (8.31),

$$f(x, y, t) = s(\alpha(x, y, t), \beta(x, y, t)), \quad t_{-1} \le t \le t_0 \tag{8.32a}$$

where $$\alpha(x, y, t) = x - v_x(t - t_{-1}) \tag{8.32b}$$

and $$\beta(x, y, t) = y - v_y(t - t_{-1}). \tag{8.32c}$$

From (8.32), assuming $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$ exist, we obtain

$$\frac{\partial f(x, y, t)}{\partial x} = \frac{\partial s}{\partial \alpha}\frac{\partial \alpha}{\partial x} + \frac{\partial s}{\partial \beta}\frac{\partial \beta}{\partial x} = \frac{\partial s}{\partial \alpha} \tag{8.33a}$$

$$\frac{\partial f(x, y, t)}{\partial y} = \frac{\partial s}{\partial \alpha}\frac{\partial \alpha}{\partial y} + \frac{\partial s}{\partial \beta}\frac{\partial \beta}{\partial y} = \frac{\partial s}{\partial \beta} \tag{8.33b}$$

$$\frac{\partial f(x, y, t)}{\partial t} = \frac{\partial s}{\partial \alpha}\frac{\partial \alpha}{\partial t} + \frac{\partial s}{\partial \beta}\frac{\partial \beta}{\partial t} = -v_x\frac{\partial s}{\partial \alpha} - v_y\frac{\partial s}{\partial \beta}. \tag{8.33c}$$

From (8.33),

$$v_x\frac{\partial f(x, y, t)}{\partial x} + v_y\frac{\partial f(x, y, t)}{\partial y} + \frac{\partial f(x, y, t)}{\partial t} = 0. \tag{8.34}$$

Equation (8.34) is called a *spatio-temporal constraint equation* and can be generalized to incorporate other types of motion, such as zooming [Martinez].

The assumption of simple translation that led to (8.29) and the additional assumption of translation with uniform velocity that led to (8.34) are highly re-

strictive. For example, they do not allow for object rotation, camera zoom, regions uncovered by translational object motion, or multiple objects moving with different v_x and v_y. However, by assuming uniform translational motion only locally and estimating the two motion parameters (d_x, d_y) or (v_x, v_y) at each pixel or at each small subimage, (8.29) and (8.34) are valid for background regions that are not affected by object motion and for regions occupied by objects which do indeed translate with a uniform velocity. Such regions occupy a significant portion of a typical sequence of image frames. In addition, if we identify the regions where motion estimates are not accurate, we can suppress motion-compensated processing in those regions. In image interpolation, for example, we can assume that v_x and v_y are zero.

Motion estimation methods can be classified broadly into two groups, that is, region matching methods and spatio-temporal constraint methods. Region matching methods are based on (8.29), and spatio-temporal constraint methods are based on (8.34). We first discuss region matching methods.

Region matching methods. Region matching methods involve considering a small region in an image frame and searching for the displacement which produces the "best match" among possible regions in an adjacent frame. In region matching methods, the displacement vector (d_x, d_y) is estimated by minimizing

$$\text{Error} = \iint_{(x,y)\in R} C[f(x, y, t_0), f(x - d_x, y - d_y, t_{-1})]\, dx\, dy \qquad (8.35)$$

where R is the local spatial region used to estimate (d_x, d_y) and $C[\cdot,\cdot]$ is a metric that indicates the amount of dissimilarity between two arguments. The integrals in (8.35) can be replaced by summation if $f(x, y, t)$ is sampled at the spatial variables (x, y). If we estimate (d_x, d_y) at time t_0, the region R is the local spatial region that surrounds the particular spatial position at which (d_x, d_y) is estimated. The size of R is dictated by several considerations. If it is chosen too large, the assumption that (d_x, d_y) is approximately constant over the region R may not be valid and evaluation of the error expression requires more computations. If it is chosen too small, the estimates may become very sensitive to noise. One reasonable choice based on these considerations is a 5×5-pixel region. There are many possible choices for the dissimilarity function $C[\cdot, \cdot]$. Two commonly used choices are the squared difference and absolute difference between the two arguments. With these choices of $C[\cdot, \cdot]$, (8.35) reduces to

$$\text{Error} = \iint_{(x,y)\in R} [f(x, y, t_0) - f(x - d_x, y - d_y, t_{-1})]^2\, dx\, dy \qquad (8.36)$$

$$\text{Error} = \iint_{(x,y)\in R} |f(x, y, t_0) - f(x - d_x, y - d_y, t_{-1})|\, dx\, dy. \qquad (8.37)$$

The function $f(x, y, t_0) - f(x - d_x, y - d_y, t_{-1})$ is called the *displaced frame difference*. In typical applications of motion-compensated processing, the system performance is not very sensitive to the specific choice of the dissimilarity function.

To the extent that (8.29) is valid, the error expression in (8.36) or (8.37) is zero at the correct (d_x, d_y).

Minimizing the Error in (8.36) or (8.37) is a nonlinear problem. Attempts to solve this nonlinear problem have produced many variations, which can be grouped into block matching and recursive methods. We discuss block matching methods first.

One straightforward approach to solve the above minimization problem is to evaluate the Error for every possible (d_x, d_y) within some reasonable range and choose (d_x, d_y) at which the Error is minimum. In this approach, a block of pixel intensities at time t_0 is matched directly to a block at time t_{-1}. This is the basis of *block matching methods*. Since the error expression has to be evaluated at many values of (d_x, d_y), this method of estimating (d_x, d_y) is computationally very expensive and many methods have been developed to reduce computations. In one simplification, we assume that (d_x, d_y) is constant over a block, say of 7×7 pixels. Under this assumption, we divide the image into many blocks and we estimate (d_x, d_y) for each block. Even though we generally choose the block size to be the same as the size of R in (8.35), it is not necessary to do so. In another simplification, we can limit the search space to integer values of (d_x, d_y). In addition to reducing the search space from continuous variables (d_x, d_y) to discrete variables, limiting the search space to integer values allows us to determine $f(n_1 - d_x, n_2 - d_y, t_{-1})$, necessary in the evaluation of the error expression, without interpolation. However, the estimates of (d_x, d_y) are restricted to discrete values.

We can reduce the computational requirements in block matching methods further by using a more efficient search procedure than a brute force search. One such method is called a three-step search method, illustrated in Figure 8.44. In the first step of this method, the error expression is evaluated at nine values of (d_x, d_y) which are marked by "1" and filled circles. Among these nine values, we choose (d_x, d_y) with the smallest Error. Suppose the smallest Error is at $(d_x = 3, d_y = -3)$. In the second step, we evaluate the error expression at eight additional values of (d_x, d_y) which are marked by "2" and filled squares. We now choose (d_x, d_y) from nine values [eight new values and $(d_x = 3, d_y = -3)$]. This procedure is repeated one more time. At the end of the third step, we have an estimate of (d_x, d_y). This procedure can be easily generalized to more than three steps to increase the range of possible (d_x, d_y). Another search method is to estimate d_x first by searching $(d_x, 0)$. Once d_x is estimated, say $\hat{d}_x$, d_y is estimated by searching $(\hat{d}_x, d_y)$. If we wish to improve the estimate further, we can reestimate d_x by searching $(d_x, \hat{d}_y)$ where $\hat{d}_y$ is the estimate of d_y obtained in the previous step. At each step in this procedure, we estimate only one parameter, which is considerably simpler than estimating two parameters jointly. These heuristic methods reduce the number of computations by reducing the number of values of (d_x, d_y) at which the error expression is evaluated. However, the Error at the estimated (d_x, d_y) may not be the global minimum.

In block matching methods, we estimate (d_x, d_y) by explicitly evaluating the Error at some specified set of (d_x, d_y). As an alternative, we can use descent algorithms such as the steepest descent, Newton-Raphson, and Davidon-Fletcher-Powell methods discussed in Section 5.2.3, to solve the nonlinear problem of

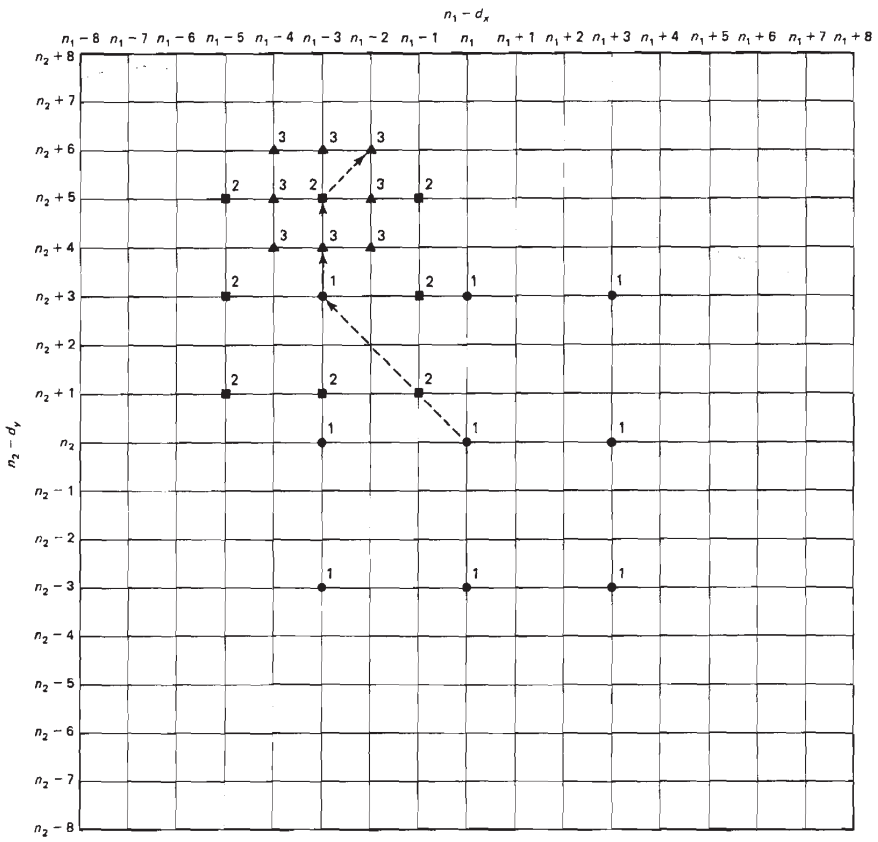


Figure 8.44 Illustration of three-step search method.

minimizing the Error with respect to (d_x, d_y). In this class of algorithms, a recursive (iterative) procedure is used to improve the estimate in each iteration. For this reason, they are called *recursive methods*.

Let $(\hat{d}_x(k), \hat{d}_y(k))$ denote the estimate of (d_x, d_y) after the kth iteration. In recursive methods, the estimate of (d_x, d_y) after the $k + 1$th iteration, $(\hat{d}_x(k + 1), \hat{d}_y(k + 1))$, is obtained by

$$\hat{d}_x(k + 1) = \hat{d}_x(k) + u_x(k) \tag{8.38a}$$

$$\hat{d}_y(k + 1) = \hat{d}_y(k) + u_y(k) \tag{8.38b}$$

where $u_x(k)$ and $u_y(k)$ are the update or correction terms. The update terms vary

depending on the descent method used. If we use the steepest descent method, for example, (8.38) becomes

$$\hat{d}_x(k + 1) = \hat{d}_x(k) - \epsilon \left. \frac{\partial \text{Error}\ (d_x, d_y)}{\partial d_x} \right|_{d_x = \hat{d}_x(k), d_y = \hat{d}_y(k)} \tag{8.39a}$$

$$\hat{d}_y(k + 1) = \hat{d}_y(k) - \epsilon \left. \frac{\partial \text{Error}\ (d_x, d_y)}{\partial d_y} \right|_{d_x = \hat{d}_x(k), d_y = \hat{d}_y(k)} \tag{8.39b}$$

where ϵ is a step size that can be adjusted and $\text{Error}(d_x, d_y)$ is the Error in (8.35) as a function of d_x and d_y for a given R. Recursive methods typically involve partial derivatives and tend to be sensitive to the presence of noise or fine details in an image. Smoothing the image before motion estimation often improves the performance of recursive methods.

In recursive methods, (d_x, d_y) is not limited to integer values and can be estimated within subpixel accuracy. Update terms typically include evaluation of partial derivatives of $\text{Error}(d_x, d_y)$, which involves evaluation of $f(x, y, t_{-1})$ and its partial derivatives at an arbitrary spatial point. In practice, $f(x, y, t_{-1})$ is known only at $x = n_1 T_1$ and $y = n_2 T_2$. To evaluate the necessary quantities at an arbitrary spatial point (x, y), we can use the spatial interpolation techniques discussed in Section 8.4.1.

In recursive methods, (d_x, d_y) is typically estimated at each pixel. In using the recursion relationship in (8.38), $(\hat{d}_x(0), \hat{d}_y(0))$ is typically obtained from the estimate at the adjacent pixel in the same horizontal scan line, in the adjacent scan line, or in the adjacent frame. These methods are called pel (picture element) recursive estimation with horizontal, vertical, and temporal recursion, respectively. Given $(\hat{d}_x(k), \hat{d}_y(k))$, we can use the recursion relation in (8.38) only once for a pixel and then move on to the next pixel. Alternatively, we can use the recursion relation more than once for a more accurate estimate of (d_x, d_y) before we move on to the next pixel.

Although we classified region matching methods into block matching methods and recursive methods to be consistent with the literature, the boundary between the two classes of methods is fuzzy. By choosing a finer grid at which the error expression is evaluated, we can also estimate (d_x, d_y) within subpixel accuracy with block matching methods. In addition, the three-step search procedure discussed as a block matching method can be viewed as a recursive method in which the estimate is improved iteratively. A major disadvantage of region matching methods is in the amount of computation required. Even though only two parameters, d_x and d_y, must be estimated, solving the nonlinear problem at each pixel or at each small subimage can be computationally very expensive.

Spatio-temporal constraint methods. Algorithms of this class are based on the spatio-temporal constraint equation (8.34), which can be viewed as a linear equation for two unknown parameters v_x and v_y under the assumption that $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$ are given. By evaluating $\partial f(x, y, t)/\partial x$,

$\partial f(x, y, t)/\partial y$ and $\partial f(x, y, t)/\partial t$ at many points (x_i, y_i, t_i) for $1 \le i \le N$ at which v_x and v_y are assumed constant, we can obtain an overdetermined set of linear equations:

$$v_x \left.\frac{\partial f(x, y, t)}{\partial x}\right|_{(x_i,y_i,t_i)} + v_y \left.\frac{\partial f(x, y, t)}{\partial y}\right|_{(x_i,y_i,t_i)} + \left.\frac{\partial f(x, y, t)}{\partial t}\right|_{(x_i,y_i,t_i)} \simeq 0, \; 1 \le i \le N. \tag{8.40}$$

The velocity estimates can be obtained by minimizing

$$\text{Error} = \sum_{i=1}^{N} \left[v_x \left.\frac{\partial f(x, y, t)}{\partial x}\right|_{(x_i,y_i,t_i)} + v_y \left.\frac{\partial f(x, y, t)}{\partial y}\right|_{(x_i,y_i,t_i)} + \left.\frac{\partial f(x, y, t)}{\partial t}\right|_{(x_i,y_i,t_i)} \right]^2. \tag{8.41}$$

Since the error expression in (8.41) is a quadratic form of the unknown parameters v_x and v_y, the solution requires solving two linear equations. More generally, suppose (8.34) is valid in a local spatio-temporal region denoted by ψ. To estimate v_x and v_y, we minimize

$$\text{Error} = \iiint_{(x,y,t)\in\psi} \left(v_x \frac{\partial f(x, y, t)}{\partial x} + v_y \frac{\partial f(x, y, t)}{\partial y} + \frac{\partial f(x, y, t)}{\partial t} \right)^2 dx\, dy\, dt. \tag{8.42}$$

The integrals in (8.42) may be replaced by summations. One such example is (8.41). Differentiating the Error in (8.42) with respect to v_x and v_y and setting the results to zero leads to

$$W\mathbf{v} = \boldsymbol{\gamma} \tag{8.43a}$$

where

$$W = \begin{bmatrix} \iiint_{(x,y,t)\in\psi} \left(\frac{\partial f(x, y, t)}{\partial x} \right)^2 dx\, dy\, dt & \iiint_{(x,y,t)\in\psi} \frac{\partial f(x, y, t)}{\partial x} \frac{\partial f(x, y, t)}{\partial y}\, dx\, dy\, dt \\ \iiint_{(x,y,t)\in\psi} \frac{\partial f(x, y, t)}{\partial x} \frac{\partial f(x, y, t)}{\partial y}\, dx\, dy\, dt & \iiint_{(x,y,t)\in\psi} \left(\frac{\partial f(x, y, t)}{\partial y} \right)^2 dx\, dy\, dt \end{bmatrix} \tag{8.43b}$$

$$\mathbf{v} = [v_x, v_y]^T \tag{8.43c}$$

$$\boldsymbol{\gamma} = - \begin{bmatrix} \iiint_{(x,y,t)\in\psi} \frac{\partial f(x, y, t)}{\partial x} \frac{\partial f(x, y, t)}{\partial t}\, dx\, dy\, dt \\ \iiint_{(x,y,t)\in\psi} \frac{\partial f(x, y, t)}{\partial y} \frac{\partial f(x, y, t)}{\partial t}\, dx\, dy\, dt \end{bmatrix}. \tag{8.43d}$$

The two linear equations in (8.43) may have multiple solutions. Suppose $f(x, y, t)$ is constant in the spatio-temporal region ψ. Then $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$ are all zero, and all the elements in W and $\boldsymbol{\gamma}$ in (8.43) are zero. Therefore, any (v_x, v_y) will satisfy (8.43a). Any velocity in a uniform intensity region will not affect $f(x, y, t)$, so the true velocity cannot be estimated from $f(x, y, t)$. Suppose $f(x, y, t)$ is a perfect step edge. The velocity along the direction parallel to the step edge will not affect $f(x, y, t)$ and therefore cannot be estimated. These problems have been studied, and a solution has been developed [Martinez]. Let λ_1 and λ_2 denote the eigenvalues of W, and let $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ denote the corresponding orthonormal eigenvectors. A reasonable solution to (8.43) is given by (see Problem 8.24)

Case 1. $\mathbf{v} = 0, \quad \lambda_1, \lambda_2 < \text{threshold}$ (8.44a)

Case 2. $\mathbf{v} = \dfrac{[\boldsymbol{\alpha}_1^T \boldsymbol{\gamma}]\boldsymbol{\alpha}_1}{\lambda_1}, \quad \lambda_1 \gg \lambda_2$ (8.44b)

Case 3. $\mathbf{v} = W^{-1}\boldsymbol{\gamma}, \quad \text{otherwise}$ (8.44c)

Case 1 includes the uniform intensity region, where the velocity is set to zero. Case 2 includes the perfect step edge region, and the velocity estimate in (8.44b) is along the direction perpendicular to the step edge.

Solving the linear equations in (8.43) requires evaluation of $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$ at arbitrary spatio-temporal positions. This can be accomplished by extending the spatial polynomial interpolation method to 3-D, which has the advantage over other approaches in computational simplicity and robustness to noise. In the 3-D polynomial interpolation, the interpolated $\hat{f}(x, y, t)$ is

$$\hat{f}(x, y, t) = \sum_{i=1}^{N} S_i \phi_i(x, y, t). \tag{8.45}$$

An example of $\phi_i(x, y, t)$ for $N = 9$ is

$$\phi_i(x, y, t) = 1, x, y, t, x^2, y^2, xy, xt, yt. \tag{8.46}$$

The coefficients S_i can be determined by minimizing

$$\text{Error} = \sum_{(n_1,n_2,n_3)\in\psi} \sum \sum \left(f(x, y, t) - \sum_{i=1}^{N} S_i \phi_i(x, y, t) \right)^2 \Bigg|_{x=n_1T_1, y=n_2T_2, t=n_3T_3} \tag{8.47}$$

One reasonable choice of the region ψ typically contains 50 pixels: 5 for n_1, 5 for n_2, and 2 for t. Minimizing the error expression in (8.47) with respect to S_i requires solving a set of linear equations. Note that the partial derivatives $\partial f(x, y, t)/\partial x$, $\partial f(x, y, t)/\partial y$, and $\partial f(x, y, t)/\partial t$ used in (8.43) can be precomputed in terms of S_i.

The motion estimation algorithms discussed above require determination of the spatio-temporal regions denoted by ψ over which the uniform translational motion can be assumed. Since a local spatial region in a frame is on the order of 5×5 pixels in size, determining a reasonable ψ requires an initial displacement estimate within a few pixels of the true displacement. In practice, it is not un-

common for the displacement between two adjacent frames to be on the order of 10 pixels. One approach to obtaining an initial displacement (or velocity) is to use a previously computed velocity in the neighborhood and then determine the appropriate ψ as shown in Figure 8.45. Another approach is the hierarchical method [Bierling and Thoma] or the multigrid method [Martinez]. The multigrid method begins with the velocity estimated on a coarse grid. The coarse grid has been obtained from the original frame by lowpass filtering and down-sampling. Down-sampling contracts the displacement (or velocity). A large velocity in the original frame is scaled down by the down-sampling factor. The velocities in the down-sampled frames can be estimated by using the spatio-temporal constraint method with an assumed initial velocity of zero. The velocities estimated on the coarse grid are interpolated to generate initial estimates of the velocities at a finer grid. The bilinear interpolation may be used in the velocity interpolation. The multigrid method can be viewed as an example of *pyramid processing*, which exploits a data structure called a pyramid. A pyramid provides successively condensed information of an image. An example of a pyramid is a higher resolution image and successively lower resolution images. Pyramid processing is useful in various applications including image coding and is discussed further in Chapter 10.

A major advantage of the spatio-temporal constraint methods over region matching methods is their computational simplicity. In addition, preliminary studies based on both synthetic and real data indicate that a spatio-temporal constraint method with polynomial interpolation for $f(x, y, t)$ performs as well as a region matching method for both noisy and noise-free image frames [Martinez].

Both the region matching and spatio-temporal constraint methods can produce substantial errors in motion estimation for some regions, either because the signal $f(x, y, t)$ cannot be modeled by uniform translational motion, or because the motion estimation algorithm does not perform well, perhaps due to the presence of noise. One means of detecting large errors is to compare the error expression in (8.35) for region matching methods or the error expression in (8.42) for spatio-temporal constraint methods with some threshold. The regions where the error is above the threshold can be declared to have unreliable motion estimates, and motion-compensated processing can be suppressed in such regions.

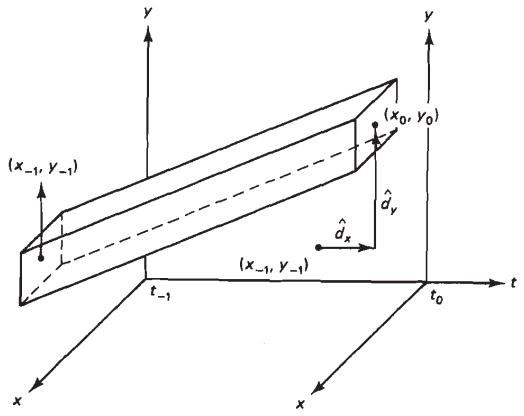


Figure 8.45 Region ψ used in (8.42) to estimate (d_x, d_y) at the spatial location (x_0, y_0) and time t_0. The estimate $(\hat{d}_x, \hat{d}_y)$ represents a previously computed displacement in the neighborhood.

8.4.3 Motion-Compensated Temporal Interpolation

Suppose we have two consecutive image frames $f(n_1, n_2, t_{-1})$ and $f(n_1, n_2, t_0)$, as shown in Figure 8.46. We wish to create a new frame $f(n_1, n_2, t)$ where $t_{-1} < t < t_0$. A simple approach is to choose the original frame that is closer in time to the desired frame. One problem with this approach is the jerkiness that results if the sequence has a large global motion.

An alternative is motion-compensated interpolation using the motion estimation algorithms discussed in the previous section. In motion-compensated interpolation, we assume the uniform translational motion model within a local spatio-temporal region. From $f(n_1, n_2, t_{-1})$ and $f(n_1, n_2, t_0)$, we compute the velocities at $f(n_1, n_2, t)$. We then project the velocities to the frame at t_{-1} or t_0 closer in time to the desired time t, as shown in Figure 8.46. Since the projected spatial point generally does not lie on the original sampling grid, spatial interpolation is needed to obtain the interpolated frame. If the velocity estimated at a particular pixel in $f(n_1, n_2, t_{-1})$ is not considered accurate enough, the velocity is assumed to be zero. In this case, the interpolated pixel value is identical to that at the same pixel location in $f(n_1, n_2, t_{-1})$ or $f(n_1, n_2, t_0)$, whichever is closer in time to the desired time t.

It is not possible to illustrate the motion rendition characteristics of motion-compensated frame interpolation by using only still pictures. However, we can get a rough idea by looking at a still frame created from two image frames by this method. Figure 8.47 shows a set of four frames: two original frames, shown in Figures 8.47(a) and (d), and the two interpolated frames, shown in Figures 8.47(b) and (c). The interpolated frame in (b) is by motion compensation. The frame in (c) is obtained by simply averaging the two key frames. This frame shows the amount of motion that occured between the two key frames. The four frames have a spatial resolution of 512 × 512 pixels. The interpolated frame corresponds to the time instant midway between the two key original frames. Note that the

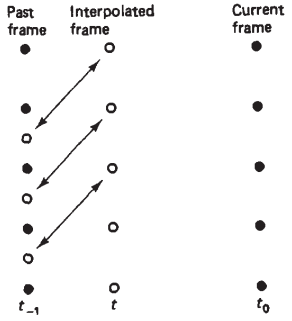


Figure 8.46 Creation of $f(n_1, n_2, t)$ by interpolation of $f(n_1, n_2, t_{-1})$ and $f(n_1, n_2, t_0)$. In this example, the displacement (d_x, d_y) is obtained at each pixel (n_1, n_2) at time t from $f(n_1, n_2, t_{-1})$ and $f(n_1, n_2, t_0)$. Each pixel at time t is projected to the corresponding spatial location at time t_{-1} (t closer to t_{-1} than to t_0 in this example) and the pixel intensity is determined from $f(n_1, n_2, t_{-1})$ at the projected pixel location. Note that spatial interpolation of $f(n_1, n_2, t_{-1})$ is usually necessary to perform this operation.

Figure 8.47 Example of temporal frame interpolation. (a) Original frame 1; (b) interpolated frame by motion compensation; (c) interpolated frame by averaging two key frames. This shows the amount of movement between the two key frames; (d) original frame 2.

interpolated image's quality in this example is essentially the same as that of the two key frames when motion compensation is used. The motion estimation method used here is the spatio-temporal constraint method with polynomial interpolation, discussed in Section 8.4.2.

Motion-compensated interpolation has been used in modifying the frame rate. Frame rate modification can be combined with time-scale modification of audio

[Lim] to modify the length of a motion picture or a TV program. Experience with typical scenes indicates that frame rate modification of video through motion-compensated interpolation can produce video of quality comparable to the original, except for somewhat unnatural motion rates for such actions as walking and talking [Martinez] which occur when the rate modification factor is sufficiently high.

8.4.4 Application of Motion Estimation Methods to Spatial Interpolation

The general idea behind motion-compensated temporal interpolation can be used to develop new algorithms for spatial interpolation. To examine these new algorithms, let us consider a specific spatial interpolation problem. As discussed in Section 7.4, an NTSC television system uses a 2:1 interlaced format, scanned at a rate of 30 frames/sec. A frame consists of 525 horizontal scan lines which are divided into two fields, the odd field consisting of odd-numbered lines and the even field consisting of even-numbered ones. Creating a frame at time t from a field at time t through spatial interpolation can be useful in many applications, including a 60 frames/sec television without bandwidth increase and improved vertical resolution of frozen frames.

The spatial interpolation techniques discussed in Section 8.4.1 may be used in creating a frame from a field, but incorporating some additional knowledge about images may improve the performance of spatial interpolation algorithms. Many elements in the visual world, such as contours and scratches, are spatially continuous. This information can be exploited in creating a frame from a field. Let $f(x, y_{-1})$ and $f(x, y_0)$ denote image intensities of two adjacent horizontal scan lines of a field. We wish to create a new horizontal scan line between $f(x, y_{-1})$ and $f(x, y_0)$. One model that takes into account the spatial continuity of such elements as contours and scratches is

$$f(x, y_0) = f(x - d_x, y_{-1}) \tag{8.48}$$

where d_x is a horizontal displacement between y_{-1} and y_0. Equation (8.48) can be viewed as a special case of the uniform translational velocity model of (8.29). The spatial variable y in (8.48) has a function very similar to the time variable t in (8.29), and there is only one spatial variable x in (8.48), while there are two spatial variables x and y in (8.29). As a result, all our discussions in Section 8.4.2 apply to the problem of estimating d_x in (8.48). For example, under the assumption of uniform velocity, (8.48) can be expressed as

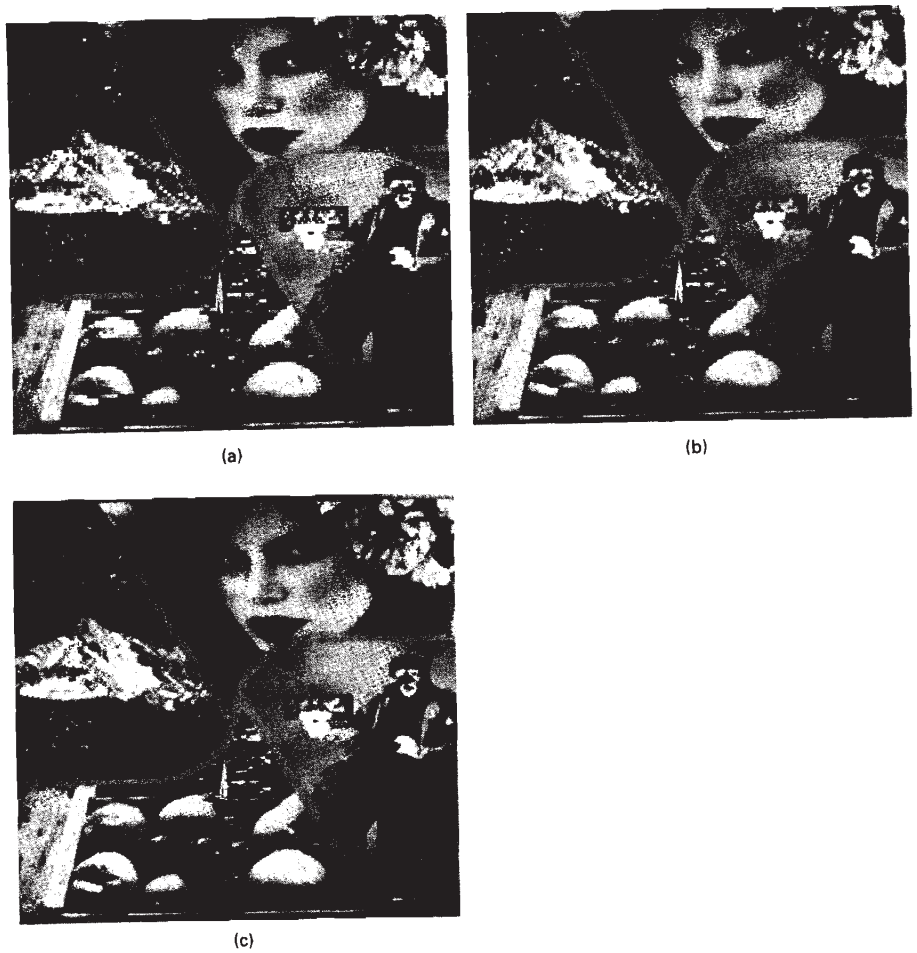
$$f(x, y) = f(x - v_x(y - y_{-1}), y_{-1}), \quad y_{-1} \leq y \leq y_0 \tag{8.49}$$

which leads directly to

$$v_x \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y} = 0. \tag{8.50}$$

Equation (8.48) can be used to develop region matching methods and (8.50) can be used to develop spatio-temporal constraint methods for estimating d_x or v_x.

Once the horizontal displacement (or velocity) is estimated, it can be used in spatial interpolation in a manner analogous to the temporal interpolation discussed in Section 8.4.3. Figure 8.48 illustrates the performance of a spatial interpolation algorithm based on (8.50). Figure 8.48(a) shows a frame of 256 × 256 pixels obtained by repeating each horizontal line of a 256 × 128-pixel image. Figure 8.48(b) shows the frame obtained by bilinear spatial interpolation. Figure 8.48(c) shows the frame obtained by estimating the horizontal displacement based on (8.50) and then using the estimate for spatial interpolation. Spatial continuity of lines and contours is preserved better in the image in Figure 8.48(c) than in the other two images in Figures 8.48(a) and (b).



(a) (b) (c)

Figure 8.48 Creation of a frame from a field by spatial interpolation. (a) Image of 256 × 256 pixels obtained from an image of 256 × 128 pixels by zero-order hold interpolation; (b) same as (a) obtained by bilinear interpolation; (c) same as (a) obtained by application of a motion estimation algorithm.

8.5 FALSE COLOR AND PSEUDOCOLOR

It is well known that the human visual system is quite sensitive to color. The number of distinguishable intensities, for example, is much smaller than the number of distinguishable colors and intensities. In addition, color images are generally much more pleasant to view than black-and-white images. The aesthetic aspect of color can be used for image enhancement. In some applications, such as television commercials, *false color* can be used to emphasize a particular object in an image. For example, a red banana in a surrounding of other fruits of natural color will receive more of a viewer's attention. In other applications, data that do not represent an image in the conventional sense can be represented by a color image. In this case, the color used is called *pseudocolor*. As an example, a speech spectrogram showing speech energy as a function of time and frequency can be represented by a color image, with silence, voiced segments, and unvoiced segments distinguished by different colors and energy represented by color brightness.

The use of color in image enhancement is limited only by artistic imaginations, and there are no simple guidelines or rules to follow. In this section, therefore, we will concentrate on three examples that illustrate the type of image enhancement that can be achieved by using color. In the first example, we transform a monochrome (black and white) image to a color image by using a very simple rule. To obtain a color image from a monochrome image, the monochrome image is first filtered by a lowpass filter, a bandpass filter, and a highpass filter. The lowpass filtered image is considered to be the blue component of the resulting color image. The bandpass filtered image is considered the green component, and the highpass filtered image is considered the red component. The three components—red, green, and blue—are combined to form a color image. Figure 8.49(a) (see color insert) shows an original monochrome image of 512 × 512 pixels. Figure 8.49(b) shows the color image obtained by using this procedure. The color is pleasant, but this arbitrary procedure does not generate a natural-looking color image. Changing classic black-and-white movies such as *Casablanca* or *It's A Wonderful Life* to color movies requires much more sophisticated processing and a great deal of human intervention.

In the second example, we consider the display of a 2-D spectral estimate on a CRT. The 2-D spectral estimate, represented by $P_x(\omega_1, \omega_2)$ in dB, is typically displayed by using a contour plot. An example of a 2-D maximum likelihood spectral estimate for the data of two sinusoids in white noise is shown in Figure 8.50(a). The maximum corresponds to 0 dB and the contours are in increments of 0.5 dB downward from the maximum point. As we discussed in Chapter 6, in such applications as detection of low-flying aircraft by an array of microphone sensors, we wish to determine the number of sinusoids present and their frequen-

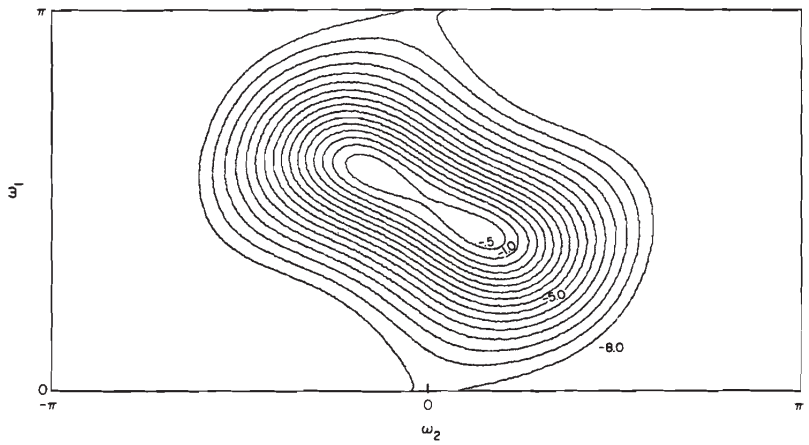


Figure 8.50 Display of spectral estimate using pseudocolor. (a) 2-D maximum likelihood spectral estimate represented by a contour plot; (b) spectral estimate in (a) represented by a color image (see color insert).

cies. An alternative way of representing the spectral estimate is to use pseudocolor. Figure 8.50(b) (see color insert) gives an example, where different amplitudes of $P_x(\omega_1, \omega_2)$ have been mapped to different colors. Comparing the two figures shows that the two peaks and their locations in the spectral estimate stand out more clearly in Figure 8.50(b)

The third example is the display of range information using color [Sullivan et al.]. In such applications as infrared radar imaging systems, range information and image intensity are available. Figure 8.51(a) (see color insert) shows an intensity image of several buildings located two to four kilometers away from the radar; the range information has been discarded. Figure 8.51(b) shows an image that uses color to display range information. The range value determines the hue, and the intensity determines the brightness level of the chosen hue. The most striking aspect of this technique is demonstrated by the observation that a horizontal line seen at close range (actually a telephone wire) is visible in Figure 8.51(b), but is completely obscured in Figure 8.51(a).

REFERENCES

For readings on gray scale modification, see [Hall et al. (1971); Troy et al.; Gonzales and Fittes; Woods and Gonzales]. For image enhancement by lowpass filtering and highpass filtering, see [O'Handley and Green; Hall and Awtrey]. For unsharp masking see [Schreiber (1970)]. For readings on homomorphic processing for image enhancement, see [Oppenheim et al.; Schreiber (1978)]. See [Peli and Lim] for adaptive image enhancement. For an overview of median filtering, see [Arce et al.]. For readings on theoretical results of median filtering, see [Gallagher and Wise; Nodes and Gallagher; Arce and McLoughlin]. For fast algorithms for median filtering, see [Huang et al.; Ataman et al.].

For a survey of edge detection methods, see [Davis; Shaw; Peli and Malah]. For readings on signal and image interpolation, see [Crochiere and Rabiner; Dubois]. For a survey of motion estimation methods, see [Musmann et al.; Aggarwal and Nandhakumar]. For region matching methods, see [Netravali and Robbins; Huang and Tsai; Srinivasan and Rao]. For spatio-temporal constraint methods, see [Paquin and Dubois; Martinez]. For readings on false color and pseudocolor, see [Gazley et al.; Sheppard; Kreins and Allison; Sullivan et al.].

H. L. Abrams, ed., *Coronary Artiography*. Boston: Little, Brown and Company, 1983.

J. K. Aggarwal and N. Nandhakumar, On the computation of motion from sequences of images—a review, *Proc. IEEE*, Vol. 76, August 1988, pp. 917–935.

G. R. Arce, N. C. Gallagher, and T. A. Nodes, Median filters: Theory for one or two dimensional filters, *Advances in Computer Vision and Image Processing*, T. S. Huang, ed., Greenwich, CT: JAI Press, 1986.

G. R. Arce and M. P. McLoughlin, Theoretical analysis of the max/median filter, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-35, January 1987, pp. 60–69.

E. Ataman, V. K. Aatre, and K. M. Wong, A fast method for real-time median filtering, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-28, August 1980, pp. 415–421.

M. Bierling and R. Thoma, Motion compensating field interpolation using a hierarchically structured displacement estimator, *Signal Processing*, Vol. 11, December 1986, pp. 387–404.

J. Canny, A computational approach to edge detection, *IEEE Trans. on Patt. Ana. Mach. Intell.*, Vol. PAMI-8, November 1986, pp. 679–698.

R. E. Crochiere and L. R. Rabiner, Interpolation and decimation of digital signals—a tutorial review, *Proc. IEEE*, Vol. 69, March 1981, pp. 300–331.

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

L. S. Davis and A. Mitiche, Edge detection in textures, *Computer Graphics and Image Processing*, Vol. 12, 1980, pp. 25–39.

E. Dubois, The sampling and reconstruction of time-varying imagery with application in video systems, *Proc. IEEE*, Vol. 73, April 1985, pp. 502–522.

R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

N. C. Gallagher, Jr. and G. L. Wise, A theoretical analysis of the properties of median filters, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-29, December 1981, pp. 1136–1141.

C. Gazley, J. E. Reiber, and R. H. Stratton, Computer works a new trick in seeing pseudocolor processing, *Aeronaut. Astronaut.*, Vol. 4, 1967, p. 76.

R. C. Gonzales and B. A. Fittes, Gray-level transformation for interactive image enhancement, *Mech. Mach. Theory*, Vol. 12, 1977, pp. 111–122.

J. E. Hall and J. D. Awtrey, Real-time image enhancement using 3 × 3 pixel neighborhood operator functions, *Opt. Eng.*, Vol. 19, May/June 1980, pp. 421–424.

E. L. Hall, R. P. Kruger, S. J. Dwyer, III, D. L. Hall, R. W. McLaren, and G. S. Lodwick, A survey of preprocessing and feature extraction techniques for radiographic images, *IEEE Trans. Computer*, Vol. C-20, 1971, pp. 1032–1044.

T. S. Huang and R. Y. Tsai, *Image Sequence Analysis.* Berlin: Springer-Verlag, 1981, Chapter 1.

T. S. Huang, G. J. Yang, and G. Y. Tang, A fast two-dimensional median filtering algorithm, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-27, February 1979, pp. 13–18.

E. R. Kreins and L. J. Allison, Color enhancement of nimbus high resolution infrared radiometer data, *Appl. Opt.*, Vol. 9, 1970, p. 681.

J. S. Lim, ed., *Speech Enhancement.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

D. Marr, *Vision, A Computational Investigation into the Human Representation of Visual Information.* New York: W. H. Freeman and Company, 1982.

D. Marr and E. Hildreth, Theory of edge detection, *Proc. R. Soc.*, London, Vol. B207, 1980, pp. 187–217.

D. M. Martinez, Model-based motion estimation and its application to restoration and interpolation of motion pictures, Ph.D. Thesis, M.I.T., Dept. of Elec. Eng. Comp. Sci., 1986.

H. G. Musmann, P. Pirsch, and H. Grallert, Advances in picture coding, *Proc. IEEE*, Vol. 73, April 1985, pp. 523–548.

A. N. Netravali and J. D. Robbins, Motion-compensated coding: some new results, *The Bell System Tech. J.*, Vol. 59, November 1980, pp. 1735–1745.

T. A. Nodes and N. C. Gallagher, Jr., Two-dimensional root structures and convergence properties of the separable median filter, *IEEE Trans. Acoust., Speech and Sig. Proc.*, December 1983, pp. 1350–1365.

D. A. O'Handley and W. B. Green, Recent developments in digital image processing at the image processing laboratory at the Jet Propulsion Laboratory, *Proc. IEEE*, Vol. 60, July 1972, pp. 821–828.

A. V. Oppenheim, R. W. Schafer, and T. G. Stockham, Jr., Nonlinear filtering of multiplied and convolved signals, *Proc. IEEE*, Vol. 56, 1968, pp. 1264–1291.

T. N. Pappas and J. S. Lim, A new method for estimation of coronary artery dimensions in angiograms, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-36, September 1988, pp. 1501–1513.

R. Paquin and E. Dubois, A spatio-temporal gradient method for estimating the displacement field in time-varying imagery, *Computer Vision, Graphics, and Image Processing*, Vol. 21, 1983, pp. 205–221.

T. Peli and J. S. Lim, Adaptive filtering for image enhancement, *J. Opt. Eng.*, Vol. 21, January/February 1982, pp. 108–112.

T. Peli and D. Malah, A study of edge detection algorithms, *Computer Graphics and Image Processing*, Vol. 20, 1982, pp. 1–20.

L. G. Roberts, Machine perception of three-dimensional solids, in *Optical and Electro-Optical Information Processing*, J. T. Tippett et al., eds., Cambridge, MA: MIT Press, 1965, pp. 159–197.

W. F. Schreiber, Image processing for quality improvement, *Proc. IEEE*, Vol. 66, December 1978, pp. 1640–1651.

W. F. Schreiber, Wirephoto quality improvement by unsharp masking, *J. Pattern Recognition*, Vol. 2, 1970, pp. 117–121.

G. B. Shaw, Local and regional edge detectors: some comparisons, *Computer Graphics and Image Processing*, Vol. 9, 1979, pp. 135–149.

J. J. Sheppard, Jr., Pseudocolor as a means of image enhancement, *Am. J. Opthalmol. Arch. Am. Acad. Optom.*, Vol. 46, 1969, pp. 735–754.

R. Srinivasan and K. R. Rao, Predictive coding based on efficient motion estimation, *IEEE Trans. on Comm.*, Vol. COM-33, August 1985, pp. 888–896.

D. R. Sullivan, R. C. Harney, and J. S. Martin, Real-time quasi-three-dimensional display of infrared radar images, *SPIE*, Vol. 180, 1979, pp. 56–64.

E. G. Troy, E. S. Deutsch, and A. Rosenfeld, Gray-level manipulation experiments for texture analysis, *IEEE Trans. Sys. Man. Cybernet.*, Vol. SMC-3, 1973, pp. 91–98.

R. E. Woods and R. C. Gonzales, Real-time digital image enhancement, *Proc. IEEE*, Vol. 69, May 1981, pp. 643–654.

PROBLEMS

8.1. Let $f(n_1, n_2)$ denote an image of 256 × 256 pixels. The histogram of $f(n_1, n_2)$ is sketched below.

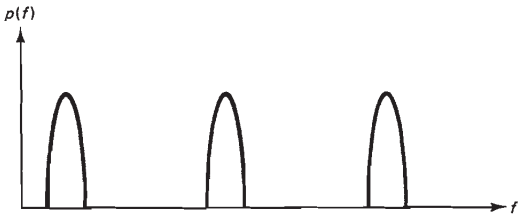


Figure P8.1

What can we say about $f(n_1, n_2)$? Approximately sketch a transformation function which is likely to improve the contrast of the image when it is used to modify the gray scale of the image.

8.2. Suppose we have an image of 8 × 8 pixels as shown below.

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	2	2	2	3	4
1	1	1	2	2	2	3	4
1	1	1	2	2	3	3	5
1	1	1	2	2	3	3	5
1	1	2	2	2	3	4	6
1	1	2	2	2	3	4	7

Figure P8.2

We wish to modify the gray scale of this image such that the histogram of the processed image is as close as possible to being constant in the range between 0 and 7. This is known as histogram equalization.

(a) Determine a transformation function that will achieve the above objective.

(b) Determine the processed image based on the transformation function you obtained in (a).

8.3. Modifying a histogram so that the output image has a histogram which has a maximum around the middle of the dynamic range and decreases slowly as the intensity increases or decreases does not always result in an output image more useful than the original unprocessed image. Discuss one such example.

8.4. In this problem, we consider an elementary probability problem closely related to the histogram modification problem. Let f denote a random variable with probability density function $p_f(f_0)$. We define g as $g = T[f]$. The function $T[\cdot]$ is a deterministic monotonically increasing function. The variable g is a random variable with probability density function $p_g(g_0)$.

(a) Let $p_f(f_0)$ be a uniform probability density function given by

$$p_f(f_0) = \begin{cases} 1, & 0 \le f_0 \le 1 \\ 0, & \text{otherwise.} \end{cases}$$

Suppose $T[f]$ is given by

$$g = T[f] = e^f.$$

Determine $p_g(g_0)$.

(b) More generally, develop a method to determine $p_g(g_0)$ given $p_f(f_0)$ and $T[\cdot]$.

(c) Let $p_f(f_0)$ be the same uniform probability density function as in (a). Suppose $p_g(g_0)$ is given by

$$p_g(g_0) = e^{-g_0}u(g_0)$$

where $u(g_0)$ is the unit step function. Determine $T[\cdot]$.

(d) More generally, develop a method to determine $T[\cdot]$, given $p_f(f_0)$ and $p_g(g_0)$.

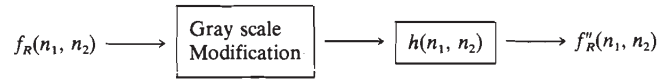
(e) Discuss how the solution to (d) can be used as a basis for developing a histogram modification method.

8.5. Consider a color image represented by $f_R(n_1, n_2)$, $f_G(n_1, n_2)$, and $f_B(n_1, n_2)$, the red, green, and blue components. We modify the gray scale of each of the three components using histogram modification.

(a) Suppose the desired histogram used in modifying the gray scale is the same for each of the three components. Does the modification affect the hue and saturation of the color image?

(b) Suppose we filter $f_R(n_1, n_2)$ with a system whose impulse response is $h(n_1, n_2)$ and then modify the gray scale of the filtered signal, as shown in the following figure.

Suppose we change the order of the filtering and gray scale modification, as shown below.



If the gray scale is modified by histogram modification by using the same desired histograms in both systems, are the results the same; that is, $f'_R(n_1, n_2) = f''_R(n_1, n_2)$?

8.6. Let V_{IN} denote the input voltage to a display monitor and I_{OUT} denote the output intensity actually displayed on the display monitor. Ideally, we wish to have I_{OUT} proportional to V_{IN}. In practice, however, the relationship between V_{IN} and I_{OUT} is nonlinear and is called *gamma*. Suppose the nonlinear relationship is given approximately by

$$I_{OUT} \propto V_{IN}^2$$

(a) What effect does the nonlinear relation have on a black-and-white image?

(b) What effect does the nonlinear relation have on a color image?

(c) One way to compensate for the gamma effect is to process V_{IN} prior to its input to the display monitor. Discuss a method of processing V_{IN} so that the gamma effect can be taken into account. In practice, such processing is not necessary, since the correction can be incorporated in the camera design.

8.7. Determine and sketch the frequency response of each of the filters in Figure 8.8. In sketching the frequency response $H(\omega_1, \omega_2)$, you need to sketch only $H(\omega_1, \omega_2)|_{\omega_1 = 0}$.

8.8. In homomorphic processing for contrast enhancement, an image $f(n_1, n_2)$ is logarithmically transformed, highpass filtered, and then exponentiated, as shown in the following figure.

We assume in this problem that the highpass filter used is a linear shift-invariant system.

(a) In practice, the scale of $f(n_1, n_2)$ is arbitrary. For example, $f(n_1, n_2)$ may be scaled such that 0 corresponds to the darkest level and 1 corresponds to the brightest. Alternatively, $f(n_1, n_2)$ may be scaled such that 0 corresponds to the darkest level and 255 corresponds to the brightest. What is the effect on the processed image $p(n_1, n_2)$ of the choice of scale for $f(n_1, n_2)$?

(b) The logarithmic operation has an undesirable behavior for the amplitude of $f(n_1, n_2)$ close to zero. One system proposed to eliminate this undesirable behavior is shown below.

In the figure, a is a positive real constant. What is a reasonable choice of the constant a in order for this to approximate the homomorphic system?

(c) In the system in (b), what effect does the choice of scale for $f(n_1, n_2)$ have on the processed image $p'(n_1, n_2)$?

(d) One sometimes-cited advantage of the homomorphic system is the property that the processed image $p(n_1, n_2)$ always has nonnegative amplitude. Does the system in (b) have this property?

8.9. The system in Figure 8.12 modifies the local contrast as a function of the local luminance mean and modifies the local luminance mean through some nonlinearity.

With proper choice of the parameters, the system can also be used as a linear shift-invariant highpass filter. How should the parameters in the system be chosen?

8.10. One limitation of highpass filtering for contrast enhancement is the accentuation of background noise.

(a) Discuss why the noise accentuation problem is typically much more severe in low local contrast regions, such as areas of uniform background, than in high local contrast regions, such as edges.

(b) Design an adaptive highpass filter that reduces the noise visibility problem discussed in (a).

8.11. In reducing the effect of cloud cover in images taken from an airplane, we can exploit the following two properties:

Property 1. Regions covered by cloud are typically brighter than regions not covered by cloud.

Property 2. The contrast in regions covered by cloud is typically lower than the contrast in regions not covered by cloud.

One reasonable approach to reducing the effect of cloud cover is the adaptive system (System 1) shown in Figure 8.12. An alternative approach that can be used to reduce the effect of cloud cover is the following adaptive system (System 2):

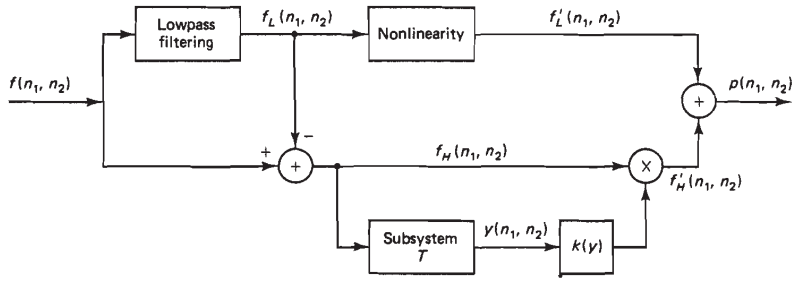


Figure P8.11

(a) Determine a reasonable Subsystem T, and sketch a reasonable $k(y)$ as a function of y and a reasonable f'_L as a function of f_L. Label the axes clearly.

(b) What is one major disadvantage of System 2 in its performance compared with System 1?

8.12. Determine and sketch the frequency response of each of the filters in Figure 8.14. In sketching the frequency response $H(\omega_1, \omega_2)$, you need to sketch only $H(\omega_1, \omega_2)|_{\omega_1 = 0}$.

8.13. For each of the following input sequences, determine the result of median filtering with window sizes of (i) 3 × 3 points and (ii) 5 × 5 points.

(a) $\delta(n_1, n_2)$

(b) $u(n_1, n_2)$

(c) $u_T(n_1)$

(d) $u(n_1, n_2) - u(n_1 - 2, n_2)$

8.14. Repeated application of a 1-D median filter to a 1-D sequence eventually leads to what is called a root signal, which is invariant under further applications of the 1-D median filter. Consider the 1-D sequence $x(n)$ shown below.

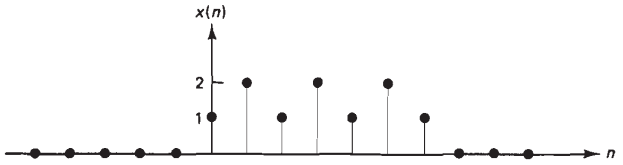


Figure P8.14

(a) Determine the root signal of $x(n)$, using a 3-point median filter.

(b) Is it possible to recover $x(n)$ from the root signal without additional information about $x(n)$?

(c) A median filter without additional specification refers to a nonrecursive median filter. It is possible to define a recursive median filter. Let $y(n)$ denote the output of a recursive 3-point median filter. The output $y(n)$ is given by

$$y(n) = \text{Median}\,[y(n-1), x(n), x(n+1)].$$

Determine the output $y(n)$ when a recursive 3-point median filter is applied to $x(n)$ above.

(d) Applying a 1-D recursive median filter to a 1-D sequence $x(n)$ is known to result in the root signal of $x(n)$ without repeated application of the filter. Are your results in (a) and (c) consistent with this?

8.15. A useful property of a median filter is its tendency to preserve step discontinuities. An $N \times N$-point median filter, however, distorts the unit step sequence $u(n_1, n_2)$. One method that preserves step discontinuities better than an $N \times N$-point median filter is to use a separable $N \times N$-point median filter. In this method, a 1-D N-point median filter is applied along the horizontal direction and then along the vertical direction. Another method is to use a star-shaped window for the median filter. The star-shaped 3 × 3-point window is shown below.

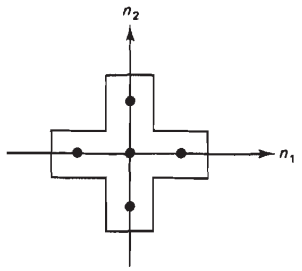


Figure P8.15

For this filter, the output $y(n_1, n_2)$ for an input $x(n_1, n_2)$ is determined by

$$y(n_1, n_2) = \text{Median}\,[x(n_1, n_2), x(n_1, n_2 + 1), x(n_1, n_2 - 1), x(n_1 - 1, n_2), x(n_1 + 1, n_2)].$$

(a) Does the star-shaped $N \times N$-point median filter distort $u(n_1, n_2)$?
(b) Which method preserves step discontinuities better, a separable $N \times N$-point median filter, or a star-shaped $N \times N$-point median filter? Consider steps with various orientations.
(c) Suppose the input is a uniform background degraded by wideband random noise. Which method reduces more background noise, a separable $N \times N$-point median filter or a star-shaped $N \times N$-point median filter?

8.16. In gradient-based edge detection algorithms, a gradient is approximated by a difference. One component of the gradient is $\partial f(x, y)/\partial x$. Suppose $\partial f(x, y)/\partial x$ is approximated by
(a) $f(n_1, n_2) - f(n_1 - 1, n_2)$
(b) $f(n_1 + 1, n_2) - f(n_1, n_2)$
(c) $f(n_1 + 1, n_2 + 1) - f(n_1 - 1, n_2 + 1) + 2[f(n_1 + 1, n_2) - f(n_1 - 1, n_2)] + f(n_1 + 1, n_2 - 1) - f(n_1 - 1, n_2 - 1)$
The differencing operation in each of the above three cases can be viewed as convolution of $f(n_1, n_2)$ with some impulse response of a filter $h(n_1, n_2)$. Determine $h(n_1, n_2)$ for each case.

8.17. Consider an image $f(n_1, n_2)$ of 7 × 7 pixels shown in Figure P8.17a. The numbers in the boxes represent the pixel intensities.

60	60	62	65	68	70	70
60	60	62	65	68	70	70
70	70	72	75	78	80	80
100	100	102	105	108	110	110
130	130	132	135	138	140	140
140	140	142	145	148	150	150
140	140	142	145	148	150	150

Figure P8.17a

	40.79	44.72	46.65	44.72	40.79	
	160.20	161.25	161.79	161.25	160.20	
	240.13	240.83	241.20	240.83	240.13	
	160.20	161.25	161.79	161.25	160.20	
	40.79	44.72	46.65	44.72	40.79	

Figure P8.17b

(a) Show that when the Sobel edge detector is used, the result of discrete approximation of $|\nabla f(x, y)|$ is given by Figure P8.17b. We will denote this result by $|\nabla f(n_1, n_2)|$. At pixel locations at the image boundary, we cannot compute $|\nabla f(n_1, n_2)|$ based on the Sobel edge detector.
(b) Suppose we choose a threshold of 100. Determine the candidate edge points.
(c) From your result in (b), edge thinning may be necessary to avoid wide strips of edges. Suppose we decide that any point among the candidate edge points is a true edge point if it is a local maximum of $|\nabla f(n_1, n_2)|$ in either the horizontal or the vertical direction. On the basis of this assumption, determine the edge points.
(d) By looking at $f(n_1, n_2)$ and using your own judgment, determine edge points of $f(n_1, n_2)$.
(e) Suppose we impose the following additional constraints in (c):
(i) If $|\nabla f(n_1, n_2)|$ has a local maximum at (n_1', n_2') in the horizontal direction but not in the vertical direction, (n_1', n_2') will be an edge point when $|\nabla_x f(n_1, n_2)|$ is significantly larger than $|\nabla_y f(n_1, n_2)|$. The functions $\nabla_x f(n_1, n_2)$ and $\nabla_y f(n_1, n_2)$ are the horizontal and vertical components, respectively, of $\nabla f(n_1, n_2)$.
(ii) The same constraint described in (i) with the horizontal direction x and vertical direction y interchanged.
Solve (c) with these two constraints imposed.
(f) Compare your results in (d) and (e).

8.18. One way to approximate the Laplacian $\nabla^2 f(x, y)$ in the discrete domain is to convolve $f(n_1, n_2)$ with $h(n_1, n_2)$, shown below.

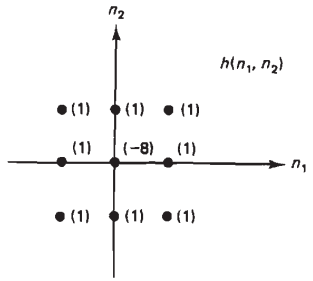


Figure P8.18

Show that the above approximation is reasonable.

8.19. In this problem, we illustrate that a reasonable generalization of $f''(x)$ to a 2-D function $f(x, y)$ for the purpose of edge detection is the Laplacian $\nabla^2 f(x, y)$. Consider an "ideal" 2-D edge, shown in Figure P8.19a. Given that the edge is in the direction perpendicular to the u-axis and that the point (0, 0) lies on the edge contour, as shown in the figure, it is possible to model the function $f(x, y)$ in the neighborhood of (0, 0) by

$$f(x, y) = g(u)|_{u = x\cos\theta + y\sin\theta}$$

where $g(u)$ is shown in Figure P8.19b. Note that $f(x, y)$ is constant in the direction of the edge line. In this case, the 2-D problem of locating the edge of $f(x, y)$ in the neighborhood of (0, 0) can be viewed as a 1-D problem of locating the edge in $g(u)$.
(a) For (x, y) on the u-axis, show that

$$|g'(u)|_{u = x\cos\theta + y\sin\theta} = |\nabla f(x, y)|.$$

This is consistent with the generalization of $|f'(x)|$ to $|\nabla f(x, y)|$ in gradient-based methods for edge detection.
(b) For (x, y) on the u-axis, show that

$$g''(u)|_{u = x\cos\theta + y\sin\theta} = \nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}.$$

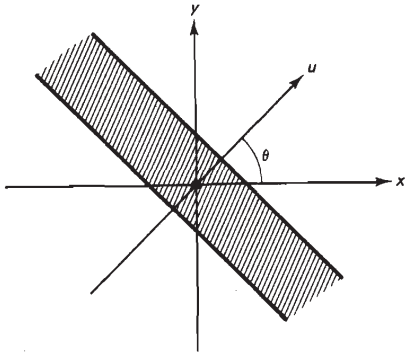


Figure P8.19a

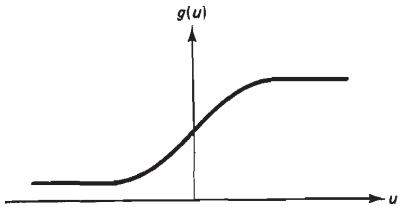


Figure P8.19b

This is consistent with the generalization of $f''(x)$ to $\nabla^2 f(x, y)$ in Laplacian-based methods for edge detection.

(c) The result in (b) is not valid for 2-D edges with sharp corners. Discuss how this could effect the performance of Laplacian-based edge detection methods. You may want to look at some edge maps obtained by Laplacian-based edge detection methods.

8.20. Let $f(x, y)$ denote a 2-D function that is sampled on a Cartesian grid. The samples of $f(x, y)$ are shown in the following figure.

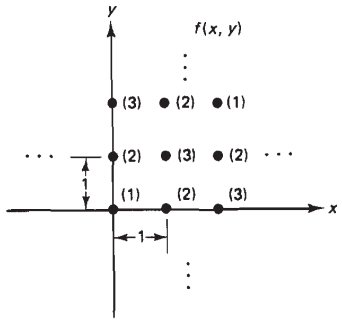


Figure P8.20

The values in parentheses in the figure represent the amplitudes of $f(x, y)$ evaluated at the spatial points corresponding to the filled-in dots. Suppose we wish to estimate $f(x, y)$ by using bilinear interpolation.

(a) Determine $f(x, y)$ for $0 \le x \le 2$, $0 \le y \le 1$.

(b) Are $\partial f(x, y)/\partial x$ and $\partial f(x, y)/\partial y$ well defined for all $0 \le x \le 2$ and $0 \le y \le 1$?

8.21. In image interpolation, $f(x, y)$ is approximated by

$$\hat{f}(x, y) = \sum_{i=1}^{N} S_i \phi_i(x, y)$$

where $\phi_i(x, y)$ is a set of predetermined functions. One method of estimating S_i from samples of $f(x, y)$ is by minimizing

$$\text{Error} = \sum_{(n1,n2)\,\epsilon\,\psi} \sum (f(x, y) - \sum_{i=1}^{N} S_i\, \phi_i(x, y))^2|_{x=n1T1,\,y=n2T2}$$

where ψ denotes the pixels over which $f(x, y)$ is approximated.

(a) Determine the set of linear equations for S_i that results from solving the above minimization problem.

(b) Suppose $N = 3$, $\phi_1(x, y) = 1$, $\phi_2(x, y) = x$, $\phi_3(x, y) = xy$, $f(0, 0) = 1$, $f(0, 1) = f(1, 0) = 2$, and $f(1, 1) = 3$. Determine $\hat{f}(x, y)$, the interpolated result obtained by solving the above minimization problem.

8.22. Let $f(x, y, t)$ denote an image intensity as a function of two spatial variables (x, y) and a time variable t. Suppose $f(x, y, t)$ is generated from one image frame by translational motion with uniform velocities of v_x and v_y along the horizontal and vertical direction, respectively. Suppose $f(x, y, 0)$ is given by

$$f(x, y, 0) = x + y + 3xy$$

(a) Determine $f(x, y, t)$.

(b) For the $f(x, y, t)$ obtained in (a), show that

$$v_x \frac{\partial f(x, y, t)}{\partial x} + v_y \frac{\partial f(x, y, t)}{\partial y} + \frac{\partial f(x, y, t)}{\partial t} = 0.$$

8.23. The motion estimation methods we discussed in Section 8.4.2 are based on the assumption of translational motion given by (8.29):

$$f(x, y, t_0) = f(x - d_x, y - d_y, t_{-1})$$

If the overall illumination varies over time, a better model is given by

$$f(x, y, t_0) = \alpha f(x - d_x, y - d_y, t_{-1})$$

where α is some unknown constant. Discuss how we would develop region matching methods based on this new signal model.

8.24. In motion estimation methods based on the spatio-temporal constraint equation, we solve two linear equations for the velocity vector (v_x, v_y). The two equations can be expressed as

$$W\mathbf{v} = \boldsymbol{\gamma}$$

where $\mathbf{v} = [v_x, v_y]^T$ and where W and $\boldsymbol{\gamma}$ are given by (8.43). Let λ_1 and λ_2 denote the eigenvalues of W, and let $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ denote the corresponding orthonormal eigenvectors.

(a) Suppose $f(x, y, t)$ is constant. Show that $\lambda_1 = \lambda_2 = 0$, and that a reasonable estimate of $\mathbf{v}$ is $\mathbf{0}$.

(b) Consider $f(x, y, t)$ which does not depend on x and which results from uniform translational motion. Let $\lambda_1 \ge \lambda_2$. Show that $\lambda_2 = 0$, and that a reasonable estimate of $\mathbf{v}$ is $\mathbf{v} = [\boldsymbol{\alpha}_1^T \boldsymbol{\gamma}]\boldsymbol{\alpha}_1/\lambda_1$.