Solutions IAT-1(IP-10EC763)

1

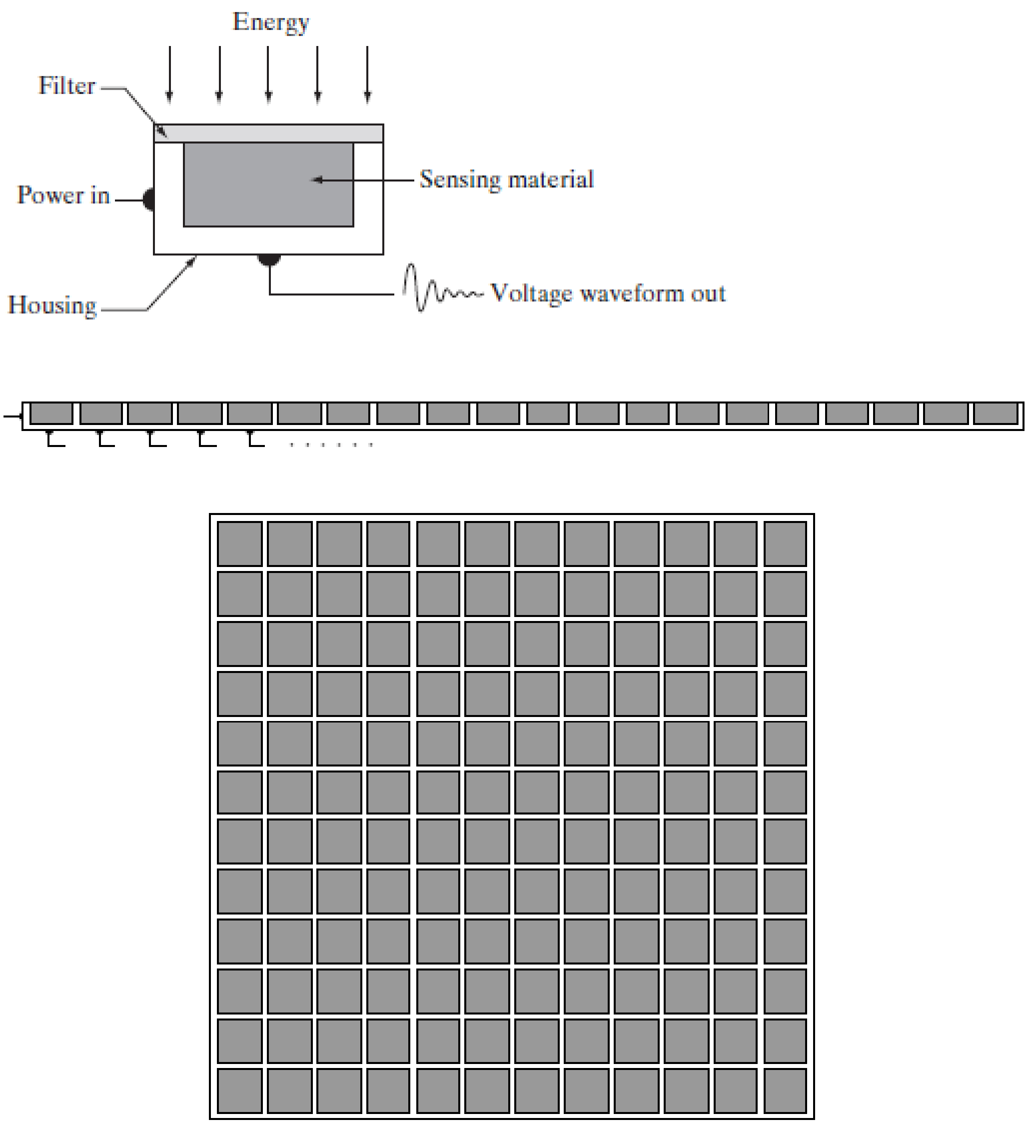


Figure 2.12 shows the three principal sensor arrangements used to transform illumination energy into digital images. The idea is simple: Incoming energy is

transformed into a voltage by the combination of input electrical power and sensor material that is responsive to the particular type of energy being detected. The output voltage waveform is the response of the sensor(s), and a digital quantity is obtained from each sensor by digitizing its response.

2.3.3 Image Acquisition Using Sensor Arrays

Figure 2.12(c) shows individual sensors arranged in the form of a 2-D array. Numerous electromagnetic and some ultrasonic sensing devices frequently are arranged in an array format. This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged arrays of 4000×4000 elements or more. CCD sensors are used widely in digital cameras and other light sensing instruments. The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. Noise reduction is achieved by letting the sensor integrate the input light signal over minutes or even hours (we discuss noise reduction by integration in Chapter 3). Since the sensor array shown in Fig. 2.15(c) is two dimensional, its key advantage is that a complete image can be obtained by focusing the energy pattern onto the surface of the array. Motion obviously is not necessary, as is the case with the sensor arrangements discussed in the preceding two sections.

The principal manner in which array sensors are used is shown in Fig. 2.15. This figure shows the energy from an illumination source being reflected from a scene element, but, as mentioned at the beginning of this section, the energy also could be transmitted through the scene elements. The first function performed by the imaging system shown in Fig. 2.15(c) is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is a lens, which projects the viewed scene onto the lens focal plane, as Fig. 2.15(d) shows. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and convert them to a video signal, which is then digitized by another section of the imaging system. The output is a digital image, as shown diagrammatically in Fig. 2.15(e). Conversion of an image into digital form is the topic of Section 2.4.

2.3.1 Image Acquisition Using a Single Sensor

Figure 2.12(a) shows the components of a single sensor. Perhaps the most familiar sensor of this type is the photodiode, which is constructed of silicon materials and whose output voltage waveform is proportional to light. The use of a filter in front of a sensor improves selectivity. For example, a green (pass) filter in front of a light sensor favors light in the green band of the color spectrum. As a consequence, the sensor output will be stronger for green light than for other components in the visible spectrum.

In order to generate a 2-D image using a single sensor, there has to be relative displacements in both the x- and y-directions between the sensor and the area to be imaged. Figure 2.13 shows an arrangement used in high-precision scanning, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The single sensor is mounted on a lead screw that provides motion in the perpendicular direction. Since mechanical motion can be controlled with high precision, this method is an inexpensive (but slow) way to obtain high-resolution images. Other similar mechanical arrangements use a flat bed, with the sensor moving in two linear directions. These types of mechanical digitizers sometimes are referred to as *microdensitometers*.

Another example of imaging with a single sensor places a laser source coincident with the sensor. Moving mirrors are used to control the outgoing beam in a scanning pattern and to direct the reflected laser signal onto the sensor. This arrangement also can be used to acquire images using strip and array sensors, which are discussed in the following two sections.

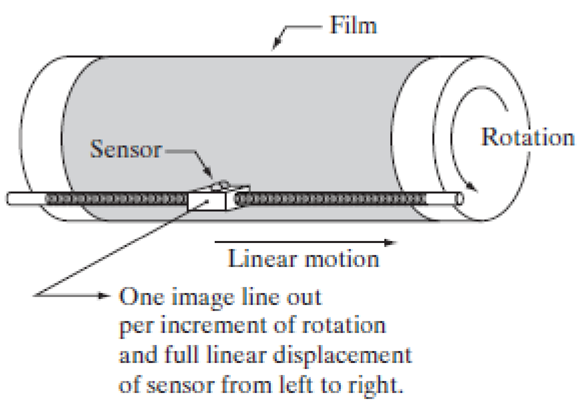


FIGURE 2.13 Combining a single sensor with motion to generate a 2-D image.

2A	Let V be the set of gray-level values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a gray-scale image, the idea is the same, but set V typically contains more elements. For example, in the adjacency of pixels with a range of possible gray-level values 0 to 255, set V could be any subset of these 256 values. We consider three types of adjacency: **(a)** 4-*adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$. **(b)** 8-*adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$. **(c)** *m-adjacency* (mixed adjacency). Two pixels p and q with values from V are m-adjacent if **(i)** q is in $N_4(p)$, *or* **(ii)** q is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

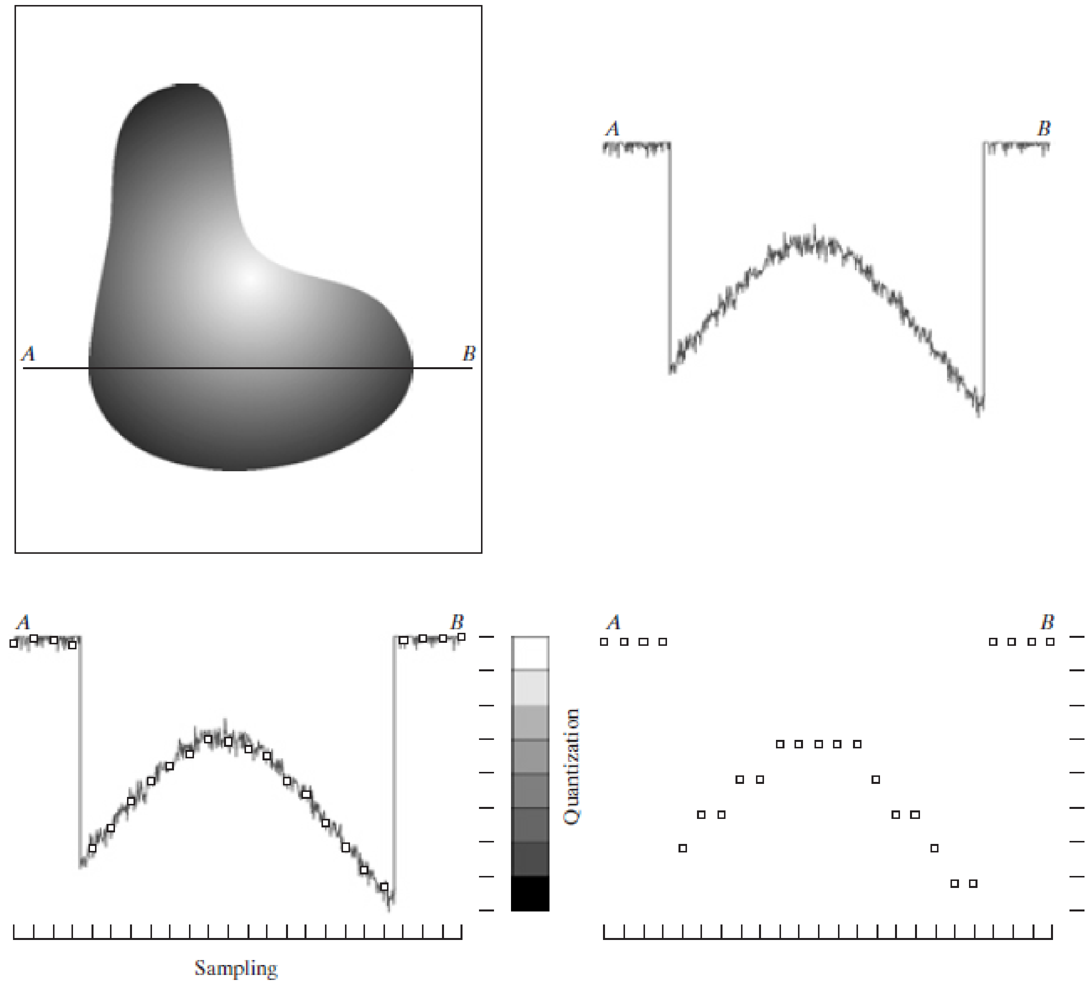
4

2.4.1 Basic Concepts in Sampling and Quantization

The basic idea behind sampling and quantization is illustrated in Fig. 2.16. Figure 2.16(a) shows a continuous image, $f(x, y)$, that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

The one-dimensional function shown in Fig. 2.16(b) is a plot of amplitude (gray level) values of the continuous image along the line segment AB in Fig. 2.16(a). The random variations are due to image noise. To sample this function, we take equally spaced samples along line AB, as shown in Fig. 2.16(c). The location of each sample is given by a vertical tick mark in the bottom part of the figure. The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities. The right side of Fig. 2.16(c) shows the gray-level scale divided into eight discrete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment is made depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization are shown in Fig. 2.16(d). Starting at the top of the image and carrying out this procedure line by line produces a two-dimensional digital image.

Sampling in the manner just described assumes that we have a continuous image in both coordinate directions as well as in amplitude. In practice, the method of sampling is determined by the sensor arrangement used to generate the image. When an image is generated by a single sensing element combined with mechanical motion, as in Fig. 2.13, the output of the sensor is quantized in the manner described above. However, sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data. Mechanical motion can be made very exact so, in principle, there is almost no limit as to how fine we can sample an image. However, practical limits are established by imperfections in the optics used to focus on the



a b
c d

FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

sensor an illumination spot that is inconsistent with the fine resolution achievable with mechanical displacements.

When a sensing strip is used for image acquisition, the number of sensors in the strip establishes the sampling limitations in one image direction. Mechanical motion in the other direction can be controlled more accurately, but it makes little sense to try to achieve sampling density in one direction that exceeds the

	sampling limits established by the number of sensors in the other. Quantization of the sensor outputs completes the process of generating a digital image. When a sensing array is used for image acquisition, there is no motion and the number of sensors in the array establishes the limits of sampling in both directions. Quantization of the sensor outputs is as before. Figure 2.17 illustrates this concept. Figure 2.17(a) shows a continuous image projected onto the plane of an array sensor. Figure 2.17(b) shows the image after sampling and quantization. Clearly, the quality of a digital image is determined to a large degree by the number of samples and discrete gray levels used in sampling and quantization. However, as shown in Section 2.4.3, image content is an important consideration in choosing these parameters.
5	histogram equalization automatically determines a transformation function that seeks to produce an output image that has a uniform histogram. When automatic enhancement is desired, this is a good approach because the results from this technique are predictable and the method is simple to implement. We show in this section that there are applications in which attempting to base enhancement on a uniform histogram is not the best approach. In particular, it is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have. The method used to generate a processed image that has a specified histogram is called *histogram matching* or *histogram specification*. **Development of the method** Let us return for a moment to continuous gray levels r and z (considered continuous random variables), and let $p_r(r)$ and $p_z(z)$ denote their corresponding continuos probability density functions. In this notation, r and z denote

the gray levels of the input and output (processed) images, respectively. We can estimate $p_r(r)$ from the given input image, while $p_z(z)$ is the *specified* probability density function that we wish the output image to have.

Let s be a random variable with the property

$$s = T(r) = \int_0^r p_r(w)\, dw \qquad (3.3\text{-}10)$$

where w is a dummy variable of integration. We recognize this expression as the continuos version of histogram equalization given in Eq. (3.3-4). Suppose next that we define a random variable z with the property

$$G(z) = \int_0^z p_z(t)\, dt = s \qquad (3.3\text{-}11)$$

where t is a dummy variable of integration. It then follows from these two equations that $G(z) = T(r)$ and, therefore, that z must satisfy the condition

$$z = G^{-1}(s) = G^{-1}\left[T(r)\right]. \qquad (3.3\text{-}12)$$

The transformation $T(r)$ can be obtained from Eq. (3.3-10) once $p_r(r)$ has been estimated from the input image. Similarly, the transformation function $G(z)$ can be obtained using Eq. (3.3-11) because $p_z(z)$ is given.

Assuming that G^{-1} exists and that it satisfies conditions (a) and (b) in the previous section, Eqs. (3.3-10) through (3.3-12) show that an image with a specified probability density function can be obtained from an input image by using the following procedure: (1) Obtain the transformation function $T(r)$ using Eq. (3.3-10). (2) Use Eq. (3.3-11) to obtain the transformation function $G(z)$. (3) Obtain the inverse transformation function G^{-1}. (4) Obtain the output image by applying Eq. (3.3-12) to all the pixels in the input image. The result of this procedure will be an image whose gray levels, z, have the specified probability density function $p_z(z)$.

Although the procedure just described is straightforward in principle, it is seldom possible in practice to obtain analytical expressions for $T(r)$ and for G^{-1}. Fortunately, this problem is simplified considerably in the case of discrete values. The price we pay is the same as in histogram equalization, where only an approximation to the desired histogram is achievable. In spite of this, however, some very useful results can be obtained even with crude approximations.

The discrete formulation of Eq. (3.3-10) is given by Eq. (3.3-8), which we repeat here for convenience:

$$\begin{aligned} s_k = T(r_k) &= \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \dots, L-1 \end{aligned} \tag{3.3-13}$$

where n is the total number of pixels in the image, n_j is the number of pixels with gray level r_j, and L is the number of discrete gray levels. Similarly, the discrete formulation of Eq. (3.3-11) is obtained from the given histogram $p_z(z_i)$, $i = 0, 1, 2, \dots, L-1$, and has the form

$$v_k = G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k \qquad k = 0, 1, 2, \dots, L-1. \tag{3.3-14}$$

As in the continuos case, we are seeking values of z that satisfy this equation. The variable v_k was added here for clarity in the discussion that follows. Finally, the discrete version of Eq. (3.3-12) is given by

$$z_k = G^{-1}[T(r_k)] \qquad k = 0, 1, 2, \dots, L-1 \tag{3.3-15}$$

or, from Eq. (3.3-13),

$$z_k = G^{-1}(s_k) \qquad k = 0, 1, 2, \dots, L-1. \tag{3.3-16}$$

Equations (3.3-13) through (3.3-16) are the foundation for implementing histogram matching for digital images. Equation (3.3-13) is a mapping from the levels in the original image into corresponding levels s_k based on the histogram of the original image, which we compute from the pixels in the image. Equation (3.3-14) computes a transformation function G from the given histogram $p_z(z)$. Finally, Eq. (3.3-15) or its equivalent, Eq. (3.3-16), gives us (an approximation of) the desired levels of the image with that histogram. The first two equations can be implemented easily because all the quantities are known. Implementation of Eq. (3.3-16) is straightforward, but requires additional explanation.

Implementation

We start by noting the following: (1) Each set of gray levels $\{r_j\}$, $\{s_j\}$, and $\{z_j\}$, $j = 0, 1, 2, \dots, L-1$, is a one-dimensional array of dimension $L \times 1$. (2) All mappings from r to s and from s to z are simple table lookups between a given pixel value and these arrays. (3) Each of the elements of these arrays, for example, s_k, contains two important pieces of information: The subscript k denotes the location of the element in the array, and s denotes the value at that location. (4) We need to be concerned only with integer pixel values. For example, in the case of an 8-bit image, $L = 256$ and the elements of each of the arrays just mentioned are integers between 0 and 255. This implies that we now work with gray level values in the interval $[0, L - 1]$ instead of the normalized interval $[0, 1]$ that we used before to simplify the development of histogram processing techniques.

In order to see how histogram matching actually can be implemented, consider Fig. 3.19(a), ignoring for a moment the connection shown between this figure and Fig. 3.19(c). Figure 3.19(a) shows a hypothetical discrete transformation function $s = T(r)$ obtained from a given image. The first gray level in the image, r_1, maps to s_1; the second gray level, r_2, maps to s_2; the kth level r_k maps to s_k; and so on (the important point here is the *ordered* correspondence between these values). Each value s_j in the array is precomputed using Eq. (3.3-13), so the process of mapping simply uses the actual value of a pixel as an index in an array to determine the corresponding value of s. This process is particularly easy because we are dealing with integers. For example, the s mapping for an 8-bit pixel with value 127 would be found in the 128th position in array $\{s_j\}$ (recall that we start at 0) out of the possible 256 positions. If we stopped here and mapped the value of each pixel of an input image by the

a b
c

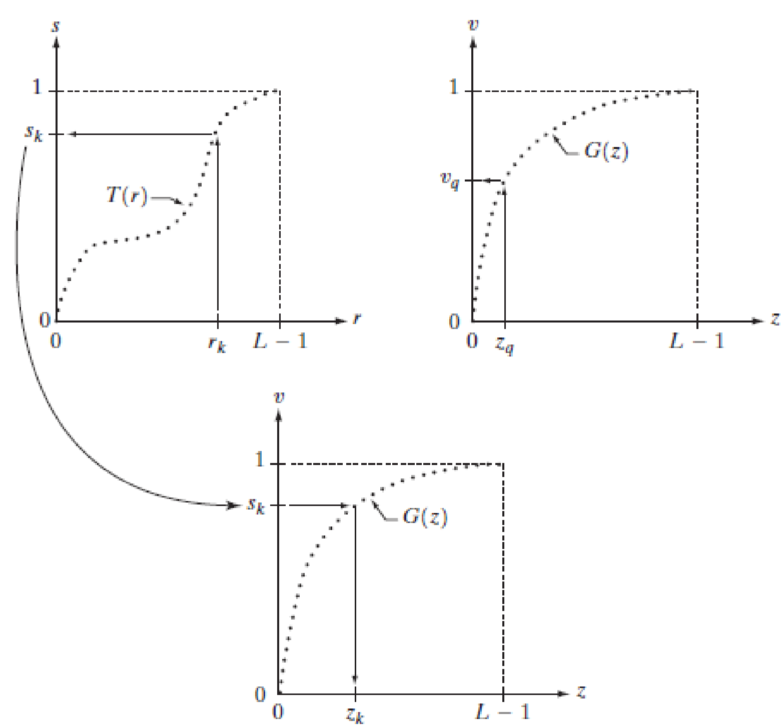


FIGURE 3.19 (a) Graphical interpretation of mapping from r_k to s_k via $T(r)$. (b) Mapping of z_q to its corresponding value v_q via $G(z)$. (c) Inverse mapping from s_k to its corresponding value of z_k.

method just described, the output would be a histogram-equalized image, according to Eq. (3.3-8).

In order to implement histogram matching we have to go one step further. Figure 3.19(b) is a hypothetical transformation function G obtained from a given histogram $p_z(z)$ by using Eq. (3.3-14). For any z_q, this transformation function yields a corresponding value v_q. This mapping is shown by the arrows in Fig. 3.19(b). Conversely, given any value v_q, we would find the corresponding value z_q from G^{-1}. In terms of the figure, all this means graphically is that we would reverse the direction of the arrows to map v_q into its corresponding z_q. However, we know from the definition in Eq. (3.3-14) that $v = s$ for corresponding subscripts, so we can use exactly this process to find the z_k corresponding to any value s_k that we computed previously from the equation $s_k = T(r_k)$. This idea is shown in Fig. 3.19(c).

Since we really do not have the z's (recall that finding these values is precisely the objective of histogram matching), we must resort to some sort of iterative scheme to find z from s. The fact that we are dealing with integers makes this a particularly simple process. Basically, because $v_k = s_k$, we have from Eq. (3.3-14) that the z's for which we are looking must satisfy the equation $G(z_k) = s_k$, or $(G(z_k) - s_k) = 0$. Thus, all we have to do to find the value of z_k corresponding to s_k is to iterate on values of z such that this equation is satisfied for $k = 0, 1, 2, \ldots, L - 1$. This is the same thing as Eq. (3.3-16), except that we do not have to find the inverse of G because we are going to iterate on z. Since we are dealing with integers, the closest we can get to satisfying the equation $(G(z_k) - s_k) = 0$ is to let $z_k = \hat{z}$ for each value of k, where $\hat{z}$ is the *smallest* integer in the interval $[0, L - 1]$ such that

$$\left(G(\hat{z}) - s_k\right) \geq 0 \qquad k = 0, 1, 2, \ldots, L - 1. \tag{3.3-17}$$

Given a value s_k, all this means conceptually in terms of Fig. 3.19(c) is that we would start with $\hat{z} = 0$ and increase it in integer steps until Eq. (3.3-17) is satisfied, at which point we let $z_k = \hat{z}$. Repeating this process for all values of k would yield all the required mappings from s to z, which constitutes the implementation of Eq. (3.3-16). In practice, we would not have to start with $\hat{z} = 0$ each time because the values of s_k are known to increase monotonically. Thus, for $k = k + 1$, we would start with $\hat{z} = z_k$ and increment in integer values from there.

6	**Bit-plane slicing** Instead of highlighting gray-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit-plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits. Figure 3.12 illustrates these ideas, and Fig. 3.14 shows the various bit planes for the image shown in Fig. 3.13. Note that the higher-order bits (especially the top four) contain the majority of the visually significant data. The other bit planes contribute to more subtle details in the image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel. . In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to one level (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255). The binary image for bit-plane 7 in Fig. 3.14 was obtained in just this manner. It is left as an exercise (Problem 3.3) to obtain the gray-level transformation functions that would yield the other bit planes. **Gray-level slicing** Highlighting a specific range of gray levels in an image often is desired. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images. There are several ways of doing level slicing, but most of them are variations of two basic themes. One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels. This transformation, shown in Fig. 3.11(a), produces a binary image. The second approach, based on the transformation shown in Fig. 3.11(b), brightens the desired range of gray levels but preserves the background and gray-level tonalities in the image. Figure 3.11(c) shows a gray-scale image, and Fig. 3.11(d) shows the result of using the transformation in Fig. 3.11(a). Variations of the two transformations shown in Fig. 3.11 are easy to formulate.

3.2.3 Power-Law Transformations

Power-law transformations have the basic form

$$s = cr^{\gamma} \tag{3.2-3}$$

where c and γ are positive constants. Sometimes Eq. (3.2-3) is written as $s = c(r + \varepsilon)^{\gamma}$ to account for an offset (that is, a measurable output when the input is zero). However, offsets typically are an issue of display calibration and as a result they are normally ignored in Eq. (3.2-3). Plots of s versus r for various values of γ are shown in Fig. 3.6. As in the case of the log transformation, power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ. As expected, we see in Fig. 3.6 that curves generated with values of $\gamma > 1$ have exactly the opposite effect as those generated with values of $\gamma < 1$. Finally, we note that Eq. (3.2-3) reduces to the identity transformation when $c = \gamma = 1$.

A variety of devices used for image capture, printing, and display respond according to a power law. By convention, the exponent in the power-law equation is referred to as *gamma* [hence our use of this symbol in Eq. (3.2-3)]. The process used to correct this power-law response phenomena is called *gamma correction*. For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5. With reference to the curve for $\gamma = 2.5$ in Fig. 3.6, we see that such display systems would tend to produce images that are darker than intended. This effect is illustrated in Fig. 3.7. Figure 3.7(a) shows a simple gray-scale linear wedge input into a CRT monitor. As expected, the output of the monitor appears darker than the input, as shown in Fig. 3.7(b). Gamma correction in this case is straightforward. All we need to do is preprocess the input image before inputting it into the monitor by performing the transformation $s = r^{1/2.5} = r^{0.4}$. The result is shown in Fig. 3.7(c). When input into the same monitor, this gamma-corrected input produces an output that is close in appearance to the original image, as shown in Fig. 3.7(d). A similar analysis would apply to other imaging devices such as scanners and printers. The only difference would be the device-dependent value of gamma (Poynton [1996]).

Gamma correction is important if displaying an image accurately on a computer screen is of concern. Images that are not corrected properly can look either bleached out, or, what is more likely, too dark. Trying to reproduce colors accurately also requires some knowledge of gamma correction because varying the value of gamma correction changes not only the brightness, but also the ratios of red to green to blue. Gamma correction has become increasingly important in the past few years, as use of digital images for commercial purposes over the Internet has increased. It is not unusual that images created for a popular Web site will be viewed by millions of people, the majority of whom will have different monitors and/or monitor settings. Some computer systems even have partial gamma correction built in. Also, current image standards do not contain the value of gamma with which an image was created, thus complicating the issue further. Given these constraints, a reasonable approach when storing images in a Web site is to preprocess the images with a gamma that represents an "average" of the types of monitors and computer systems that one expects in the open market at any given point in time.

7