UNIT 2

Intensity Transformations and Spatial Filtering: Background, Some Basic Intensity Transformation Functions, Histogram Processing, Fundamentals of Spatial Filtering, Smoothing (Lowpass) Spatial Filters, Sharpening (High pass) Spatial Filters.

Filtering in the Frequency Domain: Background, Preliminary Concepts, Sampling and the Fourier Transform of Sampled Functions, The Basics of Filtering in the Frequency Domain, Image Smoothing Using Lowpass Frequency Domain Filters, Image Sharpening Using High pass Filters.

CHAPTER 1

2.1 Background

- The term **spatial domain** refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image.
- Two principal categories of spatial processing are: intensity transformation and spatial filtering.
- **Intensity transformations** operate on single pixels of an image for tasks such as contrast manipulation and image thresholding.
- **Spatial filtering** performs operations on the neighborhood of every pixel in an image. Examples of spatial filtering include image smoothing and sharpening.
- **Spatial domain** techniques operate directly on the pixels of an image, as opposed, to the **frequency domain** in which operations are performed on the Fourier transform of an image, rather than on the image itself.

2.2 Some Basic Intensity Transformation Functions

- The spatial domain processes we discuss in this chapter are based on the expression:

$$g(x, y) = T[f(x, y)]$$

where $f(x, y)$ is an input image, $g(x, y)$ is the output image, and T is an operator on f defined over a neighborhood of point (x, y).

- The operator can be applied to the pixels of a single image or to the pixels of a set of images, such as performing the elementwise sum of a sequence of images for noise reduction.
- The point (x_0, y_0) shown is an arbitrary location in the image, and the small region shown is a neighborhood of (x_0, y_0). Typically, the neighborhood is rectangular, centered on (x_0, y_0), and much smaller in size than the image.
- The process that Figure 1 illustrates consists of moving the center of the neighborhood from pixel to pixel and applying the operator T to the pixels in the

neighborhood to yield an output value at that location. Thus, for any specific location (x_0, y_0) the value of the output image g at those coordinates is equal to the result of applying T to the neighborhood with origin at (x_0,y_0) in f.

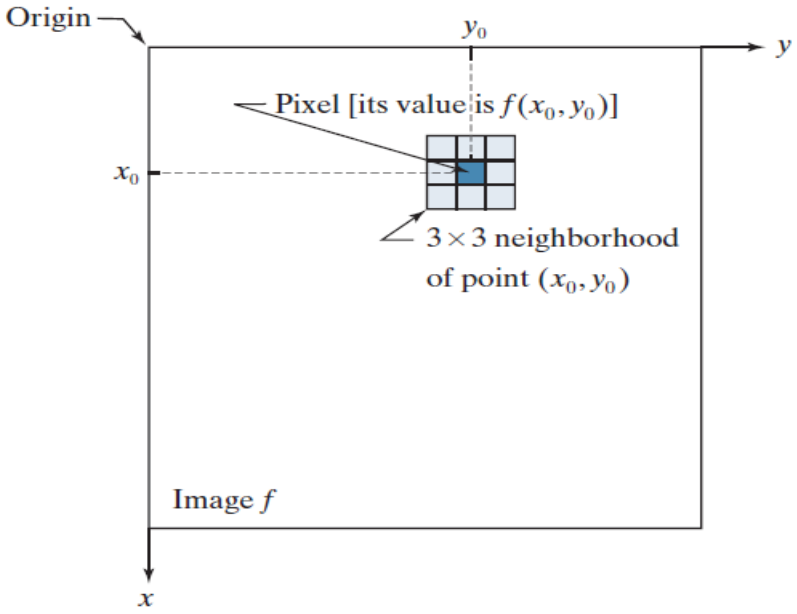

Figure 1: A 3 × 3 neighborhood about a point (x_0,y_0) in an image.. The neighborhood is moved from pixel to pixel in the image to generate an output image.

- The smallest possible neighborhood is of size 1 x 1.
- In this case, g depends only on the value of f at a single point (x, y) and T in above Equation becomes an *intensity* (also called a *gray-level*, or *mapping*) *transformation function* of the form:

$$s = T(r)$$

where, for simplicity in notation, we use s and r to denote, respectively, the intensity of g and f at any point (x, y).

- For example, $T(r)$ has the form in figure 2(a), the result of applying the transformation to every pixel in f to generate the corresponding pixels in g would be to produce an image of higher contrast than the original, by darkening the intensity levels below k and brightening the levels above k. In this technique, sometimes called ***contrast stretching*** values of r lower than k reduce (darken) the values of s, toward black. The opposite is true for values of r higher than k.
- An intensity value r_0 is mapped to obtain the corresponding value s_0.
- $T(r)$ produces a two level (binary) image. A mapping of this form is called a ***thresholding function*** as shown in figure 2(b).

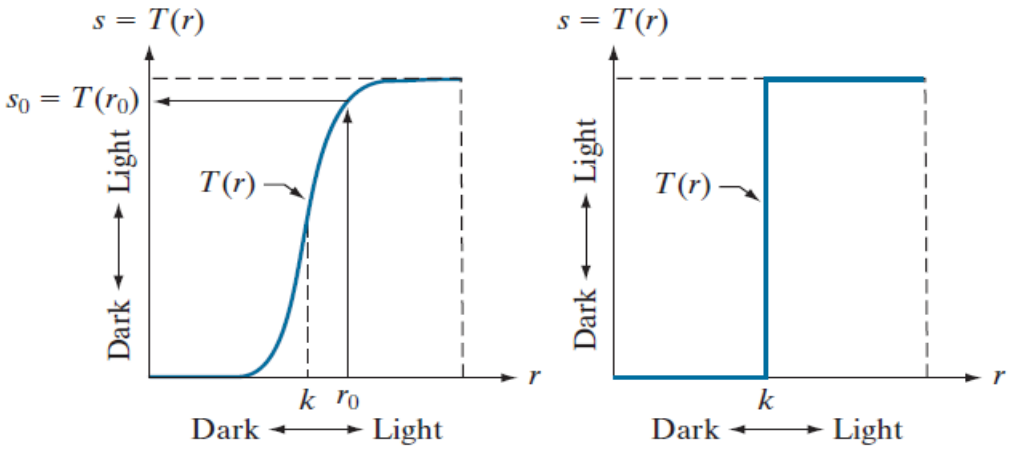

Figure 2: Intensity transformation functions. (a) Contrast stretching function. (b) Thresholding function.

- Enhancement is the process of manipulating an image so that the result is more suitable than the original for a specific application.

2.2 Some Basic Intensity Transformation Functions

- Intensity transformations are among the simplest of all image processing techniques.
- We denote the values of pixels, before and after processing, by r and s, respectively.
- These values are related by a transformation T, as given in above Equation, that maps a pixel value r into a pixel value s.
- Because we deal with digital quantities, values of an intensity transformation function typically are stored in a table, and the mappings from r to s are implemented via *table lookups.*
- Four basic types of functions used frequently in image processing as shown in figure 3:
  1. Identity function
  2. Linear (negative and identity transformations),
  3. Logarithmic (log and inverse-log transformations)
  4. Power-law (nth power and nth root transformations).
- The identity function is the trivial case in which the input and output intensities are identical.

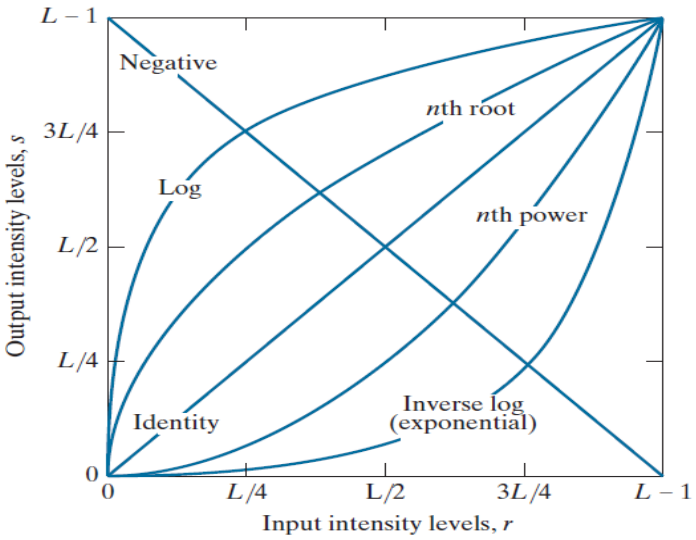

Figure 3: Some basic intensity transformation functions. Each curve was scaled *independently* so that all curves would fit in the same graph.

Image Negatives

- The negative of an image with intensity levels in the range [0,L - 1] is obtained by using the negative transformation function shown in figure 3, which has the form:

$$s = L - 1 - r$$

- Reversing the intensity levels of a digital image in this manner produces the equivalent of a photographic negative.
- This type of processing is used, for example, in enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size.

Log Transformations

- The general form of the log transformation in Figure 3 is:

$$s = c\log(1 + r)$$

where c is a constant and it is assumed that $r \geq 0$.

- The shape of the log curve in Figure 3 shows that this transformation maps a narrow range of low intensity values in the input into a wider range of output levels.
- For example, note how input levels in the range [0, L /4] map to output levels to the range [0, 3L/ 4].
- Conversely, higher values of input levels are mapped to a narrower range in the output.
- This type of transformation is used to expand the values of dark pixels in an image, while compressing the higher-level values.

- The opposite is true of the inverse log (exponential) transformation.
- Any curve having the general shape of the log function shown in Figure 3 would accomplish this spreading/compressing of intensity levels in an image.

Power-Law (Gamma) Transformation

- Power-law transformations have the form:

$$s = cr^{\gamma}$$

where c and γ are positive constants.

- Sometimes the above equation is written as $s = c\,(r + \epsilon)^{\gamma}$ to account for offsets (that is, a measurable output when the input is zero).
- Figure 4 shows plots of s as a function of r for various values of γ.

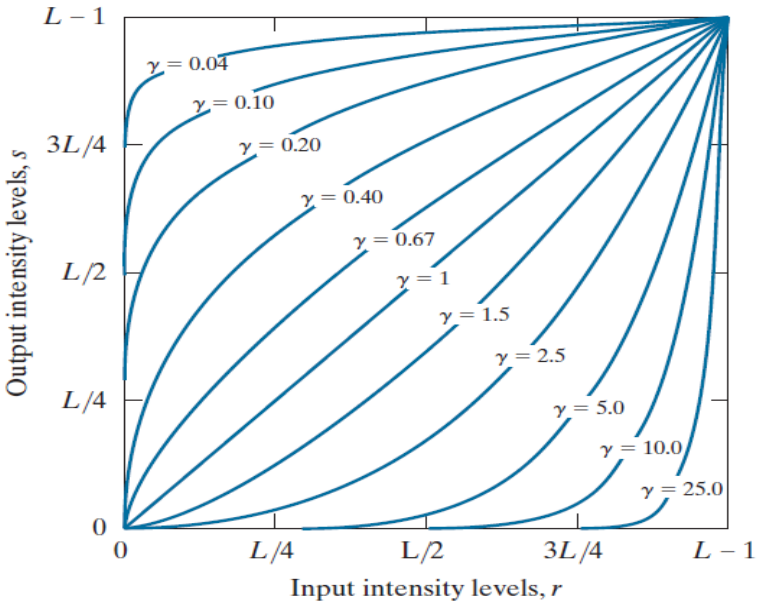

Figure 4: Plots of the gamma equation $s = c\,r^{\gamma}$ for various values of γ (c = 1 in all cases). Each curve was scaled *independently* so that all curves would fit in the same graph.

- As with log transformations, 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.
- Note also in Figure 4 that a family of transformations can be obtained simply by varying γ.
- Curves generated with values of γ > 1 have exactly the opposite effect as those generated with values of γ < 1.
- When c = γ = 1 in above Equation reduces to the identity transformation.
- The response of many devices used for image capture, printing, and display obey a power law.

- The process used to correct these power-law response phenomena is called ***gamma correction*** or ***gamma encoding.***
- As the curve for γ = 2.5 in Figure 4 shows, such display systems would tend to produce images that are darker than intended.

Piecewise Linear Transformation Functions

Contrast Stretching

- Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition.
- Contrast stretching expands the range of intensity levels in an image so that it spans the ideal full intensity range of the recording medium or display device.
- Below figure 5 shows the typical transformation used for contrast stretching.

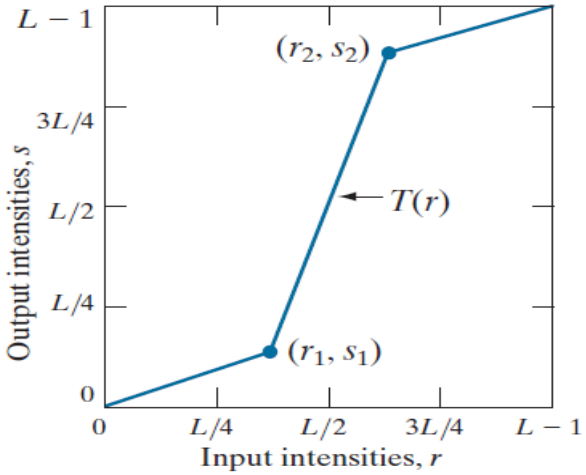

Figure 5: Contrast Stretching

- The locations of points (r_1, s_1) and (r_2, s_2) control the shape of the transformation function.
- If $r_1 = s_1$ and $r_2 = s_2$ the transformation is a linear function that produces no changes in intensity.
- If $r_1 = r_2$, $s_1 = 0$, and $s_2 = L-1$ the transformation becomes a *thresholding function* that creates a binary image.
- Intermediate values of (r_1, s_1) and (s_2, r_2) produce various degrees of spread in the intensity levels of the output image, thus affecting its contrast.
- In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the function is single valued and monotonically increasing.

Intensity Level Slicing

- There are applications in which it is of interest to highlight a specific range of intensities in an image.
- The method, called intensity-level slicing, can be implemented in several ways, but most are variations of two basic themes.

- One approach is to display in one value (say, white) all the values in the range of interest and in another (say, black) all other intensities. This transformation produces a binary image which is shown in Figure 6(a).
- The second approach, based on the transformation in figure 6(b) brightens (or darkens) the desired range of intensities, but leaves all other intensity levels in the image unchanged.

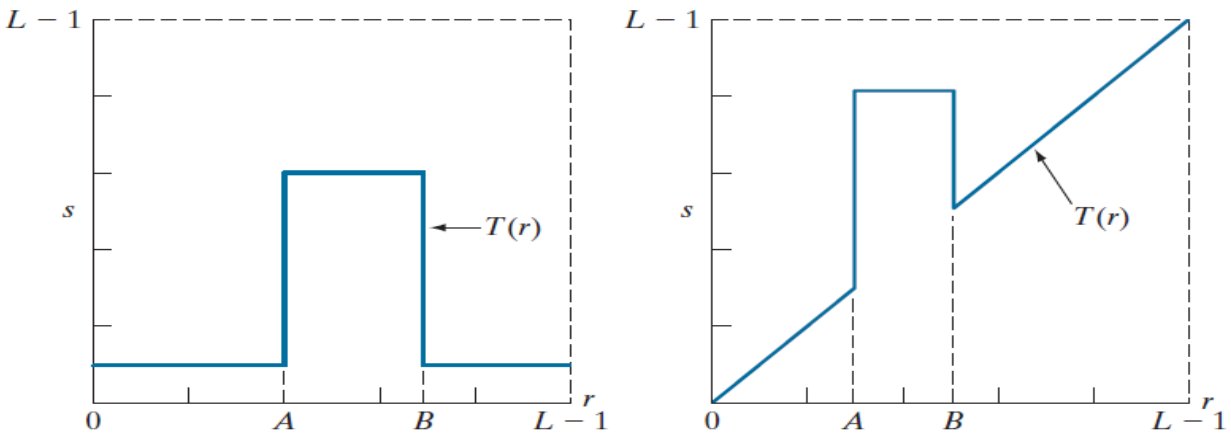

Figure 6: (a) This transformation function highlights range [*A*,*B*] and reduces all other intensities to a lower level. (b) This function highlights range [*A*,*B*] and leaves other intensities unchanged.

Bit-Plane Slicing

- Pixel values are integers composed of bits. For example, values in a 256-level grayscale image are composed of 8 bits (one byte).
- Instead of highlighting intensity-level ranges, we could highlight the contribution made to total image appearance by specific bits.
- As shown in figure 7, an 8-bit image may be considered as being composed of eight one-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image, and plane 8 all the highest-order bits
- The four higher-order bit planes, especially the first two, contain a significant amount of the visually-significant data.
- The lower-order planes contribute to more subtle intensity details in the image.

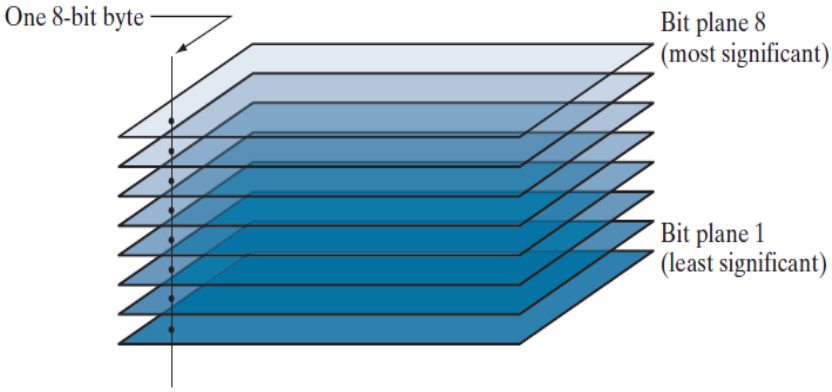

Figure 7: Bit planes of an 8-bit Image

- The binary image for the 8th bit plane of an 8-bit image can be obtained by thresholding the input image with a transformation function that maps to 0 intensity values between 0 and 127, and maps to 1 value between 128 and 255.

2.3 Histogram Processing

- Let r_k, for $k = 0,1,2,\ldots,L-1$, denote the intensities of an L-level digital image, $f(x, y)$. The *unnormalized histogram* of f is defined as:

$$h(r_k) = n_k \quad \text{for } k = 0,1,2,\ldots,L-1$$

where n_k is the number of pixels in f with intensity r_k, and the subdivisions of the intensity scale are called *histogram bins*. Similarly, the *normalized histogram* of f is defined as:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

where, as usual, M and N are the number of image rows and columns, respectively.

- Histogram shape is related to image appearance. For example, Figure 8 shows images histogram with four basic intensity characteristics: dark, light, low contrast, and high contrast.
- The *dark image* that the most populated histogram bins are concentrated on the lower (dark) end of the intensity scale.
- Similarly, the most populated bins of the *light image* are biased toward the higher end of the scale.
- An image with *low contrast* has a narrow histogram located typically toward the middle of the intensity scale.

- The components of the histogram of the *high-contrast* image cover a wide range of the intensity scale, and the distribution of pixels is not too far from uniform, with few bins being much higher than the others.

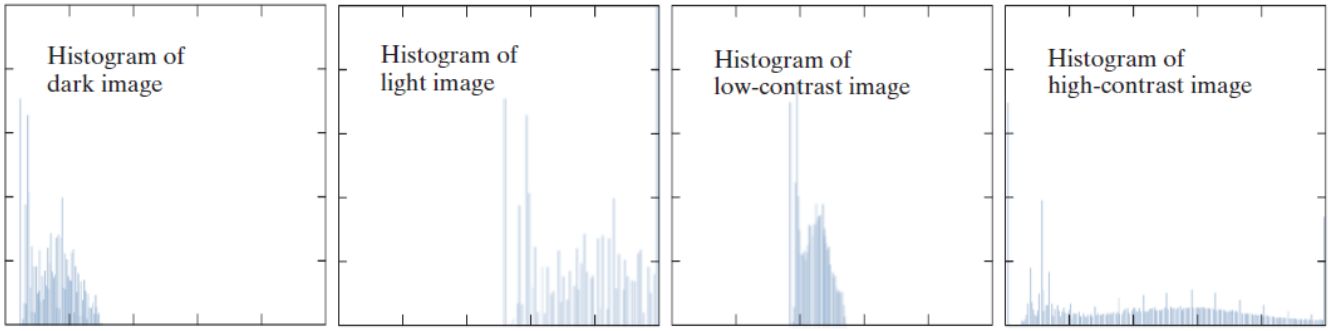

Figure 8: Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast

Histogram Equalization

- Assuming initially continuous intensity values, let the variable r denote the intensities of an image to be processed.
- We assume that r is in the range $[0, L-1]$, with $r = 0$ representing black and $r = L-1$ representing white.
- For r satisfying these conditions, the transformations (intensity mappings) of the form:

$$s = T(r) \qquad 0 \le r \le L-1$$

that produce an output intensity value, s, for a given intensity value r in the input image. We assume that:

a) $T(r)$ is a monotonic increasing function in the interval $0 \le r \le L-1$; and
b) $0 \le T(r) \le L-1$ for $0 \le r \le L-1$.

- The condition in (a) that $T(r)$ be monotonically increasing guarantees that output intensity values will never be less than corresponding input values, thus preventing artifacts created by reversals of intensity. Condition (b) guarantees that the range of output intensities is the same as the input.
- Figure 9 shows a function that satisfies conditions (a) and (b). Here, we see that it is possible for multiple input values to map to a single output value and still satisfy these two conditions.
- That is, a monotonic transformation function performs a one-to-one or many-to-one mapping. This is perfectly fine when mapping from r to s.
- Figure 9(a) presents a problem if we wanted to recover the values of r uniquely from the mapped values (inverse mapping can be visualized by reversing the direction of the arrows).

- This would be possible for the inverse mapping of s_k in Figure 9(a), but the inverse mapping of s_q is a range of values, which, of course, prevents us in general from recovering the original value of r that resulted in s_q.

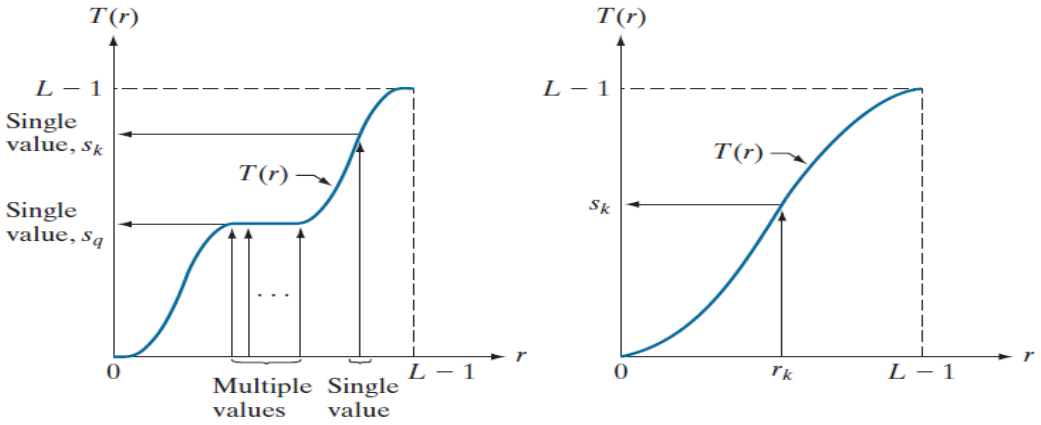

Figure 9: (a) Monotonic increasing function, showing how multiple values can map to a single value. (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.

- Figure 9(b) shows, requiring that $T(r)$ be strictly monotonic guarantees that the inverse mappings will be *single valued.*
- The intensity of an image may be viewed as a random variable in the interval $[0, L-1]$.
- Let $p_r(r)$ and $p_s(s)$ denote the PDFs of intensity values r and s in two different images.

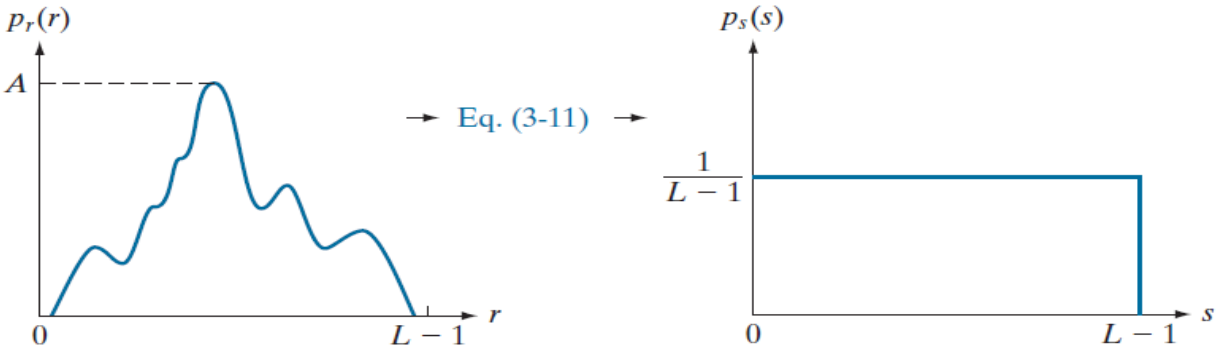

Figure 10: (a) An arbitrary PDF. (b) Result of applying to the image PDF.

- A fundamental result from probability theory is that if $p_r(r)$ and $T(r)$ are known, and $T(r)$ is continuous and differentiable over the range of values of interest, then the PDF of the transformed (mapped) variable s can be obtained as:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Equation 1

- Thus, we see that the PDF of the output intensity variable, s, is determined by the PDF of the input intensities and the transformation function used:
- A transformation function of particular importance in image processing is:

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

Equation 2

where w is a dummy variable of integration. The integral on the right side is the *cumulative distribution function* (CDF) of random variable r.

- We use above equation, to find the $p_s(s)$ corresponding to the transformation.
- We know from Leibniz's rule in calculus that the derivative of a definite integral with respect to its upper limit is the integrand evaluated at the limit.

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right] \\ &= (L-1)p_r(r)\end{aligned}$$

Equation 3

- Substituting this result for dr/ds in Equation 1, and noting that all probability values are positive, gives the result:

$$\begin{aligned}p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| \\ &= p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| \\ &= \frac{1}{L-1} \qquad 0 \le s \le L-1\end{aligned}$$

- For **discrete values,** we work with probabilities and summations instead of probability density functions and integrals.
- The probability of occurrence of intensity level r_k in a digital image is approximated by:

$$p_r(r_k) = \frac{n_k}{MN}$$

where MN is the total number of pixels in the image, and n_k denotes the number of pixels that have intensity r_k.

- $P_r(r_k)$ with $r_k \in [0, -1]$, is commonly referred to as a normalized image histogram.
- The discrete form of the transformation is:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \qquad k = 0,1,2,\ldots,L-1$$

L is the number of possible intensity levels in the image

- Thus, a processed (output) image is obtained by using above equation to map each pixel in the input image with intensity r_k into a corresponding pixel with level s_k in the output image. This is called a ***histogram equalization* or *histogram linearization* transformation.**

2.4 Fundamentals of Spatial Filtering

- Spatial filtering is used in a broad spectrum of image processing applications, so a solid understanding of filtering principles is important.
- The name *filter* is borrowed from frequency domain processing where "filtering" refers to passing, modifying, or rejecting specified frequency components of an image.
- For example, a filter that passes low frequencies is called a *lowpass filter*.
- The net effect produced by a lowpass filter is to smooth an image by blurring it.
- We can accomplish similar smoothing directly on the image itself by using *spatial filters*.
- Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors. If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*. Otherwise, the filter is a *nonlinear spatial filter*.

The Mechanics of Linear Spatial Filtering

- A linear spatial filter performs a sum-of-products operation between an image f and a ***filter kernel, w***.
- The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter.
- Other terms used to refer to a spatial filter kernel are *mask*, *template*, and *window*.
- Figure 11 illustrates the mechanics of linear spatial filtering using a 3 × 3 kernel. At any point (x, y) in the image, the response, $g(x, y)$, of the filter is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel:

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1)$$

- As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g, in the process.

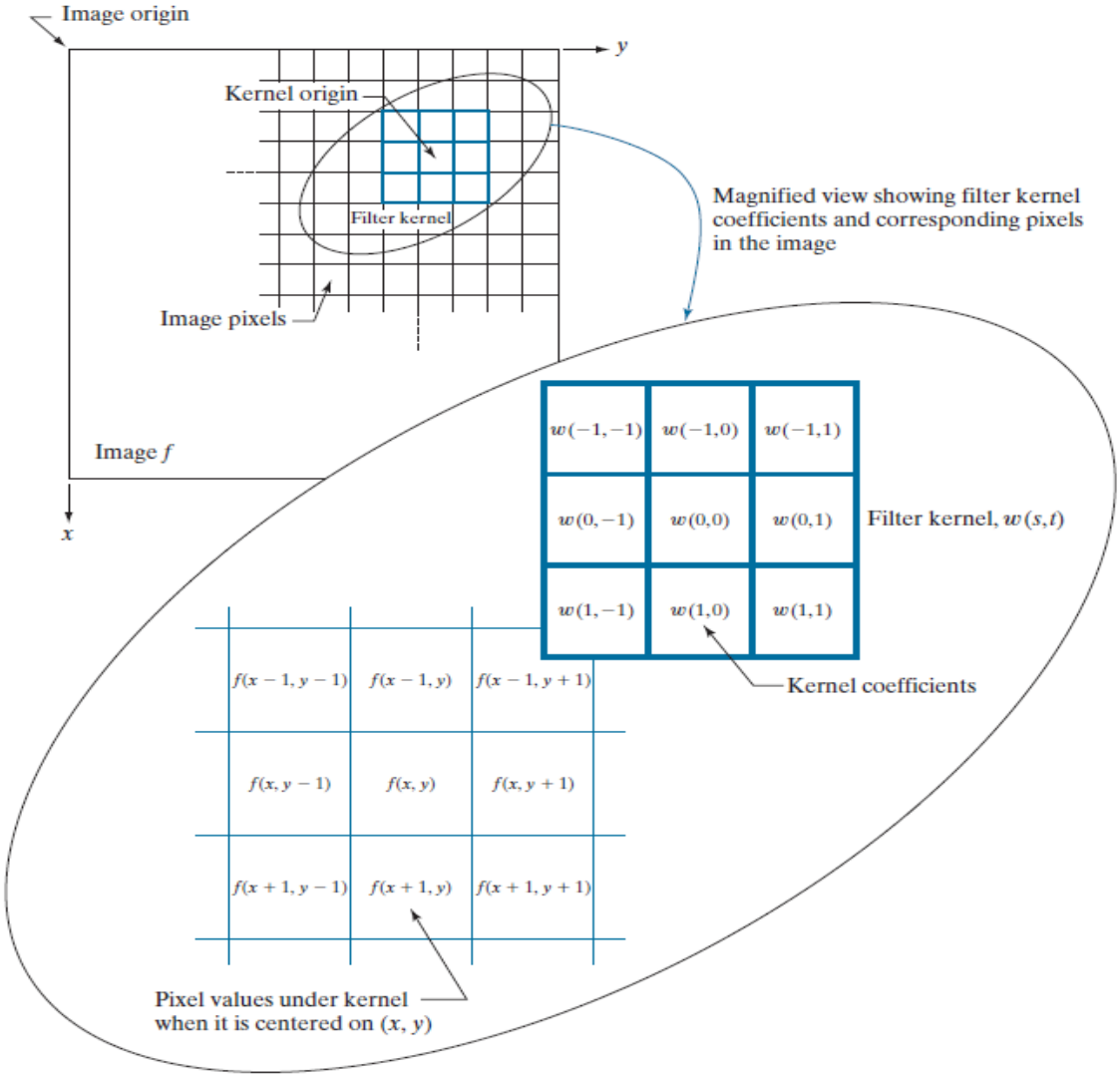

Figure 11: The mechanics of linear spatial filtering using a 3 × 3 kernel

- The center coefficient of the kernel, $w(0, 0)$, aligns with the pixel at location (x, y). For a kernel of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are nonnegative integers.
- In general, linear spatial filtering of an image of size $M \times N$ with a kernel of size $m \times n$ is given by the expression:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$

where x and y are varied so that the center (origin) of the kernel visits every pixel in f once.

Spatial Correlation and Convolution

- Correlation consists of moving the center of a kernel over an image, and computing the sum of products at each location.
- The mechanics of *spatial convolution* are the same, except that the correlation kernel is rotated by 180°.
- Thus, when the values of a kernel are symmetric about its center, correlation and convolution yield the same result.
- Figure 12 (a) shows a 1-D function, f, and a kernel, w. The kernel is of size 1 × 5, so $a = 2$ and $b = 0$ in this case.

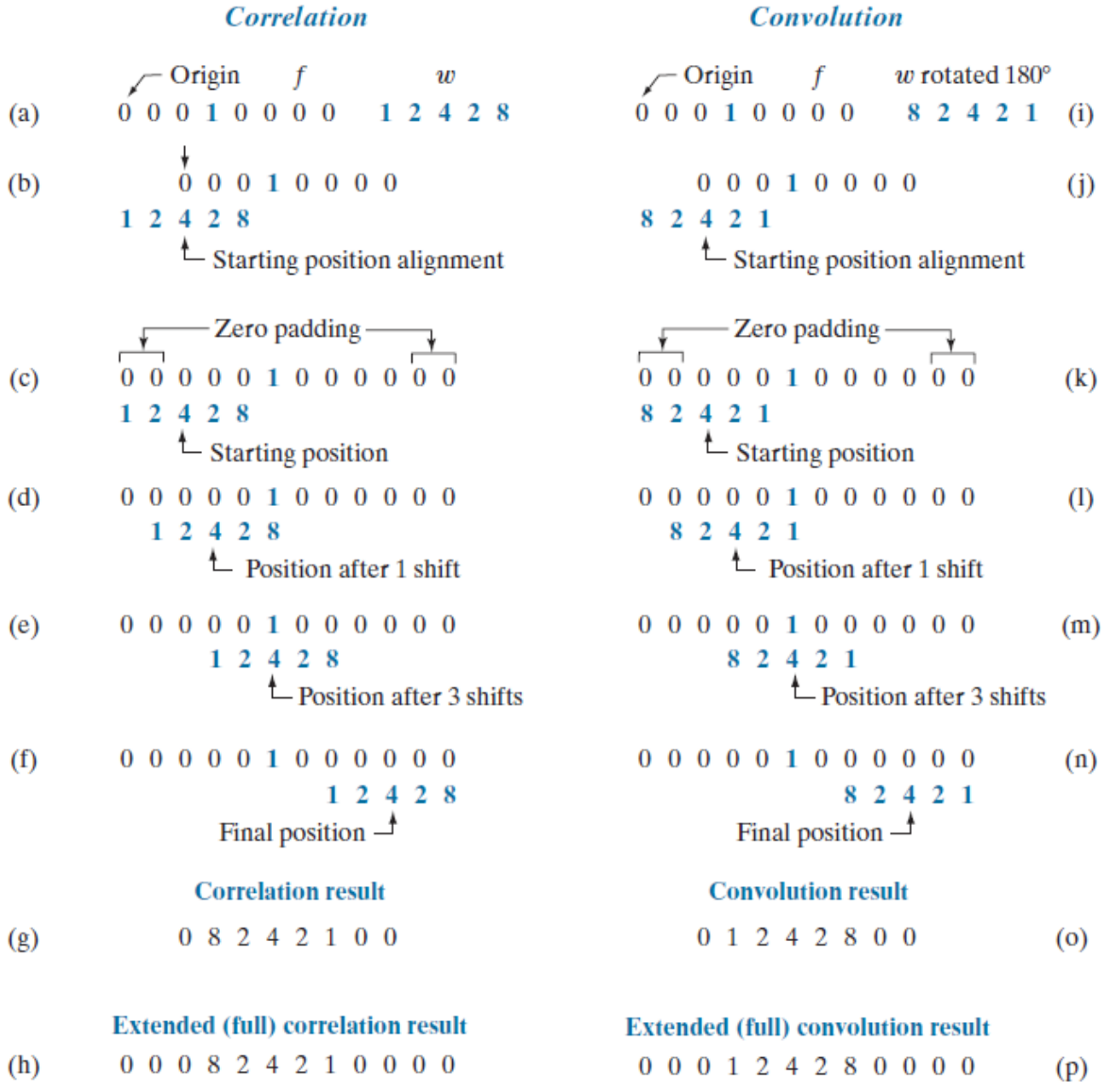

Figure 12: Illustration of 1-D correlation and convolution of a kernel, w, with a function *f* consisting of a discrete unit impulse.

- Figure 12(b) shows the starting position used to perform correlation, in which w is positioned so that its center coefficient is coincident with the origin of f.
- The first thing we notice is that part of w lies outside f, so the summation is undefined in that area.
- A solution to this problem is to *pad* function f with enough 0's on either side.
- In general, if the kernel is of size $1 \times m$, we need $(m - 1)/2$ zeros on either side of f in order to handle the beginning and ending configurations of w with respect to f.
- Figure 12(c) shows a properly padded function. In this starting configuration, all coefficients of the kernel overlap valid values.
- The first correlation value is the sum of products in this initial position, computed with $x = 0$.
- To obtain the second value of correlation, we shift the relative positions of w and f one pixel location to the right [i.e., we let $x = 1$] and compute the sum of products again. The result is $g(1) = 8$, as shown in the leftmost, nonzero location in Figure 12 (g).
- When $x = 2$, we obtain $g(2) = 2$. When $x = 3$, we get $g(3) = 4$ [see Figure 12(e)].
- Proceeding in this manner by varying x one shift at a time, we "build" the correlation result in Figure 12(g).
- Note that it took 8 values of x (i.e., $x = 0,1, 2,\ldots, 7$) to fully shift w past f so the *center* coefficient in w visited *every* pixel in f. Sometimes, it is useful to have every element of w visit every pixel in f.
- Figure 12 (h) shows the result of this *extended*, or *full*, correlation. As Figure 12 (g) shows, we can obtain the "standard" correlation by cropping the full correlation in Figure 12(h).
- The right side of Figure 12 shows the sequence of steps for performing convolution.
- The only difference here is that the kernel is *pre-rotated* by 180° prior to performing the shifting/sum of products operations.
- As the convolution in Figure 12(o) shows, the result of pre-rotating the kernel is that now we have an *exact* copy of the kernel at the location of the unit impulse.
- The correlation of a kernel w of size $m \times n$ with an image $f(x, y)$, denoted as $(w \star f)(x, y)$, is given by :

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

- As explained earlier, $a = (m - 1)/2$, $b = (n - 1)/2$, and we assume that m and n are odd integers.
- In a similar manner, the *convolution* of a kernel w of size $m \times n$ with an image $f(x, y)$, denoted by $(w \bigstar f)(x, y)$, is defined as:

$$(w \bigstar f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

where the minus signs align the coordinates of f and w when one of the functions is rotated by 180°.

2.5 Smoothing (Lowpass) Spatial Filter

- *Smoothing* (also called *averaging*) spatial filters are used to reduce sharp transitions in intensity.
- Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction.
- Smoothing is used to reduce irrelevant detail in an image, where "irrelevant" refers to pixel regions that are small with respect to the size of the filter kernel.
- Another application is for smoothing the false contours that result from using an insufficient number of intensity levels in an image.
- Linear spatial filtering consists of convolving an image with a filter kernel. Convolving a smoothing kernel with an image blurs the image, with the degree of blurring being determined by the size of the kernel and the values of its coefficients.

Box Filter Kernels

- The simplest, separable lowpass filter kernel is the *box kernel*, whose coefficients have the same value (typically 1).
- The name "box kernel" comes from a constant kernel resembling a box when viewed in 3-D. We showed a 3 × 3 box filter in Figure 13.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 13: Box filter

- An $m \times n$ box filter is an $m \times n$ array of 1's, with a normalizing constant in front, whose value is 1 divided by the sum of the values of the coefficients (i.e., 1 mn when all the coefficients are 1's).

Order-Statistic (Nonlinear) Filters

- Order-statistic filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the region encompassed by the filter.
- Smoothing is achieved by replacing the value of the center pixel with the value determined by the ranking result.

- The best-known filter in this category is the ***median filter,*** which, as its name implies, replaces the value of the center pixel by the median of the intensity values in the neighborhood of that pixel.
- Median filters provide excellent noise reduction capabilities for certain types of random noise, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of *impulse noise.*
- The *median*, ξ of a set of values is such that half the values in the set are less than or equal to ξ and half are greater than or equal to ξ.
- In order to perform median filtering at a point in an image, we first sort the values of the pixels in the neighborhood, determine their median, and assign that value to the pixel in the filtered image corresponding to the center of the neighborhood.
- For example, suppose that a 3 × 3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20.
- The median filter is by far the most useful order-statistic filter in image processing but is not the only one.
- The median represents the 50th percentile of a ranked set of numbers but ranking lends itself to many other possibilities.
- For example, using the 100th percentile results in the so-called ***max filter,*** which is useful for finding the brightest points in an image or for eroding dark areas adjacent to light regions.
- Th response of a 3 × 3 max filter is given by $R = \max \{ Z_k \mid k = 1, 2, 3,\ldots, 9\}$.
- The 0th percentile filter is the ***min filter***, used for the opposite purpose.

2.6 Sharpening (High Pass) Spatial Filter

- Sharpening highlights transitions in intensity. Uses of image sharpening range from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems.
- Image blurring could be accomplished in the spatial domain by pixel averaging (smoothing) in a neighborhood.
- Because averaging is analogous to integration, it is logical to conclude that sharpening can be accomplished by spatial differentiation.
- Smoothing is often referred to as lowpass filtering, a term borrowed from frequency domain processing. In a similar manner, sharpening is often referred to as *highpass* filtering.
- In this case, high frequencies (which are responsible for fine details) are passed, while low frequencies are attenuated or rejected.

One Dimensional Derivate

- The behavior of these derivatives in areas of constant intensity, at the onset and end of discontinuities (*step* and *ramp discontinuities*), and along *intensity ramps.*
- The definition of first derivate can be defined as:
  1. Must be zero in areas of constant intensity.
  2. Must be nonzero at the onset of an intensity step or ramp.
  3. Must be nonzero along intensity ramps.
- Similarly, the definition of second order derivate is:
  1. Must be zero in areas of constant intensity.
  2. Must be nonzero at the onset *and* end of an intensity step or ramp.
  3. Must be zero along intensity ramps.
- A basic definition of the *first-order* derivative of a one-dimensional function $f(x)$ is the difference:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- We used a partial derivative here in order to keep the notation consistent when we consider an image function of two variables, $f(x, y)$, at which time we will be dealing with partial derivatives along the two spatial axes.
- Clearly, $\partial f / \partial x = df / dx$ when there is only one variable in the function; the same is true for the second derivative.
- We define the *second-order* derivative of $f(x)$ as the difference:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

- These two definitions satisfy the conditions stated above, as we illustrate in Figure 14.

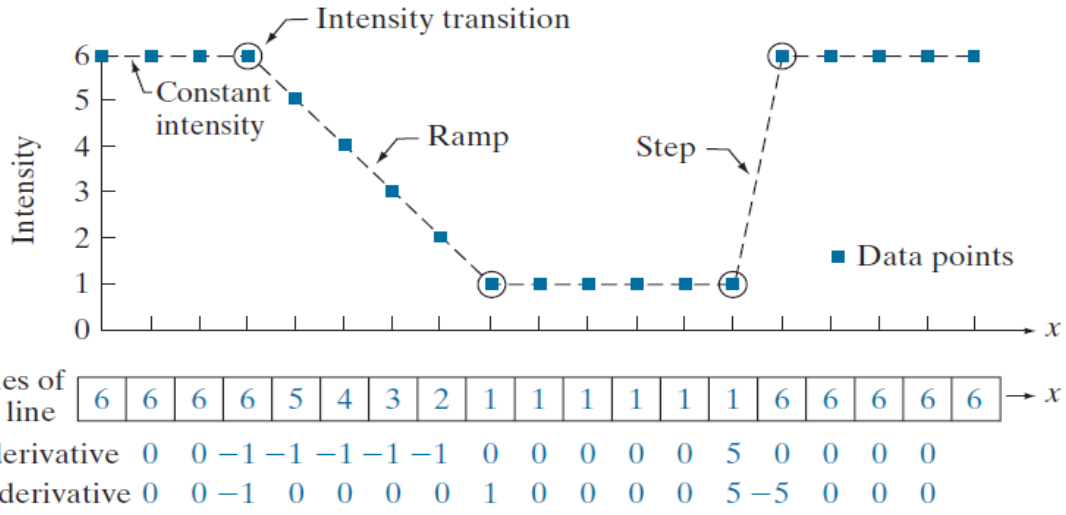

Figure 14: A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments. (b)Values of the scan line and its derivatives.

- The values denoted by the small squares in Figure 14(a) are the intensity values along a horizontal intensity profile.
- The actual numerical values of the scan line are shown inside the small boxes in 14(b).
- As Figure 14(a) shows, the scan line contains three sections of constant intensity, an intensity ramp, and an intensity step.
- The circles indicate the onset or end of intensity transitions.
- The first- and second-order derivatives, computed using the two preceding definitions, are shown below the scan line values in Figure 14(b), and are plotted in Figure 14(c).
- When computing the first derivative at a location x, we subtract the value of the function at that location from the next point.
- Similarly, to compute the second derivative at x, we use the previous and the next points in the computation.
- As we traverse the profile from left to right, we encounter first an area of constant intensity and, as Figure 14(b) and (c) show, both derivatives are zero there, so condition (1) is satisfied by both.
- Next, we encounter an intensity ramp followed by a step, and we note that the first-order derivative is nonzero at the onset of the ramp and the step; similarly, the second derivative is nonzero at the onset and end of both the ramp and the step; therefore, property (2) is satisfied by both derivatives.
- Finally, we see that property (3) is satisfied also by both derivatives because the first derivative is nonzero and the second is zero along the ramp.
- From this, we conclude that the second derivative enhances fine detail much better than the first derivative, a property ideally suited for sharpening images.
- Also, second derivatives require fewer operations to implement than first derivatives.

Using The Second Derivative for Image Sharpening—The Laplacian

- The Laplacian can be used for the implementation of 2-D, second-order derivatives and their use for image sharpening.
- The approach consists of defining a discrete formulation of the second-order derivative and then constructing a filter kernel based on that formulation.
- It can be shown that the simplest isotropic derivative operator (kernel) is the *Laplacian*, which, for a function (image) $f(x, y)$ of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Because derivatives of any order are linear operations, the Laplacian is a linear operator.
- To express this equation in discrete form, we use the definition of second derivate keeping in mind that we now have a second variable.

- In the *x*-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

- and, similarly, in the *y*-direction, we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

- It follows from the preceding three equations that the discrete Laplacian of two variables is:

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$