1.Some Basic Gray Level Transformations

We begin the study of image enhancement techniques by discussing gray-level transformation functions.These are among the simplest of all image enhancement techniques.The values of pixels, before and after processing, will be denoted by *r* and *s*, respectively. As indicated in the previous section, these values are related by an expression of the form s=T(r), where T is a transformation that maps a pixel value *r* into a pixel value *s*. Since we are dealing with digital quantities, values of the transformation function typically are stored in a one-dimensional array and the mappings from *r* to *s* are implemented via table lookups. For an 8-bit environment, a lookup table containing the values of *T* will have 256 entries.

As an introduction to gray-level transformations, consider Fig. 3.3, which shows three basic types of functions used frequently for image enhancement: linear (negative and identity transformations), logarithmic (log and inverse-log transformations), and power-law (*n*th power and *n*th root transformations).The identity function is the trivial case in which output intensities are identical to input intensities. It is included in the graph only for completeness.

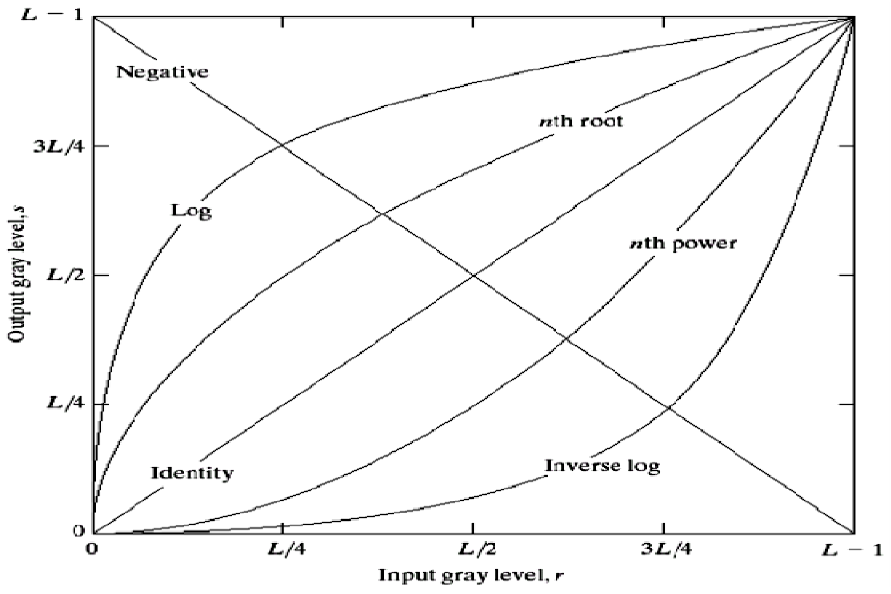
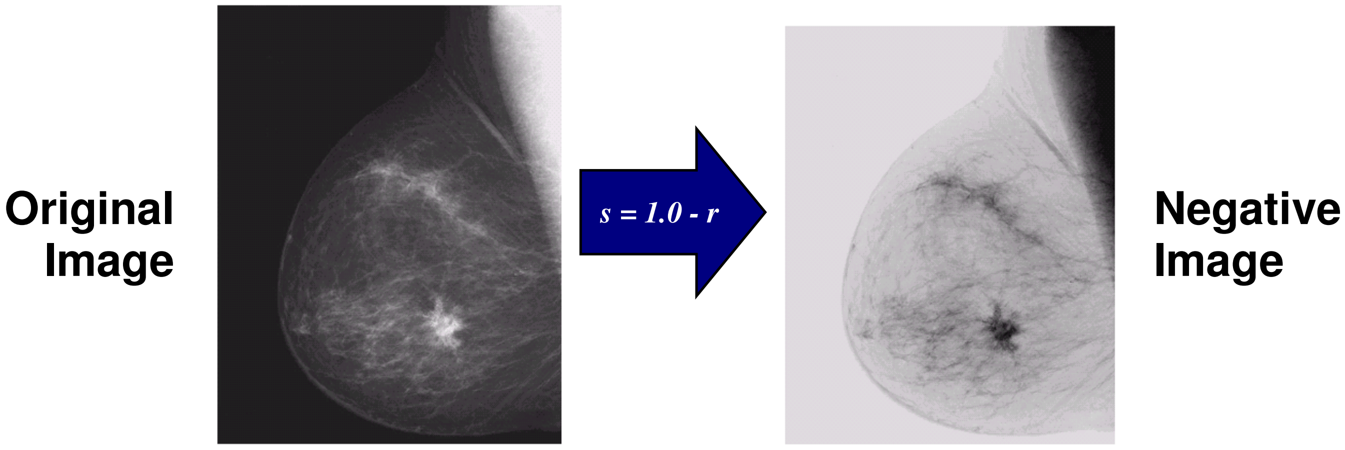


Image Negatives

The negative of an image with gray levels in the range [0,L-1]is obtained by using the negative transformation shown in Figure. 3.3, which is given by the expression

$$s = L - 1 - r.$$



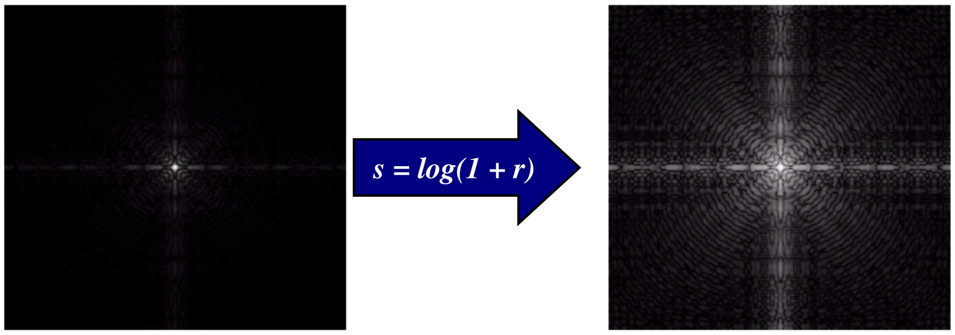
Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size. An example is shown in . The original image is a digital mammogram showing a small lesion. In spite of the fact that the visual content is the same in both images, note how much easier it is to analyse the breast tissue in the negative image in this particular case.

Log Transformations

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

where *c* is a constant, and it is assumed that r _ 0.The shape of the log curve in shows that this transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels.The opposite is true of higher values of input levels.We would use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values.The opposite is true of the inverse log transformation.Any curve having the general shape of the log functions shown in Fig. would accomplish this spreading/compressing of gray levels in an image.

In fact,

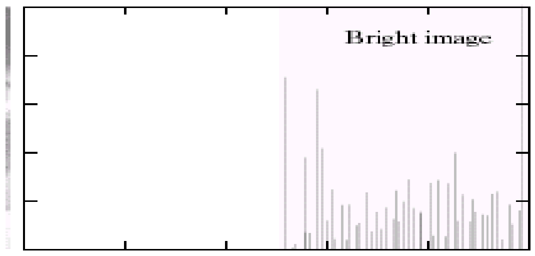


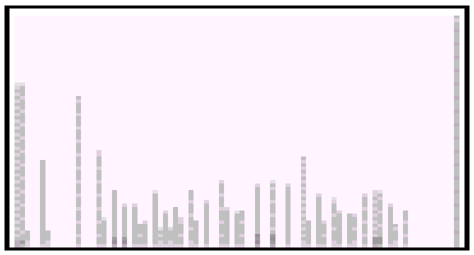
the power-law transformations discussed in the next section are much more versatile for this purpose than the log transformation. However, the log function has the important characteristic that it compresses the dynamic range of images with large variations in pixel values.A classic illustration of an application in which pixel values have a large dynamic range is the Fourier spectrum, which will be discussed in Chapter 4. At the moment, we are concerned only with the image characteristics of spectra. It is not unusual to encounter spectrum values

In fact, the power-law transformations discussed in the next section are much more versatile for this purpose than the log transformation. However, the log function has the important characteristic that it compresses the dynamic range of images with large variations in pixel values.A classic illustration of an application in which pixel values have a large dynamic range is the Fourier spectrum, which will be discussed in Chapter 4. At the moment, we are concerned only with the

image characteristics of spectra. It is not unusual to encounter spectrum values that range from 0 to or higher.While processing numbers such as these presents no problems for a computer, image display systems generally will not be able to reproduce faithfully such a wide range of intensity values. The net effectis that a significant degree of detail will be lost in the display of a typical Fourier
spectrum.

2. Histogram Processing





The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function hArkB=nk, where rk is the *k*th gray level and nk is the number
of pixels in the image having gray level rk. It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by *n*. Thus, a normalized histogram is given by pArkB=nk_n, for k=0, 1,p ,L-1. Loosely speaking, pArkB gives an estimate of the probability of occurrence of gray level rk. Note that the sum of all components of a

normalized histogram is equal to 1.

Histograms are the basis for numerous spatial domain processing techniques. Histogram manipulation can be used effectively for image enhancement, as shown in this section. In addition to providing useful image statistics, we shall see in subsequent chapters that the information inherent in histograms also is quite useful in other image processing applications, such as image compression and segmentation. Histograms are simple to calculate in software and also lend themselves to economic hardware implementations, thus making them a popular tool for real-time image processing.

As an introduction to the role of histogram processing in image enhancement, consider which is the pollen image shown in four

basic gray-level characteristics: dark, light, low contrast, and high contrast.The right side of the figure shows the histograms corresponding to these images. The horizontal axis of each histogram plot corresponds to gray level values, rk. The vertical axis corresponds to values of hArkB=nk or pArkB=nk_n if the values are normalized.Thus, as indicated previously, these histogram plots are simply plots of hArkB=nk versus rk or pArkB=nk_n versus rk.

3. Basics of Spatial Filtering

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Simple averaging filter

some neighborhood operations work with the valuesof the image pixels in the neighborhood *and* the corresponding values of asubimage that has the same dimensions as the neighborhood.The subimage iscalled a *filter,mask*, *kernel, template,* or *window*, with the first three terms beingthe most prevalent terminology.The values in a filter subimage are referred toas *coefficients*, rather than pixels.

The concept of filtering has its roots in the use of the Fourier transform for signal processing in the so-called *frequency domain*. This topic is discussed in more detail in Chapter 4. In the present chapter, we are interested in filtering operations that are performed directly on the pixels of an image.We use the term *spatial filtering* to differentiate this type of process from the more traditional frequency domain filtering.

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

The mechanics of spatial filtering are illustrated infig.The process consists simply of moving the filter mask from point to point in an image. At each point (x, y), the *response* of the filter at that point is calculated using a predefined relationship. For *linear* spatial filtering (see Section 2.6 regarding linearity), the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask.For the 3*3 mask shown in the result (or response), *R*, of linear filtering with the filter mask at a point (x, y) in the image is

which we see is the sum of products of the mask coefficients with the corresponding pixels directly under the mask. Note in particular that the coefficient w(0, 0) coincides with image value f(x, y), indicating that the mask is centered at (x, y) when the computation of the sum of products takes place. For a mask of size m*n, we assume that m=2a+1 and n=2b+1,where a and b are nonnegative integers. All this says is that our focus in the following discussion will be on masks of *odd* sizes, with the smallest meaningful size being

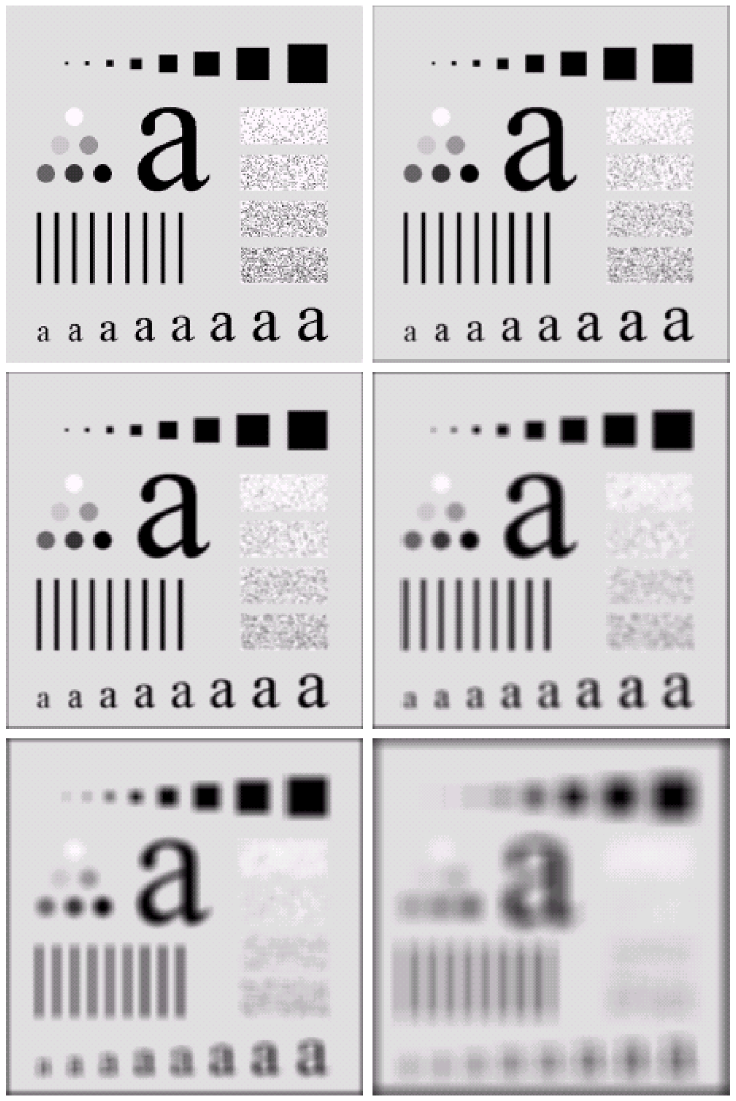
Smoothing Spatial Filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with a linear filter and also by nonlinear filtering.

Smoothing Linear Filters

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. For reasons explained in Chapter 4, they also are referred to a *lowpass filters*.

The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced “sharp” transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result



Sharpening Spatial Filters

The principal objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition. Uses of image sharpening vary and include applications ranging from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems.

In the last section, we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood. Since averaging is analogous to integration, it is logical to conclude that sharpening could be accomplished by spatial differentiation.This, in fact, is the case, and the discussion in this section deals with various ways of defining and implementing operators for sharpening by digital differentiation. Fundamentally, the strength of the response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.Thus, image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values.

In the two sections that follow, we consider in some detail sharpening filters that are based on first- and second-order derivatives, respectively. Before proceeding with that discussion, however, we stop to look at some of the fundamental properties of these derivatives in a digital context. To simplify the explanation, we focus attention on one-dimensional derivatives. In particular, we are interested in the behavior of these derivatives in areas of constant gray level (flat segments), at the onset and end of discontinuities (step and ramp discontinuities), and along gray-level ramps.These types of discontinuities can be used to model noise points, lines, and edges in an image.The behavior of derivatives during transitions into and out of these image features also is of interest.

The derivatives of a digital function are defined

4.Frequency Domain filters:

The frequency domain methods of image enhancement are based on convolution theorem. This isrepresented as,

$$g(x, y) = h(x, y) * f(x, y)$$

Where.

$$g(x, y) = \text{Resultant image}$$

$$h(x, y) = \text{Position invariant operator}$$

$$f(x, y) = \text{Input image}$$

The Fourier transform representation of equation above is,

$$G(u, v) = H(u, v)\, F(u, v)$$

The function H (u, v) in equation is called transfer function. It is used to boost the edges of inputimage f (x, y) to emphasize the high frequency components.

The different frequency domain methods for image enhancement are as follows.

1. Contrast stretching.
2. Clipping and thresholding.
3. Digital negative.
4. Intensity level slicing and
5. Bit extraction.

Contrast Stretching:

Due to non-uniform lighting conditions, there may be poor contrast between the background andthe feature of interest. Figure shows the contrast stretching transformations.

These stretching transformations are expressed asIn the area of stretching the slope of transformation is considered to be greater than unity. Theparameters of stretching transformations i.e., a and b can be determined by examining thehistogram of the image.

Clipping and Thresholding:

Clipping is considered as the special scenario of contrast stretching. It is the case in which the parameters are $\alpha = \gamma = 0$. Clipping is more advantageous for reduction of noise in input signals ofrange [a, b].

Threshold of an image is selected by means of its histogram

Digital Negative:

The digital negative of an image is achieved by reverse scaling of its grey levels to the transformation. They are much essential in displaying of medical images.

Intensity Level Slicing:

The images which consist of grey levels in between intensity at background and other objects require to reduce the intensity of the object. This process of changing intensity level is done withthe help of intensity level slicing. They are expressed as

5.Homomorphic filtering:

The illumination-reflectance model can be used to develop a frequency domain procedure for improving the appearance of an image by simultaneous gray-level range compression and contrast enhancement. An image f(x, y) can be expressed as the product of illumination andreflectance components:

$$f(x, y) = i(x, y)r(x, y).$$

Equation above cannot be used directly to operate separately on the frequency components of illumination and reflectance because the Fourier transform of the product of two functions is not separable; in other words,

$$\Im\{f(x, y)\} \neq \Im\{i(x, y)\}\Im\{r(x, y)\}.$$

Suppose, however, that we define

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y). \end{aligned}$$

Then

$$\begin{aligned} \Im\{z(x, y)\} &= \Im\{\ln f(x, y)\} \\ &= \Im\{\ln i(x, y)\} + \Im\{\ln r(x, y)\} \end{aligned}$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

where Fi (u, v) and Fr (u, v) are the Fourier transforms of ln i(x, y) and ln r(x, y), respectively. If

we process Z (u, v) by means of a filter function H (u, v) then, from

$$\begin{aligned} S(u, v) &= H(u, v)Z(u, v) \\ &= H(u, v)F_i(u, v) + H(u, v)F_r(u, v) \end{aligned}$$

where S (u, v) is the Fourier transform of the result. In the spatial domain

$$\begin{aligned} s(x, y) &= \Im^{-1}\{S(u, v)\} \\ &= \Im^{-1}\{H(u, v)F_i(u, v)\} + \Im^{-1}\{H(u, v)F_r(u, v)\}. \end{aligned}$$

By letting

$$i'(x, y) = \Im^{-1}\{H(u, v)F_i(u, v)\}$$

and

$$r'(x, y) = \Im^{-1}\{H(u, v)F_r(u, v)\},$$

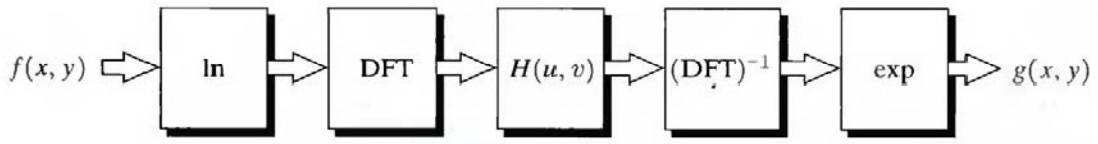
$$s(x, y) = i'(x, y) + r'(x, y).$$

Finally, as z (x, y) was formed by taking the logarithm of the original image f (x, y), the inverse
(exponential) operation yields the desired enhanced image, denoted by g(x, y); that is,

$$\begin{aligned} g(x, y) &= e^{s(x,y)} \\ &= e^{i'(x,y)} \cdot e^{r'(x,y)} \\ &= i_0(x, y)r_0(x, y) \end{aligned}$$

where

$$i_0(x, y) = e^{i'(x,y)}$$



Homomorphic filtering approach for image enhancement