UNIT - 5
IMAGE ENHANCEMENT IN SPATIAL DOMAIN

Spatial domain methods

Spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of the image.

Suppose we have a digital image which can be represented by a two dimensional random field $f(x, y)$. Spatial domain processes will be denoted by the expression

$$g(x, y) = T[f(x, y)] \quad \text{or } s = T(r)$$

The value of pixels, before and after processing, will be denoted by r and s, respectively.

Where *f(x, y)* is the input to the image, *g(x, y)* is the processed image and *T* is an operator on *f*.

The operator T applied on $f(x, y)$ may be defined over:

(i) A single pixel (x, y). In this case T is a grey level transformation (or mapping) function.

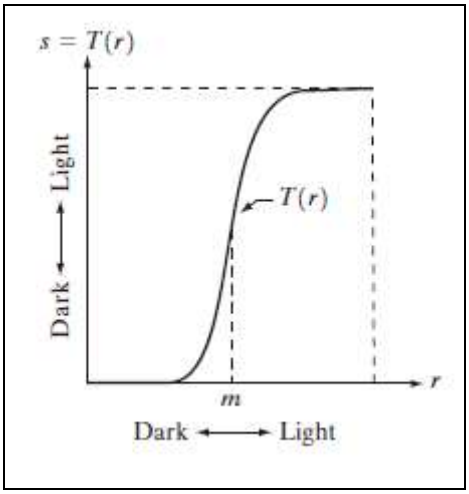
(ii) Some neighborhood of (x, y).

(iii) T may operate to a set of input images instead of a single image.

Examples of Enhancement Techniques

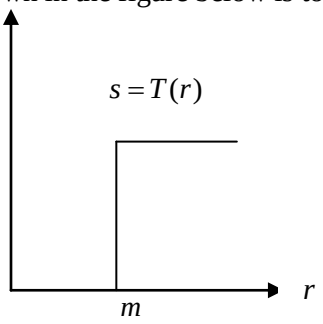
Contrast Stretching

The result of the transformation shown in the figure below is to produce an image of higher contrast than the original, by darkening the levels below m and brightening the levels above m in the original image. This technique is known as **contrast stretching**.

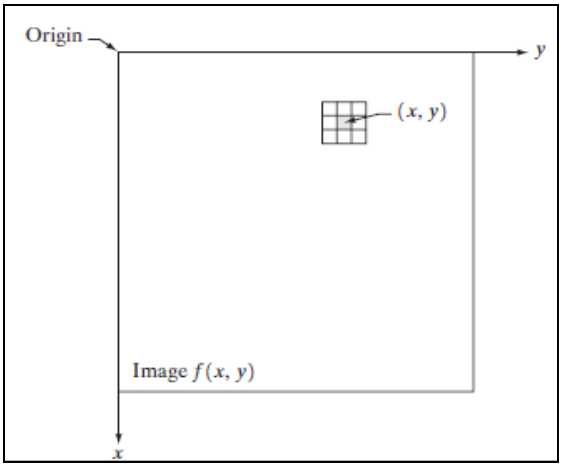


Thresholding

The result of the transformation shown in the figure below is to produce a binary image.



One of the principal approaches is based on the use of so-called *masks* (also referred to as *filters)*



So, a mask/filter: is a small (say 3X3) 2-D array, such as the one shown in the figure, in which the values of the mask coefficients determine the nature of the process, such as *image sharpening*. Enhancement techniques based on this type of approach often are referred to as *mask processing* or *filtering.*

Some Basic Grey Level Transform

We are dealing now with image processing methods that are based only on the intensity of single pixels.

Consider the following figure, which shows three basic types of functions used frequently for image enhancement:

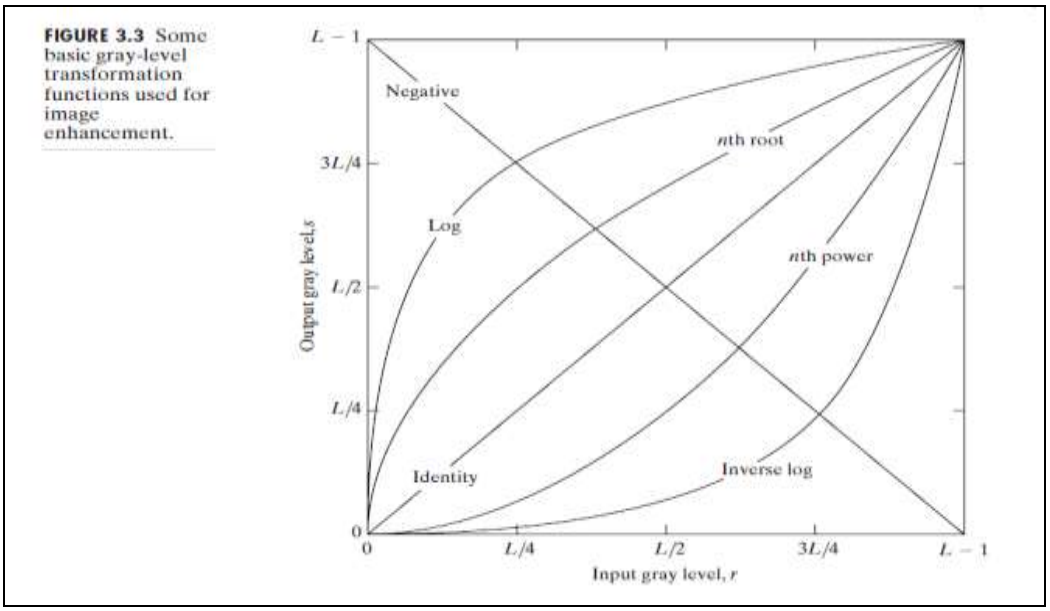


Fig: Some of the basic grey level transformation functions used for image enhancement

- The three basic types of functions used frequently for image enhancement:

1. Linear Functions:
    - Negative Transformation
    - Identity Transformation
2. Logarithmic Functions:
    - Log Transformation
    - Inverse-log Transformation
3. Power-Law Functions:
    - n^{th} power transformation
    - n^{th} root transformation

2.1.1 Identity Function

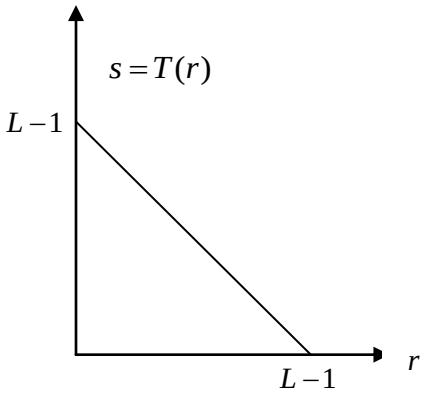
- Output intensities are identical to input intensities
- This function doesn't have an effect on an image, it was included in the graph only for completeness
- Its expression:

$$\mathbf{s = r}$$

2.1.2 Image Negatives

The negative of a digital image is obtained by the transformation function $s = T(r) = L - 1 - r$ shown in the following figure, where L is the number of grey levels. The idea is that the intensity of the output

image decreases as the intensity of the input increases. This is useful in numerous applications such as displaying medical images.



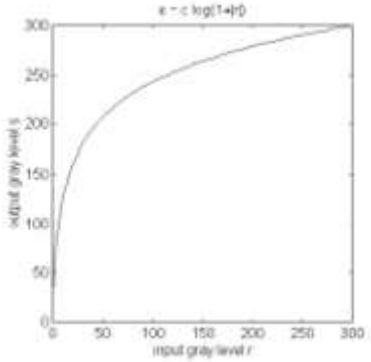
2.1.3 Log Transformation

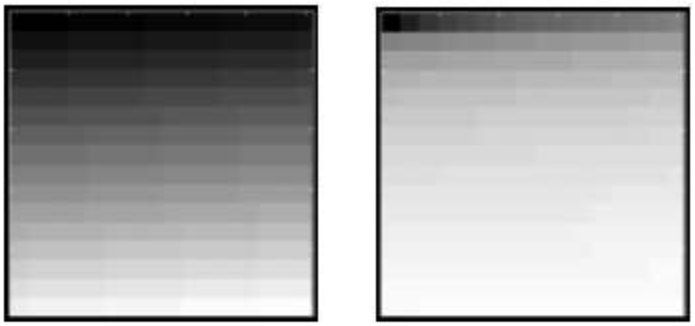
The general form of the log transformation:

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

Where c is a constant, and $r \geq 0$

- Log curve maps a narrow range of low gray-level values in the input image into a wider range of the output levels.
- Used to expand the values of dark pixels in an image while compressing the higher-level values.
- It compresses the dynamic range of images with large variations in pixel values.





Original Processed output

2.1.4 **Inverse Logarithm Transformation**

- Do opposite to the log transformations
- Used to expand the values of high pixels in an image while compressing the darker-level values.

2.1.5 Power-law transformations

- Power-law transformations have the basic form of:

$$s = c.r^{\gamma}$$

Where c and γ are positive constants

- Different transformation curves are obtained by varying γ (gamma)

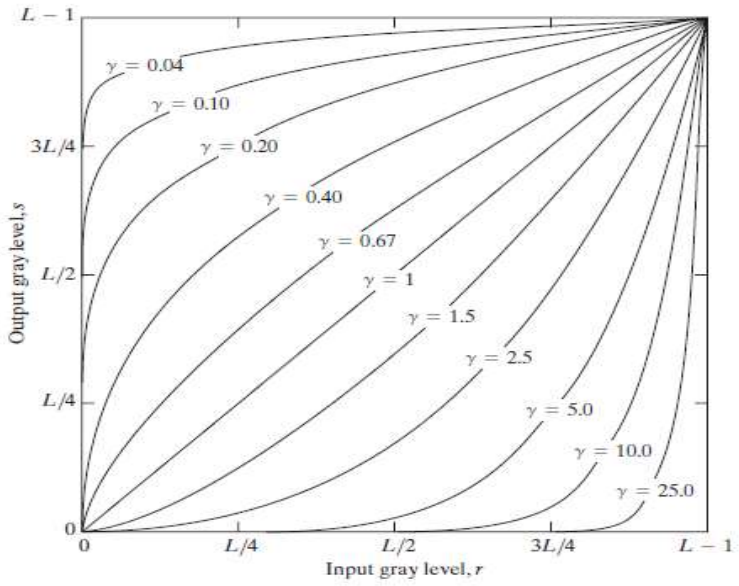
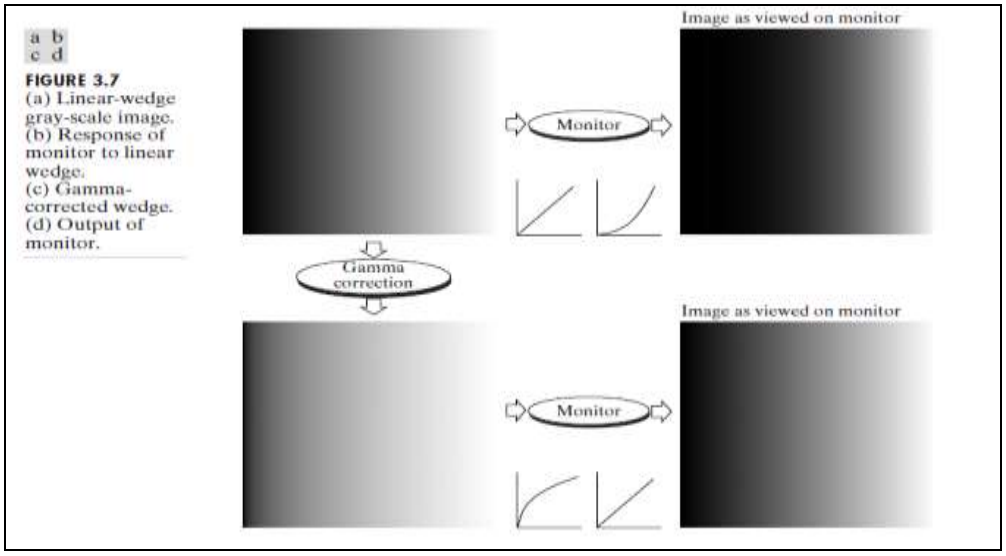


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Variety of devices used for image capture, printing and display respond according to a power law. The process used to correct these 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 γ=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.

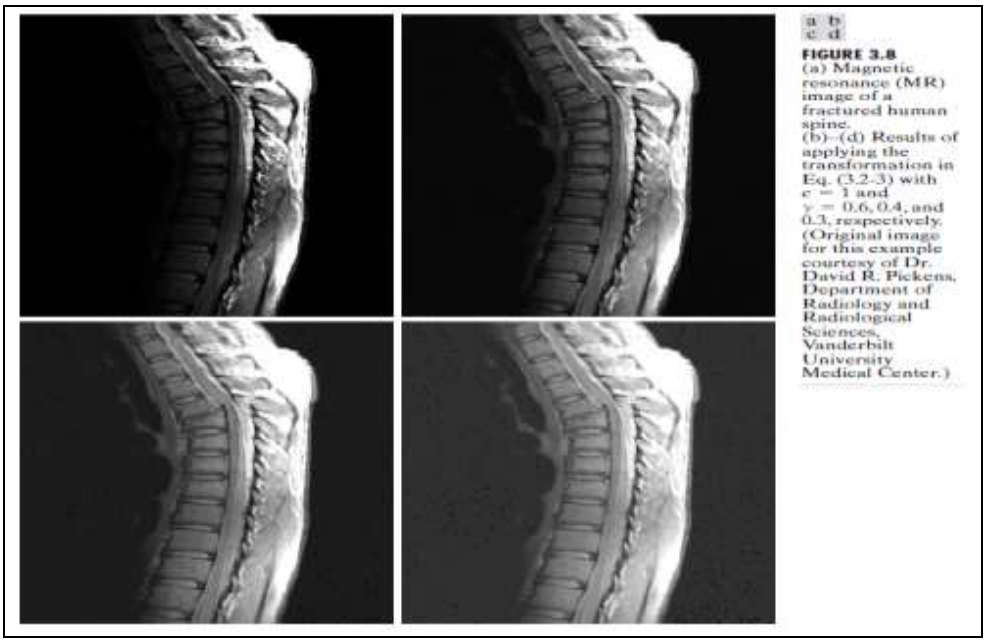
All we need to do is preprocess the input image before inputting it into the monitor by performing the transformation. The result is shown in Fig. 3.7(c).When input into the same monitor, this gamma-corrected input produces an output that is close in appearance to the original image, as shown in Fig. 3.7(d).



a b
c d

FIGURE 3.7
(a) Linear-wedge gray-scale image.
(b) Response of monitor to linear wedge.
(c) Gamma-corrected wedge.
(d) Output of monitor.

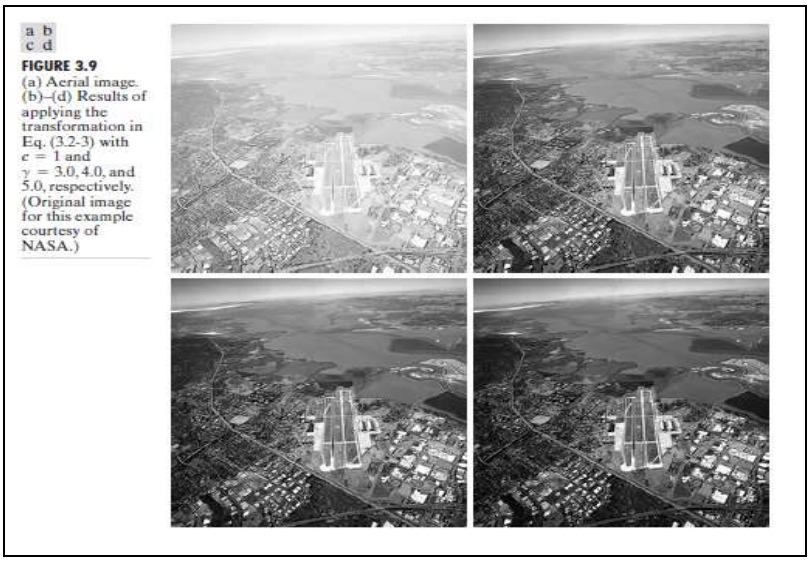
In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation. See figure 3.8



a b
c d

FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Another illustration of Power-law transformation



a b
c d

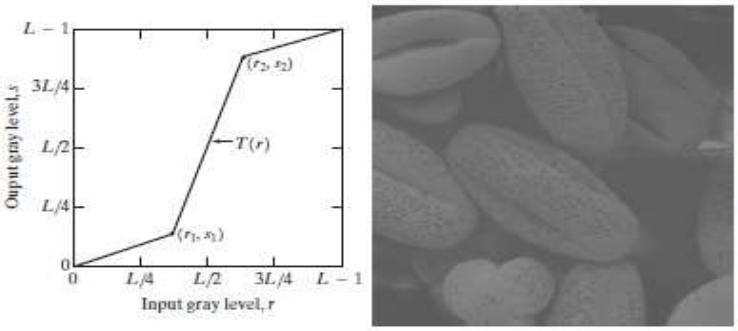
FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

Piecewise-Linear Transformation Functions

- Principle Advantage: Some important transformations can be formulated only as a piecewise function.
- Principle Disadvantage: Their specification requires more user input that previous transformations
- Types of Piecewise transformations are:
  - Contrast Stretching
  - Gray-level Slicing
  - Bit-plane slicing

2.1.3 Contrast Stretching

Low contrast images occur often due to poor or non-uniform lighting conditions, or due to nonlinearity, or small dynamic range of the imaging sensor. Contrast-stretching transformation, which is used to enhance the low contrast images

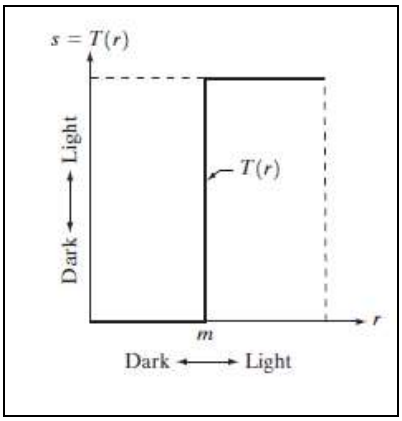


a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Figure 3.10(b) shows an 8-bit image with low contrast. Fig. 3.10(c) shows the result of contrast stretching, obtained by setting (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1]. Finally, Fig. 3.10(d) shows the result of using the *thresholding function* defined previously, with r1=r2=m, the mean gray level in the image.

- Figure 3.10(a) shows a typical transformation used for contrast stretching. The locations of points (r1, s1) and (r2, s2) control the shape of the transformation function.
- If r1 = s1 and r2 = s2, the transformation is a linear function that produces no changes in gray levels.
- If r1 = r2, s1 = 0 and s2 = L-1, the transformation becomes a *thresholding function* that creates a binary image



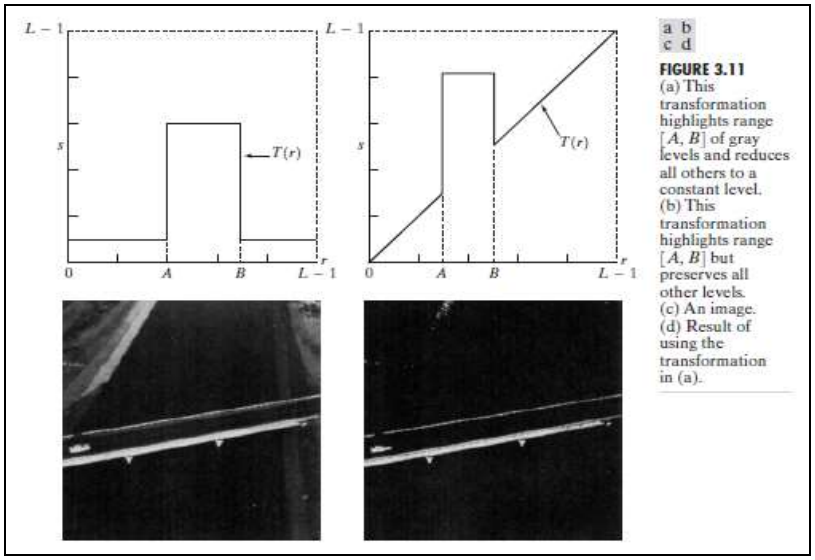
- Intermediate values of (r1, s1) and (r2, s2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.

  In general, r1 ≤ r2 and s1 ≤ s2 is assumed, so the function is always increasing

Gray-level Slicing

This technique is used to highlight a specific range of gray levels in a given image. It can be implemented in several ways, but the two basic themes are:

- 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 gray levels unchanged.
- Fig 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).



a b
c d

FIGURE 3.11
(a) This transformation highlights range [*A*, *B*] of gray levels and reduces all others to a constant level. (b) This transformation highlights range [*A*, *B*] but preserves all other levels. (c) An image. (d) Result of using the transformation in (a).

Bit-plane Slicing

Pixels are digital numbers, each one composed of 8 bits. Plane 0 contains the least significant bit and plane 7 contains the most significant bit. Instead of highlighting gray-level range, we could highlight the contribution made by each bit

- This method is useful and used in image compression.

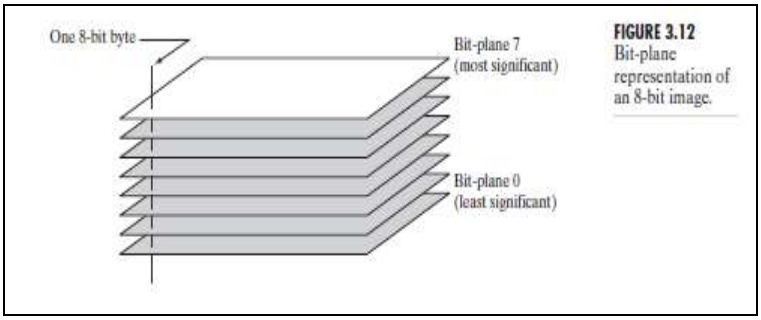


FIGURE 3.12 Bit-plane representation of an 8-bit image.

Most significant bits contain the majority of visually significant data

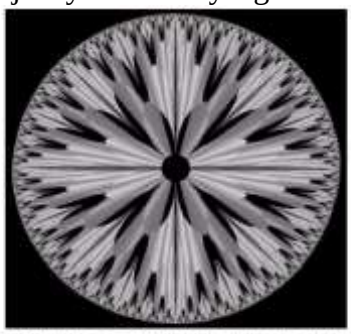


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

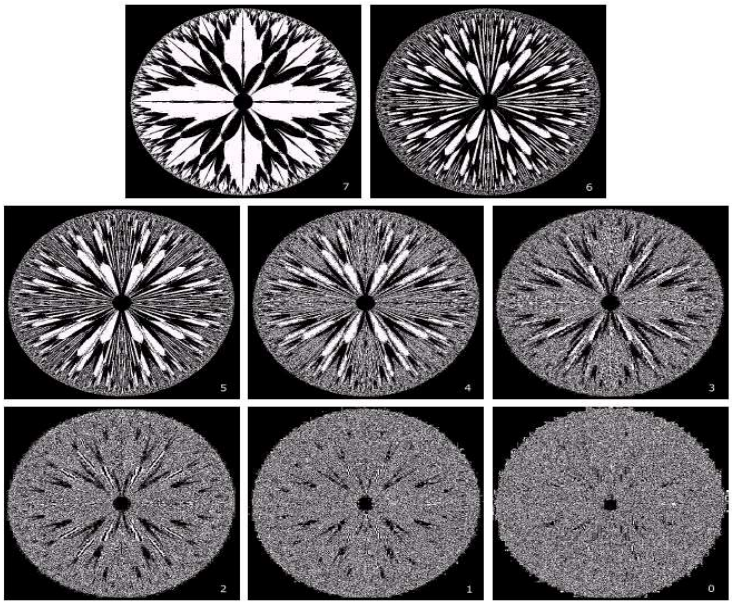


FIGURE 3.14 The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

2.2 Histogram processing. Definition of the histogram of an image.

By processing (modifying) the histogram of an image we can create a new image with specific desired properties.

Suppose we have a digital image of size $N \times N$ with grey levels in the range $[0, L-1]$. The histogram of the image is defined as the following discrete function:

$$p(r_k) = \frac{n_k}{n}$$

where

r_k is the kth grey level, $k = 0,1,\ldots,L-1$

n_k is the number of pixels in the image with grey level r_k

n is the total number of pixels in the image

The histogram represents the frequency of occurrence of the various grey levels in the image. A plot of this function for all values of k provides a global description of the appearance of the image.

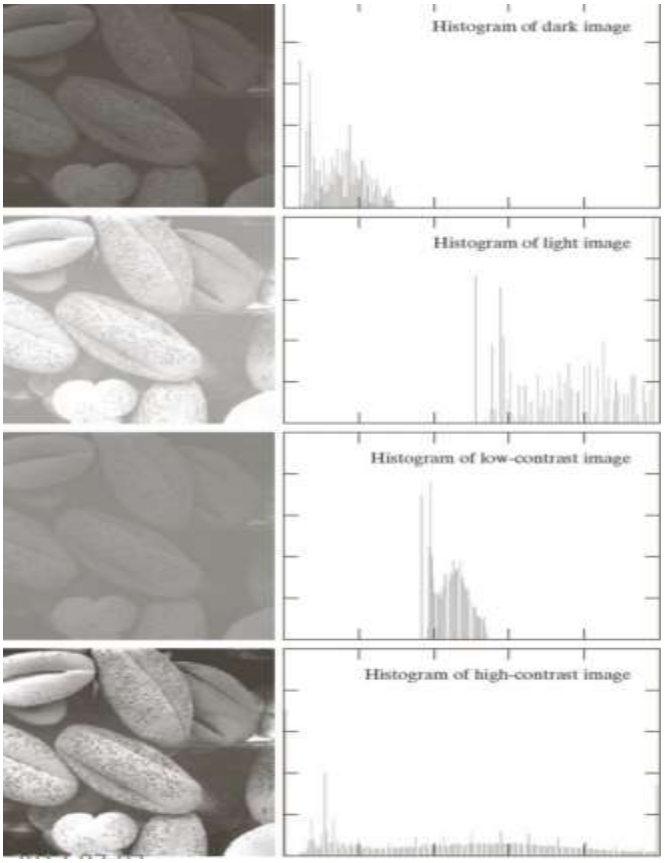


Fig: Four basic image types: (a) dark (b) light (c) low contrast (d) high contrast (e) Their corresponding histograms

- Images whose pixels occupy the entire range of intensity levels and tend to be distributed uniformly will have an appearance of high contrast
- It is possible to develop a transformation function that can automatically achieve this effect, based on the histogram of the input image.

- **Types of processing:**
  - **Histogram equalization**
  - **Histogram matching (specification)**
  - **Local enhancement**

Histogram equalisation (Automatic)

Consider transform of the form $s = T(r)$ $0 \leq r \leq 1$

We assume that *T(r)* satisfies the following conditions

(a) *T(r)* monotonically increasing in range $0 \leq r \leq 1$

(b) $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$

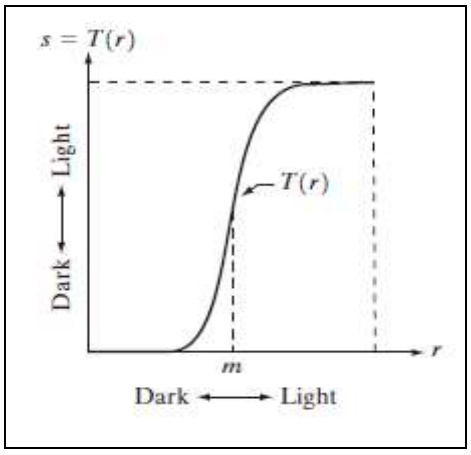


Fig: A grey level transformation function that is both single valued and monotonically increasing.

- The requirement in (a) that T(r) be single valued is needed to guarantee that the inverse transformation will exist, and the monotonicity condition preserves the increasing order from black to white in the output image
- Condition (b) guarantees that the output gray levels will be in the same range as the input levels.

The inverse transformation from *s* back to *r* is denoted $r = T^{-1}(s)$ $0 \leq S \leq 1$

(Not in syllabus)

PDF (Probability of occurrence of a particular event) For example, what's the probability for the temperature between 70and 80?

Definition of PDF

Let X be a continuous random variable. Then a probability distribution or probability density function (pdf) of X is a function f (x) such that for any two numbers a and b with a≤ b,

$$P(a \leq X \leq b) = \int_a^b f(X)dx$$

That is, the probability that X takes on a value in the interval [a; b] is the area above this interval and under the graph of the density function. The graph of f (x) is often referred to as the density curve.

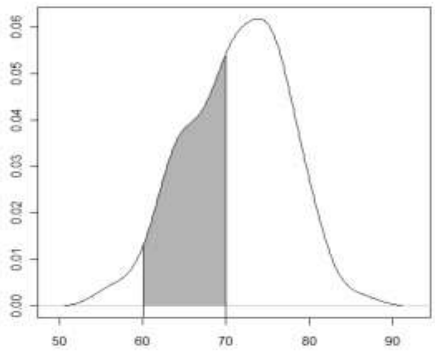


Figure: $P(60 \leq X \leq 70)$

- $p_r(r)$ denote the probability density function (pdf) of the variable r
- $p_s(s)$ denote the probability density function (pdf) of the variable s
- *If* $p_r(r)$ and T(r) is known then $p_s(s)$ can be obtained from

$$P_s(s) = P_r(r)\left|\frac{dr}{ds}\right| \quad \text{.....} \quad \text{(a)}$$

A transformation function of particular importance in image processing has the form

$$s = T(r) = \int_0^r p_r(w)dw,\ 0 \le r \le 1 \qquad (1)$$

By observing the transformation of equation (1) we immediately see that it possesses the following properties:

(i) $0 \le s \le 1$.

(ii) $r_2 > r_1 \Rightarrow T(r_2) \ge T(r_1)$, i.e., the function $T(r)$ is increasing with r.

(iii) $s = T(0) = \int_0^0 p_r(w)dw = 0$ and $s = T(1) = \int_0^1 p_r(w)dw = 1$. Moreover, if the original image has intensities **only** within a certain range $[r_{\min}, r_{\max}]$ then $s = T(r_{\min}) = \int_0^{r_{\min}} p_r(w)dw = 0$ and $s = T(r_{\max}) = \int_0^{r_{\max}} p_r(w)dw = 1$ since $p_r(r) = 0, r < r_{\min}$ and $r > r_{\max}$. Therefore, the new intensity s takes always all values within the available range [0 1].

Suppose that $P_r(r)$, $P_s(s)$ are the probability distribution functions (PDF's) of the variables r and s respectively.

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr}\left[\int_0^r P_r(w)dw\right] = P_r(r) \qquad \text{(b)}$$

From equation (a) $P_s(s) = P_r(r)\left|\frac{dr}{ds}\right|$

$$= P_r(r)\left|\frac{1}{P_r(r)}\right| = 1 \quad 0 \le s \le 1$$

For discrete values we deal with probabilities and summations instead of probability density functions and integrals. The probability of occurrence of gray level r_k in an image is approximated by

$$p(r_k) = \frac{n_k}{N^2} \qquad k=0,1,2,\ldots L-1$$

N is the total number of pixels in the image, n_k is the number of pixels that have gray level r_k and L is the total number of possible gray level in the image.

Unfortunately, in a real life scenario we must deal with digital images. The discrete form of histogram equalization is given by the relation

$$s = T(r) = \sum_{j=0}^{k} P_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$$

Where k=0, 1,2,..L-1

Above equation is called histogram equalization or histogram linearization.

The inverse transform from s back to r is denoted by

$r_k = T^{-1}(s_k)$ k=0,1,2….L-1

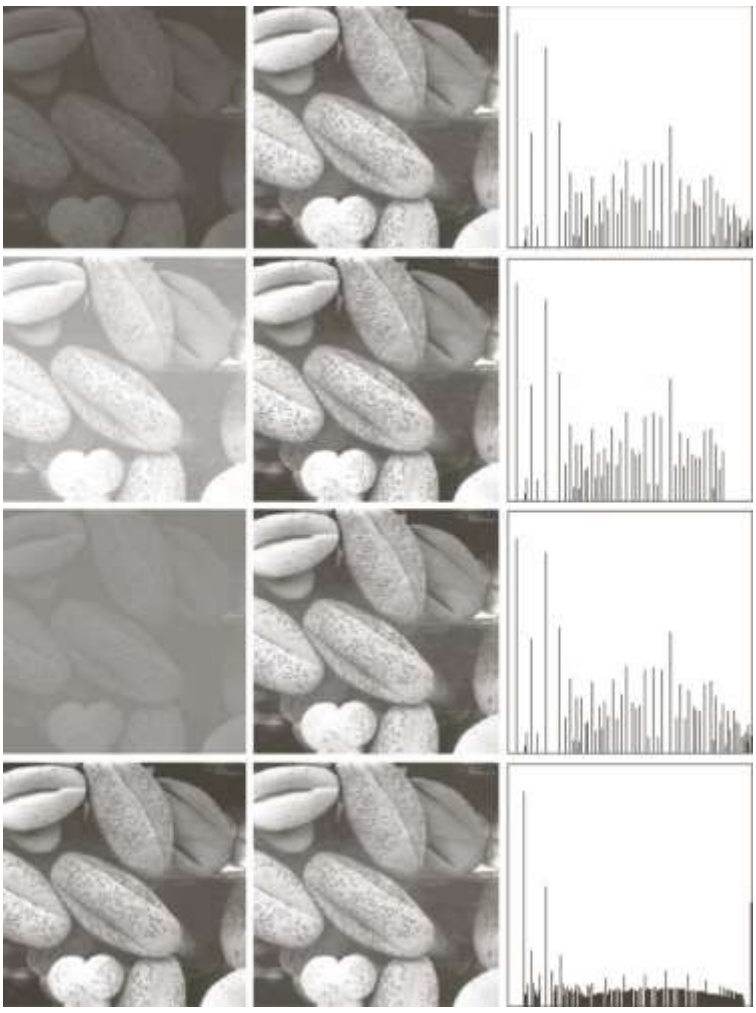
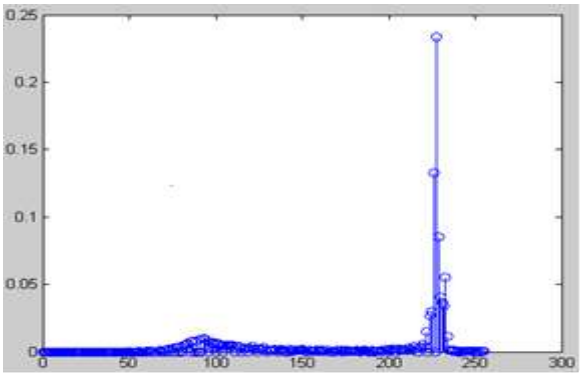


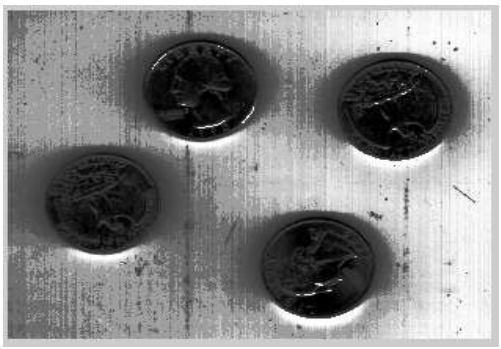
Fig: (a) Images (b) Result of histogram equalization (c) Corresponding histogram.

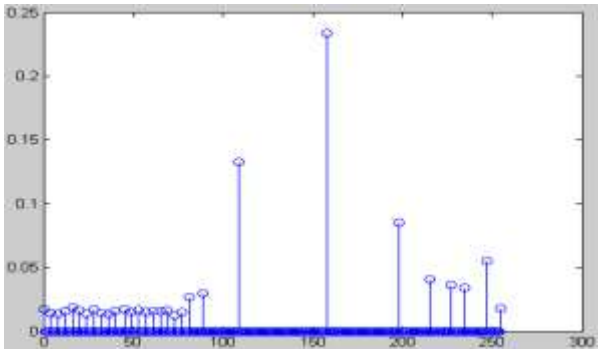
Conclusion

From the above analysis it is obvious that the transformation of equation (1) converts the original image into a new image with uniform probability density function. This means that in the new image all intensities are present [look at property (iii) above] and with equal probabilities. The whole range of intensities from the absolute black to the absolute white is explored and the new image will definitely have higher contrast compared to the original image.

Histogram equalization may not always produce desirable results, particularly if the given histogram is very narrow. It can produce false edges and regions. It can also increase image "graininess" and "patchiness."







Histogram Matching

- Histogram equalization does not allow interactive image enhancement and generates only one result: an approximation to a uniform histogram.
-

Suppose we want to specify a particular histogram shape (not necessarily uniform) which is capable of highlighting certain grey levels in the image.

Let us suppose that P_r *(r)* and P_z *(z) represent* the PDFs of *r* and *z* where r is the input pixel intensity level and *z* is the output pixel intensity

Suppose that histogram equalization is first applied on the original image *r*

$$s_k = T(r_k) = \sum_{j=0}^{k} P_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n} \qquad \text{k=0,1,2…L-1}$$

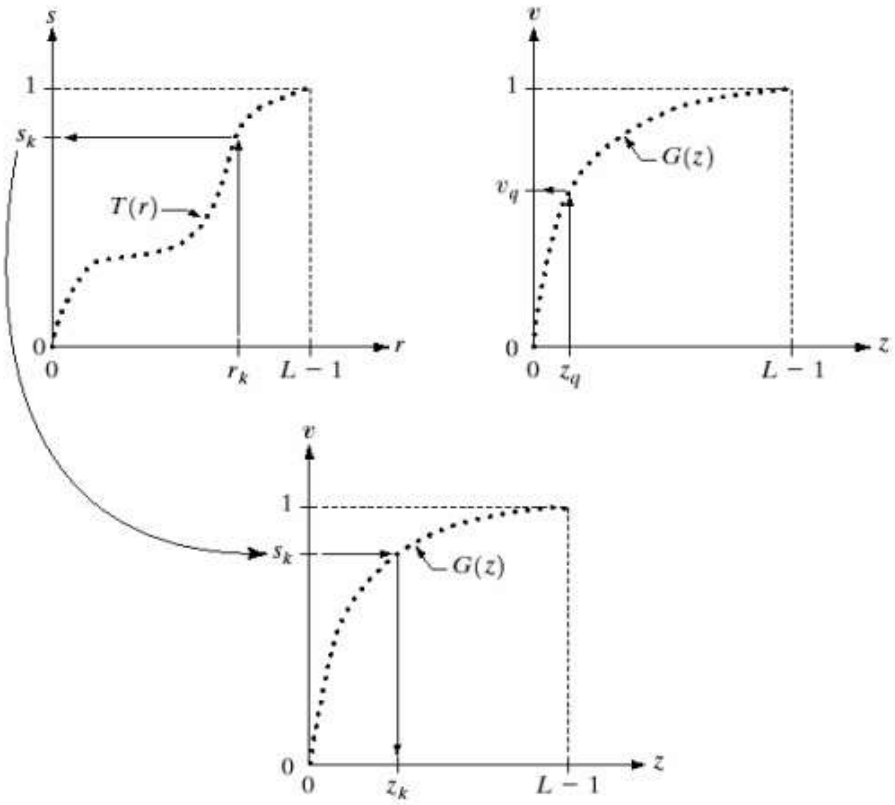
Where n is the total number of pixels in the image, n_k is the number of pixels that have gray level r_k and L is the total number of possible gray level in the image.

Suppose that the desired image *z* is available and histogram equalization is applied as well

$$v_k = G(z_k) = \sum_{i=0}^{k} P_z(z_i) = s_k \qquad \textit{Where } k=0,1,2,\ldots L-1$$

Finally $z_k = G^{-1}[T(r_k)] \qquad k = 0,1,2 \ldots . L-1$
Or $z_k = G^{-1}(s_k) \qquad k = 0,1,2 \ldots . L-1$

Implementation



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.

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, *r1*, maps to *s1*; the second gray level, *r2*, maps to *s2*; the kth level *rk*, maps to *sk*; and so on (the important point here is the ordered correspondence between these values).

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 *Pz(z)* by using Eq.

$$v_k = G(z_k) = \sum_{i=0}^{k} P_z(z_i) = s_k \qquad \text{Where } k=0,1,2,\ldots..L-1 \qquad (1)$$

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. (1) 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 $s = T(r) = \int_0^r p_r(w)dw$. This idea is shown in Fig. 3.19(c).

Since we really do not have the z's, we must resort to some sort of iterative scheme to find z from s. From above equation we have $G(z_k)=s_k$, *i.e* $G(z_k)-s_k=0$,. Let $z_k = \hat{z}$ for each value of k, where is the *smallest* integer in the interval [0, L-1] such that $G(\hat{z}) - s_k \geq 0$ where k=0,1…..L-1. Minimum value of $\hat{z}$ for which above equation is satisfied give z_k.

The procedure we have just developed for histogram matching may be summarized as follows:

1. Obtain the histogram equalization of the given image using equation $s_k = T(r_k) = \sum_{j=0}^{k} P_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$ to precompute a mapped level sk for each level rk

3. Obtain the transformation function G from the given pz(z) using $v_k = G(z_k) = \sum_{i=0}^{k} P_z(z_i) = s_k$ Where k=0,1,2,….L−1.

4. Precompute z_k for each value of s_k using the iterative scheme

5. For each pixel in the original image, if the value of that pixel is rk, map this value to its corresponding level s_k; then map level s_k into the final level z_k. Use the precomputed values from Steps (2) and (4) for these mappings.

(Example)

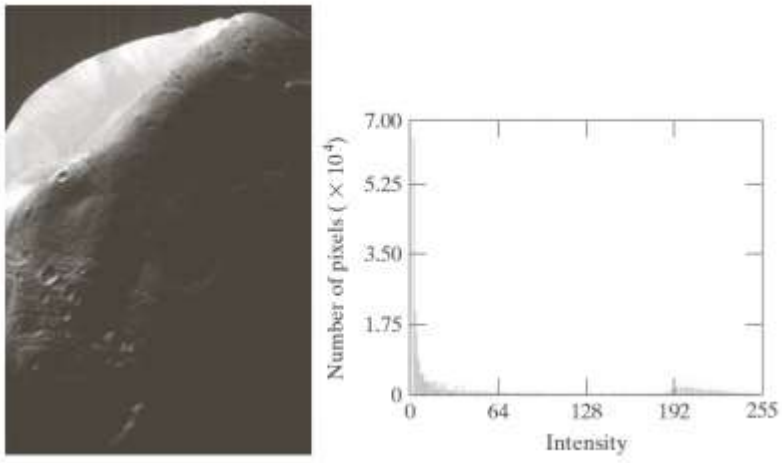


Fig: (a) Image of the mars moon photo taken by NASA's mars global surveyor (d) Histogram.

Large concentration of pixels in the dark region of the histogram.

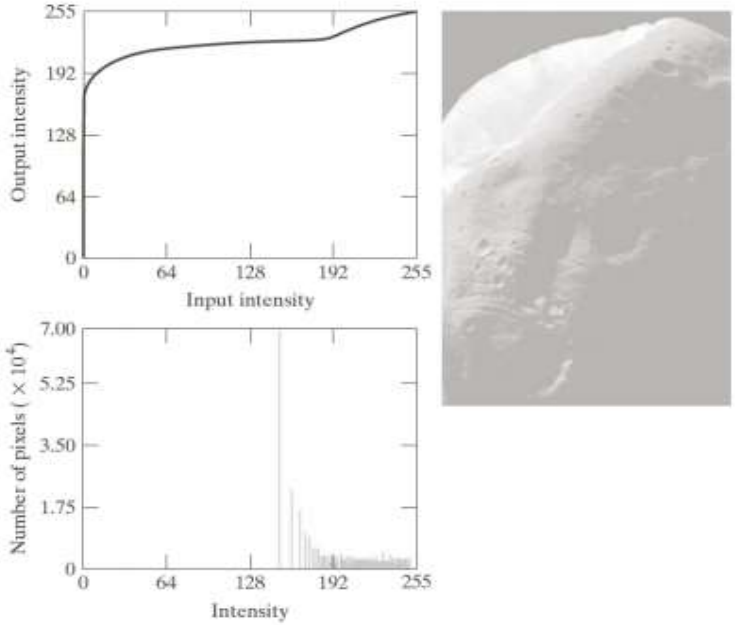


Fig: (a) Transformation function for histogram equalization (b) Histogram equalized image (c) Histogram of (b)

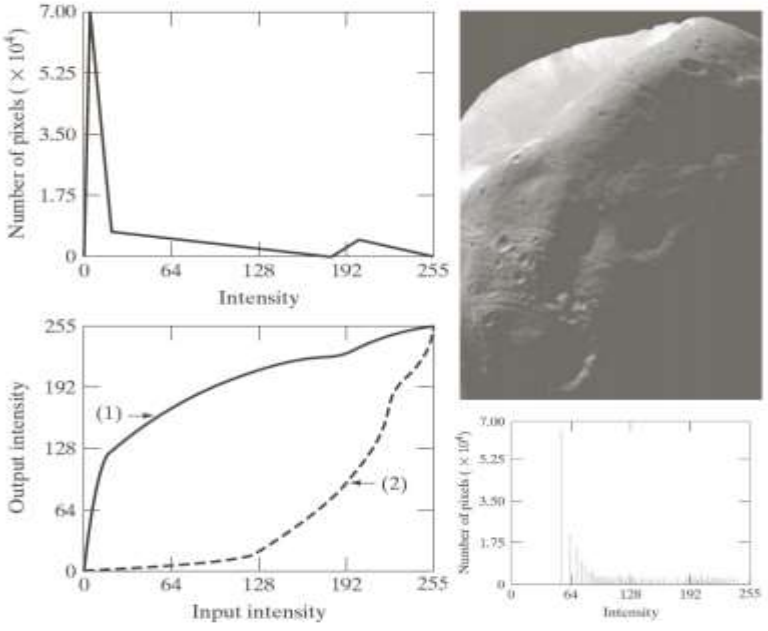
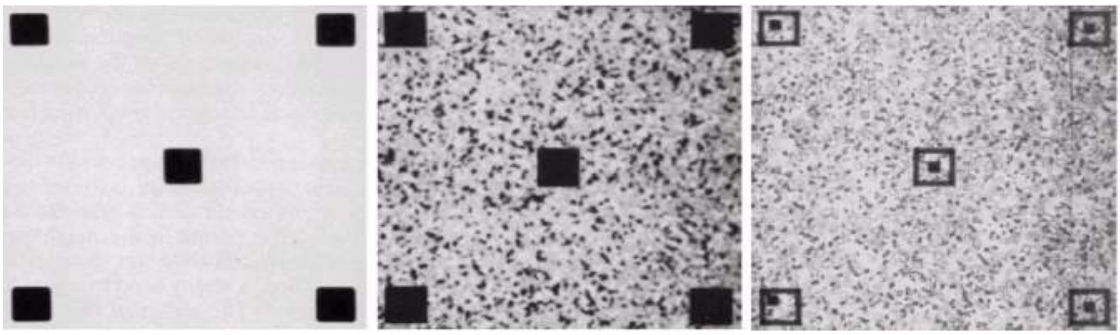


Fig: (a) specified histogram (b) Curve (1) is from equation $s_k = T(r_k) = \sum_{j=0}^{k} P_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$ **using the histogram in (a). curve (2) was obtained using iterative procedure in (c) enhanced image using mapping from curve (2) (d) Histogram of (c).**

Local histogram equalisation

Global histogram equalisation is suitable for overall enhancement. It is often necessary to enhance details over small areas.

The procedure is to define a square or rectangular neighbourhood and move the centre of this area from pixel to pixel. At each location the histogram of the points in the neighbourhood is computed and a histogram equalisation transformation function is obtained. This function is finally used to map the grey level of the pixel centred in the neighbourhood. The centre of the neighbourhood region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighbourhood changes during a pixel-to-pixel translation of the region, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible quite easily. This approach has obvious advantages over repeatedly computing the histogram over all pixels in the neighbourhood region each time the region is moved one pixel location. Another approach often used to reduce computation is to utilise non overlapping regions, but this methods usually produces an undesirable checkerboard effect.



a b c

FIGURE 3.23 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a 7×7 neighborhood about each pixel.

Some statistical indices can be easily computed from the histogram:

- Mean (average):
  - $m = \sum_{i=0}^{L-1} r_i p(r_i)$
- Variance:
  - $\sigma^2 = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i)$
  - Standard deviation: $\sigma = \sqrt{\sigma^2}$
- n-th moment:
  - $\mu_n = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$

Let (x, y) be the coordinates of a pixel in an image, and let *S*xy denote a neighborhood (subimage) of specified size, centered at (x, y).The mean value of the pixels in *S*xy can be computed using the expression

- $m_{S_{xy}} = \sum_{i=0}^{L-1} r_i p_{S_{xy}}(r_i)$

where $r_{s,t}$ is the gray level at coordinates (s,t) in the neighborhood, and $p(r_{s,t})$ is the neighborhood normalized histogram component corresponding to that value of gray level. the gray-level variance of the pixels in region *S*xy is given by

- $\sigma^2_{S_{xy}} = \sum_{i=0}^{L-1} (r_i - m_{S_{xy}})^2 p_{S_{xy}}(r_i)$

The local mean is a measure of average grey level in neughborhood S_{xy}, and the varience (or standard deviation) is a measure of contrast in that neighbourhood.

FIGURE 3.24 SEM image of a tungsten filament and support, magnified approximately 130×. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene).

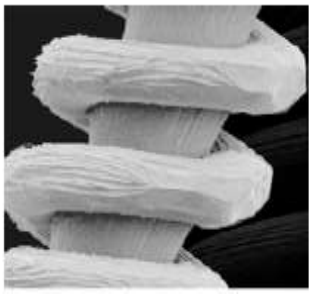


Figure 3.24 shows an SEM (scanning electron microscope) image of a tungsten filament wrapped around a support. The filament in the center of the image and its support are quite clear and easy to study. There is another filament structure on the right side of the image, but it is much darker and its size and other features are not as easily discernable.

In this particular case, the problem is to enhance dark areas while leaving the light area as unchanged as possible since it does not require enhancement. A measure of whether an area is relatively light or dark at a point (x, y) is to compare the local average gray level $m_{s_{xy}}$, to the average image gray level, called the global mean and denoted M_G. Thus, we have the first element of our enhancement scheme: We will consider the pixel at a point (x, y) as a candidate for processing if $m_{s_{xy}} \le k_0 M_G$ where k_0 is a positive constant with value less than 1.0. Since we are interested in enhancing areas that have low contrast. We also need a measure to determine whether the contrast of an area makes it a candidate for enhancement. Thus, we will consider the pixel at a point (x, y) as a candidate for enhancement if $\sigma_{s_{xy}} \le k_2 D_G$ where D_G is the global standard deviation and k_2 is a positive constant. The value of this constant will be greater than 1.0 if we are interested in enhancing light areas and less than 1.0 for dark areas. Finally, we need to restrict the lowest values of contrast we are willing to accept, otherwise the procedure would attempt to enhance even constant areas, whose standard deviation is zero. Thus, we also set a lower limit on the local standard deviation by requiring that $k_1 D_G \le \sigma_{S_{xy}}$ with k1<k2. A pixel at (x, y) that meets all the conditions for local enhancement is processed simply by multiplying it

by a specified constant, *E*, to increase (or decrease) the value of its gray level relative to the rest of the image. The values of pixels that do not meet the enhancement conditions are left unchanged. A summary of the enhancement method is as follows. Let f(x, y) represent the value of an image pixel at any image coordinates (x, y), and let g(x, y) represent the corresponding enhanced pixel at those coordinates.Then

$$g(x,y) = \begin{cases} E.f(x,y) \\ f(x,y) \end{cases} \text{if } m_{s_{xy}} \le k_0 M_G \text{ and } k_1 D_G \le \sigma_{S_{xy}} \le k_2 D_G$$

where, as indicated previously *E*, k0 , k1 , and k2 are specified parameters; M_G is the global mean of the input image; and D_G is its global standard deviation.

a b c

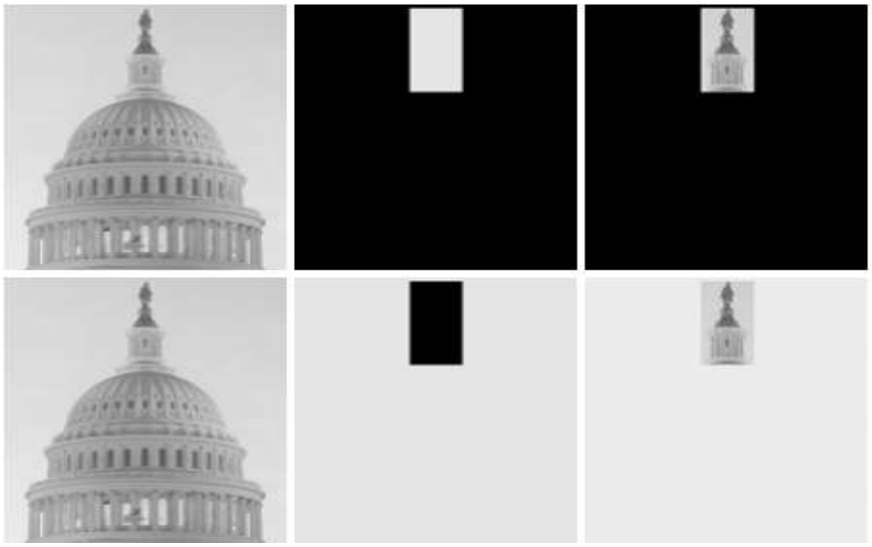
FIGURE 3.25 (a) Image formed from all local means obtained from Fig. 3.24 using Eq. (3.3-21). (b) Image formed from all local standard deviations obtained from Fig. 3.24 using Eq. (3.3-22). (c) Image formed from all multiplication constants used to produce the enhanced image shown in Fig. 3.26.

Figure 3.25(a) shows the values of $m_{s_{xy}}$ for all values of (x, y). Since the value of $m_{s_{xy}}$ for each (x, y) is the average of the neighboring pixels in a 3*3 area entered at (x, y), we expect the result to be similar to the original image, but slightly blurred. Figure 3.25(b) shows in image formed using all the values of $\sigma_{S_{xy}}$ Similarly, we can construct an image out the values that multiply f(x, y) at each coordinate pair (x, y) to form g(x, y). Since the values are either 1 or *E*, the image is binary, as shown in Fig. 3.25(c). The dark areas correspond to 1 and the light areas to *E*.

Enhancement using Arithmetic/ Logic Operation

Arithmetic/logic operations involving images are performed on a pixel-by-pixel basis between two or more images (this excludes the logic operation NOT, which is performed on a single image). As an example, subtraction of two images results in a new image whose pixel at coordinates (x, y) is the difference between the pixels in that same location in the two images being subtracted. Depending on the hardware and/or software being used, the actual mechanics of implementing arithmetic/logic operations can be done sequentially, one pixel at a time, or in parallel, where all operations are performed simultaneously.

Logic operations similarly operate on a pixel-by-pixel basis. We need only be concerned with the ability to implement the AND, OR, and NOT logic operators because these three operators are functionally complete. In other words, any other logic operator can be implemented by using only these three basic functions. When dealing with logic operations on gray-scale images, pixel values are processed as strings of binary numbers. For example, performing the NOT operation on a black, 8-bit pixel (a string of eight 0's) produces a white pixel Intermediate values are processed the same way, changing all 1's to 0's and vice versa. Thus, the NOT logic operator performs the same function as the negative transformation. The AND and OR operations are used for masking; that is, for selecting subimages in an image, as illustrated in Fig. 3.27. In the AND and OR image masks, light represents a binary 1 and dark represents a binary 0. Masking sometimes is referred to as region of interest(ROI) processing. In terms of enhancement, masking is used primarily to isolate an area for processing. This is done to highlight that area and differentiate it from the rest of the image.



a b c
d e f

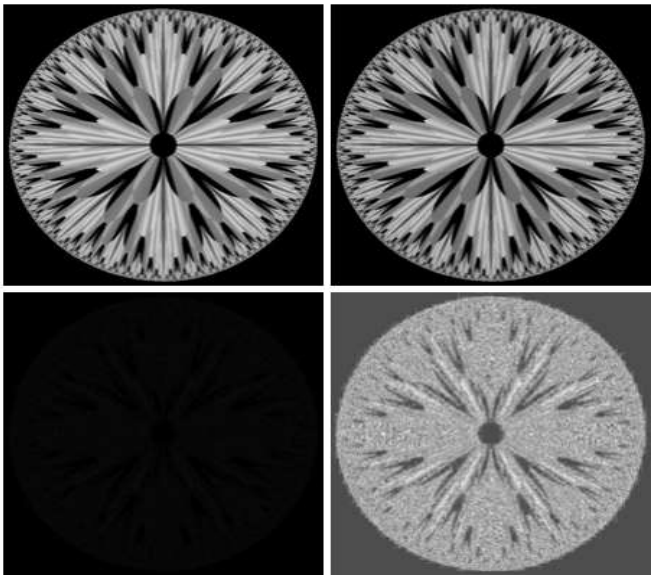
FIGURE 3.27
(a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

Spatial domain: Enhancement in the case of many realisations of an image of interest available

Image Subtraction

The difference between two image *f(x,y)* and *h(x,y)*, expressed as $g(x,y) = f(x,y) - h(x,y)$ is obtained by computing the difference between all pairs of corresponding pixels from f and h. The key usefulness of subtraction is the enhancement of differences between images. Figure 3.28(a) shows the fractal image used earlier to illustrate the concept of bit planes. Figure 3.28(b) shows the result of discarding (setting to zero) the four least significant bit planes of the original image. The images are nearly identical visually, with the exception of a very slight drop in overall contrast due to less variability of the graylevel values in the image of Fig. 3.28(b). The pixel-by-pixel difference between these two images is shown in Fig. 3.28(c). The differences in pixel values are so small that the difference image appears nearly black when displayed on an 8-

bit display. In order to bring out more detail, we can perform a contrast stretching transformation. We chose histogram equalization, but an appropriate power-law transformation would have done the job also. The result is shown in Fig. 3.28(d).This is a very useful image for evaluating the effect of setting to zero the lower-order planes.



a b
c d

FIGURE 3.28
(a) Original fractal image. (b) Result of setting the four lower-order bit planes to zero. (c) Difference between (a) and (b). (d) Histogram-equalized difference image. (Original image courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA).

Image averaging

Image averaging is obtained by finding the average of K images. It is applied in de-noising the images.

Suppose that we have an image $f(x,y)$ of size $M \times N$ pixels corrupted by noise $n(x,y)$, so we obtain a noisy image as follows.

$$g(x,y) = f(x,y) + n(x,y)$$

Where noise has zero average value

If an image $\bar{g}(x,y)$ is formed by averaging K different noisy image

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_{i(x,y)}$$

Then it follows that

$$E\{\bar{g}(x,y)\} = f(x,y)$$

and

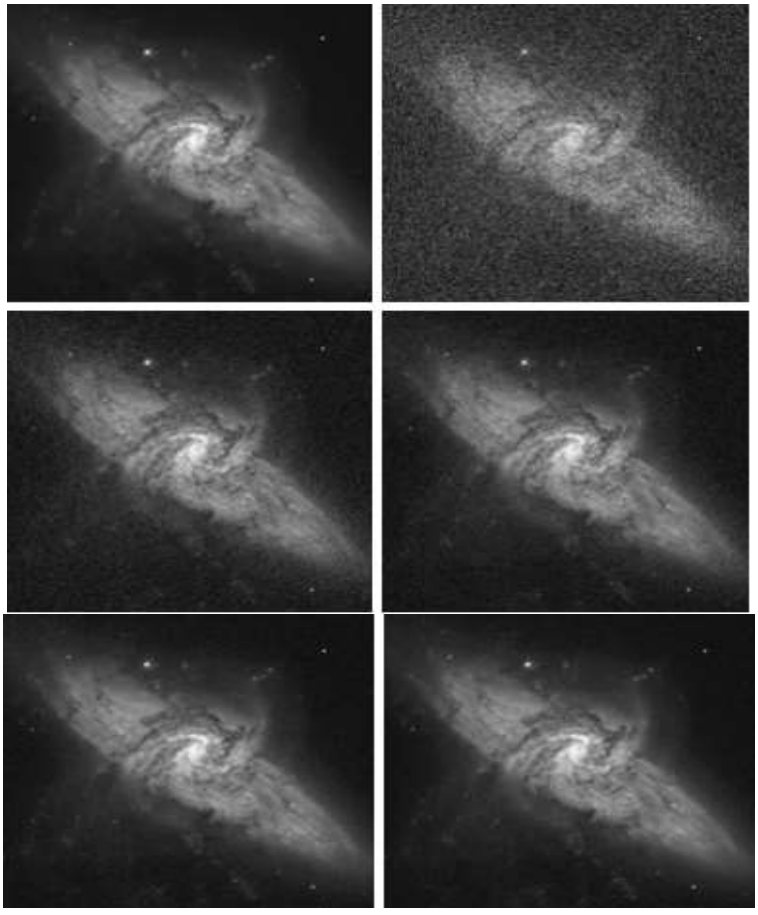
$$\sigma^{2}_{\bar{g}(x,y)} = \frac{1}{K}\sigma^{2}_{\eta(x,y)}$$

Where $E\{\bar{g}(x,y)\}$ is the expected value of $\bar{g}$ and $\sigma^{2}_{\bar{g}(x,y)}$ and $\sigma^{2}_{\eta(x,y)}$ are the variance of $\bar{g}$ and n, all at coordinates (x, y). The standard deviation at any point in the average image is

$$\sigma_{\bar{g}(x,y)} = \frac{1}{K}\sigma_{\eta(x,y)}$$

$E\{\bar{g}(x,y)\} = f(x,y)$ means that $\bar{g}(x,y)$ (output)approaches $f(x,y)$ (input) as the number of noisy images used in the averaging process increases.

An important application of image averaging is in the field of astronomy, where imaging with very low light levels is routine, causing sensor noise frequently to render single images virtually useless for analysis. Figure 3.30(a) shows an image of a galaxy pair called NGC 3314, taken by NASA's Hubble Space Telescope with a wide field planetary camera. . Figure 3.30(b) shows the same image, but corrupted by uncorrelated Gaussian noise with zero mean and a standard deviation of 64 gray levels. This image is useless for all practical purposes. Figures 3.30(c) through (f) show the results of averaging 8, 16, 64, and 128 images, respectively. We see that the result obtained with K=128 is reasonably close to the original in visual appearance



a b
c d
e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)