Unit II

Image Enhancement

Enhancement in Spatial Domain: basic gray level transformations, histogram processing, equalization, Arithmetic and logical operations between images, Basics of spatial filtering, smoothening and sharpening spatial filters, Image Enhancement in frequency Domain: smoothening and sharpening frequency domain filters, Fundamental of color image processing: color models, RGB, CMY, YIQ, HIS, Pseudo Color Image processing: Intensity filtering, gray level to color transformation, Basics of full color image processing.

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term 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 an image. Enhancing an image provides better contrast and a more detailed image as compare to non enhanced image. Image enhancement has very good applications. It is used to enhance medical images, images captured in remote sensing, images from satellite e.t.c. As indicated previously, the term spatial domain refers to the aggregate of pixels composing an image.

Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression.

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

where f(x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y), as Fig. 2.1 shows. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood.

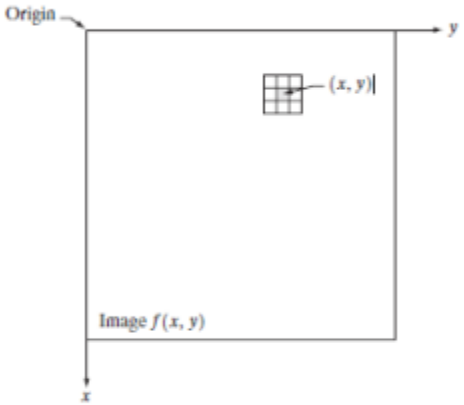


Fig.: 3x3 neighborhood about a point (x,y) in an image.

The simplest form of T is when the neighborhood is of size 1*1 (that is, a single pixel). In this case, g depends only on the value of f at (x, y), and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

$s = T(r)$

where r is the pixels of the input image and s is the pixels of the output image. T is a transformation function that maps each value of „r“ to each value of „s“.

For example, if T(r) has the form shown in Fig. 2.2(a), the effect of this transformation would be 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. In this technique, known as contrast stretching, the values of r below m are compressed by the transformation function into a narrow range of s, toward black. The opposite effect takes place for values of r above m.

In the limiting case shown in Fig. 2.2(b), T(r) produces a two-level (binary) image. A mapping of this form is called a thresholding function. One of the principal approaches in this formulation is based on the use of so-called masks (also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3*3) 2-D array, such as the one shown in Fig. 2.1, 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.

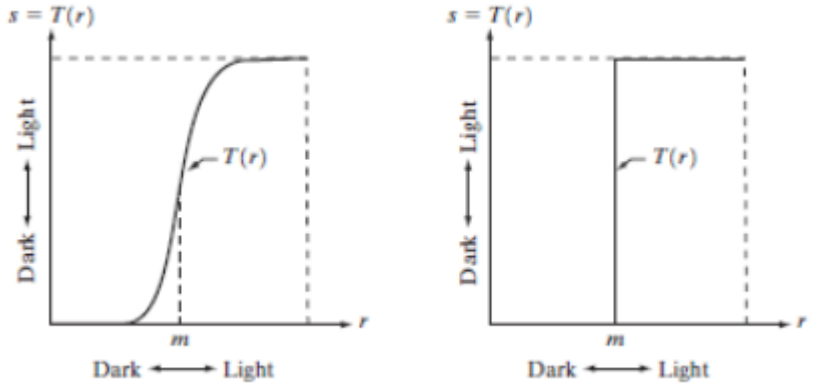


Fig. 2.2 Gray level transformation functions for contrast enhancement.

Image enhancement can be done through gray level transformations which are discussed below.

BASIC GRAY LEVEL TRANSFORMATIONS:

- Image negative
- Log transformations
- Power law transformations
- Piecewise-Linear transformation functions

LINEAR TRANSFORMATION:

First we will look at the linear transformation. Linear transformation includes simple

identity and negative transformation. Identity transformation has been discussed in our tutorial of image transformation, but a brief description of this transformation has been given here.

Identity transition is shown by a straight line. In this transition, each value of the input image is directly mapped to each other value of output image. That results in the same input image and output image. And hence is called identity transformation. It has been shown below:

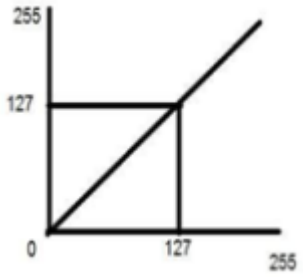


Fig. Linear transformation between input and output.

NEGATIVE TRANSFORMATION:

The second linear transformation is negative transformation, which is invert of identity transformation. In negative transformation, each value of the input image is subtracted from the L-1 and mapped onto the output image

IMAGE NEGATIVE: The image negative with gray level value in the range of [0, L-1] is obtained by negative transformation given by S = T(r) or

$$S = L - 1 - r$$

Where r= gray level value at pixel (x,y)

L is the largest gray level consists in the image

It results in getting photograph negative. It is useful when for enhancing white details embedded in dark regions of the image.

The overall graph of these transitions has been shown below.

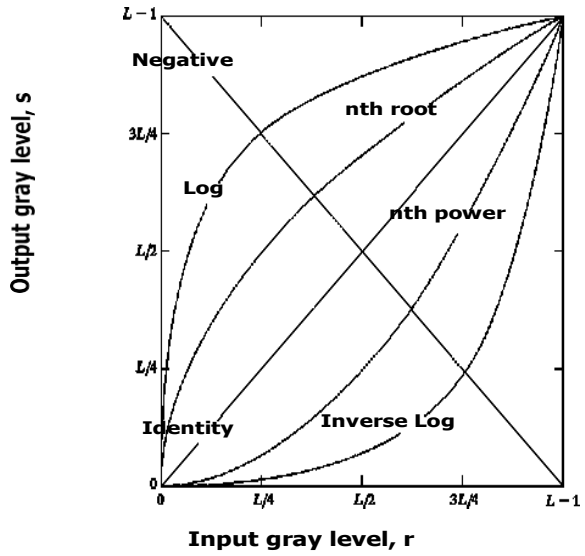


Fig. Some basic gray-level transformation functions used for image enhancement.

In this case the following transition has been done.

$$s = (L - 1) - r$$

since the input image of Einstein is an 8 bpp image, so the number of levels in this image are 256. Putting 256 in the equation, we get this

$$s = 255 - r$$

So each value is subtracted by 255 and the result image has been shown above. So what happens is that, the lighter pixels become dark and the darker picture becomes light. And it results in image negative.

It has been shown in the graph below.

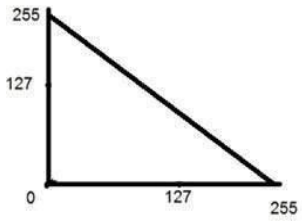


Fig. Negative transformations.

LOGARITHMIC TRANSFORMATIONS:

Logarithmic transformation further contains two type of transformation. Log transformation and inverse log transformation.

LOG TRANSFORMATIONS:

The log transformations can be defined by this formula

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

Where s and r are the pixel values of the output and the input image and c is a constant. The value 1 is added to each of the pixel value of the input image because if there is a pixel intensity of 0 in the image, then log (0) is equal to infinity. So 1 is added, to make the minimum value at least 1.

During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation. This result in following image enhancement.

An another way of representing LOG TRANSFORMATIONS: Enhance details in the darker regions of an image at the expense of detail in brighter regions.

$T(f) = C * \log (1+r)$

- Here C is constant and $r \geq 0$.

- The shape of the curve shows that this transformation maps the narrow range of low gray level values in the input image into a wider range of output image.
- The opposite is true for high level values of input image.

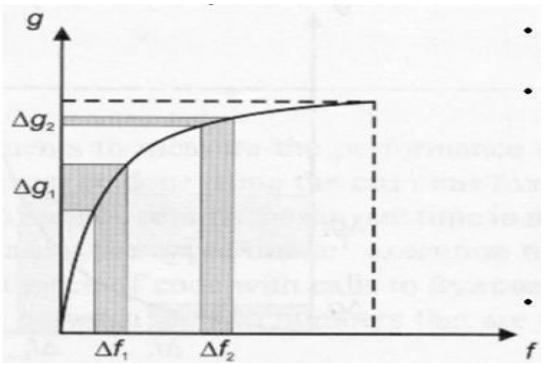


Fig. log transformation curve input vs output

POWER – LAW TRANSFORMATIONS:

There are further two transformation is power law transformations, that include nth power and nth root transformation. These transformations can be given by the expression:

$s=cr^{\gamma}$

This symbol γ is called gamma, due to which this transformation is also known as gamma transformation.

Variation in the value of γ varies the enhancement of the images. Different display devices / monitors have their own gamma correction, that's why they display their image at different intensity.

where c and g are positive constants. Sometimes Eq. (6) is written as $S = C (r +\varepsilon)^{\gamma}$ to account for an offset (that is, a measurable output when the input is zero). Plots of s versus r for various values of γ are shown in Fig. 2.10. As in the case of the log transformation, power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ.

In Fig that curves generated with values of $\gamma>1$ have exactly The opposite effect as those generated with values of $\gamma<1$. Finally, we Note that Eq. (6) reduces to the identity transformation when $c=\gamma=1$.

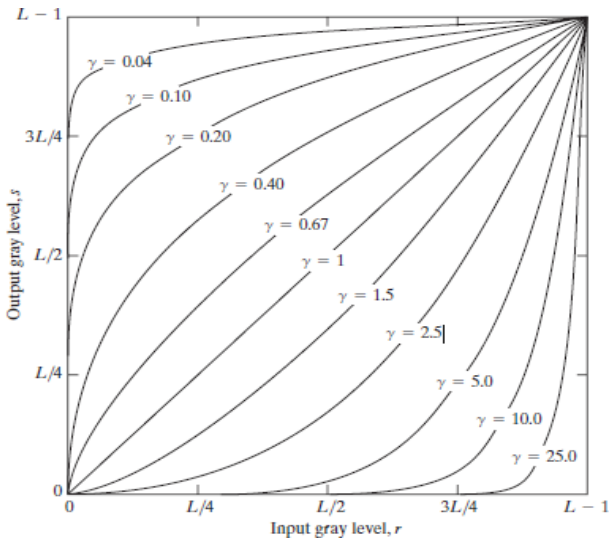


Fig. 2.13 Plot of the equation $S = cr^{\gamma}$ for various values of γ (c =1 in all cases).

This type of transformation is used for enhancing images for different type of display devices. The gamma of different display devices is different. For example Gamma of CRT lies in between of 1.8 to 2.5, that means the image displayed on CRT is dark.

Varying gamma (γ) obtains family of possible transformation curves $S = C^* r^{\gamma}$

Here C and γ are positive constants. Plot of S versus r for various values of γ is $\gamma > 1$ compresses dark values

Expands bright values

$\gamma < 1$ (similar to Log transformation) Expands dark values Compresses bright values

When $C = \gamma = 1$, it reduces to identity transformation .

CORRECTING GAMMA:

$s=cr^{\gamma}$ $s=cr^{(1/2.5)}$

The same image but with different gamma values has been shown here.

Piecewise-Linear Transformation Functions:

A complementary approach to the methods discussed in the previous three sections is to use piecewise linear functions. The principal advantage of piecewise linear functions over the types of functions we have discussed thus far is that the form of piecewise functions can be arbitrarily complex.

The principal disadvantage of piecewise functions is that their specification requires

considerably more user input.

Contrast stretching: One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition.

S= T(r)

Figure x(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=0and s2= L-1, the transformation Becomes a thresholding function that creates a binary image, as illustrated In fig. 2.2(b).

Intermediate values of ar1, s1b and ar2, s2b 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 that the function is single valued and Monotonically increasing.

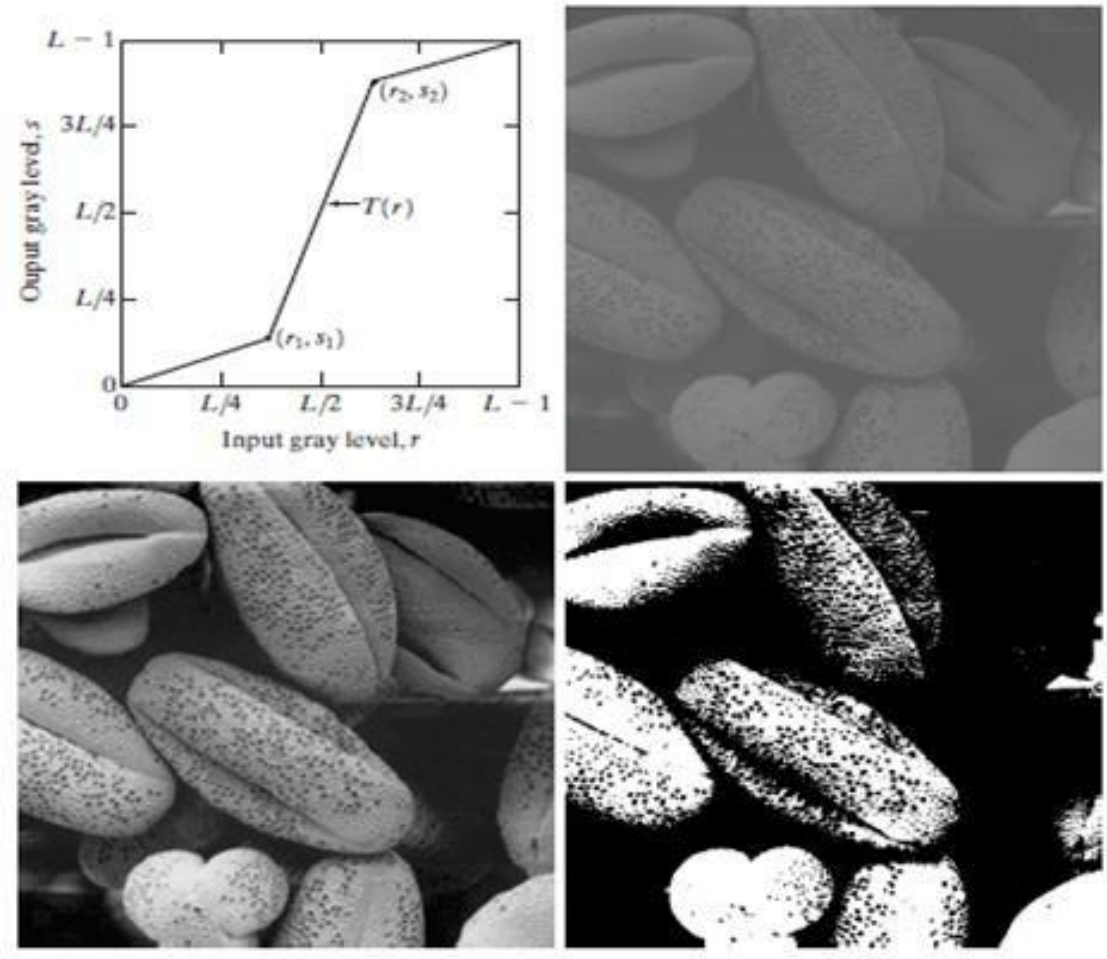


Fig. x Contrast stretching. (a) Form of transformation function. (b) A low-contrast stretching.

(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 x(b) shows an 8-bit image with low contrast. Fig. x(c) shows the result of contrast stretching, obtained by setting (r1, s1)=(rmin, 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. x(d) shows the result of using the thresholding function defined previously,

with r1=r2=m, the mean gray level in the image. The original image on which these results are based is a scanning electron microscope image of pollen, magnified approximately 700 times.

Gray-level slicing:

Highlighting a specific range of gray levels in an image often is desired. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images.

There are several ways of doing level slicing, but most of them are variations of two basic themes.One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels.

This transformation, shown in Fig. y(a), produces a binary image. The second approach, based on the transformation shown in Fig.y (b), brightens the desired range of gray levels but preserves the background and gray-level tonalities in the image. Figure y (c) shows a gray-scale image, and Fig. y(d) shows the result of using the transformation in Fig. y(a).Variations of the two transformations shown in Fig. are easy to formulate.

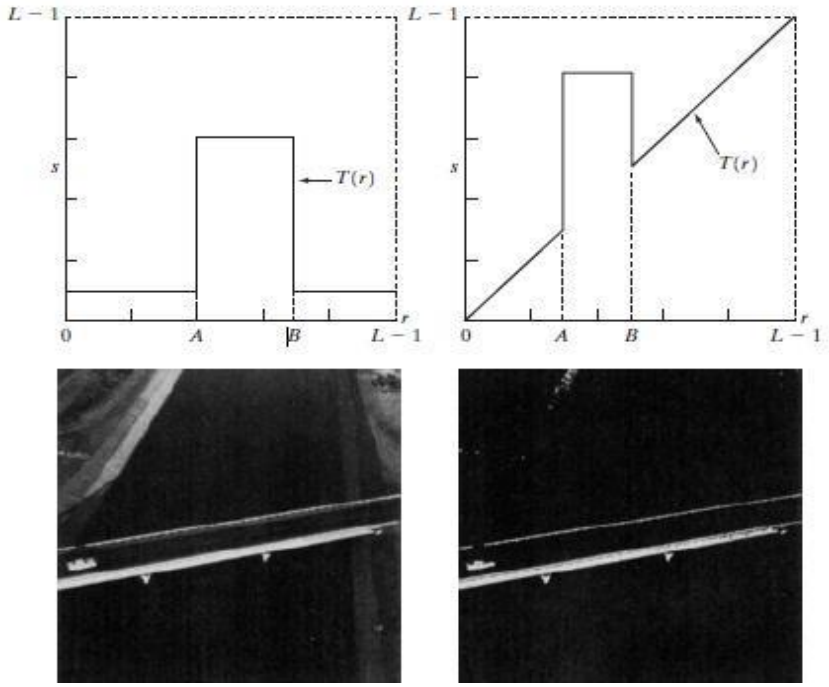


Fig. y (a)This transformation highlights range [A,B] of gray levels and reduces all others to a constant level (b) This transformation highlights range [A,B] but preserves all other levels.
(c) An image . (d) Result of using the transformation in (a).

BIT-PLANE SLICING:

Instead of highlighting gray-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits.

Figure 3.12 illustrates these ideas, and Fig. 3.14 shows the various bit planes for the image shown in Fig. 3.13. Note that the higher-order bits (especially the top four) contain the majority of the visually significant data.The other bit planes contribute to more subtle details in the image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel.

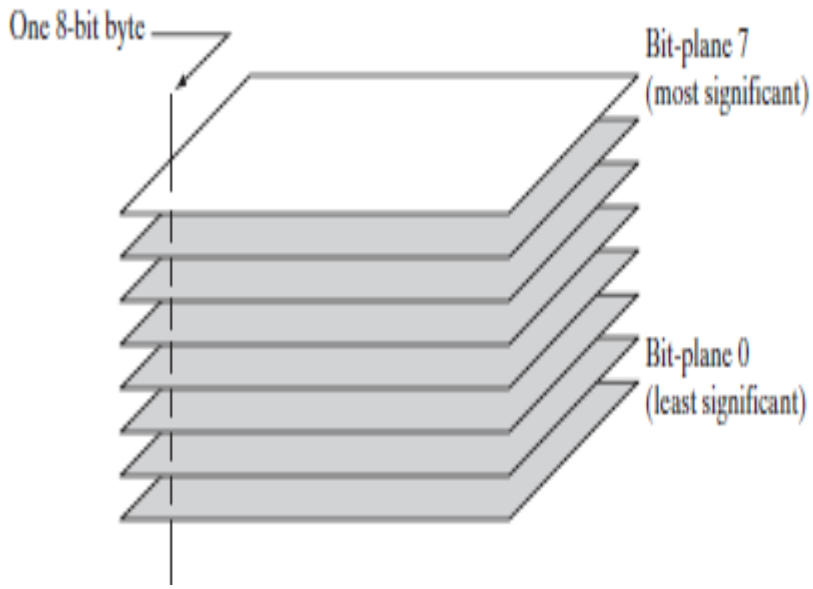


FIGURE Bit-plane representation of an 8-bit image.

In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to one level (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255).The binary image for bit-plane 7 in Fig. 3.14 was obtained in just this manner. It is left as an exercise (Problem 3.3) to obtain the gray-level transformation functions that would yield the other bit planes.

Histogram Processing:

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function of the form

H(rk)=nk

where rk is the kth gray level and nk is the number of pixels in the image having the level rk..

A normalized histogram is given by the equation

p(rk)=nk/n for k=0,1,2,…..,L-1

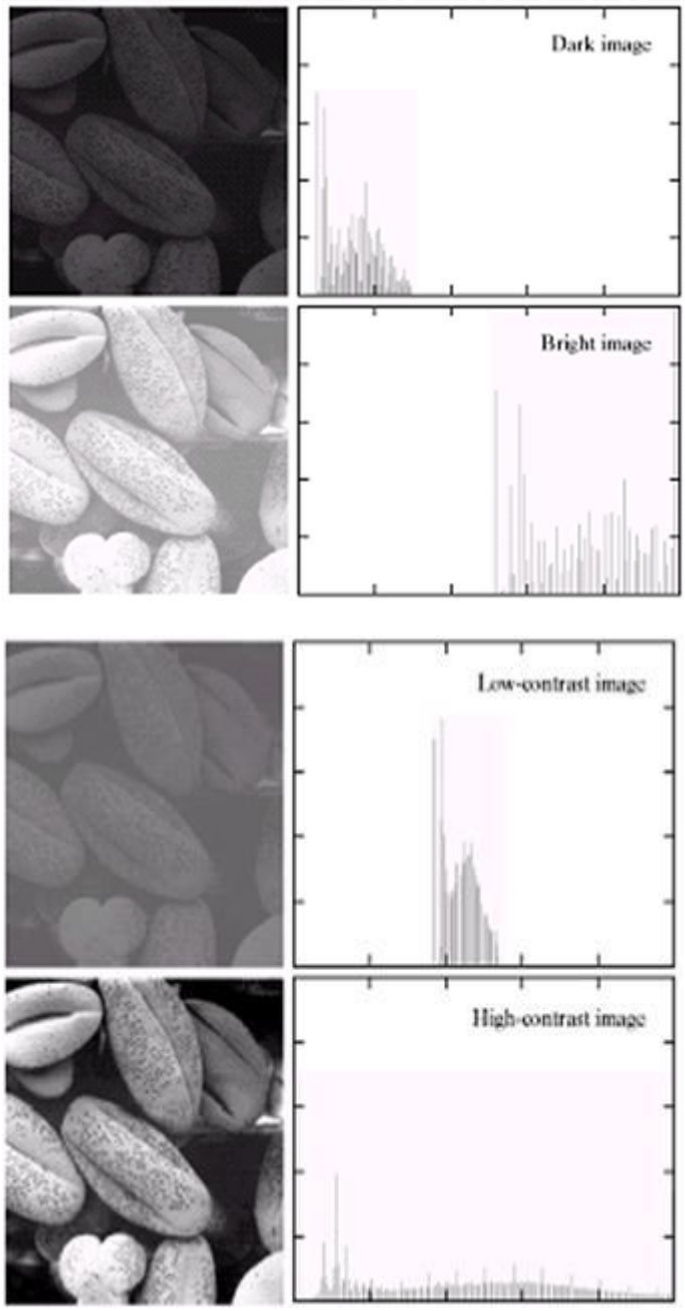
P(rk) gives the estimate of the probability of occurrence of gray level rk. The sum of all components of a normalized histogram is equal to 1.

The histogram plots are simple plots of H(rk)=nk versus rk.

In the dark image the components of the histogram are concentrated on the low (dark) side of the gray scale. In case of bright image the histogram components are baised towards the high side of

the gray scale. The histogram of a low contrast image will be narrow and will be centered towards the middle of the gray scale.

The components of the histogram in the high contrast image cover a broad range of the gray scale. The net effect of this will be an image that shows a great deal of gray levels details and has high dynamic range.

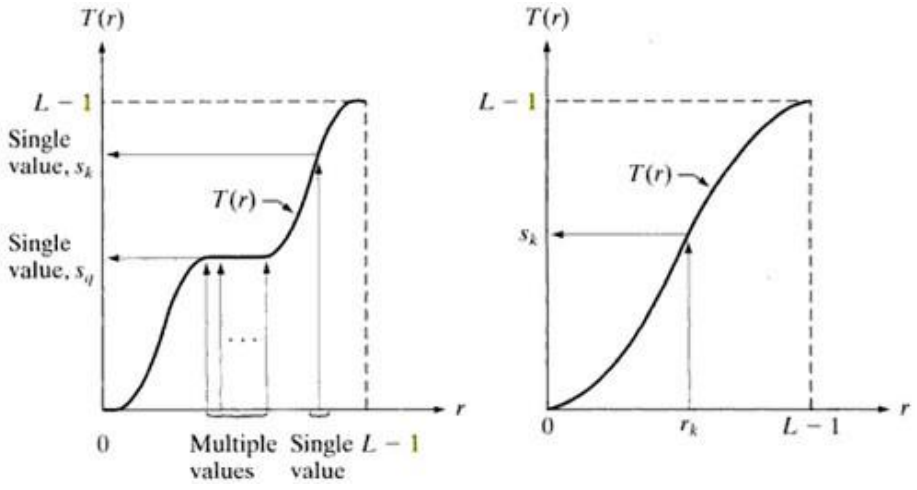


Histogram Equalization:

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail are compressed into the dark end of the histogram. If we could „stretch out" the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Let there be a continuous function with r being gray levels of the image to be enhanced. The range of r is [0, 1] with r=0 repressing black and r=1 representing white. The transformation function is of the form S=T(r) where 0<r<1

It produces a level s for every pixel value r in the original image.



a b

FIGURE
(a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

The transformation function is assumed to fulfill two condition T(r) is single valued and monotonically increasing in the internal 0<T(r)<1 for 0<r<1.The transformation function should be single valued so that the inverse transformations should exist. Monotonically increasing condition preserves the increasing order from black to white in the output image. The second conditions guarantee that the output gray levels will be in the same range as the input levels. The gray levels of the image may be viewed as random variables in the interval [0.1]. The most fundamental descriptor of a random variable is its probability density function (PDF) Pr(r) and Ps(s) denote the probability density functions of random variables r and s respectively. Basic results from an elementary probability theory states that if Pr(r) and Tr are known and T-1(s) satisfies conditions (a), then the probability density function Ps(s) of the transformed variable is given by the formula

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

Thus the PDF of the transformed variable s is the determined by the gray levels PDF of the input image and by the chosen transformations function.

A transformation function of a particular importance in image processing

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

This is the cumulative distribution function of r. L is the total number of possible gray levels in the image.

For example : Data given is as shown below,

Gray level	0	1	2	3	4	5	6	7
Frequency	400	700	1350	2500	3000	1500	550	0

$n_k = 400 + 700 + 1350 + 2500 + 3000 + 1500 + 550 + 0$

$n = 10{,}000$

Number of gray levels, $L = 8$

Gray level	n_k	PDF $= \frac{n_k}{n}$	CDF $= \sum PDF \frac{(n_k)}{n}$	(L–1) CDF	Rounding off
0	400	$\frac{400}{10000} = 0.04$	0.04	0.28	0
1	700	0.07	0.11	0.77	1
2	1350	0.135	0.245	1.715	2
3	2500	0.25	0.495	3.465	3
4	3000	0.3	0.795	5.565	6
5	1500	0.15	0.945	6.615	7
6	550	0.055	1.000	7	7
7	0	0	1.000	7	7

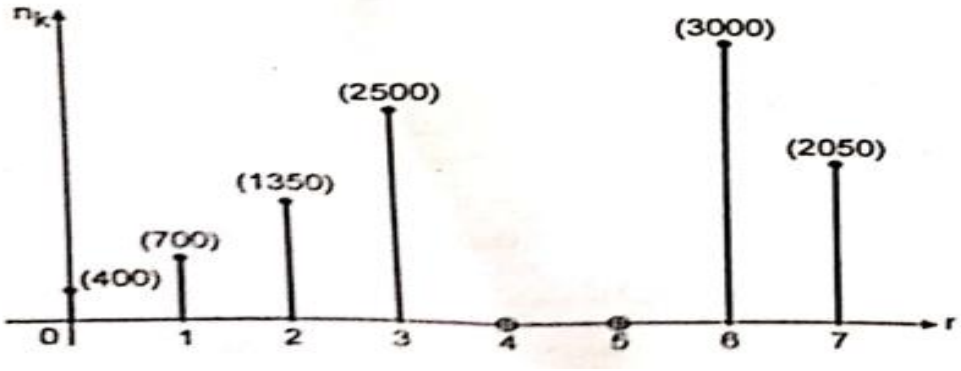


Image Difference :

whenever we go for image differencing that means we have to take the difference of pixel values between 2 images.

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

this operation suggests that g (x, y) in g (x, y), all those pixel locations will be highlighted wherever there is a difference between the corresponding locations in f (x, y) and the corresponding location in h (x, y). Wherever f (x, y) and g (x, y) are same the correspond f (x, y) and h (x, y) are same, the corresponding pixel in g (x, y) will have a value which is near to 0.

So, this kind of operation image, differencing operation mainly highlights the difference between 2 images or the locations where the 2 image contents are different. Such a kind of image difference operations is very useful particularly in medical image processing .

So in case of medical image processing, there is operation which is called say mask mode radiography. In mask mode radiography, what is done is you take the image of certain body part of a patient, an x-ray image which is captured with the help of a tv camera where the camera is normally placed opposite to a x-ray shots and then what is done is you inject a contrast media into the blood stream of the patient and after injecting this contrast media, again you take a series of images using the same tv camera of the same anatomical portion of the patient body.

Spatial Filtering technique is used directly on pixels of an image. Mask is usually considered to be added in size so that it has a specific center pixel. This mask is moved on the image such that the center of the mask traverses all image pixels.

In this article, we are going to cover the following topics –

- To write a program in Python to implement spatial domain averaging filter and to observe its blurring effect on the image without using inbuilt functions
- To write a program in Python to implement spatial domain median filter to remove salt and pepper noise without using inbuilt functions

Theory

- **Neighborhood processing in spatial domain:** Here, to modify one pixel, we consider values of the immediate neighboring pixels also. For this purpose, 3X3, 5X5, or 7X7 neighborhood mask can be considered. An example of a 3X3 mask is shown below.

f(x-1, y-1)	f(x-1, y)	f(x-1, y+1)
f(x, y-1)	f(x, y)	f(x, y+1)
f(x+1, y-1)	f(x+1, y)	f(x+1, y+1)

- **Low Pass filtering:** It is also known as the smoothing filter. It removes the high-frequency content from the image. It is also used to blur an image. A low pass averaging filter mask is as shown.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- **High Pass Filtering:** It eliminates low-frequency regions while retaining or enhancing the high-frequency components. A high pass filtering mask is as shown.

-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9

- **Median Filtering:** It is also known as nonlinear filtering. It is used to eliminate salt and pepper noise. Here the pixel value is replaced by the median value of the neighboring pixel.