Image Enhancement in the Spatial Domain.

Module - II

Introduction :

The objective of image Enhancement is to improve the interpretability of the information present in images for human viewers. An Enhancement algorithm is one that yields a better-quality image for the purpose of some particular applications, which can be either removing noise & image Contrast to improve the Quality of dark image & Edges are highlighted for some applications. Image-Enhancement techniques can be classified into 2 categories as:

(1) Spatial domain method & (2) Frequency domain method.

Spatial Domain Techniques:

Spatial domain works on image plane itself and direct manipulation of pixels in an image is done.

Frequency & Transform domain Technique: works on the fourier transform of an image & then transforms it back to the spatial domain.

→ Spatial domain tech^n are more Efficient Computationally & require less processing resources to implement.

→ Spatial domain can be denoted by the Expression:

$$g(x,y) = T[f(x,y)] \quad \text{----->} \quad \text{Eq}^n \text{ (1)}$$

where,

$f(x,y)$ is the input image

$g(x,y)$ is the output & processed image.

T → transformation function or T is an operator on "f" defined over a neighborhood of point (x,y).

→ T - Transform operator works on a single image or to a set of images, such as performing the pixel-by-pixel sum of a sequence of images for noise reduction.
→ The operator "T" is applied at each location (x,y) to yield the output "g" at that location.
→ The neighborhood of a point (x,y) can be square or rectangular subimage area centered at (x,y).

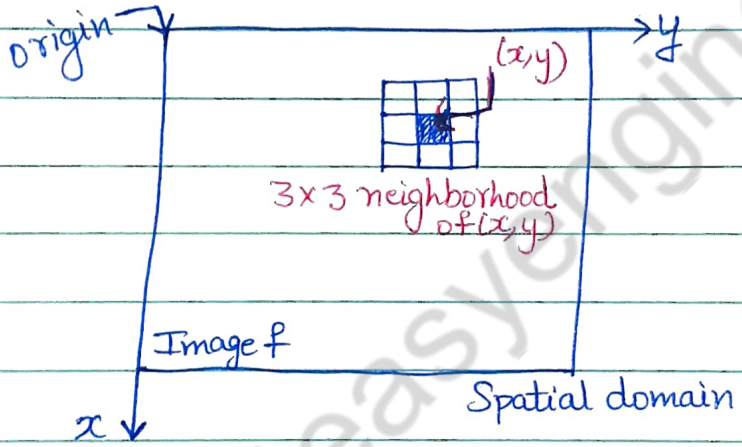


Fig 3.1 : A 3x3 neighborhood about a point (x,y) in an image in the Spatial domain

→ Consider a rectangular image f(x,y) as shown in fig 3.1 The point (x,y) shown is an arbitrary location in the image & the small region shown containing the point is a neighbor-hood of (x,y).
→ The process utilizes only the pixels in the area of the image spanned by the neighborhood.
→ The process consists of moving the origin of the neighbor-hood from pixel to pixel & applying the operator "T" to the pixels in the neighborhood to yield the o/p at that location

→ Thus for any specific location (x,y), 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,y) in f.

→ The Smallest possible neighborhood is of Size 1×1 & it's the Simplest form of transformation.

→ In this case "g" depends only on the value of "f" at a Single point (x,y) and "T" becomes an intensity (also called gray-level) transformation function of the forms.

$$s = T(r) \quad \text{----->} \text{ Eq}^{n} (2)$$

where,

"s" & "r" are variables denoting respectively, the intensity of "g" & "f" at any point (x,y), of f(x,y)

& r → denotes the gray level of & pixel value in i/p image

s → denotes the gray level of g(x,y) at any point (x,y) & Corresponding pixel value in processed image.

Because Enhancement at any point in an image depends only on the gray level & intensity at that point for processing & do not Consider neighborhood to get a processed image. This Technique called as "point processing".

There are 2 kind of functions in gray level functioning.

(a) Contrast stretching: It produces an image of higher Contrast than the original one.

→ The operation* is performed by darkening the levels below "K" & brightening the levels above "K" in the Siginal image.

[* applying the transformation to Every pixel "f" to generate the Corresponding pixels in g would be to produce an image of higher Contrast]

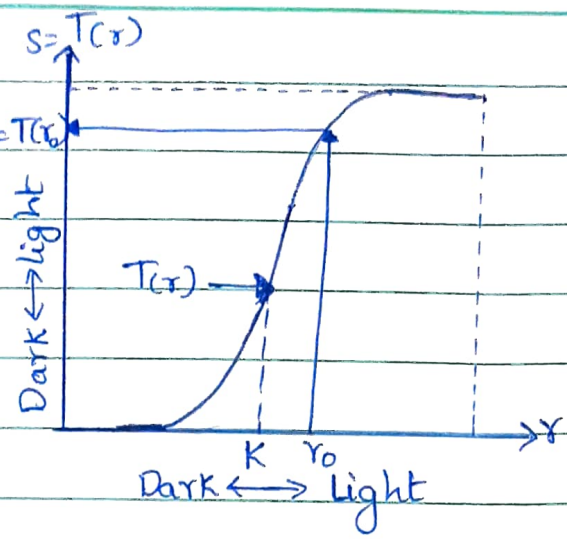


Fig 3.2 (a): Contrast Stretching

In this method, value of r lower than K are compressed by the transformation function into a narrow range of 'S' towards black. The opposite effect takes place for the values of r higher than K & value of r above K are transformed to brighter range.

Such a transformation is called as Contrast stretching. In fig 3.2 (a) observe how an intensity value r_0 is mapped to obtain the corresponding value S_0.

(b) Thresholding function: In this case, where T(r) produces a two-level (binary) image. All values of r below K are assigned & transformed as black & zero value & above K are assigned or transformed as white value. A mapping of this form is called a thresholding function.

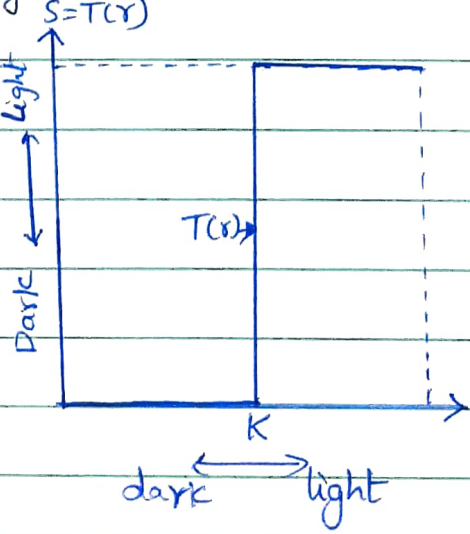
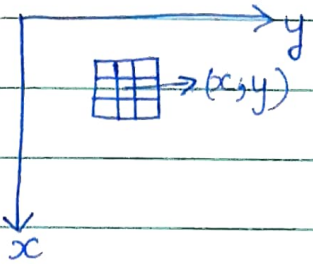


Fig 3.2 (b): Thresholding function

Mask operation : If neighborhood size is larger than 1 i.e 3×3, 5×5 or 7×7, it allows considerable flexibility. Mask operation not only considers the intensities at (x,y) but also considers intensities at neighborhood in deciding the corresponding pixel value of transformed image.



$w_{-1,-1}$	$w_{-1,0}$	$w_{-1,1}$
$w_{0,-1}$	$w_{0,0}$	$w_{0,1}$
$w_{1,-1}$	$w_{1,0}$	$w_{1,1}$

3×3

Fig 3.3. Mask operation of 3×3 size neighborhood of (x,y).

To generate corresponding pixel value of processed image following operation is performed.

$$g(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} w_{ij} f(x+i, y+j)$$

where g(x,y) gives the corresponding pixel value in processed image.

→ The coefficient "wij", determines what Enhancement operation is to be performed such as image Sharpening, image averaging, Edge Enhancement & so on.

→ Both point & Mask processing fall under Spatial Enhancement techniques as it operates on the pixel values & no fourier transform is taken.

Some Basic Intensity or Gray level Transformation functions:

Intensity transformations are among the simplest of all image processing techniques.

→ We denote the values of pixels, before & after processing by 'r' and 's', respectively. These values are related by a transformation 'T', as given in Eq^n ②.

$$s = T(r)$$

that maps a pixel value 'r' of an original (i/p) image into a pixel value 's' of a processed (o/p) image.

Note: Because we deal with digital quantities, values of an intensity transformation function are stored in a table & the mappings from 'r' to 's' are implemented via table lookups. For Eg: 8-bit image, a lookup table containing the values of 'T' will have 256 Entries.

⇒ The three basic types of functions used frequently for image Enhancement are:

① Linear (Negative & identity transformations)
② Logarithmic (log & inverse-log transformations)
③ power-law (n^{th} power & n^{th} root transformations)

as shown in figure 3.3.

→ The identity function is the trivial case in which o/p intensities are identical to i/p intensities.

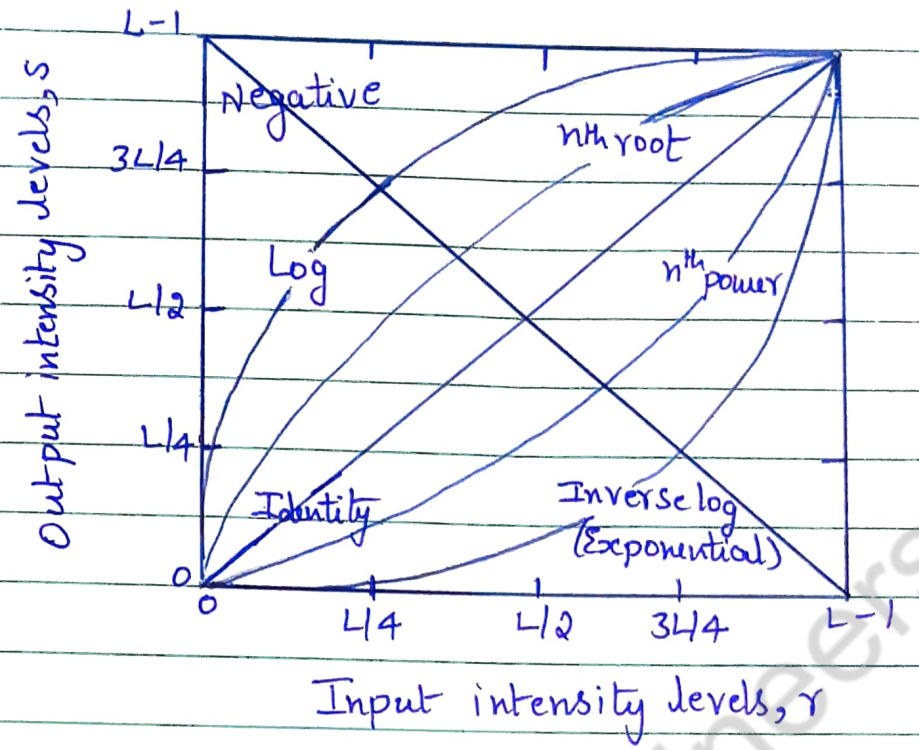


Fig 3.3. Some basic intensity transformation functions. Each curve was scaled independently so that all curves would fit in the same graph.

Image Negative: The negative of an image with intensity levels in the range $[0, L-1]$ is obtained using the negative transformation, which is given by the Expression or Eq^n (3). & Shown in fig 3.4.

$$s = L-1-r \quad \text{---->} \quad \text{Eq}^n (3)$$

or $S = T(r) = L-1-r$

where, $r = 0$, $S = L-1$ (Max value)

& when $r = L-1$, $S = 0$ (Min value).

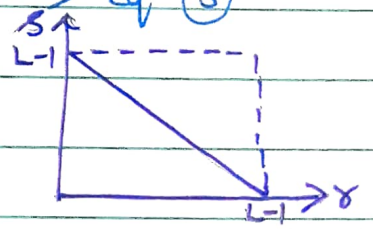


Fig 3.4(a): image Negative.

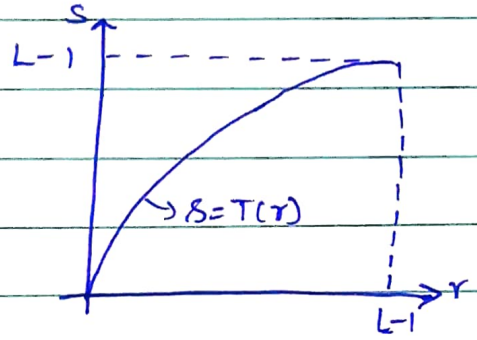
→ Reversing the intensity levels of an image in this way produces the Equivalent of a photographic negative.

→ This type of processing is used for Enhancing white or gray pixels Embedded in dark pixels of an image, Especially when the black areas are dominant in Size.

→ Thus max intensity value in original image is Converted to minimum intensity value in processed image & min intensity value is Converted to maximum intensity value in processed image resulting in a negative of the image.

→ Negative transformation is very useful in medical applications, Eg in mammogram image where it's Easier to analyse the breast tissue to detect cancer.

Log Transformation:



$$S = T(r) = c \log (1+r) \quad \text{----> Eq" (4)}$$

where, c is Constant & it is assumed that $r \geq 0$.

Fig 3.4(b): log Transformation

The general form of the log transformation is shown Eq" (4)

→ we use this type of transformation 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.

→ The log function has the particular characteristic that it Compresses the dynamic range of images with large variations in pixel values.

→ Certain Applications requires reduction in dynamic range of the original image.

→ for instance if dynamic range is very high, display

device may not be able to properly reproduce the image. Normally, a gray level display device has intensity levels ranging from 0 to 255 (256 values). If original image has minimum intensity level of 0 & max. intensity level of few thousands, dynamic range of original image is very high & display device will not be able to handle such high intensity. display device displays high intensity values & low intensity values are suppressed.

This problem can be overcome by an image Enhancement tech^n called dynamic range Compression. This can be done using logarithmic transformation.

Power-Law (Gamma) Transformations: is normally used for different image capturing devices, image printers & so on.

power-law transformations have the basic form.

$$S = T(r) = cr^{\gamma} \quad \text{-------> Eq}^{n} (5)$$

where c & γ are positive constants.

plots of 's' versus 'r' for various values of γ for c=1 is shown in fig 3.4(c).

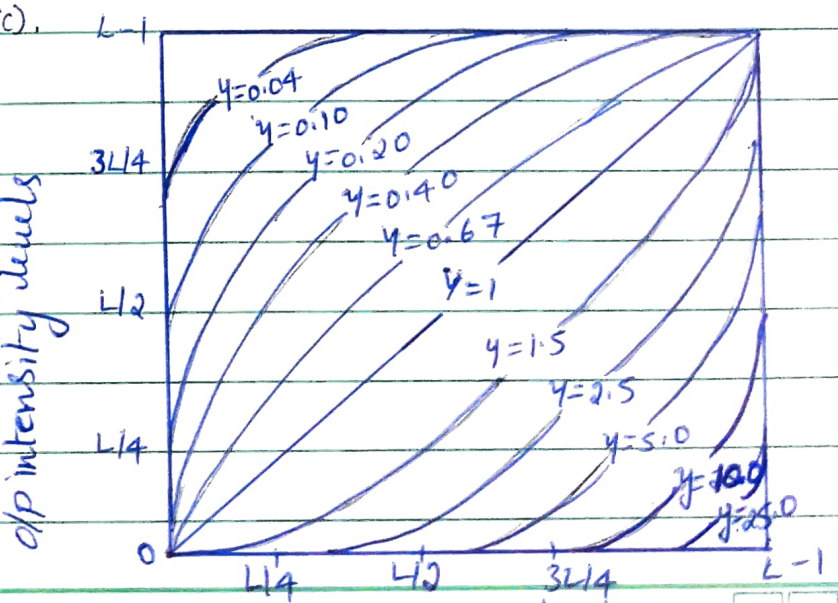


Fig 3.4(c): plots of the Eq^n $S = cr^{\gamma}$ for various values of γ.

→ power law curves with fractional values of γ map a narrow range of dark input values into a wider range of o/p values, with the opposite being true for higher values of input levels.

(or) For values of $\gamma < 1$, transformation map a narrow range of dark input a wider range of o/p values. & opposite is true for values $\gamma > 1$.

⇒ This process used to correct the power law response is exhibited by imaging devices called "gamma correction."

⇒ In gamma correction, image is converted to power law before image is actually produced to compensate for power law produced by imaging device itself to ensure that image is properly displayed.

For Eg: In a CRT display, relation between intensity to voltage follows power law with γ varying b/w 1.8 to 2.5 In figure 3.5, monitor follows power law transformation for $\gamma = 2.5$, producing a darker image as compared to actual image. To compensate for this, reverse corrective measure is used before image is given to CRT for display so that image is properly displayed.

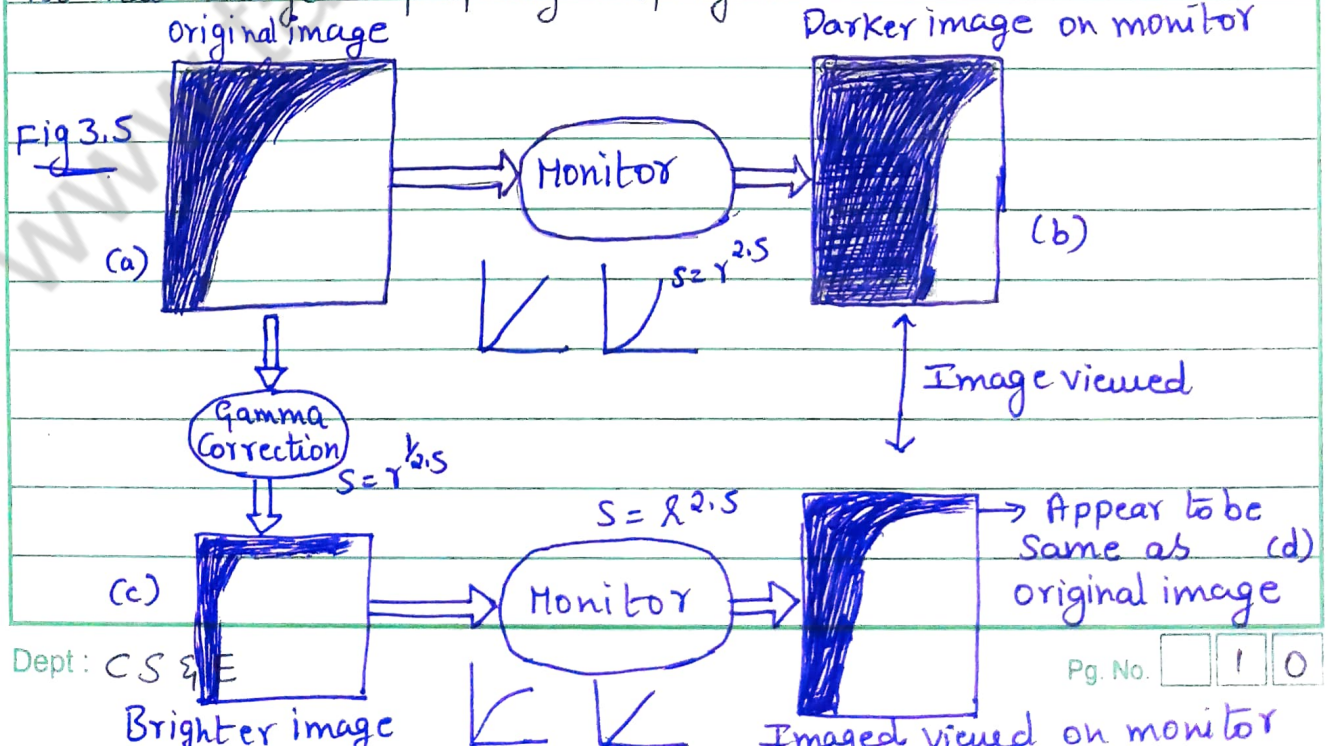


Fig 3.5: (a) Intensity ramp image (b) image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma corrected image. (d) Corrected image as viewed on the same monitor. [Compare (d) & (a)].

→ Gamma Correction is important if displaying an image accurately on a Clm screen. Images that are not corrected properly can look either bleached out or too dark.
It is important in general purpose contrast manipulation also.
→ To make an image black we use $\gamma > 1$ & $\gamma < 1$ for white image.

Piecewise - linear Transformation Functions :

The principal advantage of piecewise linear function is that these functions can be arbitrarily complex. But the disadvantage is that their specification requires considerably more user input.

Contrast stretching : it's the simplest piecewise linear transformation function.
→ we may have various low contrast images & that might result due to various reasons such as lack of illumination, problem in imaging sensor or wrong setting of lens aperture during image acquisition.
→ Contrast stretching is a process that expands ^the range of intensity levels in a image^ (or increase) (being processed) so that it spans the full intensity range of the recording medium or display device.
→ Contrast stretching transformations increase the contrast between the darks & the lights.

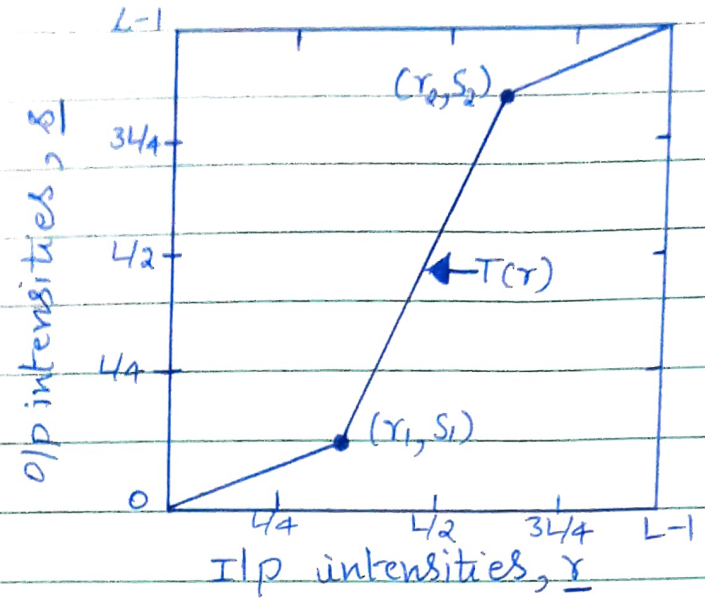


Fig 3.6: Contrast Stretching: piecewise linear transformation function.

Fig 3.6 shows a typical transformation used for Contrast Stretching. The locations of points (r1, S1) & (r2, S2) Control the shape of the transformation functions.

→ If $r_1 = S_1$ & $r_2 = S_2$ the transformation is a linear function that produces no changes in intensity.

→ If $r_1 = r_2$, $S_1 = 0$ & $S_2 = L-1$ the transformation becomes a thresholding function that creates a binary image.

→ Generally $r_1 \leq r_2$ & $S_1 \leq S_2$ so that the function is single valued & monotonically increasing.

Gray-level or Intensity Slicing:

Highlighting a specific range of intensities/gray level in an image is often desirable or interest.

For Eg, when Enhancing features such as masses of water in satellite imagery & Enhancing flaws in X-Ray images.

→ The process often called Gray/intensity-level Slicing can be implemented in 2 ways:

① one method is to display a high value for all gray/intensity value in the range of interest & a low value

for all other gray level. (or)

To display in one value (say, white) all the values in the range of interest & in another (say, black) all other intensities. This transformation is shown in figure 3.7 (a), produces a binary image.

② 2nd method is to brighten the desired ranges of gray levels but preserve the background & gray level intensities in the image. (or)

Brightens (or darkens) the desired range of intensities but leaves all other intensity levels in the image unchanged. This is shown in figure 3.7(b).

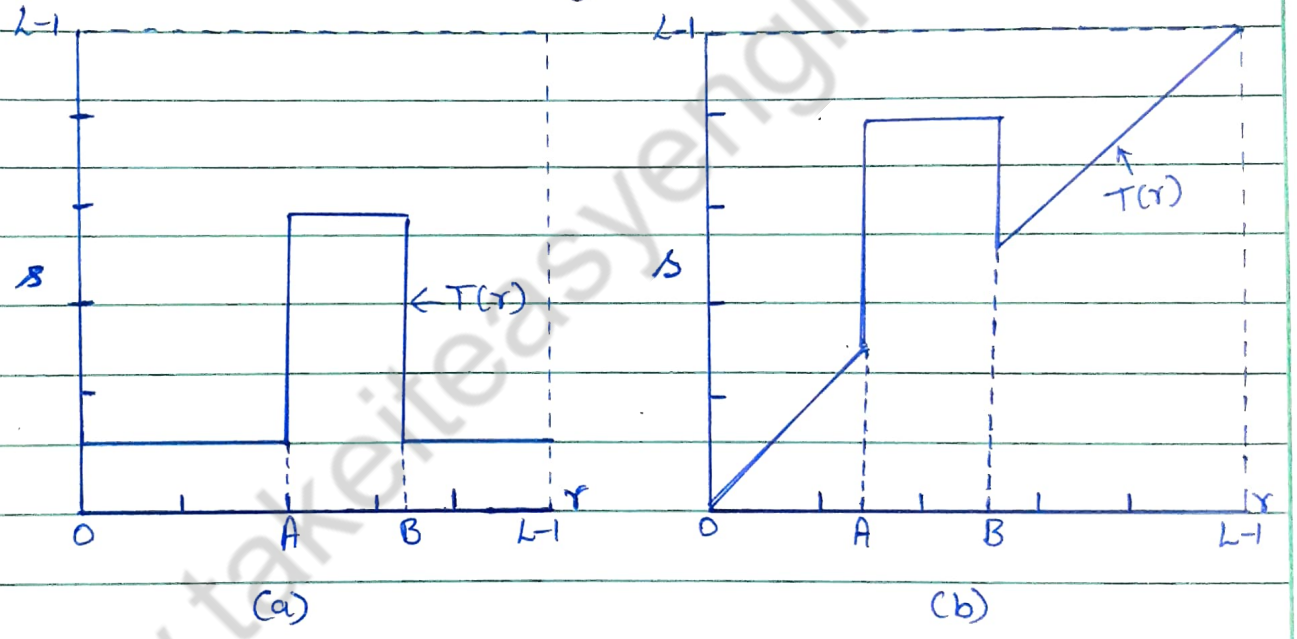


Fig 3.7: (a) This transformation fun^n highlights intensity range [A,B] & reduces all other intensities to a lower level. (b) This transformation fun^n highlights range [A,B] & preserves all other intensity levels.

Bit-plane Slicing: pixels are digital numbers composed of bits. Eg: the intensity of each pixel is a 256-level gray-scale image is composed of 8 bits (i.e one byte).

→ Instead of highlighting intensity-level ranges, we could highlight the contribution made to total image appearance by specific bits.

Figure 3.8 illustrates, 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 & plane 8 all the highest-order bits.

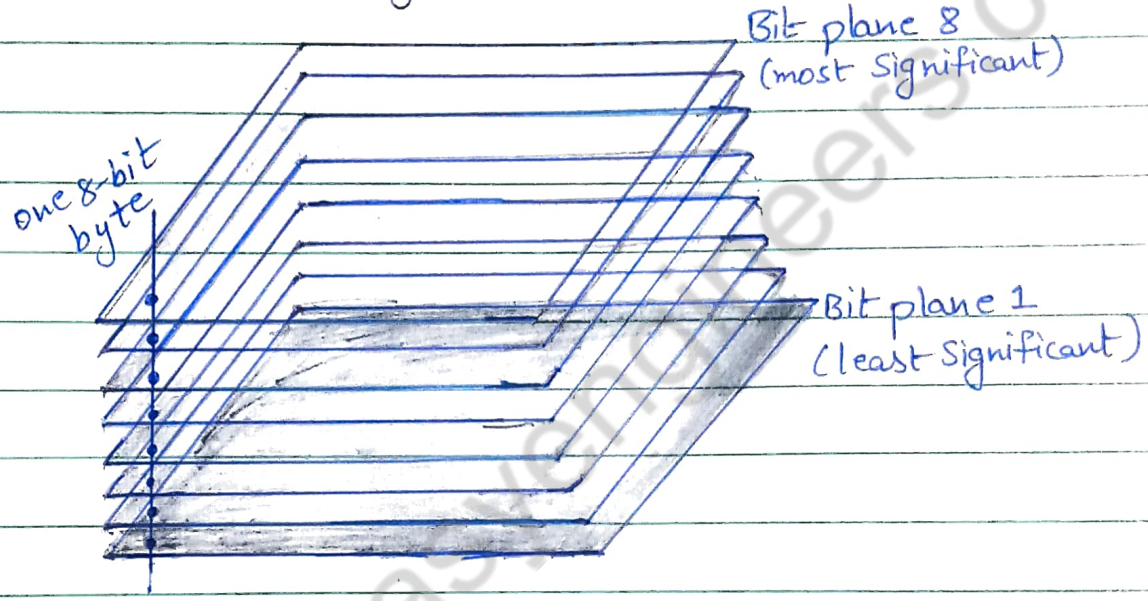


Fig 3.8: Bit-planes of an 8-bit image.

→ High order bits contains the majority of visually significant data & contribute to more the lowerorder planes contribute to more subtle intensity details in the image.

For Eg: The i/p image has a gray border whose intensity is 194.

* The corresponding pixels in the bit planes, starting with the highest-order plane, have values 1 1 0 0 0 0 1 0, which is the binary representation of decimal 194.

* The value of any pixel in the i/p image can be similarly reconstructed from its corresponding binary-valued pixels in the bit planes. [It helps in determining

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

$$h(r_k) = n_k \quad \text{for } k = 0, 1, 2, \ldots L-1 \quad \text{---> Eq" (6)}$$

where, n_k is the number of pixels in f with intensity r_k,

→ It's common practice to normalize a histogram by dividing each of it's values by the total number of pixels in the image denoted by the product MN, where as usual, M & N are the row and column dimensions of the image.

Thus, a normalized histogram of "f" is defined as,

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN} \quad \text{(or)} \quad \text{---> Eq" (7)}$$

$$p(r_k) = \frac{n_k}{n}$$ where, n → is total no of pixels, i.e $n = MN$.

→ The histogram plots are simple plots of $h(r_k) = n_k$ versus r_k.

→ The sum of all components of a normalized histogram is equal to 1.

→ Histograms are basis for numerous spatial domain processing techniques.

→ Histogram processing shows four basic intensity characteristics (contrast) based on the distribution of gray levels in the image. (or type of images).
They are: dark, light, low contrast and high contrast image.

① Dark image: In the dark image, the components of the histogram are concentrated on low (dark) side of the gray scale, as shown in fig 3.9(a)

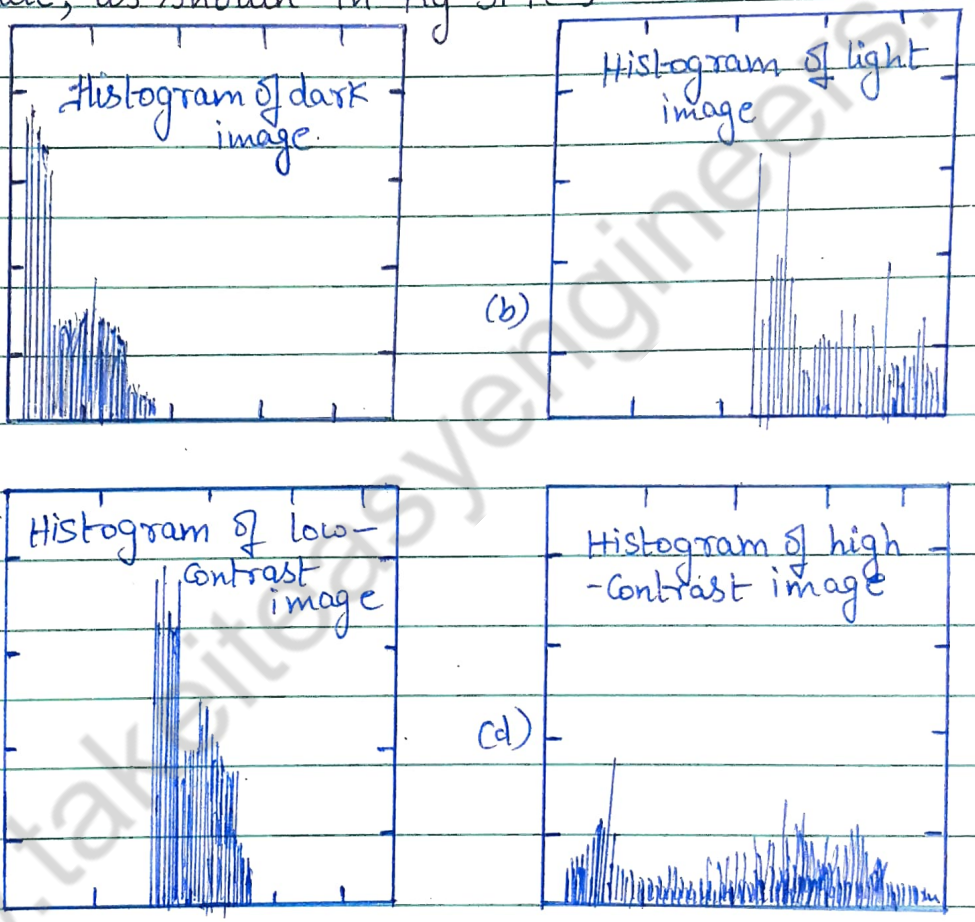


Fig 3.9: Four image types & their corresponding histogram (a) dark (b) light (c) low contrast (d) high contrast.
The horizontal axis of the histograms are values of r_k & the vertical axis are values of $P(r_k)$.

② Light image (or bright): Here, the components of the histogram are biased towards the high side of the gray scale as shown in fig 3.9(b).

③ Low-Contrast image: An image with low contrast has a narrow histogram located typically towards the middle of the gray scale, as shown in fig 3.9(c).

④ High-Contrast image: In the high-contrast image cover a wide range of the intensity/gray scale & the distribution of pixels is almost uniform, with very few vertical lines being much higher than the others.

* Histogram gives global description of image & does not give any information about the content of image.

Histogram based techniques modify histogram to make image appear in a particular way.

① Histogram Equalization
② Histogram Modification / Specification

Histogram Equalization :

Histogram Equalization is a common technique for Enhancing the appearance of images.

For Eg: we have an image which is dark. Then its histogram would be skewed towards the lower end of the grey scale & all the image detail are compressed into the dark end of the histogram. If we could "stretch out" the grey level at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Consider for a moment continuous intensity values & let the variable r denote the intensities/gray levels of an

image to be Enhanced or processed.
We assume that "r" is in the range [0, 1] where r=0 represents black & r=1 represents white (or)
We can Extend it to L levels, r [0, L-1], where r=0 is black & r = L-1 is white.

The transformation (Intensity mapping) function is of the form:

$$s = T(r) \quad 0 \le r \le L-1 \quad \text{----->Eq" (8)}$$

which produces an output intensity level 's' for Every pixel 'r' in the input image.

→ We assume that the transformation function T(r) Satisfies the following conditions:
(a) T(r) is Single valued & monotonically increasing function in the interval $0 \le r \le L-1$ (1) and
(b) $0 \le T(r) \le 1$ (L-1) for $0 \le r \le L-1$ (1)

In the 1st Condition T(r) to be Single valued is needed to Ensure that inverse transformation will Exist, & monotonically Condition preserves the increasing order from black to white in the output image.
[If transformation function is not montonically increasi-ng, it may produce Same inverted gray level in the o/p image, i.e darker place appear darker & brighter pixels appear brighter in processed image & pixel ordering is not changed.]
The 2nd Condition guarantees that the output gray levels

will be in the same range as the input levels. (i.e o/p gray levels does not Exceed the i/p gray levels or range of o/p & i/p gray levels are same).

→ In some formulations, we use the inverse transform -ation,

$$r = T^{-1}(s) \qquad 0 \le s \le L-1 \quad \text{---} \rightarrow \text{Eq}^{n}(9)$$

in which case we change Condition (a) to (a') Tr is a strictly monotonically increasing function in the interval $0 \le s \le L-1$.

(a') T(r) is a strictly monotonic increasing function in the interval $0 \le r \le L-1$.

→ The requirement in Condition (a) that T(r) be monotoni -cally increasing guarantees that output intensity values will never be less than corresponding i/p values, thus preventing artifacts created by reversals of intensity (inverted gray levels).

→ Condition (b) guarantees that the range of o/p intensities is the same as the input.

→ Finally, Condition (a') guarantees that the mappings from s back to r will be one-to-one, thus preventing ambiguities.

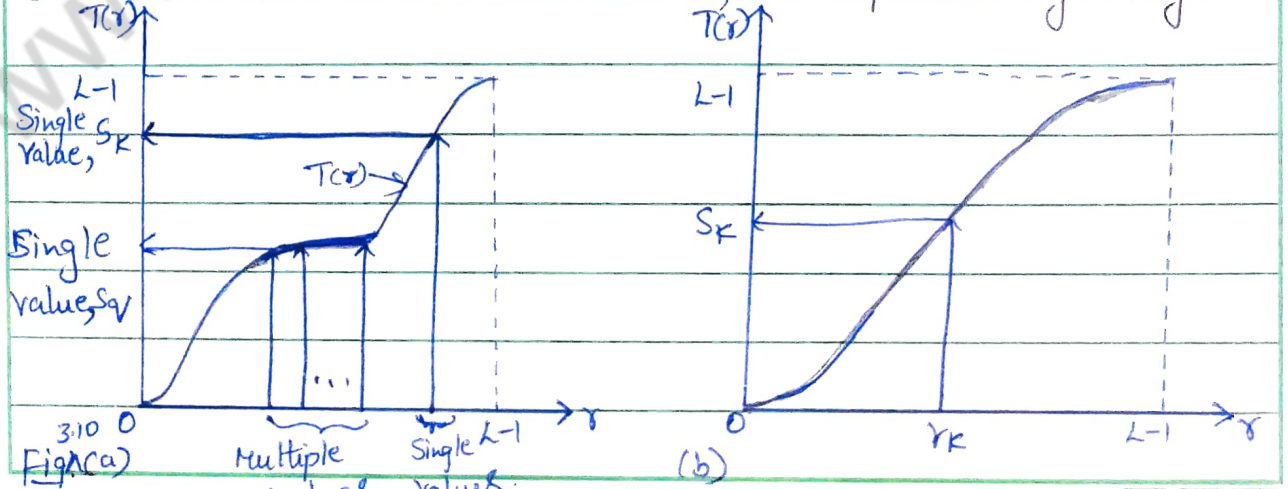


Fig 3.10

Fig 3.10 (a): Monotonic 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.

Fig 3.10 (a) shows a function that satisfies conditions (a) and (b). Here we see that it's possible for multiple input values to map to a single o/p value, & still satisfy these two conditions.
i.e a monotonic transformation function performs a one-to-one or many-to-one mapping. As fig 3.10 (b) shows, requiring that T(r) be strictly monotonic guarantees that the inverse mapping will be single valued. (i.e mapping is one-to-one in both directions).

→ The intensity levels in an image may be viewed as random variables in the interval $[0, L-1]$.
→ Let $p_r(r)$ and $p_s(s)$ denote the probability density functions (PDF) of r & s, respectively, where the subscripts on p are used to indicate that p_r & p_s are different functions in general.
→ A fundamental result from basic probability theory is that if $p_r(r)$ & $T(r)$ are known, & $T(r)$ is continuous & differentiable over the range of values of interest, then the PDF of the transformed (mapped) variable "s" can be obtained using the simple formula,

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \quad \text{---------->} \quad \text{Eq}^{n} (10)$$

Thus we see that the PDF of the o/p intensity variable, s, is determined by the PDF of the i/p intensities & the transformation function used [r & s are related by T(r)]
→ A transformation function of particular importance in

image processing has the form:

$$s = T(r) = (L-1)\int_0^r P_r(w)\,dw \quad \text{-----} \rightarrow \text{Eq}^n (11)$$

where w is a dummy variable of integration. The right side of this Eq^n is recognized as the cumulative distribution function (CDF) of random variable r.

⇒ Because PDFs always are positive & the integral of a function is the area under the function, it follows that the transformation function of Eq^n (11) satisfies condition(a). This is because area under the function can't decrease as "r" increases.

⇒ when the upper limit in this Eq^n is $r = (L-1)$, the integral evaluates to 1, (area under a PDF curve always is 1), Thus, the maximum value of s is $L-1$ & condition(b) is satisfied also.

Thus derivative of s w.r.t r is given by, (or)

We know from Leibniz's rule in basic calculus that the derivative of a definite integral with respect to it's upper limit is the integrand evaluated at the limit. That is,

$$\frac{ds}{dr} = \frac{dT(r)}{dr}$$

$$= (L-1)\frac{d}{dr}\left[\int_0^r P_r(w)\,dw\right]$$

$$= (L-1)P_r(r) \quad \text{-----} \rightarrow \text{Eq}^n (12)$$

Substituting this result of dr/ds & keeping in mind that all probability values are positive, yields,

& Substitute value of dr/ds in Eq^n (10).

$$P_S(s) = P_r(r)\left|\frac{dr}{ds}\right|$$

$$= P_r(r)\left|\frac{1}{(L-1)P_r(r)}\right|$$

$$= \frac{1}{L-1} \qquad 0 \le s \le L-1 \quad \text{---->} \text{ Eq}^n \text{ (13)}$$

We recognize the form of $P_S(s)$ in the line of this Eq^n as a uniform probability density function.

* From this Eq^n (13), it's important to note that $T(r)$ depends on $P_r(r)$, the resulting $P_S(s)$ always is uniform, independently of the form of $P_r(r)$.

Thus Enhances the Contrast of original image i.e Dynamic range is going to be Enhanced. (The above holds good for Continuous functions).

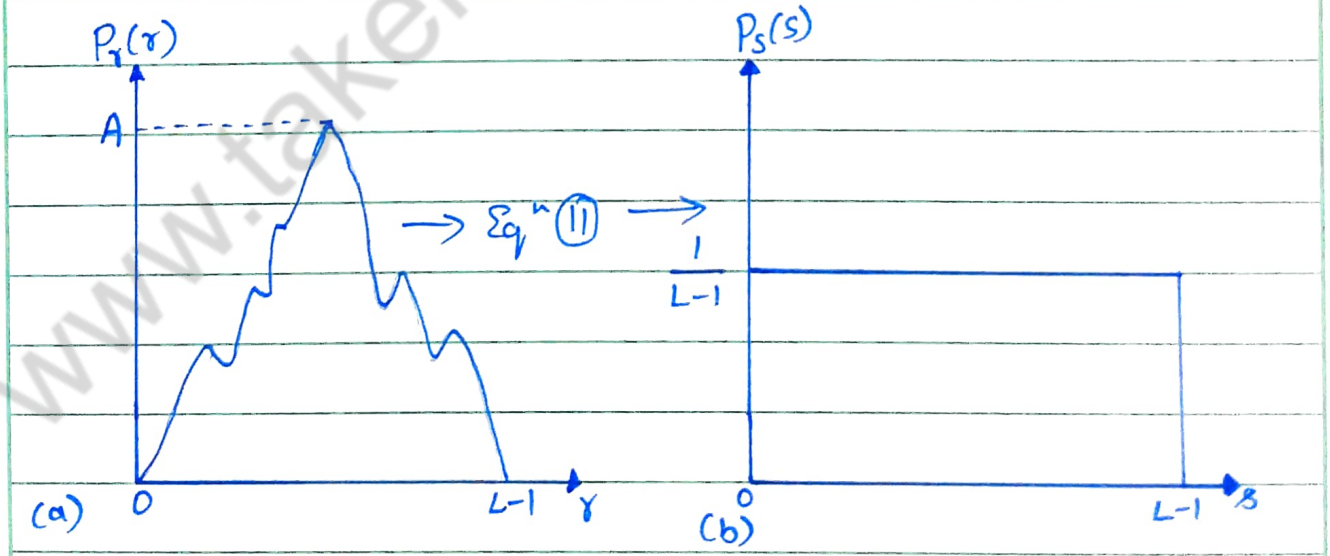


Fig 3.11 (a) Anarbitrary PDF (b) Result of applying Eq^n (11) to the i/p PDF (to all intensity levels, r) The resulting PDF is always uniform, independently of the Shape of the input.

For discrete values,

The probability of occurrence of gray level r_k in an image is

$$P_r(r_k) = \frac{n_k}{MN} \quad \longrightarrow \text{Eq}^n (14)$$

where $k = 0, 1, 2, \ldots, L-1$

MN = the total number of pixels in the image

n_k = The number of pixels that have gray level r_k

L = The total no of possible gray levels in the image

The discrete form of the transformation in Eq^n (11) is

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \quad \longrightarrow \text{Eq}^n (15)$$

$$= \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \quad \text{for } k = 0, 1, 2, \ldots, L-1.$$

⇒ Thus, a processed (o/p) image is obtained by mapping each pixel in the (i/p) image with intensity r_k into a corresponding pixel with level s_k in the output image.

⇒ The transformation (mapping) $T(r_k)$ in Eq^n (15) is called a histogram Equalization & histogram linearization transformation.

Histogram Equalization results in single processed image & does not interactive Enhancement. we can go with Matching techniques where we have particular histogram shapes capable of highlighting certain gray-level ranges. The method used to generate a processed image that has a specified histogram is called histogram matching.

Histogram Specification / Matching :

Histogram Equalization does not allow interactive image Enhancement & generates only one result: an approximation to a uniform histogram. Histogram Equalization is a good approach when automatic Enhancement is desired as the results from this technique are predictable & method is simple to implement. This approach is not suitable when a specified histogram shape is required.

→ The method used to generate images that have a specified histogram is called histogram matching or histogram Specification.

In histogram Specification, we have to perform an histogram technique to get desired target histogram.

Development of the method :

Consider Continuous intensities r & z as random variables with PDFs $P_r(r)$ & $P_z(z)$ respectively.

Here r & z → are the intensity (or) gray levels of the input & output images respectively.

$p_r(r)$ can be Estimated from given input image and $p_z(z)$ is the Specified PDF (probability density function) desired for the output image (target histogram)

Let S be a random variable with the property,

it gives an image with uniform distribution & it Equalizes given image

$$S = T(r) = (L-1)\int_0^r P_r(w)\,dw \quad \ldots\rightarrow \text{Eq}^n\ (16)$$

where w is dummy variable of integration.

Define a function G_1 on variable z with the property

$$G_1(z) = (L-1)\int_0^z P_z(v)\,dv = s \quad \ldots\ldots\ldots\ldots \rightarrow \text{Eq}^n (17)$$

where v is a dummy variable of integration.

From Equation 16 and 17 we have,

$$G_1(z) = s = T(r)$$

Therefore, that z must satisfy the condition,

$$z = G_1^{-1}(s) = G_1^{-1}[T(r)] \quad \ldots\ldots\ldots\ldots \rightarrow \text{Eq}^n (18)$$

The transformation function T(r) can be obtained using Eqⁿ (16), once $P_r(r)$ has been Estimated from input image. Similarly, function $G_1(z)$ can be obtained from Eqⁿ (17) because $p_z(z)$ is given.

Assuming that G_1^{-1} Exists and Satisfies both Conditions (using value montonically increasing & lies in range [0, 1]). From Eqⁿ 16, 17 & 18 shows that an image whose intensity levels have a Specified PDF can be obtained using the following procedure:

1. obtain $P_r(r)$ from the input image to use in Eqⁿ [16]
2. Use the Specified PDF, $P_z(z)$ in Eqⁿ [17] to obtain the function $G_1(z)$.
3. Compute the inverse transformation $z = G_1^{-1}(s)$; this is a mapping from s to z,
4. obtain the output image by first Equalizing the input image using Eqⁿ (16); the pixels value in this image are

the S values. For each pixel with value s in the Equaliz-ed image, perform the inverse mapping $z = G^{-1}(s)$ to obtain the corresponding pixel in the output image. When all pixels have been processed with this transform-ation, the PDF of the output image, $P_z(z)$, will be Equal to the Specified PDF.

The procedure described is not possible to obtain analytical Expressions for G^{-1}.
Fortunately, this approach appears to Simple for discrete values.
The discrete formulation is given as follow;

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_r(r_j) \quad k = 0, 1, 2, \ldots, L-1 \longrightarrow Eq^n (19)$$

→ from i/p image

Similarly, the transformation function:

$$G(z_q) = (L-1)\sum_{i=0}^{q} P_z(z_i) \ldots\ldots \longrightarrow Eq^n (20)$$

where $P_z(z_i)$ is the ith value of the Specified histogram. & for a value q, So that

$$G(z_q) = s_k \longrightarrow Eq^n (21)$$

Finally, we obtain the desired value z_q from the inverse transformation;

$$z_q = G^{-1}(s_k) \longrightarrow Eq^n (22)$$

When performed over all pixels, this is a mapping from

the s values in the histogram Equalized image to the Corresponding z values in the output image.

In practice, there is no need to Compute the inverse of G. Because Gray intensity levels are integers, it's a simple matter to Compute all the possible values of G using Eq^n 20 for $q = 0, 1, 2, \ldots, L-1$. These values are rounded to their nearest integer values spanning the range $[0, L-1]$ & Stored in a look-up table. Then, given a particular value of s_k, we look for the closest match in the table.

For Eg: If the 27th Entry in the table is the closest value to s_k, then $q = 26$ (start Counting intensities at 0) and z_{26} is the best Solution to Eq^n 22. Thus the given value s_k would map to z_{26}.

Because the z's are integers in the range $[0, L-1]$, it follows that $z_0 = 0$, $z_{L-1} = L-1$, & generally $z_q = q$. Therefore z_{26} = intensity value 26.

$\Rightarrow$ we repeat this procedure to find the mapping from Each value s_k to the value z_q that is its closest match in the table. These mappings are the Solution to the histogram Specification problem.

Given an i/p image, a Specified histogram, $p_z(z_i)$, $i = 0, 1, 2, \ldots, L-1$ & s_k's are the values resulting from Eq^n 19.

The procedure for discrete histogram Specification as follows:

1) Compute the histogram, $p_r(r)$, of the input image & use it in Eq^n 19 to map the intensities in the input image to the intensities in the histogram-Equalized image. Round the resulting values, s_k, to the integer ranges $[0, L-1]$.

2) Compute all values of function $G(z_q)$ using Eq^n 20 for $q = 0, 1, 2, \ldots, L-1$, where $p_z(z_i)$ are the values of the specified histogram. Round the values of G to integers in the range $[0, L-1]$. Store the rounded values of G in a lookup table.

3) For Every value of $s_k, k = 0, 1, 2, \ldots, L-1$, use the stored values of G from Step 2 to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k. Store these mappings from s to z. When more than one value of z_q gives the same match (i.e the mapping is not unique), choose the smallest value by Convention.

4) Form the histogram-specified image by mapping Every Equalized pixel with value s_k to the corresponding pixel with value z_q in the histogram-specified image, using the mappings found in Step 3.

Local Histogram processing:

The histogram processing methods discussed previous are global, in the sense that pixels are modified by a transformation function based on the intensity distribution (gray-level content) of an Entire image.

→ Global approach is suitable for overall Enhancement, there are cases in which it's necessary to Enhance details over small areas in an image.

→ The number of pixels in these areas may have negligible influence on the Computation of a global transformation whose shape does not necessarily guarantee the desired local Enhancement.

→ The solution is to devise transformation functions based on the intensity distribution in a neighborhood of every pixel in the image.

Using Histogram Statistics for image Enhancement:

Statistics obtained directly from a histogram can be used for image Enhancement [than from the pixels directly].

Let r denote a discrete random variable representing discrete gray-levels in the range $[0, L-1]$ & let $p(r_i)$ denote the normalized histogram component corresponding to the ith value of r, then the n^{th} moment of r about its mean is defined as,

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i) \quad \text{Eq}^n (23)$$

where m is the mean (average intensity) value of r

$$m = \sum_{i=0}^{L-1} r_i p(r_i) \quad \text{Eq}^n (24)$$

Eg: The second moment (also the variance of r) is:

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) \quad \text{Eq}^n (25)$$

Mean → is a measure of average intensity

Variance → (Std. deviation) is a measure of Contrast in an image.

Two uses of the mean & variance for Enhancement purposes:

* The global mean & Variance (global means for the Entire image) are useful for adjusting overall Contrast & intensity.
* The mean & Std. deviation for a local region are useful for correcting for large-scale changes in intensity & Contrast.

Enhancement using Arithmetic/Logic operations:

Array: An array operation involving one or more images is carried out on a pixel-by-pixel basis.

Eg: Consider following 2x2 images:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The array product of these 2 image is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

& the matrix product is given by:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}.$$

Arithmetic operations:

Arithmetic operations between images are array operations which means that arithmetic operations are carried out between corresponding pixel pairs.

The four arithmetic operations are denoted as

$$s(x,y) = f(x,y) + g(x,y)$$
$$d(x,y) = f(x,y) - g(x,y)$$
$$p(x,y) = f(x,y) \times g(x,y)$$
$$v(x,y) = f(x,y) \div g(x,y)$$

→ operations are performed between corresponding pixel pairs in f and g for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$, where M and N are the row & column sizes of the images. s, d, p & v are images of size MxN also.

Addition or Image Averaging:

Image Averaging can be used to remove noise in a given image if noise in the image has zero additive mean.

Let $g(x,y)$ denote a noisy image formed by the addition of noise, $\eta(x,y)$ to a noiseless image $f(x,y)$; i.e

$$g(x,y) = f(x,y) + \eta(x,y) \longrightarrow \text{Eq}^n (26)$$

where the assumption is that Every pair of coordinates (x,y), the noise is uncorrelated & has zero average value.

The objective of an image averaging is to reduce the noise content by adding a set of noisy images, $\{g_i(x,y)\}$. This is a technique used frequently for image Enhancement.

i.e, $$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y) \longrightarrow \text{Eq}^n (27)$$

& the Expectation Values of $\bar{g}$ is given by,

$$E\{\bar{g}(x,y)\} = f(x,y) \longrightarrow \text{Eq}^n (28)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \longrightarrow \text{Eq}^n (29)$$

where $E\{\bar{g}(x,y)\}$ is the Expected Value of $\bar{g}$, & $\sigma^2_{\bar{g}(x,y)}$ & $\sigma^2_{\eta(x,y)}$ are the Variances of $\bar{g}$ & η respectively, all at Coordinates

The Standard deviation at any point in average image,

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

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x,y)} \rightarrow (30)$$

from Equation (29) & (30) as K increases, indicates that the variability (noise) of the pixel values at each location (x,y) decreases.

→ An important application of image addition or averaging is in the field of astronomy, where imaging under very low light levels frequently causes sensor noise to render single images virtually useless for analysis.

② Image Subtraction:

An app^n of image subtraction is in the enhancement of differences between images & another area is medical imaging called mask mode radiography.

Given 2 images f(x,y) and h(x,y), the differences b/w 2 image is expressed as,

$$g(x,y) = f(x,y) - h(x,y) \rightarrow \text{Eq}^n (31)$$

is obtained by computing differences between all pairs of corresponding pixels from "f" & "h".

→ In mask mode radiography, h(x,y) the mask is an X-Ray image captured by TV camera [Instead of X-Ray film] placed opposite to an X-Ray source.

→ The procedure consists of injecting an X-Ray contrast medium into the patient's bloodstream, taking a series of images called live images [Samples which are denoted as f(x,y)] of the same anatomical region as h(x,y) &

Subtracting the mask from the series of incoming live images after injection of the contrast medium. The net effect of subtracting the mask from each sample live image is that the areas that are different between f(x,y) & h(x,y) appear in the output image, g(x,y) as enhanced detail. This is used to detect arterial disorder.

Implementation: Most images are displayed using 8bit's & hence we except image values not be outside the range 0 to 255. The values in a difference image can range from -255 to maximum 255, so some sort of scaling is required to display the results.

There are two ways to scale a difference image:

Method 1: * Add 255 to every pixel & divide by 2.

* All resulting pixels will fall in the range 0 to 255 but may not cover the entire range.

* This is a simple method & fast to implement but will not utilize the entire gray scale range to display the results. & generally cause loss in accuracy.

Method 2: This method is used when more accuracy & full average of 8-bit range is desired.

1) Obtain the value of minimum difference

2) Add the negative of minimum value to the pixels in the difference image. This will result in a modified image whose minimum value is 0.

3) All pixels in images are scaled to interval [0,255] by multiplying each pixel by quantity 255/Max, where Max is maximum pixel value in modified difference image. This approach is complicated & difficult to implement.

→ Image Subtraction is used in segmentation (In tracking moving vehicles, moving objects can be detected from stationary components using image subtraction.

3) Multiplication & division: An important app^n of image multiplication (& division) is shading correction.

Consider imaging sensor produces images that can be modified as the product of a "perfect image" denoted by $f(x,y)$, times a shading function, $h(x,y)$; i.e

$$g(x,y) = f(x,y) * h(x,y) \quad \rightarrow \text{Eq}^n (32)$$

* if $h(x,y)$ is known, we obtain $f(x,y)$ by multiplying the sensed image by the inverse of $h(x,y)$ (i.e dividing g by h).

* if $h(x,y)$ is not known, but access to the imaging system is possible, we can obtain an approximation to the shading function by imaging a target of constant intensity.

→ Another common use of image multiplication is in masking, also called region of interest (ROI), operations.

→ The process consists of multiplying a given image by a mask image that has 1's in the ROI & 0's elsewhere. There can be more than one ROI in the mask image & rectangular ROI shapes are used frequently for ease of implementation.

→ Division of two image is multiplication of one image by reciprocals of other image.

Logic operations:

Logic operations are also operated on a pixel by pixel basis.

→ when dealing with binary images, we Consider foreground (1-valued) & background (0-valued) sets of pixels.

→ we define regions (objects) as being Composed of foreground pixels, the set operations ($A \cup B$), ($A \cap B$) & A^c (Complement) become operations between the Coordinates of objects in a binary image.

we refer to union, intersection & Complement as the OR, AND & NOT logical operations.

NOT Logical operator: The NOT Operation of a Set A is the Set of Element not in A.

if A is a given Set of foreground pixels, NOT(A) is the Set of all pixels in the image that are not in A, i.e this operation turns all Elements in A to 0(black) & all the Elements not in A to 1 (white).

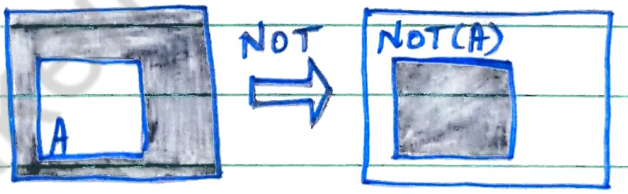
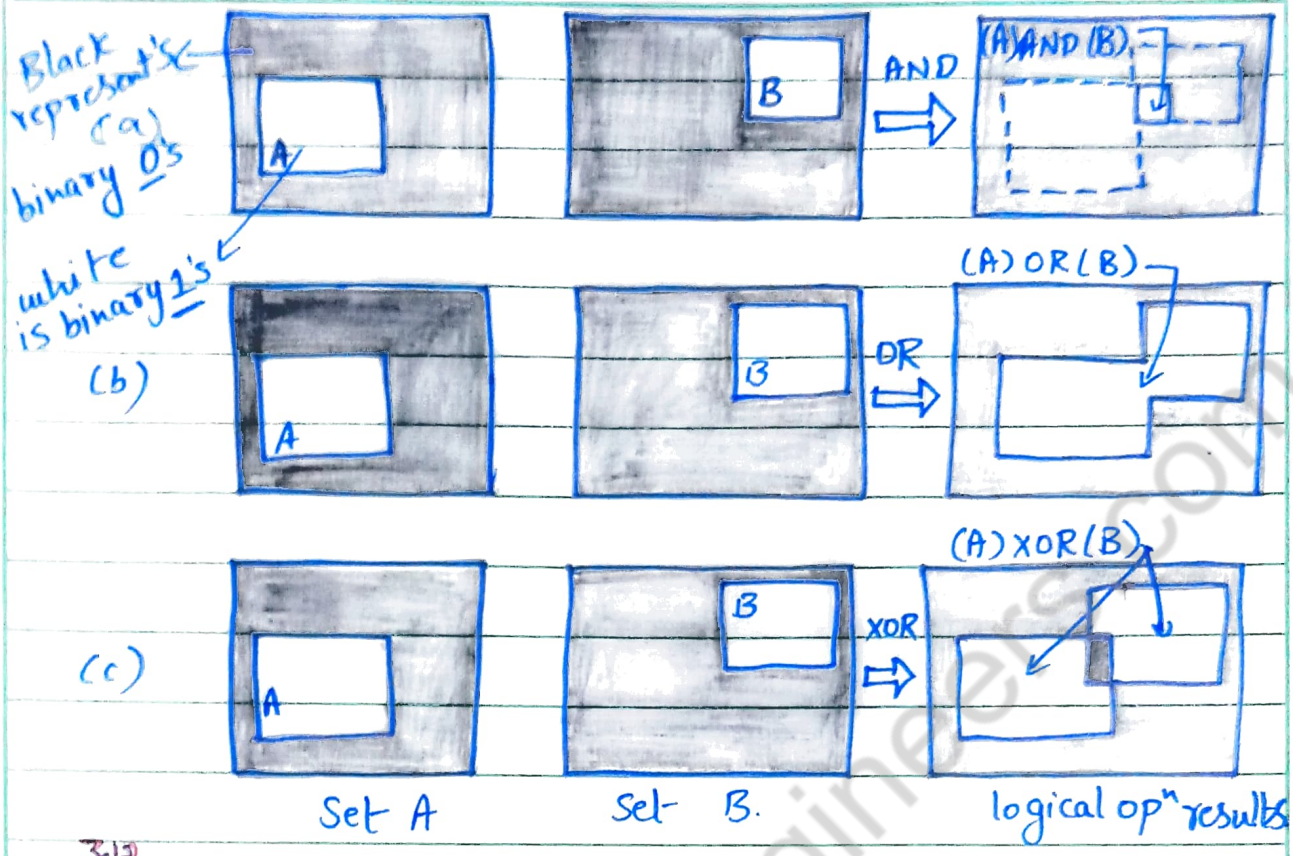


Fig 3.12(a) : NOT logical op^n.

AND and OR operations: Consider 2 regions (sets) A & B Composed of foreground pixels. The "OR" of these 2 sets is the Set of Elements (Coordinates) belonging either to A or B or to both.

The "AND" operations is the Set of Elements that are Common to A and B.

The XOR (Exclusive OR) operations, which is the Set of foreground pixels belonging to A or B, but not both.



3.12

Fig^ Illustration of logical operations (a) AND op^n (b) OR operation (c) XOR operations.

→ Any other logic operator can be implemented by using only these 3 basic functions. Logic operations are used Extensively in image morphology. [module-5].

Fundamentals of Spatial Filtering:

Spatial filtering is one of the principal tools used in a broad spectrum of applications.

→ The name filter is borrowed from frequency domain processing, where "filtering" refers to accepting (passing) or rejecting certain frequency components.

→ For Eg, a filter that passes low frequencies is called a lowpass filter.

→ The net Effect produced by a lowpass filter is to blur (smooth) an image.

→ we directly accomplish Smoothing on the image itself by using Spatial filters (called Spatial masks, Kernels, templates & windows.)

The Mechanics of Spatial Filtering:

* Spatial filtering operations are performed directly on the pixel values (amplitude / gray scale) of the image.
* Spatial filter Consists of a neighborhood and a predefined operation performed on the image pixels defining the neighborhood.
* Filtering results in creating a new pixel with coordinated of the neighborhood's center & the value is the result of filtering operation.
* A processed (filtered) image is generated as the center of the filter visits Each pixel in the input image.
* If the opereration is performed on the image pixels is linear, then the filter is called a linear Spatial filter, otherwise, the filter is nonlinear.

→ The process consists of moving the filter mask from point to point in the image. At each point (x,y) the response is calculated using a predefined relation -ship.

→ Fig 3.13 illustrates the mechanics of linear spatial filter -ing using a 3×3 neighborhood.

At any point (x,y) in the image, the response g(x,y) is given by a sum of products of the filter coefficients & the corresponding image pixels in the area spanned by the filter mask:

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

In the above Eq^n, the center co-efficient of the filter, w(0,0) aligns with the pixel at location (x,y).

→ For a mask of size m×n, we assume that

$$m = 2a+1 \text{ and } n = 2b+1,$$

where a & b are positive integers.

* In general, linear spatial filtering of an image of size M×N with a filter of size m×n is given by the expression.

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \quad \longrightarrow \text{Eq}^n (33)$$

where x & y are varied so that each pixel in w visits every pixel in f.

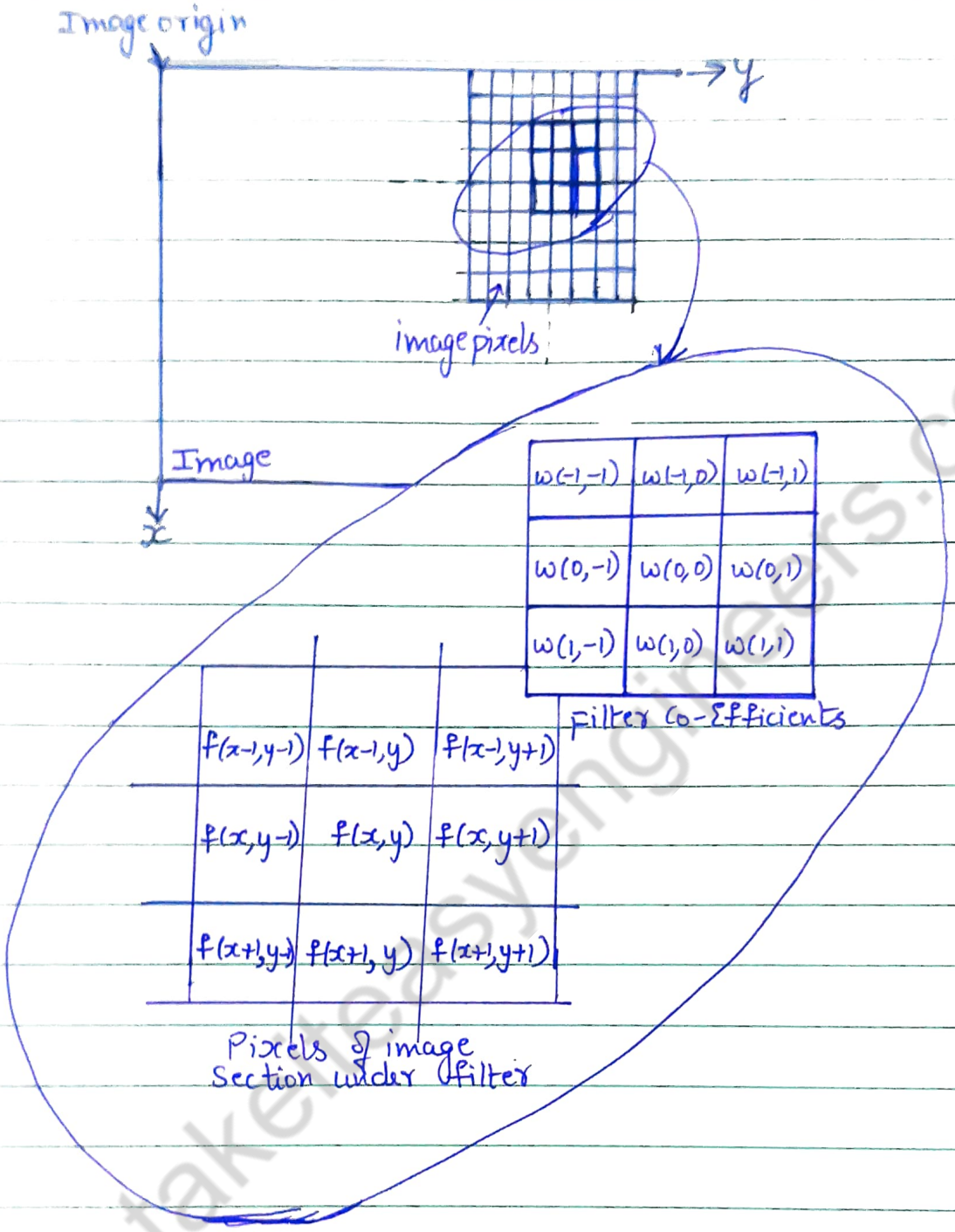


Fig 3.13 : The mechanics of linear Spatial filtering using a 3x3 filter mask. The form chosen to denote the coordinates of the filter mask co-efficients writing Expressions for linear filtering.

Spatial Correlation & Convolution

* Correlation is a process of moving the filter mask over the image and Computing the sum of products at Each location.
* Convolution is the Same Except that filter is first rotated by 180 degree.

For Eg, Consider 1-D illustration.

Correlation

origin f w

a) 0 0 0 1 0 0 0 0 0 1 2 3 2 8

b) 0 0 0 1 0 0 0 0
1 2 3 2 8
↑ Starting position alignment

c) Zero padding
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8

d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
↑ position after one shift

e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
↑ position after four shifts

f) 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
↑ Final Position

Full Correlation result

g) 0 0 0 8 2 3 2 1 0 0 0 0

Cropped Correlation result

h) 0 8 2 3 2 1 0 0

Convolution

origin f w rotated 180°

0 0 0 1 0 0 0 0 8 2 3 2 1 (i)

0 0 0 1 0 0 0 0 (j)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (k)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (l)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (m)
8 2 3 2 1

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 (n)
8 2 3 2 1

Full Convolution result

0 0 0 1 2 3 2 8 0 0 0 0 (o)

Cropped Convolution result

0 1 2 3 2 8 0 0 (p)

Fig 3.14 : Illustration of 1D Correlation & Convolution of a filter with a discrete unit impulse. [Note that Correlation & Convolution are functions of displacement].

Fig 3.14 (a) shows a 1-D function, f and a filter, w & Fig (b) shows the starting position to perform Correlation.

→ The first thing we note is that there are parts of the functions that do not overlap. The solution to this problem is to pad f with Enough 0's on Either Side to allow Each pixel in w to visit Every pixel in f.

→ If the filter is of Size m, we need m-1 0's on Either Side of f.

→ Fig 3.14 (c) shows a padded function.

* The first value of Correlation is the Sum of products of f and w for the initial position shown in fig 3.19(c) (The Sum of product is 0) This corresponds to a displacement $x = 0$.

* To obtain the 2nd value of Correlation, we Shift w one pixel location to the right (a displacement of $x=1$) & Compute the Sum of products. Result is again 0.

* The first nonzero result is when $x=3$, in which case the 8 in w overlaps the 1 in f & the result of Correlation is 8.

* proceeding in this manner, we obtain the full Correla-tion result in fig 3.14(g).

[It took 12 values of x (i.e, $x = 0, 1, 2 \ldots, 11$) to fully Slide w past f so that Each pixel in w visited

Every pixel in f.]

⇒ we always work with correlation arrays that are the same size as f, finally crop the full correlation to the size of the original function shown in fig 3.19(h).

* Correlation is a function of displacement of the filter. i.e, the 1st value of correlation corresponds to zero displacement of the filter, 2nd corresponds to one unit displacement & so on.

→ A function containing a single 1 with the rest being zero is called a discrete unit impulse.

* Correlation of a function with a discrete unit impulse yields a rotated version of a function at the location of the impulse.

* To perform a convolution, we need to pre-rotate the filter by 180 degree & perform the same operation as in correlation.

⇒ In a 2D case, for a filter of size m x n, we pad the image with m-1 rows of zero's at the top & bottom & n-1 columns of zero's on the left & right as shown in fig 3.15 Here m & n are equal to 3, so we pad f with two rows of 0s above & below & two columns of 0's to the left & right, as in fig 3.15(b).

Fig 3.15 - Correlation (Middle row) & Convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0's shown in blue to simplify visual analysis.

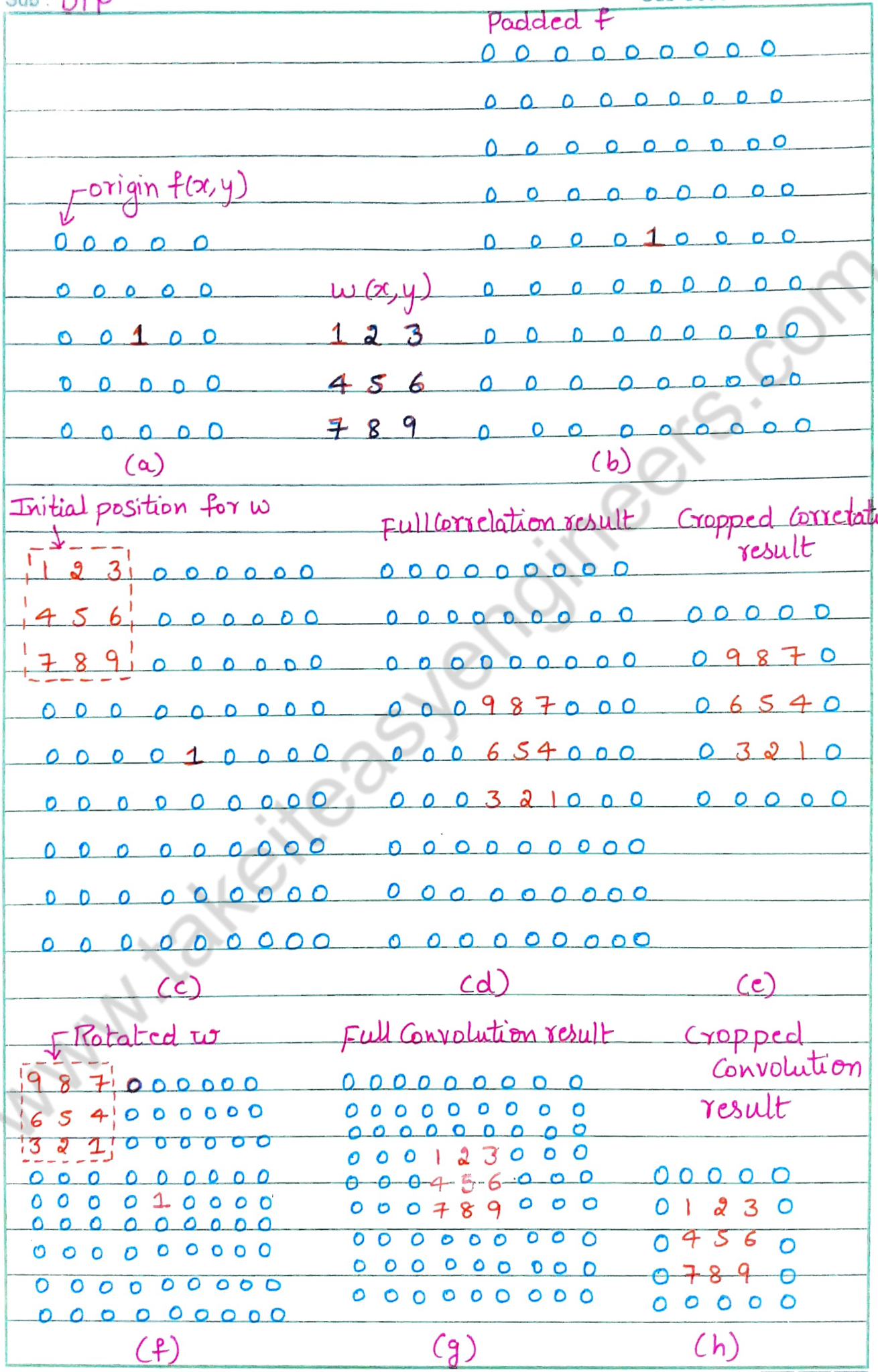


Fig 3.15(c) shows the initial position of the filter mask for performing correlation & Fig 3.15(d) shows the full correlation result. Fig 3.15(e) shows the cropped result.

→ Again result is rotated by 180°. For Convolution, we pre-rotate the mask as before & repeat the sliding sum of products as Explained in 1-D Eg. Fig 3.15(f) to (h) shows the result.

* Correlation of a filter $w(x,y)$ of size $m \times n$ with an image $f(x,y)$ is given by:

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \quad \rightarrow \text{Eq}^{n} (34)$$

This Eq^{n} is Evaluated for all values of the displacement variables x & y so that all Elements of w visit Every pixel in f, where we assume that f has been padded appropria-tely. $[a=(m-1)/2, b=(n-1)/2]$

Similarly,

* Convolution of $w(x,y)$ & $f(x,y)$ is given by:

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \quad \rightarrow \text{Eq}^{n} (35)$$

where the minus signs on the right flip f (i.e rotated it by 180°).

Vector Representation of Linear Filtering:

Sometimes it is convenient ..... to write the sum of products as

$$R = w_1 z_1 + w_2 z_2 + \ldots\ldots + w_{mn} z_{mn}$$

$$R = \sum_{k=1}^{mn} w_k z_k \rightarrow \text{Eq}^n (36)$$

$$= W^T Z$$

where R → mask filter for correlation or convolution.
W's → Co-efficients of an mxn filter
Z's → Corresponding image intensities encompassed by the filter

Equation 36 used as given for correlation but to use same Eq^n 36 for convolution rotate the mask by 180°.

For Eg: for a 3x3 filter mask,

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Fig: 3x3 filter mask

Eq^n (36) becomes

$$R = w_1 z_1 + w_2 z_2 + \ldots\ldots + w_9 z_9$$

$$= \sum_{k=1}^{9} w_k z_k \rightarrow \text{Eq}^n (37)$$

$$= W^T Z$$

where W & Z are 9-dimensional vectors formed from the coefficients of the mask & the image intensities encompassed by the mask respectively.

Generating Spatial Filter Masks:

* Generating an mxn linear spatial filter requires specification of mn mask coefficients.
* These coefficients are selected based on what the

filter is supposed to do keeping in mind that all we can do with linear filtering is to implement a sum of products.

For Eg, we need to replace the pixels in image by the average intensity of a 3x3 neighborhood centered on those pixels.

→ The average value at any location (x,y) in the image is the sum of the nine intensity values in the 3x3 neighborhood centered on (x,y) divided by 9.

Letting z_i, $i = 1, 2, \ldots, 9$ denote these intensities, the average is,

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i \qquad \left[R = \sum_{i=1}^{9} w_i z_i\right]$$

where this is same as Eq" 37, with coefficient values $w_i = 1/9$.

Smoothing Spatial Filters

Smoothing filters are used for blurring & noise reduction.

→ Blurring is used in preprocessing tasks such as removal of small details from an image prior to (large) object Extraction, & bridging of small gaps in lines or curves. * Noise reduction can be accomplished by bluring with a linear & also by non-linear filter.

Smoothing Linear Filters :

The output (response) of a smoothing, linear spatial filter is the average of the pixels contained in the neighborhood of the filter mask. It's also called as averaging or low-pass filters.

→ The process is performed by replacing the value of every pixel in the image by the average of the gray level intensity levels in the neighborhood defined by a filter mask, the resulting image will have reduced "sharp" transitions in intensities.

→ The application of smoothing filter is noise reduction as random noise typically consists of sharp transition in gray level. However, Edges are characterized by sharp intensity transitions will be blurred.

For Eg : 1) A box filter - Spatial averaging filter 3x3.
2) weighter average filter - attempt to reduce blurring

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

Fig 3.16 (a) Box filter (3x3 filter mask)

The response of 3×3 is given by,

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

→ An m×n mask would have a normalizing constant Equal to 1/mn. A Spatial averaging filter in which all Co-Efficients are Equal Sometimes is called a box filter.

→ The 2nd type of mask called weighted average, where the pixels are multiplied by different Co-Efficients, thus giving more importance (or weight) to some pixels at the Expense of others.

Weighted Average reduces the blurring Effect.

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$$g(x,y) = \frac{1}{16}\sum_{i=1}^{16} z_i$$

$$g(x,y) = \frac{1}{16}\sum_{i=-1}^{1}\sum_{j=-1}^{1} w_{ij}\, f(x+i, y+j)$$

Fig 3.16 (b) Weighted Mask.

In fig 3.16 (b) the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the Calculation of the average. The other pixels are inversely weighted as a function of their distance from the center of the mask.

→ The basic strategy here, the center point the highest & then reducing the values of the coEfficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the Smoothing process.

The general implementation for filtering an MxN image with a weighted averaging filter of size mxn (m & n odd) is given by the Expression:

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)} \quad \rightarrow \text{Eq}^{n} (38)$$

where $x = 0, 1, 2, \ldots M-1$ & the denomination is simply $y = 0, 1, 2, \ldots N-1$ the sum of mask coefficients & is $a = \frac{m-1}{2}$ & $b = \frac{n-1}{2}$ Constant i.e Computed only once.

Fig 3.17

a b c d e f

Fig. 3.35 (a) Original image, of size 500 x 500 pixels. (b) – (f) Result of smoothing with square averaging filter masks of size n = 3, 5, 9, 15, and 35 respectively.

Order-Statistic (Nonlinear) Filters:

Order-Statistic filters are nonlinear Spatial filters whose response is based on ordering (Ranking) the pixels in the neighborhood & then replacing the value of the center pixel by the value determined by the ranking result.

→ The best Example is median filter.

→ Median filter replaces the value of a pixel by the median of the gray intensity values in the neighborhood pixel.

→ Median filters are popular because they provide Excellent noise-reduction capabilities for certain type of random noise with less blurring than linear Smoothing filters of similar size.

→ The median-filters are quite Effective against the impulse noise also called Salt-&-pepper noise (it appears as white & black dots imposed on an image).

→ The median, ξ, of a set of values is such that half the value in the set are less than or Equal to ξ, & 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 pixel in the neighborhood, determine their median & assign that value to the Corresponding pixel in the filtered image.

For Eg: in a 3x3 neighborhood the median is the 5th largest value, in a 5x5, it is 13th largest value & so on.

For Eg: The 3x3 neighborhood has value (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are Sorted (ranked) as (10, 15, 20, 20, 20, 20, 25, 100), which results in a median of 20.

Consider another Example, a 3x3 neighborhood as shown below,

85 98 100 99 105 102 90 101 108

Before Sort

85	98	100
99	105	102
90	101	108

Sort the gray values in ascending orders:

85 90 98 99 [100] 101 102 105 108

↓ ξ

After Sort

85	90	98
99	100	101
102	105	108

→ g(x,y)

The median will be 100.

→ Thus, the principal function of median filters is to force points with distinct intensity levels to be more like their neighbors.

* Median filter is the most useful order-statistic filter in image processing.

Median filter is 50th percentile filter & other order-Statistics filter includes Max filter & Min filter.

Max filter [100th percentile] used for finding the brightest points in an image.

It's given by,

$$R = \max\{z_k \mid k = 1, 2, \ldots, 9\}.$$

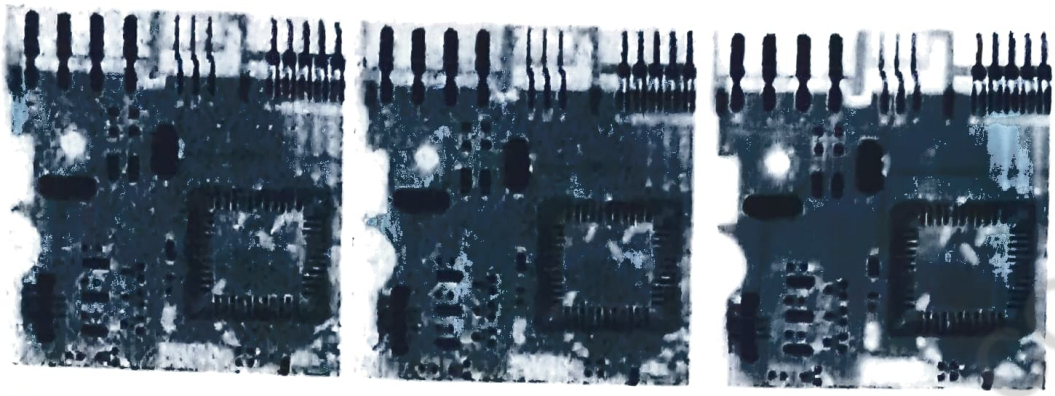
Min filter [0th percentile] for finding the darkest points in an image.

It's given by,

$$R = \min\{z_k \mid k = 1, 2, \ldots, 9\}$$

Fig 3.18

Fig pg. No 157



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 arithmetic mean filter. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph L. Paseman, Lixi, Inc.)

Sharpening Spatial Filters

The Main objective of Sharpening is to highlight transitions in intensity, i.e to highlight fine details in an image & to Enhance details that have been blurred.

→ The various applications of Image Sharpening includes Electronic printing & medical imaging to industrial inspection & autonomous guidance in military systems.

→ pixel Averaging is done in order to Smoothing or blurring Effect of an image. Averaging is same as integration.

* Hence, it is logical that Sharpening can be accomplis-hed by spatial differentiation.

→ Image differentiation Enhances Edges & other discontinui-ties (such as noise) & demphasizes the areas with slow varying gray levels values.

→ The derivatives of a digital function are defined in terms of differences.

Derivative operations can be,

① The first (gradient) order derivative

② The 2nd (Laplacian) order derivative

* The first derivative must be:

1) zero in areas of Constant intensity.

2) Non-zero at the onset & End of an intensity step or ramp.

3) Non-zero along ramps of Constant Slope.

* The 2nd derivative must be:

1) Zero in areas of Constant intensity
2) Non-zero at the onset & End of an intensity step or ramp.
3) Zero along ramps of Constant slope.

First-order derivative of a one-dimensional function f(x) is the difference,

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \quad \rightarrow \text{Eq}^{n} (1)$$

The 2nd-order derivative given by,

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \quad \rightarrow \text{Eq}^{n} (2)$$

⇒ It can be verified that these 2 definitions satisfy the conditions for derivatives.

⇒ To illustrate this & to examine the similarities & differences between first & 2nd-order derivatives of a digital function, Consider an example a horizontal gray level profile of scan line as shown in fig 3.19(b).

* The value inside the small squares are the intensity values in the scan line, which are plotted as Red dots above it in Fig 3.19 (a).
* As the figure shows, the scan line contains an intensity ramp, 3 sections of constant intensity & an intensity step.
* Circles indicate the onset or End of intensity transitions
* The 1st & 2nd order derivatives computed using the 2 preceding definitions are included below the scan line in Fig 3.19(b) & are plotted in fig 3.19(c).

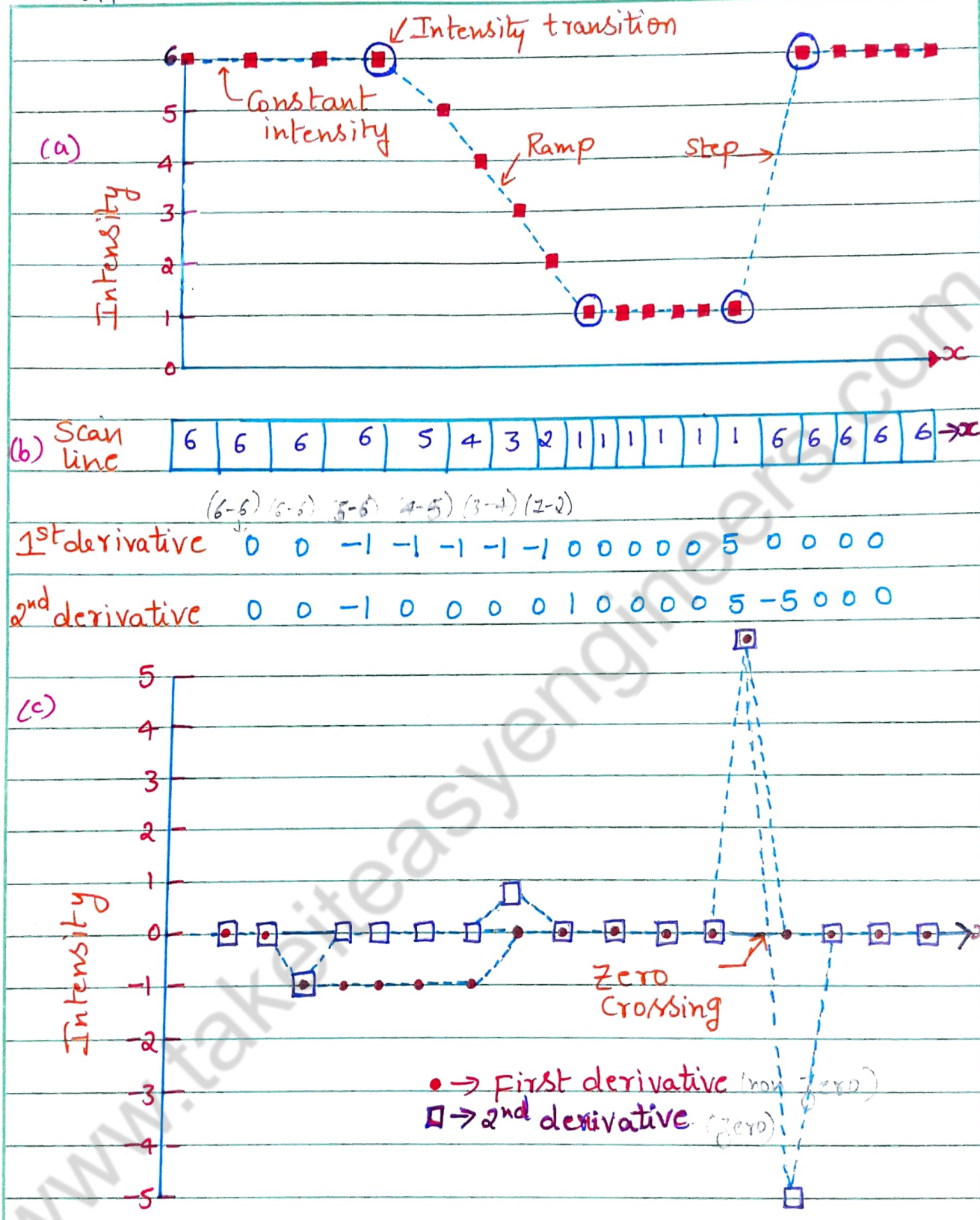


Fig : 3.19 Illustration of the first & 2nd derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image.
[dashed lines to join data points are just visualization aid]

→ When Computing the first derivative at a location x, we Subtract the value of the function at that location from the next point. So, this is a "look-ahead" operation.
→ Similarly, to Compute the 2nd derivative at x, we use the previous & the next points in the Computation.

Consider the properties of the first & 2nd derivatives as it traverse the profile from left to right
First, we see an area of Constant intensity & as fig 3.19(b) & (c) show, both derivatives are zero there, so Condition (1) is Satisfied for both.
2nd we Encounter an intensity ramp followed by a step, & we note that 1st order derivative is non-zero at the onset of the ramp & the step, 2nd order is non-zero at the onset & End of both the ramp & the step. Therefore Condition (2) Satisfied for both.
Finally property (3) is also Satisfied for both derivatives 1st derivative is nonzero & 2nd is zero along the ramp. The Sign of the 2nd derivative changes at the onset & End of a Step or ramp.
In fig 3.19(c), In a Step transition a line joining these two values crosses the horizontal axis midway between the 2 Extremes. This zero crossing property is quite useful for locating Edges.

The response of 1st order & 2nd order derivative can be Summarized as below:
① 1st order derivative generally produce a thicker Edges in an image.
② 2nd order derivative give Stronger response to fine

details such as thin lines & isolated points

③ 1st order derivatives have stronger response at gray-level.

④ 2nd order derivative produces a double response at step changes in gray level.

2nd order derivative is better suited in most image Enhancement applications, as it has the ability to Enhance fine details.

Use of 2nd order derivative for Image sharpening - The Laplacian.

The idea is defining a discrete formulation of the 2nd order derivative & then constructing a filter mask based on that formulation.

→ Hence, we consider isotropic filter's, whose response is independent of the direction of the discontinuities in the image to which the filter is applied.

→ The simplest isotropic derivative operator is the Laplacian, for a function (image) $f(x,y)$ of 2 variable is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \rightarrow \text{Eq}^n (3)$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator.

In the x-direction we have,

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \quad \rightarrow \text{Eq}^n (4)$$

Similarly in the y-direction we have,

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \rightarrow \text{Eq}^n (5)$$

Therefore, from 3 Eq^n 3, 4, & 5 the discrete Laplacian of 2 variable is,

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

$\hookrightarrow$ Eq^n (6)

This Equation can be implemented using the filter mask in fig 3.20(a)

0	1	0
1	-4	1
0	1	0

(a)

1	1	1
1	-8	1
1	1	1

(b)

0	-1	0
-1	4	-1
0	-1	0

(c)

-1	-1	-1
-1	8	-1
-1	-1	-1

(d)

Fig 3.20(a) Filter mask used to implement Eq^n (6).
(b) Mask used to implement an Extensions of this Eq^n (6) that includes the diagonal terms.
(c) & (d) Two other implementation of the laplacian found frequently.

Fig 3.20(b) shows Extension of Eq^n (6) by adding 2 more terms to Eq^n (6), one for each of the 2 diagonal directions.
Each diagonal term also contains a $-2f(x,y)$ term, the total subtracted from the difference terms would be $-8f(x,y)$.

→ Since the laplacian is a derivative operator, it's use highlights intensity discontinuities in the image & deemphasize regions with slow varying intensity levels.

→ It produces images having grayish Edge lines & other discontinuities & a dark, feature-less background.

→ Background features can be preserved together with the Sharpening Effect of the Laplacian by adding the Laplacian image to the original.

→ If the definition of the Laplacian has a negative central co-efficient, the Laplacian image must be subtracted rather than added to obtain a Sharpening result.

In general;

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)] \quad \rightarrow Eq^n (7)$$

where $f(x,y)$ & $g(x,y)$ are the i/p & sharpened images, respectively.

The constant is $c = -1$, if the filters in fig 3.20(a) or (b) are used & $c = 1$ if either of the other 2 filters is used.

i.e $g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \rightarrow \text{if the center coefficient of laplacian mask is negative} \\ f(x,y) + \nabla^2 f(x,y) \rightarrow \text{if center coefficient of mask is positive.} \end{cases}$

i.e $g(x,y) = f(x,y) - \nabla^2 f(x,y)$

$= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) - f(x,y-1)] + 4f(x,y)$

$= 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$

This can be implemented using the mask shown here →

0	-1	0
-1	5	-1
0	-1	0

If diagonal neighbors are also included the mask is as shown below,

-1	-1	-1
-1	9	-1
-1	-1	-1

The result obtained with mask containing the diagonal terms are usually Sharper than those obtained using basic mask.

Unsharp Masking and Highboost filtering

An Unsharp Masking is used to Sharpen images by subtracting an unsharp (Smoothed) version of an image from the original image.

This process, called unsharp masking,

It consists the following steps:

① Blur the original image

② Subtract the blurred image from the original (the result is called the mask).

③ Add the mask to the original.

Unsharp masking is expressed as,

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y) \quad \rightarrow Eq^n (8)$$

where, $\bar{f}(x,y)$ → blurred image,

Then we add a weighted mask to the original image:

$$g(x,y) = f(x,y) + K * g_{mask}(x,y) \quad \rightarrow Eq^n (9)$$

where, K is a weight ($K \geq 0$) for general.

when $K = 1$ we have unsharp masking

$K > 1$ the process referred to as highboost filtering

$K < 1$ de-Emphasizes the Contribution of the unsharp mask.

Fig 3.21 Explains how unsharp masking works. The fig 3.21(a) Intensity profile can be viewed as a horizontal scan through a vertical Edge transition from a dark to a light region in an image

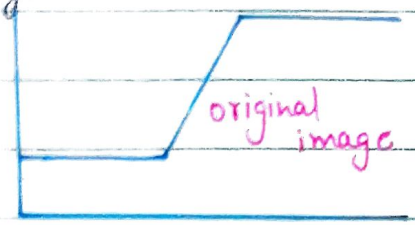
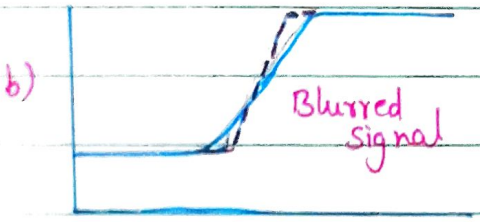
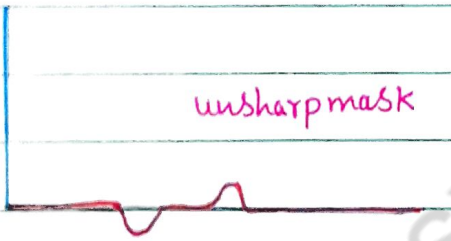


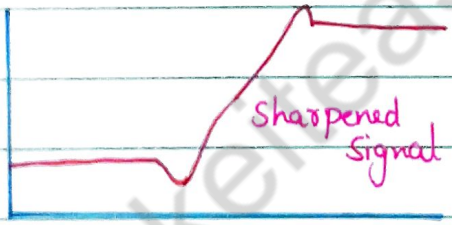
Fig 3.21 : 1-D illustration of the mechanics of unsharp masking
(a) original Signal.



(b) Blurred Signal with original shown dashed for reference



(c) unsharp mask
[obtained by subtracting the blurred signal from the original]



(d) Sharpened signal obtained by adding (c) to (a).
[i.e adding mask to the original Signal]

Highboost filtering: is generalization of unsharp masking
It's denoted using f_{hb} at (x,y) & is given by,

$$f_{hb}(x,y) = A f(x,y) - \bar{f}(x,y) \quad \rightarrow \text{Eq}^{n}\ (10)$$

where $A \geq 1$

$\bar{f}(x,y)$ → blurred version of $f(x,y)$

Eq^{n} 10 can be written as,

$$f_{hb}(x,y) = (A-1) f(x,y) + \underbrace{f(x,y) - \bar{f}(x,y)}_{f_{mask}(x,y)} \quad \rightarrow \text{Eq}^{n}\ (11)$$

Substituting Eqn 8 in (11) we get,

$$g_{hb}(x,y) = (A-1) f(x,y) + g_{mask}(x,y) \rightarrow \text{Eqn } 12$$

If Laplacian is used, $f_s(x,y)$ can be obtained from the Eqn below,

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{[Center coefficient is -ve]} \\ f(x,y) + \nabla^2 f(x,y) & \text{[Center coefficient is +ve]} \end{cases}$$

Eqn (15) can be given as,

$$g_{hb} = \begin{cases} Af(x,y) - \nabla^2 f(x,y) & \text{if center coefficient is negative} \\ Af(x,y) + \nabla^2 f(x,y) & \text{if center coefficient is positive.} \end{cases}$$

Thus highboost filter can be implemented using the two masks shown below.

0	-1	0
-1	A+4	-1
0	-1	0

-1	-1	-1
-1	A+8	-1
-1	-1	-1

Using First-order Derivatives for (Nonlinear) Image Sharpening - The Gradient.

First derivatives in image processing are implemented using the magnitude of the gradient.

→ For a function $f(x,y)$, the gradient of f at coordinates (x,y) is defined as the 2-dimensional column vector.

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \rightarrow \text{Eq}^n (1)$$

→ The gradient vector points in the direction of the greatest rate of change of f at location (x,y).

→ The magnitude (length) of gradient is the value of rate of change at (x,y) in the direction of gradient.

magnitude of vector ∇f is given by,

$$M(x,y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$
$$= |\nabla f| = [g_x^2 + g_y^2]^{1/2} \rightarrow \text{Eq}^n (2)$$

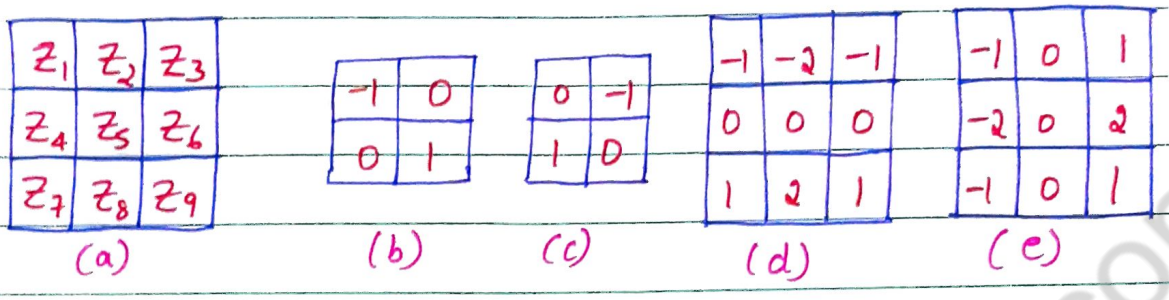
→ $M(x,y)$ is an image of the same size as the original & is called gradient image. Created when x & y are allowed to vary overall pixel locations in f.

$$|\nabla f| = [g_x^2 + g_y^2]^{1/2}$$
$$= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$
$$\simeq \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}$$

Magnitude makes $M(x,y)$ non linear. It's more suitable in some applications to use this Eq^n 3.

$$\nabla f \simeq |g_x| + |g_y|$$
$$\text{or } M(x,y) \simeq |g_x| + |g_y| \rightarrow \text{Eq}^n (3)$$

For Eg: Consider A 3x3 region of image as shown in fig 3.22 (a).



(a) (b) (c) (d) (e)

Fig 3.22 :(a) A 3x3 region of an image (the z's are intensity values). (b)-(c) Roberts Cross gradient operators. (d)-(e) Sobel operators. All the mask coefficients sum to zero, as Expected of a derivative operator.

The center gray level z_5 denotes $f(x,y)$ at an arbitrary location (x,y); z_1 denotes $f(x-1,y-1)$; & So on.

From the definition of 1st-order-derivatives we have,

$$g_x = (z_8 - z_5) \quad \text{&}$$

$$g_y = (z_6 - z_5).$$

Two other definitions proposed by Robert's use cross differences:

$$g_x = (z_9 - z_5) \text{ and } g_y (z_8 - z_6) \rightarrow \text{Eq}^n (4)$$

Substitute in Eq^n (2),

$$M(x,y) = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2} \rightarrow \text{Eq}^n (5)$$

if we use Eq^n (3), then

$$M(x,y) \approx |z_9 - z_5| + |z_8 - z_6| \rightarrow \text{Eq}^n (6)$$

These 2 Equation Can be implemented using the 2 linear filter masks in fig 3.22 (b) & (c).

These masks are referred to as Roberts cross-gradiant operators, which leads to masks of Even Size, which is inconvenient.

→ The smallest mask with central symmetry are 3x3 & the gradient can be approximated for such masks as follows:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad -(7)$$

&

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad -(8)$$

These Equations can be implemented using the masks in fig 3.22 (d) & (e). These are called Sobel operators. For Eq, Substituting g_x & g_y in Eq^n ③, yields,

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)| \quad \rightarrow \text{Eq}^n (9)$$

→ The idea of using a weight value 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point.

→ The coefficient in all masks shown sum to zero, indicating that mask will give a zero response in an area of constant intensity, as is expected of a derivative operator.

Combining Spatial Enhancement Methods.

* Use Laplacian to highlight fine detail
  → It also produces noisier results than the gradient.
* Use gradient to Enhance prominent Edges.
  → The gradient has a stronger response in ramps & steps areas than does the Laplacian.
  → The response of the gradient to noise is lower than Laplacian.

→ The response to noise can be lowered by smoothing the gradient with an averaging filter.

* Combining Laplacian & gradient operators.

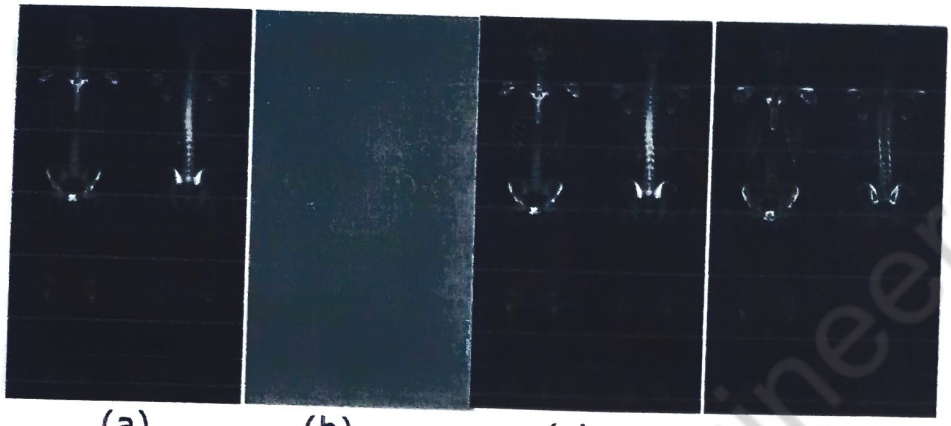
→ Smooth the gradient & multiply it by the Laplacian image (preserve details in the strong areas while reducing noise in the flat areas).

→ The above result is added to the original image.

Fig 3.23 shows an Example for combining Spatial Enhancement Methods.

Combining Spatial Enhancement Methods

(a) original (b) Laplacian, (c) a+b, (d) Sobel of (a)



(a) (b) (c) (d)

Combining Spatial Enhancement Methods

e f
g h

FIGURE 3.46 *(Continued)* (e) Sobel image smoothed with a 5×5 averaging filter. (f) Mask image formed by the product of (c) and (e). (g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)