Unit-2: Image Enhancement and Filter in Spatial Domain

Point Operation:

Point operations refer to running the same conversion operation for each pixel in a grayscale image. The transformation is based on the original pixel and is independent of its location or neighboring pixels.

Let "***r***" and "***s***" be the gray value at a point (x, y) of the input image f (x, y) and the output image g (x, y) respectively, then the point operation can be defined by the following formula.

r is the intensity at point (x,y) of original image
s is the intensity at point (x,y) of output image

s= T (r)

Where, T is the point operator of a certain gray-level mapping relationship between the original image and the output image. Point operations are often used to change the grayscale range and distribution.

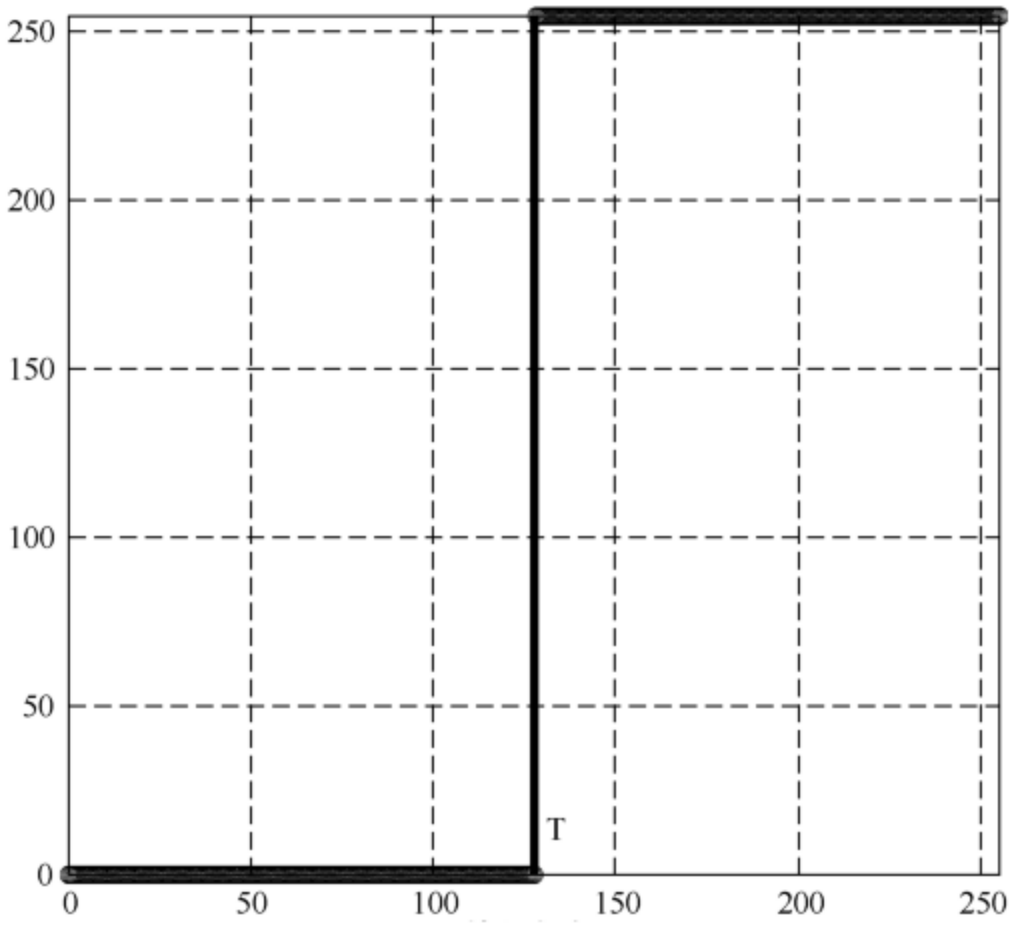
Point Operations on single images: The gray value of the output image ***"g"*** at a particular pixel ***(x, y)*** depends ONLY on the gray value of the same pixel in ***"f"***. Examples of these operations include *contrast stretching, segmentation based on gray value,* and *histogram equalization;*

tn vneko
euta input image
ko gray level value tn arko ko λn

Point Operations on multiple images: The gray value of the output pixel ***g(x, y)*** depends on the gray values of the same pixel in a set of input images ***f(x, y, tn)*** or ***f(x, y, λn).*** Examples are *segmentation based on variations in time or color, multiple-frame averaging for noise smoothing, change detection,* and *spatial detector normalization;*

Thresholding

Grayscale Threshold Transform converts a grayscale image into a black and white binary image. The user specifies a value that acts as a dividing line. If the gray value of a pixel is smaller than the dividing, the intensity of the pixel is set to 0, otherwise it is set to 255. The value of the dividing line is called the ***threshold***. The grayscale threshold transform is often referred to as thresholding, or binarization.



Here in graph, a line is dividing the graph in equally two parts from point T, the intensity of pixel below point 125 is set to 0 which is black, and above point 125 is set to 255 which is white.

Example1: find the threshold (black and white) image of input image

1	6	4
7	9	4
10	14	5

Solution: The maximum value of image pixel (L) = 16, n = 4 (*n is the number of bits and L is calculate in term of* 2^n*)*

T = L/2 = 16/2 = 8

s = (L-1), if (r >= T)

s = 0, if (r < T)

0	0	0
0	15	0
15	15	0

Example 2: Clipping with $r_1 = 3$ and $r_2 = 8$

1	6	4
7	9	4
10	14	5

Solution: here, L = 16

S = (L-1), if (3<=r<=8)

S = 0, otherwise

0	15	15
15	0	15
0	0	15

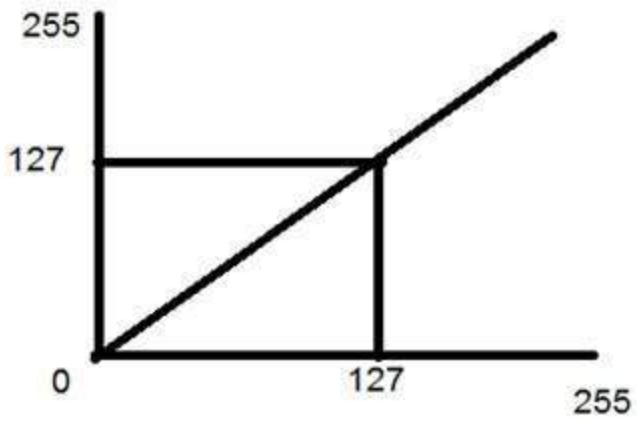
Linear Transformation

Linear transformation includes two types of transformation

- Identity Transformation
- Negative Transformation

Identity Transformation

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:



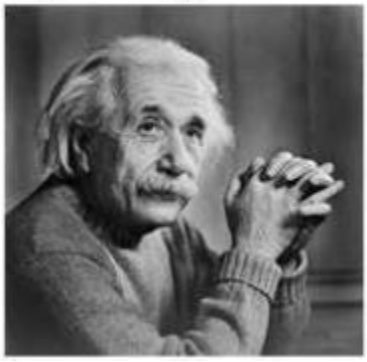
Now if you will look at this particular graph, you will see a straight transition line between input image and output image.

It shows that for each pixel or intensity value of input image, there is a same intensity value of output image. That means the output image is exact replica of the input image. It can be mathematically represented as:

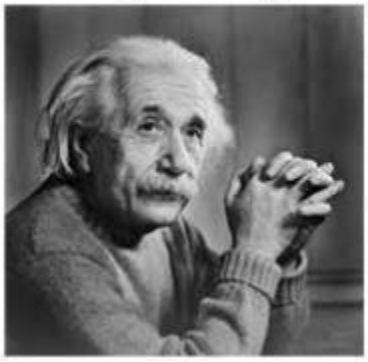
$$g(x,y) = f(x,y)$$

The input and output image would be in this case are shown below.

Input image



Output image



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.

In this case the following transition has been done.

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

Where, L = highest intensity value, r = intensity of each pixel

Example: find the negative digital image of given input image

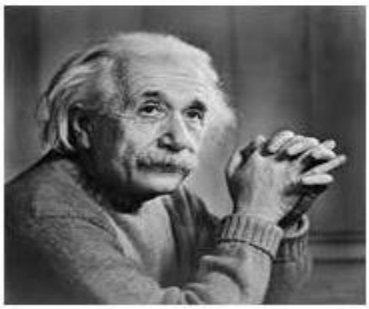
1	5	4
3	7	6
5	8	9

Solution: the highest intensity of this image is L = 16

S = (L-1) – r = (16-1) – r = 15-r

14	10	11
12	8	9
10	7	6

Let's examine the image bellow:



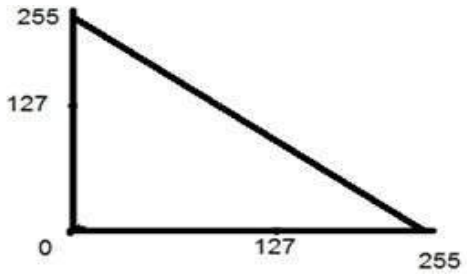
Input Image

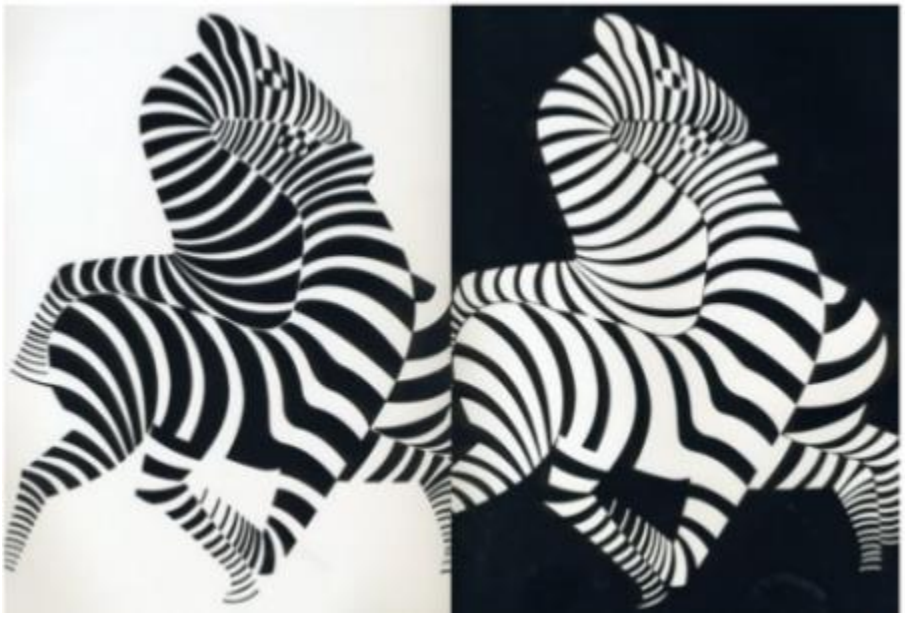
Output Image

Since the input image of Einstein is an 8 bpp (bit per pixel) image, so the number of levels (L) in this image are 256 (2^8). Putting 256 in the equation, we get this

$$s = (L-1)-r = (256-1)-r = 255 - r$$

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.





Log transformation

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.

Log curve maps a narrow range of low gray-level values in the input image into a wide range of the output levels. 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.

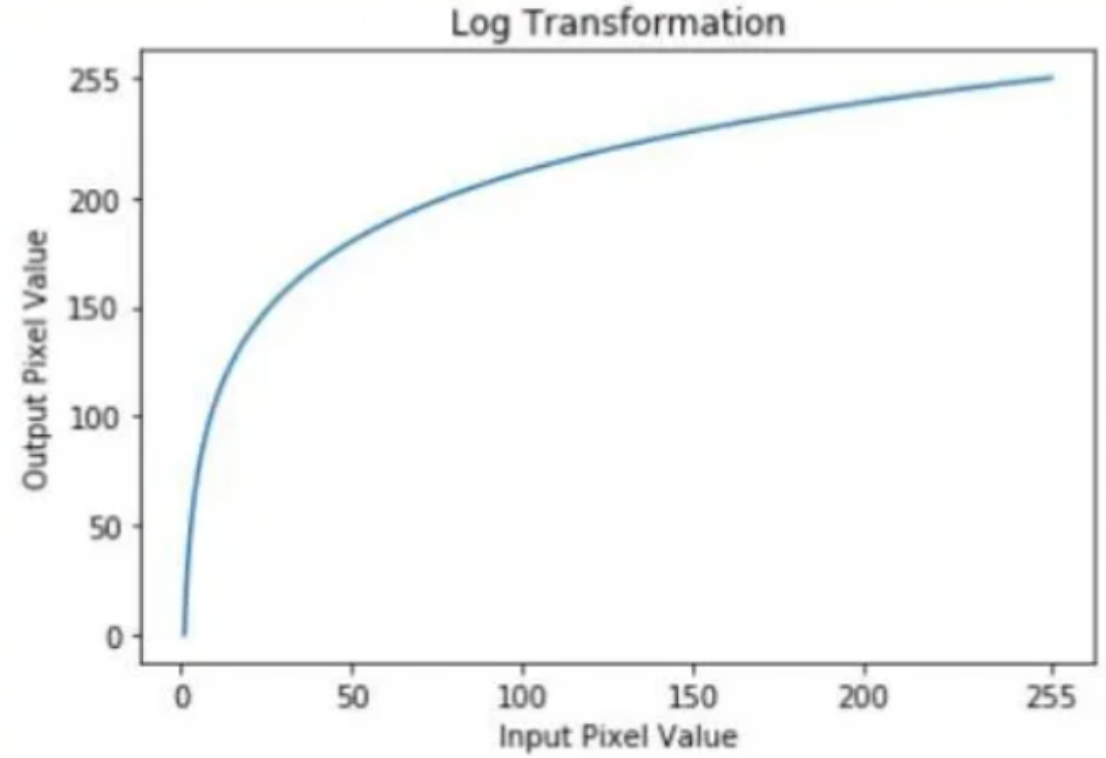
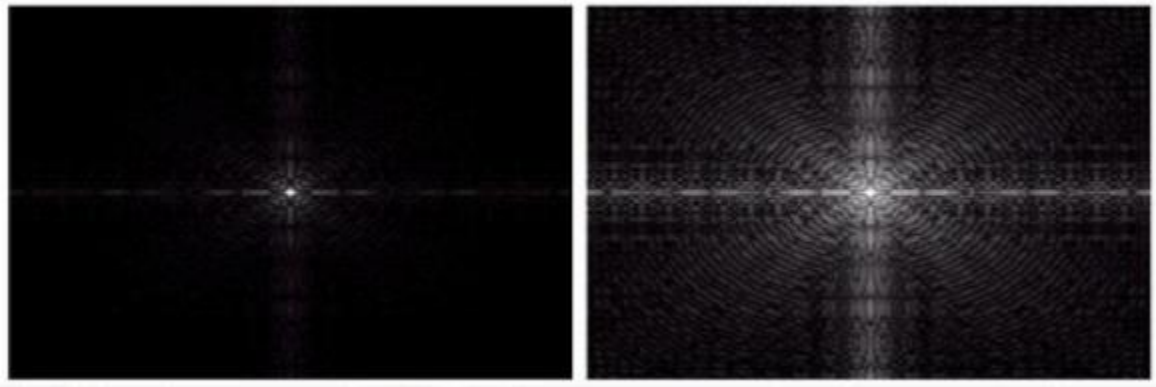


Fig: Log Transformation Graph

The value of ***c*** in the log transform adjusts the kind of enhancement you are looking for. Following figure shows the log transformation where lower intensity pixels are increased as compared to higher intensity.



Input Image Log Transform Image

Inverse log transformation is opposite to log transformation.

Example: you have given an image

110	120	90
91	94	98
90	91	99

Find the output image using logarithm transformation when,

i. C = 1
ii. C = L/(Log(1+L))

Solution:

L = 128, n = 7

We know the equation

S= c log(r+1)

For C = 1

r	S
110	$S = 1 \cdot \log_{10}(1+110) = \log(111) = 2.04 \approx 2$
120	$S = 1 \cdot \log_{10}(1+120) = \log(121) = 2.08 \approx 2$
90	$S = 1 \cdot \log_{10}(1+90) = \log(91) = 1.95 \approx 2$
91	$S = 1 \cdot \log_{10}(1+91) = \log(92) = 1.96 \approx 2$
94	$S = 1 \cdot \log_{10}(1+94) = \log(95) = 1.97 \approx 2$
98	$S = 1 \cdot \log_{10}(1+98) = \log(99) = 1.99 \approx 2$
99	$S = 1 \cdot \log_{10}(1+99) = \log(100) = 2$

The output image is:

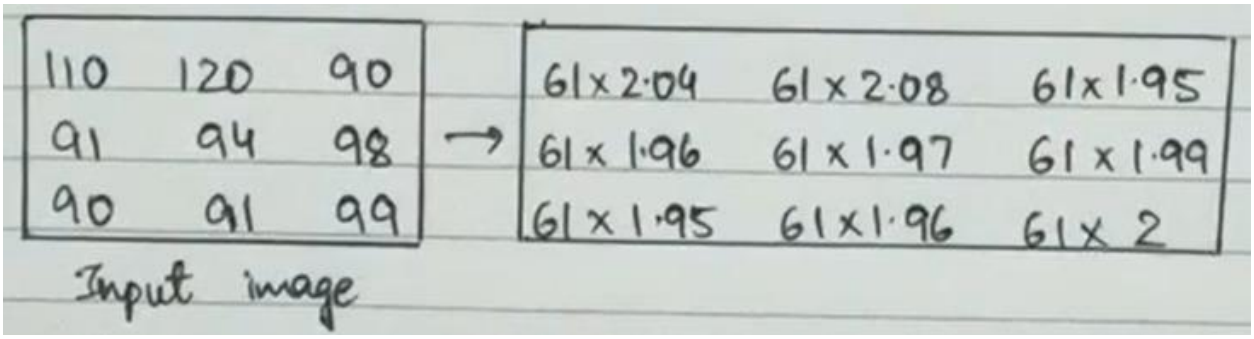
2	2	2
2	2	2
2	2	2

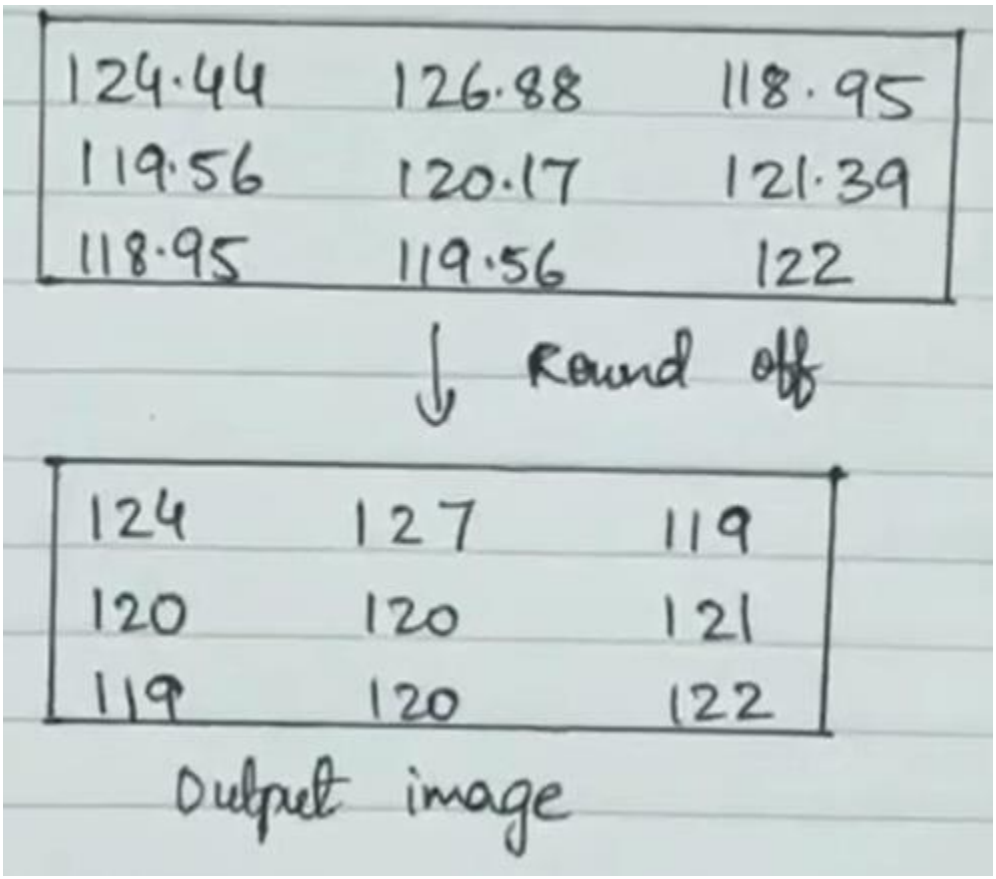
For C = L/(Log(1+L))

$$C = \frac{L}{\log_{10}(1+L)} = \frac{128}{\log_{10}(1+128)} = \frac{128}{\log_{10}(129)} = 60.66$$

The value of C = 60.66 = 61

S = 61*log (r+1)





Power – Law transformation (Gamma Transformation)

This transformation can be done by the expression:

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

Where, C and γ are positive constant, 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.

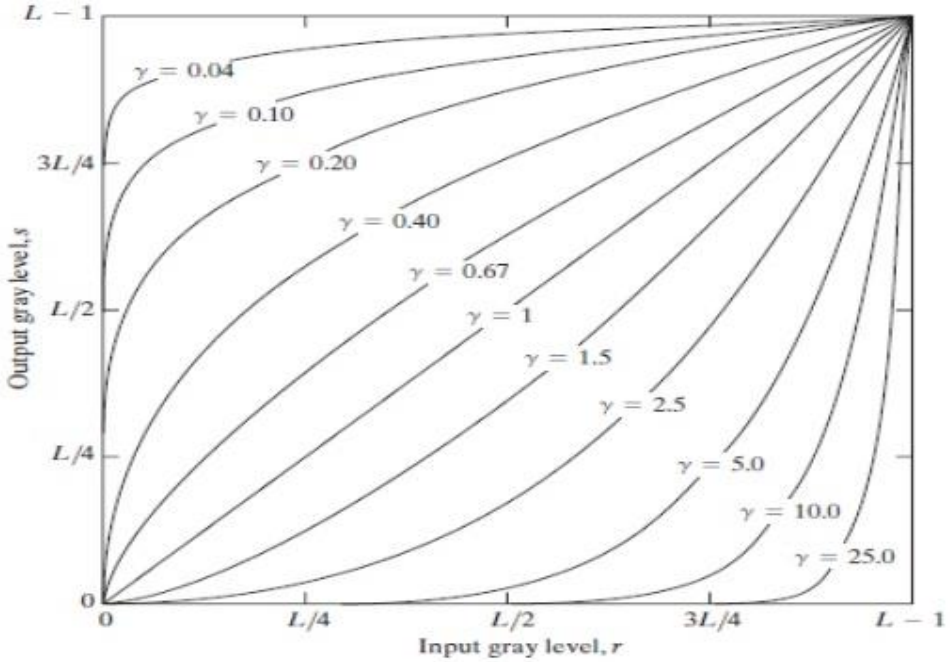
This type of transformation is used for enhancing images for different type of display devices. The gamma of different display devices is different.

For example Gamma of CRT lies in between of 1.8 to 2.5, which means the image displayed on CRT is dark.

gamma ko value 1.8 esto xa vne intensity value tw thorae xa so dark vyo image

Value of gamma is inversely proportional to the intensity level.

Different transformation curve will be obtains by varying the value of gamma as shown in graph below:



gamma ko value baddae jada output intensityghtdae gako xa

The same image but with different gamma values is displaying different gray levels.



Gamma = 10 Gamma = 8 Gamma = 6 bright image produce vyo

Example: You have given an image

110	120	90
91	94	98
90	91	99

Find the output image for c =1 and $\gamma = 0.2$

Solution:

$S = c^* r \wedge \gamma = 1^* r^{0.2} = r^{0.2}$

r	S
110	$S = 1.(110)^{0.2} = 2.56 \approx 3$
120	$S = 1.(120)^{0.2} = 2.60 \approx 3$
90	$S = (90)^{0.2} = 2.45 \approx 2$
91	$S = (91)^{0.2} = 2.46 \approx 2$
94	$S = (94)^{0.2} = 2.48 \approx 2$
98	$S = (98)^{0.2} = 2.50 \approx 3$
99	$S = (99)^{0.2} = 2.50 \approx 3$

Output image is

3	3	2
2	2	2
2	2	3

Piecewise Linear Transformation

The principal of piecewise linear transformation is, rather than using well defined mathematical function we can use user defined transform.

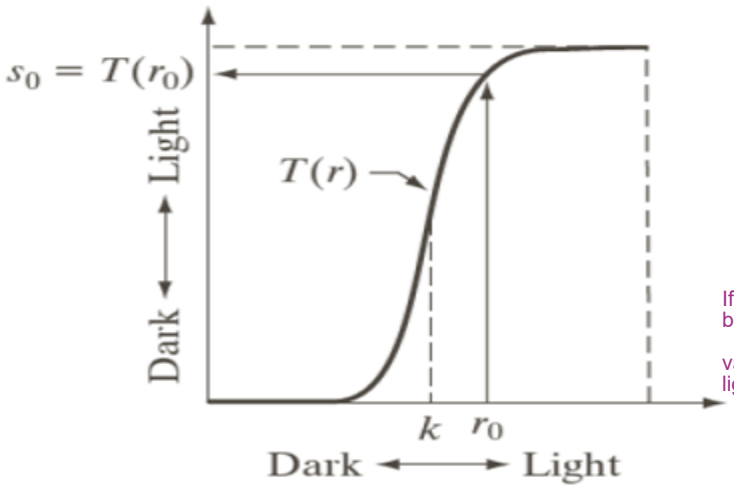
- *Principal advantage:* the form of piecewise function can be arbitrarily complex (*more options to design*), some important transformations can be formulated only as piecewise functions.
- *Principal disadvantage*: their specification requires more user input.

Some commonly used piece-wise linear transformations are:

Contrast Stretching:

The goal of the contrast-stretching transformation is to enhance the contrast between different parts of an image, that is, enhances the gray contrast for areas of interest, and suppresses the gray contrast for areas that are not of interest.

Graph below shows two possible functions:



If value less hunxa vne tyo dark ho binary ma pani value 0 dark

value increase hudae janxa light hudae gako

If ***T(r)*** has the form as shown in the figure above, the effect of applying the transformation to every pixel to generate the corresponding pixels produce higher contrast than the original image, by:

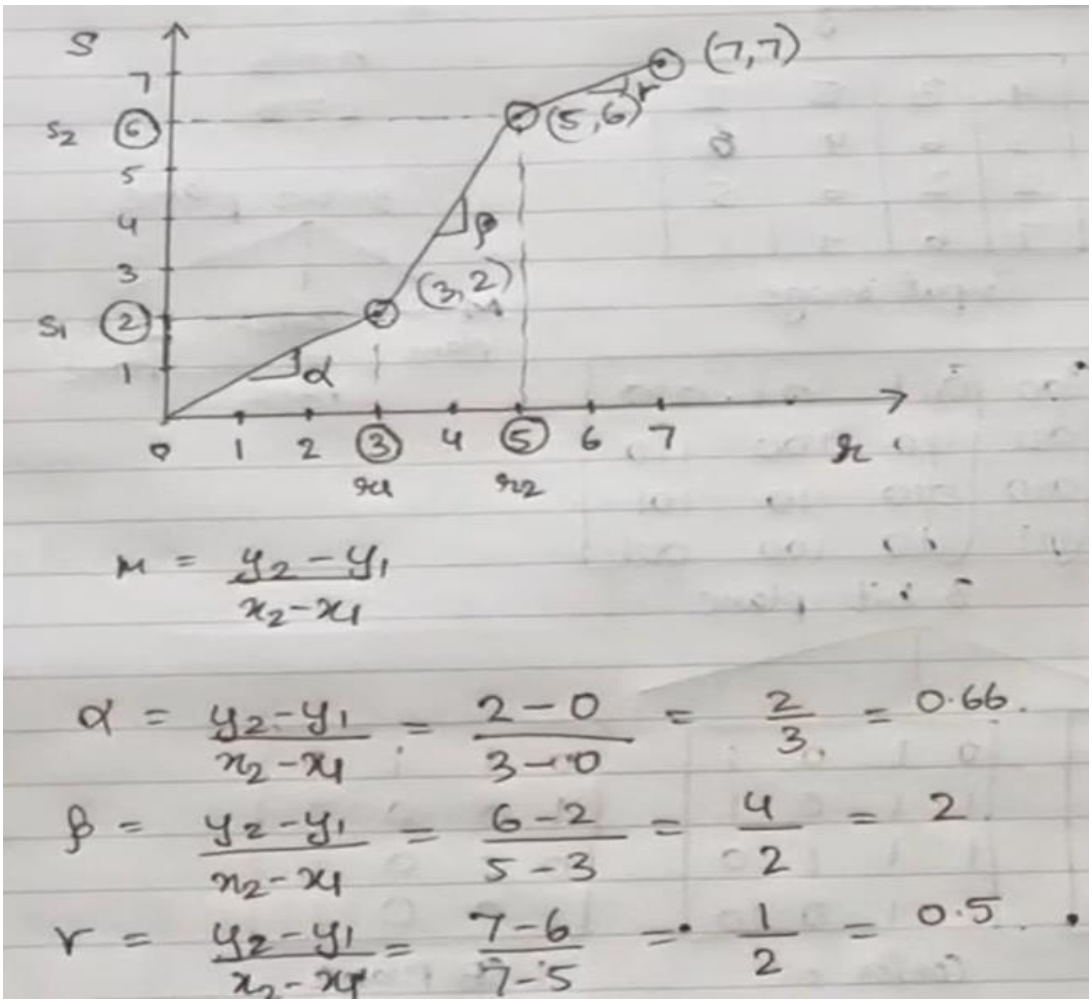
- Darkening the levels below ***k*** in the original image
- Brightening the levels above ***k*** in the original image

So, contrast stretching is a simple image enhancement technique that improves the contrast in an image by stretching the range of intensity values. This is typically used in linear function.

Example: find the transform image using contrast stretching of input image. Given that $r_1 = 3$, $r_2 = 5$, $s_1 = 2$, $s_2 = 6$

4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

Solution: let's make graph with respect to r and s



$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-0}{3-0} = \frac{2}{3} = 0.66.$$

$$\beta = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-2}{5-3} = \frac{4}{2} = 2.$$

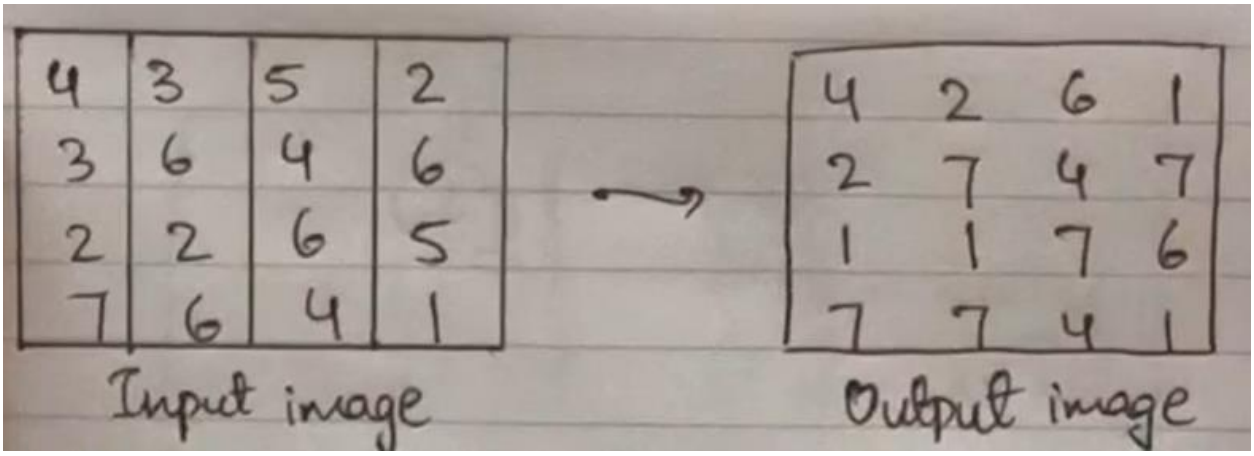
$$\gamma = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7-6}{7-5} = \frac{1}{2} = 0.5.$$

$r_1 = 3$
$r_2 = 5$

$$S = \begin{cases} \alpha \cdot r & 0 \le r < 3 \quad 0,1,2 \\ \beta \cdot (r - r_1) + S_1 & 3 \le r < 5 \quad 3,4 \\ \gamma (r - r_2) + S_2 & 5 \le r \le 7 \quad 5,6,7 \end{cases}$$

last ko ma bahek sabai less than hunxa paxadi ko wala

r	s
0	$s = \alpha \cdot r = 0.66 \times 0 = 0$
1	$s = \alpha \cdot r = 0.66 \times 1 = 0.66 \approx 1$
2	$s = \alpha \cdot r = 0.66 \times 2 = 1.32 \approx 1$
3	$s = \beta(r - r_1) + s_1 = 2(3-3) + 2 = 2$
4	$s = \beta(r - r_1) + s_1 = 2(4-3) + 2 = 4$
5	$s = \gamma(r - r_2) + s_2 = 0.5(5-5) + 6 = 6$
6	$s = \gamma(r - r_2) + s_2 = 0.5(6-5) + 6 = 6.5 \approx 7$
7	$s = \gamma(r - r_2) + s_2 = 0.5(7-5) + 6 = 7$



Gray-level slicing (Intensity-level slicing)

This technique is used to highlight a specific range of gray levels in a given image, the process, often called intensity-level slicing, and can be implemented in several ways.

- Display all high values (*say, white*) in the range of interest, and low value (*say, black*) for other intensities. As shown in Fig: a, this transformation produces a binary image.
- The second approach Brightens or darkens the desired range of intensities but leaves all other intensities in the image unchanged, as shown in Fig: b.

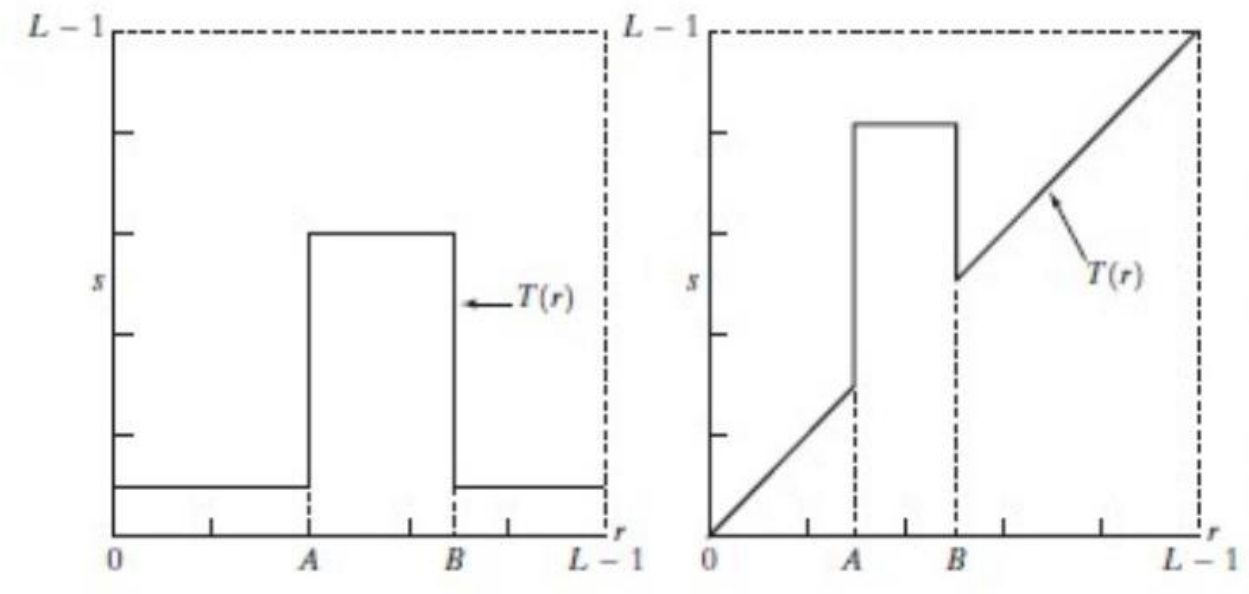
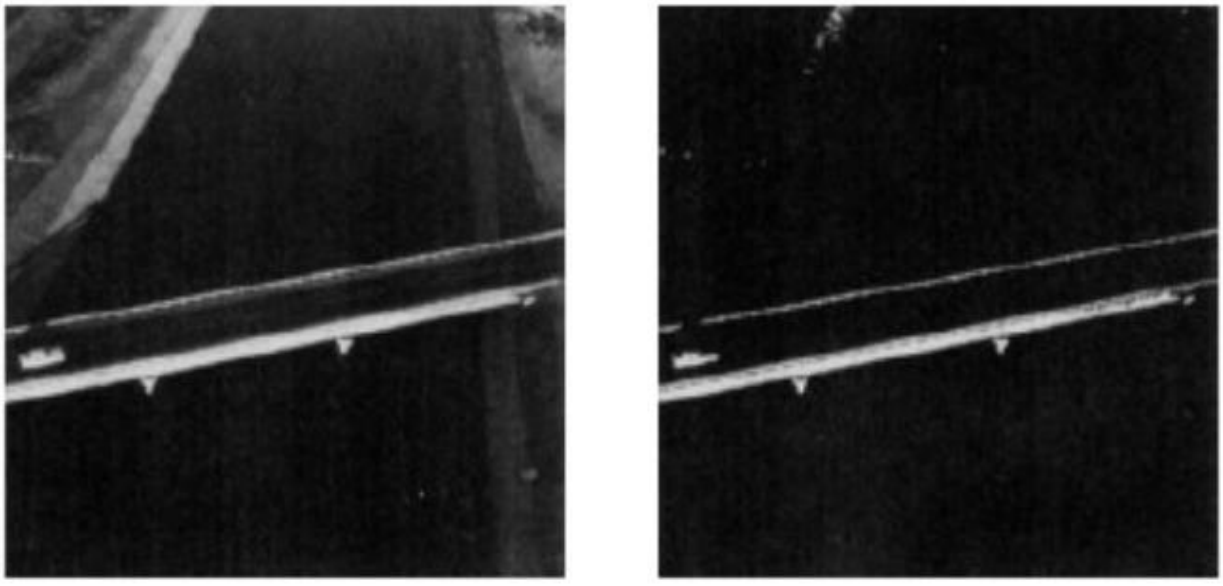


Fig: a Fig: b

Here, in Fig: a, increase the gray level in rang AB and decrease other range in constant level. But in Fig: b, increase the gray level in range AB and other range leave as it is.



Input Image Output Image

Example: you have given an image

4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

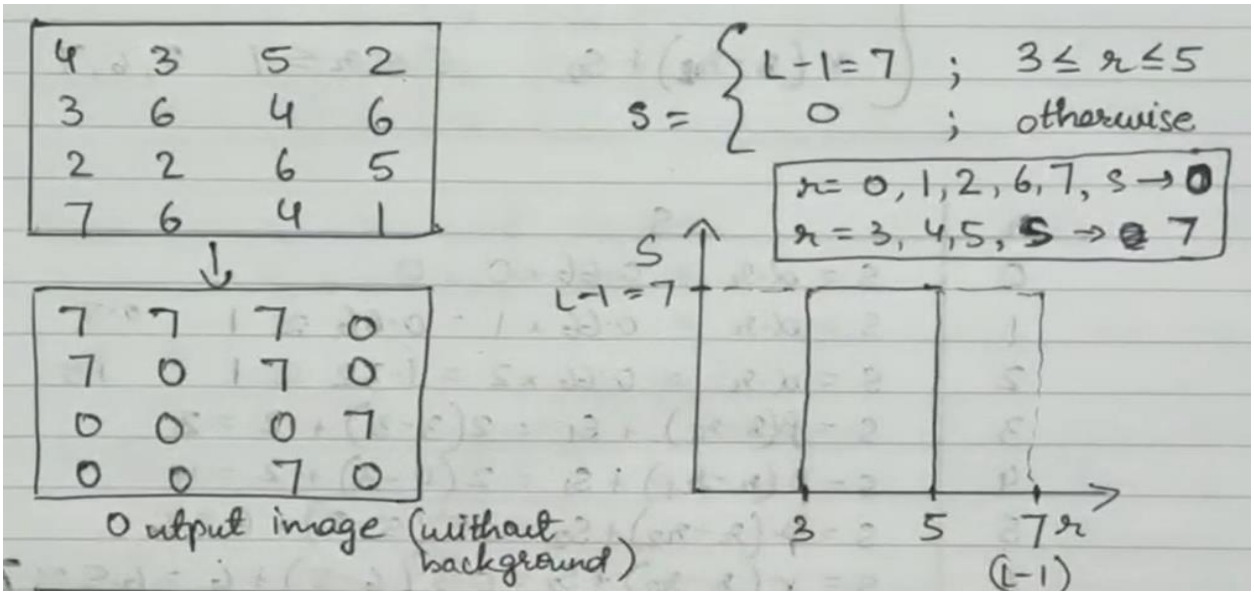
Where, $r_1 = 3$ and $r_2 = 5$

Using intensity slicing find the output

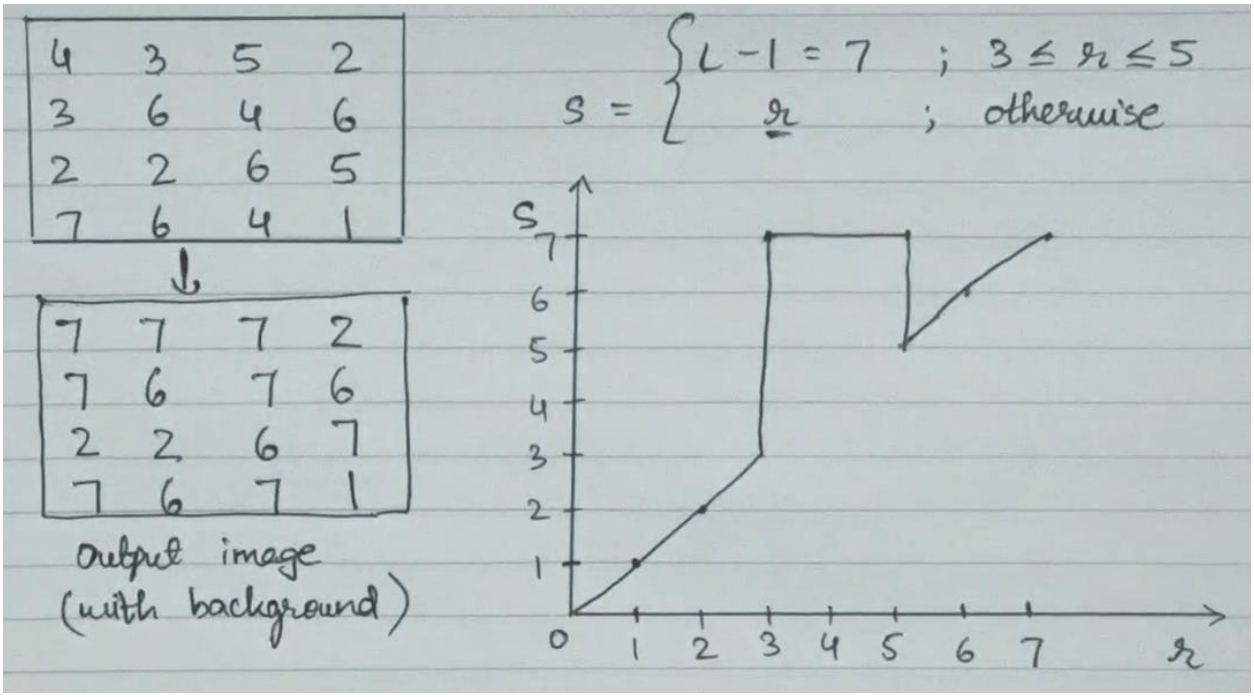
i. Without background
ii. With background

Solution:

i. Without background



ii. With background



Bit- plane slicing:

Pixels are digital number composed of bits, For example, the intensity of each pixel in a 256 gray scale image is composed of 8 bits *(i.e. 1 byte).* harek pixel ko intensity chai 8 bit le represent garnu milxani

Instead of highlighting intensity-level range, we could highlight the contribution made by each bit, this method is useful and used in image compression.

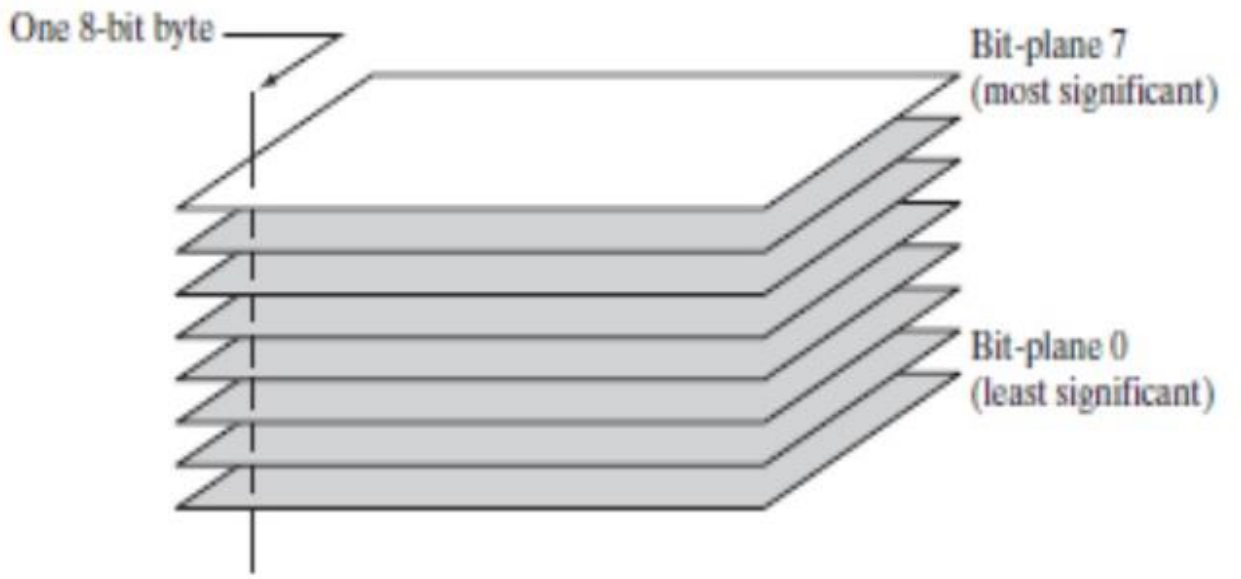
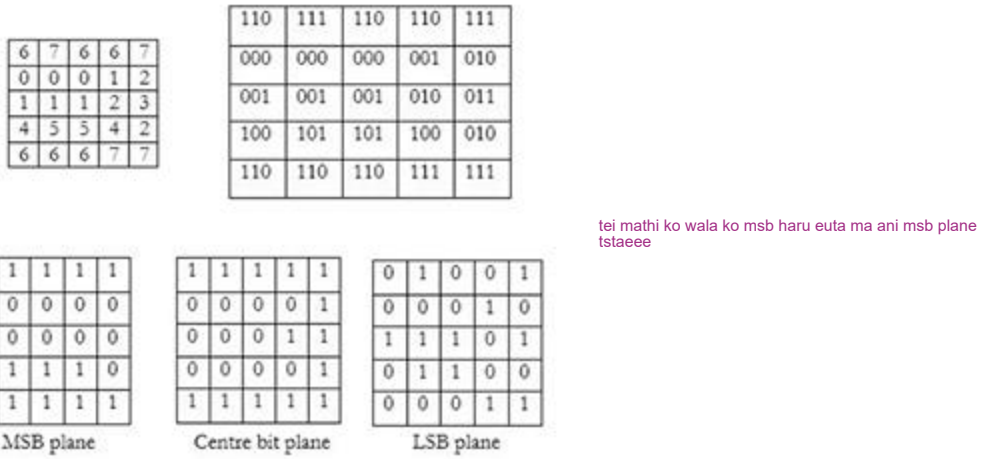


Fig: Bit-plane representation of 8-bit image

Assume that each pixel is represented by 8 bits, the image is composed of eight 1-bit panes. Plane 0 containing the lowest order bit of all pixels in the image plane remaining 7 are the higher bits. sabase low plane 0 ma tyo image ko sabai pixels ko lowest bit rkahne

Only the most significant bits contain the majority of visually significant data. The other bit planes constitute the most suitable details. Separating a digital image into its bits planes is useful for analyzing the relative importance played by each bit of the image. It helps in determining the adequacy of the number of bits used to quantize each pixel.



Example: find the 3-bit plane slicing of given image

1	6	5
3	4	7
3	2	0

Solution: here, L = 8, n = 3

Convert all the pixel value in 3 bit binary

001	110	101
011	100	111
011	010	000

MSB Plane

0	1	1
0	1	1
0	0	0

Middle Plane

0	1	0
1	0	1
1	1	0

LSB Plane

1	0	1
1	0	1
1	0	0

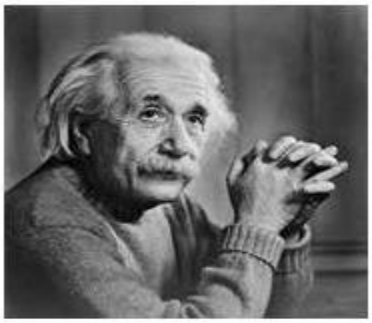
Histogram Processing

In digital image processing, the histogram is used for graphical representation of a digital image. A graph is a plot by the number of pixels for each intensity value. Nowadays, image histogram is present in digital cameras. Photographers use them to see the distribution of intensity captured.

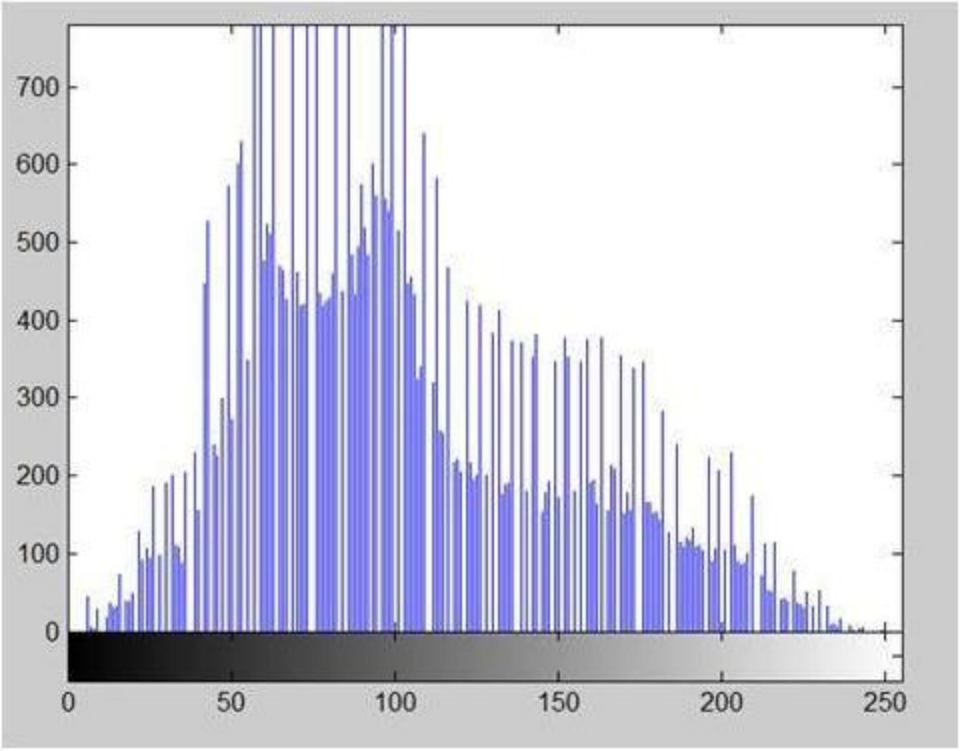
x ma chai intensity
value ani y ma chai
number of pixels ani tyo
vako haru kati wota pixels kati xa gardae jana milne vyo

In a graph, the horizontal axis of the graph is used to represent intensity variations whereas the vertical axis is used to represent the number of pixels in that particular intensity which is also called the frequency of intensity.

For example:



The histogram of the above picture of the Einstein would be something like this



The x-axis of the histogram shows the range of pixel intensity. Since it is an 8 bpp image, which means it has 2^8 (256) levels of gray or shades of gray in it. That's why the range of x axis starts from 0 and end at 255 with a gap of 50. The y-axis indicates the count of pixel in particular intensity.

As you can see from the graph, that most of the bars that have high frequency lies in the first half portion which is the darker portion. That means that the image we have got is darker. And this can be proved from the image too.

Type text here

Histogram Normalization:

Normalize a histogram is a technique that is used in transforming the discrete distribution of intensities into a discrete distribution of probabilities. To do so, we need to divide each value of the histogram by the total number of pixel.

The histogram ***h(i)*** (i=0,1,…..,255) is the probability of an arbitrary pixel taking the gray level ***i***, which can be approximated as:

$$h[i] = \frac{\text{Number of pixels of gray level } i}{\text{Total number of pixels}}$$

The cumulative density function is:

$$H[j] = \sum_{i=0}^{j} h[i], \qquad (j = 0, 1, \cdots, 255)$$

H(0) = 0/255, H(1) = 1/255, H(3) = 3/255, …….

Hence, ***H(255) = 1 (255/255)***

For a gray level image to be properly displayed on screen, its pixel values have to be within a proper range. For an 8-bit digital image there are 2^8 = 256 (from 0 to 255) gray levels. However, after applying certain processing operations to the input image, the gray levels of the resulting image are no longer necessarily within the proper range for display. In this case the image needs to be normalized or rescaled:

$$I_N = (I - \text{Min})\frac{255}{\text{Max} - \text{Min}}$$

For example: if the intensity range of the image is 50 to 180 and the desired range is 0 to 255 the process need to subtracting 50 from each of pixel intensity (*50-50, 51-50, 52-50,……,180-50*), which makes the new range 0 to 130. Then, the each pixel intensity is multiplied by ***255/(Max-Min)*** = 255/(180-50) = 255/130, making the range 0 to 255.

Histogram Equalization

Histogram equalization is used for equalizing all the pixel values of an image. Transformation is done in such a way that uniform flattened histogram is produced. Histogram equalization increases the dynamic range of pixel values and makes an equal count of pixels at each level which produces a flat histogram with high contrast image.

To make the histogram equalization of an image first we have to calculate the PMF (Probability mass function) of all the pixels in the image.

$$h[i] = \frac{\text{Number of pixels of gray level } i}{\text{Total number of pixels}}$$

Then, we have to calculate the CDF (Cumulative distribution function) which is the summation of PMF.

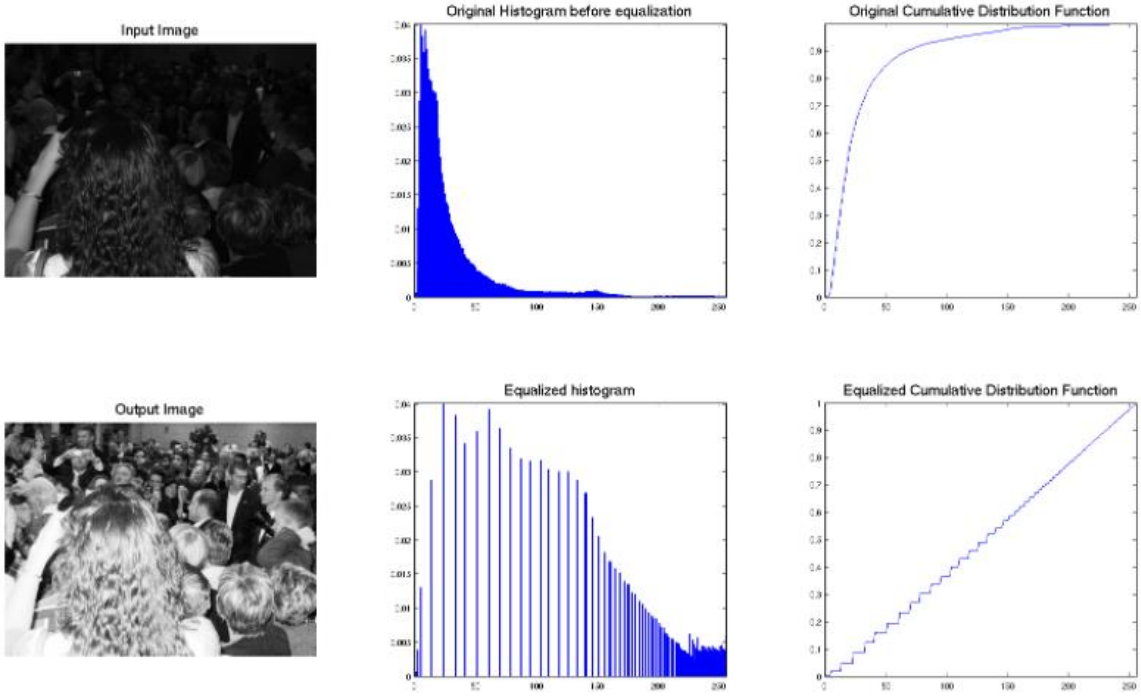
$$H[j] = \sum_{i=0}^{j} h[i], \qquad (j = 0, 1, \cdots, 255)$$

Now, the new CDF can be calculate, that is the histogram equalization of an image.

New CDF = CDF*L-1

Now plot the histogram graph for new gray level with respect to its frequency.

Following figure shows the use of histogram statistics for enhances the digital image.



Example: perform histogram equalization on the following 3-bit digital image. The gray level distribution of the image is given below:

Gray Level (r_k)	0	1	2	3	4	5	6	7
No. of pixel p_k	8	10	10	2	12	16	4	2

Total no of pixel (x) = 8+10+10+2+12+16+4+2 = 64

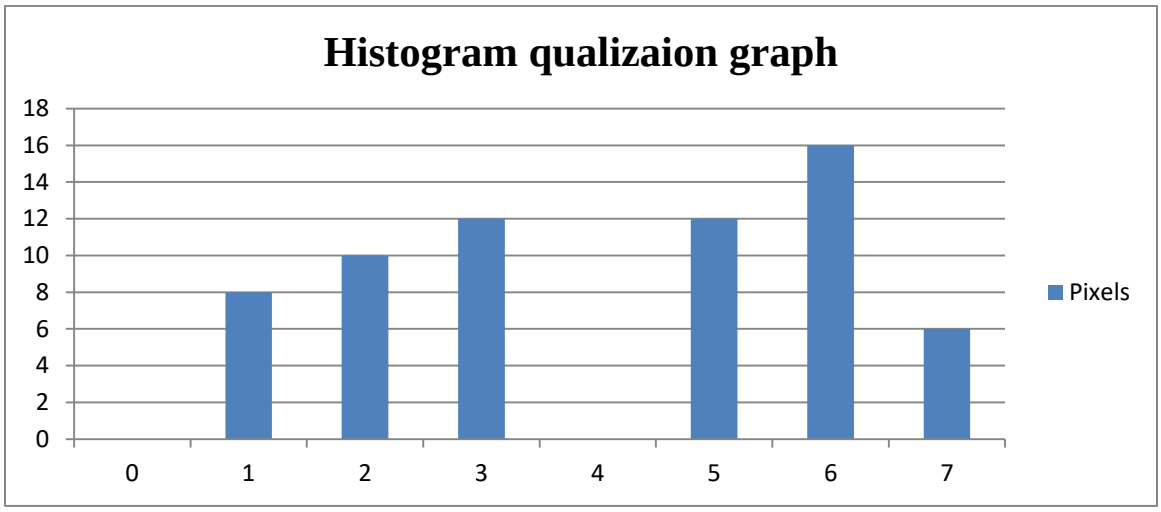
Maximum gray level of 3-bit digital image (L) = $2^3 = 8$

Now calculate the PMF, CDF, and New CDF

r_k	**p_k**	**PMF (p_k/x)**	**CDF**	**New CDF (7xCDF)**	**Round off Value**	**No. of pixel**
0	8	0.125	0.125	0.875	1	8
1	10	0.15625	0.28125	1.96875	2	10
2	10	0.15625	0.4375	3.0625	3	10
3	2	0.03125	0.46875	3.28125	3	2
4	12	0.1875	0.65625	4.59375	5	12
5	16	0.25	0.90625	6.34375	6	16

6	4	0.0625	0.96875	6.78125	7	4
7	2	0.03125	1	7	7	2

Now we can draw a histogram graph for new CDF with their corresponding pixels.



pixel

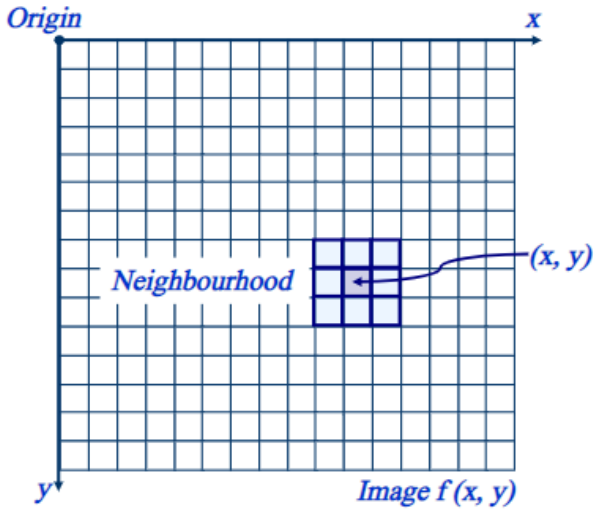
new cdf ko round off value (intensity)

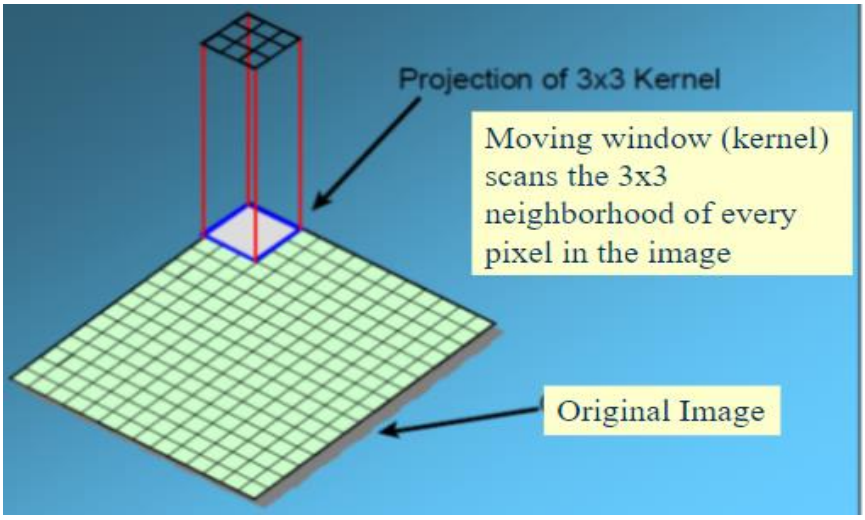
Spatial Operations

Basics of Spatial Filtering

Filtering is the process of accepting (*passing*) or rejecting certain frequency components from the digital image. A filter that passes low frequency is called a ***lowpass filter*** and passes high frequency is called ***highpass filter***. The lowpass filter is used for ***blurring*** (*smoothing*) and hithpass is used for ***shaping*** the image. The value in a filter sub-image are referred as coefficients, rather than pixels. The filtering operation can directly perform on the image itself by using spatial filters also called *spatial masks, kernels, templates, and windows*.

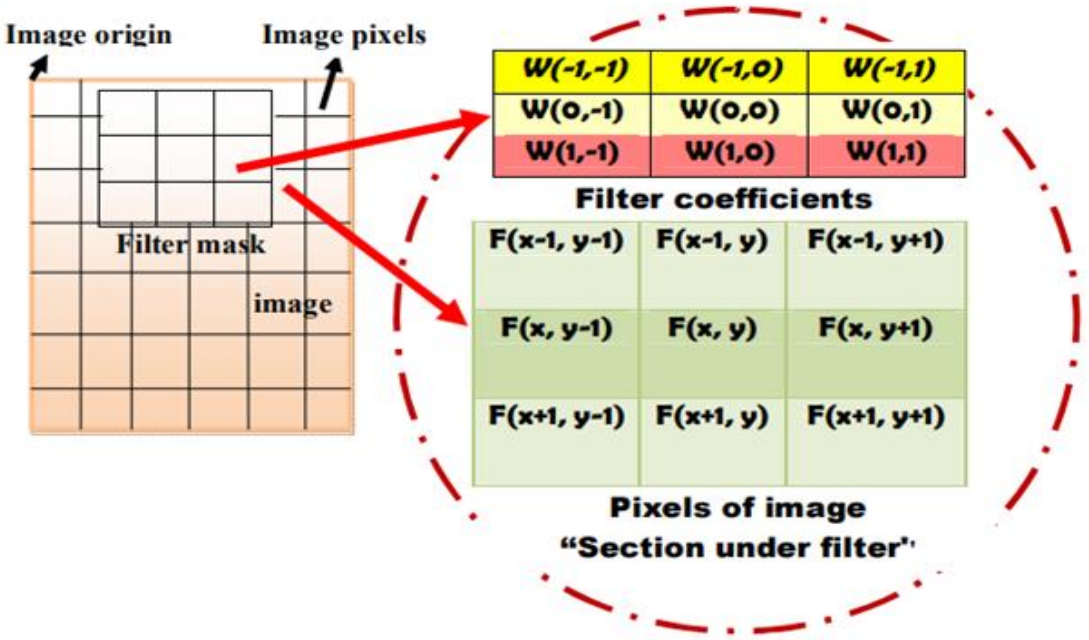
A spatial filter consists of a neighborhood (*rectangle around a central pixel*) and a pre-defined operation that is performed on the image pixel encompassed by the neighborhood. Filtering creates new pixel with the coordinates equals to the coordinates of the center of the neighborhood, and whose value is the result of filtering operation. A processed (*filtered*) picture is generated as the center of the filter visits each pixels of the input image.





Linear Spatial Filters:

Figure below illustrates the mechanics of linear spatial filtering using a 3x3 neighborhood. At any point ***(x,y)*** in the image, the response ***g(x,y),*** of the filter is the *sum of product of the filter coefficients and the image pixel encompassed by the filter.*



g(xy) = w(-1,-1) f(x-1,y-1) + w(-1,0) f(x-1,y) + w(-1,1) f(x-1,y+1) + w(0,-1) f(x,y-1) + w(0,0) f(x,y) + w(0,1)f(x,y+1) + w(1,-1) f(x+1,y-1) + w(1,0) f(x+1,y) + w(1,1) f(x+1,y+1)

Observe that the center coefficient of the filter, ***w(0,0)*** aligns with the pixel at location ***(x,y).***

For a mask of size ***m*n:***

Assume that ***m = 2a+1*** and ***n = 2b+1*** (where ***a*** and ***b*** are positive integers)

Filter of odd size, with smallest being of size ***3x3***. In general, linear spatial filtering of an image of size ***M*N*** with a filter of size ***m*n*** is given by the following equation:

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

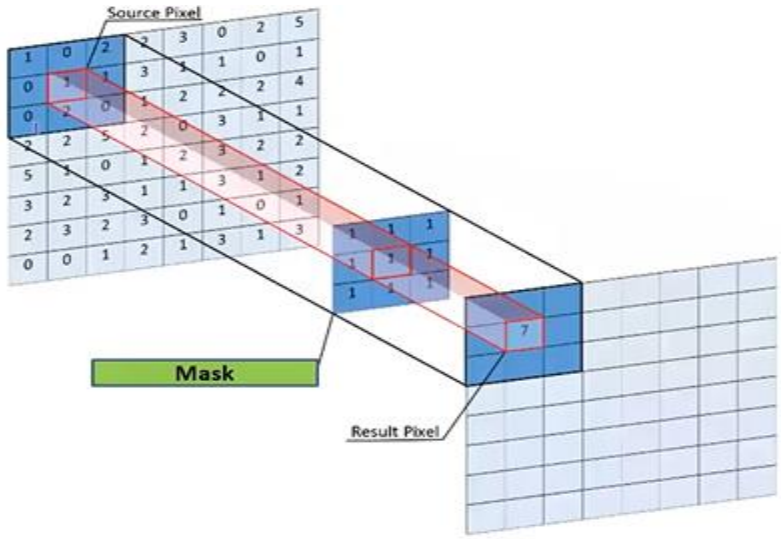
tyo tala ko operation w(-1,-1)f(x-1,y-1) esto heresi afae niskinxa

Where ***x*** and ***y*** are varied so that each pixel in ***w*** visits every pixel in ***f***.

Spatial Correlation and convolution

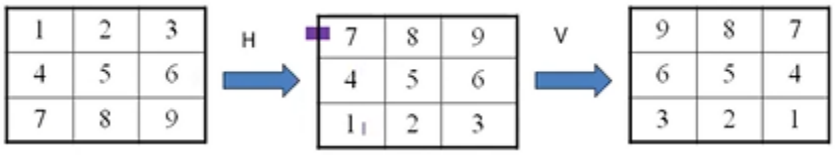
Correlation: It is the process of moving a filter mask over the image and computing the sum of products at each location. The central pixel of filtered image will replace by the resulted value. We use correlation to check similarities between two images. spatial linear filter operation nae kaam hunxa sabai result aauxa ani correlation use garerw compare garinxa input image rw output image lai

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



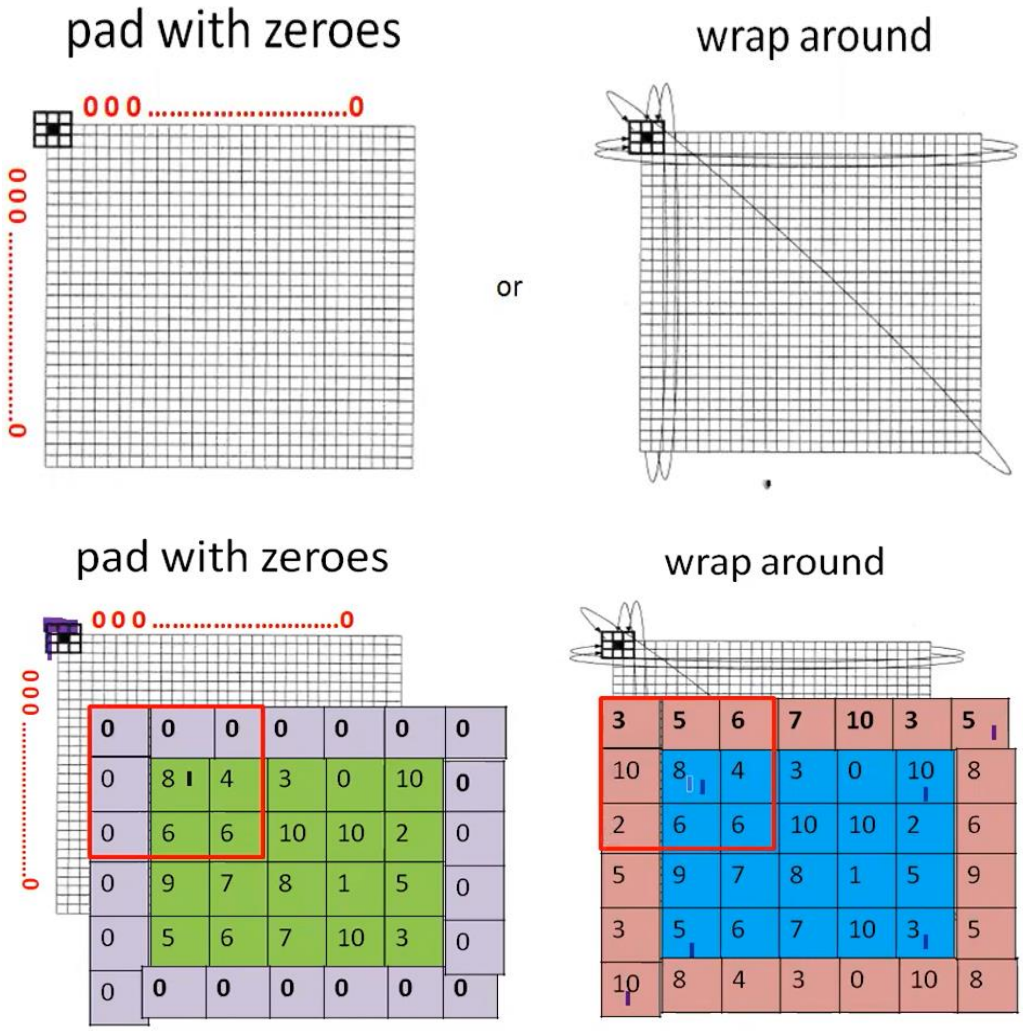
$g(x,y)= (1x1)+(0x1)+(2x1)+ (0x1)+(1x1)+(1x1)+ (0x1)+(2x1)+(0x1) = 7$

Convolution: It is also the same process as correlation, except that the filter mask is first rotated by 180^0



Then find the filtered image by processing same as correlation.

Handle pixels close to boundaries



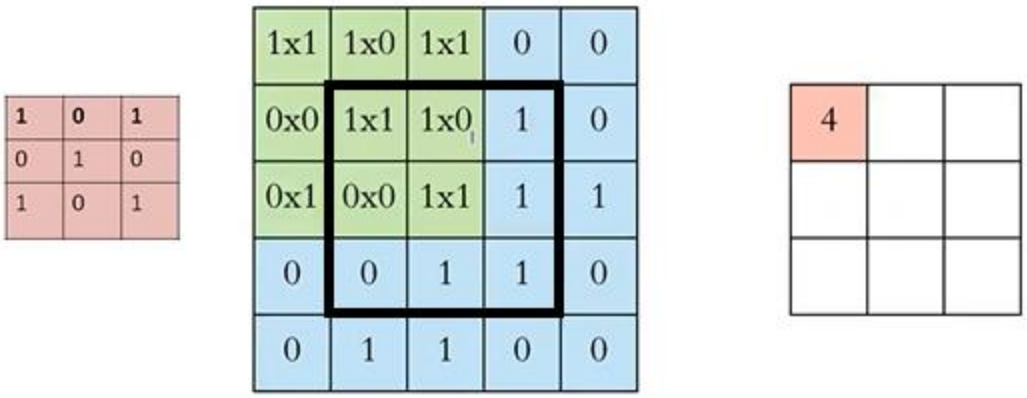
Example1: find the filtered image using Correlation, we have an image ***f(x,y)*** and a mask ***w(a,b)***

F(x,y) =

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

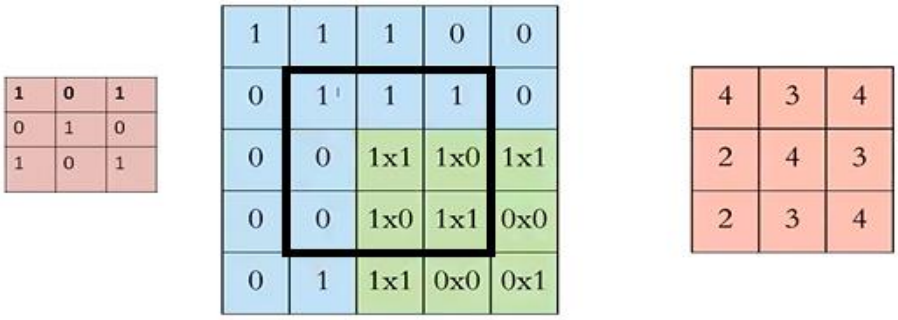
mask =

1	0	1
0	1	0
1	0	1



$g(x,y)$ = (1x1) + (1x0) +(1x1) + (0x0) +(1x1) + (1x0) +(0x1) + (0x0) + (1x1)

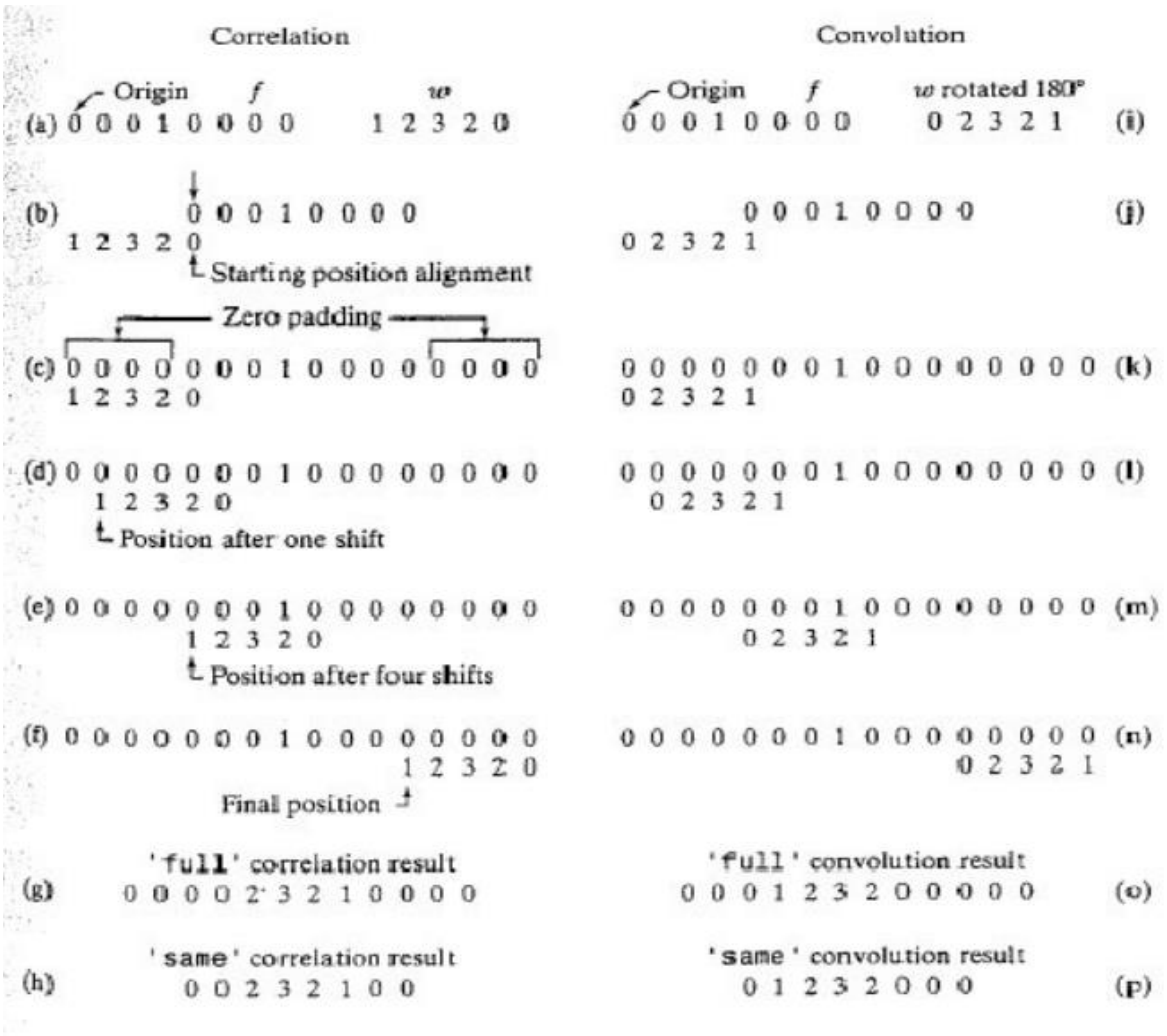
= 1+0+1+0+1+0+0+0+1 = 4



$g(x,y)$ = (1x1) + (1x0) +(1x1) + (1x0) +(1x1) + (0x0) +(1x1) + (0x0) + (0x1)

= 1+0+1+0+1+0+1+0+0 = 4

Example2: Filtered image using Correlation for 1-D image



Template Matching Correlation

template is the small portion of the original image and using this template as mask then convolution is perfomed

8	4	3	0	1	2	3	6
6	6	0	0	2	5	7	8
9	7	8	1	5	6	0	3
5	6	7	1	3	8	6	0
3	8	4	6	5	4	3	8
6	0	7	0	7	9	8	8
0		3	0	6	6	4	1
3	8	6	7	1	4	0	4

8	1	5
7	1	3
4	6	5

Template

0	0	0	0	0	0	0	0	0	0
0	8	4	3	0	1	2	3	6	0
0	6	6	0	0	2	5	7	8	0
0	9	7	8	1	5	6	0	3	0
0	5	6	7	1	3	8	6	0	0
0	3	8	4	6	5	4	3	8	0
0	6	0	7	0	7	9	8	8	0
0	0		3	0	6	6	4	1	0
0	3	8	6	7	1	4	0	4	0
0	0	0	0	0	0	0	0	0	0

g(x,y) = (8x0) + (1x0) + (5x0) + (7x0) + (1x8) + (3x4) + (4x0) + (6x6) +(5x6) = 86

New replace, central value of mask window i.e. ***8*** by ***86***

Then, move mask window by one bit forward

8	1	5
7	1	3
4	6	5

0	0	0	0	0	0	0	0	0	0
0	8	4	3	0	1	2	3	6	0
0	6	6	0	0	2	5	7	8	0
0	9	7	8	1	5	6	0	3	0
0	5	6	7	1	3	8	6	0	0
0	3	8	4	6	5	4	3	8	0
0	6	0	7	0	7	9	8	8	0
0	0		3	0	6	6	4	1	0
0	3	8	6	7	1	4	0	4	0
0	0	0	0	0	0	0	0	0	0

g(x,y) = 129, then, replace ***4*** by ***129***

Repeat this step for all the pixels of the image.

8	1	5
7	1	3
4	6	5

0	0	0	0	0	0	0	0	0	0
0	8	4	3	0	1	2	3	6	0
0	6	6	0	0	2	5	7	8	0
0	9	7	8	1	5	6	0	3	0
0	5	6	7	1	3	8	6	0	0
0	3	8	4	6	5	4	3	8	0
0	6	0	7	0	7	9	8	8	0
0	0		3	0	6	6	4	1	0
0	3	8	6	7	1	4	0	4	0
0	0	0	0	0	0	0	0	0	0

Here the intensity value of mask is same with mask window of original image. So, the mask template is taken from the original image. And the filtered value of this portion is highest. i.e. ***g(x,y) = 226***

Spatial Lowpass Filtering

Lowpass filtering is the technique which is used to remove the high frequency component like sharper edges, called noise. This technique is used to make a blur in the image commonly known as ***"blurring" or "Smoothing".*** In this filtering we remove the high frequency elements and select the low frequency elements.

Averaging or Mean Filtering

This is the one kind of lowpass filtering, which is defined as *sum of product of the average filter coefficients and the image pixel encompassed by the filter.*

This is represented by following equation:

$$g(x,y) = \frac{1}{N}\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

Where,

N *= Total number of intensity in the filter mask.*

s *and* ***t*** *= coefficient of filter mask*

Average filter mask is created by average of each coefficient, the coefficient of average filter mask is same i.e. 1. we can calculate average filter mask by following way:

$$h(x,y) = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$ Or $$h(x,y) = \begin{bmatrix}1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\end{bmatrix}$$

Now, we can apply convolution operation, which means filter mask will move over each pixel of input image to calculate the average filtered coefficient. Then final output is the average filtered image.

Example: Apply 3x3 averaging (or smoothing or lowpass) filter on given image f(x,y):

4	3	2	1
3	1	2	4
5	1	6	2
2	3	5	6

Solution: first of all we have to take an averaging filter mask of 3x3 size

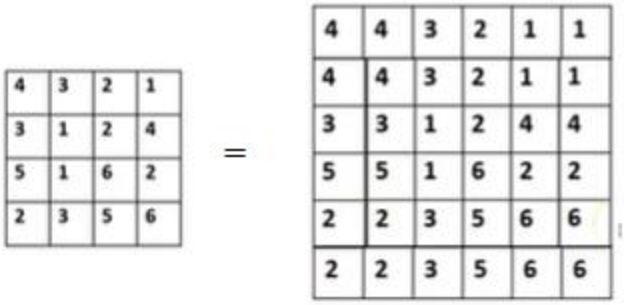
Let's take a 3x3 filter mask as:

1	1	1
1	1	1
1	1	1

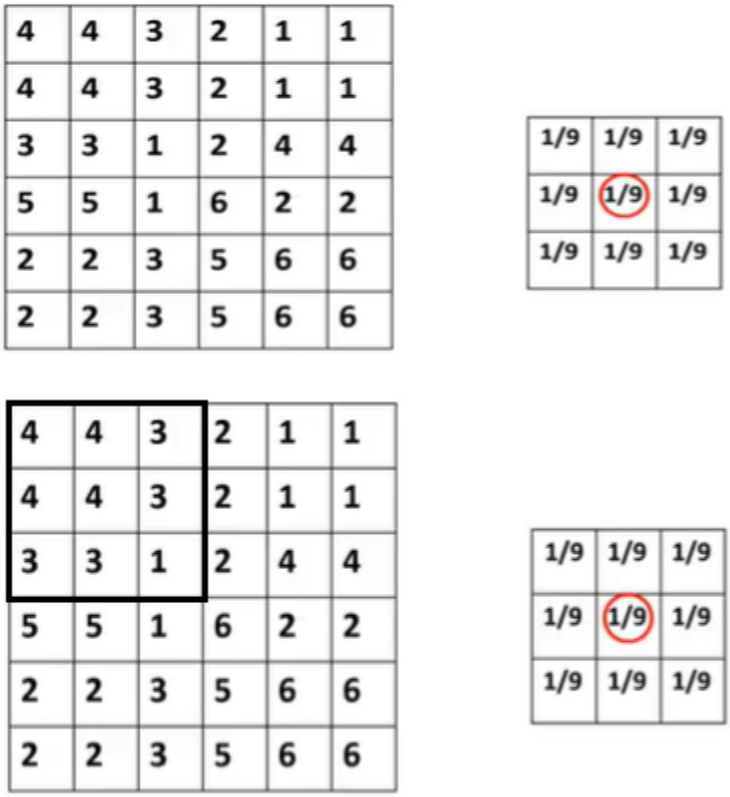
Then find the average filter mask by dividing each pixel intensity by total number of intensity i.e. 9

$$h(x,y) = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$ Or $$h(x,y) = \begin{bmatrix}1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\end{bmatrix}$$

Now, replicate the input image to get the filtered value of boundary pixels



Now, apply the convolution operation, move the 3x3 filter mark over each pixel of this input image and calculate filtered intensity value of each pixel.



G(x1,y1) = (4x1/9) + (4x1/9) +(3x1/9) + (4x1/9) + (4x1/9) + (3x1/9) + (3x1/9) + (3x1/9) + (1x1/9)

= 3.2

4	4	3	2	1	1
4	4	3	2	1	1
3	3	1	2	4	4
5	5	1	6	2	2
2	2	3	5	6	6
2	2	3	5	6	6

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

G(x2,y2) = (4x1/9) + (3x1/9) +(2x1/9) + (4x1/9) + (3x1/9) + (2x1/9) + (3x1/9) + (1x1/9) + (2x1/9)

= 2.9

G(x3,y3) = (3x1/9) + (2x1/9) +(1x1/9) + (3x1/9) + (2x1/9) + (1x1/9) + (1x1/9) + (2x1/9) + (4x1/9)

= 2.1

G(x4,y4) = (2x1/9) + (1x1/9) +(1x1/9) + (2x1/9) + (1x1/9) + (1x1/9) + (2x1/9) + (4x1/9) + (4x1/9)

= 2

Like this move filter mask to last region i.e. g(x16,y16)

4	4	3	2	1	1
4	4	3	2	1	1
3	3	1	2	4	4
5	5	1	6	2	2
2	2	3	5	6	6
2	2	3	5	6	6

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

G(x16,y16) = (6x1/9) + (2x1/9) +(2x1/9) + (5x1/9) + (6x1/9) + (6x1/9) + (5x1/9) + (6x1/9) + (6x1/9)

= 4.8

Now the resulted filtered image is:

4	4	3	2	1	1
4	3.2	2.9	2.1	2	1
3	3.2	3	2.4	2.6	4
5	2.7	3.1	3.3	4.1	2
2	2.7	3.5	4.1	4.8	6
2	2	3	5	6	6

=

4	4	3	2	1	1
4	3	3	2	2	1
3	3	3	2	3	4
5	3	3	3	4	2
2	3	4	4	5	6
2	2	3	5	6	6

We can crop the image to get in original size as:

4	3	2	1
3	1	2	4
5	1	6	2
2	3	5	6

=

3	3	2	2
3	3	2	3
3	3	3	4
3	4	4	5

Input Image Filtered Image

This is the final lowpass or average filtered image.

Alternatively we can use zero padding instead of replication to get boundary pixel value as:

0	0	0	0	0	0
0	4	3	2	1	0
0	3	1	2	4	0
0	5	1	6	2	0
0	2	3	5	6	0
0	0	0	0	0	0

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

And then, apply the convolution operation.

Weighted average filter:

This is also a kind of lowpass filter, which is defined as *sum of product of the weighted average filter coefficients and the image pixel encompassed by the filter*.

This is represented by following equation:

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

16 ko lagi mask ko sabai jodeko ho k

In this filer technique, the filter mask is multiply by different coefficient, the value of center coefficient will be higher than other coefficient to give more importance in the calculation of average. Other pixels are inversely weighted as a function of their distance from center of the mask. The diagonal neighbors are further away from center than orthogonal neighbors. Thus the weight of diagonal neighbor is less then orthogonal neighbors.

The filter mask of weighted average filtering is given bellow:

$$h(x,y) = \frac{1}{16}$$

1	2	1
2	4	2
1	2	1

jasko distance dherae tesko weight kam

Now, apply the convolution operation, i.e. filter mask will move over each pixel of input image to calculate the weighted average filtered coefficient. Then final output is the average filtered image.

Type text here

Nonlinear (or Order-Statistic) Filters: mask present hudenw raexa

The response or output of this filter is based on ordering (ranking) the pixels contained in the image area encompassed by the filter and then replacing the value of the center pixel with the value determined by ranking result. The best known filter in this category is *median filter*.

Mean/Box Filter:

Similarly we can also calculate the mean filter by calculating the mean value of the sliding window in the input image and replacing the center value of the window by mean value.

Let's take the same example:

2	4	4
1	3	6
8	3	6

New, calculate the mean value of the given image

2 4 4 1 3 6 8 3 6

Mean = (2+4+4+1+3+6+8+3+6)/9 = 37/9 = 4.11 = 4

Now center value of image i.e. 3 is replaced by 4. Output image is

2	4	4
1	4	6
8	3	6

Median Filter:

In this filter, a window slides along the input image, and intensity value of the center pixel of that pixel window is replaced by the median intensity value of the pixels within that window. The median filter is quite popular because for certain type of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filter of similar size. Median filters are particularly effective in the presence of impulse noise also called salt-and-pepper noise because of its appearance as white and black dots superimposed on an image.

Median is calculated by ordering the given set of value in ascending order and the middle value of the order is median.

Example: we have an image with intensity values as

2	4	4
1	3	6
8	3	6

Now, order the all intensity value of image in ascending order

1 2 3 3 ④ 4 6 6 8

Median value is 4

Then, replace the center value of window i.e. 3 by 4. And output of image is

2	4	4
1	4	6
8	3	6

Minimum Filter

The transformation replaces the central pixel with the darkest one in the running window. This filter is used to remove the salt noise (*i.e. white pixel noise*) from the image.

For example: the input image is

2	4	4
1	3	6
8	3	6

Intensity value of the image is 1 2 3 3 4 4 6 6 8

The minimum value of the image is 1, then, center value of image i.e. 3 is replaced by 1. The output image is

2	4	4
1	1	6
8	3	6

Maximum Filter

The maximum and minimum filters are shift-invariant. Whereas the minimum filter replaces the central pixel with the darkest one in the running window, the maximum filter replaces it with the lightest one. This filter is used to remove paper noise (*i.e. black pixel noise*) from the image.

For example: the input image is

2	4	4
1	3	6
8	3	6

Intensity value of the image is 1 2 3 3 4 4 6 6 8

The maximum value of the image is 8, then, center value of image i.e. 3 is replaced by 8. The output image is

2	4	4
1	8	6
8	3	6

Assignment: Consider the following image, what will be the new value of the pixel (2, 2) if smoothing is done using 3x3 neighborhood.

1	1	4	5	3
4	1	2	3	5
2	3	5	7	8
4	5	6	6	6
7	5	4	3	5

Calculate the filter value using

a. Mean filter
b. Weighted average filter
c. Median filter
d. Minimum filter
e. Maximum filter

High-Pass Filtering (Sharpening)

A high-pass filter can be used to make an image appear sharper. A high pass filter tends to retain the high frequency information within an image while reducing the low frequency information. The kernel of the high pass filter is designed to increase the brightness of the center pixel relative to neighboring pixels. The kernel array usually contains a single positive value at its center, which is completely

surrounded by negative values. If there is no change in intensity, nothing happens. But if one pixel is brighter than its immediate neighbors, it gets boosted.

The sum of values of filter mask should be zero.

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

For better result we can take average filter mask i.e. each value is divide by total number of pixel, but it's optional

New, we can perform convolution operation to input image using this filter mask for highpass filter.

Class work: considering following input image find the filtered output image using highpass filtering operation with 3x3 filter mask.

21	19	17	25	28
71	76	73	68	59
153	164	164	157	155
200	201	190	185	180
205	210	215	230	232

Unsharp Masking and High-boost filtering

It is often desirable to emphasize high frequency components representing the image details (*by means such as sharpening*) without eliminating low frequency components representing the basic form of the signal. In this case, the high-boost filter can be used to enhance high frequency component while still keeping the low frequency components. This is popularly used in printing and publishing industry to sharpen the image.

The process of subtract an unsharp(smooth) version of an image from the original image is called unsharp masking which consist of following steps:

- Blur the original image
- Subtract the blurred image from the original image the (*resulting difference is called mask*)
- Add mask to original image

Let f'(x,y) denotes the blurred image, unsharp masking is express in the following equation, first we obtain the mask:

$g_{mask}(x,y) = f(x,y) - f'(x,y)$ (*Mask = original image – blurred image*)

Add a weighted portion of the mask back to the original image

$f_{unsharp}(x,y) = f(x,y) + k * g_{mask}(x,y)$ Where, k is a constant

When, k=1, technique is called unsharp masking.

tyo mask nae unsharp masking bata nikaleko ho

When k>1, technique is called high-boost masking.

Generalization of unsharp masking is called highboost filtering.

$f_{hb}(x,y) = f(x,y) + k * g_{mask}(x,y)$

Following graph shows the illustration of the mechanics of unsharp masking

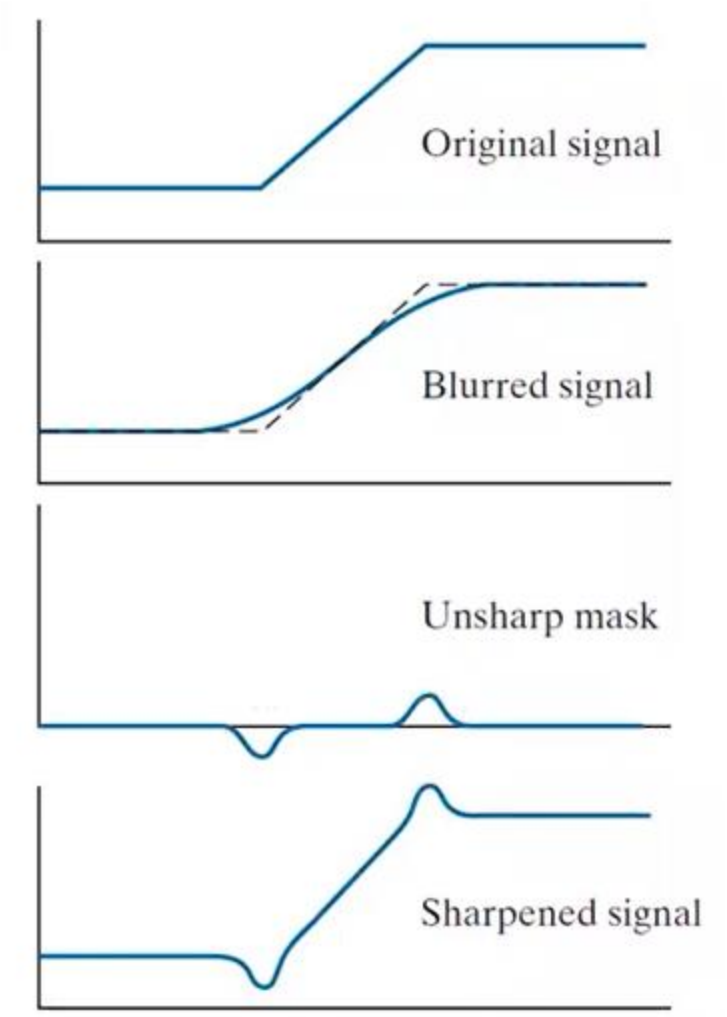


Fig: a) Original Signal b) Blurred signal with original shown dashed for reference c) Unsharp mask d) Sharpened signal obtained by adding (c) to (b)

Now, by the help of masking equation we can create filter mask for unsharp and high-boost filtering.

0	-1	0
-1	a+4	-1

0	-1	0

-1	-1	-1
-1	a+8	-1
-1	-1	-1

Where, a=1, this is unsharp filter mask and a>1, this is high-boost filter mask.

Example: Apply high-boost filter on the image given below on the center point. Use the mask with a = 1.7

1	2	3
4	5	6
7	8	9

The filter mask is

-1	-1	-1
-1	a+8	-1
-1	-1	-1

F_{hb} = -1(1+2+3+4+6+7+8+9) + 5(1.7+8)

= -1(40) + 5(9.7)

= -40 + 48.5

= 8.5

Now, center pixel of original image i.e. 5 is replaced by 8.5. The output image is:

1	2	3
4	8.5	6
7	8	9

Gradient based filter or first derivative filter

An image gradient is a directional change in the intensity or color in an image. The gradient of the image is one of the fundamental building blocks in image processing.

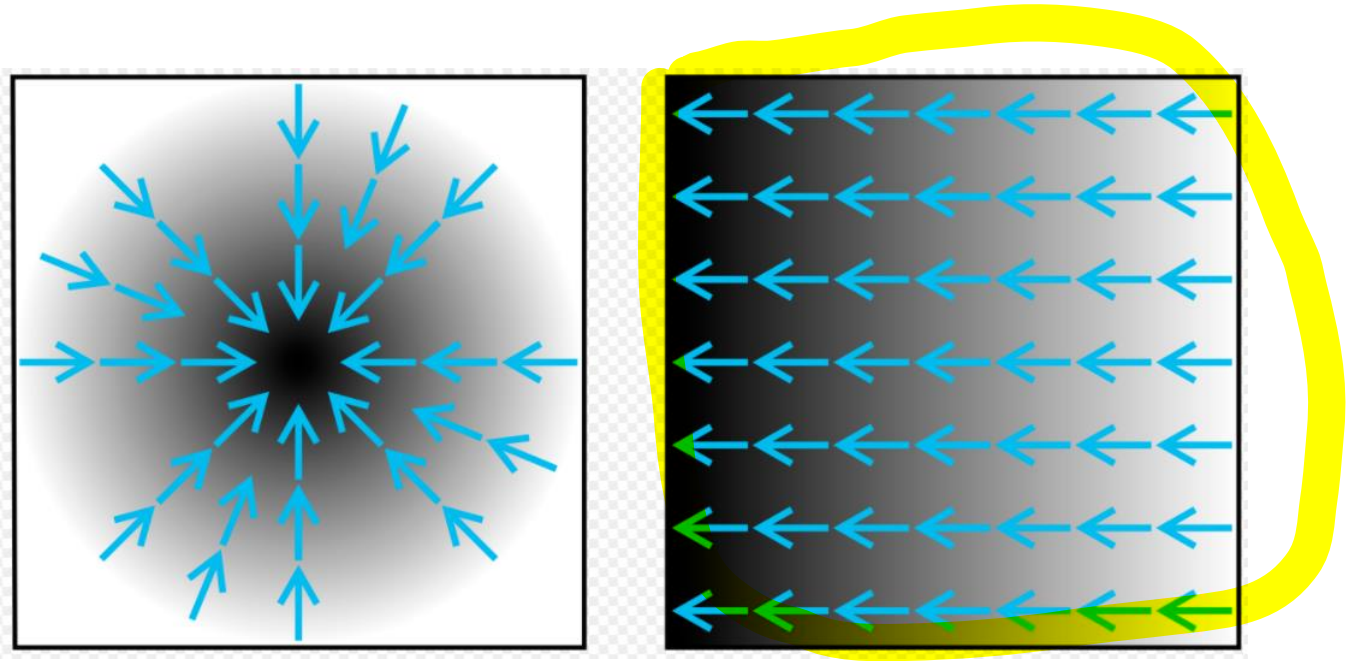


Fig: Two types of gradients, with blue arrows to indicate the direction of the gradient Dark areas indicate higher values

The first order derivatives in image processing are implemented using the magnitude of the gradient. This magnitude expresses the rate at which the gradient changes in direction. This technique is used to detect the edge of the image.

The formula for the 1st order derivative is:

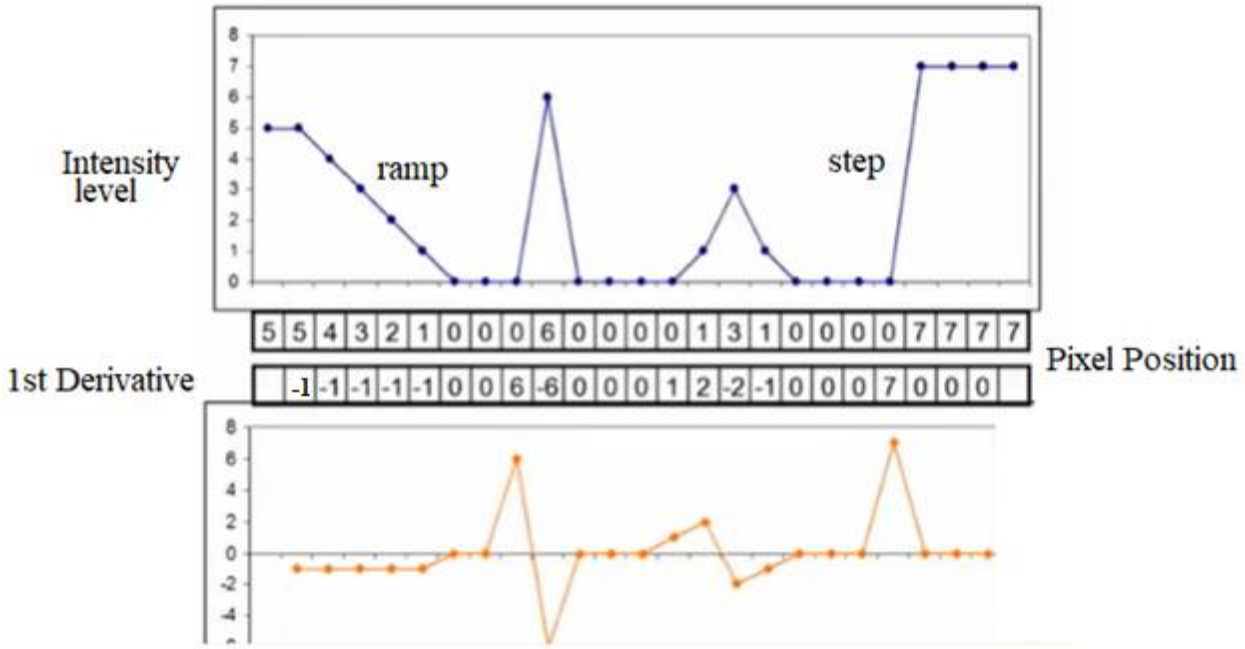
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

For the 1st derivative following condition should be satisfy

- Difference must be zero in areas of constant intensity
- Difference must be nonzero at the onset (*initial*) of an intensity step or ramp
- Difference must be nonzero along the ramp

yeslai difference must be nonzero at step or ramp vndine

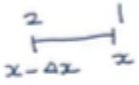
This function calculates the difference between two subsequent values of pixel and measure the rate of change of function.



The derivative is used to calculate the gradient differences between two consequent pixels. There are three types of differences:

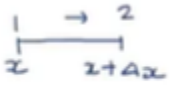
gradient vneko directional change euta direction ho rw tw points haru vnerw garna miliraxa x x+1 esto sab

Backward Difference:



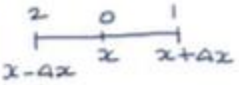
$$= [f(x) - f(x-\Delta x)] / \Delta x$$

Forward Difference:



$$= [f(x+\Delta x) - f(x)] / \Delta x$$

Central Difference:



= [f(x+ Δx) – f(x-Δx)] / 2Δx

Gradient operators are represented as:

$$\text{Vector: } \nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

$$\text{Magnitude: } \nabla f = \text{mag}(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

$$\text{Direction of gradient: } \tan^{-1}\left(\frac{Gy}{Gx}\right)$$

There are three types of gradient operators

Robert (Cress Gradient) operator

This operator find the gradient difference in cross or diagonal pixel position.

The filter mask of Robert operator is:

$$g(x) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad g(y) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

yo ali differnet tarika le ghumyaxa tyakka switch jsto vaxa g(y) ma col 1 rw col 2 switch

By using one of this filter mask, we can perform convolution operation on input image to calculate g(x) and g(y).

The magnitude of the gradient is calculated by equation

$$\text{Magnitude: } \nabla f = \text{mag}(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

The direction of the gradient is calculated by equation

Direction of gradient: $\tan^{-1}\left(\frac{Gy}{Gx}\right)$

Prewitt Operator

This method takes the central difference of the neighboring pixels.

The filter mask of Prewitt Operator is:

g(x) =

-1	-1	-1
0	0	0
1	1	1

g(y) =

-1	0	1
-1	0	1
-1	0	1

By using one of this filter mask, we can perform convolution operation on input image to calculate g(x) and g(y).

The magnitude of the gradient is calculated by equation

Magnitude: $\nabla f = \text{mag}(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$

The direction of the gradient is calculated by equation

Direction of gradient: $\tan^{-1}\left(\frac{Gy}{Gx}\right)$

Sobel Operator

This method also takes the central difference of the neighboring pixels. It provides both a differentiating and a smoothing effect.

The filter mask of Sobel Operator is:

g(x) =

-1	-2	-1
0	0	0
1	2	1

g(y) =

-1	0	1
-2	0	2
-1	0	1

By using one of this filter mask, we can perform convolution operation on input image to calculate g(x) and g(y).

The magnitude of the gradient is calculated by equation

$$\text{Magnitude: } \nabla f = \text{mag}(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

The direction of the gradient is calculated by equation

$$\text{Direction of gradient: } \tan^{-1}\left(\frac{Gy}{Gx}\right)$$

Class work: Apply Robert, Prewitt and Sobel operators in the pixel (1,1) for both x and y coordinates in the following image. And also calculate magnitude of each operation.

50	50	100	100
50	50	100	100
50	50	100	100

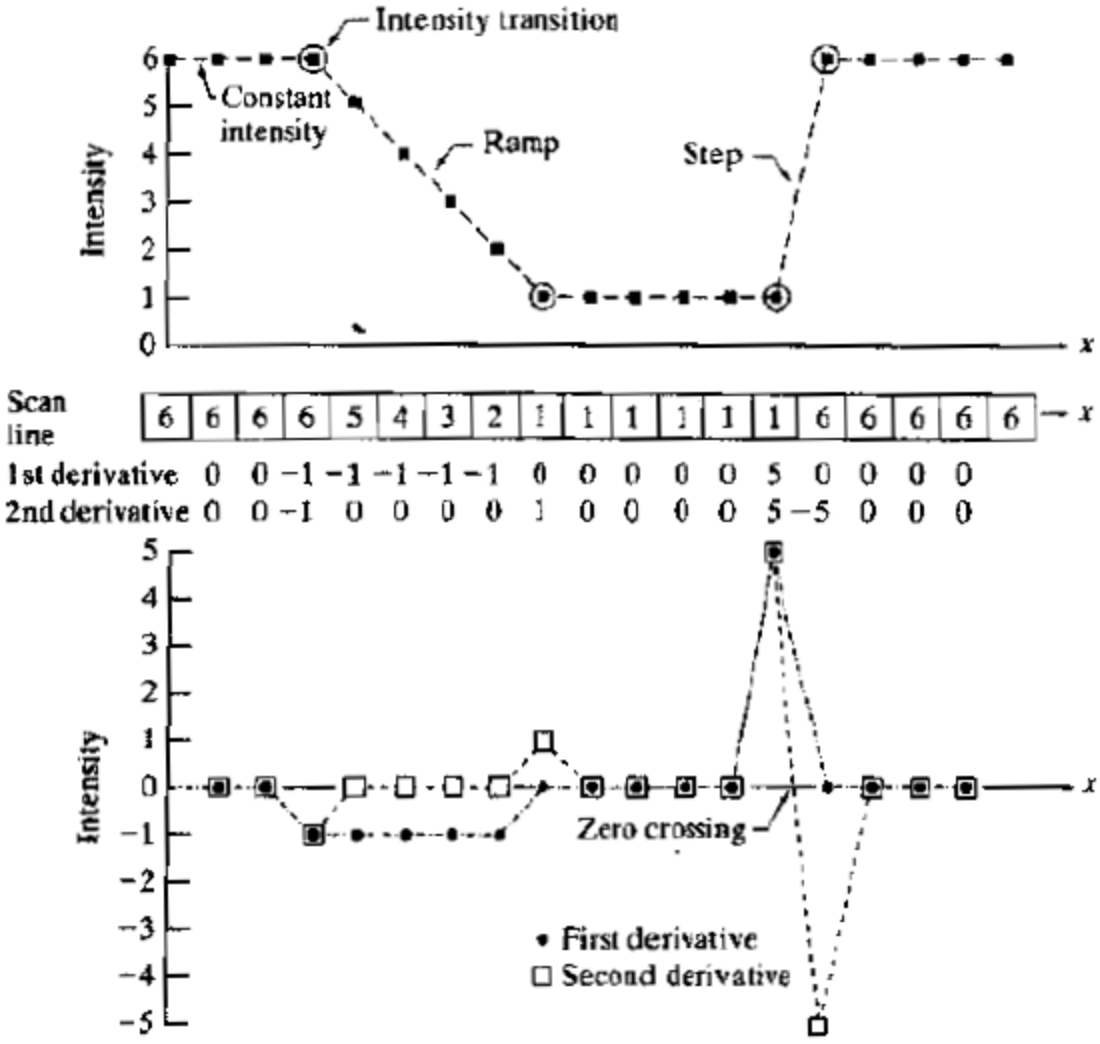
Second order derivative filtering

In the second order derivative, edge is present at the location where second order derivative is zero, which is also called zero crossing.

1-D second order derivative is calculated by following equation:

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

Following graph illustrate the first and second order derivative of 1-D function



For the 2-D function we have to find the second derivative for x and y coordinates as:

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

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

Laplacian Filter:

This filter highlights gray level discontinuities in an image. By adding second order derivative of x and y coordinates we can find the filter mask which is called Laplacian Filter.

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

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

In the laplacian filter, sum of coefficient of filter mask is zero.

high boost filtering ko mask
jstae xa tesam chai a+4 esto
hunthyo esma chai 4 matra xa ani tesma bich ko
-ve hune vni thenw

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

0	1	0
1	-4	1
0	1	0

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

1	1	1
1	-8	1
1	1	1

Using any of these filter mask we can perform convolution operation to filter the input image.

Example: Apply laplacian filter on the given image on the center pixel.

8	5	4
0	6	2
1	3	7

Solution: let's take filter mask

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

Now apply convolution operation

g(x,y) = (8x0) + (5x-1) + (4x0) + (0x-1) + (6x4) + (2x-1) + (1x0) + (3x-1) + (7x0)

= 14

Now center pixel of input image i.e. 6 is replace by 14. Output image is

8	5	4
0	14	2
1	3	7

Enhance Laplacian Filter

The enhance laplacian filter is calculate by using enhance laplacian filter mask. Which is obtain by increasing center value of laplacian mask by 1 and apply convolution operation.

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

enhance

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

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

enhance

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

Magnification by replication and interpolation

Magnification is the process of zooming or enlarging the image pixel. This technique is use to enhance the image quality.

There are two types of magnification

- Replication
- Interpolation

Replication:

In this technique image pixel is replicated or copy the pixel in corresponding row and column.

Method 1: pixel replication

Suppose we have 3x3 image

1	2	3
4	5	6
7	8	9

Replicate each pixel in corresponding row and column

1	1	2	2	3	3
1	1	2	2	3	3
4	4	5	5	6	6
4	4	5	5	6	6
7	7	8	8	9	9
7	7	8	8	9	9

Hence the 3x3 image is enlarged to 6x6 image.

Method 2: zero interlacing

Let's take same 3x3 image and do zero interlacing

1	0	2	0	3	0
0	0	0	0	0	0
4	0	5	0	6	0
0	0	0	0	0	0
7	0	8	0	9	0
0	0	0	0	0	0

Interpolation: calculation involve hunxa estimate the value of pixel using neighbouring pixel jsto ho

This is also a technique of enlarging the image pixel.

Method 1: nearest neighbor interpolation

In this method, pixel value of (x,y) is obtained by replicating nearest pixel value.

Suppose we have an image

1			2
	p		
		q	
3			4

To find the value of pixel p and q using nearest pixel replicate the nearest pixel value

1			2
	1		
		4	
3			4

Method 2: Linear Interpolation

In this technique first create space between two consecutive pixel and place the average value of those two pixels.

Suppose we have an image

1	2	3
4	5	6
7	8	9

Now enlarge 3x3 image into 6x6 image

1	1.5	2	2.5	3	1.5
2.5	3	3.5	4	4.5	2.25
4	4.5	5	5.5	6	3
5.5	6	6.5	7	7.5	3.75
7	7.5	8	8.5	9	4.5
3.5	3.75	4	4.25	4.5	2.25

(1+2)/2= 1.5, (2+3)/2=2.5, (3+0)/2=1.5, (1+4)/2 =2.5

End of Unit-2