

Digital Image Processing



Part 3

Introduction to Image Processing

Content

1. Fundamental of spatial filtering

- Spatial Domain
- Frequency Domain

2. Linear Filters

3. Smoothing Filters

- Average Filter
- Gaussian Filtering

4. Nonlinear Filter

- Median filter

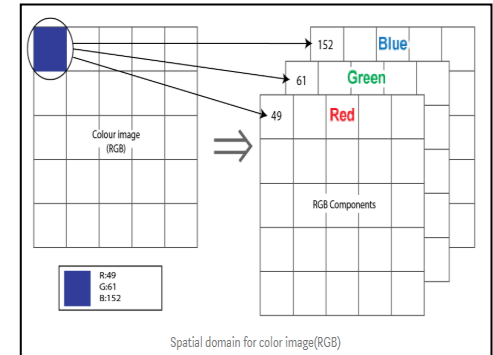
5. Sharpening Filters

- Laplacian Filter

6. Edge Detection

- Sobel Operators

7. Combining image enhancement methods



$$\mathbf{h} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Fundamental of spatial filtering

Filtering is a fundamental operation in image processing. It can be used for:

- 1- *image enhancement*
- 2- *Noise reduction*
- 3- *Edge detection,*
- 4- *Sharpening.*

The **concept of filtering** has been applied in the:

Frequency domain, where it rejects some frequency components while accepting others.

Spatial domain, Spatial filtering modifies an image by replacing the value of each pixel with a function of the values of the pixel and its neighbors (filtering is pixel neighborhood operation).

- Commonly used spatial filtering techniques include **median, average, Gaussian filtering**, etc.
- The **filtering function** sometimes is called **filter mask**, or **filter kernel**.
- They can be broadly classified into two different categories :
 1. **Linear filtering.**
 2. **Nonlinear filtering (Order-statistic filters).**

Spatial Domain

An image can be represented in the form of a **2D matrix** where each element of the matrix represents **pixel intensity**.

This **state of 2D matrices** that depict the **intensity distribution of an image** is called **Spatial Domain**. It can be represented as shown below.

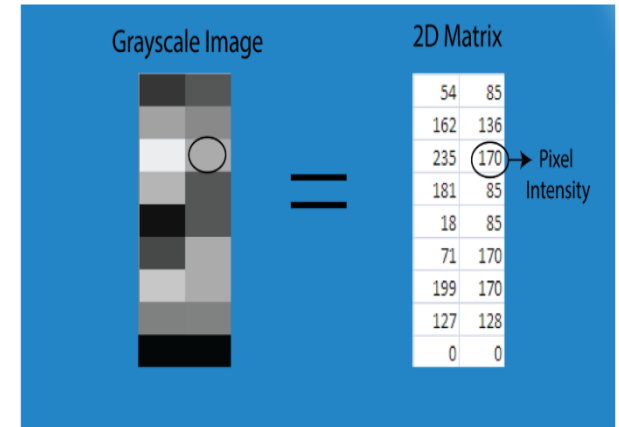
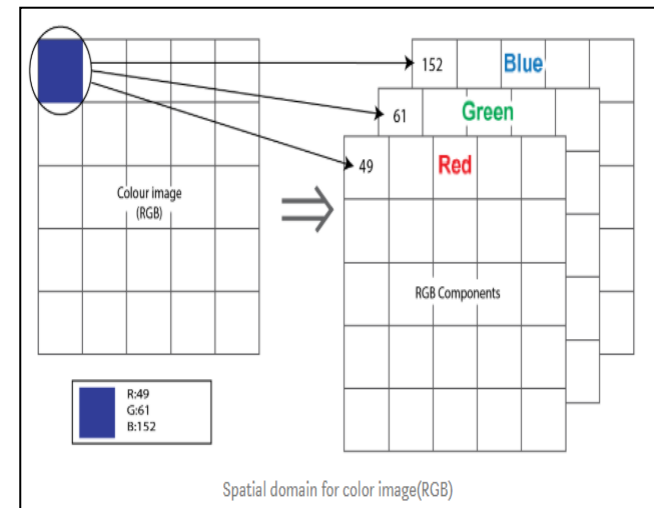


Illustration of Spatial Domain

- For the **RGB image**, the spatial domain is represented as a **3D vector of 2D matrices**. Each **2D matrix** contains the intensities for a **single color** as shown below.
- Each **pixel intensity** is represented as $I(x, y)$ where **x, y** is the **co-ordinate of the pixel** in the **2D matrix**. Different operations are carried out in this value.



Spatial domain for color image(RGB)

Frequency Domain

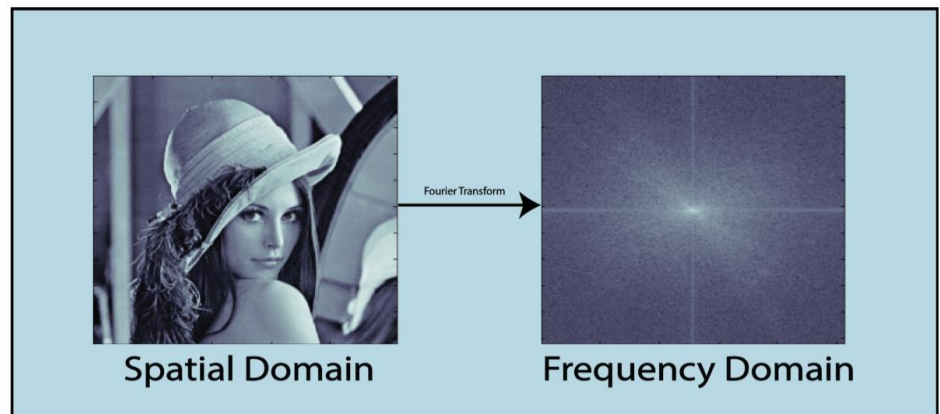
In frequency-domain methods are based on the **Fourier Transform of an image**.

Roughly, the term frequency in an image **tells about the rate of change of pixel values**.

The tie between spatial- and frequency-domain processing is the Fourier transform.

- We use the **Fourier transform** to go from the spatial
- To the frequency domain; to return to the spatial domain we use the **inverse Fourier transform**.

The figure below depicts the conversion of the image from the **spatial domain** to the **frequency domain** using **Fourier Transformation**.



Q: Why we need a domain other than spatial domain ?

Answer

Many times, image processing tasks are best performed in a domain other than the spatial domain. Moreover, **it is easy to detect some features in a particular domain, i.e., a new information can be obtained in other domains.**

Image Transformation mainly follows three steps-



Step-1. Transform the image.

Step-2. Carry the task(s) in the transformed domain.

Step-3. Apply inverse transform to return to the spatial domain.

Linear Filters

A linear spatial filter performs a sum-of-products operation between an image **f** and a filter kernel, **w**.

- The kernel is an array whose **size** defines the **neighborhood of operation**, and whose **coefficients** determine the **nature of the filter**.
- Other terms used to refer to a spatial filter kernel are mask, template, and window.
- We use the term filter kernel or simply kernel.

The equation below illustrates the mechanics of linear spatial filtering using a kernel. At any point (x, y) in the image, the response, g(x, y), of the filter, is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel:

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

As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g, in the process

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

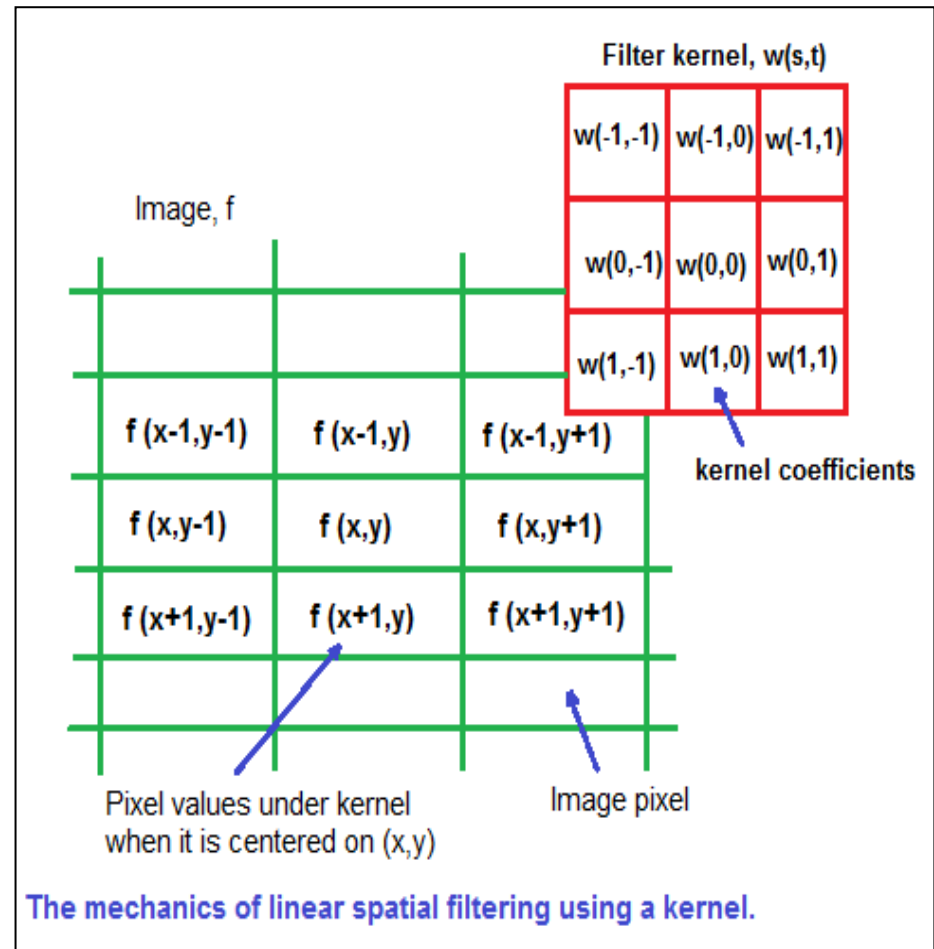
➤ For a kernel of size $m \times n$, we assume that $m=2a+1$ and $n=2b+1$, where a and b are nonnegative integers.

This means that our focus is on kernels of odd size in both coordinate directions.

➤ In general, linear spatial filtering of an image of size $M \times N$ with a kernel of size $m \times n$ is given by the expression

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

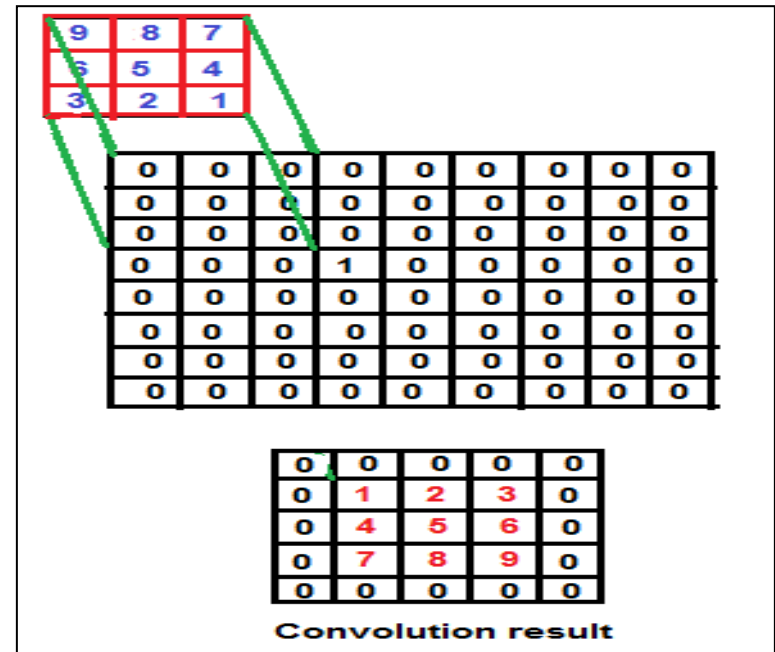
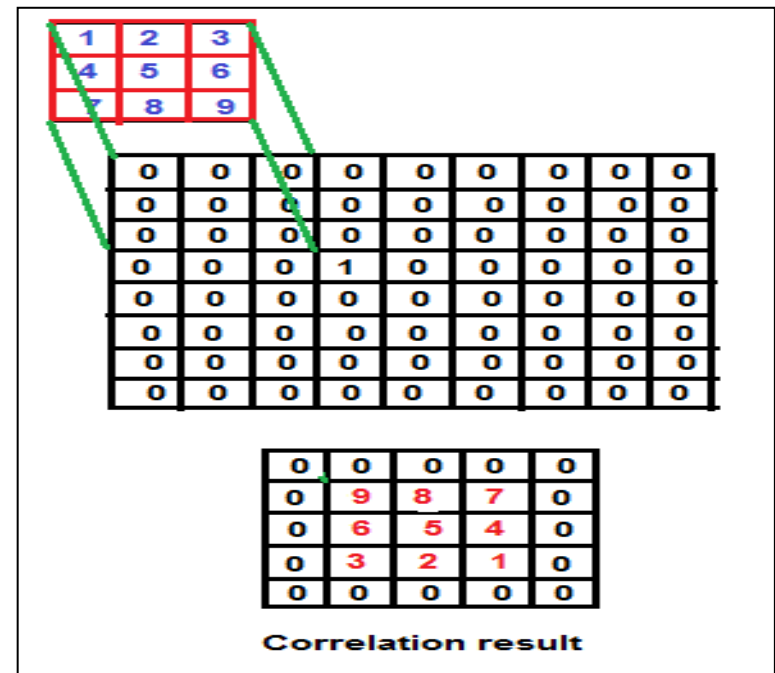
where x and y are varied so that the center (origin) of the kernel visits every pixel in f once.



- **Correlation** consists of moving the center of a kernel over an image, and computing the sum of products at each location.
- We used correlation to check similarity between two image

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

- The mechanics of spatial **convolution** are the same, **except that the correlation kernel is rotated by 180°**. Thus, when the values of a kernel are symmetric about its center, correlation and convolution yield the same result.



Smoothing Filters

Smoothing (also called averaging) spatial filters are **used to reduce sharp transitions in intensity**.

Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction.

Smoothing prior to image resampling to reduce aliasing, is also a common application.

- The **commonly used smoothing filters** are **Averaging** and **Gaussian filters**.
- It can be **performed using the convolution operation**.

$$s(x, y) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} h(m, n) f(x - m, y - n)$$

Where: $h(m, n)$ is a filtering mask of size $M \times N$.

- **Each element in this filter mask** usually represents the **weights** used in the linear combination.

- The types of different linear filters correspond to different filter masks.
- The sum of all the elements in $h(m, n)$ will affect the overall intensities of the output image.

Therefore, it is sensible to normalize $h(m, n)$ such that the sum is equal to one.

- In the case of negative values in $h(m, n)$, the sum is typically set to zero.
- For smoothing (low pass filters), the values in the mask must be positive; otherwise, this mask will contain some edges (sharpening filters).

Average Filter

The average filtering is also called **mean filtering**. Where the output pixel value is the mean of its neighborhood. Thus, the filtering mask is as follows (3*3 as an example).

$$\mathbf{h} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \quad h = 1/9 \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

In practice, **the size of the mask controls the degree of smoothing and loss of the details**

Example :

Used average filter to smooth sub image as shown

6	8	9	10	12
7	15	6	14	13
3	7	5	9	3
14	16	10	7	8

Answer :

$$(6*1/9) + (8*1/9) + (9*1/9) + (7*1/9) + (15*1/9) + (6*1/9) + (3*1/9) + (7*1/9) + (5*1/9) = 7.33$$

OR

$$(6+8+9+7+15+6+3+7+5)/9 = 7.33$$

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

6	8	9	10	12
7	15	6	14	13
3	7	5	9	3
14	16	10	7	8

6	8	9	10	12
7	7.33			13
3				3
14	16	10	7	8

$$(8*1/9) + (9*1/9) + (10*1/9) + (15*1/9) + (6*1/9) + (14*1/9) + (7*1/9) + (5*1/9) + (9*1/9) = 9.22$$

OR

$$(8+9+10+15+6+14+7+5+9)/9 = 9.22$$

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

6	8	9	10	12
7	15	6	14	13
3	7	5	9	3
14	16	10	7	8

6	8	9	10	12
7	7.33	9.22		13
3				3
14	16	10	7	8

Gaussian Filtering

Gaussian filtering is another important filter. The weights in the filter mask are of a Gaussian function.

$$h(m,n) = \frac{1}{z} \exp(-0.5(m^2 + n^2) \backslash \sigma)$$

Where: σ is the variance and controls the degree of smoothing.

The larger value σ is, the larger degree smoothing can be achieved.



Figure (a) The Lenna image; (b) (c) (d) filtered images using mean filtering with mask size 3, 7, 11; (e) (f) (g) filtered images using Gaussian filtering with different variances at 1, 5, 9.

Notes:

1. The coefficient of average or mean mask sum to one, so the image brightness will be retained.
2. The coefficients are all positive, so it will tend to blur the image.
3. This type of mean filter smooth out local variations within an image, so it essentially a low pass filter. So a low filter can be used to attenuate image noise that is composed primarily of high-frequency components.

Nonlinear Filter

The order-statistical filters are usually **non-linear filters**, which are hardly represented by convolution.

- The value of a given pixel in the output image is represented by some statistic within its support neighborhood in the original image, such as the median filter.

There are some other filters as well such as the **max/min filter**.

- The **median filter** simply replaces the value of the pixel with the median value within its neighborhood.
- The **max/min filter** replaces the value of the pixel with the maximum/minimum value within its neighborhood.

Those filters are normally **non-linear** and **cannot be easily implemented in the frequency domain**.

- However, the common elements of a filter are:
 - (1) A neighborhood
 - (2) An operation on the neighborhood include the pixel itself.
- Typically the neighborhood is a rectangular of different size, for example 3x3, 5x5 ... and smaller than the image itself.

Figure below shows the filtered image using a median filter of different mask sizes.



Figure: Filtered Lenna image using median filtering of different masks 3, 7, and

Median filter

The **median filter** is a **non-linear filter** (order filter). These filters are based on a specific type of image statistics called order statistics.

- Typically, these filters operate on small sub image, “Window”, and replace the center pixel value (similar to the convolution process).
- **Order statistics** is a **technique that arranges the entire pixel in sequential order**, given an $N \times N$ window (W) the pixel values can be ordered from smallest to the largest.

$$I_1 \leq I_2 \leq I_3 \dots \dots \dots < I_N$$

Where: $I_1, I_2, I_3 \dots \dots \dots, I_N$
are the intensity values of
the subset of pixels in the
image.



Example:

Given the following 3X3 neighborhood :

5	5	6
3	4	5
3	4	7

1. Sort the value in order of size (3,3,4,4,5,5,5,6,7) .
2. Select the middle value , un this case it is 5.
3. This 5 is then placed in center location.

- A **median filter** can use a neighborhood of any size, but 3X3, 5X5 and 7X7 are typical.
- Note that the **output image must be written to a separate image** (a buffer); so that the results are not corrupted as this process is performed. (The median filtering operation is performed on an image by applying the sliding window concepts, similar to what is done with convolution).
- The window is overlaid on the upper left corner of the image, and the median is determined. This value is put into the output image (buffer) corresponding to the center location of the window. The window is then slide one pixel over, and the process is repeated.

Example 1:

Using median filters to remove salt and pepper noise, from sub image $I(r, c)$.

First pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

2, 5, 16, 19, **20**, 21, 25, 43, 51

2. By **using median filter** we select the **median value 20** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	2	25	23	34	19
43	51	19	18	40	42
15	18	25	23	38	40
52	44	34	12	10	13

Second pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

2, 15, 16, 18, **19**, 20, 23, 25, 51

2. By **using median filter** we select the **median value 19** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	20	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	20	19	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Example 2:

Using min. filters to remove salt and pepper noise, from sub image I(r , c).

First pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

5, 13, 16, 19, 20, 21, 25, 25, 43, 51

2. By **using min. filter** we select the **minimum value 5** and put it in buffer image in the **position (1,1)**

Second pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

13,15,16,19,20,22,23,25,51

2. By **using min. filter** we select the **minimum value 13** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	13	25	23	34	19
43	51	19	22	40	42
15	18	25	23	38	40
52	44	34	12	10	13

5	16	20	15	17	20
21	5	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	5	13	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Example 3:

Using max. filters to remove salt and pepper noise, from sub image $I(r, c)$.

First pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

5,13,16,19,20, 21,25,25,43,**51**

2. By **using max. filter** we select the **maximum value 51** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	13	25	23	34	19
43	51	19	22	40	42
15	18	25	23	38	40
52	44	34	12	10	13

Second pixel:

1. **Rearrange** the 3x3 neighbor pixels in ascending way such as:

13,15,16,19,20,22,23,25,**51**

2. By **using max. filter** we select the **maximum value 51** and put it in buffer image in the **position (1,1)**

5	16	20	15	17	20
21	51	?	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

5	16	20	15	17	20
21	51	51	?	?	19
43	?	?	?	?	42
15	?	?	?	?	40
52	44	34	12	10	13

Sharpening Filters

The **objective of sharpening** is to **draw attention** to the **fine details of an image**. This is also **related to the situation** where an image that has been blurred and now needs to be de-blurred.

In contrast to the process of image :

Smoothing that normally uses **pixel averaging techniques**,
Sharpening can be conducted using **spatial differentiation**.

The **image differentiation** actually:

- Enhances edges
- Discontinuities
- Depresses the areas of slowly changing gray-level values.

The Sharpening filter indicates the filter should have **positive coefficients near its center** and **negative coefficients in the outer periphery**.

Laplacian Filter

These filters will **tend to bring out**, or **enhance details** in the image. Example of convolution masks for the Laplacian-type filters are:

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

0	1	0
1	-4	1
0	1	0

Rotating by 90

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

1	1	1
1	-8	1
1	1	1

Rotating by 45

The **sum of the coefficients** in this kernel is **zero**, this mean that:

- when the kernel is over an **area of constant** (background area) or **slowly varying gray level**, the **result of convolution** is **zero** or some **very small number**.
- when **gray level is varying rapidly within the neighborhood**, the **result of the convolution** can be **large number**.

This number can be positive or negative, because the kernel contains both positive and negative coefficients; we therefore need to choose an output image representation that supports negative number.

The **main advantage** of Laplacian filter is to **increase the degree of sharpening**. It **enhance details in all directions equally**.

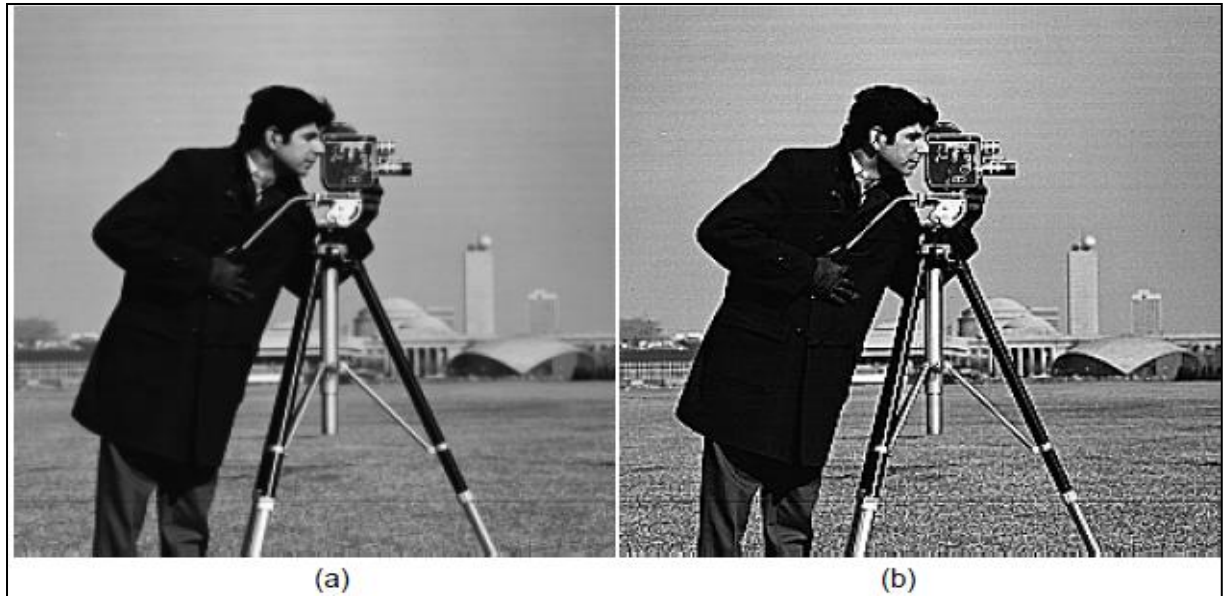


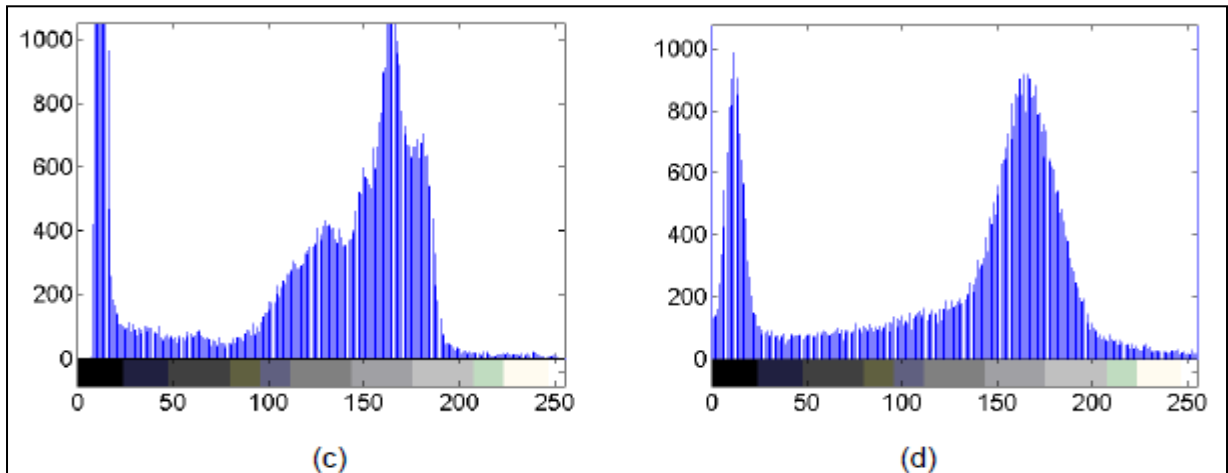
Image sharpening and histograms :

(a) original image

(c) histogram

(b) sharpened image

(d) histogram .



Edge Detection

Edge detection operators are **based on the idea** that **edge information in an image is found by looking at the relationship a pixel has with its neighborhood.**

- **If a pixel's gray level value is similar to those around it, there is probably not an edge at that point.**
- **If a pixel has a neighbor with widely varying gray levels, it may represent an edge point.**

In other words, an **edge is defined by a discontinuity in gray-level.** Ideally, an edge separates two distinct objects.

Definitions:

- **Edge** is a location in an image where there is a sudden variation in the gray level or color of pixels.
- **Edge** is a collection of pixel values whose brightness value changes abruptly.
- **Edges** represent borders between regions on an object or in a scene.

Sobel Operators

The **Sobel edge detection masks** look for edges in both the horizontal and vertical directions and then combine this information into a single metric.

The masks are as follows:

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

Row Mask

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

Column Mask

These masks are each convolved with the image. At each pixel location we now have two numbers:

- **S1**, corresponding to the result from the row mask.
- **S2**, corresponding to the result from the column mask.

We use these numbers to compute two matrices, the edge magnitude and the edge direction, which are defined as follows:


$$\text{Edge Magnitude} = \sqrt{s_1^2 + s_2^2}$$

$$\text{Edge Direction} = \tan^{-1} \frac{s_2}{s_1}$$

Example:

Compute the convolution of the Sobel operators s1 and s2 with the image and show the edge magnitude image:

3	5	7
2	4	3
1	8	6



3	5	7
2	11	3
1	8	6

Answer :

3	5	7
2	4	3
1	8	6

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

$$= (3*-1) + (5*-2) + (7*-1) + (2*0) + (4*0) + (3*0) + (1*1) + (8*2) + (6*1) = -20 + 23 = 3$$

Row Mask

3	5	7
2	4	3
1	8	6

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

$$= (3*-1) + (5*0) + (7*1) + (2*-2) + (4*0) + (3*2) + (1*-1) + (8*0) + (6*1) = 11$$

Column Mask

$$\text{Edge Magnitude} = \sqrt{s_1^2 + s_2^2} = \sqrt{3^2 + 11^2} = \sqrt{9 + 121} = \sqrt{130} = 11.4$$

$$\text{Edge Direction} = \tan^{-1} \frac{s_2}{s_1} = \tan^{-1} \frac{11}{3} = \tan^{-1}(3.6)$$

Combining image enhancement methods

In real applications, **it is hard to know what kind of noise has been added to an image.**

- Therefore, **it is difficult to find a unique filter that can appropriately enhance this noisy image.**

However, **it is possible if several de-blurring methods can be combined in a framework in order to pursue a maximum demising outcome.** We explain one of the combinatorial techniques using an X-ray example.

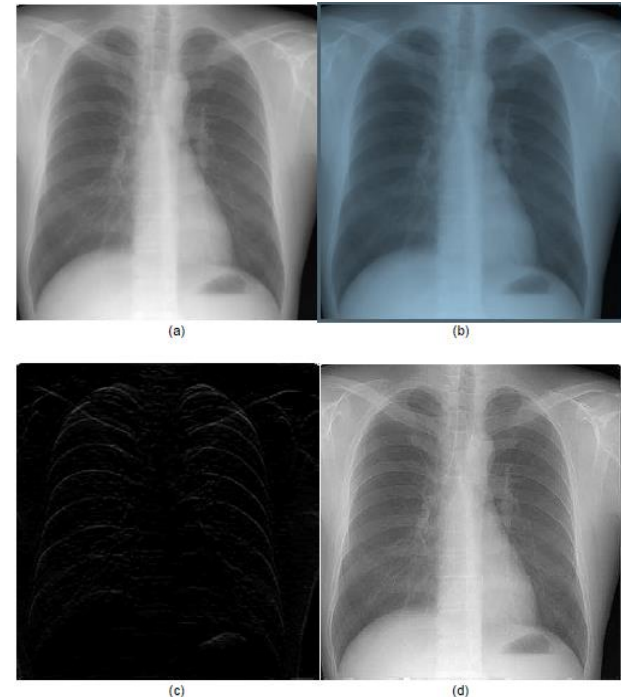
Figure (a) illustrates a male chest's X-ray image. The purpose of the process is to highlight the middle cross section of the image using the combination of sharpening and edge detection.

Figure (b) shows the result of applying the median filter.

Figure (c) is the outcome of using Sobel edge detection.

Figure (d) demonstrates the combination of the Laplacian and Sobel process.

Figure (d) shows the edge details in the graph, which highlights the structure of the central cross section of the image.



The End