

LECTURE NOTES ON
DIGITAL IMAGE PROCESSING

All JNTU World



All JNTU World
Get The Most Out Of Imagineering

► What is Digital Image Processing?

Digital Image

Image is a representation, likeness, or imitation of an object or thing

Common Digital image formats include: Binary images or single bit images, Gray scale images, Color images

A digital image is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels. Pixel values typically represent gray levels, colours, heights, opacities etc. Pixels are the elements of a digital image

— a two-dimensional function $f(x,y)$, x and y are spatial coordinates.
The amplitude of f is called intensity or gray level at the point (x, y)

What is meant by Digital Image Processing? Explain how digital images can be represented?

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude OR intensity OR gray level of f at any pair of coordinates (x, y) .

When x , y , and the amplitude values of f are all **finite, discrete** quantities, we call the image a digital image. each of these has a particular location and value.

These elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used to denote the elements of a digital image.

The field of digital image processing refers to processing digital images by means of a digital computer.

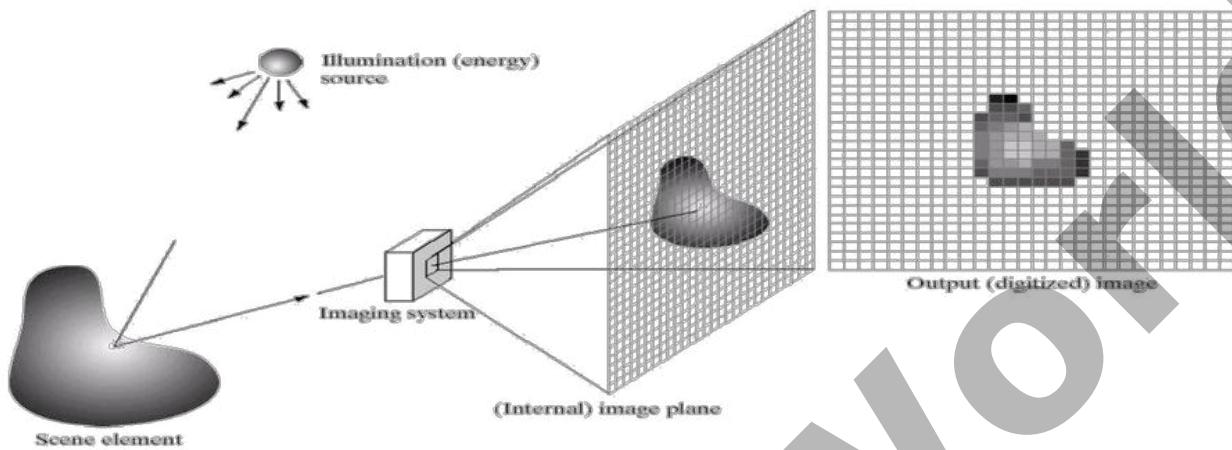
- A color image is just three functions pasted together. We can write this as a “vector-valued” function:

$$r(x, y) \\ f(x, y) \quad g(x, y)$$

$b(x, y)$

All JNTU World

Reflectance in $[0, 1]$, illumination in $[0, \infty]$



Representing Digital Images:

Continuous image, say a film positive is a function of 2 continuous variables $f(s,t)$
This can be converted into a digital image by scanning;

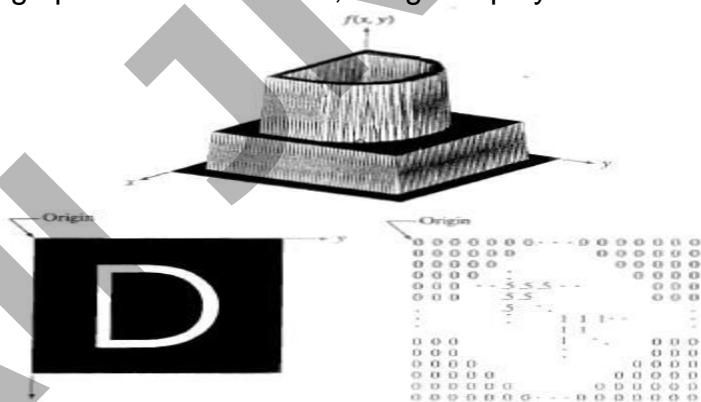
sampling and quantization

Sample into a 2D array $f(x,y)$, M rows and N columns, (x,y) = discrete coordinates, $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$

Section of the real plane spanned by the coordinates of an image = *spatial domain* x and y are called *spatial variables* or *spatial coordinates*

$f(x,y)$ can be represented in three ways:

Image plotted as a surface, Image displayed as a visual intensity array, Numerical array

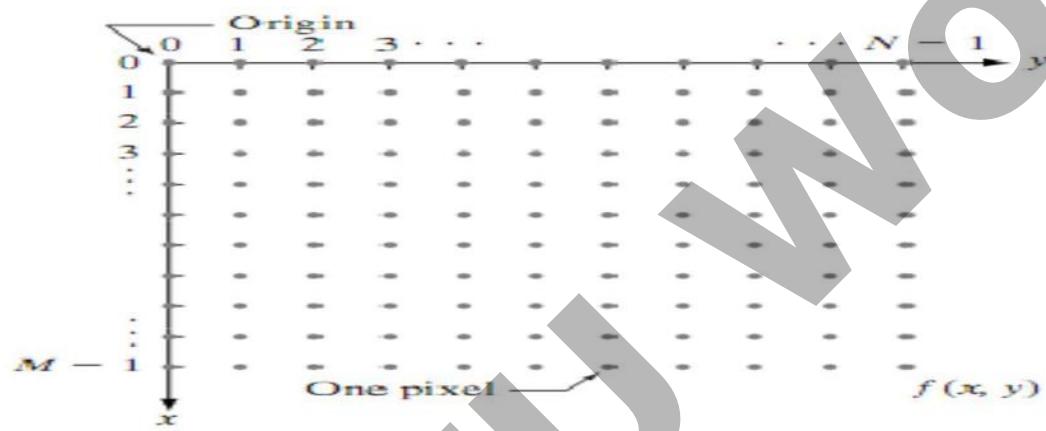


Dynamic range = ratio of maximum measurable intensity to minimum detectable intensity level in the system
 Rule: upper limit determined by *saturation*, lower limit determined by *noise*

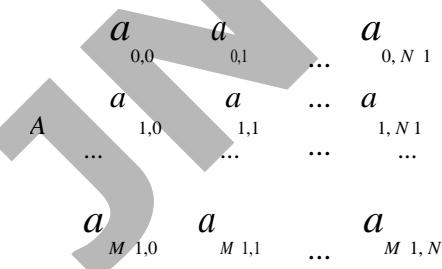
Contrast = difference in intensity between the highest and the lowest intensity levels in an image

High dynamic range => high contrast expected
 Low dynamic range => dull, washed-out gray Look

Coordinate convention used to represent digital images



► The representation of an $M \times N$ numerical array as



► The representation of an $M \times N$ numerical array

$$\begin{array}{ccccccccc}
 f(1,1) & f(1,2) & \dots & f(1, N) & & f(0,0) & f(0,1) & \dots & f(0, N-1) \\
 f(x, y) & f(2,1) & & f(2, N) & & f(x, y) & f(1,0) & f(1,1) & \dots & f(1, N-1) \\
 \dots & \dots & \dots & \dots & & \dots & \dots & \dots & \dots & \dots \\
 f(M,1) & f(M,2) & \dots & f(M, N) & & f(M,0) & f(M,1) & \dots & f(M, N-1)
 \end{array}$$

- Discrete intensity interval $[0, L-1]$, $L=2^k$
- The number b of bits required to store a $M \times N$ digitized image $\mathbf{b} = M \times N \times k$
- Image with 2^k intensity levels => “*k-bit image*” (ex: 256 8-bit image)

A Simple Image Formation Model

Images are denoted by two-dimensional functions $f(x,y)$. Value of amplitude of f at (x,y) is a positive scalar quantity.

Images are generated by physical process: intensity values are proportional to the energy radiated by a physical source => $0 < f(x,y) < \infty$

$f(x,y)$ may be characterized by 2 components:

- (1) The amount of source illumination *incident* on the scene: *illumination* $i(x,y)$
- (2) The amount of illumination *reflected* by the objects of the scene: *reflectance* $r(x,y)$

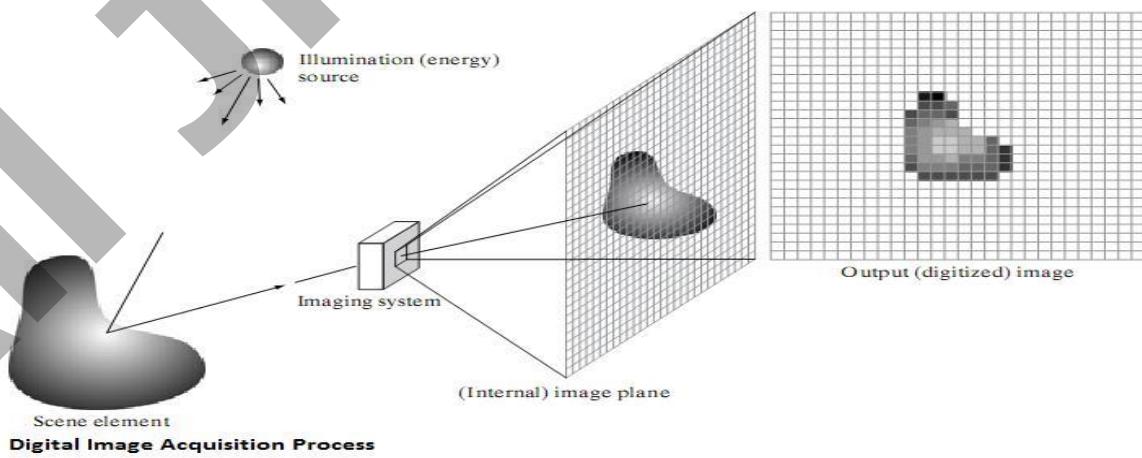
$$f(x,y) = i(x,y) r(x,y), \quad \text{where } 0 < i(x,y) < \infty \text{ and } 0 < r(x,y) < 1$$

Example of typical ranges of illumination $i(x,y)$ for visible light (average values):

- Sun on a clear day: ~ 90,000 lm/m², down to 10,000 lm/m² on a cloudy day
- Full moon on a clear evening: ~0.1 lm/m²
- Typical illumination level in a commercial office: ~1000 lm/m²

Typical values of reflectance $r(x,y)$:

- 0.01 for black velvet, • 0.65 for stainless steel
- 0.8 for flat white wall paint, • 0.9 for silver-plated metal, • 0.93 for snow



Types or Levels of image processing

Image to Image Transformation

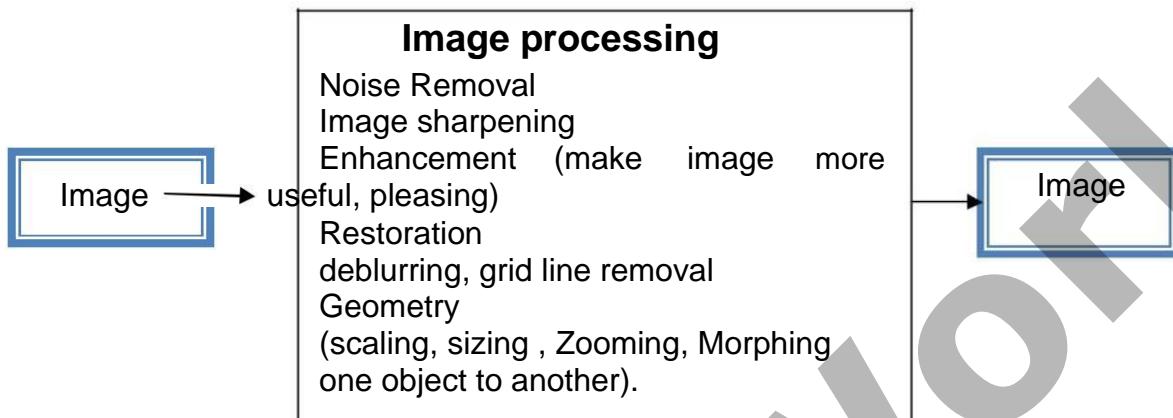
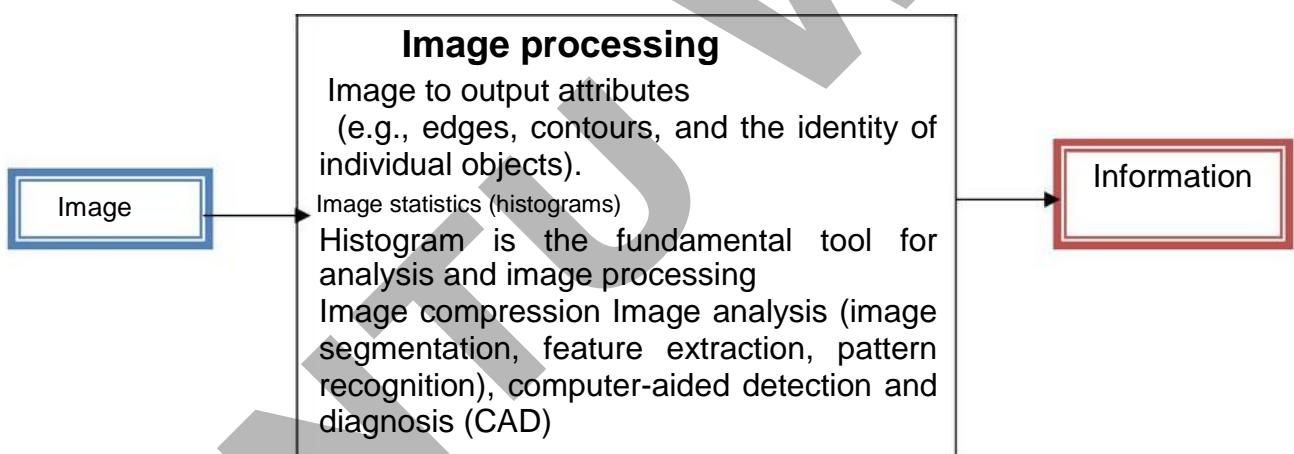
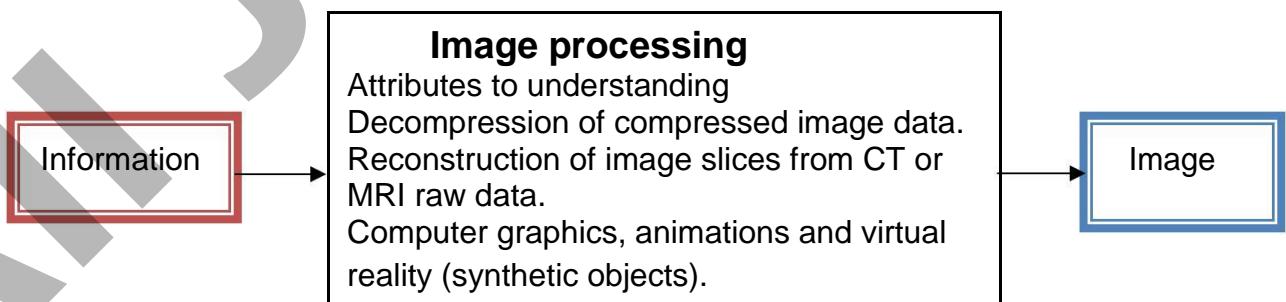


Image-to-information transformations

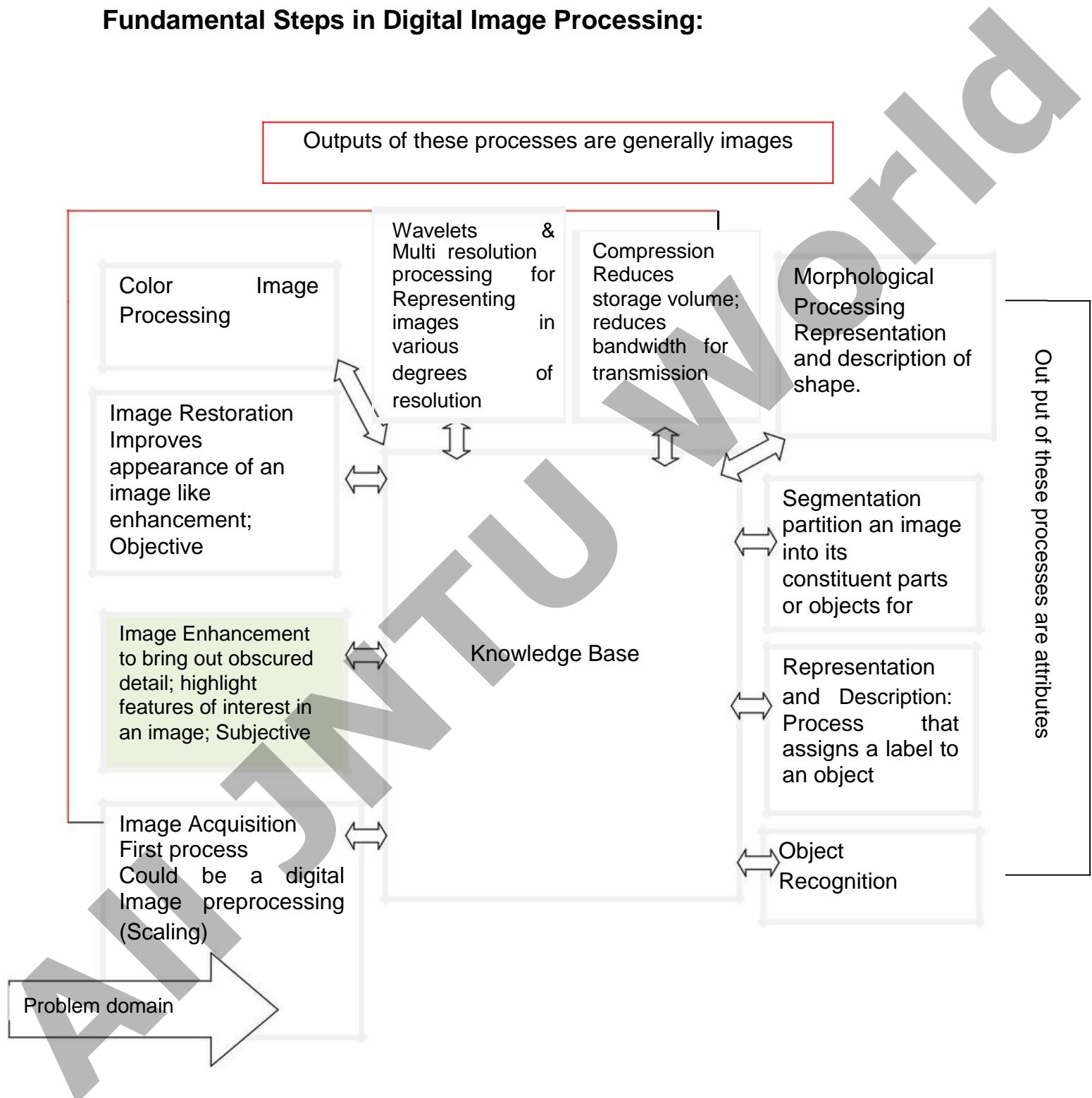


Information-to-image transformations



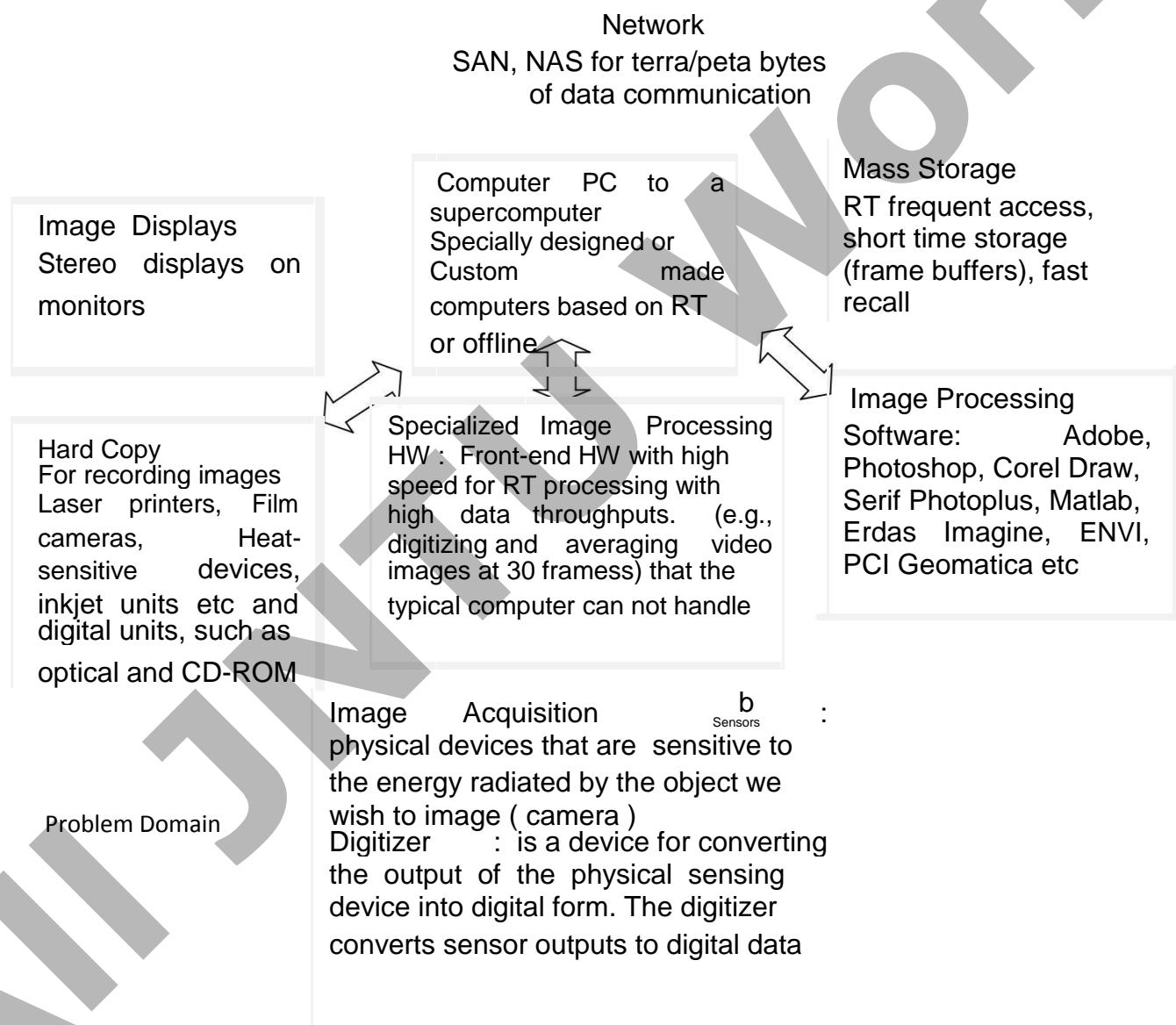
What are the fundamental steps in Digital Image Processing?

Fundamental Steps in Digital Image Processing:



What are the components / Elements of an Image Processing System?

Basic components comprising a typical general-purpose system used for digital image processing.



Explain the process of image acquisition. Image Sensing and Acquisition:

Transform of illumination energy into digital images:

The incoming energy is transformed into a voltage by the combination of input electrical power and sensor material.

Output voltage waveform = response of the sensor(s)

A digital quantity is obtained from each sensor by *digitizing* its response.

CCD cameras: widely used in modern applications: private consumers, industry, astronomy...

CCD: Charge Couple Device

Image acquisition using sensor arrays

Rectangular grid of electron-collection sites laid over a thin silicon wafer.

Image readout of the CCD one row at a time, each row transferred in parallel to a serial output register.

(1) Image Acquisition Using a Single Sensor:

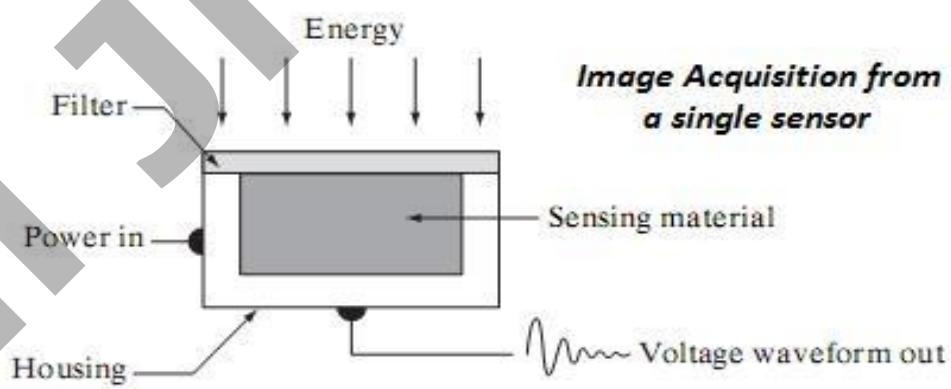
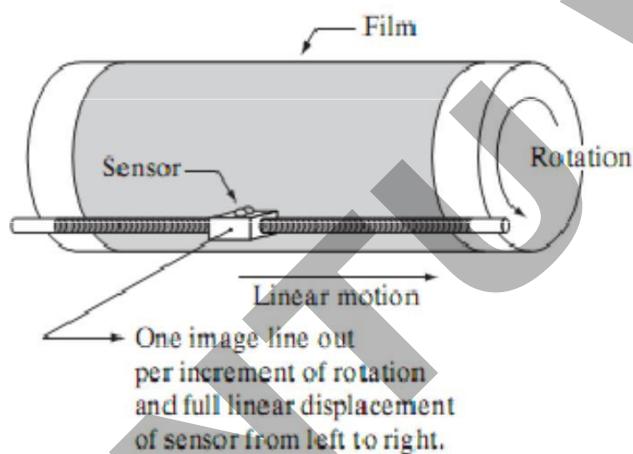


Figure shows the components of a single sensor. Perhaps the most familiar sensor of this type is the photodiode, which is constructed of silicon materials and whose output voltage waveform is proportional to light. The use of a filter in front of a sensor improves selectivity. For example, a green (pass) filter in front of a light sensor favors light in the green band of the color spectrum. As a consequence, the sensor output will be stronger for green light than for other components in the visible spectrum.

In order to generate a 2-D image using a single sensor, there has to be relative displacements in both the x- and y-directions between the sensor and the area to be imaged.

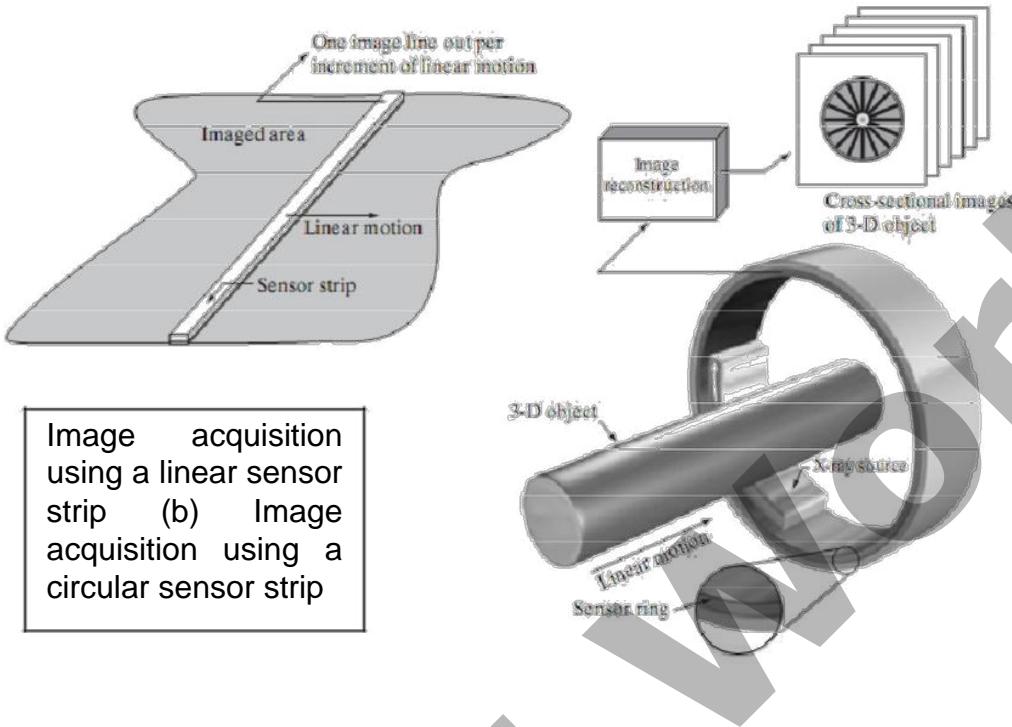
High-precision scanning scheme is shown in figure, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The single sensor is mounted on a lead screw that provides motion in the perpendicular direction.



(2) Image Acquisition Using Sensor Strips:

The strip provides imaging elements in one direction. Motion perpendicular to the strip provides imaging in the other direction. This is the type of arrangement used in most flat bed scanners.





Air borne and Space borne applications use Sensing devices with 4000 or more in-line sensors.

Imaging system is mounted on an aircraft or space craft that fly at a constant altitude and speed over the geographical area to be imaged.

One dimensional imaging sensor strips that respond to various bands of the electromagnetic spectrum are mounted perpendicular to the direction of flight.

The imaging strip gives one line of an image at a time, and the motion of the strip completes the other dimension of a two-dimensional image. Lenses or other focusing schemes are used to project the area to be scanned onto the sensors

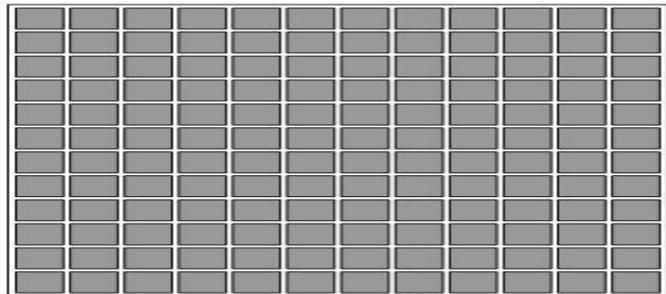
Sensor strips mounted in a ring configuration are used in medical and industrial imaging to obtain cross-sectional ("slice") images of 3-D objects (x-ray, computerized axial tomography (CAT), positron emission tomography (PET))

(3) Image Acquisition Using Sensor Arrays:

Figure shows individual sensors arranged in the form of a 2-D array. This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged arrays of 4000×4000 elements or more.

The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. Noise reduction is achieved by letting the sensor integrate the input light signal over minutes or even hours.

The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and converts them to a video signal, which is then digitized by another section of the imaging system.



Array sensor

Define spatial and gray level resolution. Explain about iso preference curves.

Spatial and Gray-Level Resolution:

Sampling is the principal factor determining the spatial resolution of an image. Basically, spatial resolution is the smallest discernible detail in an image. Suppose that we construct a chart with vertical lines of width W , with the space between the lines also having width W . A line pair consists of one such line and its adjacent space. Thus, the width of a line pair is $2W$, and there are $1/2 W$ line pairs per unit distance.

A widely used definition of resolution is simply the smallest number of discernible line pairs per unit distance; for example, 100 line pairs per millimeter.

Gray-level resolution similarly refers to the smallest discernible change in gray level. Due to hardware considerations, the number of gray levels is usually an integer power of 2. The most common number is 8 bits, with 16 bits and in some applications 10 or 12 bits levels are used.

- Spatial resolution
 - A measure of the smallest discernible detail in an image
 - stated with *line pairs per unit distance*, **dots (pixels) per unit distance**, *dots per inch (dpi)*
- Intensity resolution
 - The smallest discernible change in intensity level

Basic Relationships Between Pixels

- ▶ Neighborhood
- ▶ Adjacency
- ▶ Connectivity
- ▶ Paths
- ▶ Regions and boundaries

An Image is denoted by a function $f(x,y)$.

Each element $f(x,y)$ at location (x,y) is called a pixel.

There exist some basic but important relationships between pixels

Neighborhood

- ▶ pixel p at coordinates (x,y) will have Neighbors
 - 4-neighbors of p , denoted by $N_4(p)$
 - Each of these neighbors is at an equal unit distance from $p(x,y)$
- If $p(x,y)$ is a boundary pixel, it will have less number of neighbors.

	$(x-1, y)$	
$(x,y-1)$	$p(x,y)$	$(x,y+1)$
	$(x+1, y)$	

- 4 diagonal neighbors of p , denoted by $N_D(p)$:

$(x-1,y-1)$		$(x-1,y+1)$
	$p(x,y)$	
$(x+1,y-1)$		$(x+1, y+1)$,
- The points of $N_4(p) \cup N_D(p)$ put together are called 8 neighbors of p
- 8 neighbors of p , denoted $N_8(p)$
- $N_8(p) = N_4(p) \cup N_D(p)$

$(x-1,y-1)$	$(x-1, y)$	$(x-1,y+1)$
$(x,y-1)$	$p(x,y)$	$(x,y+1)$
$(x+1,y-1)$,	$(x+1,y)$	$(x+1, y+1)$,

Adjacency

2 pixels p and q are adjacent if they are connected

Let V be the set of intensity values

- 4-adjacency: Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
- 8-adjacency: Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

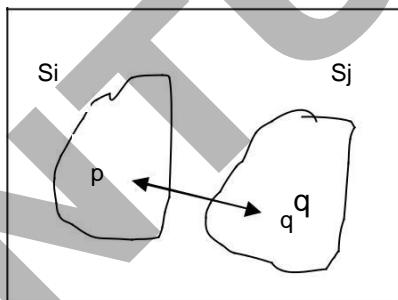
- Adjacency Let V be the set of intensity values
- m -adjacency: Two pixels p and q with values from V are m -adjacent if
 - (i) q is in the set $N_4(p)$, or
 - (ii) q is in the set $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V .

Two image subsets s_i and s_j are adjacent if $\exists p \in s_i$ and $\exists q \in s_j$ such that p and q are adjacent

Adjacency for 2 image regions like this that if there are 2 image subsets - S_i and S_j , we say that S_i and S_j will be adjacent if there exists a point p in image region S_i and a point q in image region S_j such that p and q are adjacent.

So, consider image region S_i and image region S_j ;

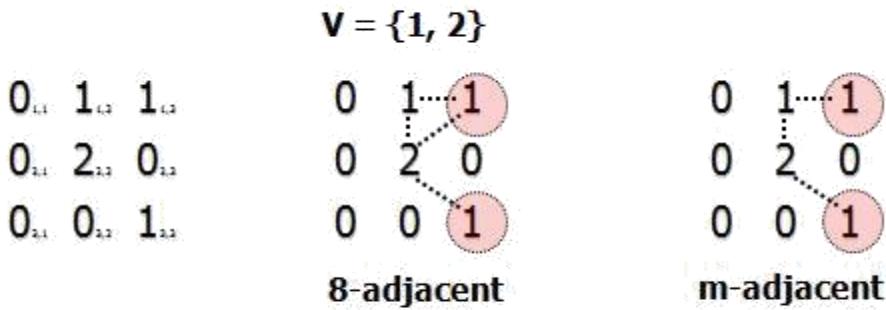
then if some point p in image S_i and some other point q in the image S_j so that this p and q , they are adjacent. So, if p and q are adjacent, then this image region S_i is adjacent to image region S_j . That means S_i and S_j ; they must appear one after the other, one adjacent to the other. So, this is the adjacency relation.



Boundary of a region is a global concept. Forms a closed path.

The idea of edge is a local concept.

Examples: Adjacency and Path



The 8-path from (1,3) to (3,3):

- (i) (1,3), (1,2), (2,2), (3,3)
- (ii) (1,3), (2,2), (3,3)

The m-path from (1,3) to (3,3):

- (1,3), (1,2), (2,2), (3,3)

Path

- A (digital) path (or curve) from pixel p with coordinates (x_0, y_0) to pixel q with coordinates (x_n, y_n) is a sequence of distinct pixels with coordinates $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$.
Where (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.
- Here n is the *length* of the path.
- If $(x_0, y_0) = (x_n, y_n)$, the path is *closed* path.
We can define 4-, 8-, and m-paths based on the type of adjacency used

Connectivity

Two pixels are said to be connected if they are adjacent in some sense.

They are neighbors 4, D or 8

Their gray levels or intensity values are same or similar

For a binary image B, pixels p and q are connected if

$$p \in N(q) \quad \text{or} \quad q \in N(p) \quad \text{and} \quad B(p) = B(q)$$

Let V be the set of gray values used to define connectivity for two points p and q belong to set V.

$$p, q \in V$$

Three types of connectivity are defined:

4 connectivity : $p, q \in V$ and $p \in N_4(q)$

8 connectivity : $p, q \in V$ and $p \in N_8(q)$

M connectivity or Mixed connectivity:

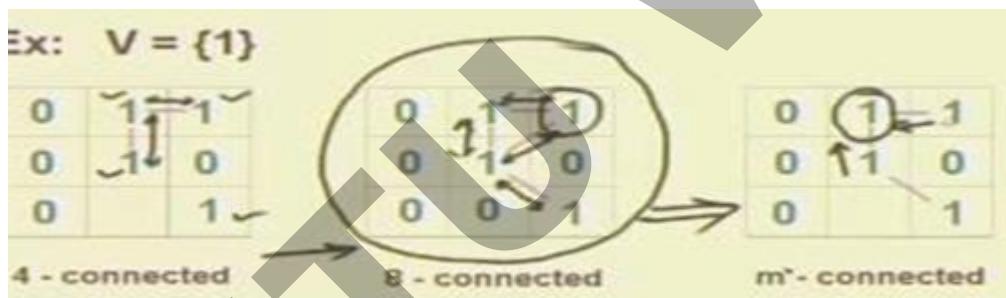
$p, q \in V$ are m connected if,

$q \in N_4(p)$ or

$q \in N_D(p)$ and $N_4(p) \cap N_4(q)$

where $N_4(p) \cap N_4(q)$ are set of pixels that are four neighbors of both p and q and whose values are from p.

Mixed connectivity is a modification of 8 connectivity; eliminates multiple path connections that arise through 8 connectivity. The M connectivity or mixed connectivity has been introduced to avoid this multiple connection path. So, you just recollect the restriction that we have put in case of mixed connectivity. In case of mixed connectivity we have said that 2 points are M connected if one is the 4 neighbor of other or one is 4 neighbor of other and at the same time, they do not have any common 4 neighbor



- Connected in S

Let S represent a subset of pixels in an image. Two pixels p with coordinates (x_0, y_0) and q with coordinates (x_n, y_n) are said to be connected in S if there exists a path

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

DISTANCE MEASURES

Consider three pixels $p(x, y)$ $q(s, t)$ $z(u, v)$

Properties of distance measure:

1. Valid distance measure : D is a distance function or metric if $D(p,q) \geq 0$;

$$D(p, q) = 0 \quad \text{if} \quad p=q$$

2. Distance is a symmetric measure. $D(p, q) = D(q, p)$

3. Inequality property: $D(p, z) \leq D(p,q) + D(q,z)$

What is the property that should be followed by this distance function D ?

So for this, let us take 3 points. We take 3 points here; p having a coordinate (x, y) , q having a coordinate (s, t) and I take another point z having the coordinate (u, v) . Then D is called a distance measure is a valid distance measure or valid distance metric. If $D(p, q)$ is greater than or equal to 0 for any p and q , any 2 points p and q ; $D(p, q)$ must be greater than or equal to 0 and $D(p, q)$ will be 0 only if p is equal to q .

So, that is quite obvious because the distance of the point from the point itself has to be equal to 0. **Then the distance metric distance function should be symmetric** that is if I measure the distance from p to q , that should be same as the distance if I measure from q to p . That is the second property that must hold true. That is $D(p, q)$ should be equal to $D(q, p)$.

And, there is a third property which is an inequality. That is if I take a third point z , then the distance between p and z that is $D(p, z)$ must be less than or equal to the distance between the p and q plus the distance between q and z and this is quite obvious, again from our school level mathematics you know that if I have say 3 points (p, q) and I have another point z and if I measure the distance between p and z , this must be less than the distance between pq plus the distance between pz .

EUCLIDEAN DISTANCE

$P(x,y)$ is one pixel and $q(s,t)$ is another pixel.

Distance between p and q pixels is

$$D(p, q) = [(x-s)^2 + (y-t)^2]^{1/2}$$

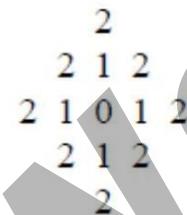
$$D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$$

The pixels having distance less than or equal to some value r from (x,y) are the points contained in a disk of radius r centered at (x,y)

In digital domain, There are various other distance measures. Those distance measures are say;

city block distance, chess board distance and so on

$$D_4 = |(x-s)| + |(y-t)|$$



Now, coming to the second distance measure which is also called D_4 distance or city block distance or this is also known as Manhattan distance; so this is defined as $D_4(p, q)$ is equal to x minus s absolute value plus y minus t absolute value.

The pixels having a D_4 distance $< r$ from (x,y) form a diamond centered at (x,y) Example: pixels where $D_4 \leq 2$

Note: Pixels with $D_4=1$ are the 4-neighbors of (x,y)

Chess Board Distance or D8 Distance :

Chess board distance is defined as

$$d_8(x, y) = \max \{ |x - s|, |y - t| \}$$

$\{ (x, y) \mid d_8(x, y) \leq 1 \}$ forms a square set of pixels

centered at p

In case of chess board distance, it is the maximum of the distances that you cover along x direction and y direction.

Now, we come to the third distance measure which is the chess board distance. As you have seen that in case of city block distance, the distance between 2 points was defined as the sum of the distances that you cover along x direction plus the distance along the y direction.

In case of chess board distance, it is the maximum of the distances that you cover along x direction and y direction.

So, this is $D_8(p, q)$ which is equal to max of $x - s$ and $y - t$ where we take the absolute value of both $x - s$ and $y - t$ and following the same argument, here you find that the set of points with a chess board distance of less than or equal to r , now forms a square centered at point p . So here, all the points with a chess board distance of equal to 1 from point p , they are nothing but the 8 neighbors of point p .

Similarly, the set of points with a chess board distance will be equal to 2 will be just the points outside the points having a chess board distance equal to 1. So, if you continue like this you will find that all the points having a chess board distance of less than or equal to r from a point p will form a square with point p at the center of the square. So, these are the distance different distance measures that can be used in the digital domain.

The D_8 distance (also called the *chessboard distance*) between p and q is given by:

$$D_8(p, q) = \max(|x - s|, |y - t|)$$

The pixels having a D_8 distance less than some r from (x, y) form a square centered at (x, y)

Example: pixels where $D_8 \leq 2$

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Note: Pixels with $D_8=1$ are the 8-neighbors of (x, y)

Applications of distance measures:

Shape Matching

Medical axis transformation

BASIC IDEA : ANALOGUE TO DIGITAL

BASIC IDEA:

The basic idea behind converting an analog signal to its digital signal is



FROM ANALOG TO DIGITAL

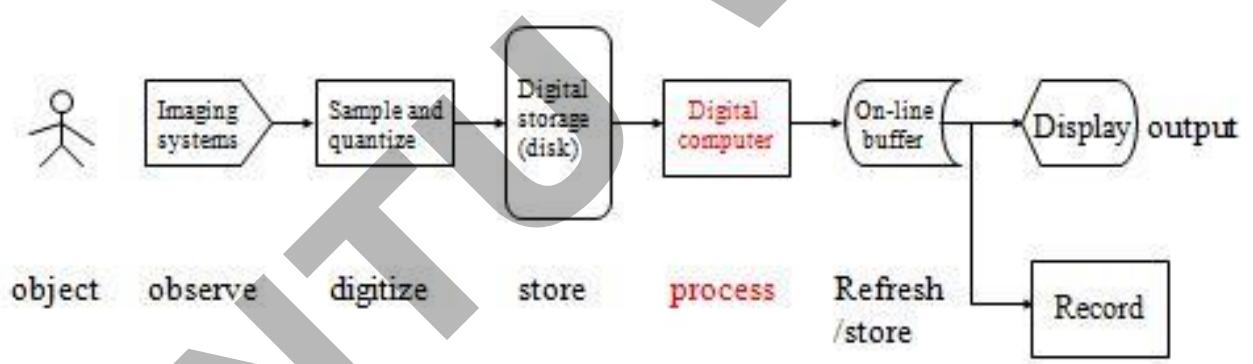


Image Processing ?

- Coding/compression
- Enhancement, restoration, reconstruction
- Analysis, detection, recognition,
understanding
- Raster data matrix representation
- Visualization

	Columns (j)					
Lines or rows (i)	1	2	3	4	5	1
2	10	15	17	20	21	2
3	15	16	18	21	23	90
4	17	18	20	22	22	120
5	18	20	22	24	25	103
6	100	93	97	101	105	150
7	103	90	70	120	133	135
8	200	100	0	123	222	215
9	200	100	0	123	222	215

Digital number of column 5, row 4 at band 2 is expressed as $BV_{5,4,2} = 105$.

Explain about image sampling and quantization process.

Image Sampling and Quantization:

The output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed.

To create a digital image, we need to convert the continuous sensed data into digital form.

This involves two processes: sampling and quantization.

Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization.

Basic Concepts in Sampling and Quantization:

Method of sampling determined by the sensor arrangement:

- **Single sensing element combined with motion:**
spatial sampling based on number of individual mechanical increments
- **Sensing strip:** the number of sensors in the strip establishes the sampling limitations in one image direction; in the other: same value taken in practice
- **Sensing array:** the number of sensors in the array establishes the limits of sampling

in both directions

SAMPLING AND QUANTIZATION

Sampling and quantization are the two important processes used to convert continuous analog image into digital image.

Image sampling refers to discretization of spatial coordinates whereas quantization refers to discretization of gray level values.

Normally, the sampling and quantization deals with integer values. After the image is sampled, with respect to x and y coordinates the number of samples used along the x and y directions are denoted as N and M , respectively. The N and M are usually the integer powers of 2. Hence N and M can be represented by the mathematical equation as follows:

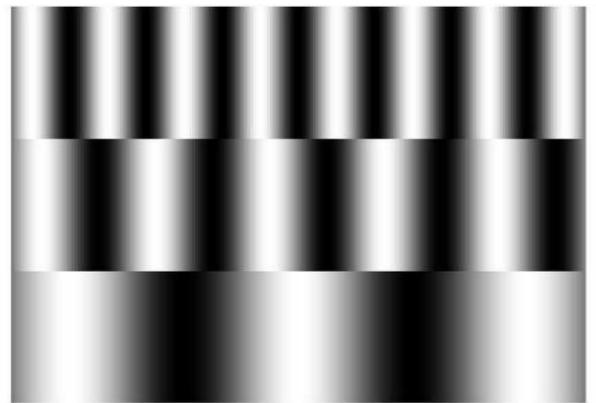
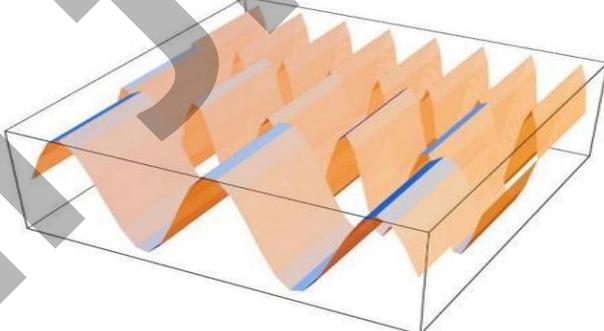
Discretization: A process in which signals or data samples are considered at regular intervals.

$$M=2^n \quad N=2^k$$

Similarly, when we discretize the gray levels, we use the integer values and the number of integer values that can be used is denoted as G . The number of integer gray level values used to represent an image usually are an integer powers of 2.

The finer the sampling (i.e., the larger M and N) and quantization (the larger K) the better the approximation of the continuous image function $f(x,y)$.

Spatial frequency



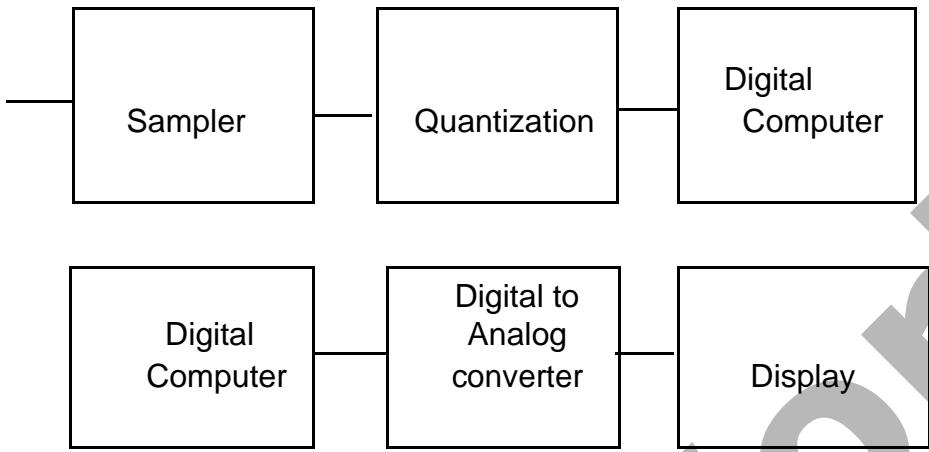


Image Sampling

Image sampling is required to represent the image in a digital computer, in a digital form. For this, instead of considering every possible point in the image space, we will take some discrete set of points and those discrete set of points are decided by grid.

So, if we have a uniform rectangular grid; then at each of the grid locations, we can take a particular point and we will consider the intensity at that particular point. So, this is the process which is known as **sampling**.

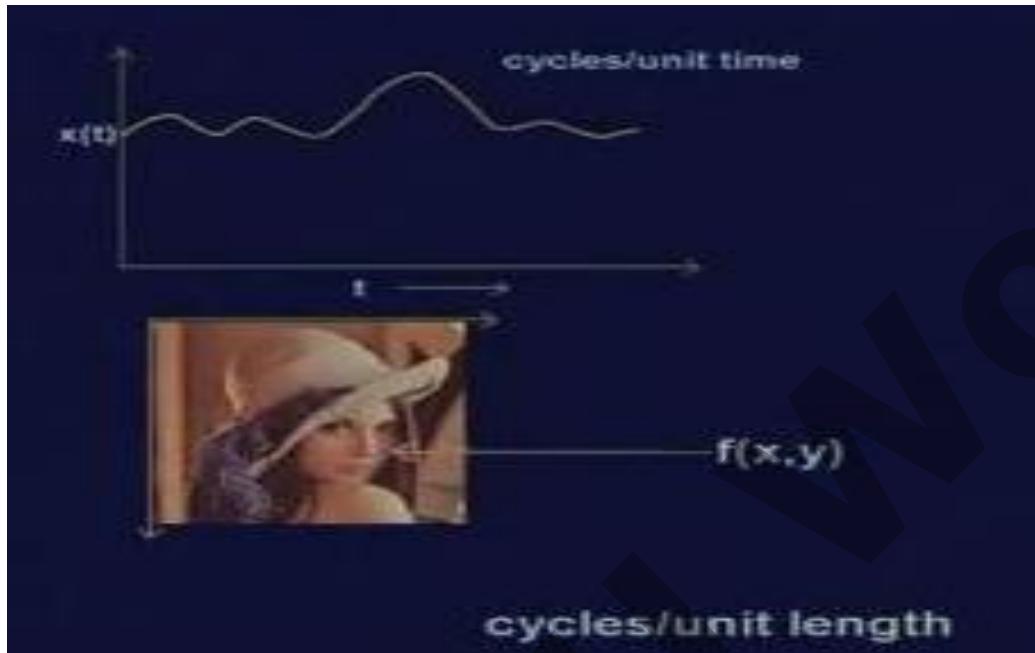
An image when it is digitized will be represented in the form of a matrix like this

A continuous image $f(x,y)$ is normally approximated by equally spaced samples arranged in the form of an $N \times M$ array where each element of the array is a discrete quantity.

$$f(x, y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, M-1) \\ f(1,0) & f(1,1) & \dots & f(1, M-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1, M-1) \end{bmatrix}$$

So, whatever process we have done during digitization; during visualization or during display, we must do the reverse process. So, for displaying the images, it has to be first converted into the analog signal which is then displayed on a normal display.

for an image, when you measure the frequency; it has to be cycles per unit length, not the cycles per unit time as is done in case of a time varying signal.



Now, in this figure we have shown that as in case of the signal $X(t)$, we had its frequency spectrum represented by $X(w)$ and we say that the signal $X(t)$ is **band limited** if $X(\omega)$ is equal to 0 for $\omega > \omega_0$ where ω_0 is the bandwidth of the signal $X(t)$.

Similarly, in case of an image, because the image is a 2 dimensional signal which is a variable, which is a function of 2 variables x and y ; so it is quite natural that in case of image, we will have frequency components which will have 2 components - one in the x direction and other in the y direction. So we call them, ω_x and ω_y .

So, you see that the image is band limited if $f(\omega_x, \omega_y)$ is equal to 0 for $\omega_x > \omega_{x0}$ and $\omega_y > \omega_{y0}$. So in this case, the maximum frequency component in the x direction is ω_{x0} and the maximum frequency component in the y direction is ω_{y0} .

2-D Sampling

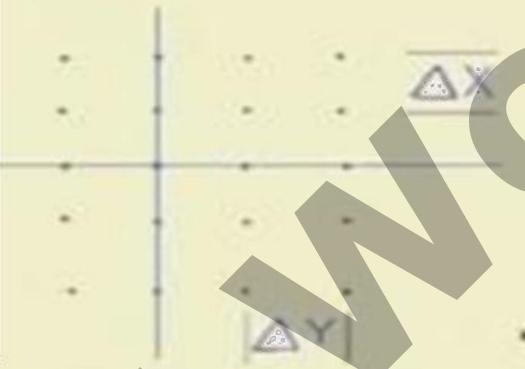
$$f_s(x, y) = f(x, y) \text{comb}(x, y; \Delta x, \Delta y)$$

$$= \sum_{m, n=-\infty}^{\infty} f(m \Delta x, n \Delta y) \delta(x - m \Delta x, y - n \Delta y)$$

$$X_s(t) = X(t) \cdot \text{comb}(t, \Delta t)$$

$$= \sum_{m=-\infty}^{\infty} X(m \Delta t) \delta(t - m \Delta t)$$

$$\text{comb}(t; \Delta t) = \sum_{m=-\infty}^{\infty} \delta(t - m \Delta t)$$



$$\text{comb}(t; \Delta t) = \sum_{m=-\infty}^{\infty} \delta(t - m \Delta t)$$

Now, let us see what will happen in case of 2 dimensional sampling or when we try to sample an image.

The original image is represented by the function f of x y and as we have seen, in case of a TWO 1 dimensional signal that if x of t is multiplied by comb of t delta t for the sampling operation; in case of image also $f(x, y)$ has to be multiplied by comb of x y delta x y to give you the sampled signal $f_s(x, y)$.

Now, this comb function because it is again a function of 2 variables x and y is nothing but a 2 dimensional array of the delta functions where along x direction, the spacing is delta x and along y direction, the spacing is delta y .

So again, as before, this $f_s(x, y)$ can be represented in the form $f(m \Delta x, n \Delta y)$ multiplied by delta function x minus $m \Delta x$, y minus $n \Delta y$ where both m and n varies from minus infinity to infinity.

2-D Sampling

Following similar argument as in 1-D case

$$F_s(\omega_x, \omega_y) = F(\omega_x, \omega_y) \otimes COMB(\omega_x, \omega_y)$$

$$COMB(\omega_x, \omega_y) = \Im\{comb(x, y; \Delta x, \Delta y)\}$$

$$\begin{aligned} &= \omega_{xs} \otimes \omega_{ys} \sum_{m, n = -\infty}^{\infty} \delta(\omega_x - m\omega_{xs}, \omega_y - n\omega_{ys}) \\ &= \omega_{xs} \omega_{ys} comb(\omega_x, \omega_y; \frac{1}{\Delta x}, \frac{1}{\Delta y}) \end{aligned}$$

where

$$\omega_{xs} = \frac{1}{\Delta x} = \text{sampling frequency along } x$$

$$\omega_{ys} = \frac{1}{\Delta y} = \text{sampling frequency along } y$$

So, as we have done in case of 1 dimensional signal; if we want to find out the frequency spectrum of this sampled image, then the frequency spectrum of the sampled image $f_s(\omega_x, \omega_y)$ will be same as $F(\omega_x, \omega_y)$ which is the frequency spectrum of the original image $f(x, y)$ which has to be convoluted with $comb(\omega_x, \omega_y)$ where $comb(\omega_x, \omega_y)$ is nothing but the Fourier transform of $comb(x, y; \Delta x, \Delta y)$.

And if you compute this Fourier transform, you find that $comb(\omega_x, \omega_y)$ will come in the form of $\omega_{xs} \otimes \omega_{ys} \cdot comb(\omega_x, \omega_y, 1/\Delta x, 1/\Delta y)$ where this ω_{xs} and this ω_{ys} , ω_{xs} is nothing but $1/\Delta x$ which is the sampling frequency along the x direction and ω_{ys} is equal to $1/\Delta y$ which is nothing but the sampling frequency along the y direction.

$$F_s(\omega_x, \omega_y) = F(\omega_x, \omega_y) \otimes \text{COMB}(\omega_x, \omega_y)$$

Region of support of $F(x, y)$

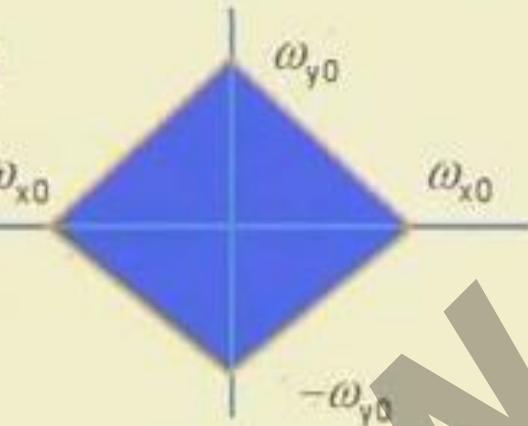


IMAGE QUANTIZATION

Quantization is mapping of a continuous variable u to a discrete variable u'

$$u' \in \{ r_1, r_2, r_3, \dots, r_L \}$$

u ----- quantization ----- u'

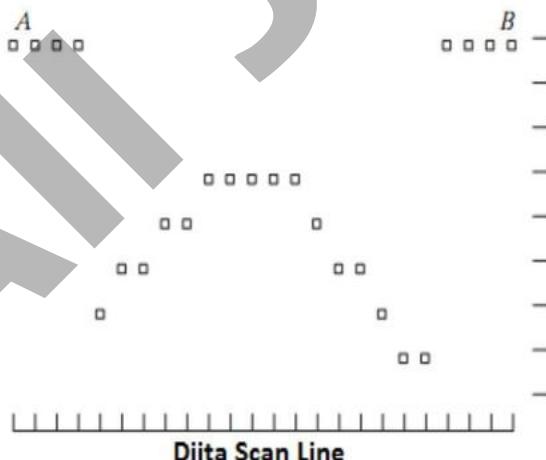
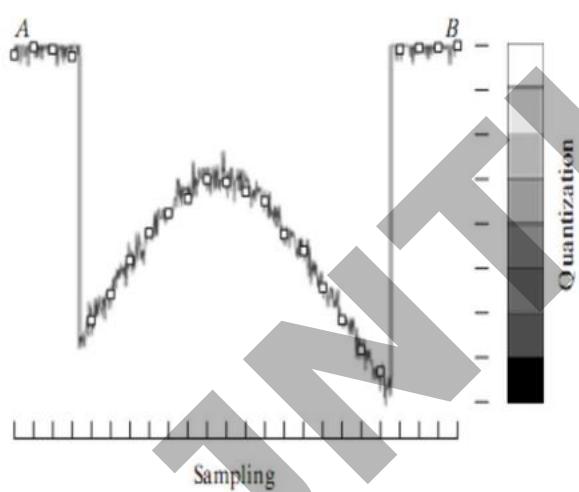
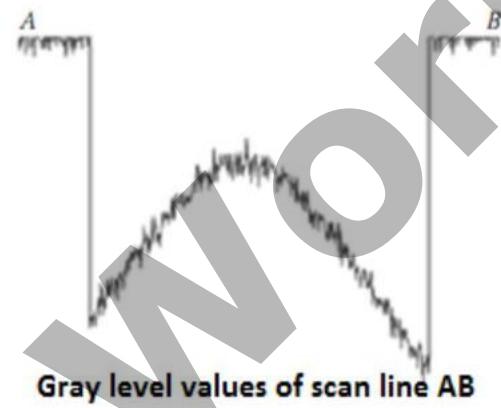
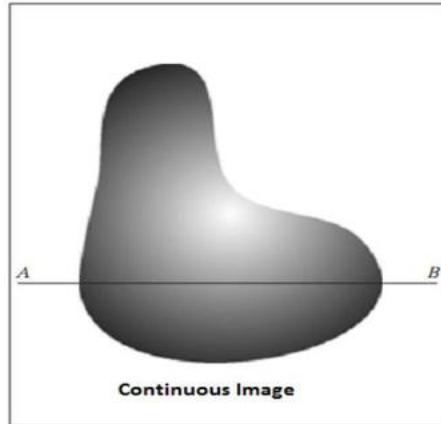
if your input signal is a u , after quantization the quantized signal becomes u' where u' is one of the discrete variables as shown in this case as r_1 to r_L . So, we have L number of discrete variables r_1 to r_L and u' takes a value of one of these variables.

. These samples are discrete in time domain. But still, every sample value is an analog value; it is not a discrete value. So, what we have done after sampling is instead of considering all possible time instants; we have considered the signal values at some discrete time instants and at each of this discrete time instants, I get a sample value. Now, the value of this sample is still an analog value.

$$u' = r_k \text{ if } t_k < u < t_{k+1},$$

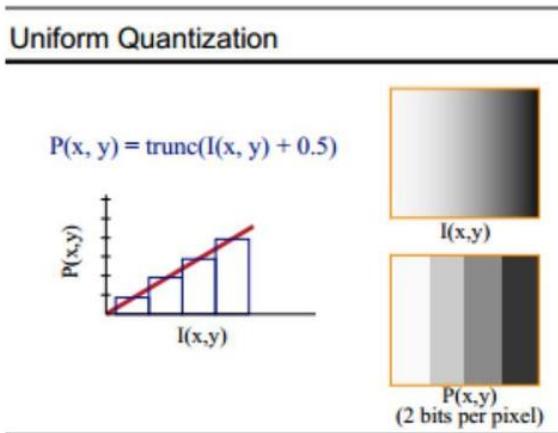
we have defined a number of transition levels or decision levels which are given as t_1, t_2, t_3, t_4 up to t_L plus 1 and here t_i is the minimum value and t_L plus 1 is the maximum value and we also defined a set of the reconstruction levels that is r_k . So, what we have shown in the previous slide that the reconstructed value r prime u'

takes one of the discrete values r_k . So, the quantized value will take the value r_k if the input signal u lies between the decision levels t_k and t_k plus 1. So, this is how you do the quantization.



Quantization

The basic idea behind sampling and quantization is illustrated in Figure. Figure shows a continuous image, $f(x, y)$, that we want to convert to digital form.



Similar is the case with an image. So here, in case of an image, the sampling is done in 2 dimensional grids where at each of the grid locations, we have a sample value which is still analog.

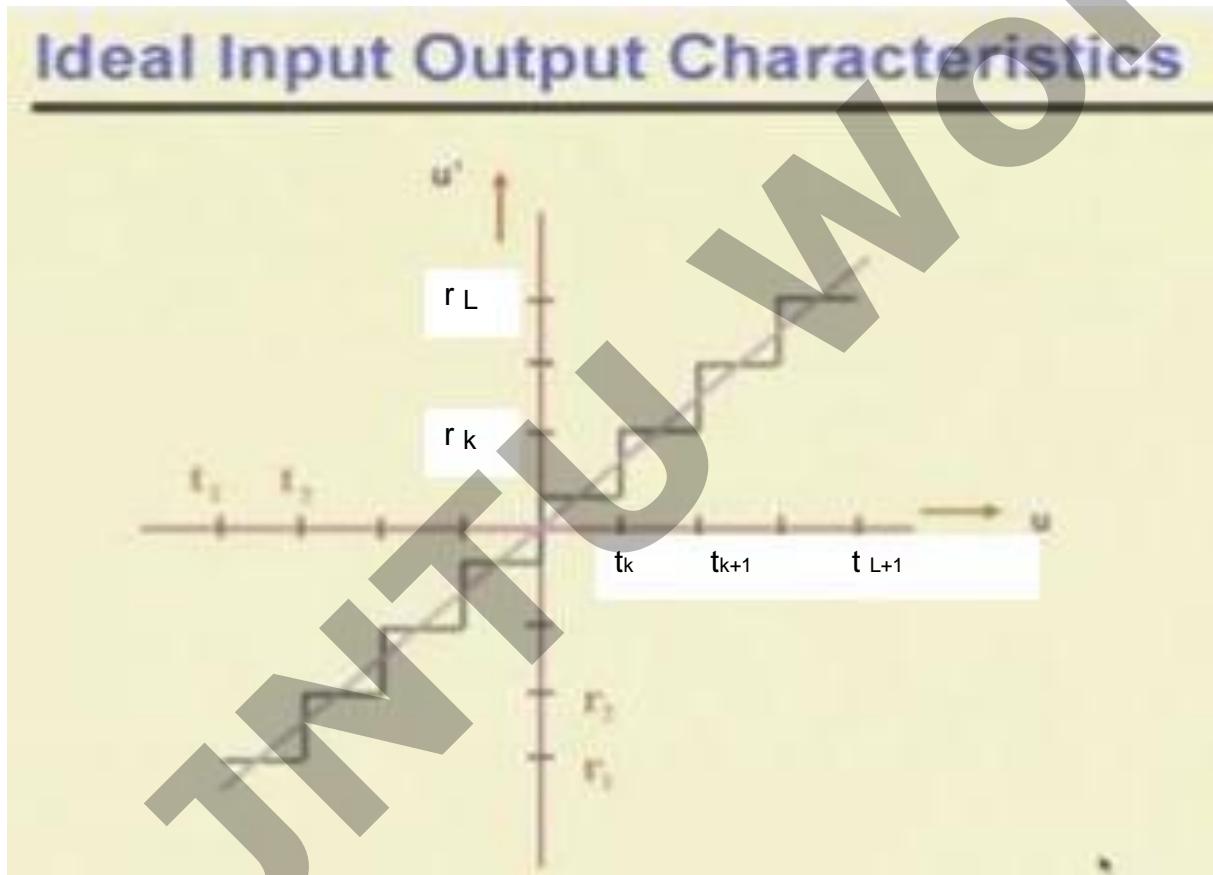
Now, if I want to represent a sample value on a digital computer, then this analog sample value cannot be represented and the quantization comes into picture.

Image quantization:

Mapping is generally a stair case function.

Define a set of decision or transition levels $[t_k, k=1,2,\dots,L+1]$ where t_1 is minimum value and t_{L+1} is maximum value.

So, this particular figure shows that if your input signal u lies between the transition levels t_1 and t_2 ; then the reconstructed signal or the quantized signal will take a value r_1 . If the input signal lies between t_2 and t_3 , the reconstructed signal or the quantized signal will take a value r_2 . Similarly, if the input signal lies between t_k and t_{k+1} , then the reconstructed signal will take the value of r_k and so on.

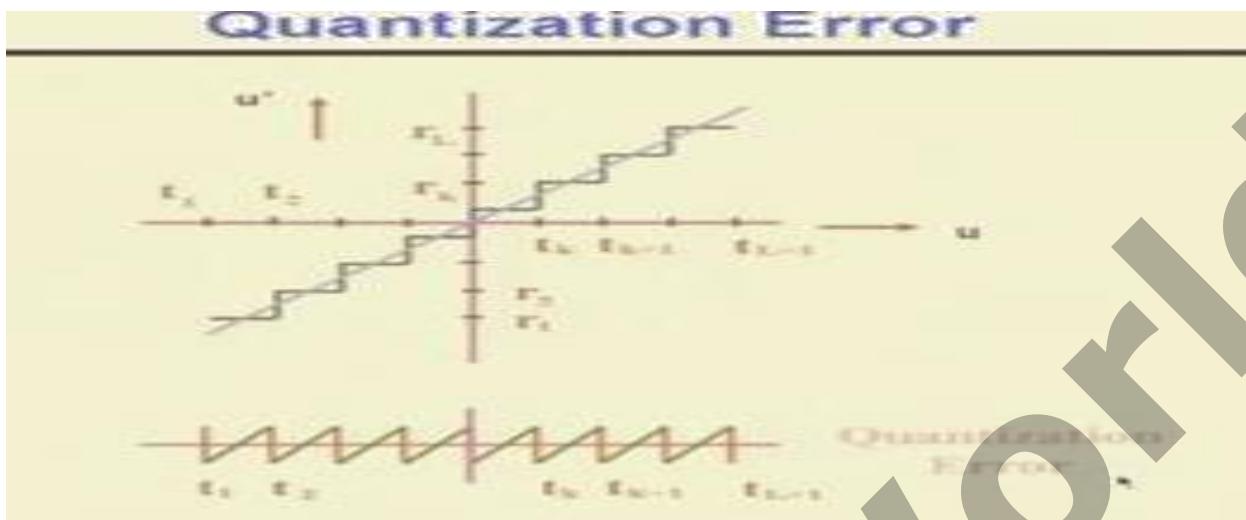


So, this peak staircase function shows what are the quantization function that will be used and this central linear line which is inclined at an angle of 45 degree with the u axis, this shows that what should be the ideal input output characteristics.

So, here we have shown the same figure. Here you find that whenever this green line which is inclined at 45 degree with the u axis crosses the staircase function; at this point, whatever is your signal value, it is same as the reconstructed value.

So, only at these cross over points, your error in the quantized signal will be 0. At all other points, the error in the quantized signal will be a non zero value. So, at this point, the error will be maximum, which will maximum and negative which will keep on reducing. At this point, this is going to be 0 and beyond this point, again it is going to increase.

So, if I plot this quantization error, you find that the plot of the quantization error will be something like this between every transition levels. So, between t_1 and t_2 the error value is like this, between t_2 and t_3 the error continuously increases, between t_3 and t_4 the error continuously increases and so on. Now, what is the effect of this error on the reconstructed signal?



So, because from the quantized signal I can never get back the original signal; so we are always introducing some error in the reconstructed signal which can never be recovered and this particular error is known as quantization error or quantization noise.

Obviously, the quantization error or quantization noise will be reduced if the quantizer step size that is the transition intervals say t_k to $t_k + 1$ reduces; similarly, the reconstruction step size r_k to $r_k + 1$ that interval is also reduced

So, for quantizer design, the aim of the quantizer design will be to minimize this quantization error. So accordingly, we have to have an optimum quantizer and this optimum **mean square error quantizer known as Lloyd-Max quantizer**, this minimizes the mean square error for a given number of quantization levels and here we assume that let u be a real scalar random variable with a continuous probability density function p_u of u and it is desired to find the decision levels t_k and the reconstruction levels r_k for an n L level quantizer which will reduce or minimize the quantization noise or quantization error.

The *quality of a digital image* is determined to a large degree by the number of samples and discrete intensity levels used in sampling and quantization.

Discrete Fourier transform in image processing ...

In image processing,

The Fourier Transform is used

in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components.

The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent.

In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

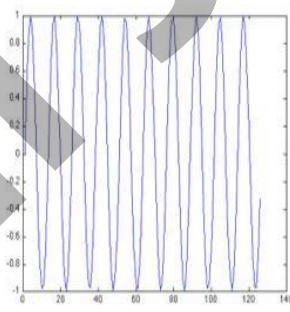
How does it work?

Discrete Fourier Transform is a variant of Fourier Transform and is used in digital image processing and is known as **2D DFT**.

The DFT is a Fourier Transform on sampled signal or image.

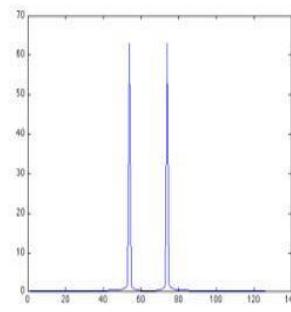
The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size.

1D and 2D Fourier transform comparison



(a) Time Domain

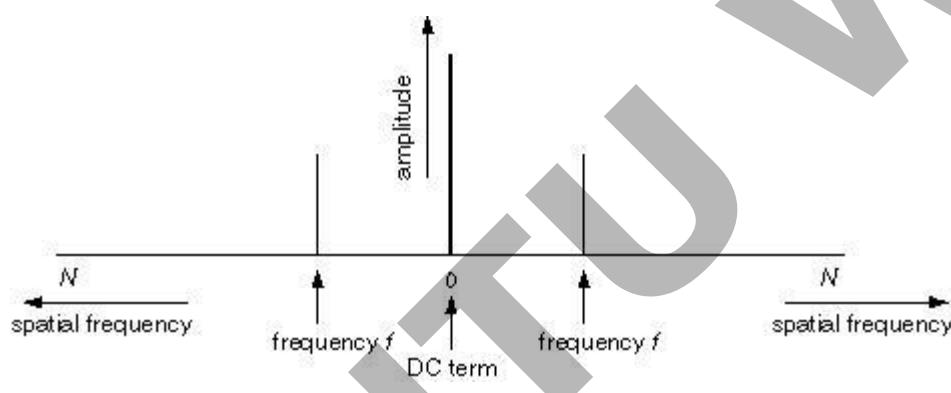
FFT
Fast Fourier
Transform



(b) Frequency Domain

Position of peaks will depend on signal frequency

On 2D images, you are watching the waveform from top view You can imagine that the white parts are top of the wave and black parts are the bottom parts (see picture on the right) Fourier transform:



In the 2D-image Fourier transforms, there is always a peak at origin The value at origin is the average intensity/color of the image.

No negative values In the 2D-image

Fourier transform, higher frequencies represent detail in a picture and lower frequencies represent the gradient changes in the picture

The above feature is used to apply filters to images

The 2-D discrete Fourier transform

The DFT is a transform of a discrete, complex 2-D array of size $M \times N$ into another discrete, complex 2-D array of size $M \times N$

Both $T(u,v)$ and $f(x,y)$ are 2-D periodic

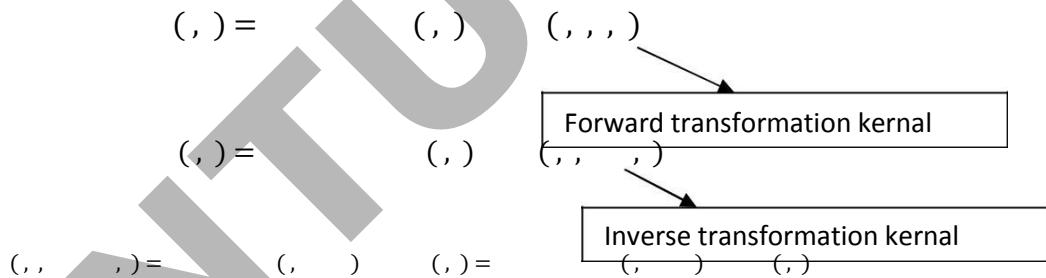
One view of the DFT is as an approximation to the CFT

$f(x,y)$ is the image in the spatial domain and the exponential term is the basis function corresponding to each point $F(u,v)$ in the Fourier space.

The equation can be interpreted as: the value of each point $T(u,v)$ is obtained by multiplying the spatial image with the corresponding base function and summing the result.

$$T(u,v) = \sum f(x,y) \cdot \text{Basis fn}$$

The basis functions are sine and cosine waves with increasing frequencies, i.e. $F(0,0)$ represents the DC-component of the image which corresponds to the average brightness and $F(M-1,N-1)$ represents the highest frequency.



Forward transformation kernel and Inverse transformation kernel:

Let us see what is that class of transformation. You will find that if we define a transformation of this form say $T(u,v)$ is equal to double summation $f(x,y)$ where $f(x,y)$ is the 2 dimensional signal into $g(x,y,u,v)$ where both x and y vary from 0 to capital

$N - 1$. So, we are assuming that our 2 dimensional signal $f(x, y)$ is an N by N array, capital N by capital N array and the corresponding inverse transformation is given by $f(x, y)$ is equal to double summation again, we have this transformation matrix transform coefficients $T(u, v)$ into $h(x, y, u, v)$ where this $g(x, y, u, v)$ is called the forward transformation kernel and $h(x, y, u, v)$ is called the inverse transformation kernel or the basis functions.

2 dimensional discrete Fourier transform (2D DFT)

we had $g(x, y, u, v)$ which was of this form $e^{-j2\pi \frac{ux}{N}}$ plus $e^{-j2\pi \frac{vy}{N}}$ and of course, we had this multiplicity term $\frac{1}{N}$ upon capital N . So, this was in the **forward transformation kernel** in case of 2 dimensional discrete Fourier transform or 2D DFT.

For $N \times N$ square image,

Forward transformation kernel or basis function is:

$$(x, y, u, v) = \frac{1}{N} e^{-j2\pi \frac{ux}{N}} e^{-j2\pi \frac{vy}{N}}$$

$$(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} e^{-j2\pi \frac{ux}{N}} e^{-j2\pi \frac{vy}{N}}$$

In Cos and Sin form,

$$(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \left[\frac{1}{2} [e^{-j2\pi \frac{ux}{N}} e^{-j2\pi \frac{vy}{N}} + e^{j2\pi \frac{ux}{N}} e^{j2\pi \frac{vy}{N}}] + j \frac{1}{2} [e^{-j2\pi \frac{ux}{N}} e^{-j2\pi \frac{vy}{N}} - e^{j2\pi \frac{ux}{N}} e^{j2\pi \frac{vy}{N}}] \right]$$

Inverse transformation kernel or basis function is :

$$(\quad, \quad, \quad) = \underline{\hspace{2cm}}$$

$$(,) = \underline{\hspace{2cm}} \quad (,)^0 \quad (+)$$

For M X N Image,

In the 2-D case:

forward DFT $T(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)]$

for $u=0 \rightarrow M-1$ and $v=0 \rightarrow N-1$

Kernal

Inverse DFT $f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) \exp[j2\pi(ux/M + vy/N)]$

for $x=0 \rightarrow M-1$ and $y=0 \rightarrow N-1$

Kernal

The discrete function $f(x, y)$ represents samples of the continuous function at $f(x_0 + x\Delta x, y_0 + y\Delta y)$

$\Delta u = 1/(M\Delta x)$ and $\Delta v = 1/(N\Delta y)$

Relation between spatial sample space and frequency intervals

$\Delta u, \Delta v$ relations shown above express the relation between spatial sample space and frequency intervals

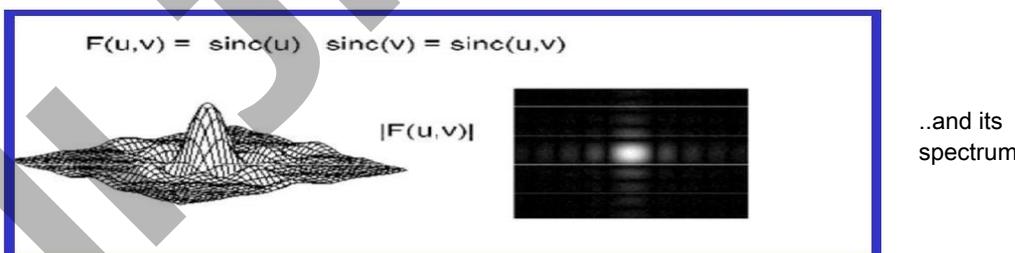
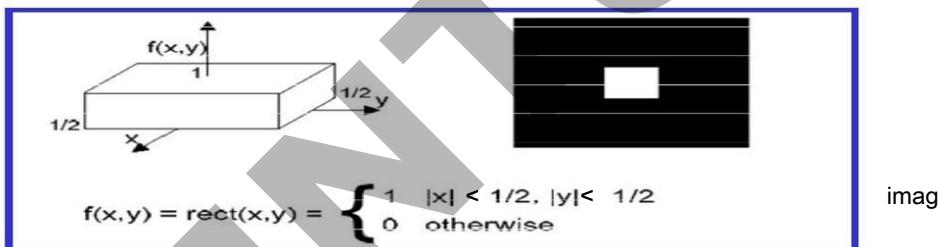
$$\Delta u = \frac{1}{M} \quad \Delta v = \frac{1}{N}$$

There are various forms of the FFT and most of them restrict the size of the input image that may be transformed, often to where n is an integer.

$$T(u, v) = \frac{1}{MN} \left[\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{2\pi ux}{M} + \frac{2\pi vy}{N}\right) - \right. \\ \left. j \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \sin\left(\frac{2\pi ux}{M} + \frac{2\pi vy}{N}\right) \right]$$

$T(u, v)$ is the component with frequency $u < M/2, v < N/2$

M, N are image sizes, $f(x, y)$ is pixel intensity at position x, y



Magnitude and Phase

The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the real and imaginary part or with magnitude and phase.

In image processing, often only the magnitude of the Fourier Transform is displayed, as it contains most of the information of the geometric structure of the spatial domain image.

The Fourier image can also be re-transformed into the correct spatial domain after some processing in the frequency domain...(both magnitude and phase of the image must be preserved for this). The Fourier domain image has a much greater range than the image in the spatial domain. Hence, to be sufficiently accurate, its values are usually calculated and stored in float values.

$$\text{2D DFT pair } f(x,y) \longleftrightarrow F(u,v) \quad \text{OR} \quad T(u,v)$$

$$F(u,v) = R(u,v) + jI(u,v)$$

real part

imaginary part

(Fourier)
Magnitude spectrum

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$$

(Fourier)
Phase spectrum

$$\phi(u,v) = \angle F(u,v) = \tan^{-1} \left[\frac{I(u,v)}{R(u,v)} \right]$$

Power spectrum

$$P(u,v) = |F(u,v)|^2$$

- ▶ Each DFT coefficient is a complex value
 - ▶ There is a single DFT coefficient for each spatial sample
 - ▶ A complex value is expressed by two real values in either Cartesian or polar coordinate space.

- ▶ Cartesian: $R(u,v)$ is the *real* and $I(u, v)$ the *imaginary component*
 - ▶ Polar: $|F(u,v)|$ is the *magnitude* and $\phi(u,v)$ the *phase*
-
- ▶ Notes on the magnitude spectrum:
 - ▶ Magnitudes are generally referred to as the “spectrum” but this should be understood as the magnitude spectrum.
 - ▶ Typically has an extremely large dynamic range and it is typical to log-compress those values for display
 - ▶ For presentation, the DC component, $F(0,0)$, is placed at the center. Low frequency components are shown near the center and frequency increases with distance from center.
 - ▶ The magnitude spectrum contains information about the shape of objects.
 - ▶ A strong edge in the source will generate a strong edge in the magnitude spectrum (rotated 90 degrees)
 - ▶ The phase spectrum contains information about their actual location in the source.
 - ▶ An image of lots of ‘Q’s will have the same magnitude spectra but not the same phase spectra.

Properties of the 2-D Fourier transform

Display

The dynamic range of the Fourier spectra is generally higher than can be displayed

- A common technique is to display the function

$$D(u, v) = c \log[1 + |F(u, v)|]$$

- where c is a scaling factor and the logarithm function performs a “compression” of the data, c is usually chosen to scale the data into the range of the display device, [0-255] typically ([1-256] for 256 gray-level MATLAB image)

Separability and Symmetric

Kernel Properties

A kernel is said to be separable if

$$g(x, y, u, v) = g_1(x, u)g_2(y, v)$$

A kernel is said to be symmetric if

$$g(x, y, u, v) = g_1(x, u)g_1(y, v)$$

Now, these transformations, this class of transformation will be separable if we can write $g(x, y, u, v)$ in the form $g_1(x, u)$ into $g_2(y, v)$. So, if $g(x, y, u, v)$ can be written in the form $g_1(x, u)$ into $g_2(y, v)$, then this transformation will be a separable transformation.

So, in that case, these class of transformations will be separable obviously because $g(x, y, u, v)$ we have written as product of 2 functions : $g_1(x, u)$ into $g_2(y, v)$. And since $g_1(x, u)$ and $g_2(y, v)$; so this function g_1 and g_2 , they are functionally same, so this I can write as $g_1(x, u)$ into $g_1(y, v)$ and in this case, the function will be called as symmetric.

So here, what we have is this particular transformation or class of transformations is called separable as well as symmetric and the same is also true for the inverse transformation kernel that is $h(x, y, u, v)$.

$$e^{-j2\pi(\frac{ux+vy}{N})} = e^{-j2\pi(\frac{ux}{N})} e^{-j2\pi(\frac{vy}{N})}$$

The discrete transform pair can be written in separable forms

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \exp[-j2\pi ux/N] \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi vy/N]$$

for $u, v = 0, 1, \dots, N-1$

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \exp[j2\pi ux/N] \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi vy/N]$$

for $x, y = 0, 1, \dots, N-1$

So, $F(u, v)$ or $f(x, y)$ can be obtained in 2 steps by successive applications of the 1-D Fourier transform or its inverse.

2D DFT can be implemented as a series of 1D DFTs along each column, followed by 1D DFTs along each row.

Translation /Modulation/shifting

The translation properties of the Fourier transform pair are

$$f(x, y) \exp[j2\pi(u_0x + v_0y)/N] \Leftrightarrow F(u - u_0, v - v_0)$$

and

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) \exp[-j2\pi(ux_0 + vy_0)/N]$$

where the double arrow indicates a correspondence between a function and its Fourier transform (or vice versa)

Multiplying $f(x, y)$ by the exponential and taking the transform results in a shift of the origin of the frequency plane to the point (u_0, v_0) .

For our purposes, $u_0=v_0=N/2$. Therefore,

$$\begin{aligned}\exp[j2\pi(u_0x+v_0y)/N] &= e^{j\pi(x+y)} \\ &= (-1)^{x+y}\end{aligned}$$

and

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-N/2, v-N/2)$$

So, the origin of the Fourier transform of $f(x,y)$ can be moved to the center of the corresponding $N \times N$ simply by multiplying $f(x,y)$ by $(-1)^{x+y}$ before taking the transform.

Note: This does not affect the magnitude of the Fourier transform

- ▶ **Translation** of the source will cause the phase spectrum to change but leave the magnitude spectrum unchanged since the phase spectrum encodes location information while the magnitude spectrum encodes shape information.

Conclusion: the origin of the Fourier transform can be moved to the center of the frequency square by multiplying the original function by $(-1)^{x+y}$

Distributivity. Fourier transform is distributive over addition (not multiplication)

$$\vec{\mathcal{F}}(f+g) = \vec{\mathcal{F}}(f) + \vec{\mathcal{F}}(g)$$

The Fourier transform (and its inverse) are distributive over addition but not over multiplication. • So,

$$\Im\{f_1(x,y) + f_2(x,y)\} = \Im\{f_1(x,y)\} + \Im\{f_2(x,y)\}$$

$$\Im\{f_1(x,y) \times f_2(x,y)\} \neq \Im\{f_1(x,y)\} \times \Im\{f_2(x,y)\}$$

Rotation of the source corresponds to an identical rotation of the magnitude and phase spectra

Periodicity of the Fourier transform

The discrete Fourier transform (and its inverse) are *periodic* with period N.
 $T(u,v) = T(u+N,v) = T(u,v+N) = T(u+N,v+N)$

- Although $T(u,v)$ repeats itself infinitely for many values of u and v , only N values of each variable are required to obtain $f(x,y)$ from $T(u,v)$
 - i.e. Only one period of the transform is necessary to specify $T(u,v)$ in the frequency domain.
 - Similar comments may be made for $f(x,y)$ in the spatial Domain

$$\begin{aligned} (+, +) &= \sum \sum \\ (-, +) &= - \\ (-, -) &= - \\ (+, -) &= - \\ (+, +) &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(xu+yv)/N} \\ (-, -) &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) e^{j2\pi(xu+yv)/N} \end{aligned}$$

Conjugate

If $f(x,y)$ is real (true for all of our cases), the Fourier transform exhibits conjugate symmetry

$F(u,v) = F^*(-u,-v)$ or, the more interesting

$|F(u,v)| = |F(-u,-v)|$ where $F^*(u,v)$ is the complex conjugate of $F(u,v)$

Scale

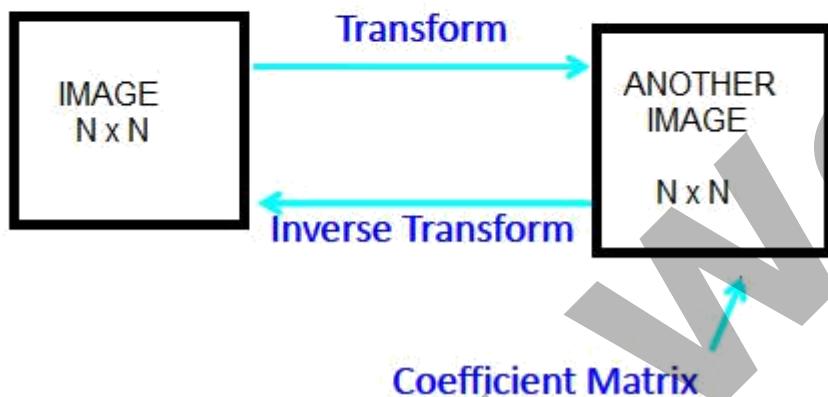
For two scalars a and b ,

$$(,) \leftrightarrow (,)$$

The DFT coefficients produced by the 2D DFT equations [here](#), are arranged in a somewhat awkward manner as shown in the diagram below.

Introduction

What is Image Transform?



Applications: USE OF IMAGE TRANSFORMS

Preprocessing

- Filtering -
- Enhancement, etc

Data Compression

Feature Extraction

- Edge Detection
- Corner detection, etc

What does the Image Transform do?

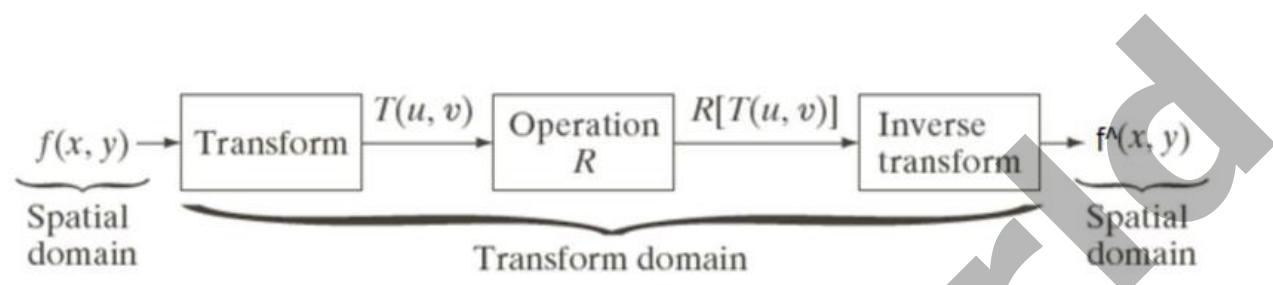
It represents the given image as a series summation of a set of Unitary Matrices.

What is a Unitary Matrix?

A Matrix 'A' is a unitary matrix if

$A^{-1} = A^*T$ where A^* is conjugate of A
Unitary Matrix -----> Basis Function

General approach for operating in linear transform domain



$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) g(x, y, u, v)$$

Forward transform of $f(x, y)$

Input image

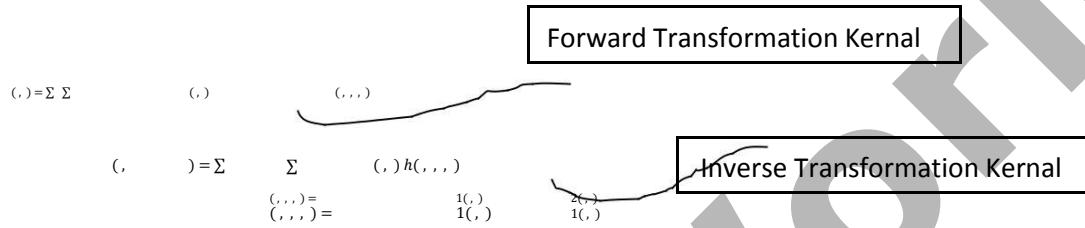
Forward transformation kernel

Recover $f(x, y)$:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) h(x, y, u, v)$$

Inverse transformation kernel

Discrete Fourier transformation is nothing but a special case of a class of transformations or a class of separable transformations



Forward transformation kernel and Inverse transformation kernel:

Let us see what is that class of transformation. You will find that if we define a transformation of this form say $T(u, v)$ is equal to double summation $f(x, y)$ where $f(x, y)$ is the 2 dimensional signal into $g(x, y, u, v)$ where both x and y vary from 0 to capital N minus 1. So, we are assuming that our 2 dimensional signal $f(x, y)$ is an N by N array, capital N by capital N array and the corresponding inverse transformation is given by $f(x, y)$ is equal to double summation again, we have this transformation matrix transform coefficients $T(u, v)$ into $h(x, y, u, v)$ where this $g(x, y, u, v)$ is called the forward transformation kernel and $h(x, y, u, v)$ is called the inverse transformation kernel or the basis functions.

Now, these transformations, this class of transformation will be separable if we can write $g(x, y, u, v)$ in the form $g_1(x, u)$ into $g_2(y, v)$. So, if $g(x, y, u, v)$ can be written in the form $g_1(x, u)$ into $g_2(y, v)$, then this transformation will be a separable transformation.

So, in that case, these class of transformations will be separable obviously because $g(x, y, u, v)$ we have written as product of 2 functions - $g_1(x, u)$ into $g_2(y, v)$. And since $g_1(x, u)$ and $g_2(y, v)$; so this function g_1 and g_2 , they are functionally same, so this I can write as $g_1(x, u)$ into $g_1(y, v)$ and in this case, the function will be called as symmetric.

So here, what we have is this particular transformation or class of transformations is called separable as well as symmetric and the same is also true for the inverse transformation kernel that is $h(x, y, u, v)$.

2 dimensional discrete Fourier transformation

we had $g(x, y, u, v)$ which was of this form $e^{-j2\pi/N}$ into ux plus vy and of course, we had this multiplicity term 1 upon capital N. So, this was in the forward transformation kernel in case of 2 dimensional discrete Fourier transform or 2D DFT.

$$(x, y, u, v) = \frac{1}{N} e^{-j2\pi/N(ux + vy)}$$

$$(x, y, u, v) = 1(x, y) 1(u, v)$$

$$(x, y, u, v) = \frac{1}{\sqrt{N}} e^{-j2\pi/N(\sqrt{u^2 + v^2})} e^{-j2\pi/N(u x + v y)}$$

ADVANCED LEVEL TRANSFORMS

Until now we considered discrete sinusoidal transforms. Here, we will give and brief overview of advanced level transforms used in digital image processing

Discrete Fourier Transform

- Due to its computational efficiency the DFT is very popular . • However, it has strong disadvantages for some applications
 - It is complex
 - It has poor energy compaction
- Energy compaction – is the ability to pack the energy of the spatial sequence into as few frequency coefficients as possible
 - This is very important for image compression.. if compaction is high, we only have to transmit a few coefficients – instead of the whole set of pixels

DCT

The first thing to note is that there are various versions of the DCT these are usually known as DCT-I to DCT-IV

- They vary in minor details
- the most popular is the DCT-II, also known as even symmetric DCT, or as “the DCT DCT is not complex but real valued.

$$f(u, v) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$
$$\begin{aligned} f(u, v) &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x, 0) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \\ f(u, v) &= \frac{1}{\sqrt{N}} f(0, v) + \sum_{x=1}^{N-1} f(x, 0) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \\ f(u, v) &= \frac{1}{\sqrt{N}} f(0, v) + \sum_{x=1}^{N-1} f(x, 0) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \end{aligned}$$

So, you find that in case of discrete cosine transformation, if you analyze this, you find that both the forward transformation kernel and also the inverse transformation kernel, they are identical and not only that, these transformation kernels are separable as well as symmetric because in this ; $g_1(x, u)$ equal to alpha u cosine twice x plus 1 $u\pi$ divided by twice N and $g_1(y, v)$ can be alpha times v into cosine 2y plus 1 $v\pi$ upon twice N.

So, this transformation that is discrete cosine transformation is separable as well as symmetric and the inverse forward inverse transformation kernel and the forward transformation kernel, they are identical.

Now, we have to see what the values of alpha u and alpha v. Here, alpha u is given by square root of 1 upon capital N where u is equal to 0 and it is equal to square root of twice by capital N for values of u equal to 1, 2 to capital N minus 1.

let us see how the basis functions or the basis images look like in case of discrete cosine transform.

Now, using these kernels, now we can write the expressions for the 2 dimensional discrete cosine transformation in the form of $c(u, v)$

$$(,)$$

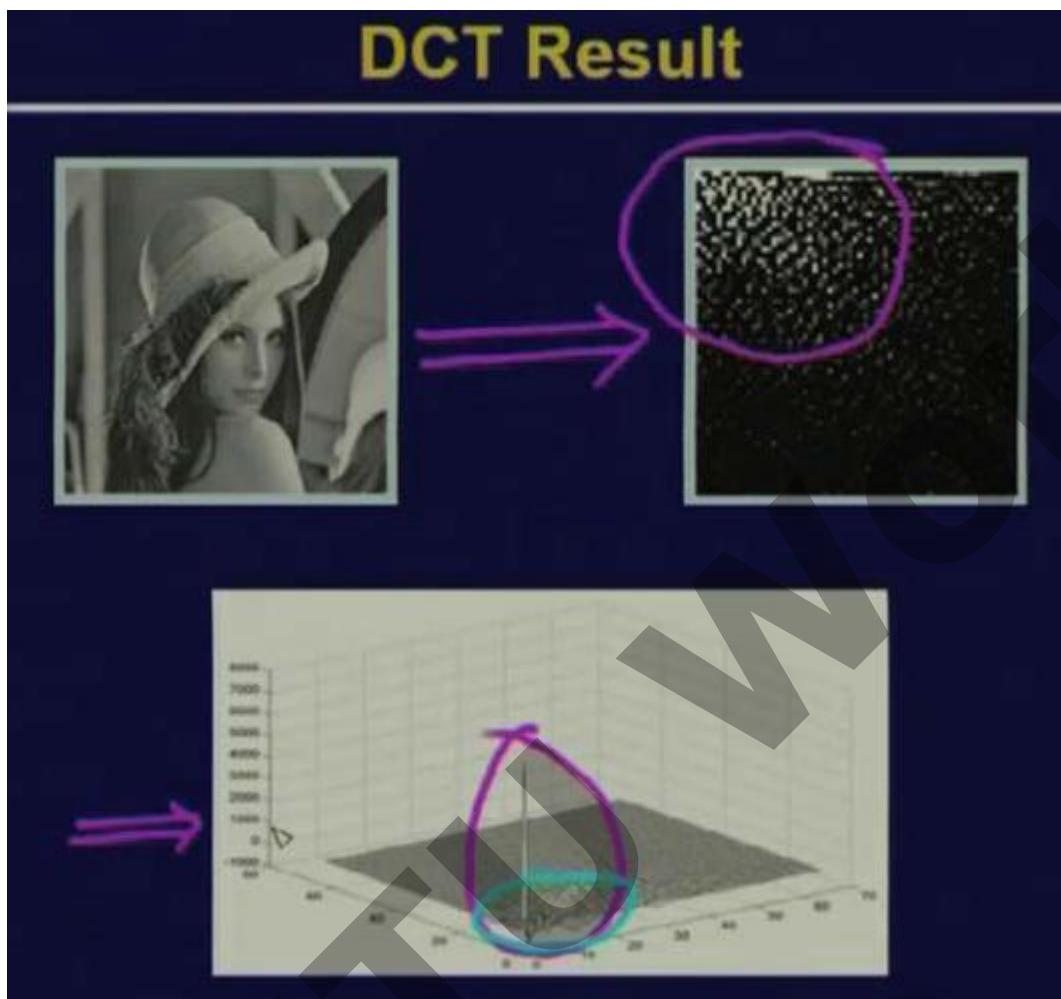
$$= (\cdot \cdot) \cdot \cdot \cos \frac{(2 + 1)}{2} \cos \frac{(2 + 1)}{2}$$

$$(,)$$

$$= (\cdot \cdot) (\cdot \cdot) \cos \frac{(2 + 1)}{2} \cos \frac{(2 + 1)}{2}$$

Now, you find that there is one difference in case of forward discrete cosine transformation. The terms $\alpha(u)$ and $\alpha(v)$ were kept outside the summation, double summation whereas incase of inverse discrete cosine transformation, the terms $\alpha(u)$ and $\alpha(v)$ are kept inside the double summation. The reason being, in case of forward transformation because the summation is taken over x and y varying from 0 to capital N minus 1, so $\alpha(u)$ and $\alpha(v)$, these terms are independent of this summation operation whereas, in case of inverse discrete cosine transformation, the double summation is taken over u and v varying from 0 to capital N minus 1. So, this terms $\alpha(u)$ and $\alpha(v)$ are kept ah are kept inside the double summation operation.

DCT Result



Now, if you closely look at this output coefficients, you find that in case of discrete cosine transformation, the energy of the coefficients are concentrated mostly in a particular region where the coefficients are near the origin that is $(0, 0)$ which is more visible in the case of a 3 dimensional plot. So, you find that here in this particular case, the energy is concentrated in a small region in the coefficients space near about the $(0, 0)$ coefficients. So, this is a very very important property of the discrete cosine transformation which is called **energy compaction property**.

DFT	DCT
separable as well as symmetric	separable as well as symmetric
FFT for faster implementation	faster implementation of discrete cosine transformation or FDCT
discrete Fourier transform is periodic with period capital N where N is the number of samples	the magnitude of the coefficients are periodic with a period twice N where N is the number of samples
	periodicity is not same as incase of discrete Fourier transformation
	in case of discrete cosine transformation is twice of the period in case of discrete Fourier transformation and we will see later that this particular property helps to obtain data compression, the smother data compression using the discrete cosine transformation and not using the discrete Fourier transformation.
Energy is concentrated at the centre	energy compaction property because most of the signal energy or image energy is concentrated in a very few number of coefficients near the origin or near the (0, 0) value in the frequency domain in the uv plane.
Complex	Real

WALSH TRANSFORM

Suppose we have a function $f(x)$, $x = 0, \dots, N-1$ where $N = 2^n$.

If we use the ***binary representation*** for the argument x we need n bits to represent it.

We define a new transform called **Walsh Transform**

$$f(x) \Leftrightarrow W(u)$$

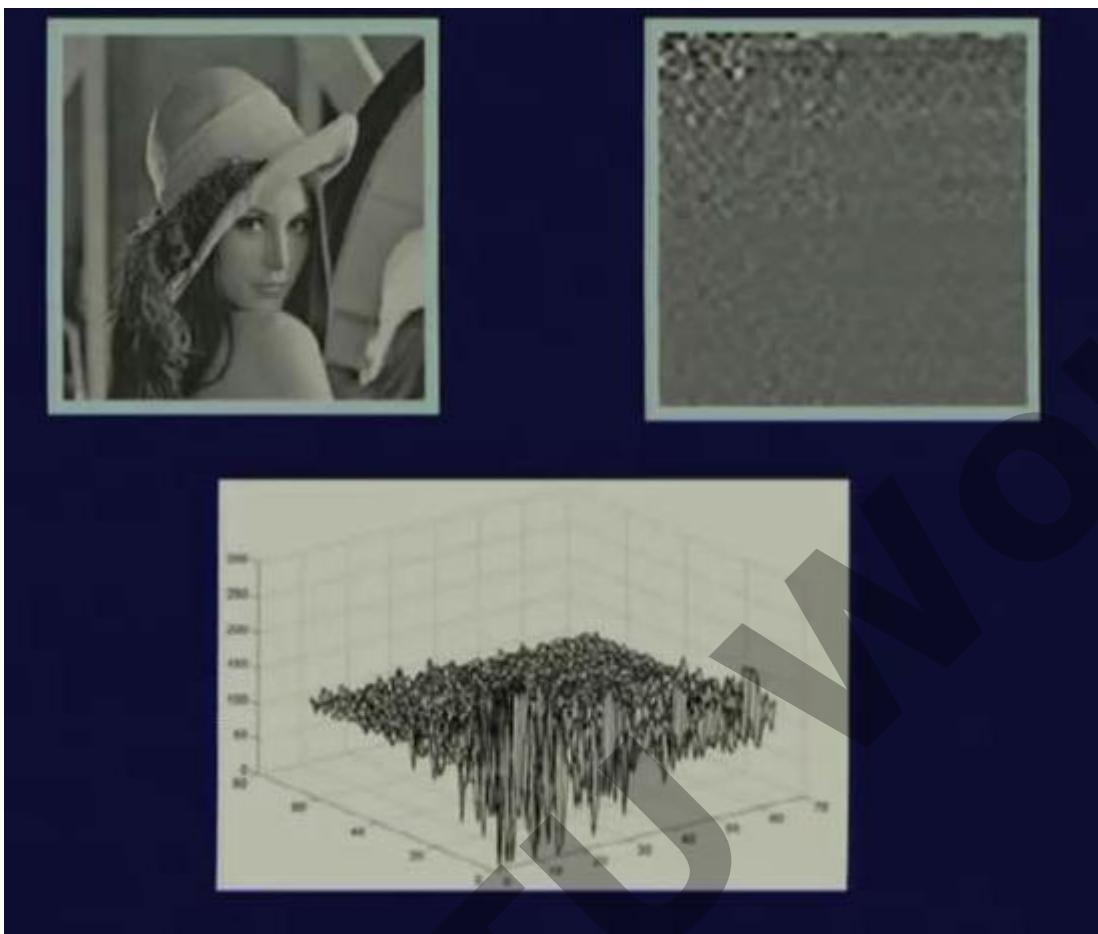
$$(x)_{10} = (b_{n-1}(x)b_{n-2}(x)\dots b_0(x))_2, \quad b_i(x) \text{ 0 or 1 } \text{ for } i = 0, \dots, n-1$$

$$(u)_{10} = (b_{n-1}(u)b_{n-2}(u)\dots b_0(u))_2$$

N is number of samples, n is number of bits for x or u , $b_k(x)$ is k th bit in digital binary representation of x

Example $f(x)$, $x=0,1,2,\dots,7$ (for 8 samples), $n=2$

$$f(6) = (110)_2 \quad \begin{array}{l} \xrightarrow{\hspace{1cm}} \\ \xrightarrow{\hspace{1cm}} \end{array} b_2(6)=1, \quad b_1(6)=1, \quad b_0(6)=0,$$



Walsh transform is separable and symmetric. Energy compaction is not as that of DCT

2-D WALSH TRANSFORM

Forward

$$W(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \left[\prod_{i=0}^{n-1} (-1)^{(b_i(x)b_{n-1-i}(u) + b_i(y)b_{n-1-i}(v))} \right]$$

or

$$W(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) (-1)^{\sum_{i=0}^{n-1} (b_i(x)b_{n-1-i}(u) + b_i(y)b_{n-1-i}(v))}$$

Inverse transform kernel is identical to forward transformation kernel

Note that

The forward and inverse Walsh kernels are identical for 2-D. This is because the array formed by the kernels is a symmetric matrix having orthogonal rows and columns, so its inverse array is the same as the array itself.

- ◆ The concept of frequency exists also in Walsh transform basis functions. We can think of frequency as the **number of zero crossings** or the number of transitions in a basis vector and we call this number **sequency**. The Walsh transform exhibits the property of **energy compaction**.

Hadamard transform

HADAMARD TRANSFORM

In a similar form as the Walsh transform, the 2-D Hadamard transform is defined as

Forward

$$H(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \left[\prod_{i=0}^{n-1} (-1)^{(b_i(x)b_i(u)+b_i(y)b_i(v))} \right] \text{ or}$$

$$H(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \underbrace{(-1)^{\sum_{i=0}^{n-1} (b_i(x)b_i(u)+b_i(y)b_i(v))}}_{} \quad$$

Forward Transformation Kernel

Inverse

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} H(u, v) \left[\prod_{i=0}^{n-1} (-1)^{(b_i(x)b_i(u) + b_i(y)b_i(v))} \right] \text{etc.}$$

$$N = 2^n$$

Forward and inverse transforms are same.

- ◆ An extended version of the Hadamard transform is the **Ordered Hadamard Transform** for which a fast algorithm called **Fast Hadamard Transform (FHT)** can be applied. This exhibits the energy compaction property.

- ◆ An important property of Hadamard transform is that, letting H_N represent the matrix of order N , the recursive relationship is given by the expression

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

- Here, we will introduce the Hadamard transform for $N=2^n$ samples. There are some alternative forms of Hadamard transform for other number of samples.
- Hadamard transform derivation begins with matrix:

$$\Xi_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Hadamard transform for $N=2$ is defined with transformation matrix:

$$H_2 = \frac{1}{\sqrt{2}} \Xi_2$$

- Hadamard transform of larger dimensions can be defined using recursively as:

$$H_N = \frac{1}{\sqrt{N}} \Xi_N \quad \text{where} \quad \Xi_N = \begin{bmatrix} \Xi_{N/2} & \Xi_{N/2} \\ \Xi_{N/2} & -\Xi_{N/2} \end{bmatrix}$$

Example for
 $N=8$:

$$H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 & 0 \\ 1 & 7 \\ 2 & 3 \\ 3 & 4 \\ 4 & 1 \\ 5 & 6 \\ 6 & 2 \\ 7 & 5 \end{matrix}$$

Number on the
end of each row
is number of
sign changes in
the row (from 1
to -1 and vice
versa).

Index sequency.

$$H_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Walsh Hadamard Transform

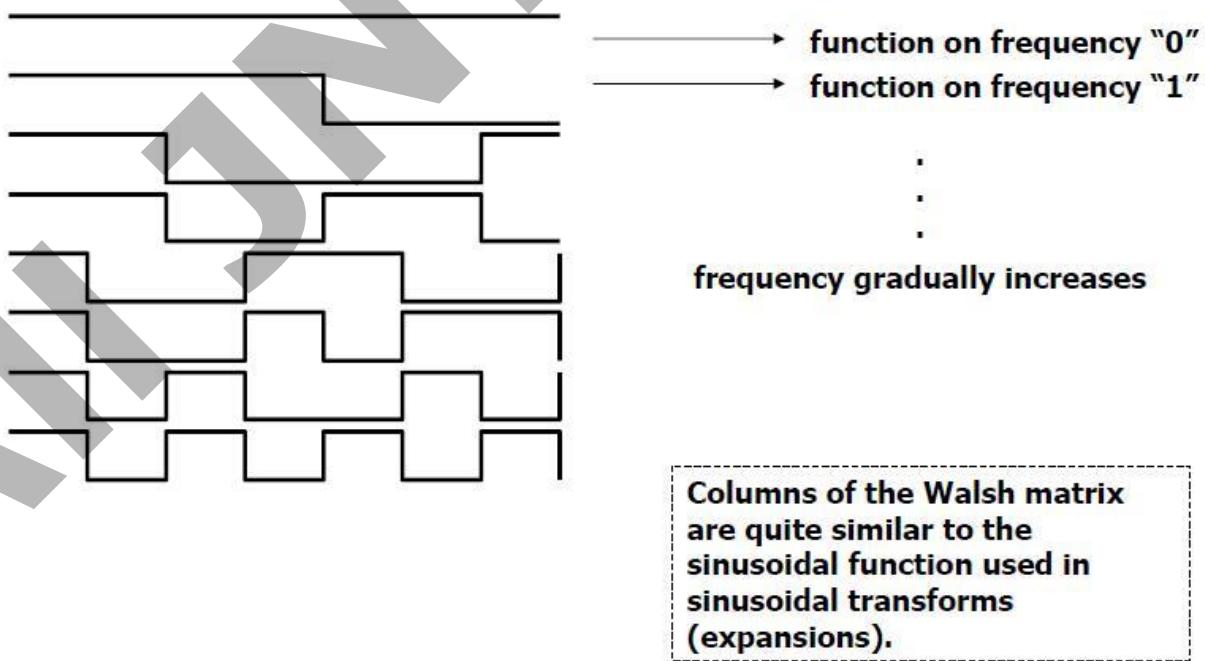
Walsh functions are orthogonal and have only +1 and -1 values. In general, the Walsh transform can be generated by the Hadamard matrix as follows:

Walsh ordering

Very often instead of the Hadamard transform we are using reordered form of the Hadamard transform (this reordering is called Walsh ordering and sometimes the corresponding transform is called the Walsh transform).

Reordering is performed in such manner that on the top of the transform matrix is put the row of the Hadamard transform matrix with the smallest number of sign changes (from 1 to -1 and from -1 to 1) and below rows are ordered in increasing order of sign changes.

Hadamard and Walsh transforms are equivalent to each other but the Walsh transform order has analogy with the sinusoidal transform with respect to increasing of frequencies represented with corresponding coefficients.



Haar transform

Haar transform has ties with Multi Resolution Analysis.

Its basis functions are ortho normal wavelets.

Haar transform is separable and symmetric and can be expressed in the matrix form as

$$T = H F H^T$$

F is an $N \times N$ Image matrix, H is $N \times N$ transformation matrix, T is resulting $N \times N$ Transform.

The Haar transform matrix is real and orthogonal:

$$H = H^*, \quad H^{-1} = H^T, \quad \text{i.e. } H^T H = I$$

For Haar transform, H contains the Haar basis function defined over continuous closed interval $0 \leq z \leq 1$,

$Z \in [0,1]$ for $k = 0, 1, 2, 3, \dots, N-1$, where $N = 2^n$

The shape of Haar function, of an index k , is determined by two parameters: p and q ,

where $k = 2^p - 1$

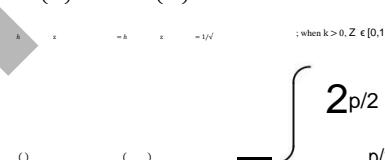
where $0 \leq p \leq n-1$, and

$q=0$ or 1 for $p=0$ and $1 \leq q \leq 2$

for $p \neq 0$

Then Haar basis functions are:

And



; when $k > 0, Z \in [0,1]$

$$\begin{cases} 2^{p/2} & (q-1)/2^p \leq z \leq (q-0.5)/2^p \\ -2^{p/2} & (q-0.5)/2^p \leq z \leq q/2^p \\ 0 & \text{otherwise, } Z \in [0,1] \end{cases}$$

From the above equation, one can see that p determines the amplitude and width of the non-zero part of the function, while q determines the position of the non-zero part of the Haar function

For $N=2$, the first row of 2×2 matrix H_2 is computed using $h(z) = 1/\sqrt{2}$. The first row of H_2 is computed using $h(z)$ which is equal to $1/\sqrt{2}$ for both elements. For $N=2$, the second row of 2×2 matrix H_2 is computed using $h(z)$ for $z=0/2, 1/2$.

$$\begin{array}{c}
 \text{for } z=0/2, \text{ will be } 1/\sqrt{2} \\
 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
 \end{array}
 \quad
 \begin{array}{c}
 \text{and } h(1/2) \\
 \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\sqrt{2} & -\sqrt{2} \end{pmatrix}
 \end{array}$$

KLT or HOTELLING TRANSFORM (PCA)

In **statistics**, principal components analysis (PCA) is a technique to simplify a dataset; more formally it is a transform used for reducing dimensionality in a dataset while retaining those characteristics of the dataset that contribute most to its variance. These characteristics may be the 'most important', but this is not necessarily the case, depending on the application.

PCA (principle component analysis) is also called the Karhunen-Loève transform or the Hotelling transform.

Many problems present themselves in terms of an eigen value problem:

$$\mathbf{A} \cdot \mathbf{v} = \lambda \cdot \mathbf{v}$$

In this equation \mathbf{A} is an n -by- n matrix, \mathbf{v} is a non-zero n -by-1 vector and λ is a scalar (which may be either real or complex). value of λ for which this equation has a solution is known as an eigenvalue of the matrix \mathbf{A} . It is sometimes also called the characteristic value. The vector, \mathbf{v} , which corresponds to this value is called an eigenvector

The KLT analyzes a set of vectors or images, into basis functions or images where the choice of the basis set depends on the statistics of the image set - depends on image co-variance matrix.

Consider a set of vectors (corresponding for instance to rows of an image)

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{pmatrix}$$

First form a mean vector of population $\underline{m}_x = E[\mathbf{x}]$, where $E[\cdot]$ is the 'expectation' or mean operator.

The **mean vector** of the population is defined as

$$\underline{m}_x = E\{\underline{x}\} \Rightarrow \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} E\{x_1\} \\ E\{x_2\} \\ \vdots \\ E\{x_n\} \end{bmatrix}$$

The **covariance matrix** of the population is defined as

$$\underline{C}_x = E\{(\underline{x} - \underline{m}_x)(\underline{x} - \underline{m}_x)^T\}$$

For M vectors from a random population, where M is large enough

$$\underline{m}_x = \frac{1}{M} \sum_{k=1}^M \underline{x}_k$$

We define the covariance matrix

$$Cx = E[(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T]$$

which equals the outer product of the vector $\mathbf{x} - \mathbf{m}_x$ with itself. Hence if \mathbf{x} is a length N vector, Cx is a $N \times N$ symmetric matrix such that $Cx^T = Cx$.

By finding the eigen values and eigenvectors of the covariance matrix, we find that the eigen vectors with the largest eigen values correspond to the dimensions that have the strongest correlation in the dataset. The original measurements are finally projected onto the reduced vector space.

Let A be a matrix whose rows are formed from the eigen vectors of C_x . First row of A is the eigen vector correspond to the largest eigen value and the last row of the eigen vector correspond to the smallest eigen value.

Suppose A is the transformation matrix that maps the vecors of x into vecots of y by using the following transformation

$$Y = A [x - mx]$$

The above transform is the called the Karhunen Loeve or Hotelling transform

Disadvantages:

KLT depends on the covariance of the data and therefore needs to be computed for every image

DIGITAL IMAGE PROCESSING

UNIT 2: IMAGE ENHANCEMENT - SPATIAL DOMAIN

Process an image so that the result will be more suitable than the original image for a specific application.

Highlighting interesting detail in images

Removing noise from images

Making images more visually appealing

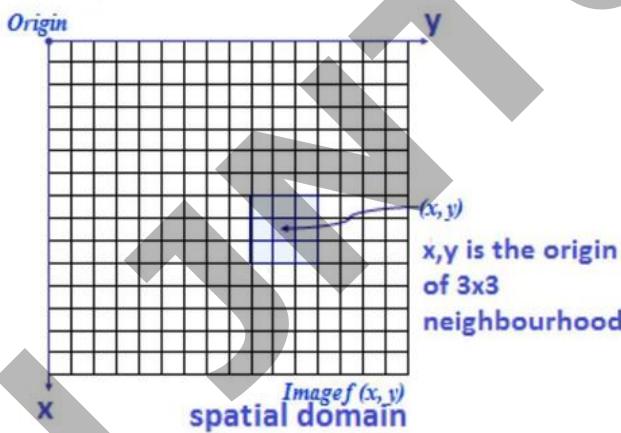
So, a technique for enhancement of x-ray image may not be the best for enhancement of microscopic images.

These spatial domain processes are expressed by:

$$G(x,y) = T(f(x,y)) \text{ depends only on the value of } f \text{ at } (x,y)$$

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

T is called a gray-level or intensity transformation operator which can apply to single image or binned images.



Window origin is moved from image origin along the 1st row and then second row etc.

At each location, the image pixel value is replaced by the value obtained after applying T operation on the window at the origin.

the neighborhood size may be different. We can have a neighborhood size of 5 by 5, 7 by 7 and so on depending upon the type of the image and the type of operation that we want to have.

Suppose we have a digital image which can be represented by a two dimensional random field $f(x, y)$.

An image processing operator in the spatial domain may be expressed as a mathematical function applied to the input image $f(x, y)$ to produce a new image $g(x, y) = Tf(x, y)$ as follows.

The operator T applied on $f(x, y)$ may be defined over some neighborhood of (x, y) .

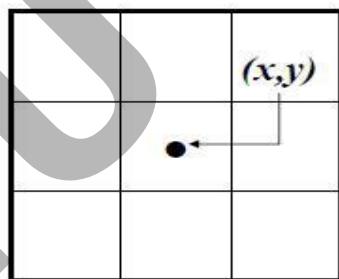
- (i) A single pixel (x, y) . In this case T is a gray level transformation (or mapping) function.
- (ii) Some neighborhood of (x, y) .
- (iii) T may operate to a set of input images instead of a single image.

the neighborhood of a point (x, y) is usually a square sub image which is centered at point (x, y) .

Mask/Filter

Neighborhood of a point (x, y) can be defined by using a square/rectangular (commonly used) or circular subimage area centered at (x, y) .

The center of the subimage is moved from pixel to pixel starting at the top-left corner.



Spatial Processing :

intensity transformation -> works on single pixel
for contrast manipulation

image thresholding

spatial filtering → Image sharpening (working on neighborhood of every pixel)
or Neighborhood Processing:

Spatial domain techniques

Point Processing:
Contrast stretching
Thresholding

Intensity transformations / gray level transformations

> Image Negatives
> Log Transformations
> Power Law Transformations
Piecewise-Linear Transformation Functions
Contrast stretching
Gray-level slicing
Bit-plane slicing
Mask processing / spatial filtering

Spatial filters

Smoothening filters Low pass filters
Median filters

Sharpening filters High boost filters

Derivative filters

Spatial Domain : (image plane)

Techniques are based on direct manipulation of pixels in an image

Frequency Domain :

Techniques are based on modifying the Fourier transform of an image

There are some enhancement techniques based on various combinations of methods from these two categories.

In spatial domain operations, the enhancement techniques work directly on the image pixels and then these spatial domain operations can have 3 different forms.

One is the point processing,
other one is the histogram based processing techniques and(histogram based processing technique is also a form of point processing technique.)
the third one is mask processing techniques.

Point Processing OR Gray level transformation

The smallest possible neighbourhood size is 1x1.

The simplest spatial domain operations occur when the neighbourhood is simply the pixel itself. In this case T is referred to as a gray level transformation function or a point processing operation

Point processing operations take the form $s = T(r)$

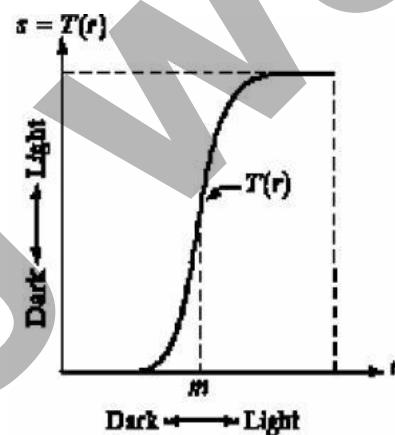
where s refers to the processed image pixel value (gray-level mapping function) and
r refers to the original image pixel value or gray level of the pixel
 T gray level (or intensity or mapping) transformation function

Contrast Stretching

Produce higher contrast than the original by

darkening the levels below m in the original image

Brightening the levels above m in the original image

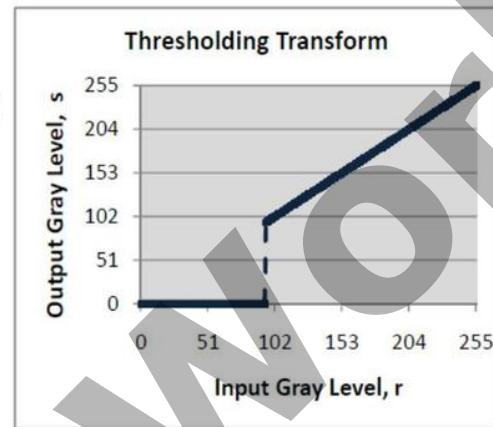


Thresholding (piece wise linear transformation)

Produce a two-level (binary) image

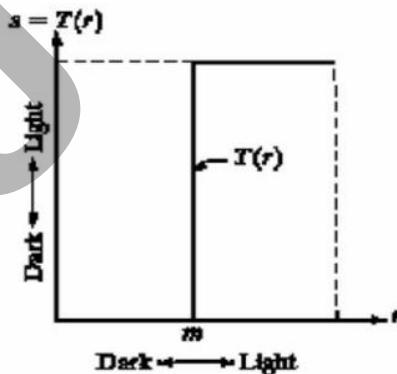
For any $0 < t < 255$ the threshold transform Thr_t can be defined as:

$$s = Thr_t(r) = \begin{cases} 0 & \text{if } r < t \\ r & \text{otherwise} \end{cases}$$



Thresholding has another form used to generate binary images from the gray-scale images, i.e.:

$$s = Thr_t(r) = \begin{cases} 0 & \text{if } r < t \\ 255 & \text{otherwise} \end{cases}$$

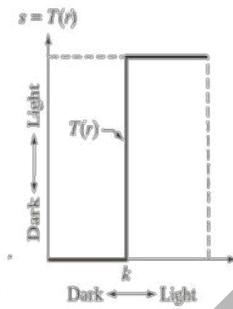
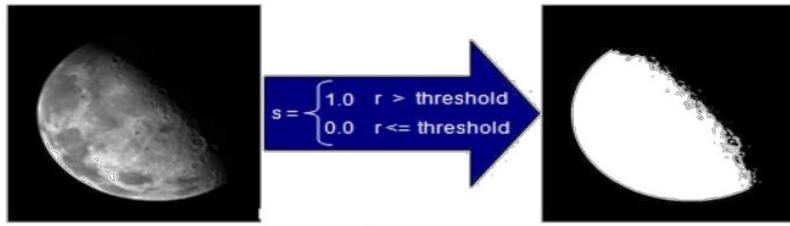


From display point of view, Image grey level values are usually in $[0, 255]$ range Where 0 is black and 255 is white. These are say L levels, where $L=256=2^8=2^k$ There is no reason why we have to use this range and Grey levels can also be assumed to be in the range $[0.0, 1.0]$ as binary.

This transformation would produce a higher contrast image than the original by darkening the intensity levels below m and brightening the levels above m

Values of r lower than m are darkened and more than m are brightened.
In case of thresholding Operation, $T(r)$ produces a two level image.

Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background



Intensity Transformations and Spatial Filtering

Basics

Use of spatial masks for filtering is called spatial filtering

- May be linear or nonlinear

• Linear filters

- Lowpass: Attenuates (or eliminate) high frequency components such as characterized by edges and sharp details in an image. Removes noise. Net effect is image blurring
- Highpass: Attenuate (or eliminate) low frequency components such as slowly varying characteristics
Net effect is a sharpening of edges and other details
- Bandpass: Attenuate (or eliminate) a given frequency range
Used primarily for image restoration (are of little interest for image enhancement)

Smoothing linear filters /averaging filters / low-pass filters (linear filter)
Uses Blurring
Noise reduction

Mask Processing or Filter

Neighborhood is bigger than 1x1 pixel

Use a function of the values of f in a predefined neighborhood of (x,y) to determine the value of g at (x,y)

The value of the mask coefficients determine the nature of the process

Used in techniques

Image Sharpening

Image Smoothing

Spatial filtering

- Generally involves operations over the entire image
- Operations take place involving pixels within a neighborhood of a point of interest
- Also involves a predefined operation called a spatial filter
- The spatial filter is also commonly referred to as:
 - Spatial mask
 - Kernel
 - Template
 - Window

Spatial domain: Image Enhancement

Three basic type of functions are used for image enhancement.
image enhancement point processing techniques:

- Linear (Negative image and Identity transformations)
- Logarithmic transformation (log and inverse log transformations)
- Power law transforms (nth power and nth root transformations)

Grey level slicing
Bit plane slicing

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

Intensity transformations (Gray level transformations)

Linear function
Negative and identity transformations
Logarithm function
Log and inverse-log transformation
Power-law function
nth power and nth root transformations

Image Negatives

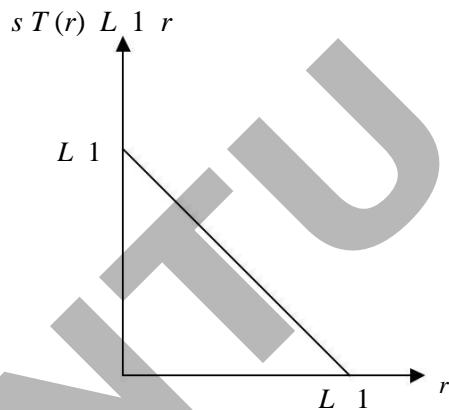
Here, we consider that the digital image that we are considering that will have capital L number of intensity levels represented from 0 to capital L minus 1 in steps of 1.

The negative of a digital image is obtained by the transformation function $s = T(r) = L - 1 - r$, where L is the number of grey levels.

The idea is that the intensity of the output image decreases as the intensity of the input increases.

This is useful in numerous applications such as displaying medical images.

So, we find that whenever r is equal to 0, then s will be equal to L minus 1 which is the maximum intensity value within our digital image and when r is equal to capital L minus 1 that is the maximum intensity value in the original image; in that case, s will be equal to 0. So, the maximum intensity value in the original image will be converted to the minimum intensity value in the processed image and the minimum intensity value in the processed image will be converted to maximum intensity value in the minimum intensity value in the original image will be converted to maximum intensity value in the processed image.



Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

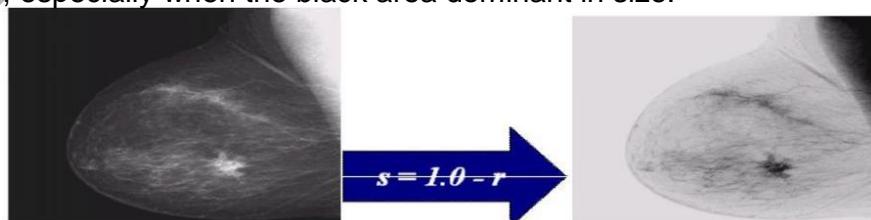
$$S = \text{intensity}_{\max} - r = s = L - 1 - r$$

An image with gray level in the range $[0, L-1]$ where $L = 2^n$; $n = 1, 2, \dots$

Negative transformation : $s = L - 1 - r$

Reversing the intensity levels of an image.

Suitable for enhancing white or gray detail embedded in dark regions of an image, especially when the black area dominant in size.



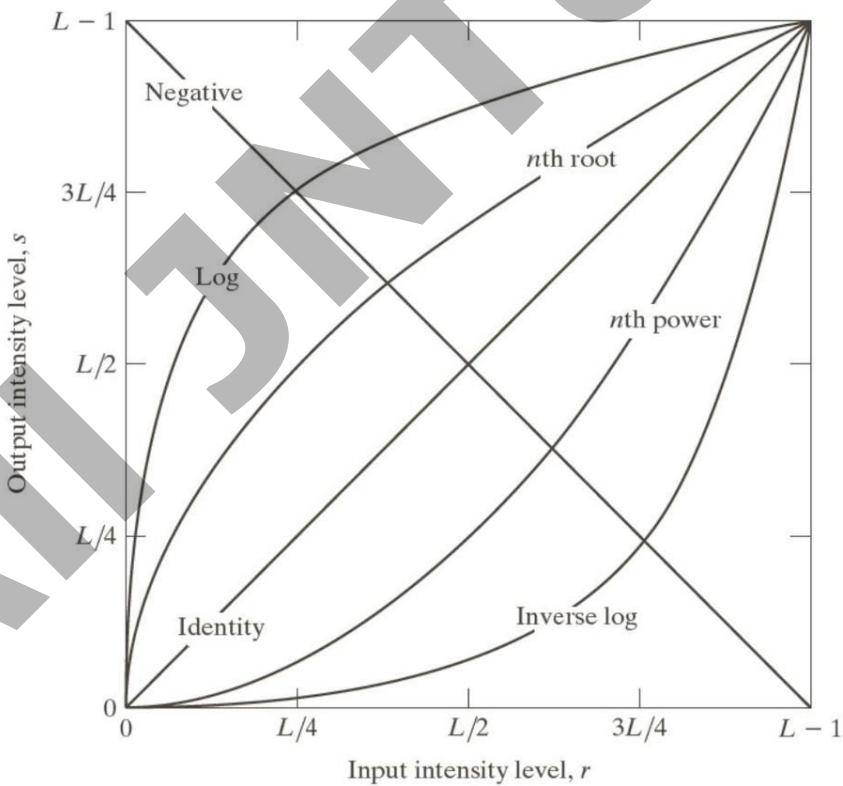
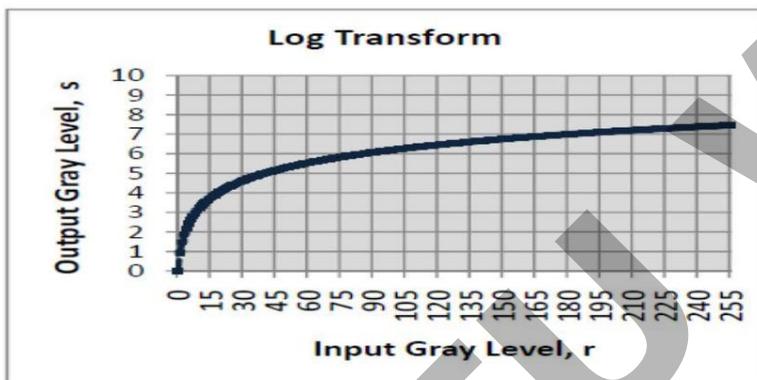
Logarithmic Transformations

The general form of the log transformation
is $s = c * \log (1 + r)$
 C is a constant and r is assumed to be ≥ 0

The log transformation maps a narrow range of low input grey level values into a wider range of output values. The inverse log transformation performs the opposite transformation

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

We usually set c to 1. Grey levels must be in the range [0.0, 1.0]



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

c is a constant and r ≥ 0

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

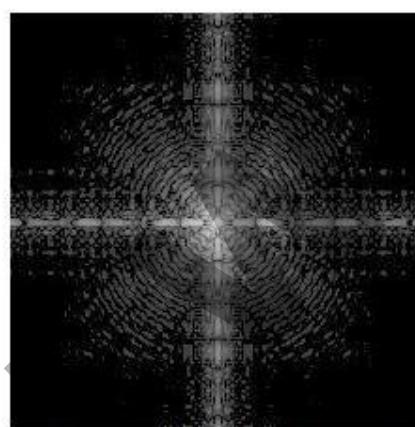
Used to expand the values of dark pixels in an image while compressing the higher level values.

It compresses the dynamic range of images with large variations in pixel values Example of image with dynamic range: Fourier spectrum image It can have intensity range from 0 to 10^6 or higher.

We can't see the significant degree of detail as it will be lost in the display.



(a) Original image



(b) Result of Log transform with $c = 1$

Inverse Logarithm Transformations

Do opposite to the Log Transformations

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

Identity Function

Output intensities are identical to input intensities.

Is included in the graph only for completeness

Power Law Transformations

Why power laws are popular?

A cathode ray tube (CRT), for example, converts a video signal to light in a nonlinear way. The light intensity is proportional to a power (γ) of the source voltage V_s

For a computer CRT, γ is about 2.2

Viewing images properly on monitors requires γ -correction

Power law transformations have the following

form $s = c * r^\gamma$

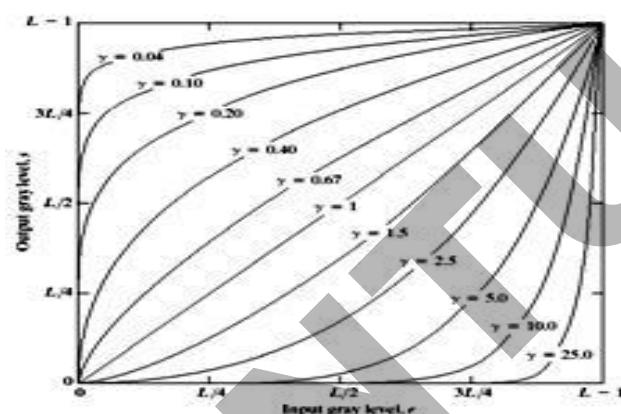
c and γ are positive constants

$$s = r^\gamma$$

We usually set c to 1. Grey levels must be in the range [0.0, 1.0]

Gamma correction is used for display improvements

Some times it is also written as $s = c(r + \epsilon)^\gamma$, and this offset is to provide a measurable output even when input values are zero



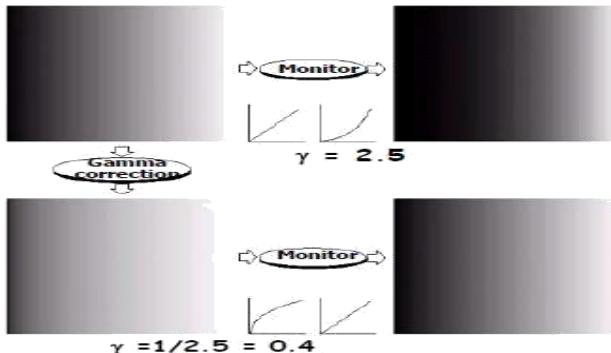
Plots of $s = cr^\gamma$ for various values of γ (c = 1 in all cases)

Varying γ gives a whole family of
curves c and γ are positive constants

$$s = cr^\gamma$$

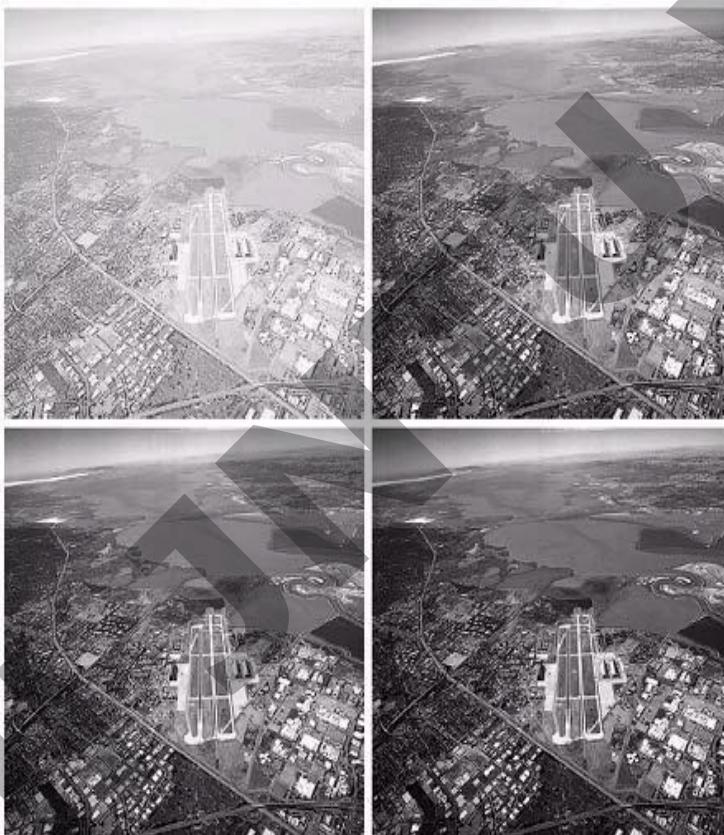
Power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels.

$c = \gamma = 1 \iff$ Identity function



Effect of decreasing gamma

When the γ is reduced too much, the image begins to reduce contrast to the point where the image started to have very slight “wash-out” look, especially in the background



- (a) image has a washed-out appearance, it needs a compression of gray levels
needs $\gamma > 1$
- (b) result after power-law transformation with $\gamma = 3.0$
(suitable)
- (c) transformation with $\gamma = 4.0$
(suitable)
- (d) transformation with $\gamma = 5.0$
(high contrast, the image has areas that are too dark, some detail is lost)

a	b
c	d

Piecewise Linear Transformation Functions

Piecewise functions can be arbitrarily complex

- A disadvantage is that their specification requires significant user input
- Example functions
 - Contrast stretching
 - Intensity-level slicing
 - Bit-plane slicing

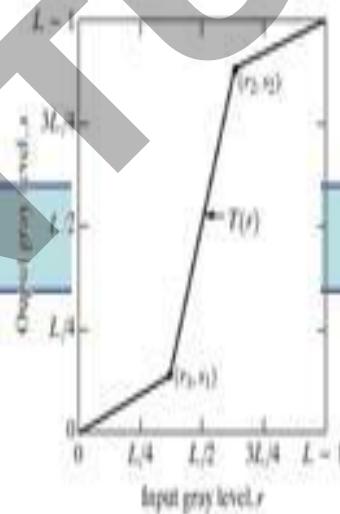
Contrast Stretching

Low contrast images occur often due to poor or non uniform lighting conditions, or due to nonlinearity, or small dynamic range of the imaging sensor.

Purpose of contrast stretching is to process such images so that the dynamic range of the image will be very high, so that different details in the objects present in the image will be clearly visible. Contrast stretching process expands dynamic range of intensity levels in an image so that it spans the full intensity range of the recording medium or display devices.



Low Contrast Image



Contrast stretched image

Control points (r_1, s_1) and (r_2, s_2) control the shape of the transform $T(r)$

- if $r_1=s_1$ and $r_2=s_2$, the transformation is linear and produce no changes in intensity levels

- $r_1=r_2$, $s_1=0$ and $s_2=L-1$ yields a thresholding function that creates a binary image
- Intermediate values of (r_1,s_1) and (r_2,s_2) produce various degrees of spread in the intensity levels

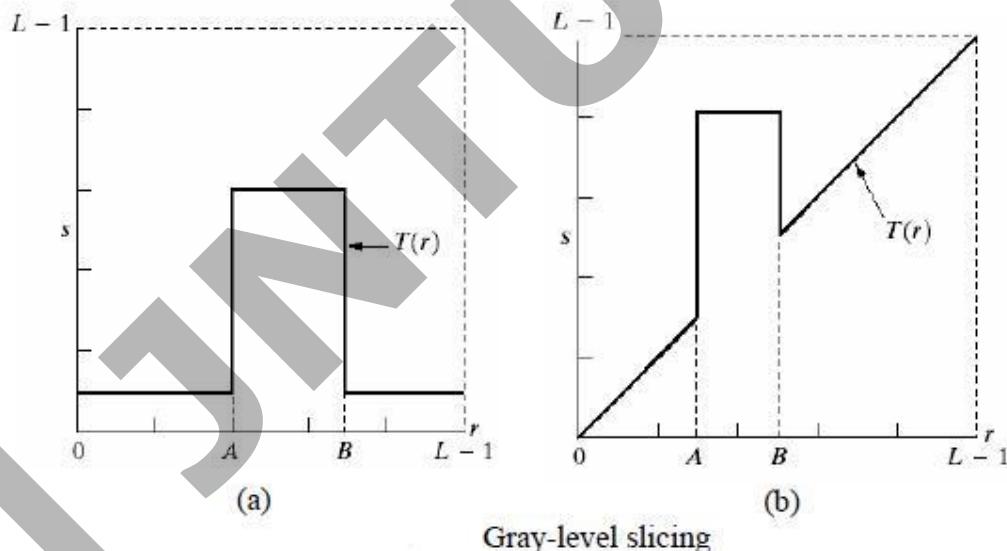
In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the junction is single valued and monotonically increasing.

If $(r_1,s_1)=(r_{\min},0)$ and $(r_2,s_2)=(r_{\max},L-1)$, where r_{\min} and r_{\max} are minimum and maximum levels in the image. The transformation stretches the levels linearly from their original range to the full range $(0,L-1)$

Gray Level (Intensity level) Slicing

Purpose: Highlight a specific range of gray values

But what difference we are going to have is the difference of intensity values that we have in the original image and the difference of intensity values we get in the process image. That is what gives us the enhancement



Highlighting a specific range of gray levels in an image

Display a high value of all gray levels in the range of interest and a low value for all other gray levels

(a) transformation highlights range $[A,B]$ of gray level and reduces all others to a constant level

(b) transformation highlights range $[A,B]$ but preserves all other levels

Used to highlight a specific range of intensities in an image that might be of interest

Two common approaches

- Set all pixel values within a range of interest to one value (white) and all others to another value (black)
 - Produces a binary image
That means, Display high value for range of interest, else low value ('discard background')
 - Brighten (or darken) pixel values in a range of interest and leave all others Unchanged. That means , Display high value for range of interest, else original value ('preserve background')

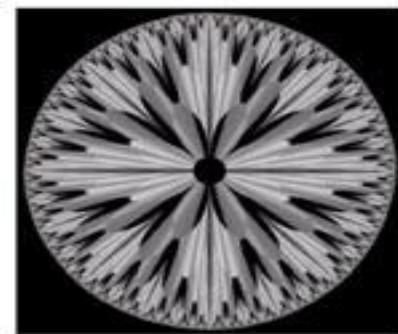
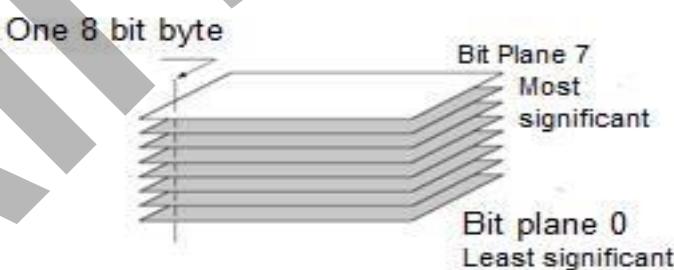
So, in such cases, for enhancement, what we can use is the gray level slicing operation and the transformation function is shown over here. Here, the transformation function on the left hand side says that for the intensity level in the range A to B, the image will be enhanced for all other intensity levels, the pixels will be suppressed.

On the right hand side, the transformation function shows that again within A and B, the image will be enhanced but outside this range, the original image will be retained and the results that we get is something like this.

Bit Plane Slicing

Only by isolating particular bits of the pixel values in a image we can highlight interesting aspects of that image.

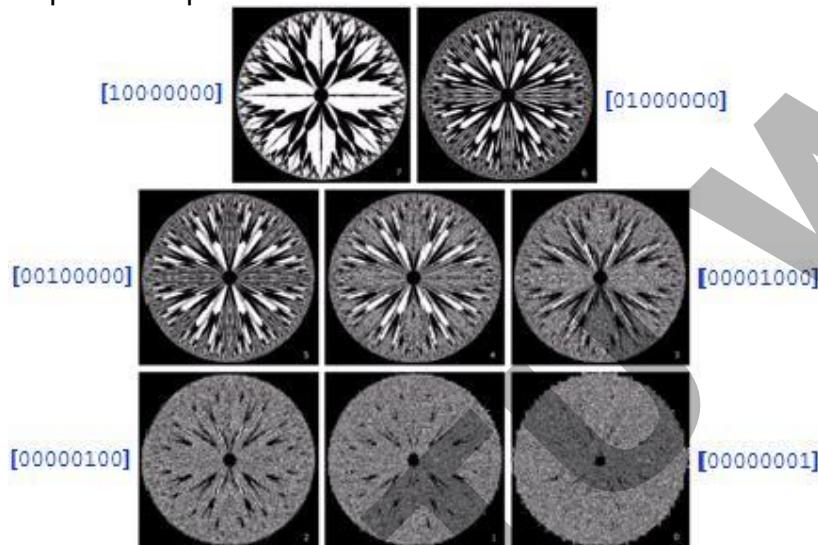
High order bits contain most of the significant visual information
Lower bits contain subtle details



Highlighting the contribution made to total image appearance by specific bits
 Suppose each pixel is represented by 8 bits
 Higher-order bits contain the majority of the visually significant data
 Useful for analyzing the relative importance played by each bit of the image

Pixels are digital values composed of bits

- For example, a pixel in a 256-level gray-scale image is comprised of 8 bits
- We can highlight the contribution made to total image appearance by specific bits
- For example, we can display an image that only shows the contribution of a specific bit plane



Useful for compression. Reconstruction is obtained by:

$$I(i, j) = \sum_{n=1}^N 2^{n-1} I_n(i, j)$$

0 to 127 can be mapped as 0, 128 to 256 can be mapped as 1

For an 8 bit image, the above forms a binary image. This occupies less storage space.

HISTOGRAM

Spreading out the histogram frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

Uses of Histograms: Image enhancements

- Image statistics
- Image compression
- Image segmentation

Histogram processing. Definition of the histogram of an image.

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

So, in this particular case, we will find that because we are considering the discrete images; so this function the histogram $h_k(r_k)$ will also be discrete $h_k(r_k) = n_k$. So here, r_k is a discrete intensity level, n_k is the number of pixels having intensity level r_k and $h_k(r_k)$ which is same as n_k also assumes discrete values.

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

NORMALISED HISTOGRAM

So instead of taking a simple histogram as just defined, we sometimes take a normalized histogram. So, a normalized histogram is very easily derived from this original histograms or the normalized histogram is represented as $p(r_k)$ is equal to n_k by n . ($n=N^2$)

So, as before this n_k is the number of pixels having intensity value r_k and n is the total number of pixels in the digital image. So, find that from this expression that $p_k(r_k)$ equal to n_k/N^2 , this $p_k(r_k)$ actually tells you that what is the probability of occurrence of a pixel having intensity value equal to r_k and such type of histograms give as we said; information, a global description of the appearance of an image.

In many cases, we talk about what is called a normalized histogram.

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

$$S_k \quad T(r_k)$$

$$p_r(r_k) = \frac{n_k}{N^2} \quad (N^2 = M \times N, \text{ where } M \text{ rows} = N \text{ columns})$$

L 1 ***n***
K 0 ***N***
where
 r_k is the k th grey level, $k = 0, 1, \dots, L-1$

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

S_k : processed intensity

N^2 is the total number of pixels in the image

One approach is image enhancement using image subtraction operation and the other approach is image enhancement using image averaging operation. So, first let us start discussion on histogram processing and before that let us see that what we mean by the histogram of an image.

Four basic image types and their corresponding histograms

- Dark
- Light
- Low contrast
- High contrast

Histograms commonly viewed in plots as

$h(r_k) = n_k$ versus r_k x axis is r_k and y axis is $h(r_k)$ or $p(r_k)$

$p(r_k) = n_k / MN$ versus r_k

Image Dynamic Range, Brightness and Control

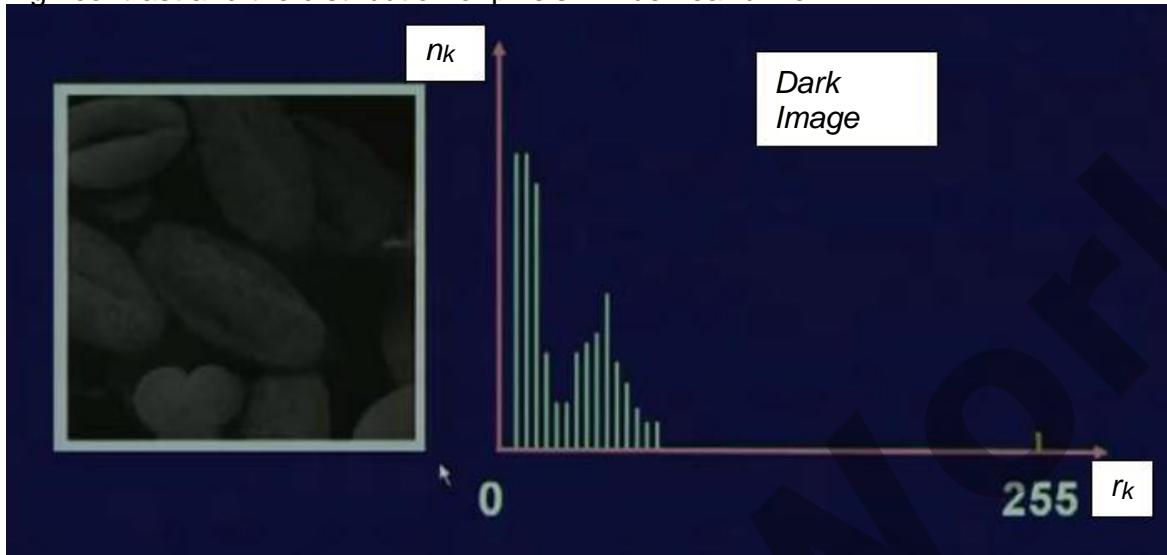
The dynamic range of an image is the exact subset of gray values (0, 1, 2, ..., L-1) that are present in the image. The image histogram gives a clear indication on its dynamic range.

When the dynamic range of the image is concentrated on the lower side of the gray scale, the image will be dark image.

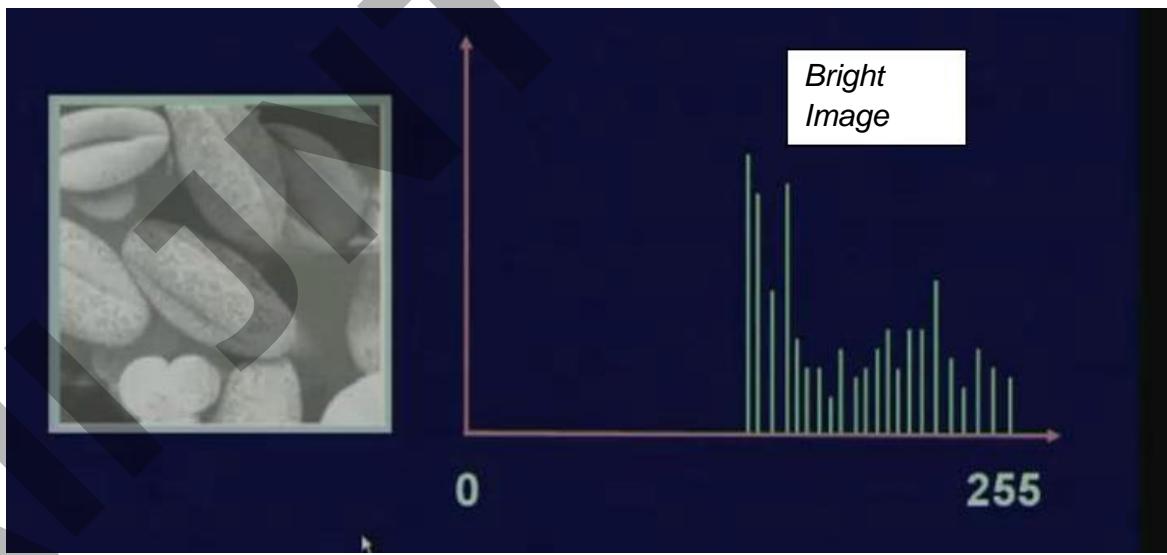
When the dynamic range of an image is biased towards the high side of the gray scale, the image will be bright or light image

An image with a low contrast has a dynamic range that will be narrow and concentrated to the middle of the gray scale. The images will have dull or washed out look.

When the dynamic range of the image is significantly broad, the image will have a high contrast and the distribution of pixels will be near uniform.

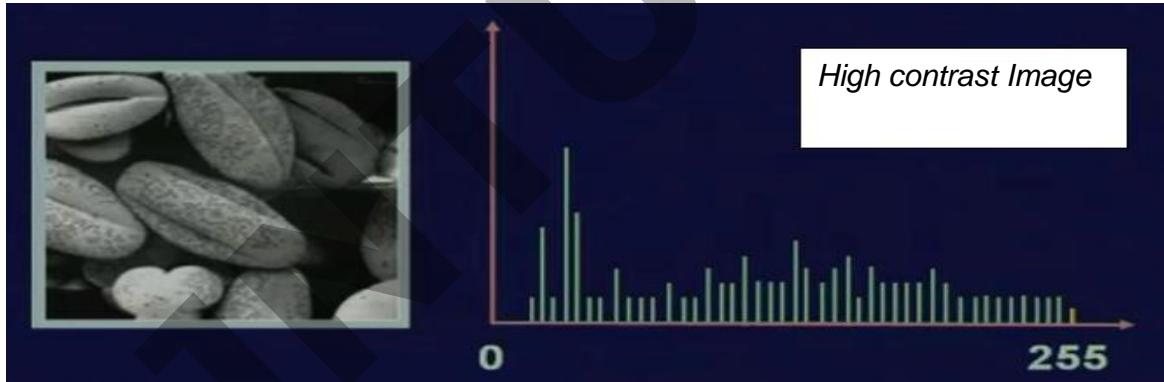
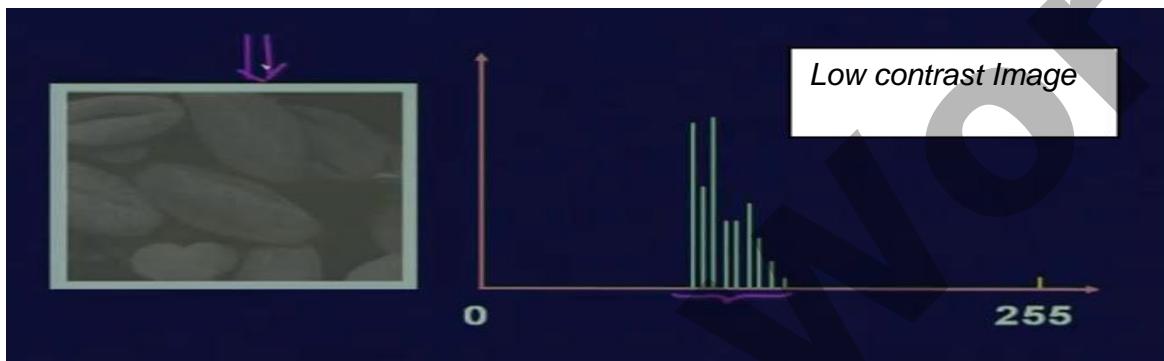


So, here we find that the first image as you see that it is a very very dark image. It is very difficult to find out what is the content of this particular image and if we plot the histogram of this particular image, then the histogram is plotted on the right hand side. You find that this plot says that most of the pixels of this particular image have intensity values which are near to 0.



Here, you find that this image is very bright and if you look at the histogram of this particular image; you find that for this image, the histogram shows that most of the pixels of this image have intensity values which are near to the maximum that is near value 255 and because of this the image becomes very bright.

Let us come to a third image category where the pixel values cover wide range of Intensity scale and close to uniformly distributed and has high dynamic range. This can be achieved through image transformations.



So, when we talk about this histogram based processing, most of the histogram based image enhancement techniques, they try to improve the contrast of the image; whether we talk about the histogram equalization or the histogram modification techniques

Now, when we talk about this histogram based techniques, this histogram based techniques; the histograms just give you a description a global description of the image. It does not tell you anything about the content of the image and that is quite obvious in these cases. Just by looking at the histogram, we cannot say that what is the content of the image.

Histogram Equalization or Image equalization or Histogram Linearization

let us see that how these histograms can be processed to enhance the images. So, the first one that we will talk about is the image equalization or histogram equalization operation.

Histogram equalization is a process for increasing the contrast in an image by spreading the histogram out to be approximately uniformly distributed

- The gray levels of an image that has been subjected to histogram equalization are spread out and always reach white
 - The increase of dynamic range produces an increase in contrast
- For images with low contrast, histogram equalization has the adverse effect of increasing visual graininess

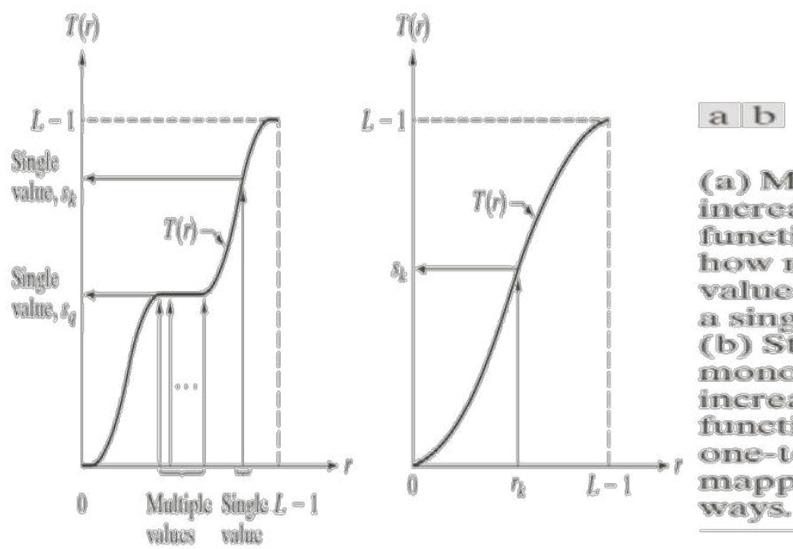
The intensity transformation function we are constructing is of the form

$$s = T(r), \quad 0 \leq r \leq L - 1, \quad r \text{ denotes intensity with } r=0 \text{ is black, } r=L-1 \text{ represents white.}$$

An output intensity level s is produced for every pixel in the input image having intensity r

- We assume
 - $T(r)$ is monotonically increasing in the interval $0 \leq r \leq L-1$
 - $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$
- If we define the inverse $r = T^{-1}(s) \quad 0 \leq s \leq L-1$

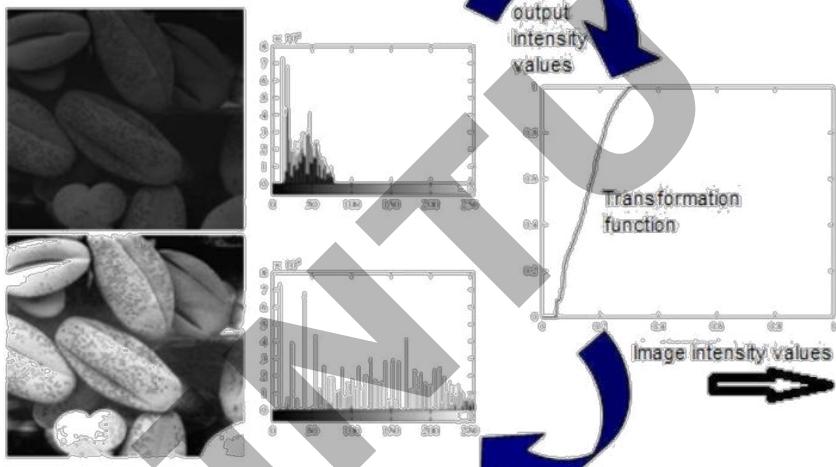
Then $T(r)$ should be strictly monotonically increasing, so that output will be in the same input range, and ensures output s will never be less than input r values.



a | b

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

Equalization Transformation Function



Histogram equalization

Linearisation requires construction of a transformation function s_k

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = (L - 1) \sum_{j=0}^k \frac{n_j}{M \times N}$$
$$= \frac{(L - 1)}{M \times N} \sum_{j=0}^k n_j \quad k = 0, 1, 2, \dots, L - 1$$

where r_k is the k th gray level, n_k is the number of pixels with that gray level, $M \times N$ is the number of pixels in the image, and $k = 0, 1, \dots, L - 1$

- This yields an s with as many elements as the original image's histogram (normally 256 for 8 bit images)
- The values of s will be in the range $[0, 1]$. For constructing a new image, s would be scaled to the range $[1, 256]$

Thus the processed output image is obtained by mapping each pixel in the input image with intensity r_k into a corresponding pixel with level s_k in the output image.

The transformation (mapping) $T(r_k)$ is called Histogram equalization or Histogram Linearization transformation

Discrete transformation function

Histogram equalization

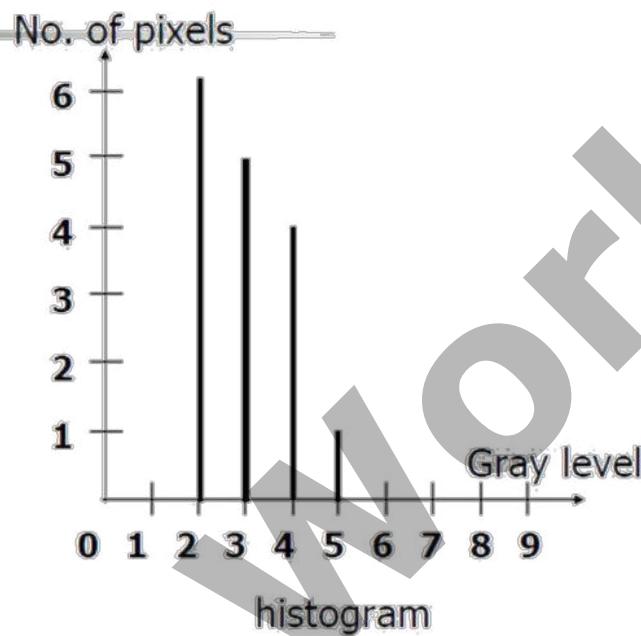
Thus, an output image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k in the output image

Example

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

4x4 image

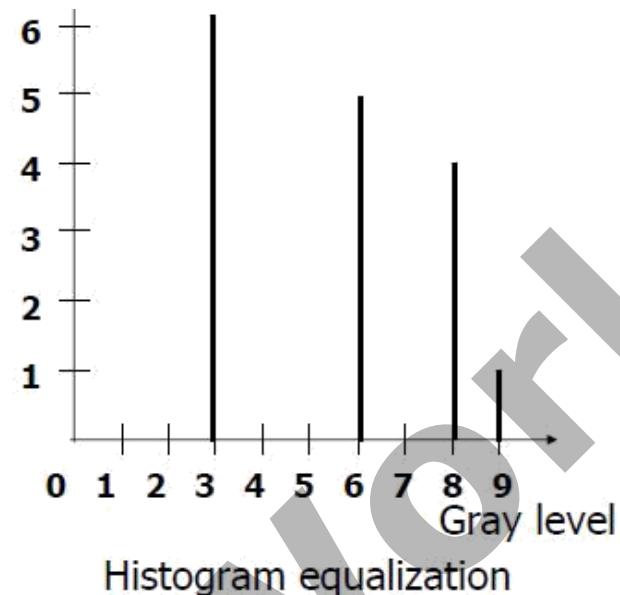
Gray scale = [0,9]



Gray Level(j)	0	1	2	3	4	5	6	7	8	9
No. of pixels	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^k n_j$	0	0	6	11	15	16	16	16	16	16
$s = \frac{\sum_{j=0}^k n_j}{n}$	0	0	6	11	15	16	16	16	16	16
$s \times 9$	0	0	3.3 ~3	6.1 ~6	8.4 ~8	9	9	9	9	9

3	6	6	3
8	3	8	6
6	3	6	9
3	8	3	8

Output image
Gray scale = [0,9]



It is clearly seen that

Histogram equalization distributes the gray level to reach the maximum gray level (white) because the cumulative distribution function equals 1 when $0 \leq r \leq L-1$

If the cumulative numbers of gray levels are slightly different, they will be mapped to little different or same gray levels as we may have to approximate the processed gray level of the output image to integer number. Thus the discrete transformation function can't guarantee the one to one mapping relationship.

Now, the first condition is very very important because it maintains the order of the gray levels in the processed image.

That is a pixel which is dark in the original image should remain darker in the processed image; a pixel which is brighter in the original image should remain brighter in the processed image.

So, the intensity ordering does not change in the processed image and that is guaranteed by the first condition that is $T(r)$ should be single valued and monotonically increasing in the range 0 to 1 of the values of r .

The second condition that is $0 \leq T(r) \leq 1$ is the one which ensures that the processed image that you get, that does not lead to a pixel value which is higher than the maximum intensity value that is allowed.

So, this ensures that the processed image will have pixel values which are always within the available minimum and maximum range and it can be found that if these conditions are satisfied by $T(r)$;

let us see that how we can have a discrete formulation of these derivations.

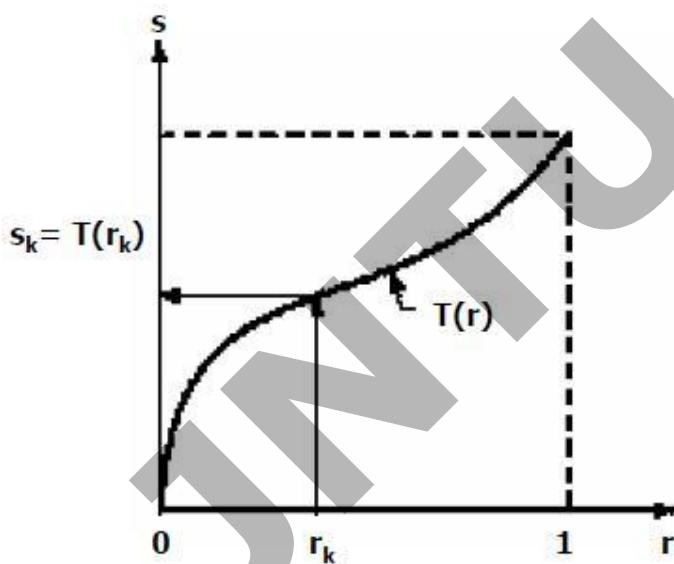
The intensity transformation function we are constructing is of the form

$S_k = T(r_k)$, $0 \leq r \leq L-1$, r denotes intensity with $r=0$ is black, $r=L-1$ represents white.

An output intensity level s is produced for every pixel in the input image having intensity r

- We assume
 - $T(r)$ is monotonically increasing in the interval $0 \leq r \leq L-1$
 - $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$
- If we define the inverse $r = T^{-1}(s)$ $0 \leq s \leq L-1$

Then $T(r)$ should be strictly monotonically increasing, so that output will be in the same input range, and ensures output s will never be less than input r values.



So, for discrete formulation, what we have seen earlier is that $p_r(r_k)$ is given by $\frac{n_k}{n}$ where n is the number of pixels having intensity value r and n is the total number of pixels in the image. And a plot of this $p_r(r_k)$ for all values of r_k gives us the histogram of the image.

So, the technique to obtain the histogram equalization and by that the image enhancement will be; first we have to find out the cumulative distribution function the CDF of r_k and so we will get s_k which is given by $T(r_k)$ and this $T(r_k)$ now is the cumulative distribution function which is p_i^r of say r where i will vary from 0 to k and this is nothing but sum of n_i by n where i will vary from 0 to k .

The inverse of this is obviously r_k is equal to $T^{-1}(s_k)$ for $0 \leq s_k \leq 1$

Enhancement Using Arithmetic/Logic Operations

Algebraic

- Addition
- Subtraction
- Multiplication
- Division

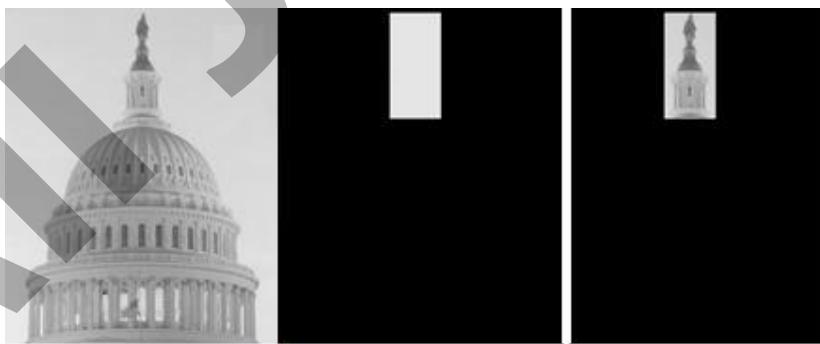
Logical

- AND
- OR
- NOT
- XOR

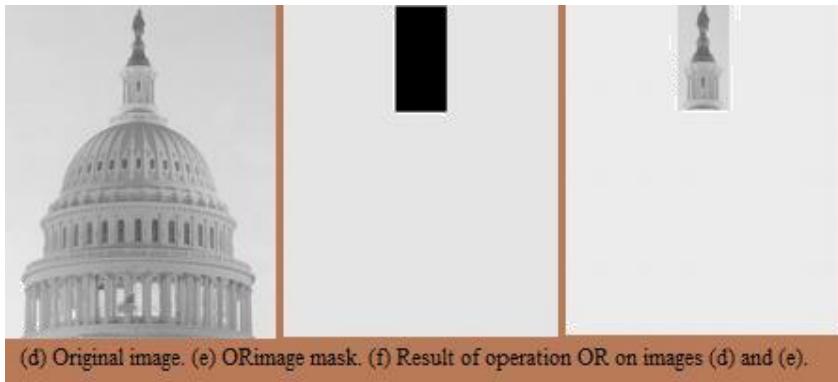
Arithmetic/logic operations involving images are performed on a pixel-by-pixel basis between two or more images

logic operation NOT, which is performed on a single image. We need only be concerned with the ability to implement the AND, OR, and NOT logic operators because these three operators are *functionally complete*. When dealing with logic operations on gray-scale images, pixel values are processed as strings of binary numbers.

Depending on the hardware and/or software being used, the actual mechanics of implementing arithmetic/logic operations can be done sequentially, one pixel at a time, or in parallel, where all operations are performed simultaneously



(a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a)&(b)



The AND and OR operations are used for *masking*; that is, for selecting subimages in an image as above.

Arithmetic operations

Addition, subtraction, multiplication and division

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

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

$$P(x, y) = f(x, y) \times g(x, y),$$

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

Images are to be of the same size. $X=0, 1, 2, \dots, M-1$, $y=0, 1, 2, \dots, N-1$

Addition:

Image averaging will reduce the noise. Images are to be registered before adding.

An important application of image averaging is in the field of astronomy, where imaging with very low light levels is routine, causing sensor noise frequently to render single images virtually useless for analysis

$$g(x, y) = f(x, y) + \eta(x, y)$$

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

As K increases, indicate that the variability (noise) of the pixel values at each location (x, y) decreases

In practice, the images $g_i(x, y)$ must be registered (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

Subtraction

A frequent application of image subtraction is in the enhancement of differences between images. Black (0 values) in difference image indicate the location where there is no difference between the images.

One of the most commercially successful and beneficial uses of image subtraction is in the area of medical imaging called *mask mode radiography*

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

Take an image. Make the Least Significant image as zero. The difference in these two images would be an enhanced image.

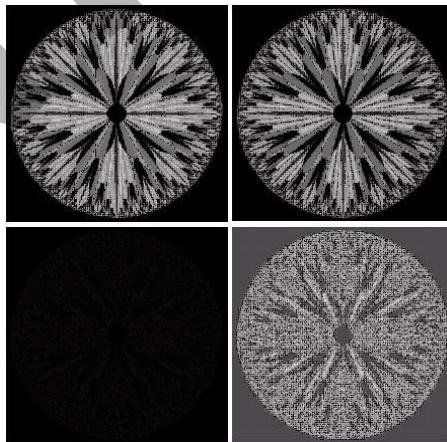
Image of a digital angiography. Live image and mask image with fluid injected. Difference will be useful to identify the blocked fine blood vessels.

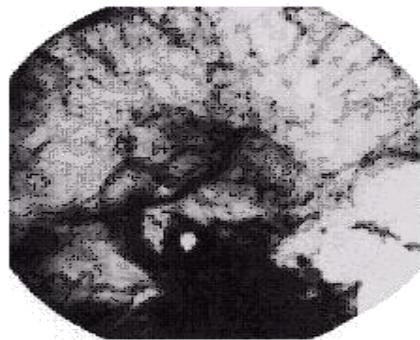
The difference of two 8 bit images can range from -255 to 255, and the sum of two images can range from 0 to 510.

Given and $f(x,y)$ image, $f_m = f - \min(f)$ which creates an image whose min value is zero.

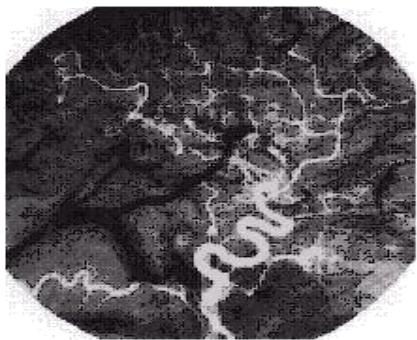
$fs = k [fm/\max(fm)]$, fs is a scaled image whose values of k are 0 to 255. For 8 bit image $k=255$,

- a). original fractal image
- b). result of setting the four lower-order bit planes to zero
refer to the bit-plane slicing
the higher planes contribute significant detail
the lower planes contribute more to fine detail
image b). is nearly identical visually to image a), with a very slight drop in overall contrast due to less variability of the gray-level values in the image.
- c). difference between a). and b).
(nearly black)
- d). histogram equalization of c).
(perform contrast stretching transformation)





mask image



an image (taken after injection of a contrast medium (iodine) into the bloodstream) with mask

An image multiplication and Division

An image multiplication and Division method is used in shading correction.

$$g(x, y) = f(x, y) \times h(x, y)$$

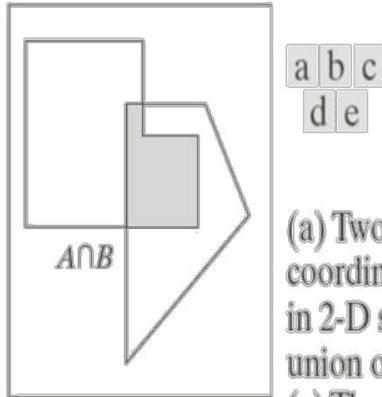
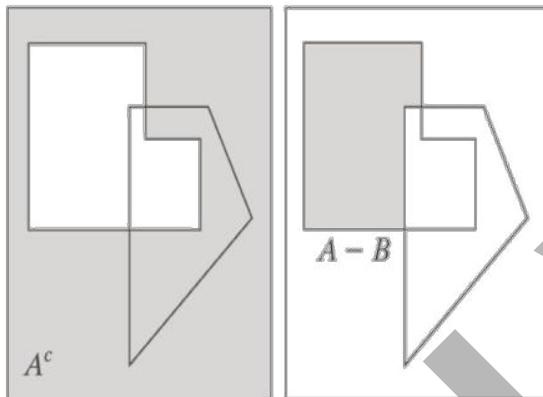
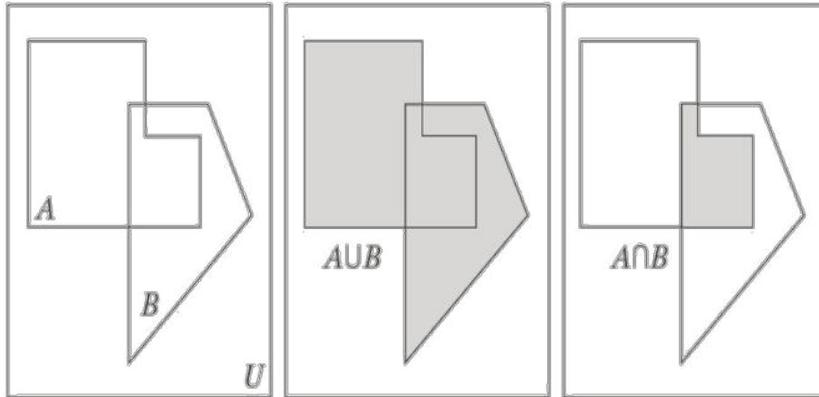
$g(x, y)$ is sensed image

$f(x, y)$ is perfect image

$h(x, y)$ is shading function.

If $h(x,y)$ is known, the sensed image can be multiplied with inverse of $h(x,y)$ to get $f(x,y)$ that is dividing $g(x,y)$ by $h(x,y)$

Another use of multiplication is Region Of Interest (ROI). Multiplication of a given image by mask image that has 1s in the ROI and 0s elsewhere. There can be more than one ROI in the mask image.



a	b	c
d	e	

- (a) Two sets of coordinates, A and B , in 2-D space. (b) The union of A and B .
(c) The intersection of A and B . (d) The complement of A .
(e) The difference between A and B . In (b)–(e) the shaded areas represent the member of the set operation indicated.

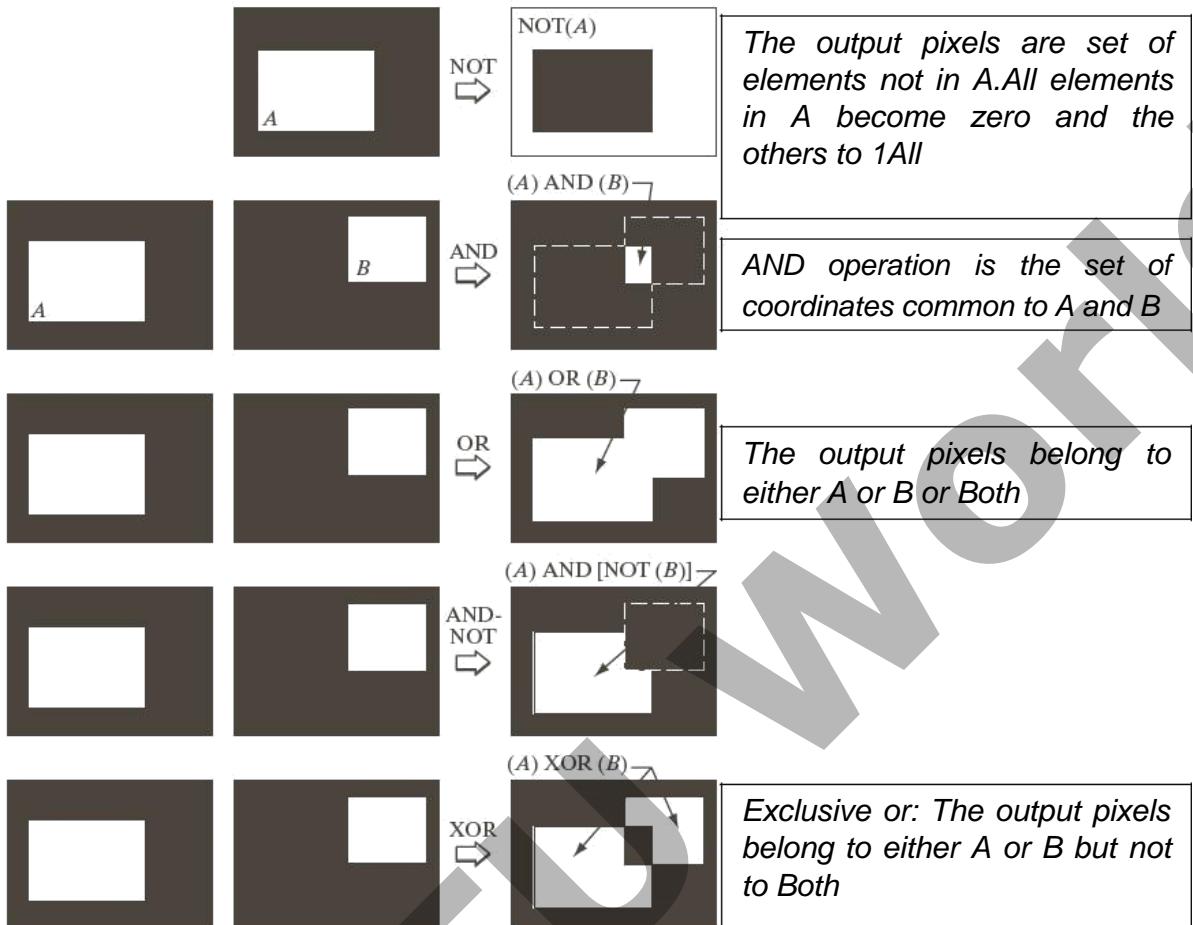


Image averaging

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

For the noise process $n(x, y)$ the following assumptions are made.

- (i) The noise process $n(x, y)$ is ergodic.
- (ii) It is zero mean, i.e., $E[n(x, y)] = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} n(x, y) = 0$
- (iii) It is white, i.e., the autocorrelation function of the noise process defined as $R[k, l] = E\{n(x, y)n(x+k, y+l)\} = \frac{1}{(M-k)(N-l)} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} n(x, y)n(x+k, y+l)$ is zero, apart for the pair $[k, l] = [0, 0]$.

Let $g(x, y)$ denote a corrupted image by adding noise $\eta(x, y)$ to a noiseless image $f(x, y)$:

$$g(x, y) \underset{(x, y)}{f(x, y)}$$

All JNTU World

The noise has zero mean value

At every pair of coordinates $E[z_i] = 0$

$z_i = (x_i, y_i)$ the noise is uncorrelated

$$E[z_i z_j] = 0$$

The noise effect is reduced by averaging a set of K noisy images. The new image is

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

The intensities at each pixel of the new image may be viewed as random variables. The mean value and the standard deviation of the new image show that the effect of noise is reduced

Spatial filters	Sharpening filters	Smoothing linear filters	High boost /High frequency emphasis filter /Unsharp masking	Order static filter / (non-linear filter) / median filter
Smoothening filters Low pass filters Median filters	High boost filters Derivative filters	averaging filters low-pass filters (linear filter) Uses Blurring Noise reduction	Principle: Subtract an unsharp image from the original image	Principle function: Force points with distinct intensity levels to be more like their neighbours Objective: Replace the value of the pixel by median of the intensity values in the neighborhood of that pixel
	Sharpening spatial filters Objective: High light fine details in an image Applications: Electronic printing Medical imaging Industrial		Uses: Printing industry Publishing industry Process steps: •1. blur the original image •2. subtract the blurred image from	Uses: Noise reduction Less blurring Reduce impulse noise

	inspection Autonomous target detection		the original (the difference is the mask) •3. add the weighted mask to the original	

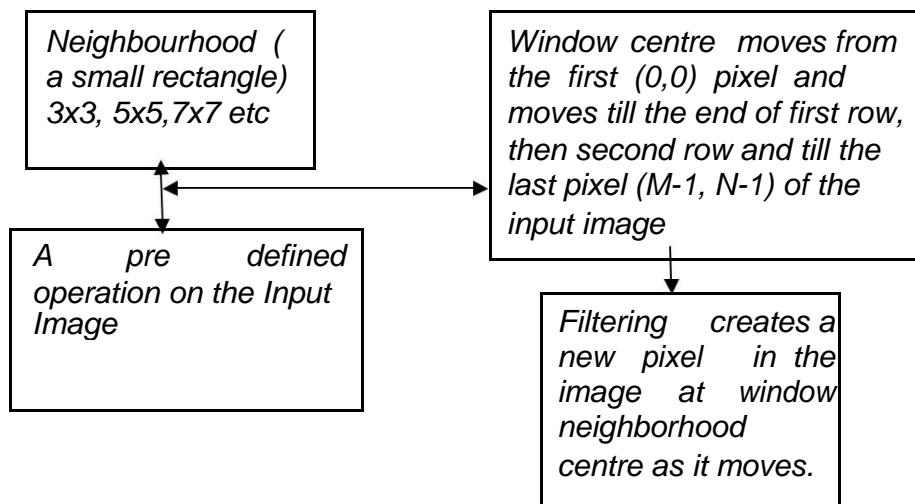
Spatial filtering

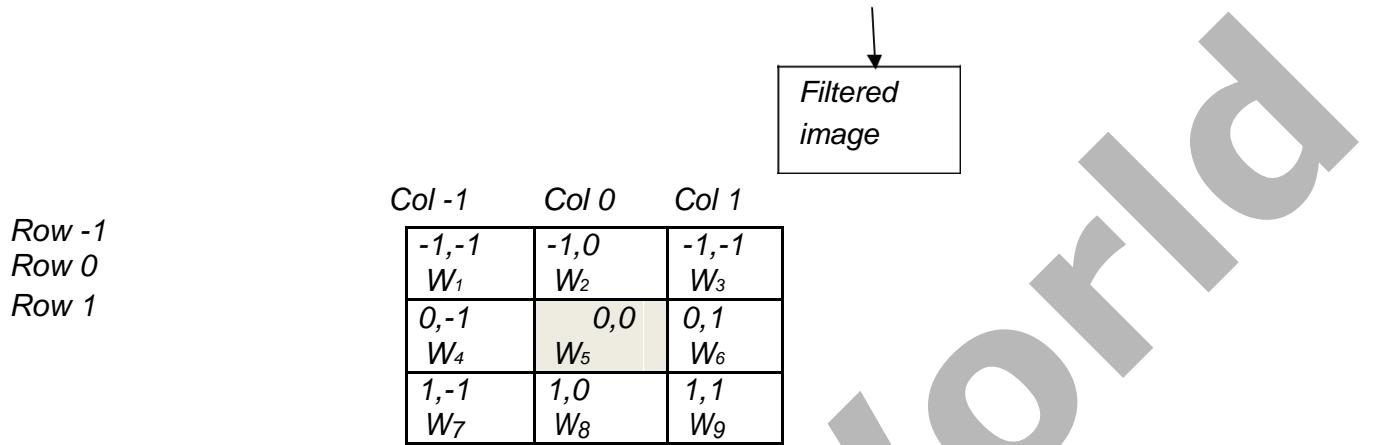
- Generally involves operations over the entire image
- Operations take place involving pixels within a neighborhood of a point of interest
- Also involves a predefined operation called a spatial filter
- The spatial filter is also commonly referred to as:
 - Spatial mask
 - Kernel
 - Template
 - Window

Spatial filters : spatial masks, kernels, templates, windows

Linear Filters and Non linear filters based on the operation performed on the image. Filtering means accepting (passing) or rejecting some frequencies.
Mechanics of spatial filtering

$$f(x, y) \rightarrow \text{Filter} \rightarrow g(x, y)$$

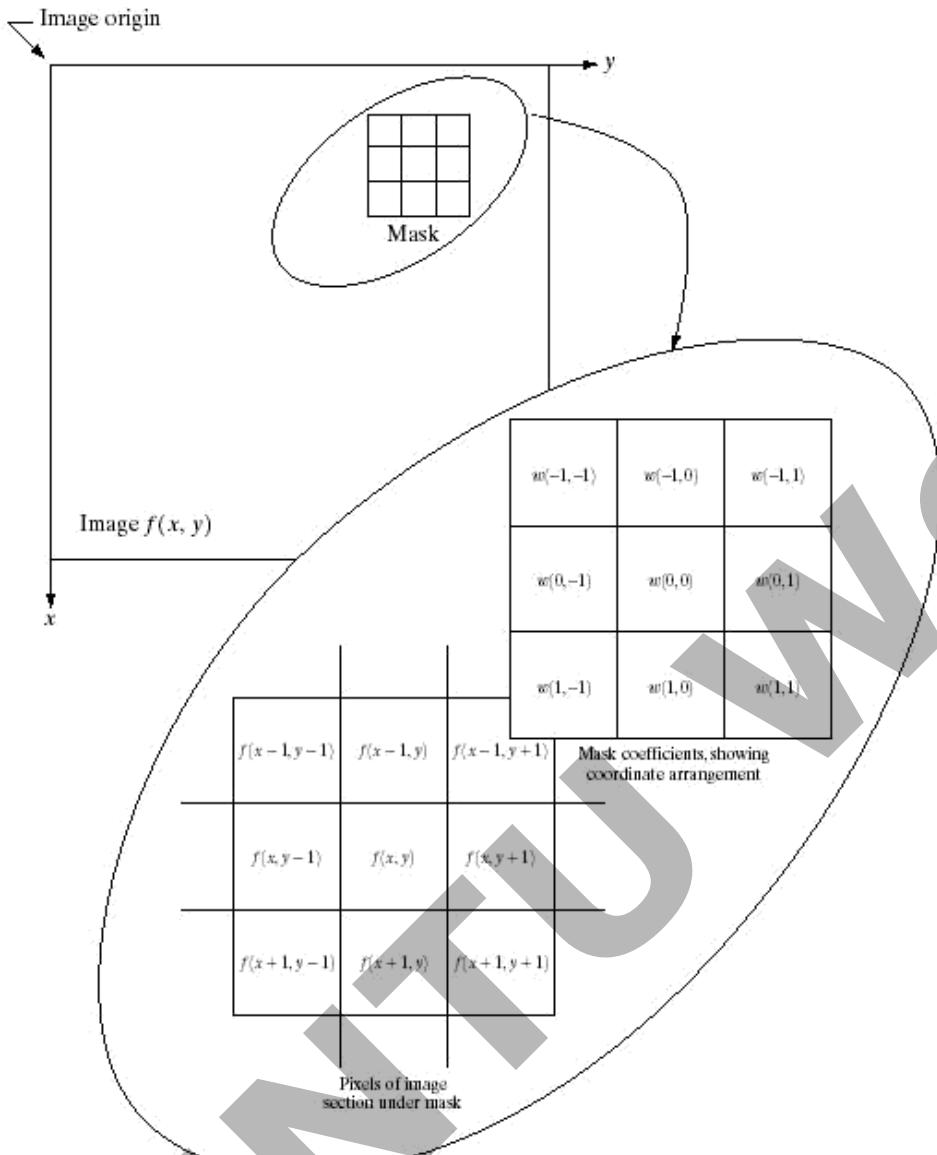




At any point (x,y) in the image, the response $g(x,y)$ of the filter is the sum of products of the filter coefficients and the image response and the image pixels encompassed by the filter.

Observe that the filter $w(0,0)$ aligns with the pixel at location (x,y)

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



The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

*Vector representation of a linear filtering
Characteristics response R of a mask, either for convolution or correlation is*

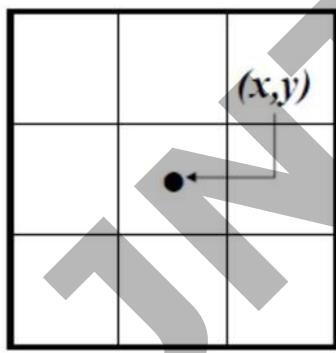
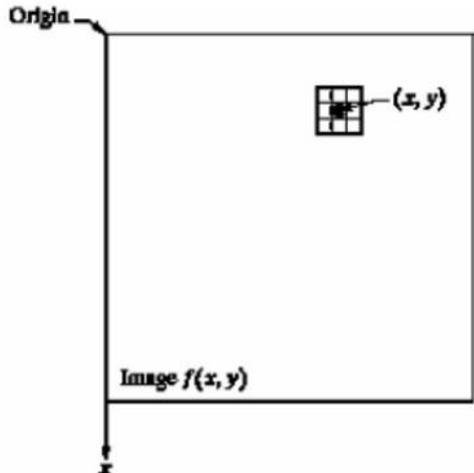
$$R = w_1 z_1 + w_2 z_2 + w_3 z_3 + \dots + w_{mn} z_{mn}$$

$$\begin{aligned} & mn \\ & = \sum w_k z_k \quad = w^T Z \end{aligned}$$

$k=1$

Ws are coefficients of an $m \times n$ filter and zs are corresponding image intensities encompassed by the filter.

Use the filter as it is for correlation and rotate by 180 degrees for convolution.



- Procedures that operate directly on pixels.

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

where

- $f(x,y)$ is the input image
- $g(x,y)$ is the processed image
- T is an operator on f defined over some neighborhood of (x,y)

- Neighborhood of a point (x,y) can be defined by using a square/rectangular (common used) or circular subimage area centered at (x,y)
- The center of the subimage is moved from pixel to pixel starting at the top of the corner

simply move the filter mask from point to point in an image.
at each point (x,y) , the response of the filter at that point is calculated using a predefined relationship

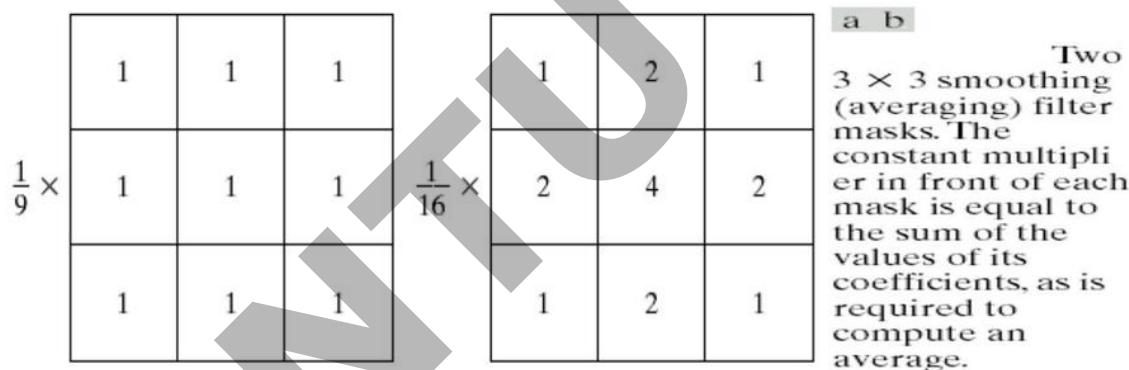
<i>Smoothening (blurring) filter</i> <i>Low Pass filter</i> <i>integration</i>	<i>Ideal LF</i> <i>Butterworth LF</i> <i>Gaussian LF</i>	<i>Noise reduction by removing sharp edges and Sharp intensity transitions</i> <i>Side effect is: This will blur sharp edges</i>	<i>Average: Average Filter</i> $R=1/9 [\sum z_i]$, <i>i=1 to 9</i> <i>Box filter (if all coefficients are equal)</i> <i>Weighted Average: Mask will have different coefficients</i>	<i>Linear Filter</i>
<i>Order statistic</i>		<i>Salt and pepper noise or impulse noise removal</i>	<i>1. Median filter 50 percentile</i>	<i>Non linear filter</i>
<i>Order statistic</i>		<i>Max filter finds bright objects</i>	<i>2. Max filter (100 percentile) t 3. Min filter (zero percentile)</i>	
<i>Sharpening filters</i> <i>differentiation</i>	<i>Highlights sharpening intensity</i>	<i>Image sharpening</i> <i>Second derivative filter is better for edge detection</i>	<i>Differentiation or first order or gradient</i> <i>Second derivative (Laplacian filter)</i>	<i>gradient is Linear operator</i> <i>magnitude is Non linear</i>
		<i>Image sharpening</i>	<i>Second derivative is Laplacian</i>	<i>Linear</i>
<i>Unsharp masking</i> <i>High Boost filtering</i>		<i>image sharpening</i>		
<i>First order derivatives for image</i>		<i>image sharpening</i>		<i>Non Linear</i>

sharpening				
------------	--	--	--	--

Smoothing Linear Filter or averaging filters or Low pass filters

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. they also are referred to as *lowpass filters*.

The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced “sharp” transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable side effect that they blur edges



Weighted Average mask: Central pixel usually have higher value. Weightage is inversely proportional to the distance of the pixel from centre of the mask.

The general implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$ (m and n odd) is given by the expression, $m=2a+1$ and $n=2b+1$, where a and b are nonnegative integers.

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

an important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest, such that the intensity of smaller objects blends with the background; after filtering and thresholding.

Smoothing Spatial Filter

Weighted average

1	2	1
2	4	2
1	2	1

$\frac{1}{16} \times$

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

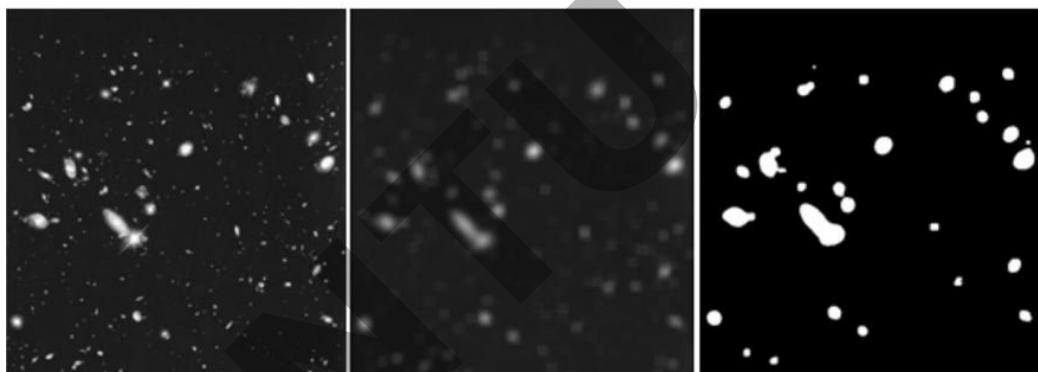


Image from Hubble telescope, image processed by 5x5 averaging window, and image after thresholding (nasa)

Examples of Low Pass Masks (Local Averaging)

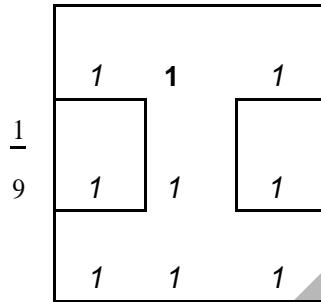
$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$\frac{1}{25} \times$	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1

Popular techniques for lowpass spatial filtering

Uniform filtering

The most popular masks for low pass filtering are masks with all their coefficients positive and equal to each other as for example the mask shown below. Moreover, they sum up to 1 in order to maintain the mean of the image.

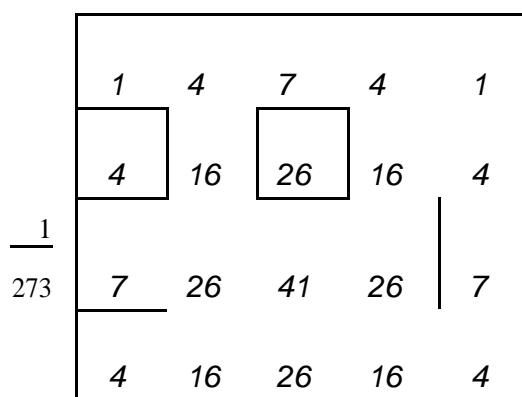


Gaussian filtering

The two dimensional Gaussian mask has values that attempts to approximate the continuous function

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

In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. The following shows a suitable integer-valued convolution kernel that approximates a Gaussian with a of 1.0.



1	4	7	4	1
---	---	---	---	---

All JNTU world

Median filtering

Order-Statistics (non linear)Filters

The best-known example in this category is the *Median filter*, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median).

Order static filter / (non-linear filter) / median filter Objective: Replace the value of the pixel by the median of the intensity values in the neighbourhood of that pixel

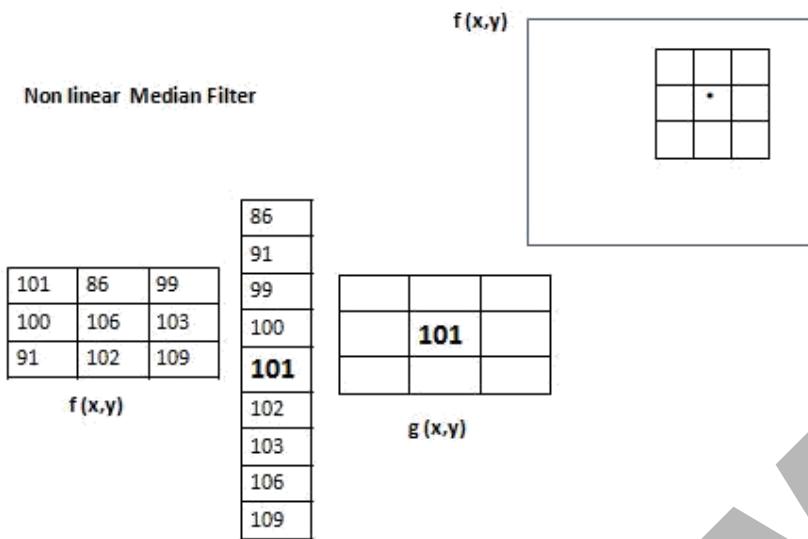
Although the median filter is by far the most useful order-statistics filter in image processing, it is by no means the only one. The median represents the 50th percentile of a ranked set of numbers, but the reader will recall from basic statistics that ranking lends itself to many other possibilities. For example, using the 100th percentile results in the so-called *max filter*, which is useful in finding the brightest points in an image. The response of a 3*3 max filter is given by $R=\max [z_k | k=1, 2, \dots, 9]$

The 0th percentile filter is the *min filter*, used for the opposite purpose. Example nonlinear spatial filters

- Median filter: Computes the median gray-level value of the neighborhood. Used for noise reduction.
- Max filter: Used to find the brightest points in an image
- Min filter: Used to find the dimmest points in an image

$$R = \max\{z | k = 1, 2, \dots, 9\}$$

$$R = \min\{z | k = 1, 2, \dots, 9\}$$



Directional smoothing or directional averaging filter

To protect the edges from blurring while smoothing, a directional averaging filter can be useful. Spatial averages $g(x, y :)$ are calculated in several selected directions (for example could be horizontal, vertical, main diagonals)

$$g(x,y:) = \frac{1}{N} \sum_{(k,l) \in W} f(x+k, y+l)$$

and a direction is found such that $f(x, y) - g(x, y :)$ is minimum. (Note that w is the neighbourhood along the direction and N is the number of pixels within this neighbourhood).

Then by replacing $g(x, y :)$ with $g(x, y :)$ we get the desired result.

Sharpening Spatial Filters

In the last section, we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood.

Since averaging is analogous to integration, it is logical to conclude that sharpening could be accomplished by spatial differentiation. This, in fact, is the case, and the discussion in this section deals with various ways of defining and implementing operators for sharpening by digital differentiation.

Fundamentally, the strength of the response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied. Thus, image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values.

first derivative (1) must be zero in flat segments (areas of constant gray-level values); (2) must be nonzero at the onset of a gray-level step or ramp; and (3) must be nonzero along ramps. Similarly, any definition of a second derivative (1) must be zero in flat areas; (2) must be nonzero at the onset and end of a gray-level step or ramp; and (3) must be zero along ramps of constant slope.

A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference

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

Similarly, we define a second-order derivative as the difference

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

Let us consider the properties of the first and second derivatives as we traverse the profile from left to right. First, we note that the first-order derivative is nonzero along the entire ramp, while the second-order derivative is nonzero only at the onset and end of the ramp. Because edges in an image resemble this type of transition, we conclude that first-order derivatives produce “thick” edges and second-order derivatives, much finer ones

a second-order derivative to enhance fine detail (including noise) much more than a first-order derivative.

- (1) First-order derivatives generally produce thicker edges in an image.
- (2) Second-order derivatives have a stronger response to fine detail, such as thin lines and isolated points.
- (3) First-order derivatives generally have a stronger response to a gray-level step.
- (4) Second-order derivatives produce a double response at step changes in gray level.

Use of Second Derivatives for Enhancement–The Laplacian

We are interested in *isotropic* filters, whose response is independent of the direction of the discontinuities in the image to which the filter is applied. In other words, isotropic filters are *rotation invariant*, in the sense that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.

Development of the method

It can be shown (Rosenfeld and Kak [1982]) that the simplest isotropic derivative operator is the *Laplacian*, which, for a function (image) $f(x, y)$ of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

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

In order to be useful for digital image processing, this equation needs to be expressed in discrete form.

we use the following notation for the partial second-order derivative in the x -direction:

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

and, similarly in the y -direction, as

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

The digital implementation of the two-dimensional Laplacian is obtained by summing these two components:

$$\begin{aligned}\nabla^2 f = & [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] \\ & - 4f(x, y).\end{aligned}$$

This equation can be implemented using the mask gives an isotropic result for rotations in increments of 90°. . The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms , one for each of the two diagonal directions.

Since each diagonal term

also contains a $-2f(x, y)$ term, the total subtracted from the difference terms now would be $-8f(x, y)$

Laplacian operator

The **Laplacian** of a 2-D function $f(x, y)$ is a second order derivative defined as

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

In practice it can be also implemented using a 3x3 mask as follows (**why?**)

$$\frac{\partial^2 f}{\partial x^2} = z_5 - (z_2 + z_4 + z_6 + z_8)$$

The main disadvantage of the Laplacian operator is that it produces double edges (**why?**).

Because the Laplacian is a derivative operator, its use highlights gray-level discontinuities in an image and deemphasizes regions with slowly varying gray levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian operation simply by adding the original and Laplacian images. As noted in the previous paragraph, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we *subtract*, rather than add, the Laplacian image to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image enhancement is as follows:

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

(a) Filter mask used to implement the digital Laplacian,

- (b) Mask used to implement an extension that includes the diagonal neighbors. (c) and
 (d) Two other implementations
 of the Laplacian.

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

Unsharp masking and high-boost filtering

A process used for many years in the publishing industry to sharpen images consists of subtracting a blurred version of an image from the image itself. This process, called *unsharp masking*, is expressed as

$$fs(x, y) = f(x, y) - b(x, y)$$

where $fs(x, y)$ denotes the sharpened image obtained by unsharp masking, and $b(x, y)$ is a blurred version of $f(x, y)$.

A slight further generalization of unsharp masking is called *high-boost filtering*. A high-boost filtered image, f_{hb} , is defined at any point (x, y) as

$$f_{hb}(x, y) = Af(x, y) - b(x, y)$$

where $A \geq 1$ and, as before, b is a blurred version of f . This equation may be written as

$$f_{hb}(x, y) = (A - 1)f(x, y) + f(x, y) - b(x, y).$$

$$f_{hb}(x, y) = (A - 1)f(x, y) + fs(x, y)$$

as the expression for computing a high-boost-filtered image.

One of the principal applications of boost filtering is when the input image is darker than desired. By varying the boost coefficient, it generally is possible to obtain an overall increase in average gray level of the image, thus helping to brighten the final result

High Boost Filtering

A high pass filtered image may be computed as the difference between the original image and a lowpass filtered version of that image as follows:

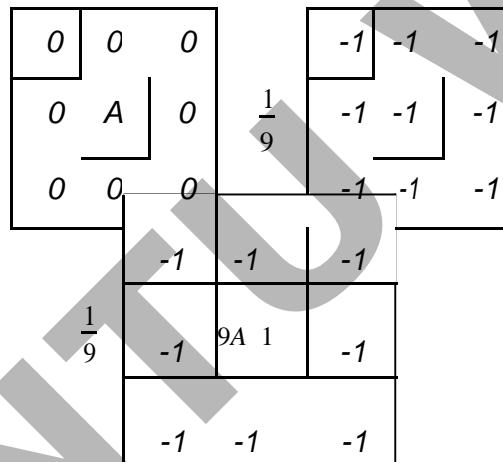
$$(\text{Highpass part of image}) = (\text{Original}) - (\text{Lowpass part of image})$$

Multiplying the original image by an amplification factor denoted by A , yields the so called high boost filter:

$$(\text{Highboost image}) = (A) (\text{Original}) - (\text{Lowpass}) = (A^{-1}) (\text{Original}) + (\text{Original}) -$$

$$(\text{Lowpass}) = (A^{-1}) (\text{Original}) + (\text{Highpass})$$

The general process of subtracting a blurred image from an original as given in the first line is called **unsharp masking**. A possible mask that implements the above procedure could be the one illustrated below.



The high-boost filtered image looks more like the original with a degree of edge enhancement, depending on the value of A .

A determines nature of filtering

SHARPENING HIGH PASS FILTER

High-Pass Filtering

- Shape of impulse response: +ve coefficients near its centre, -ve coefficients in periphery.
- E.g. 3x3 mask with +ve value in the middle, surrounded by 8 neighbours of -ve values.

$$1/9 \times \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Highpass filter example

A highpass filter may be computed as:

Highpass = Original – Lowpass

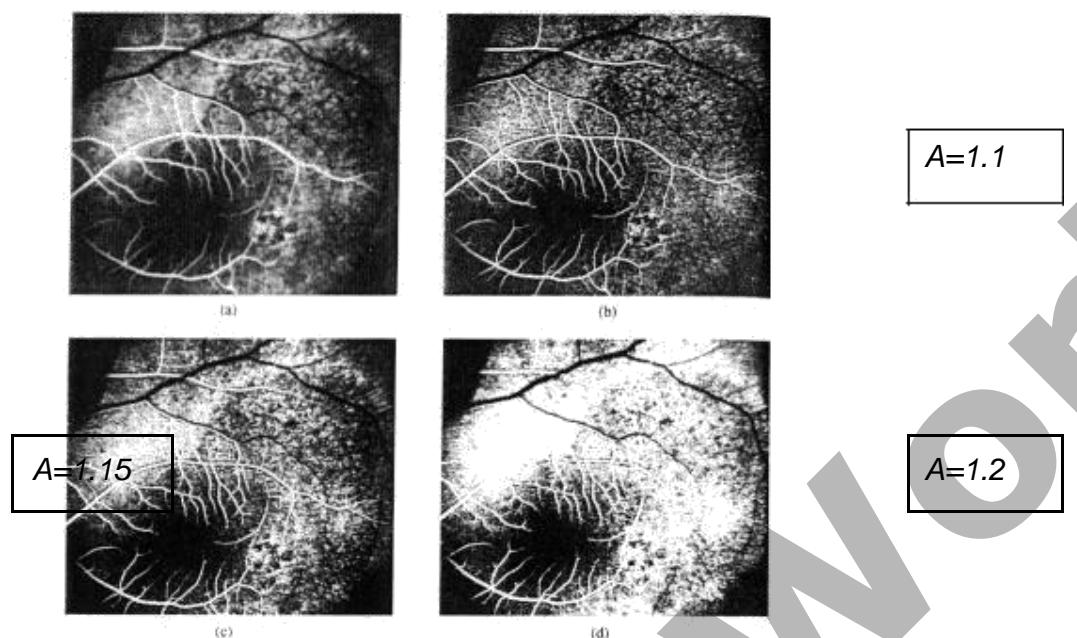
- Multiplying the original by an amplification factor yields a highboost or high-frequency-emphasis filter

$A=1$ for Highpass Filter

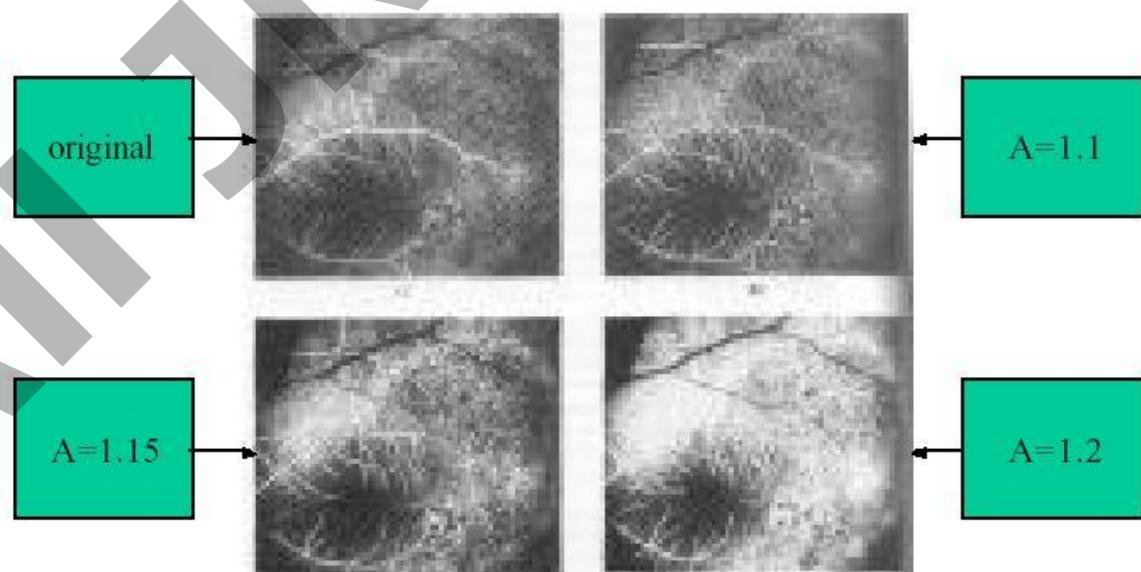
$$\begin{aligned} \text{Highboost} &= A(\text{Original}) - \text{Lowpass} \\ &= (A-1)(\text{Original}) + \text{Original} - \text{Lowpass} \\ &= (A-1)(\text{Original}) + \text{Highpass} \end{aligned}$$

If $A>1$, part of the original image is added to the highpass result (partially restoring low frequency components)

- Result looks more like the original image with a relative degree of edge enhancement that depends on the value of A
- May be implemented with the center coefficient value $w=9A-1$ ($A \geq 1$)



High-Boost Filtering: Illustration



High-Boost Filtering (cont'd.)

- Results look more like original with variable degree of edge enhancement (dependent on A).
- Subtracting a blurred image from original: unsharp masking, used in printing/publishing industry.

$$\begin{matrix} & \begin{matrix} -1 & -1 & -1 \\ -1 & w & -1 \\ -1 & -1 & -1 \end{matrix} \\ 1/9 x & \end{matrix} \quad (w = 9A - 1)$$

- A determines nature of filter.

Use of First Derivatives for Enhancement—The Gradient
Use of first derivatives for Image Sharpening (Non linear)

About two dimensional high pass spatial filters

An edge is the boundary between two regions with relatively distinct grey level properties. The idea underlying most edge detection techniques is the computation of a local derivative operator.

The magnitude of the first derivative calculated within a neighborhood around the pixel of interest, can be used to detect the presence of an edge in an image.

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 two-dimensional column vector.

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

The magnitude of this vector is given by

$$\begin{aligned}\nabla f &= \text{mag}(\nabla f) \\ &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}.\end{aligned}$$

The components of the gradient vector itself are linear operators, but the magnitude of this vector obviously is not because of the squaring and square root operations. On the other hand, the partial derivatives are not rotation invariant (isotropic), but the magnitude of the gradient vector is. Although it is not strictly correct, the magnitude of the gradient vector often is referred to as the *gradient*. In keeping with tradition, we will use this term in the following discussions, explicitly referring to the vector or its magnitude only in cases where confusion is likely.

$$\nabla f \approx |G_x| + |G_y|.$$

to denote image points in a 3*3 region, for example, the center point, z5 , denotes $f(x, y)$, z1 denotes $f(x-1, y-1)$, and so on

The gradient of an image $f(x, y)$ at location (x, y) is a vector that consists of the partial derivatives of $f(x, y)$ as follows.

$$\begin{array}{c} f(x, y) \\ \overline{} \\ f(x, y) \\ \overline{} \\ f(x, y) \\ \overline{} \\ y \end{array}$$

The magnitude $M(x, y)$ of this vector, generally referred to simply as the gradient f is

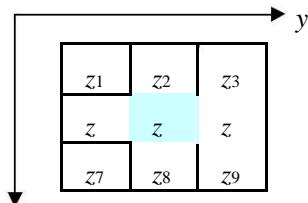
$$f(x, y) \text{ mag}(f(x, y)) = \sqrt{\frac{f(x,y)^2}{x} + \frac{f(x,y)^2}{y}}^{1/2}$$

Size of $M(x, y)$ is same size as the original image. It is common practice to refer to this image as **gradient image** or simply as gradient.

Common practice is to approximate the gradient with absolute values which is simpler to implement as follows.

$$f(x, y) \begin{vmatrix} f(x, y) & f(x, y) \\ x & y \end{vmatrix}$$

Consider a pixel of interest $f(x, y)$ at point z_5 and a rectangular neighborhood of size 3x3 pixels (including the pixel of interest) as shown below.



Roberts operator

Above Equation can be approximated at point z_5 in a number of ways. The simplest is to use the difference $(z_5 - z_8)$ in the x direction and $(z_5 - z_6)$ in the y direction. This approximation is known as the **Roberts** operator, and is expressed mathematically as follows.

$$f \begin{vmatrix} z_5 & z_8 \\ z_5 & z_6 \end{vmatrix}$$

Another approach for approximating the equation is to use cross differences

$$f \begin{vmatrix} z_5 & z_9 \\ z_6 & z_8 \end{vmatrix}$$

Above Equations can be implemented by using the following masks. The original image is convolved with both masks separately and the absolute values of the two outputs of the convolutions are added.

1	0
-1	0

1	-1
0	0

Roberts operator

1	0
0	-1

0	1
-1	0

Roberts operator

- Roberts cross-gradient operators, 2x2

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6)$$

$$\nabla f = [G_x^2 + G_y^2]^{1/2} = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2}$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

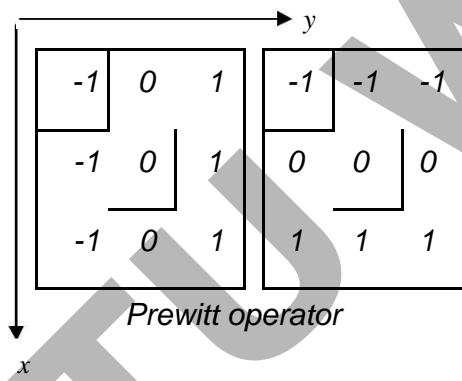
Prewitt operator

$$f(x, y) \left| \begin{array}{c} f(x, y) \\ x \\ f(x, y) \\ y \end{array} \right|$$

Another approximation of above equation but using now a 3 3 mask is the following.

$$f \quad |(z_7 \ z_8 \ z_9) \ (z_1 \ z_2 \ z_3) \ (z_3 \ z_6 \ z_9) \ (z_1 \ z_4 \ z_7)|$$

(4) This approximation is known as the **Prewitt** operator. Equation (4) can be implemented by using the following masks. Again, the original image is convolved with both masks separately and the absolute values of the two outputs of the convolutions are added.

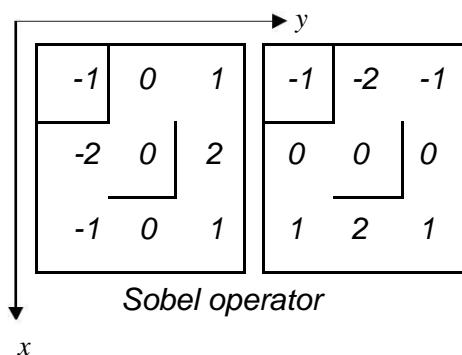


Sobel operator.

Definition and comparison with the Prewitt operator (gives weightage to centre pixel)
The most popular approximation of equation (1) but using a 3 3 mask is the following.

$$f \quad |(z_7 \ 2z_8 \ z_9) \ (z_1 \ 2z_2 \ z_3) \ (z_3 \ 2z_6 \ z_9) \ (z_1 \ 2z_4 \ z_7)|$$

This approximation is known as the **Sobel** operator.



If we consider the left mask of the Sobel operator, this causes differentiation along the y direction.

■ Sobel operators, 3x3

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

$$\nabla f \approx |G_x| + |G_y|$$

the weight value 2 is to achieve smoothing by giving more importance to the center point

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

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

Lecture Notes

UNIT 3 : Image Enhancement in the *Frequency Domain*

- Image Enhancement (Frequency Domain) Filtering in frequency domain
- Obtaining frequency domain filters from spatial filters
- Generating filters directly in the frequency domain
- Low pass (smoothing) and High pass (sharpening) filters in Frequency domain

Filtering in the Frequency Domain

Filters in the frequency domain can be divided in four groups:

Low pass filtersIMAGE GETS SMOOTHENED and BLURRED

Remove frequencies away from the origin (Origin will be at the centre in frequency domain)
Commonly, frequency response of these filters is symmetric around the origin;
The largest amount of energy is concentrated on low frequencies, but it represents just image luminance and visually not so important part of image

High pass filtersEDGES DETECTION or SHARENS THE IMAGE

Remove signal components around origin
Small energy on high frequency corresponds to visually very important image features such as edges and details.
Sharpening = boosting high frequency pixels

Band pass filters

Allows frequency in the band between lowest and the highest frequencies;

Stop band filters

Remove frequency band.

To remove certain frequencies during filtering, set their corresponding $F(u)$ coefficients to zero

Low pass filters (smoothing filters)

- Ideal Low Pass filters
- Butterworth low pass filters
- Gaussian low pass filters

High Pass Filters (Sharpening filters)

- Ideal High pass filters
- Butterworth High pass filters
- Gaussian High pass filters

Laplacian in frequency domain
High boost , high frequency emphasis filters
Homomorphic filters

Some Basic Properties of the Frequency Domain (wrt spatial domain)

Frequency is directly related to the spatial rate of change (of brightness or gray values). Therefore, slowest varying frequency component ($u=v=0$) corresponds to the average intensity level of the image. Corresponds to the origin of the Fourier Spectrum.

Higher frequencies corresponds to the faster varying intensity level changes in the image. The edges of objects or the other components characterized by the abrupt changes in the intensity level corresponds to higher frequencies.

Spatial Domain

Frequency Domain

Frequency Domain Filtering

Images can be processed in either the spatial or frequency domain

Frequency domain filters can achieve the same results as that of spatial filtering by altering the DFT coefficients directly

Frequency domain filtering can be generalized as the multiplication of the spectrum \mathbf{F} of an image by a transfer function \mathbf{H} .

$$G(u,v) = H(u,v)F(u,v)$$

In spatial domain,

Let $g(x,y)$ be a desired image formed by the convolution of an image $f(x,y)$ and a linear, position invariant operator, $h(x,y)$:

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

We can select $H(u,v)$ so that the desired image

$$g(x, y) = \mathcal{F}^{-1}[H(u,v)F(u,v)]$$

exhibits some highlighted features of $f(x,y)$. For instance, edges in $f(x,y)$ can be accentuated by using a function $H(u,v)$ that emphasizes the high frequency components of $F(u,v)$.

We have access to Fourier transform magnitude (spectrum) and phase angle. Phase is not useful for visual analysis.

Convolution

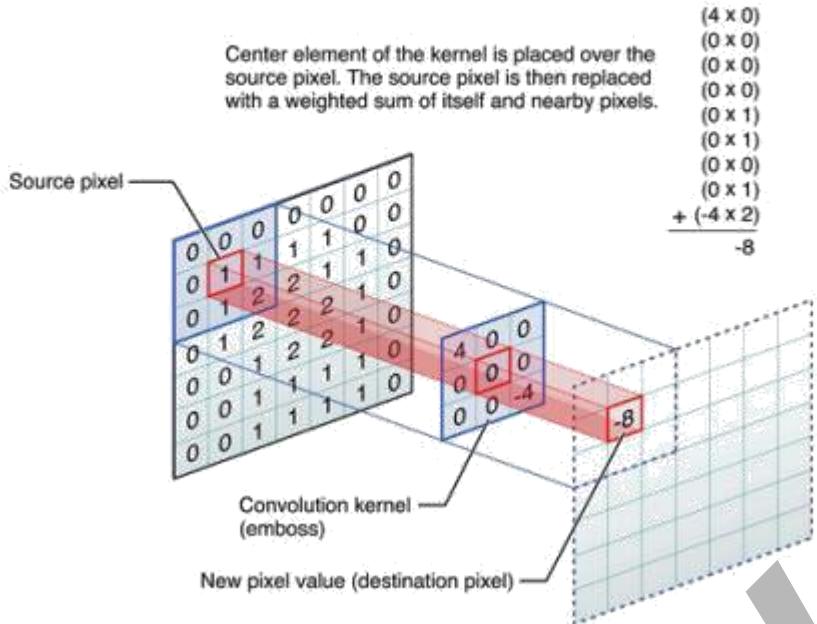
Convolution in the spatial domain corresponds to multiplication in the Fourier domain.

Note that many implementations of convolution produce a larger output image than this because they relax the constraint that the kernel can only be moved to positions where it fits entirely within the image. Instead, these implementations typically slide the kernel to all positions where just the top left corner of the kernel is within the image. Therefore the kernel 'overlaps' the image on the bottom and right edges. One advantage of this approach is that the output image is the same size as the input image. Unfortunately, in order to calculate the output pixel values for the bottom and right edges of the image, it is necessary to *invent* input pixel values for places where the kernel extends off the end of the image. Typically pixel values of zero are chosen for regions outside the true image, but this can often distort the output image at these places. Therefore in general if you are using a convolution implementation that does this, it is better to clip the image to remove these spurious regions. Removing $n - 1$ pixels from the right hand side and $m - 1$ pixels from the bottom will fix things.

If the image size is $M \times N$, kernel size is $m \times n$, the convolved image size will

$$M-m+1, N-n+1$$

Flip the kernel or filter horizontally and vertically prior to array multiplication with the image.



Convolution, the mathematical, *local* operation defined in Section 3.1 is central to modern image processing. The basic idea is that a window of some finite size and shape-- the *support*--is scanned across the image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window itself. The window with its weights is called the *convolution kernel*. This leads directly to the following variation on eq. . If the filter $h[j,k]$ is zero outside the (rectangular) window $\{j=0,1,\dots,J-1; k=0,1,\dots,K-1\}$, then, using eq. , the convolution can be written as the following finite sum:

$$c[m,n] = a[m,n] \otimes h[m,n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} h[j,k] a[m-j, n-k]$$

$$(\cdot, \cdot) = (\cdot, \cdot) * h(\cdot, \cdot) = \sum h[\cdot, \cdot] [\cdot - 1, \cdot - 1]$$

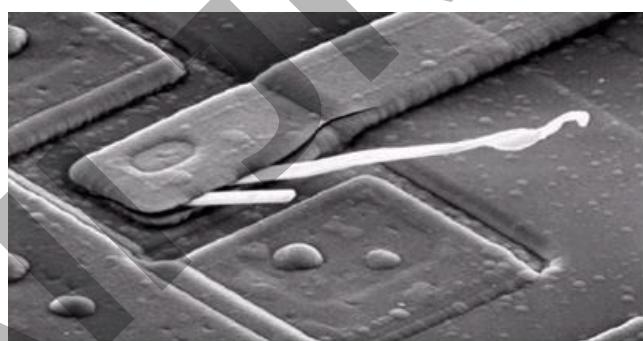
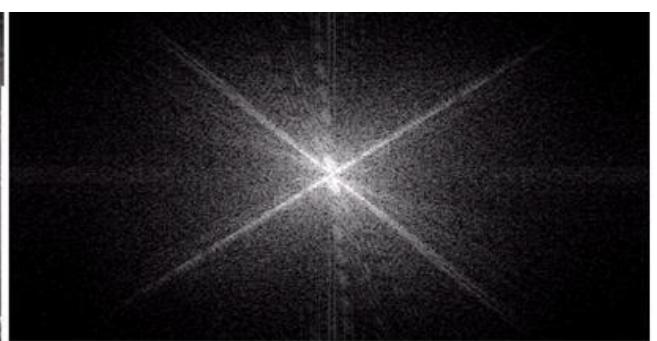
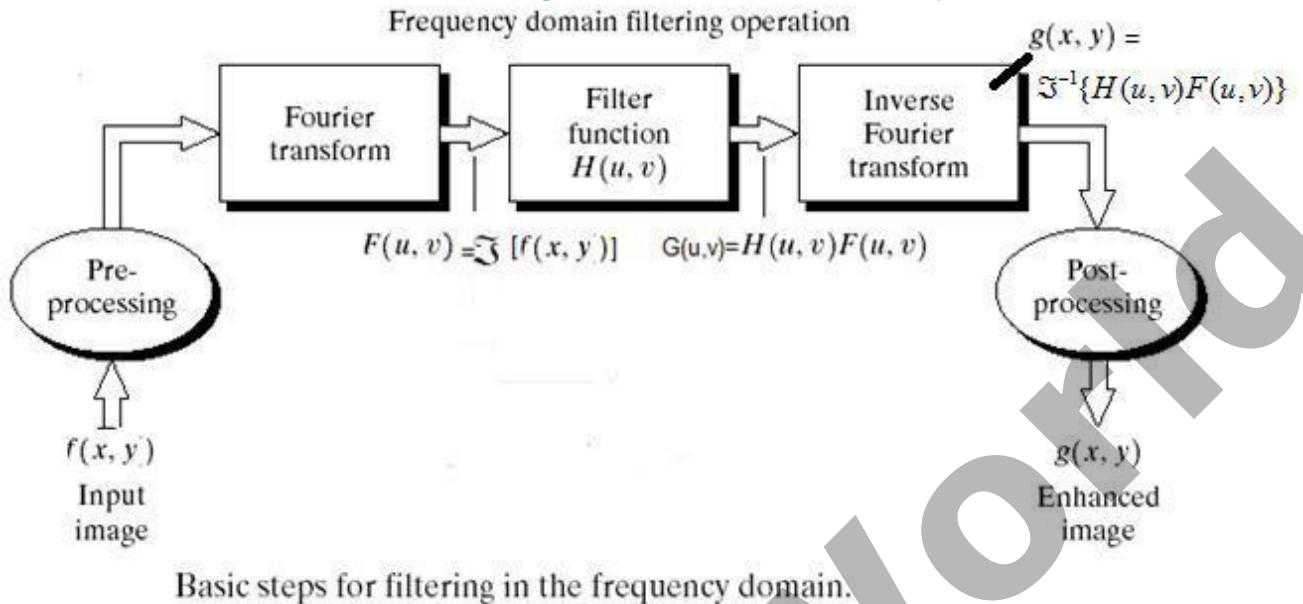


Image of a damaged integrated



Fourier spectrum

Basic Steps for Filtering in the Frequency Domain:



$g(x,y)$ Filters affect the real and imaginary parts equally, and thus no effect on the phase. These filters are called **zero-phase-shift** filters

Zero Pad the input image $f(x,y)$ to $p = 2M$, and $q = 2N$, if arrays are of same size.

If functions $f(x,y)$ and $h(x,y)$ are of size $A \times B$ and $C \times D$, respectively, zero padding is:
 $P \geq A + C - 1$, $Q \geq B + D - 1$

If both filters are of the same size, it is more efficient to do filtering in the frequency domain
Design a much smaller filter in the spatial domain that approximates the performance

1. Multiply the input padded image by $(-1)^{x+y}$ to center the transform.
2. Compute $F(u,v)$, the DFT of the image from (1).
3. Multiply $F(u,v)$ by a filter function $H(u,v)$.
4. Compute the inverse DFT of the result in (3).
5. Obtain the real part of the result in (4).
6. Multiply the result in (5) by $(-1)^{x+y}$.

Given the filter $H(u,v)$ (filter transfer function OR filter or filter function) in the frequency domain, the Fourier transform of the output image (filtered image) is given by:

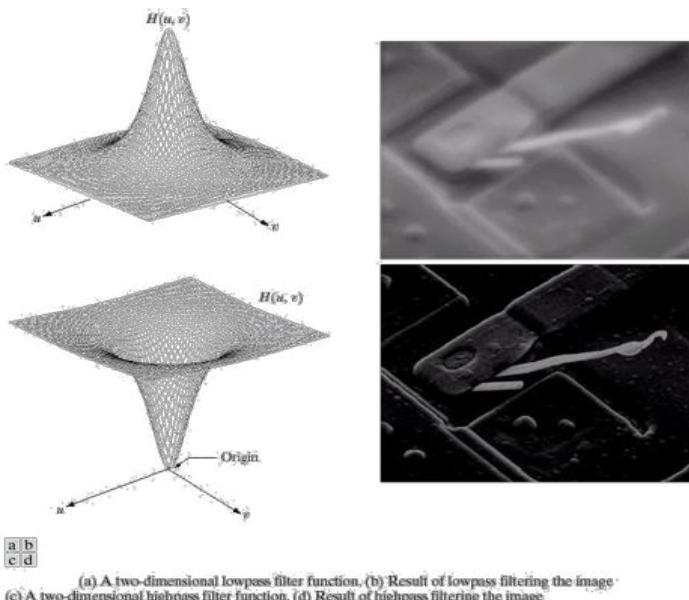
$$G(u,v) = H(u,v) F(u,v) \quad \text{Step (3) is array multiplication}$$

The filtered image $g(x,y)$ is simply the inverse Fourier transform of $G(u,v)$.

$$g(x,y) = F^{-1}[G(u,v)] = F^{-1}[H(u,v) F(u,v)] \quad \text{Step (4)}$$

$$g_p(x,y) = \{\text{real}[\mathcal{F}^{-1}[G(u,v)]]\}(-1)^{x+y}$$

F, H, g , are arrays of same size as in input image. F^{-1} is IDFT.



- **lowpass filter:** A filter that attenuates high frequencies while passing the low frequencies.
- Low frequencies represent the gray-level appearance of an image over smooth areas.
- **highpass filter:** A filter that attenuates low frequencies while passing the high frequencies.
- High frequencies represents the details such as edges and noise.

Correspondence between filtering in spatial and frequency domains

Filtering in frequency domain is multiplication of filter times fourier transform of the input image

$$G(u, v) = H(u, v) F(u, v)$$

Let us find out equivalent of frequency domain filter $H(u, v)$ in spatial domain.

Consider $f(x, y) = \delta(x, y)$, we know $F(u, v) = 1$

Then filtered output $F^{-1}[H(u, v) F(u, v)] = F^{-1}[H(u, v)]$

But this is inverse transform of the frequency domain filter

But this is nothing but filter in the spatial domain.

Conversely, if we take a forward Fourier transform of a spatial filter, we get its fourier domain representation.

Therefore, two filters form a transform pair

$$h(x, y) \Leftrightarrow H(u, v),$$

Since $h(x, y)$ can be obtained from the response of a frequency domain filter to an impulse, $h(x, y)$ spatial filter is sometimes referred as finite impulse response filter (FIR) of $H(u, v)$

$$f(x, y) * h(x, y) \Leftrightarrow F(u, v) H(u, v)$$

or Inverse DFT of the product $F(u, v) H(u, v)$ yields $f(x, y) * h(x, y)$

How to generate $H(u,v)$ from $h(x,y)$?

To find $H(u,v)$ from $h(x,y)$

If $h(x,y)$ is given in the spatial domain, we can generate $H(u,v)$ as follows:

1. Form $h_p(x,y)$ by padding with zeroes.
2. Multiply by $(-1)^{x+y}$ to center its spectrum.
3. Compute its 2D DFT to obtain

$H(u,v)$ To find $h(x, y)$ from $H(u,v)$:

1 Centering: $H(u, v) * (-1)^{u+v}$

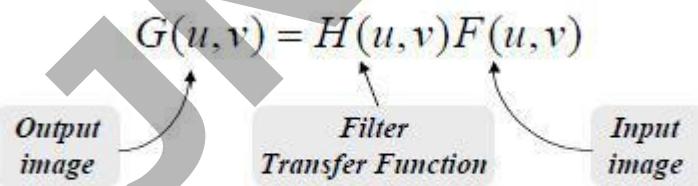
2 Inverse Fourier transform

3 Multiply real part by

$(-1)^{x+y}$ Properties of $h(x, y)$:

- 1 It has a central dominant circular component (providing the blurring)
- 2 It has concentric circular components (rings) giving rise to the ringing effect.

Implementation:



- The analogous operation in spatial domain can be implemented as follow

$$g(x, y) = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} h(x-i, y-k) f(i, k)$$

with $x, y = 0, 1, 2, \dots, N-1$.

- Note that all functions are properly extended to generate a linear convolution in frequency domain.

- h is the spatial domain representation of the $H(u,v)$ and it is called *spatial convolution mask*.
- The relationship between $h(x,y)$ and $H(u,v)$ is defined as follow

$$H(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} h(x,y) e^{-j2\pi \frac{ux+vy}{N}}$$

where $u, v = 0, 1, 2, \dots, N-1$.

- Now, if we restrict $h(x,y)$ as follow

$$\hat{h}(x,y) = \begin{cases} h(x,y), & 0 \leq x, y \leq n-1 \\ 0, & n \leq x, y \leq N-1 \end{cases}$$

- This restriction in effect creates an $n \times n$ convolution mask \hat{h} , with Fourier transform of

$$\hat{H}(u,v) = \frac{1}{N} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} \hat{h}(x,y) e^{-j2\pi \frac{ux+vy}{N}}$$

where $u, v = 0, 1, 2, \dots, N-1$.

- The objective is to find the coefficients of $\hat{h}(x,y)$ such that mean square error is minimized.

$$e^2 = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} |\hat{H}(u,v) - H(u,v)|^2$$



Image smoothing using Low Pass Filters

Ideal LPF

Butterworth LPF has filter order.

For higher order, BLPF tends to ILPF

For lower order BLPF tends to Gaussian

Gaussian LPF

Low Pass Filtering

- ▶ It attenuates high frequency components of an image while leaving low frequency components intact.
- ▶ Low pass filtering can be naturally accomplished in the frequency domain since the frequency components are explicitly isolated in the spectrum.
- ▶ The DFT coefficients correspond to frequency components of the source and that their frequency increases with increasing distance from the center of the shifted spectrum.
- ▶ Low pass filtering is then accomplished by zeroing the amplitude of the high frequency DFT coefficients. Determining which DFT coefficients should be zeroed amounts to determining how far the coefficients are from the center of the shifted spectrum.
- ▶ Typically, a threshold radius is chosen such that all DFT coefficients outside of this threshold radius have their magnitude set to zero while all DFT coefficients falling inside of the threshold are unchanged (passed through).

IDLF:

A 2D low pass filter that passes all frequencies without attenuation, within a circle of radius D_0 from the origin and cuts off all frequencies outside the circle is a ILPF

IPLF Can not be realized with electronic components but can be simulated on computers.

Smoothing Frequency Domain Filters:

Ideal Low pass Filters

$H(u,v)$ are discrete filter functions of $P \times Q$ size.

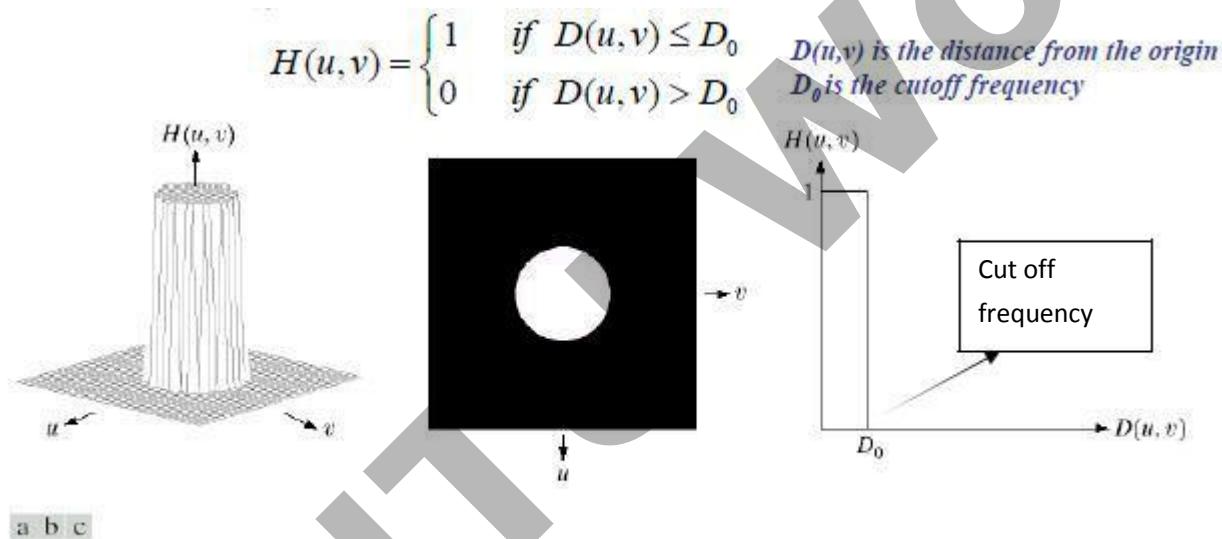
The discrete variables are in the size $u=0,1,2,\dots,P-1; v=0,1,2,\dots,Q-1$ (P,Q are padded sizes)

The centre of the frequency rectangle is

$$D(u,v) = [(u)^2 + (v)^2]$$

$$D(u,v) = \sqrt{[(u-P/2)^2 + (v-Q/2)^2]}$$

- In 2D, the cutoff frequencies lie on a circle.



a b c

(a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

The point of transition between $H(u,v)=0$ and $H(u,v)=1$ is cut off frequency which is D_0 . D_0 is a positive constant

To find $h(x, y)$:

Centering: $H(u, v) * (-1)^{u+v}$

Inverse Fourier transform

Multiply real part by $(-1)^{x+y}$

Properties of $h(x, y)$:

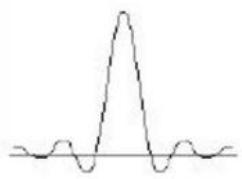
It has a central dominant circular component (providing the blurring)

It has concentric circular components (rings) giving rise to the ringing effect.

As the filter radius increases, less and less power is removed/filtered out, more and more details are preserved.

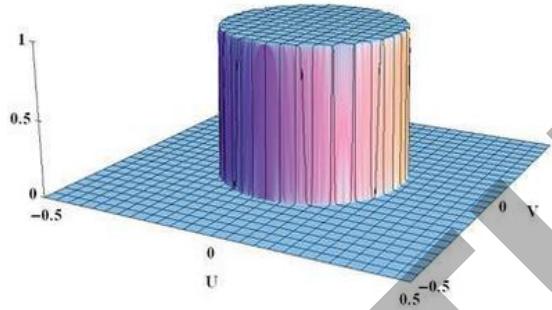
- Ringing effect is the consequence of applying ideal lowpass filters

- Sharp cutoff frequencies produce an overshoot of image features whose frequency is close to the cutoff frequencies (**ringing effect**).

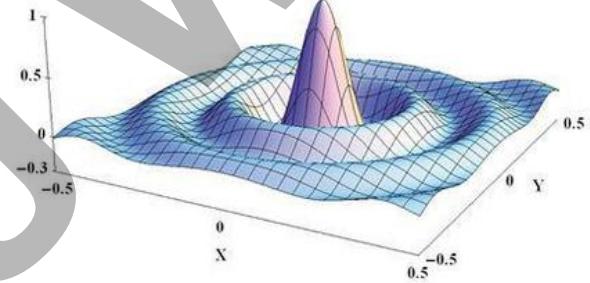


Example of $h(x)$ from the inverse FT of a disc (ILPF) with radius 5.

The percentage of power enclosed in circles of 10, 30, 60, 160, 460 pixels is generally around 87, 93.1, 95.7, 97.8, 99.2 %.



(a) $\mathcal{H}(u, v)$.



(b) $h(x, y)$.

Ideal low-pass filter in (a) the frequency domain and (b) the spatial domain.

Cutoff Frequency of an Ideal Lowpass Filter:

One way of specifying the cutoff frequency is to compute circles enclosing specified amounts of total image power.

- **Calculating P_T which is the total power of the transformed image**

$$P_T = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} P(u, v) \quad u = 0, 1, \dots, M-1, v = 0, 1, \dots, N-1$$

The cutoff frequency D_o can be determined by specifying the a percent of the total power enclosed by a circle centered at the origin. The radius of the circle is the cutoff frequency D_o .

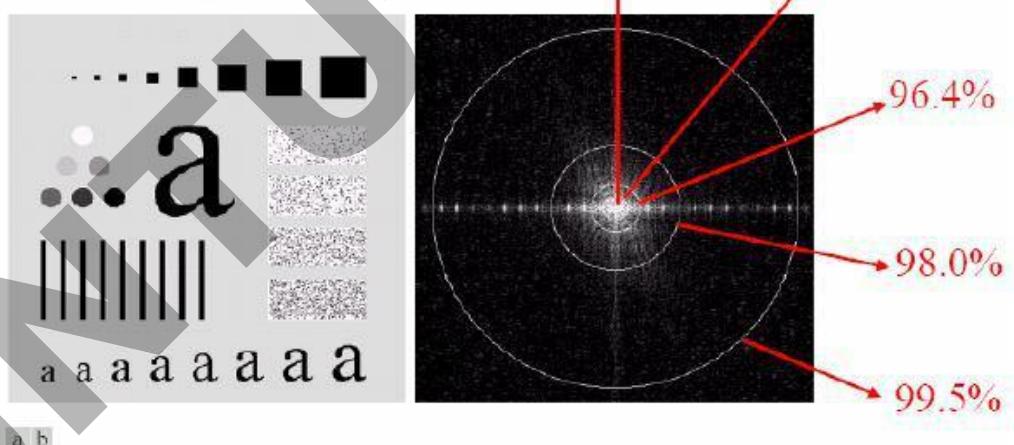
$$\alpha = 100 \left[\sum_u \sum_v P(u, v) / P_T \right] \quad u \leq \text{radius}(D_o), v \leq \text{radius}(D_o)$$

Total image power P_T is obtained by summing the components of the power spectrum. Enclosed power = Remained power after filtration

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

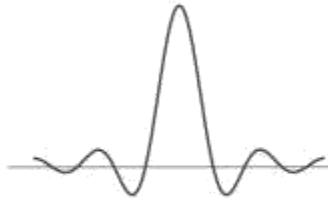
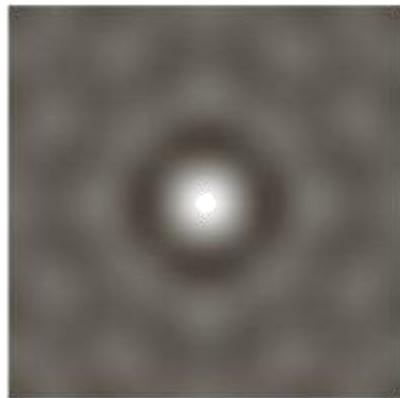
$$\text{Total image power : } P_T = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} P(u, v)$$

$$\text{Enclosed power : } \alpha = 100 \left[\frac{\sum_u \sum_v P(u, v)}{P_T} \right]$$



(a) An image of size 500×500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Explaining the blurring and “ringing” properties of ILPFs



Representation in the spatial domain of an ILPF of radius 5 and size 1000x1000

Intensity profile of a horizontal line passing through the centre of the image

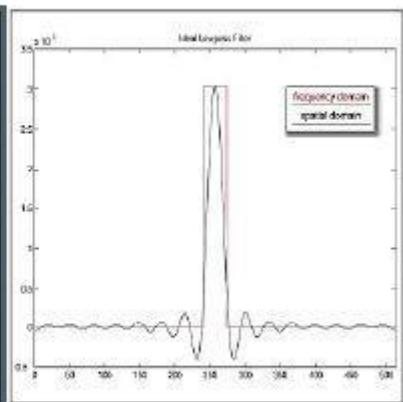
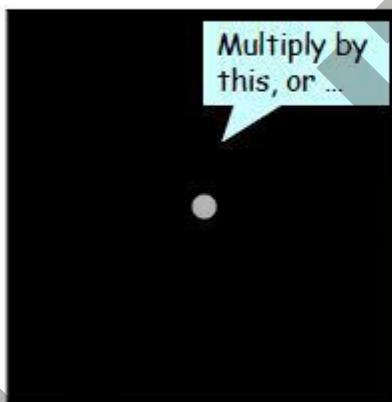
Filtering in the spatial domain can be done by convolving $h(x,y)$ with image $f(x,y)$. Imagine each pixel in the image as a discrete impulse whose strength is proportional to the intensity of the image at that location.

Convolving a sinc with an impulse copies the sinc at the location of the impulse.

Thus centre lobe of the sinc causes **blurring** while outer smaller lobes are responsible for **ringing**.

Ideal Lowpass Filter

Image size: 512x512
FD filter radius: 16



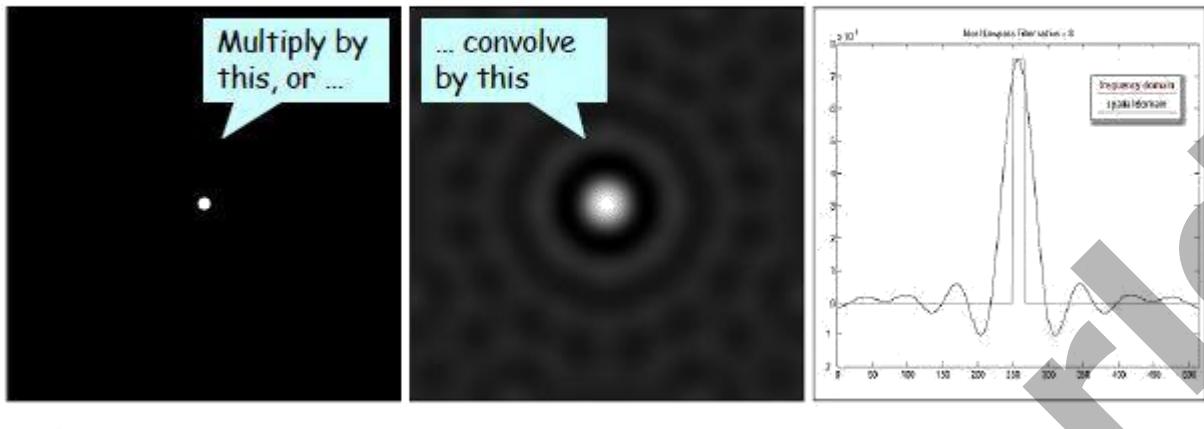
Fourier Domain Rep.

Spatial Representation

Central Profile

Ideal Lowpass Filter

Image size: 512x512
FD filter radius: 8



Fourier Domain Rep.

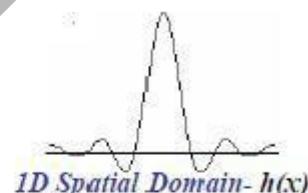
Spatial Representation

Central Profile

Blurring and Ringing properties of ILPF:

The spatial domain filters center component is responsible for blurring.

- The circular components are responsible for the ringing artifacts.*



inverse DFT



2D Frequency Domain- $H(u,v)$

2D Spatial Domain - $h(x,y)$

Butterworth Low Pass Filter

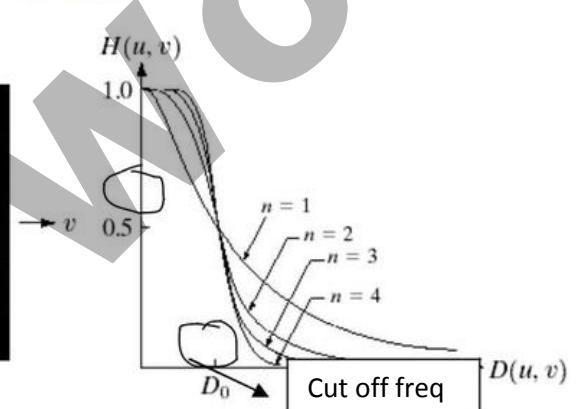
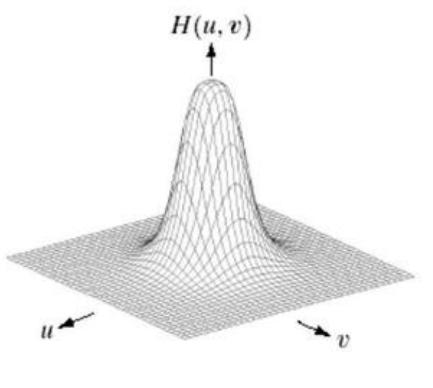
Smoothing Frequency Domain Filters: Butterworth Lowpass Filters

- In practice, we use filters that attenuate high frequencies smoothly (e.g., Butterworth LP filter) → less ringing effect

The transfer function of a Butterworth LPF of order n, and with a cut off frequency at a distance D_0 from origin is given by:

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$$

*D(u,v) is the distance from the origin
 D_0 is the cutoff frequency.
n is the order of the filter*



a b c

(a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

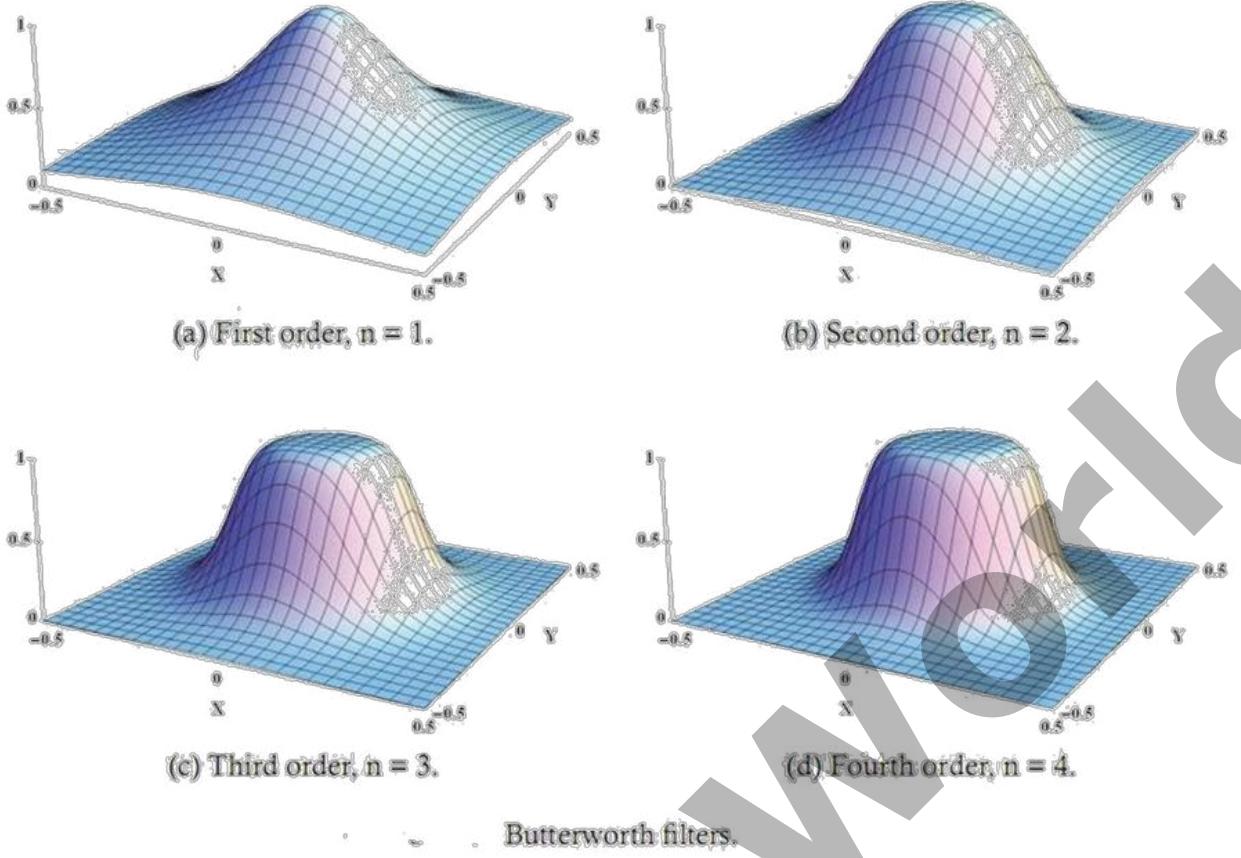
Unlike in ILPF, BLPF transfer function does not have a sharp discontinuity that gives a clear cut off between passed and filtered frequencies.

For BLPF, the cutoff frequency is defined as the frequency at which the transfer function has value which is half of the maximum

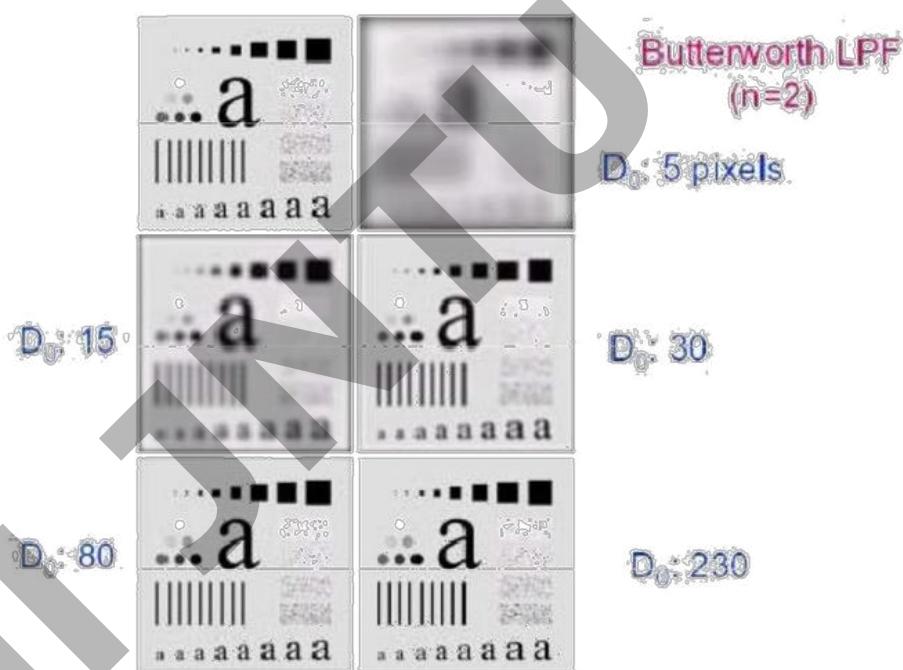
The BLPF with order of 1 does not have any ringing artifact.

• *BLPF with orders 2 or more shows increasing ringing effects as the order increases.*

- ▶ Butterworth filter is a low-pass filter with smooth edges such that there is no (or minimal) ringing in the spatial domain
 - ▶ Parameterized on “order” – defines the sharpness of the filter

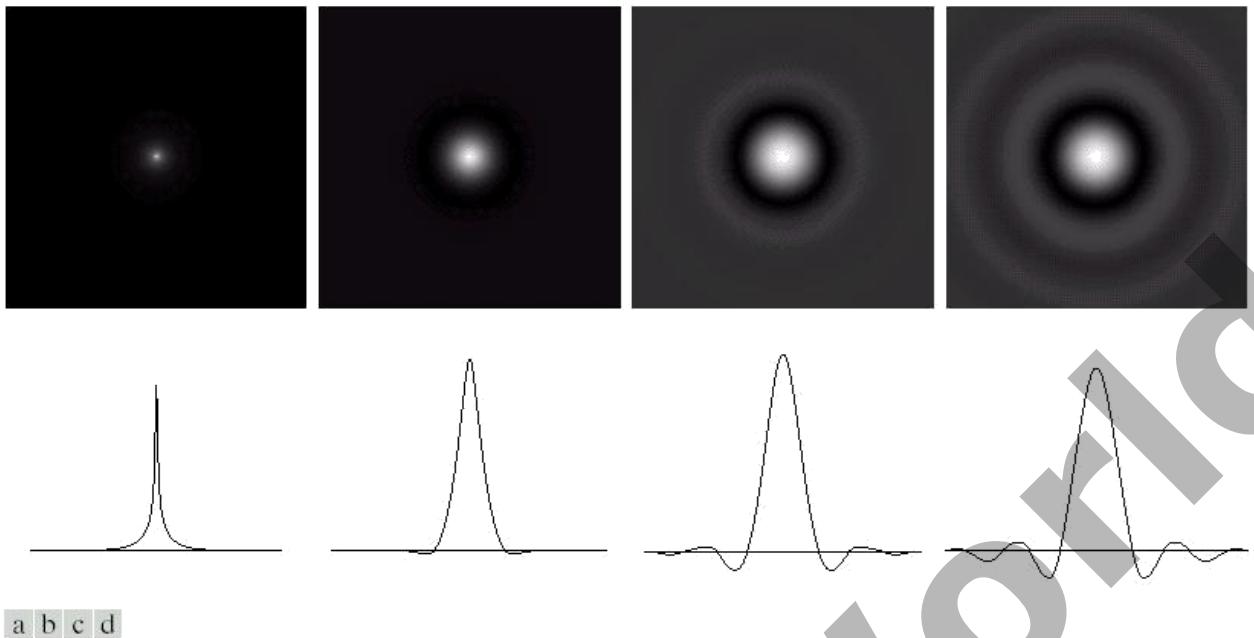


Butterworth filters.



Unlike in ILPF, here Smooth transition in blurring is a function of increasing cut off frequency. Due to smooth transition between low and high frequencies, ringing is not visible.

BLPF has no ringing in spatial domain for order 1. Ringing is visible and similar to ILPF for order $n=20$ in BLPF.



(a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

Unlike ILPF, BLPF does not have a sharp discontinuity that gives a clear cut off between assed and filtered frequencies

If $D(u,v)=D_0$, defines a cut off frequency locus at $H(u,v)$ is down by 50%

Smoothing Frequency Domain Filters: Gaussian Lowpass Filters

Gaussian Lowpass Filters (GLPF) in two dimensions is given

$$H(u,v) = e^{-(u^2 + v^2)/2\sigma^2}$$

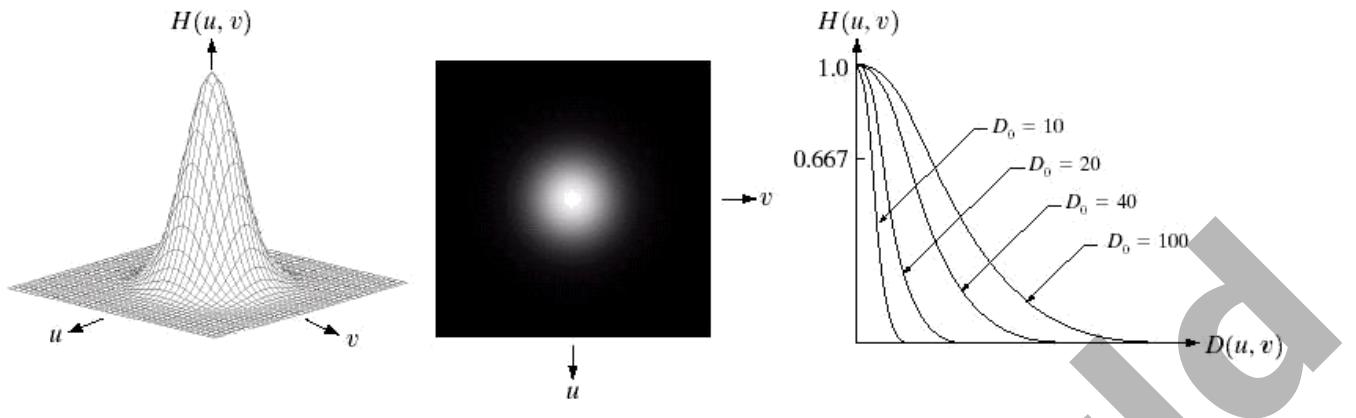
$\sigma^2 = D_0^2$ for commonality sake, σ is a measure of the spread around centre.

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

$D(u,v)$ is the distance from the origin
 D_0 is the cutoff frequency.

σ is a measure of spread about centre and is equal to D_0

Ringing is not produced from this Gaussian filter.



a b c

(a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0 .

(GLPF): *The inverse Fourier transform of the Gaussian Low pass filter is also Gaussian in the Spatial domain.*

- Therefore there is no ringing effect of the GLPF.

Ringing artifacts are not acceptable in fields like medical imaging. Hence use Gaussian instead of the ILPF/BLPF.

Gaussian low pass filter

- ▶ Other well-known frequency domain low pass filters include the Chebyshev and the Gaussian transfer functions.
 - ▶ The Gaussian low pass filter has the very interesting property of having the same form in both the Fourier and spatial domains. In other words, the **DFT of a Gaussian function is itself a Gaussian function**.
 - ▶ A Gaussian low pass filter introduces no ringing when applied either in the spatial or frequency domains. The Gaussian transfer function is given as

Sharpening Frequency-Domain Filters

The high-frequency components are: edges and sharp transitions such as noise. Sharpening can be achieved by high pass filtering process, which attenuates low frequency components without disturbing the high-frequency information in the frequency domain.

The filter transfer function, $H_{hp}(u,v)$, of a high pass filter is given by:

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

Where $H_{lp}(u,v)$, is the transfer function of the corresponding lowpass filter.

Proof:

$$\begin{aligned} F_{hp}(u,v) &= F(u,v) - F_{lp}(u,v) = F(u,v) - H_{lp}F(u,v) \\ &\Downarrow \\ F_{hp}(u,v) &= (1 - H_{lp})F(u,v) \\ &\Downarrow \\ H_{hp} &= (1 - H_{lp}) \end{aligned}$$

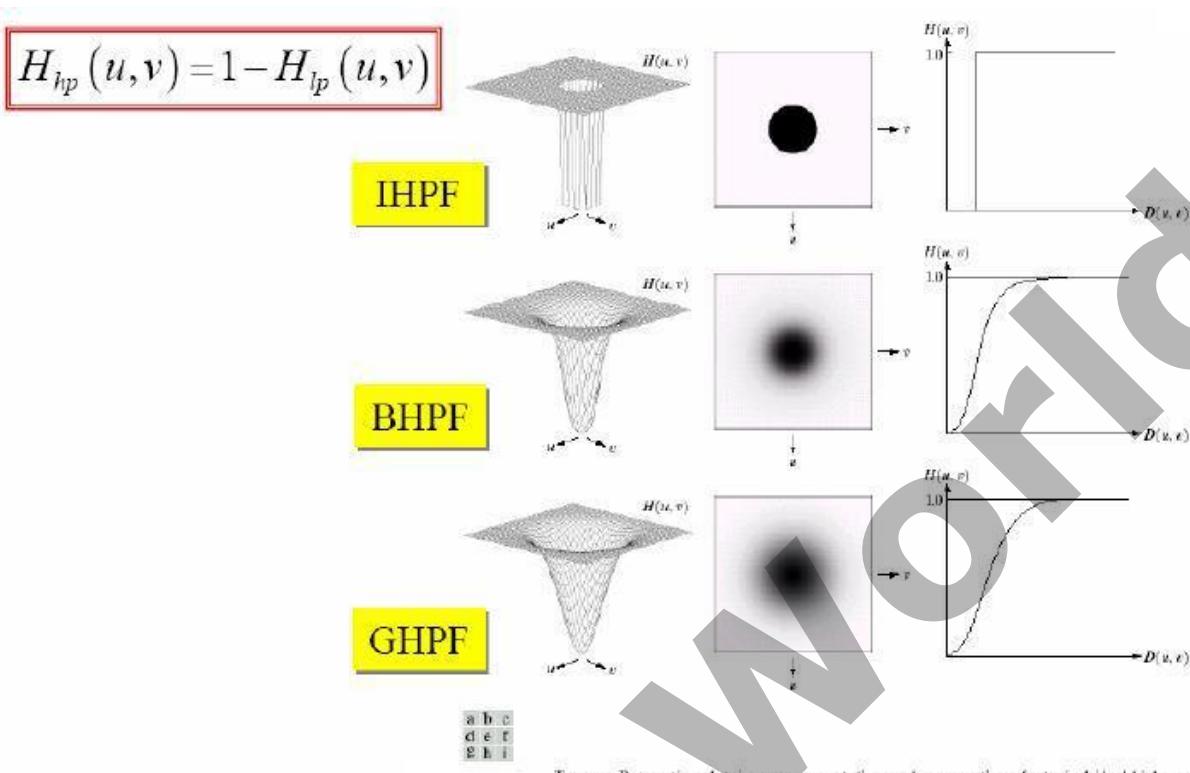
$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases} \quad \begin{matrix} D(u,v) \text{ is the distance from the origin} \\ D_0 \text{ is the cutoff frequency} \end{matrix}$$

$$H_{hp} = 1 - H_{lp} = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

We will consider only 3 types of sharpening high pass filters :

- Ideal Highpass filters,
- Butterworth Highpass filters,

•Gaussian Highpass filters



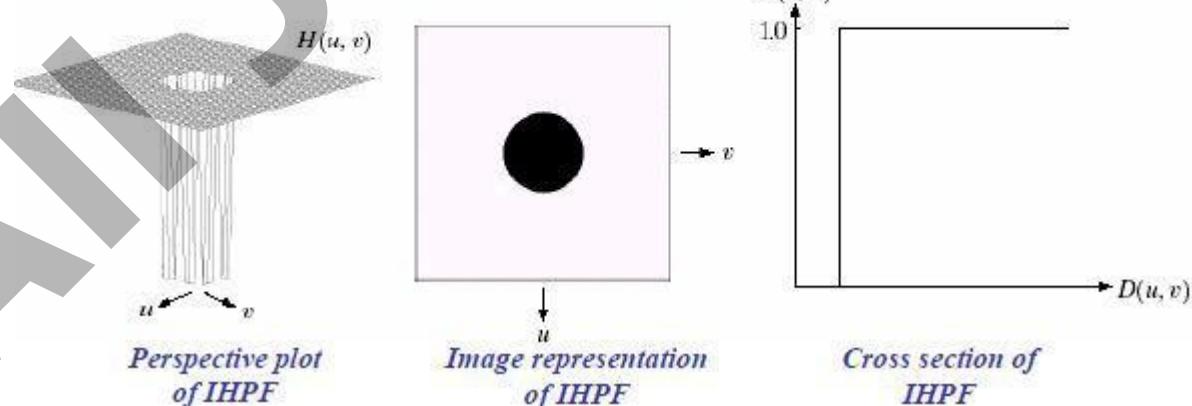
Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Ideal High pass Filter (IHPF):

Simply cuts off all the low frequencies lower than the specified cutoff frequency. The filter transfer function:

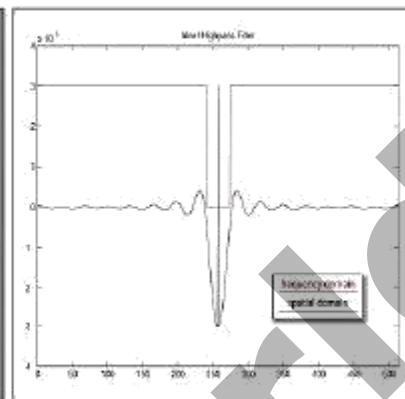
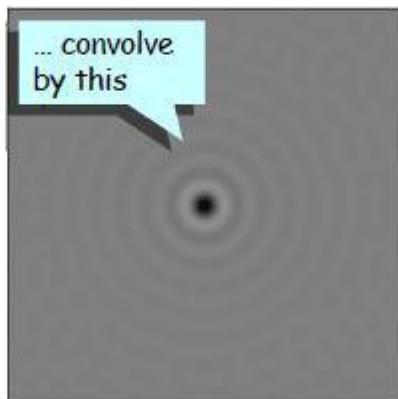
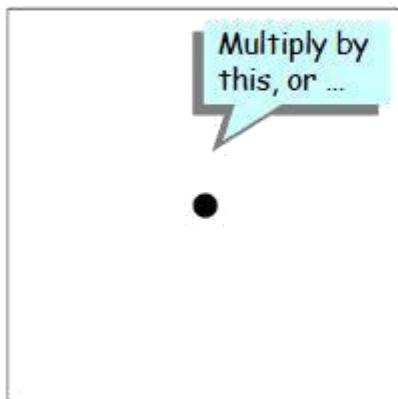
$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$

*D(u,v) is the distance from the origin
D₀ is the cutoff frequency*

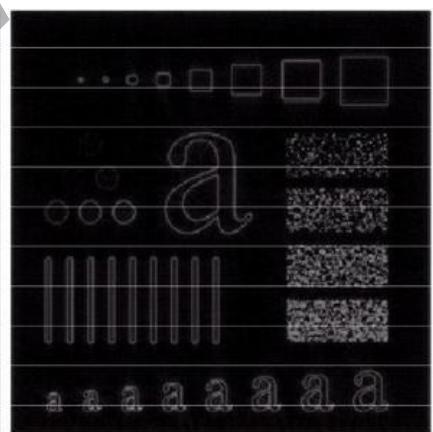
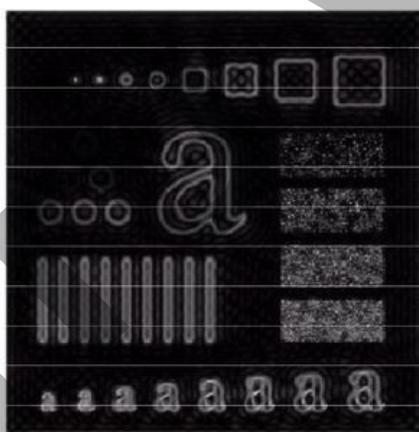
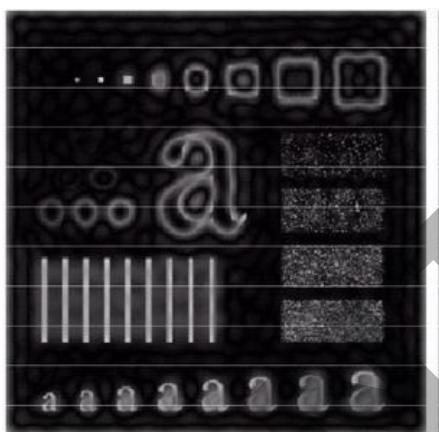


Ideal Highpass Filter

Image size: 512x512
FD notch radius: 16



The ringing artifacts occur at low cutoff frequencies



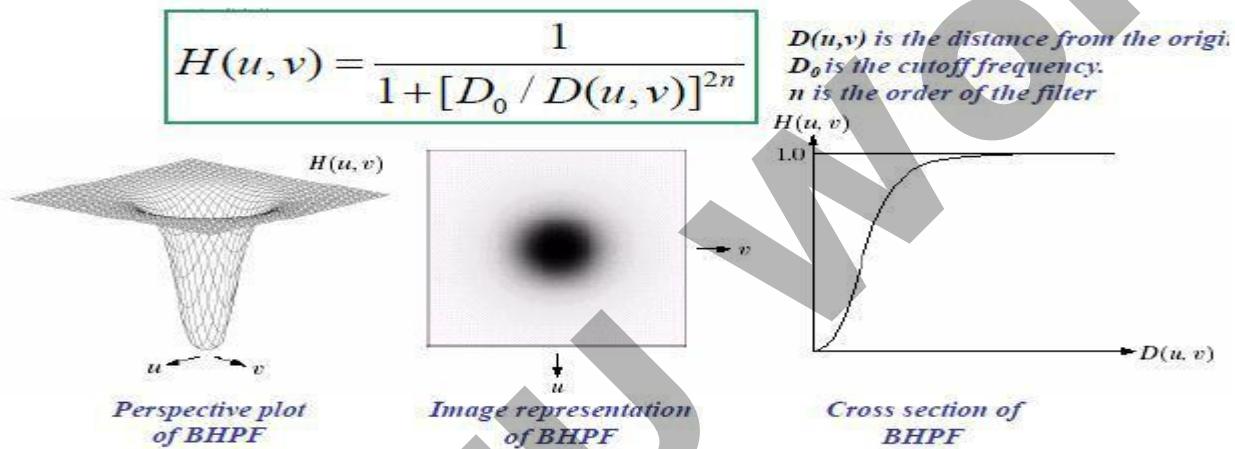
High Pass Filters

- Ideal high-pass filter (IHPF)
- Butterworth high-pass filter (BHPF)
- Gaussian high-pass filter (GHPF)
- Difference of Gaussians
- Unsharp Masking and High Boost filtering

Sharpening Frequency-Domain Filters

• Butterworth Highpass Filter (BHPF):

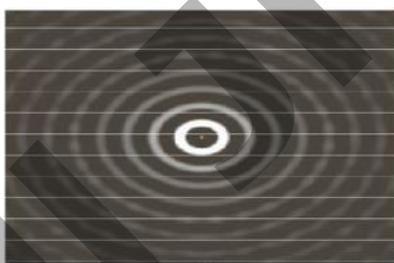
The transfer function of BHPF of order n and with a specified cutoff frequency is given by:



Smoother results are obtained in BHPF when compared IHPF. There is almost no ringing artifacts.

Spatial Representation of High pass filters

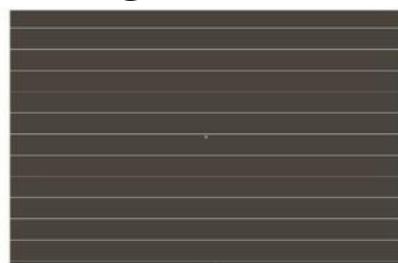
IHPF



BHPF



GHPF



a b c

FIGURE 4.53 Spatial representation of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding intensity profiles through their centers.

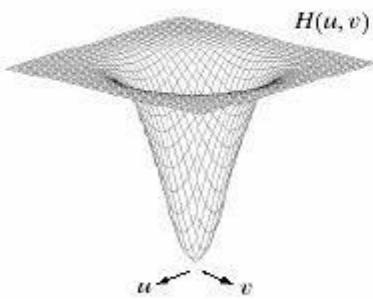
Sharpening Frequency-Domain Filters

• Gaussian High pass Filter (GHPF):

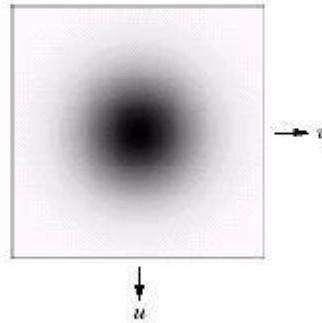
The transfer function of GHPF is given by:

$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$

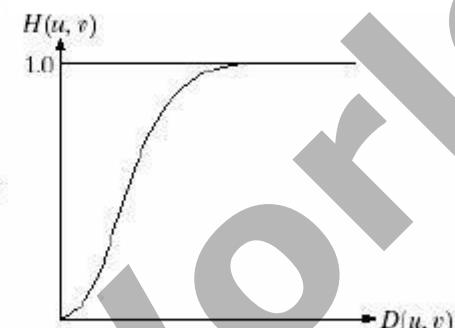
*D(u,v) is the distance from the origin
D₀ is the cutoff frequency.*



*Perspective plot
of BHPF*

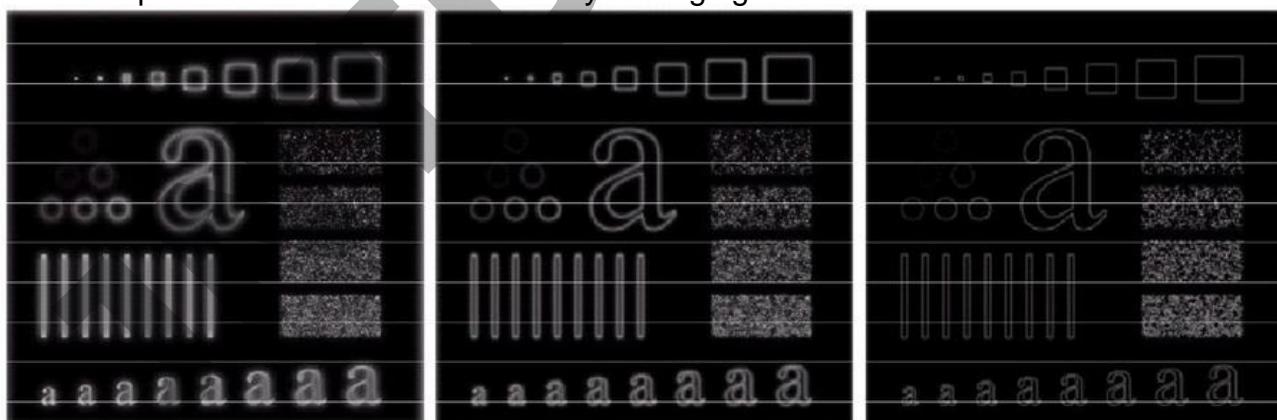


*Image representation
of BHPF*



*Cross section of
BHPF*

Gaussian High pass Filter (GHPF): Smoother results are obtained in GHPF when compared BHPF. There is absolutely no ringing artifacts



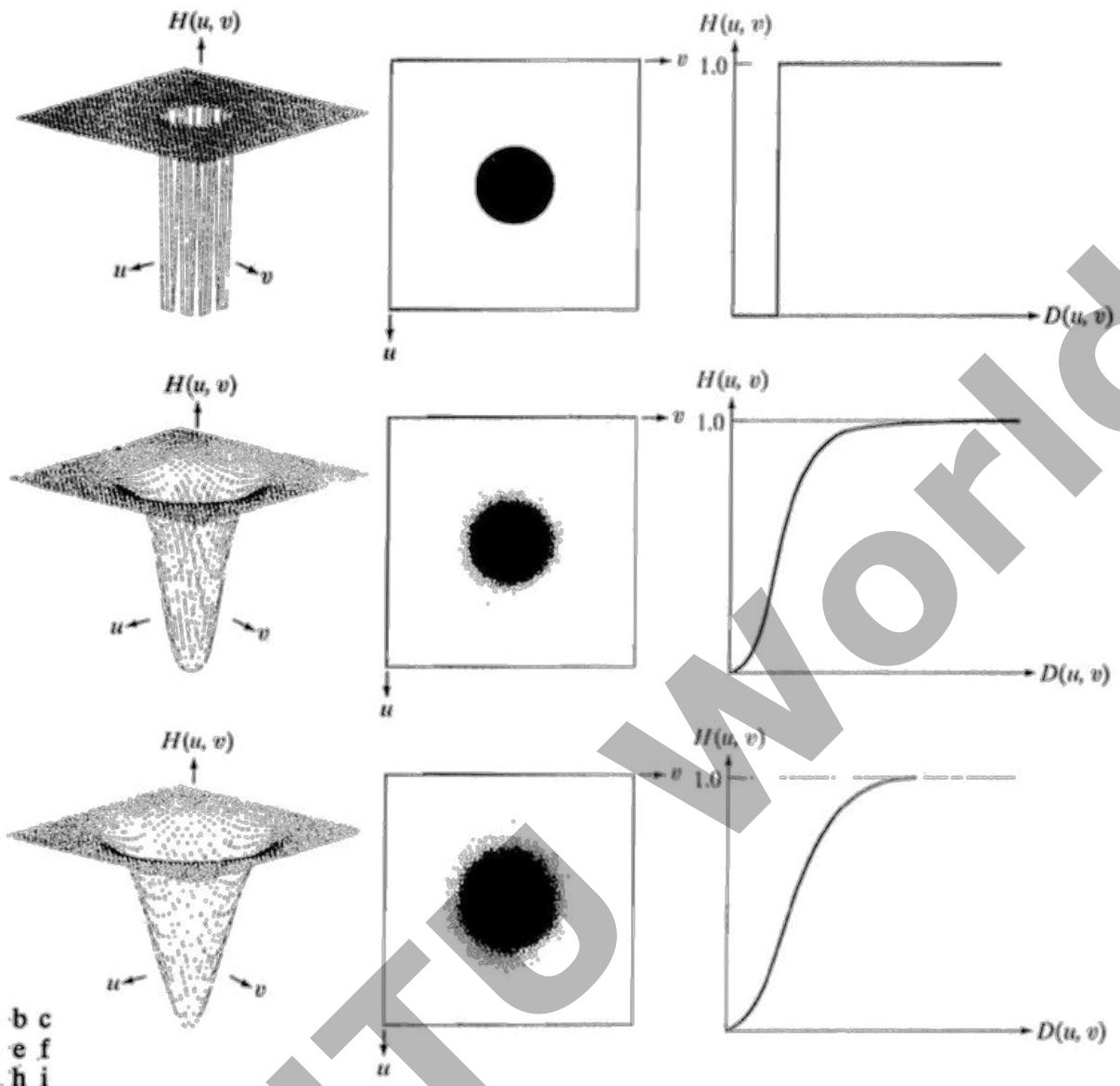


FIGURE .. : Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

The Laplacian in the Frequency Domain

$$H(u, v) = -4\pi^2(u^2 + v^2)$$

Or, with respect to the centre of the frequency rectangle:

$$H(u, v) = -4\pi^2 D^2(u, v)$$

The Laplacian image is obtained by:

$$\nabla^2 f(x, y) = IDFT[H(u, v)F(u, v)]$$

Enhancement of the image is achieved using ($H(u, v)$ negative):

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

The Laplacian in the Frequency Domain
 $\nabla^2 f(x, y) = IDFT[H(u, v)F(u, v)]$

- ⇒ Introduction of large scaling factors
- ⇒ Practical solution: normalize $f(x, y)$ to the range [0,1] before computing the DFT, and divide $\nabla^2 f(x, y)$ by its maximum value

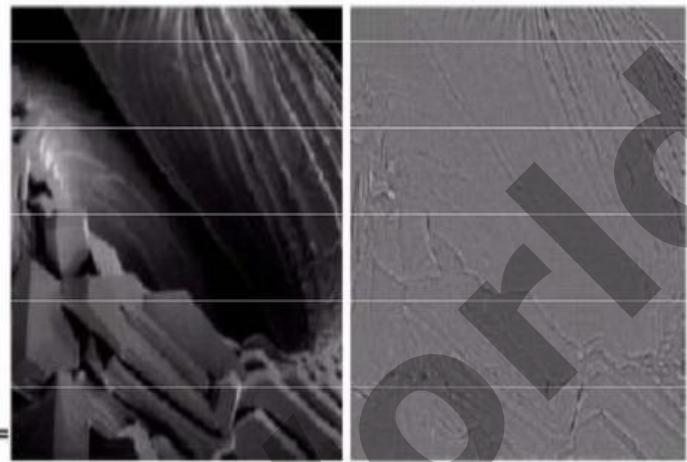
Example: using high pass filtering and thresholding for image enhancement

HIGH BOOST FILTERING

$$f_b(x, y) = f(x, y) - \bar{f}(x, y)$$

$$f_{lp}(x, y) = f(x, y) - f_{lp}(x, y)$$

$$H_{lp}(u, v) = 1 - H_{lp}(u, v)$$



$$f_{hb}(x, y) = Af(x, y) - f_{lp}(x, y)$$

$$\begin{aligned} f'_{lp}(x, y) &= (A-1)f(x, y) + f(x, y) - f_{lp}(x, y) \\ &= (A-1)f(x, y) + f_{lp}(x, y) \end{aligned}$$

$$H_{hb}(u, v) = (A-1) + H_{lp}(u, v)$$

$A=2$

$A=2.6$

Frequency Domain Analysis of Unsharp Masking and Highboost Filtering

Unsharp Masking: $g_{mask}(x, y)$ $f(x, y) f_{LP}(x, y)$

Highboost filtering:

(alternative definition) $g(x, y) (A-1) f(x, y)$
 $f_{HP}(x, y)$

$$f_{LP}(x, y) f(x, y) * h_{LP}(x, y) \mathbf{F}(f_{LP}(x, y)) F(u, v) H_{LP}(x, y)$$

All JNTU World

Band Filters

- ▶ Low pass filtering is useful for reducing noise but may produce an image that is overly blurry.
- ▶ High pass filtering is useful for sharpening edges but also accentuates image noise.
- ▶ Band filtering seeks to retain the benefits of these techniques while reducing their undesirable properties.
- ▶ Band filtering isolates the mid-range frequencies from both the low-range and high-range frequencies.
 - ▶ A band stop (or notch) filter attenuates mid-level frequencies while leaving the high and low frequencies unaltered.
 - ▶ A band pass filter is the inverse of a band stop; leaving the mid-level frequencies unaltered while attenuating the low and high frequencies in the image.
- ▶ A band of frequencies may be specified by giving the center frequency and the width of the band. The band width determines the range of frequencies that are included in the band.
- ▶ A band stop filter is essentially a combination of a low and high pass filter, which implies that ideal, Butterworth, and Gaussian band stop filters can be defined.

Homomorphic Filtering

Goals : simultaneous gray-level compression and contrast enhancement Separate the illumination-reflectance model

$$f(x, y) = i(x, y)r(x, y)$$

$$z(x, y) = \ln f(x, y) = \ln i(x, y) + \ln r(x, y)$$

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

process $Z(u, v)$ by $H(u, v)$

$$S(u, v) = H(u, v)Z(u, v) = H(u, v)F_i(u, v) + H(u, v)F_r(u, v)$$

$$s(x, y) = \mathcal{I}^{-1}\{S(u, v)\} = \mathcal{I}^{-1}\{H(u, v)F_i(u, v)\} + \mathcal{I}^{-1}\{H(u, v)F_r(u, v)\}$$

Let

$$i'(x, y) = \mathcal{I}^{-1}\{H(u, v)F_i(u, v)\}$$

$$r'(x, y) = \mathcal{I}^{-1}\{H(u, v)F_r(u, v)\}$$

$$s(x, y) = i'(x, y) + r'(x, y)$$

$$g(x, y) = e^{s(x, y)} = e^{i'(x, y)}e^{r'(x, y)} = i_0(x, y)r_0(x, y)$$

Separate the illumination and reflection models and then operate on these two components separately

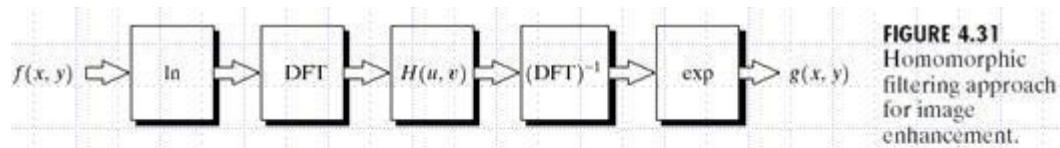


FIGURE 4.31
Homomorphic filtering approach for image enhancement.

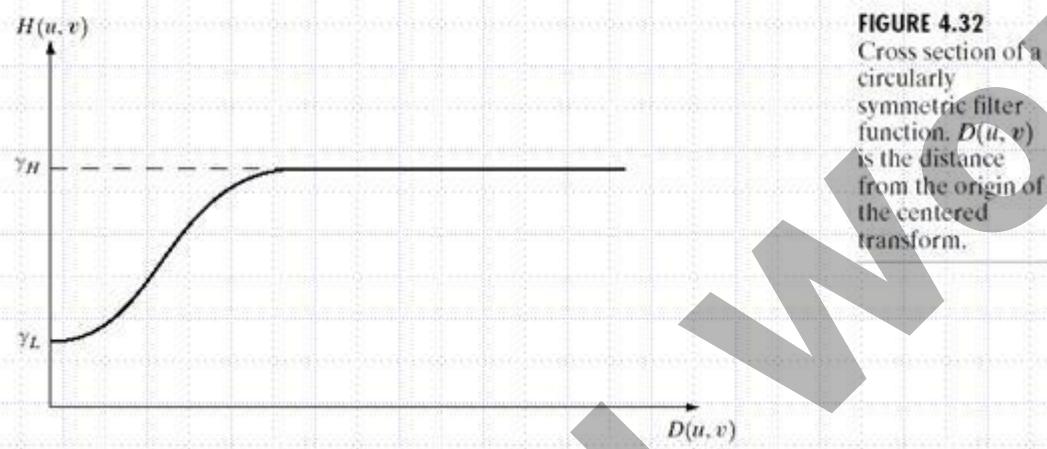


FIGURE 4.32
Cross section of a circularly symmetric filter function. $D(u, v)$ is the distance from the origin of the centered transform.

Specify $H(u, v)$ to emphasize the frequency part for enhancement

Specify $H(u, v)$ by using a modified Gaussian highpassfilter

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{-c(D^2(u, v)/D_0^2)}] + \gamma_L$$

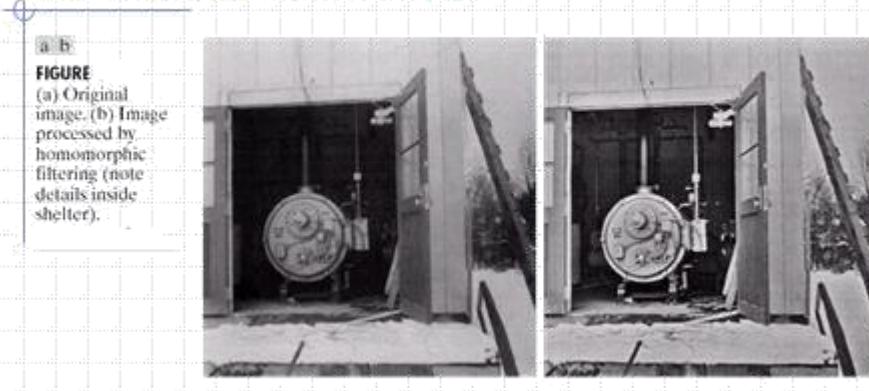
c for controlling the sharpness of the slope of the function

Illumination component : slow spatial component : vary abruptly, particularly at the junctions of dissimilar objects

Low frequency part : $F_i(u, v)$

High frequency part : $F_r(u, v)$

Homomorphic Filtering



Notch Filter

Consider the following filter transfer function:

$$H(u,v) = \begin{cases} 0 & \text{if } (u,v) = (M/2, N/2) \\ 1 & \text{otherwise} \end{cases}$$

This filter will set $F(0,0)$ to zero and leave all the other frequency components. Such a filter is called the notch filter, since it is constant function with a hole (notch) at the origin.

$F(0,0)$ is the average intensity of an image

$$F(0,0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

HOMOMORPHIC FILTERING

an image can be modeled mathematically in terms of illumination and reflectance as follow:

$$f(x,y) = I(x,y) r(x,y)$$

Note that:

$$F\{f(x,y)\} \neq F\{I(x,y)\} F\{r(x,y)\}$$

To accomplish separability, first map the model to natural log domain and then take the Fourier transform of it.

$$z(x,y) = \ln\{f(x,y)\} = \ln\{I(x,y)\} + \ln\{r(x,y)\}$$

Then,

$$F\{z(x,y)\} = F\{\ln I(x,y)\} + F\{\ln r(x,y)\}$$

or

$$Z(u,v) = I(u,v) + R(u,v)$$

Now, if we process $Z(u,v)$ by means of a filter function $H(u,v)$ then,

- Now, if we process $Z(u,v)$ by means of a filter function $H(u,v)$ then,

$$\begin{aligned} S(u,v) &= H(u,v)Z(u,v) = \\ &= H(u,v)I(u,v) + H(u,v)R(u,v) \end{aligned}$$

- Taking inverse Fourier transform of $S(u,v)$ brings the result back into natural log domain,

$$\begin{aligned} s(x,y) &= F^{-1}\{S(u,v)\} \\ &= F^{-1}\{H(u,v)I(u,v)\} + F^{-1}\{H(u,v)R(u,v)\} \end{aligned}$$

By letting

$$i'(x,y) = F^{-1}\{H(u,v)I(u,v)\}$$

$$r'(x,y) = F^{-1}\{H(u,v)R(u,v)\}$$

- Now, to get back to spatial domain, we need to get inverse transform of natural log, which is exponential,

$$\begin{aligned}
 s(x, y) &= i'(x, y) + r'(x, y) \\
 g(x, y) &= \exp[s(x, y)] \\
 &= \exp[i'(x, y)] \cdot \exp[r'(x, y)] \\
 &= i_o(x, y)r_o(x, y)
 \end{aligned}$$

Where $i_o(x, y)$ is illumination and $r_o(x, y)$ is reflectance components of the output image.

- This method is based on a special case of a class of systems known as *homomorphic systems*.

- The overall model in block diagram will look as follow:



For homomorphic filter to be effective it needs to affect the low- and high-frequency components of the Fourier transform in different way.

- To compress the dynamic range of an image, the low frequency components ought to be attenuated to some degree.
- On the other hand, to enhance the contrast, the high frequency components of the Fourier transform ought to be magnified.

Lecture Notes

UNIT 4:

Image Restoration

Degradation model

Algebraic approach to restoration

Inverse filtering, least mean square filters.

Constrained Least square restoration, Interactive restoration

Restoration techniques involve **modeling of degradation** and applying the **inverse process** in order to recover the image

Degradation gray value altered

Distortion pixel shifted (**Geometric restoration (image registration)**)

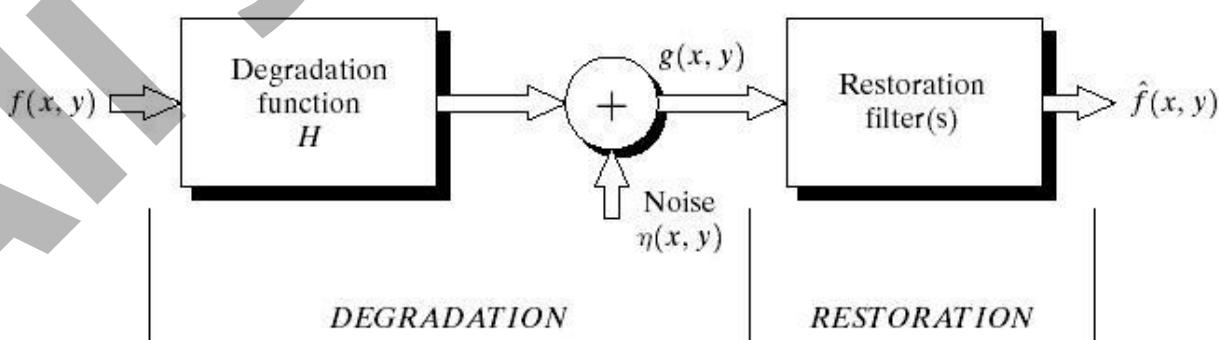
So, in case of image restoration, the image degradation model is very very important. So, we have to find out what is the phenomena or what is the model which has degraded the image and once that degradation model is known; then we have to apply the inverse process to recover or restore the desired image.

But conventionally, this kind of simple filtering is not known as restoration.

In Restoration, if we know a degradation model by which the image has been degraded and on that degradation model, on the degraded image, some noise has been added.

Concept of Image Restoration

Image restoration is to restore a degraded image back to the original image while image enhancement is to manipulate the image so that it is suitable for a specific application.



Use a priori knowledge of the degradation
Modeling the degradation and apply the inverse process

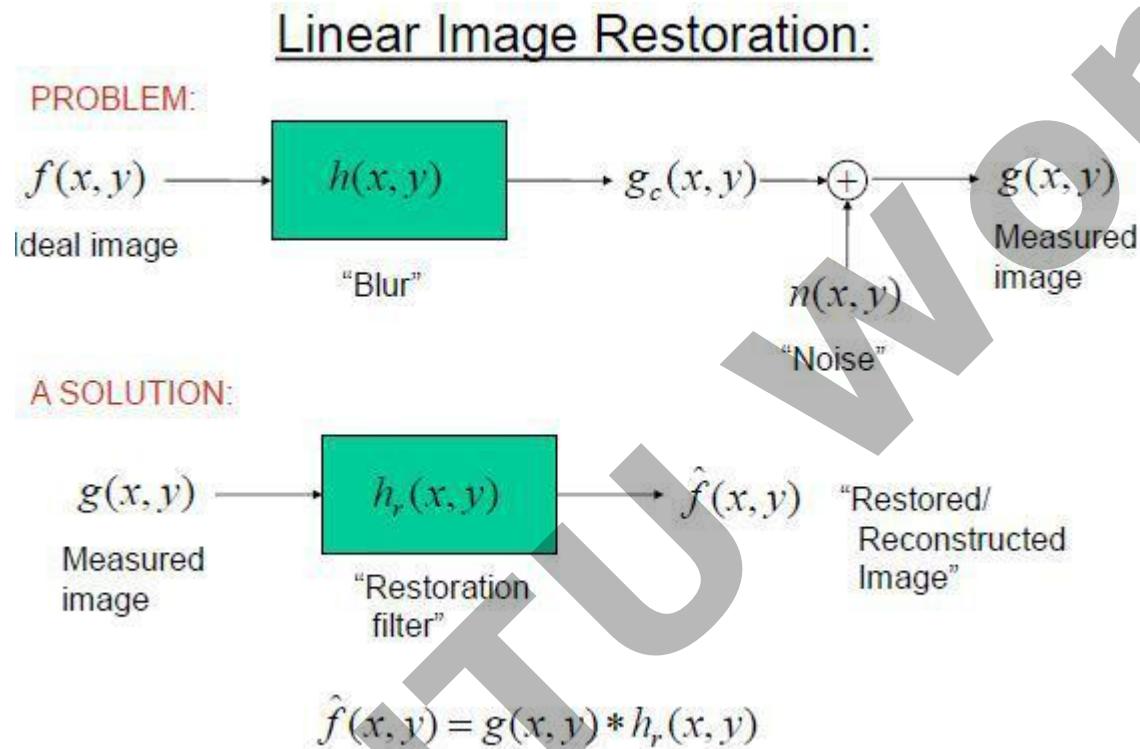


Image Degradation Model: *Spatial domain representation can be modeled by:*

Degradation model:

where $h(x,y)$ is a system that causes image distortion

$$g(x, y) = f(x, y) \cdot h(x, y) \quad (x, y)$$

and $n(x,y)$ is noise.

Frequency domain representation can be modeled by:

$$G(u, v) = H(u, v)F(u, v) + N(u, v)$$

A degradation function and additive noise that operate on an input image $f(x, y)$ to produce a degraded image $g(x, y)$:

We will assume that a degradation function exists, which, together with additive noise, operates on the input image $f(x, y)$ to produce a degraded image $g(x, y)$.

The objective of restoration is to obtain an estimate for the original image from its degraded version $g(x, y)$ while having some knowledge about the degradation function H and the noise $\eta(x, y)$.

The third expression is in matrix form.

$$g = Hf + \eta$$

g is a column matrix or column vector of dimension $m \times n$ the image is of dimension $m \times n$. f is also column vector of the same dimension $m \times n$.

This degradation matrix H , is of dimension $mn \times mn$,

Knowledge of the degradation function is essential because that is what is our restoration problem. That is we try to restore or recover the original image using acquired knowledge of the degradation function.

Model for image degradation/restoration process

We will assume that a degradation function exists, which, together with additive noise, operates on the input image $f(x,y)$ to produce a degraded image $g(x,y)$.

The objective of restoration is to obtain an estimate for the original image from its degraded version $g(x,y)$ while having some knowledge about the degradation function H and additive noise $\eta(x,y)$.

The key is finding an appropriate model of the image degradation that can be inverted

- Additive noise
 - $g(x, y) = f(x, y) + \eta(x, y)$
- Linear blurring
 - $g(x, y) = f(x, y) * h(x, y)$
- Linear blurring and additive noise
 - $g(x, y) = f(x, y) * h(x, y) + \eta(x, y)$
- Linear blurring and complex noise
 - $g(x, y) = [f(x, y) * h(x, y)][1 + m(x, y)] + n(x, y)$

If degradation H is a linear, position-invariant process, the degraded image in **spatial domain** is

$$g(x, y) = h(x, y) \underset{\text{convolution}}{\otimes} f(x, y) + \eta(x, y)$$

Therefore, in the **frequency domain** it is

$$G(u, v) = H(u, v)F(u, v) + N(u, v)$$

Given $g(x, y)$ and some knowledge about the degradation function H and the noise η , obtain an estimate $\hat{f}(x, y)$ of the original image

$h(x, y)$: spatial representation of the degradation function

Different restoration approaches

- ★ Frequency domain
 - Inverse filter
 - Wiener (minimum mean square error) filter
- ★ Algebraic approaches

- Unconstrained optimization
- Constrained optimization
- The regularization theory

Typical image noise models are

- Gaussian: poor illumination**
- Impulse: faulty switch during imaging**, Salt-and-Pepper
- Rayleigh: range image**
- Gamma/Exp: laser imaging**
- Uniform is least used.**

Parameters can be estimated based on histogram on small flat area of an image

Estimation of noise parameters

mean and variance of intensity levels:

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_s(z_i)$$

$$\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_s(z_i)$$

If the shape of pdf is Gaussian, the mean and variance completely specify it. Otherwise, these estimates are needed to derive a and b .

For impulse noise, we need to estimate actual probabilities P_a and P_b from the histogram.

Sources of Noise Three major sources:

- During acquisition (often random) (faulty CCD elements)
- During transmission (channel interference)
- During image processing (compression)

What causes the image to blur?

- Camera: translation, shake, out-of-focus
- Environment: scattered and reflected light
- Device noise: CCD/CMOS sensor and circuitry
- Quantization noise

Restoration in the presence of noise only – Spatial filtering

Spatial Filtering

- Method of choice when only additive noise is present
 - i.e. Gaussian or Impulse
- Mean Filter
- Median Filter

When the only degradation in the image is noise:

$$g(x, y) = f(x, y) + \eta(x, y)$$

$$G(u, v) = F(u, v) + N(u, v)$$

the noise terms are unknown, so, subtracting them from $g(x,y)$ or $G(u,v)$ is not a realistic option.

Therefore, spatial filtering is a good candidate when only additive random noise is present.

We already discussed spatial filters before; here we discuss noise rejection capabilities of different spatial filters.

- 1. Mean filters** (Arithmetic mean filter, Geometric mean filter, Harmonic mean filter, Contraharmonic mean filter)
- 2. Order-statistic filters** (Max and Min filters, Midpoint filter, Alpha-trimmed mean filter, Adaptive local noise reduction filter,)
- 3. Adaptive filters** (Adaptive local noise reduction filter, Adaptive median filter)

NOISE

Classification of noise is based upon:

- the shape of probability density function (analog case of noise)
- Histogram (discrete case of noise)

Uncorrelated noise is defined as the variations within an image that have no spatial dependences from image to image

How can we determine which noise model is appropriate for our situation?

- Capture image of uniform object – Calculate intensity histogram
- Compare to noise models to select the most similar looking
- Perform least mean square fit to obtain specific model parameters

What is the best way to remove noise?

- Get more data !

Capture N images of the same scene $g_i(x,y) = f(x,y) + n_i(x,y)$
– Average to obtain new image
 $g_{ave}(x,y) = f(x,y) + n_{ave}(x,y)$

What is the second best way to remove noise?

- **Use all information available !**

- Derive noise model for the input image
- Generate a synthetic image with the same noise distribution as the input image
- Perform experiments on the synthetic image to select the noise removal method and parameters that minimize restoration error
- Apply this method to the input image

the term noise has the following meanings:

1. An undesired disturbance within the frequency band of interest; the summation of unwanted or disturbing energy introduced into a communications system from man-made and natural sources.
2. A disturbance that affects a signal and that may distort the information carried by the signal.
3. Random variations of one or more characteristics of any entity such as voltage, current, or data.
4. A random signal of known statistical properties of amplitude, distribution, and spectral density.

Noise has long been studied. People analyze its property, type, influence and what can be done about it.

Most of the research is done in mathematics and close related to Probability Theory

Types of mixing noise with signal

In many applications it is assumed that noise is **additive** and statistically **independent of the signal**

$$g(x, y) = f(x, y) + \eta(x, y)$$

For thermal noise Often, noise is **signal-dependent**. Examples: **speckle**, **photon noise**,...
Many noise sources can be modeled by a **multiplicative model**:

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

In CMOS sensors there is a **fixed-pattern noise** and mixture of **additive** and **multiplicative** noise

The major sources of noise during image acquisition from electro-optics devices are photon noise, thermal noise, on-chip electronic noise, amplifier noise, and quantization noise.

Estimation of Noise

Consists of finding an image (or subimage) that contains only noise, and then using its histogram for the noise model

- Noise only images can be acquired by aiming the imaging device (e.g. camera) at a blank wall

In case we cannot find "noise-only" images, a portion of the image is selected that has a known histogram, and that knowledge is used to determine the noise characteristics

- After a portion of the image is selected, we subtract the known values from the histogram, and what is left is our noise model
- To develop a valid model many sub-images need to be evaluated

There are different noise models.

For good strategy in removing noise and restoring image quality one needs to determine noise distribution.

(Check the histogram in a reasonable size smooth region with visibly small variation in values)

Once the noise model is estimated use an appropriate filter.

Histogram in the same region indicates level of success.

Denoising (i.e. removing noise) often introduce other side effects.

Advanced de-noising filters are based on adaptive strategy, i.e. the procedure tries to adapt the application of the filter as it progresses

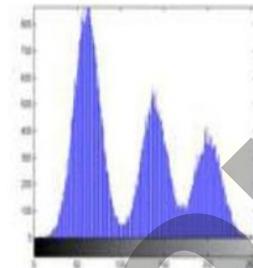
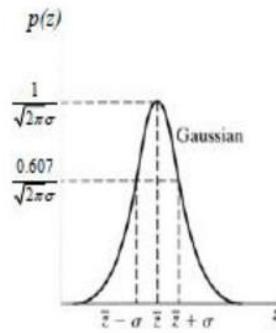
Frequency domain filters provide powerful de-noising methods.

Noise in Colour images may have different characteristics in different colour channels, but removing noise uses the same strategy

Noise pdfs

1. Gaussian (normal) noise is very attractive from a mathematical point of view since its DFT is another Gaussian process.

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}}$$



Here z represents intensity, \bar{z} is the mean (average) value of z and σ is its standard deviation. σ^2 is the variance of z .

Electronic circuit noise, sensor noise due to low illumination or high temperature.

2. Rayleigh noise is specified as

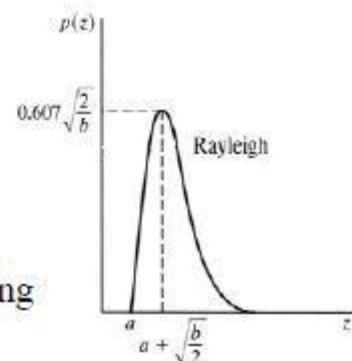
$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-\frac{(z-a)^2}{b}} & z \geq a \\ 0 & z < a \end{cases}$$

The mean and variance are given by

$$\bar{z} = a + \sqrt{\pi b / 4}$$

$$\sigma^2 = \frac{b(4 - \pi)}{4}$$

The Rayleigh density is useful for approximating skewed histograms. Used in range imaging.



Radar range and velocity images typically contain noise that can be modeled by the Rayleigh distribution

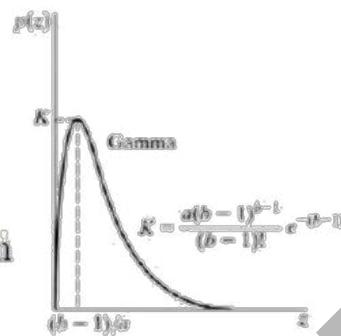
3. Erlang noise is specified as

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Here $a > 0$ and b is a positive integer. The mean and variance are given by

$$\bar{z} = b/a$$

$$\sigma^2 = b/a^2$$



When the denominator is the gamma function, the pdf describes the gamma distribution.

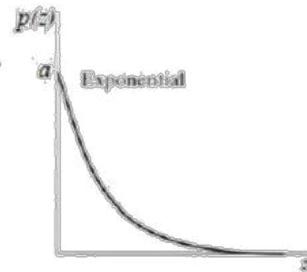
4. Exponential noise is specified as

$$p(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Here $a > 0$. The mean and variance are given by

$$\bar{z} = 1/a$$

$$\sigma^2 = 1/a^2$$

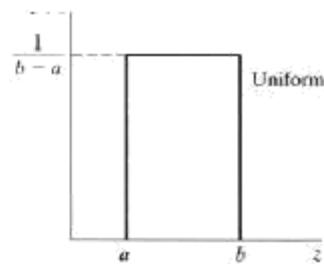


Exponential pdf is a special case of Erlang pdf with $b=1$.

Used in laser imaging.

5. Uniform noise is specified as

$$\text{Histogram Uniform} = \begin{cases} \frac{1}{b-a} & a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$



The mean and variance are given by

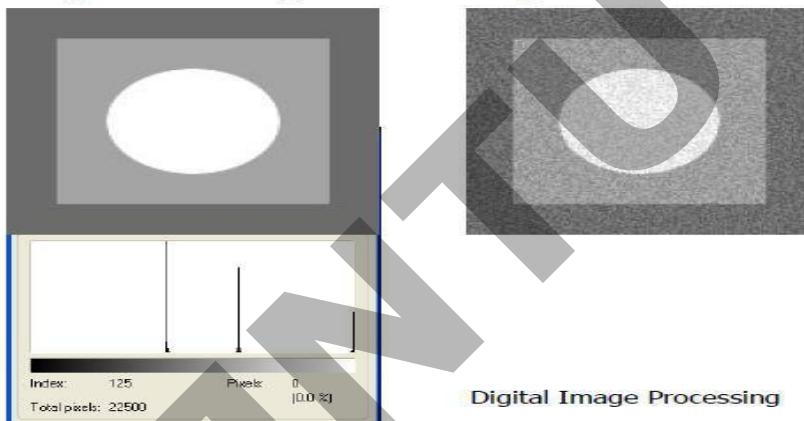
$$\bar{z} = \frac{a+b}{2}$$

$$\sigma^2 = \frac{(b-a)^2}{12}$$

The gray level values of the noise are evenly distributed across a specific range

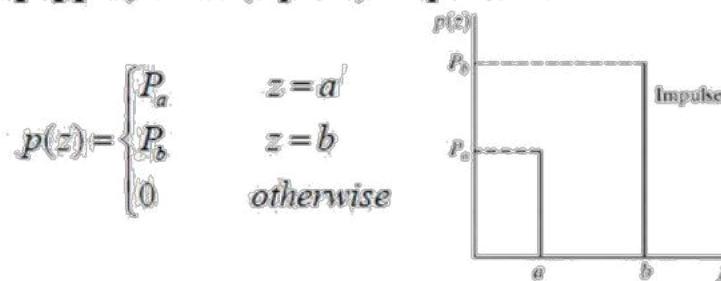
- Quantization noise has an approximately uniform distribution

Original image: **image disturbed by uniform noise:**



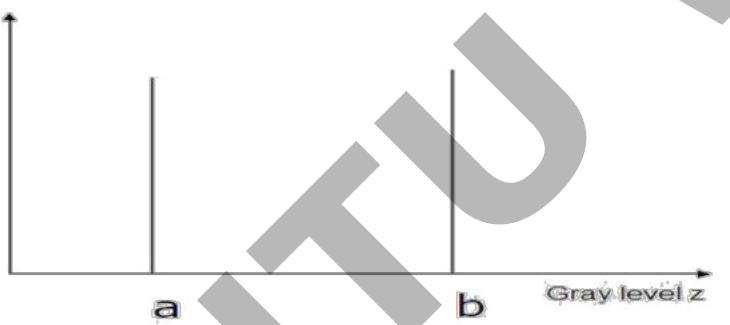
Digital Image Processing

6. Impulse (salt-and-pepper) noise (bipolar) is specified as



If $b > a$, intensity b will appear as a light dot on the image and a appears as a dark dot. If either P_a or P_b is zero, the noise is called unipolar. Frequently, a and b are *saturated* values, resulting in positive impulses being white and negative impulses being black. This noise shows up when quick transitions – such as faulty switching – take place.

Histogram

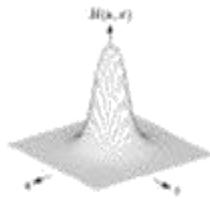


The salt-and-pepper type noise (also called impulse noise, shot noise or spike noise) is typically caused by malfunctioning pixel elements in the camera sensors, faulty memory locations, or timing errors in the digitization process. There are only two possible values, a and b , and the probability of each is typically less than 0.2 – with numbers greater than this the noise will swamp out the image. • For an 8-bit image, the typical value for pepper noise is 0, and 255 for salt-noise

Recovering from Periodic Noise

Recall: Butterworth LPF

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$$



Butterworth bandreject filter

$$H(u, v) = \frac{1}{1 + [\frac{D(u, v)W}{D^2(u, v) - D_0^2}]^{2n}}$$

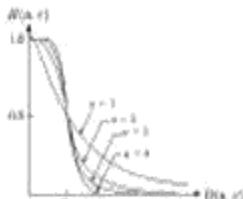


FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4

Reduction of periodic noise by frequency domain filters

The main idea: periodic noise appears as concentrated bursts of energy in frequency domain at the locations corresponding to frequencies of the periodic interference.

Restoration in the presence of Noise: Periodic Noise Removal by Frequency Domain Filtering:

- *Bandreject, bandpass and notch filters can be used for periodic noise removal.*
- *Bandreject filters remove/attenuate a band of frequencies about the origin of the Fourier transform.*

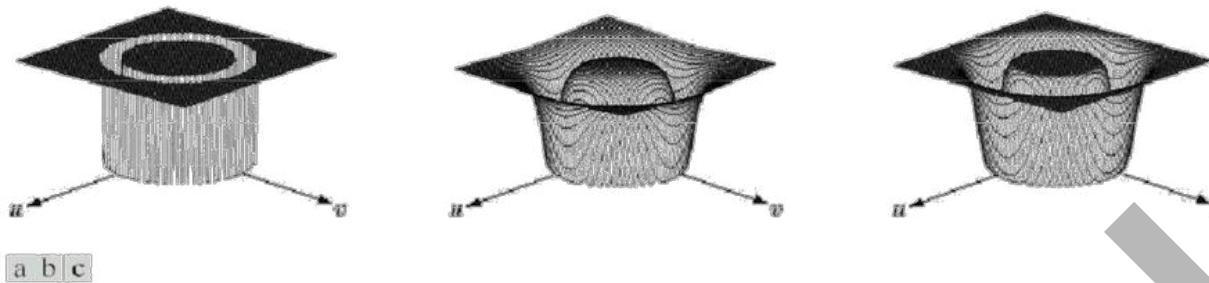


FIGURE 5.15 From left to right, perspective plots of ideal, Butterworth (of order 1), and Gaussian bandreject filters.

Restoration in the presence of Noise: Periodic Noise Removal by Frequency Domain Filtering:

- An Ideal Bandreject filter is given by:*

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u,v) \geq D_0 + \frac{W}{2} \end{cases}$$

- W is the width of the band*
- D₀ is the radial center*
- D(u,v) distance from the origin.*

•*Butterworth Bandreject filter is given by:*

$$H(u,v) = \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - D_0^2} \right]^{2n}}$$

- W is the width of the band*
- D₀ is the radial center*
- D(u,v) distance from the origin.*
- n is the order of the filter*

•*Gaussian Bandreject filter is given by:*

$$H(u,v) = 1 - e^{-\frac{1}{2} \left[\frac{D^2(u,v) - D_0^2}{D(u,v)W} \right]^2}$$

- W is the width of the band*
- D₀ is the radial center*
- D(u,v) distance from the origin.*

Periodic Noise Removal by Frequency Domain Filtering:

•*Bandpass filters perform the opposite function of the bandreject filters and the filter transfer function of a bandpass filter is given by:*

$$H(u,v)_{bp} = 1 - H(u,v)_{br}$$

Notch filter:

Periodic noise in images is typically caused by electrical and/or mechanical systems, such as mechanical jitter (vibration) or electrical interference in the system during image acquisition

- It appears in the frequency domain as impulses corresponding to sinusoidal Interference

• It can be removed with band reject and notch filters Periodic Noise

Frequency Domain Filtering

- Band reject Filter

Notch Filter

Adaptive Median Filter:

- Effects of coding/decoding

Image noise as a random variable:

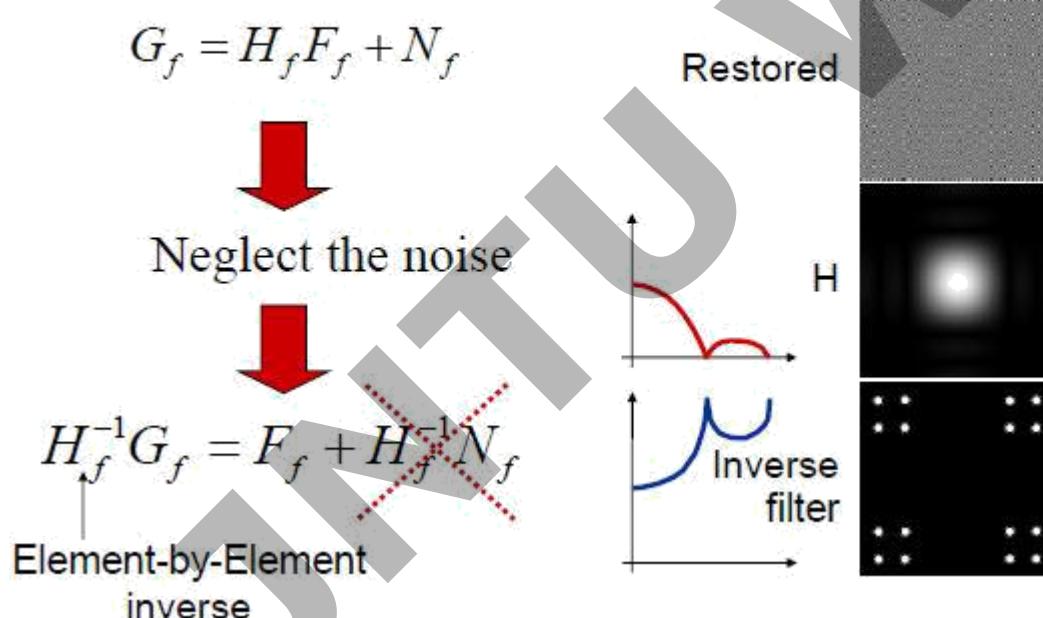
- For every (x, y) , $\eta(x, y)$ can be considered as a random variable.

In general, we assume that the noise $\eta(x, y)$ is not dependent on the underlying signal $f(x, y)$.

- In general, we assume that the value of $\eta(x, y)$ is not correlated with $\eta(x', y')$.

(Spatially uncorrelated noise)

Basic Inversion Idea



Assumptions

Noise

- Independent of spatial location
- Exception: periodic noise ...
- Uncorrelated with image

Degradation function H

- Linear
- Position-invariant

Degradation Operator $H(x,y)$ Definition

$$g(x, y) = H[f(x, y)] + \eta(x, y)$$

$$\eta(x, y) = 0$$

$$g(x, y) = H[f(x, y)] \text{ is Degradation Operator}$$

what we have is we have a degraded image $g(x, y)$ which now let us represent it is like this H of $f(x, y)$ plus $\eta(x, y)$ where in this particular case, we assume that this H is the degradation operator which operates on the input image $f(x, y)$ and that when added with the additive noise $\eta(x, y)$ gives us the degraded image $g(x, y)$.

g The blurred image

H The distortion operator, also called the point spread function (PSF). The distortion operator, when convolved with the image, creates the distortion.

f The original true image

η Additive noise, introduced

This degradation operator H is a linear operator.

$$f_1(x, y) \quad f_2(x, y)$$

$$H[k_1 f_1(x, y) + k_2 f_2(x, y)]$$

$$= k_1 H[f_1(x, y)] + k_2 H[f_2(x, y)] \longleftrightarrow \text{Linear Operator}$$

Superposition theorem or Additivity

$$\begin{aligned}
 \text{If } k_1 = k_2 , \quad k_1 H [f_1 (x,y)] + k_2 H [f_2 (x,y)] &= \quad H [f_1 (x,y)] + \quad H [f_2 (x,y)] \\
 &= H [f_1 (x,y) + \quad f_2 (x,y)]
 \end{aligned}$$

Position Invariant

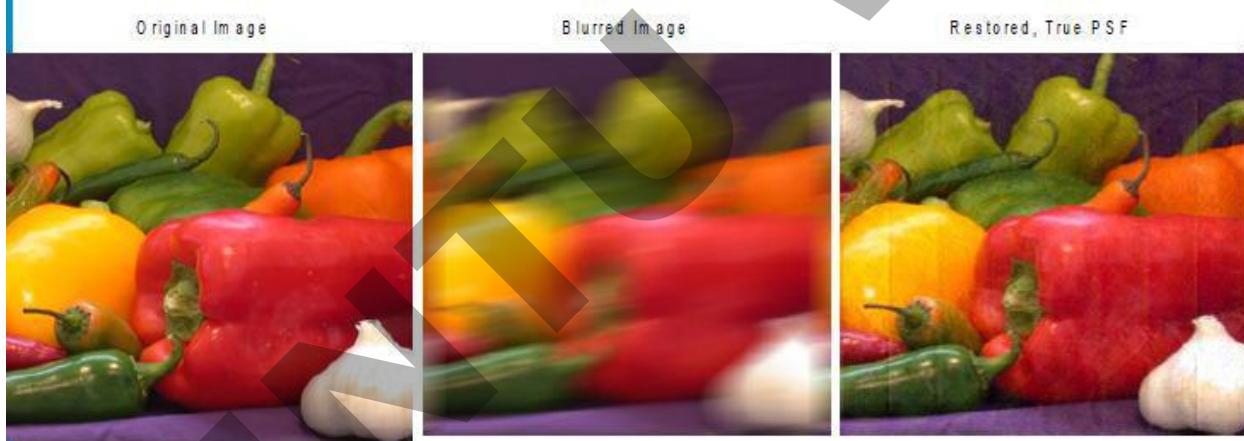
$$H[f (x-\alpha, y-\beta)] = g (x-\alpha, y-\beta)$$

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

Image Deblurring

Based on this model, the fundamental task of deblurring is to deconvolve the blurred image with the PSF that exactly describes the distortion. Deconvolution is the process of reversing the effect of convolution.

Note! The quality of the deblurred image is mainly determined by knowledge of the PSF.



Estimating the Degradation model or the degradation function :

Degradation model:

$$g(x, y) = f(x, y) * h(x, y) + (x, y) \quad \text{Or} \quad G(u,v) = F(u,v)H(u,v) + N(u,v)$$

Purpose: to estimate $h(x,y)$ or $H(u,v)$

Why?

If we know exactly $h(x,y)$, regardless of noise, we can do deconvolution to get $f(x,y)$ back from $g(x,y)$.

This is why convolution is sometimes also known as *the superposition integral* and deconvolution is sometimes also called *signal separation*

The process of image restoration by use of the estimated degradation function is sometimes called blind deconvolution

Blind Deconvolution Problem

$$\text{Observed image} = \text{Unknown true image} + \text{Unknown point spread function} + \text{Unknown noise}$$
$$u_{obs} = u_{orig} + k + n$$

Goal: Given u_{obs} , recover both u_{orig} and k

There are 3 principal methods of estimating the degradation function for Image Restoration: (Blind convolution: because the restored image will be only an estimation.)::

1.Observation, 2) Experimentation, 3) Mathematical modeling

what we have is the degraded image $g(x, y)$ and by looking at this degraded image $g(x, y)$, we have to estimate what is the degradation function involved.

The degradation function H can be estimated by visually looking into a small section of the image containing simple structures, with strong signal contents, like part of an object and the background.

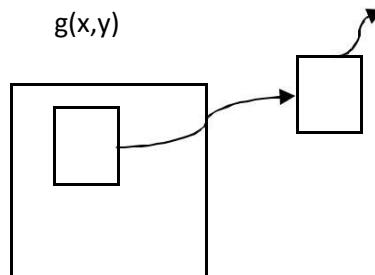
Given a small sub image $g_s(x, y)$, we can manually (i.e. filtering) remove the degradation in that region (by observing the gray levels) with an estimated sub image $f(x, y)$, and assuming that the additive noise is negligible in such an area with a strong signal content. To reduce the effects of noise, we look for an area of strong signal (area of high contrast) and try to process that sub image to un-blur it as much as possible (for instance, by sharpening sub image with a sharpening filter).

Let denote the original sub image by $g_s(x, y)$ and its “restored” version by $f'(x, y)$
we can assume

$$g_s(x, y) \longleftrightarrow G_s(u, v)$$

$$f_s(x, y) \longleftrightarrow F_s(u, v)$$

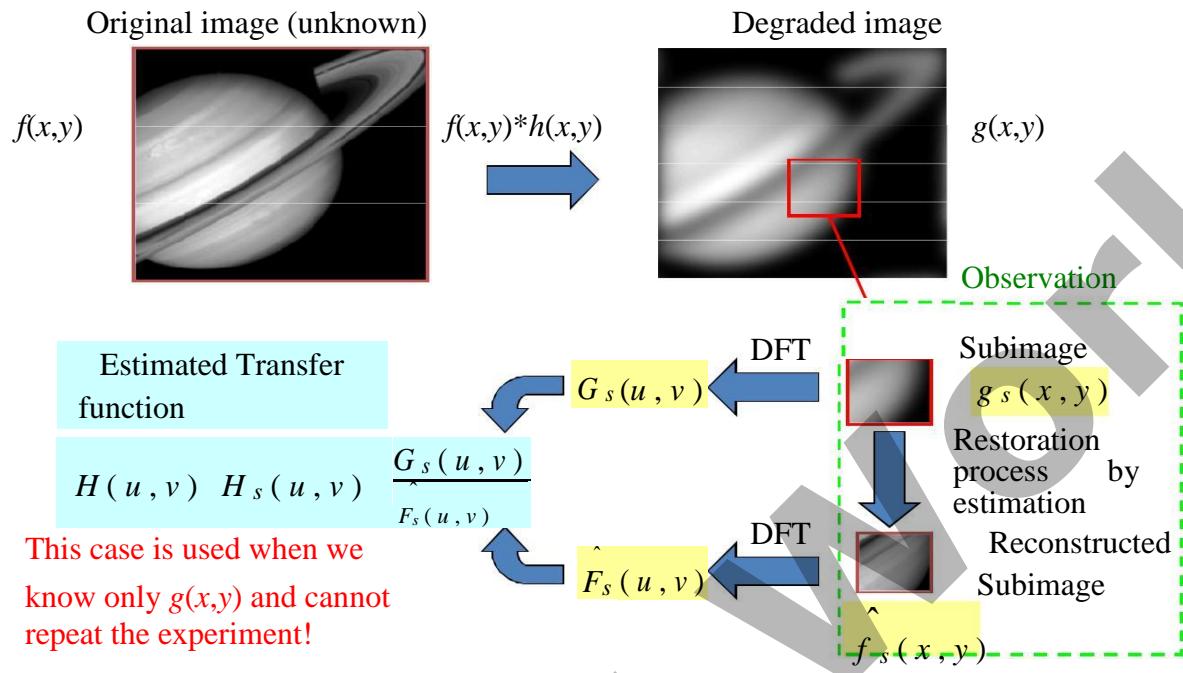
$$H_s(u, v) = \frac{G_s(u, v)}{\hat{F}_s(u, v)}$$



Having $H_s(u, v)$ estimated for such a small sub image, the shape of this degradation function can be used to get an estimation of $H(u, v)$ for the entire image.

Since noise term is neglected, the sub image considered should be a strong image portion with less noise content.

Estimation by Image Observation



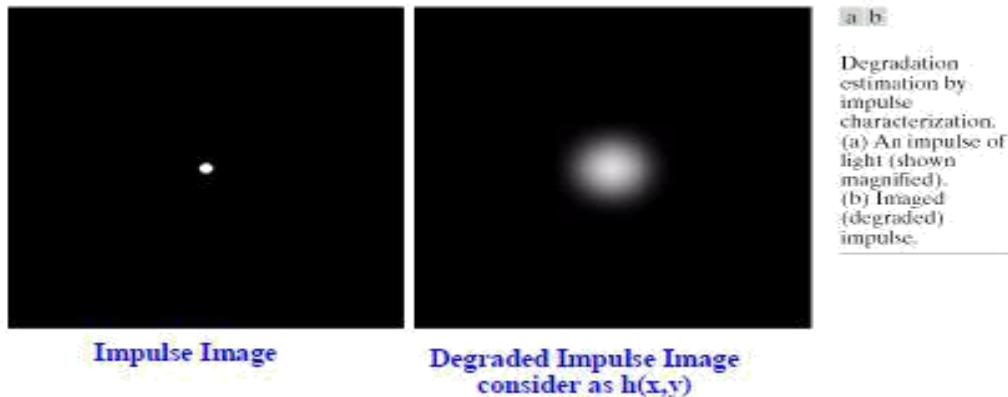
Estimation by Image Experimentation:

So, what we do in case of this experimentation? Here, we try to get an imaging setup which is similar to the imaging setup using which the degraded image has been obtained.

So here, our purpose will be to find out the point spread function or the impulse response of this imaging setup. It is the impulse response which fully characterizes any particular system. So, once the impulse response is known, the response of that system to any arbitrary input can be computed from the impulse response.

So here, the first operation that we have to do is we have to simulate an impulse. So, first requirement is **impulse simulation**.

Now, how do you simulate an impulse? An impulse can be simulated by a very bright spot of light and because our imaging setup is a camera, so we will have a bright spot as small as possible of light falling on the camera and this bright spot if it is very small, Then it is equivalent to an impulse and using this bright spot of light as an input, whatever image that we get that is the response to that bright spot of light which in our case is an impulse.



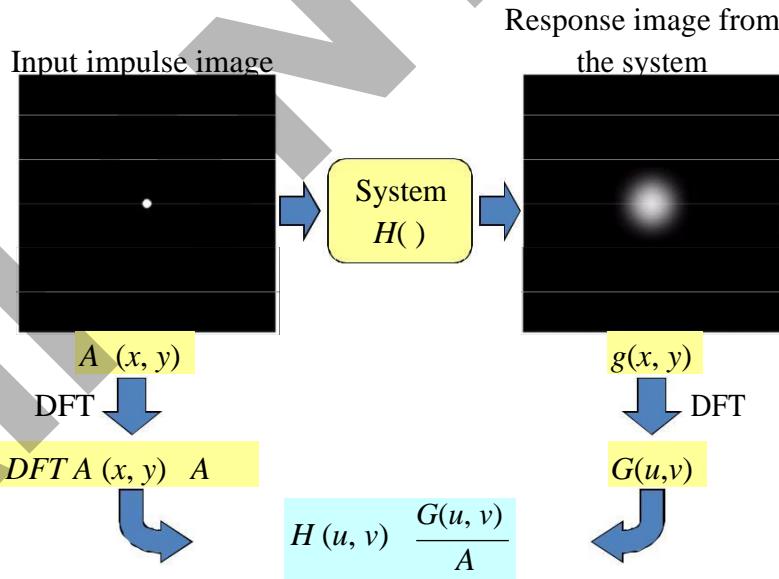
So, that is what has been shown in this particular slide. The left most image is the simulated impulse. Here you find that at the center, we have a bright spot of light. Of course, this spot is shown in a magnified form, in reality this spot will be even smaller than this and on the right hand side, the image that you have got, this is the image which is captured by the camera when this impulse falls on this camera lens.

So, this is my impulse, simulated impulse and this is what is my impulse response. So, once I have the impulse and this impulse response; then from this, I can find out what is the degradation function of this imaging system.

Now, we know from our earlier discussion that for a very very narrow impulse, the **Fourier transformation of an impulse is a constant**.

Estimation by Experiment

Used when we have the same equipment set up and can repeat the experiment.



If we have the acquisition device producing degradation on images, we can use the same device to obtain an accurate estimation of the degradation.

So, in this case, this $G(u, v)$ is the Fourier transform of the observed image and here, this Fourier transform is nothing but the Fourier transform of the image that we have got which is response to the simulated impulse that has fallen on the camera. A is the Fourier transform of the impulse falling on the lens and the ratio of these 2 that is $G(u, v)$ by this constant A that gives us what is the deformation or what is the degradation model of this particular imaging setup

- This can be achieved by applying an impulse (bright dot) as an input image . The Fourier transform of an impulse is constant, A, therefore***

$$H(u, v) = \frac{G(u, v)}{A}$$

Where, A is a constant describing the strength of the impulse. Note that the effect of noise on an impulse is negligible.

Simply take the Fourier transform of the degraded image and after normalization by a constant A , use it as the estimate of the degradation function $H(u, v)$.

So, I get the degradation function and the same degradation function, we assume that it is also valid for the actual imaging system. Now, in this point, regarding this, one point should be kept in mind that the intensity of the light which is the simulated impulse should be very very high so that the effect of noise is reduced.

•Estimation by Mathematical Modeling:

Sometimes the environmental conditions that causes the degradation can be modeled by mathematical formulation.

For example the atmospheric turbulence can be modeled by:

This equation is similar to Gaussian LPF and would produce blurring in the image according to the values of k. For example if k=0.0025, the model represents severe turbulence, if k=0.001, the model represents mild turbulence and if k=0.00025, the model represents low turbulence.

$$H(u, v) = e^{-k(u^2 + v^2)^{5/6}}$$

k is a constant that depends on the nature of the Turbulence

Once a reliable mathematical model is formed the effect of the degradation can be obtained easily.

If the value of K is large, that means the turbulence is very strong whereas if the value of K is very low, it says that the turbulence is not that strong

So, this is the one which gives you modeling of degradation which occurs because of turbulence. Now, there are other approaches of degradation mathematical model to estimate the degradation which are obtained by fundamental principles. So, from the basic principles also, we can obtain what should be the degradation function.

Possible classification of restoration methods

Restoration methods could be classified as follows:

deterministic: we work with sample by sample processing of the observed (degraded) image

stochastic: we work with the statistics of the images involved in the process

non-blind: the degradation process H is known **blind:**

the degradation process H is unknown

semi-blind: the degradation process H could be considered partly known

From the viewpoint of implementation:

direct

iterative

recursive

Inverse Filtering: (un constrained)

- * In most images, adjacent pixels are highly correlated, while the gray levels of widely separated pixels are only loosely correlated.
- * Therefore, the autocorrelation function of typical images generally decreases away from the origin.
- * Power spectrum of an image is the Fourier transform of its autocorrelation function, therefore, we can argue that the power spectrum of an image generally decreases with frequency
- * Typical noise sources have either a flat power spectrum or one that decreases with frequency more slowly than typical image power spectra.
- * Therefore, the expected situation is for the signal to dominate the spectrum at low frequencies while the noise dominates at high frequencies.

Until now our focus was the calculation of degradation function $H(u,v)$. Having $H(u,v)$ calculated/estimated the next step is the restoration of the degraded image.

There are different types of filtering techniques for obtaining or for restoring the original image from a degraded image.

The simplest kind of Approach to restoration is direct inverse filtering technique.

The simplest way of image restoration is by using Inverse filtering:

Now, the concept of inverse filtering is very simple. Our expression is that $G(u, v)$ that is the Fourier transform of the degraded image is given by $H(u, v)$ into $F'(u, v)$ where $H(u, v)$ is the degradation function in the frequency domain and estimate $F'(u, v)$ is the Fourier transform of the original image, $G(u, v)$ is the Fourier transform of the degraded image. The division is an array operation.

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)}$$

, $\hat{F}(u, v)$ is the Fourier transform of the restored image

$$\hat{F}(u, v) = F(u, v) + \frac{N(u, v)}{H(u, v)}$$

Unknown random function
 Must not be very small. Otherwise the noise dominates

Noise is enhanced when $H(u, v)$ is small.

To avoid the side effect of enhancing noise, we can apply this formulation to freq. component (u, v) with in a radius D_0 from the center of $H(u, v)$.

So, this expression says that even if $H(u, v)$ is known exactly, the perfect reconstruction may not be possible because $N(u, v)$ is not known.

Again if $H(u, v)$ is near zero, $N(u, v)/H(u, v)$ will dominate the $F'(u, v)$ estimate.

Now, because this $H(u, v)$ into $F(u, v)$, this is a point by point multiplication. That is for every value u and v , the corresponding F component and the corresponding H component will be multiplied together to give you the final matrix which is again in the frequency domain. This problem could be reduced by limiting the analysis to frequencies near the origin.

The solution is again to carry out the restoration process in a limited neighborhood about the origin where $H(u, v)$ is not very small. This procedure is called **pseudoinverse filtering**.

- *In Inverse filtering, we simply take $H(u, v)$ such that the noise does not dominate the result. This is achieved by including only the low frequency components of $H(u, v)$ around the*

origin. Note that, the origin, $H(M/2, N/2)$, corresponds to the highest amplitude component.

Consider the degradation function of the atmospheric turbulence for the origin of the frequency spectrum,

$$H(u, v) = e^{-k[(u-M/2)^2 + (v-N/2)^2]^{1/6}}$$

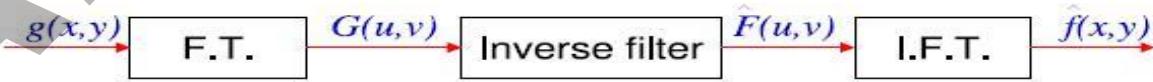
If we consider a Butterworth Lowpass filter of $H(u, v)$ around the origin we will only pass the low frequencies (high amplitudes of $H(u, v)$).

- As we increase the cutoff frequency of the LPF more smaller amplitudes will be included. Therefore, instead of the degradation function the noise will be dominating.

$$\hat{F}(u, v) = F(u, v) + \frac{N(u, v)}{H(u, v)}$$

Must not be very small. Otherwise the noise dominates

Restoration with an inverse filter



Minimum Mean Square Error (Wiener) Filtering

Least Square Error Filter

Wiener filter (constrained)

Direct Method (Stochastic Regularization)

Inverse Filter considers degradation function only and does not consider the noise part.

How do say that up to what extent of (u, v) value we should go? That is again image dependent. So, it is not very easy to decide that to what extent of frequency components we should consider for the reconstruction of the original image if i go for direct inverse filtering.

So, there is another approach which is the minimum mean square error approach or it is also called the Wiener filtering approach. In case of Wiener filtering approach, the Wiener filtering tries to reconstruct the degraded image by minimizing an error function. So, it is something like this

Restoration: *Wiener filter*

Degradation model:

$$g(x, y) = h(x, y) * f(x, y) + \eta(x, y)$$

Wiener filter: a statistical approach to seek an estimate f that minimizes the statistical function (mean square

$$\text{error: } e^2 = E \{ (f - f^*)^2 \}$$

* Assumptions:

Image and noise are uncorrelated

Image and/or noise has zero mean

Gray levels in the estimate are linear function of the levels in the degraded image

minimum mean square error or Wiener filtering approach for restoration of a degraded image.

$$\text{Mean Square Error MSE} = \frac{1}{N} \sum_{(x,y)} (g(x, y) - f(x, y))^2$$

Restoration with a Wiener filter

$$G(u,v) = H(u,v) F(u,v) + N(u,v)$$

$$\hat{F}(u,v) = W(u,v) G(u,v)$$



- In frequency domain:

$$\begin{aligned}\hat{F}(u,v) &= \left[\frac{H^*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2 + S_n(u,v)} \right] G(u,v) \\ &= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_n(u,v)/S_f(u,v)} \right] G(u,v) \\ &= \left[\frac{1}{|H(u,v)|} \cdot \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_n(u,v)/S_f(u,v)} \right] G(u,v)\end{aligned}$$

* $H(u, v)$: degradation function

* $|H(u, v)|^2 = H^*(u, v)H(u, v) \rightarrow H^*(u, v)$: complex conjugate

* $S_n(u, v) = |N(u, v)|^2$: the power spectrum of the noise
 $G(u, v)$ transform of the degraded image

The Wiener filter

$$\hat{F}(u,v) = W(u,v) G(u,v)$$

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$$

where

$$K(u,v) = S_n(u,v)/S_f(u,v)$$

$S_f(u,v) = |F(u,v)|^2$ power spectral density of $f(x,y)$

$S_n(u,v) = |N(u,v)|^2$ power spectral density of $n(x,y)$

- * $S_f(u, v) = |F(u, v)|^2$: the power spectrum of the undegraded image
- * $S_\eta(u, v)/S_f(u, v)$: noise-to-signal power ratio
→ If $S_\eta(u, v) = 0$: inverse filter
- Approximation of the noise-to-signal power ratio

$$S_\eta(u, v)/S_f(u, v) \cong R (= \eta_A/f_A: \text{constant})$$

$$\rightarrow \eta_A = \frac{1}{MN} \sum_u \sum_v S_\eta(u, v), \quad f_A = \frac{1}{MN} \sum_u \sum_v S_f(u, v)$$

$$* \text{ Hence, } \hat{F}(u, v) = \left| \frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + R} \right| G(u, v)$$

R is constant that is added to all terms of modulus $H(u, v)^2$

Now, in this case, you might notice that if the image does not contain any noise; then obviously, $S_\eta(u, v)$ which is the power spectrum of the noise will be equal to 0 and in that case, this wiener filter becomes identical with the inverse filter. But if the degraded image also contains additive noise in addition to the blurring; in that case, the wiener filter and the inverse filter is different.

Constrained Least Squared Filter (uses mean and variance of noise only)

Now, unlike in case of Wiener filtering where the performance of the Wiener filtering depends upon the correct estimation of the value of R that is the performance of Wiener filtering depends upon how correctly you can estimate what is the power spectrum of the original un degraded image.

Constrained Least Square Filter method uses the mean of the noise which we will write as say m_η and the variance of the noise which we will write as σ^2_η

So, we will see that how the reconstruction using this constant least square filter approach makes use of this noise parameter like mean of the noise and the variance of the noise.

Degradation model:

$$g(x, y) \quad f(x, y) \quad h(x, y) \quad (x, y)$$

In matrix form, $\mathbf{g} = \mathbf{Hf} + \boldsymbol{\eta}$

Now here, you will notice that the value of \mathbf{H} is very very sensitive to noise. So, to take care of that what we do is we define an optimality criteria and using that optimality criteria, the reconstruction has to be done.

The optimality criteria that we will use is the image smoothness.

So, you know from our earlier discussion that the second derivative operation or the Laplacian operator, it tries to enhance the irregularities or discontinuities in the image.

Optimal criteria in this case is to minimize the Laplacian of the reconstructed image

Objective: to find the minimum of a criterion function

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

Subject to the constraint $\|\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}\|_2 \leq \|\boldsymbol{\eta}\|_2$

where this $\hat{\mathbf{f}}$, this is the reconstructed image.

So, we will try to minimize these optimality criteria subject to the constraint that \mathbf{g} minus $\mathbf{H}\hat{\mathbf{f}}$ square is equal to $\boldsymbol{\eta}$ square and that is why it is called constant least square filtering.

Again, without going into the details of mathematical derivation, we will simply give the frequency domain solution of this particular constant least square estimation where the frequency domain solution now is given by $\hat{F}(u, v)$ is equal to $H^*(u, v)$ upon $|H(u, v)|^2 + \gamma |P(u, v)|^2$, this times $G(u, v)$.

$$\hat{F}(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + \gamma |P(u, v)|^2} G(u, v)$$

We get a constrained least square filter

$$\text{Where } P(u, v) = \text{Fourier transform of } p(x, y) = \begin{matrix} 0 & 1 & 0 \\ 1 & 4 & 1 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 0 \end{matrix}$$

γ is adaptively adjusted to achieve the best result

Constrained least squares restoration using different values f or γ . Note that γ is a scalar constant, not a ratio of frequency domain functions as with the Wiener case, that are unlikely to be constant. In this case, γ is determined manually

Again as before, this H^* star indicates that it is the complex conjugate of H . Here again, we have a constant term given as γ where the γ is to be adjusted so that the specified constant that is $g - Hf$ hat square is equal to n square this constant is met.

So, as we said that this γ has to be adjusted manually for obtaining the optimum result and the purpose is that this adjusted value of γ , the γ is adjusted so that the specified constant is maintained. However, it is also possible to automatically estimate the value of γ by an iterative approach.

UNIT 5 : IMAGE SEGMENTATION

All unit world

UNIT 5 : IMAGE SEGMENTATION

Input is Image  output is features of images

Segmentation is an approach for **Feature extraction** in an image

Features of Image: Points, lines, edges, corner points, regions

Attributes of features :

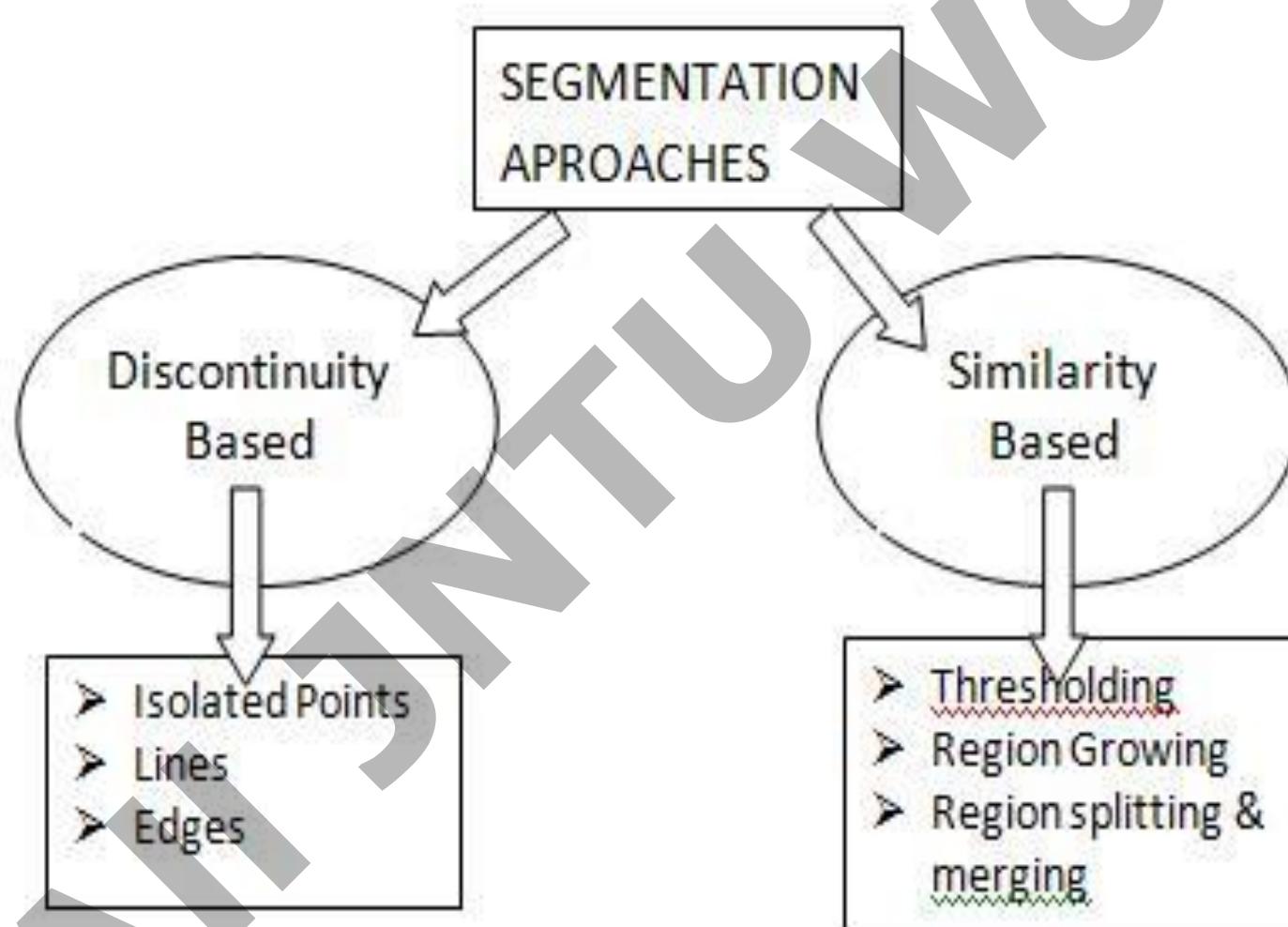
Geometrical (orientation, length, curvature, area, diameter, perimeter etc)

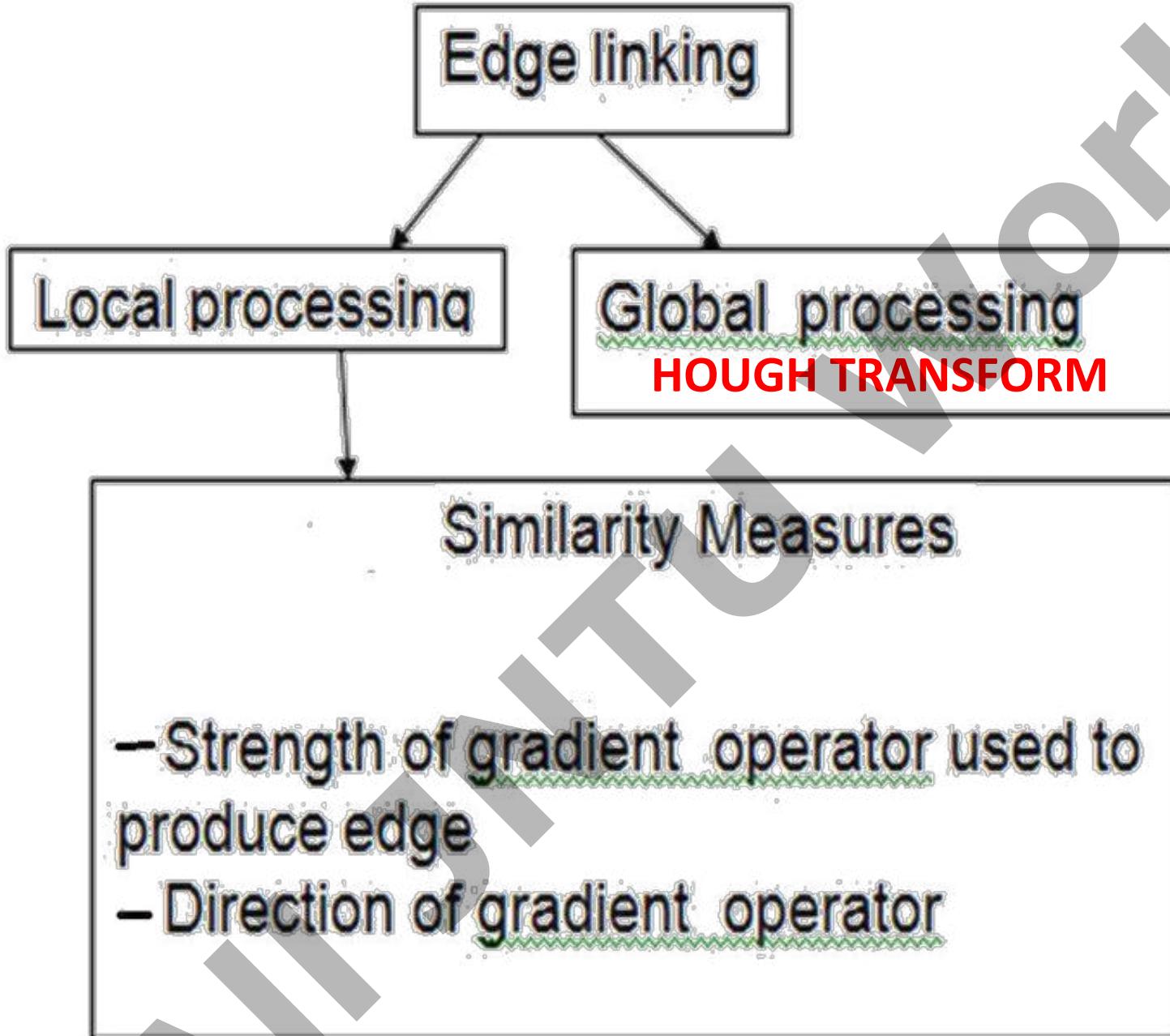
Topological attributes: overlap, adjacency, common end point, parallel, vertical etc

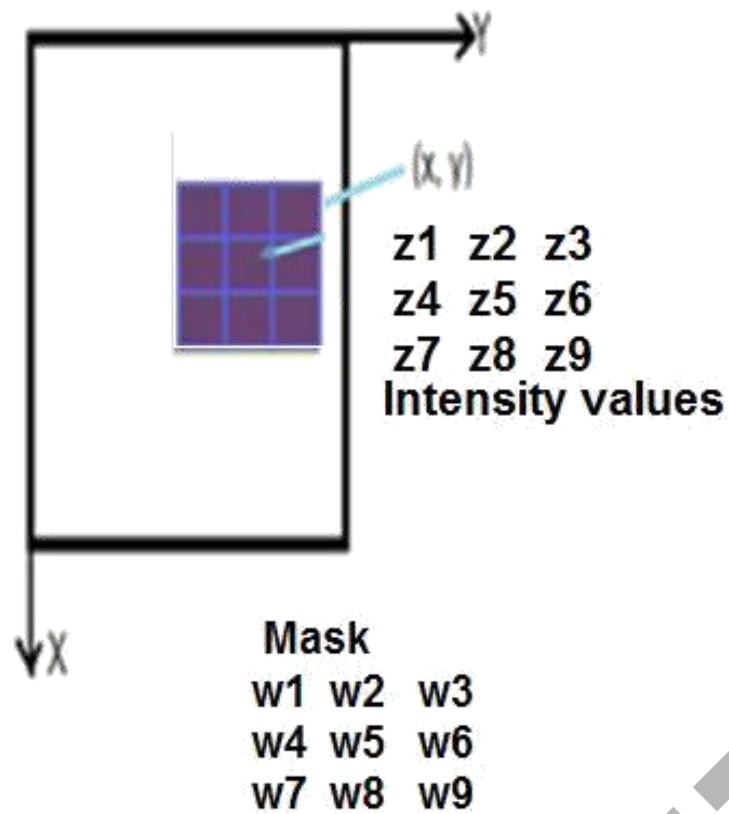
Image segmentation refers to the process of partitioning an image into groups of pixels which are **homogeneous** with respect to some criterion..

SEGMENTATION APPROACHES

Based on gray level value properties, segmentation is mainly categorized into one of the two categories;







Point detection:

$W_{-1,-1}$	$W_{-1,0}$	$W_{-1,1}$
$W_{0,-1}$	$W_{0,0}$	$W_{0,1}$
$W_{1,-1}$	$W_{1,0}$	$W_{1,1}$

$$R = \sum_{i=-1}^1 \sum_{j=-1}^1 W_{i,j} f(x + i, y + j)$$

Or $R = \sum_{k=1}^9 w_k Z_k$; Z_k is the intensity at the pixel, w_k is the 3×3 mask coefficient. Response of the mask at the center point of the region is R.

Isolated Point detection:

So, when it comes to an isolated point detection, we can use a mask having^{or} the coefficient as given below.

Now, we say that an isolated point at a location say (x, y) is detected in the image where the mask is centered if the corresponding modulus R value, is greater than certain threshold say T where this T is a non negative threshold value.

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$|R(x,y)| \geq T$, isolated point is detected,
negative threshold

where T is Non

$$g(x,y) = \begin{cases} 1 & \text{if } |R(x,y)| \geq T \\ 0 & \text{otherwise} \end{cases}$$

Detection of Lines,

Apply all the 4 masks on the image

$$\begin{matrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{matrix}$$

$$\begin{matrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{matrix}$$

$$\begin{matrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{matrix}$$

$$\begin{matrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{matrix}$$

Horizontal

- 45 deg

vertical

45 deg

$$|R_i| > |R_j| , \text{ for all values of } j \text{ not equal to } i \quad \forall_{j \neq i}$$

There will be four responses R₁, R₂, R₃, R₄.

Suppose that at a particular point,

$$|R_1| > |R_j| , \text{ where } j=2,3,4 \text{ and } j \neq 1$$

Then that point is on Horizontal line.

Like this, we can decide which point is associated with which line.

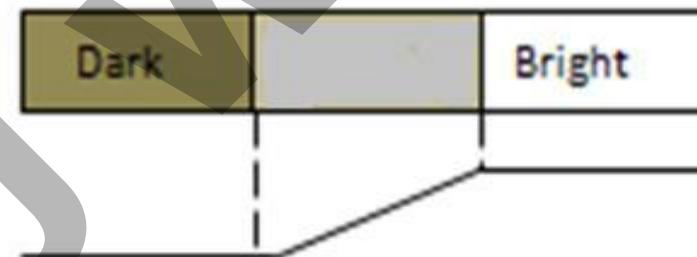
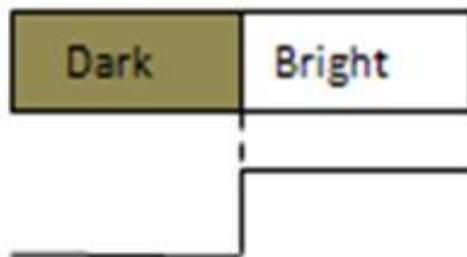
Line detection is a low level feature extraction.

All JNTU World

Detection of an edge in an image:

What is edge:

An ideal Edge can be defined as a set of connected pixels each of which is located at an orthogonal step transition in gray level



Gray level profile
of a horizontal
line through the
image

Gray level profile
of a horizontal line
through the
image

Calculation of Derivatives of Edges:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

The magnitude of this vector is given by

$$\begin{aligned}\nabla f &= \text{mag}(\nabla \mathbf{f}) \\ &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}.\end{aligned}$$

$$\nabla f \approx |G_x| + |G_y|.$$

There are various ways in which this first derivative operators can be implemented

Prewitt Edge Operator

$$\begin{matrix} 1 & 1 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & -1 & -1 & 1 & 0 & -1 \end{matrix}$$

Horizontal
Gx

Vertical
Gy

Sobel Edge Operator (noise is taken care)

$$\begin{matrix} 1 & 2 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 2 & 0 & -2 \\ -1 & -2 & -1 & 1 & 0 & -1 \end{matrix}$$

Horizontal
Gx

vertical
Gy

The **direction** of the edge that is the direction of gradient vector f. Direction $\alpha(x,y) = \tan^{-1}(Gy/Gx)$

The direction of an edge at a pixel point (x,y) is orthogonal to the direction $\alpha(x,y)$

Edge Linking

we have to detect the position of an edge and by this, what is expected is to get the **boundary** of a particular segment.

For this there are two approaches : One is local processing

HOUGH TRANSFORM

The second approach is global processing (**HOUGHS transformation**)

Similarity Measures

- Strength of gradient operator used to produce edge
- Direction of gradient operator

EDGE LINKING BY

LOCAL PROCESSING

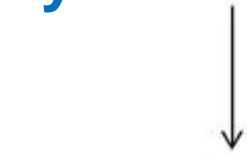
A point (x, y) in the image which is already operated by the sobel edge operator. T is threshold

In the edge image take two points x, y and x', y' and to link them

Use similarity measure

**first one is the strength of the gradient
operator the direction of the gradient**

By sobel edge operator.



$$\nabla f \approx |G_x| + |G_y|.$$

$$|\nabla f(x, y) - \nabla f(x', y')| \leq T$$

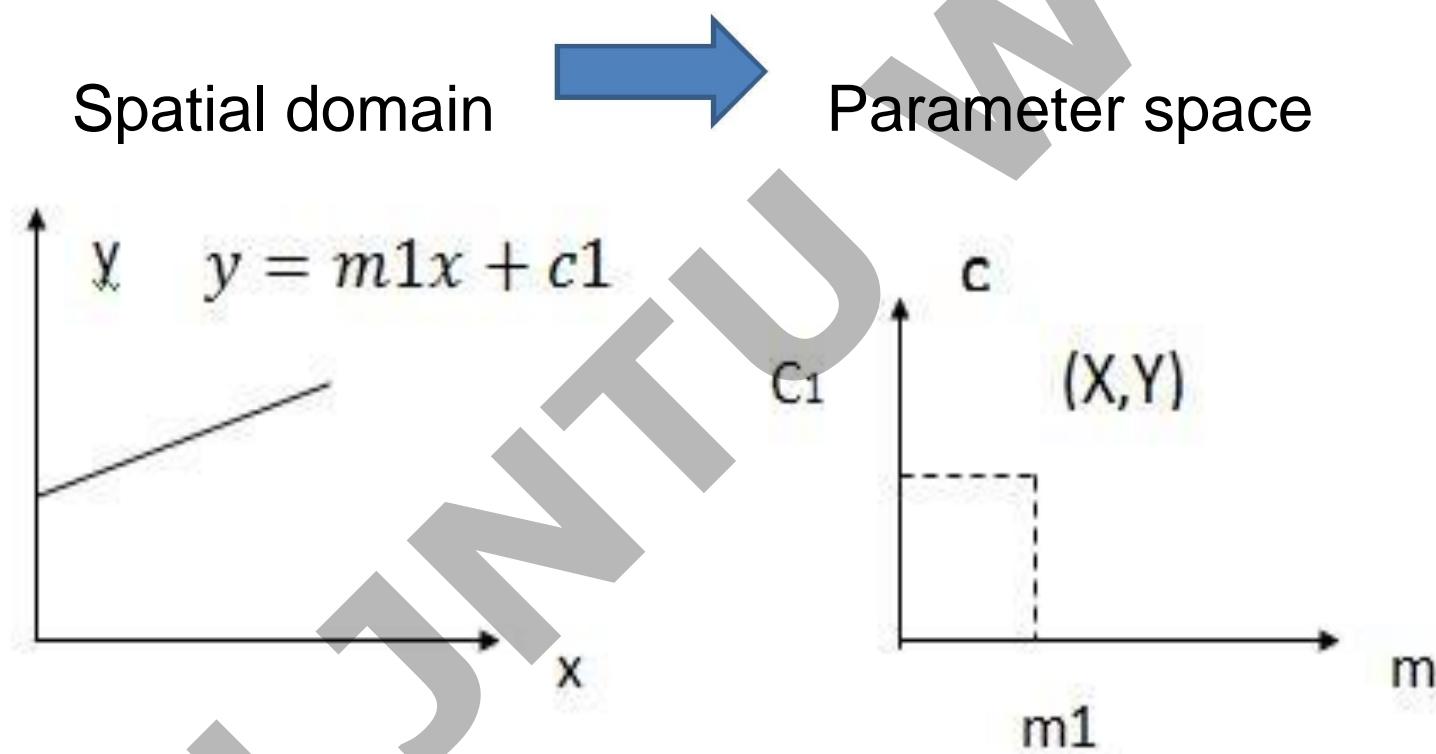
$$[|\alpha(x, y) - \alpha(x', y')|] \leq A$$

These 2 points are similar and those points will be linked together and such operation has to be done for each and do this for every other point in the edge detected image

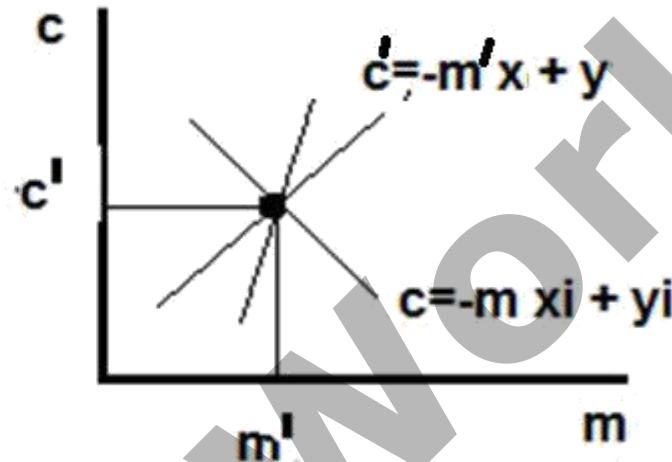
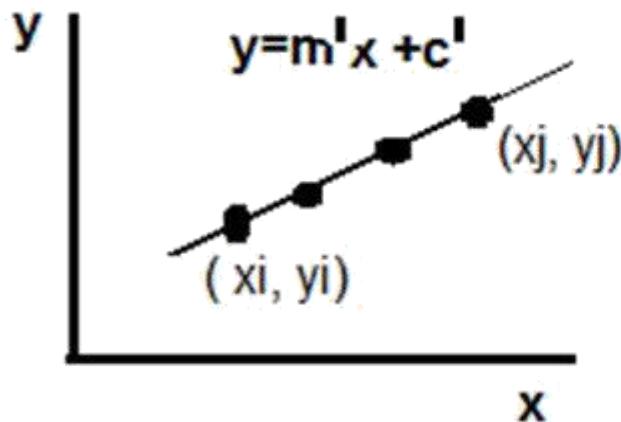
HOUGH TRANSFORM

Global processing

The Hough transform is a mapping from the spatial domain to a parameter space for a particular straight line, the values of m and c will be constant



- Mapping this straight line in the parameter space.



So, we have seen 2 cases

Case one: a straight line in the xy plane is mapped to a point in the mc plane and

Case two: if we have a point in the xy plane that is mapped to a straight line in the mc plane

and this is the basis of the Hough transformation by using which we can link the different edge points which are present in the image domain

Image Space

Lines

Points

Collinear points

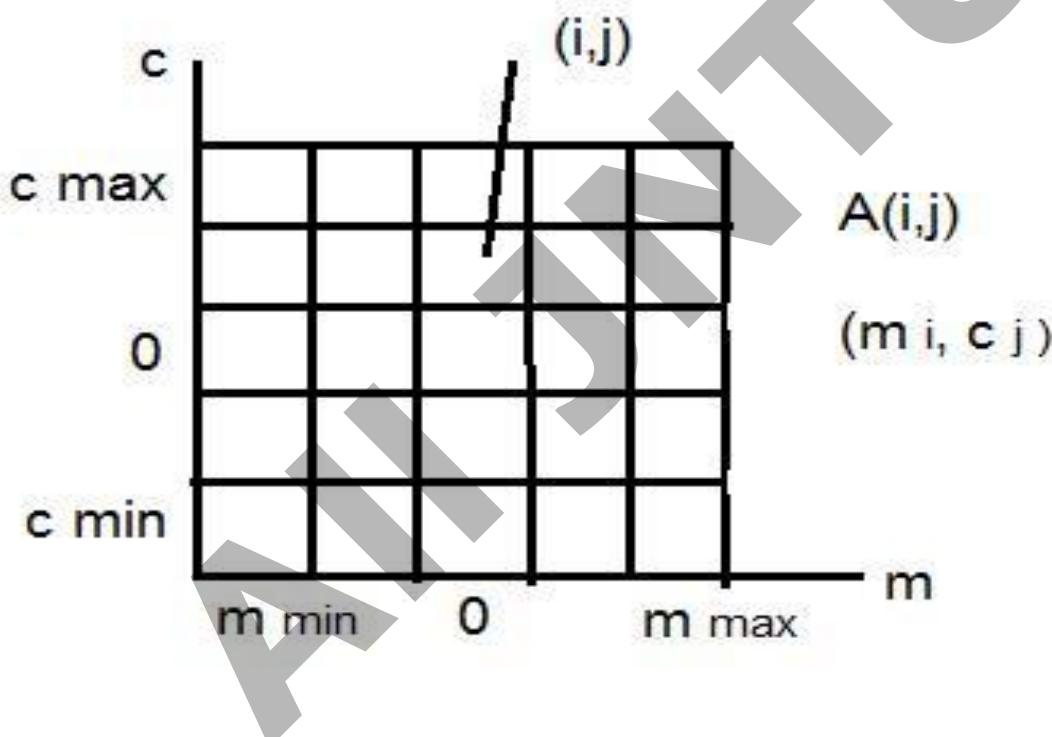
Parameter Space

Points

Lines

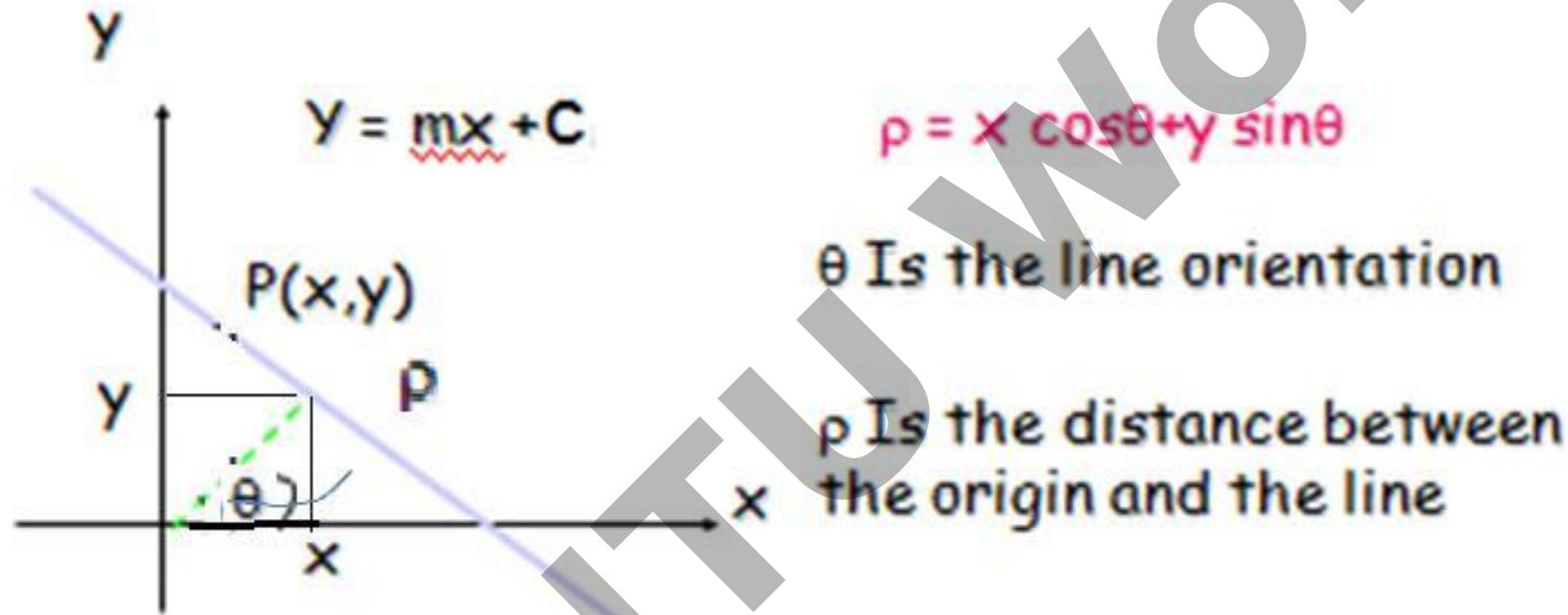
Intersecting lines

So for **implementation of Hough Transform**, what we have to do is this entire mc space has to be subdivided into a number of accumulator cells.



- At each point of the parameter space, count how many lines pass through it.
- This is a “bright” point in the parameter image. It can be found by thresholding. This is called the accumulator

when this straight line tries to be vertical, the slope m tends to be infinity ; to solve this make use of the normal representation of a straight line Use the “Normal” equation of a line:



A Point in Image Space is now represented as a SINUSOID
 $= x \cos \theta + y \sin \theta$ Therefore, use $(,)$ space
 $= x \cos \theta + y \sin \theta$
= magnitude
drop a perpendicular from origin to the line
= angle perpendicular makes with x-axis

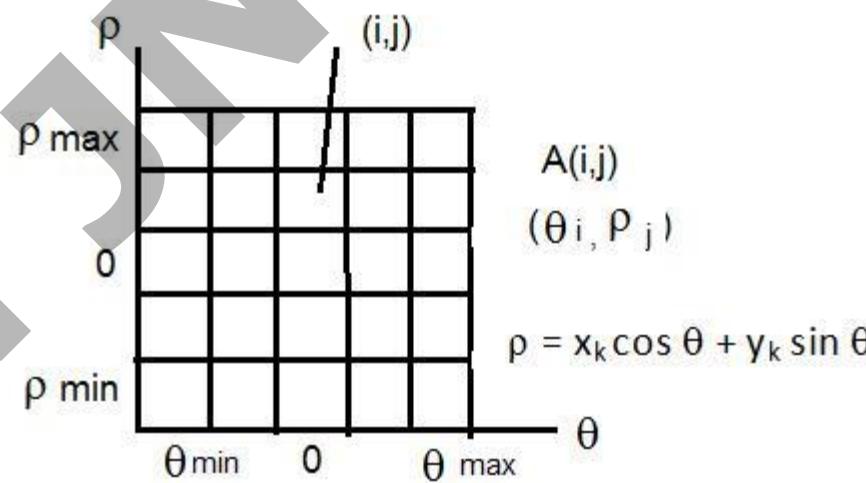
So, unlike in the previous case where the parameters were the slope m and c , now parameters become ρ and θ .

- Use the parameter space (ρ, θ)
- The new space is FINITE
- $0 < \rho < D$, where D is the image diagonal $\rho = \sqrt{M^2 + N^2}$, $M \times N$ is image size.

- $0 < \theta < \pi$ (or $= \pm 90$ deg)

- In (ρ, θ) space

point in image space == sinusoid in (ρ, θ) space
where sinusoids overlap, accumulator = max
maxima still = lines in image space



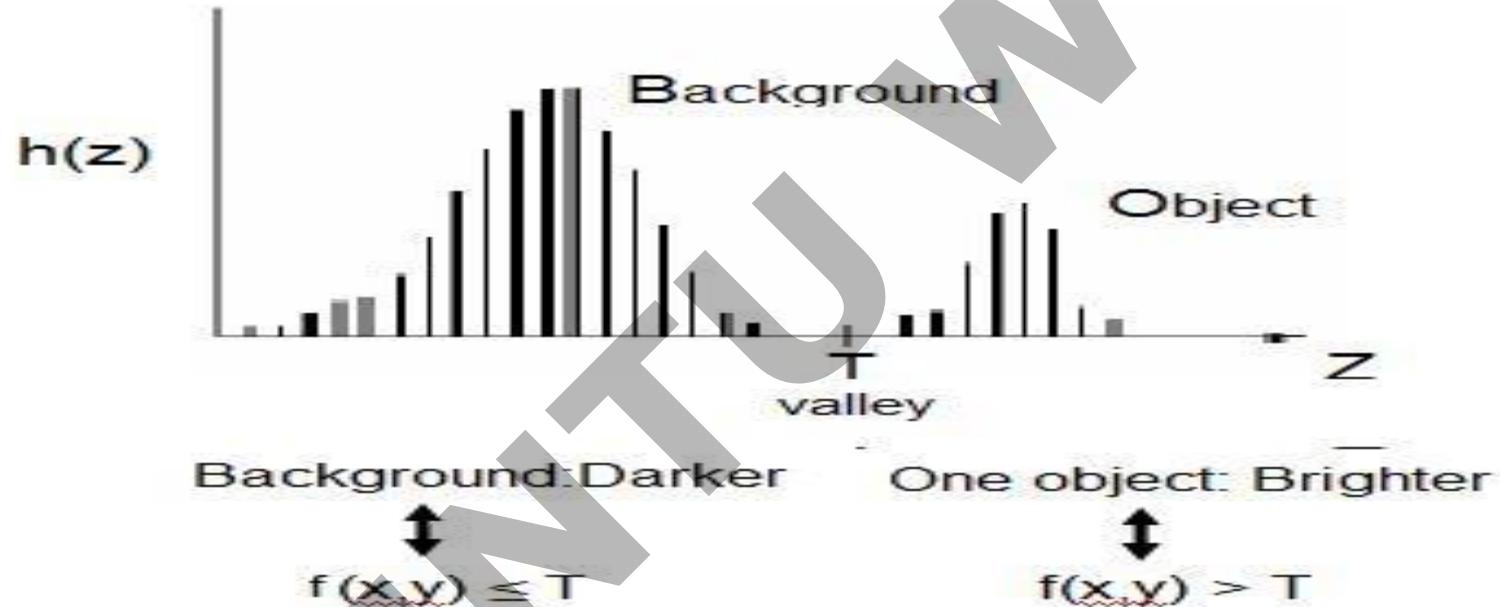
Global Thresholding : a threshold value is selected where the threshold value depends only on the pixel intensities in the image

Dynamic or adaptive thresholding: Threshold depends on pixel value and pixel position. So, the threshold for different pixels in the image will be different.

Optimal thresholding : estimate that how much is the error incorporated if we choose a particular threshold. Then, you choose that value of the threshold where by which your average error will be minimized

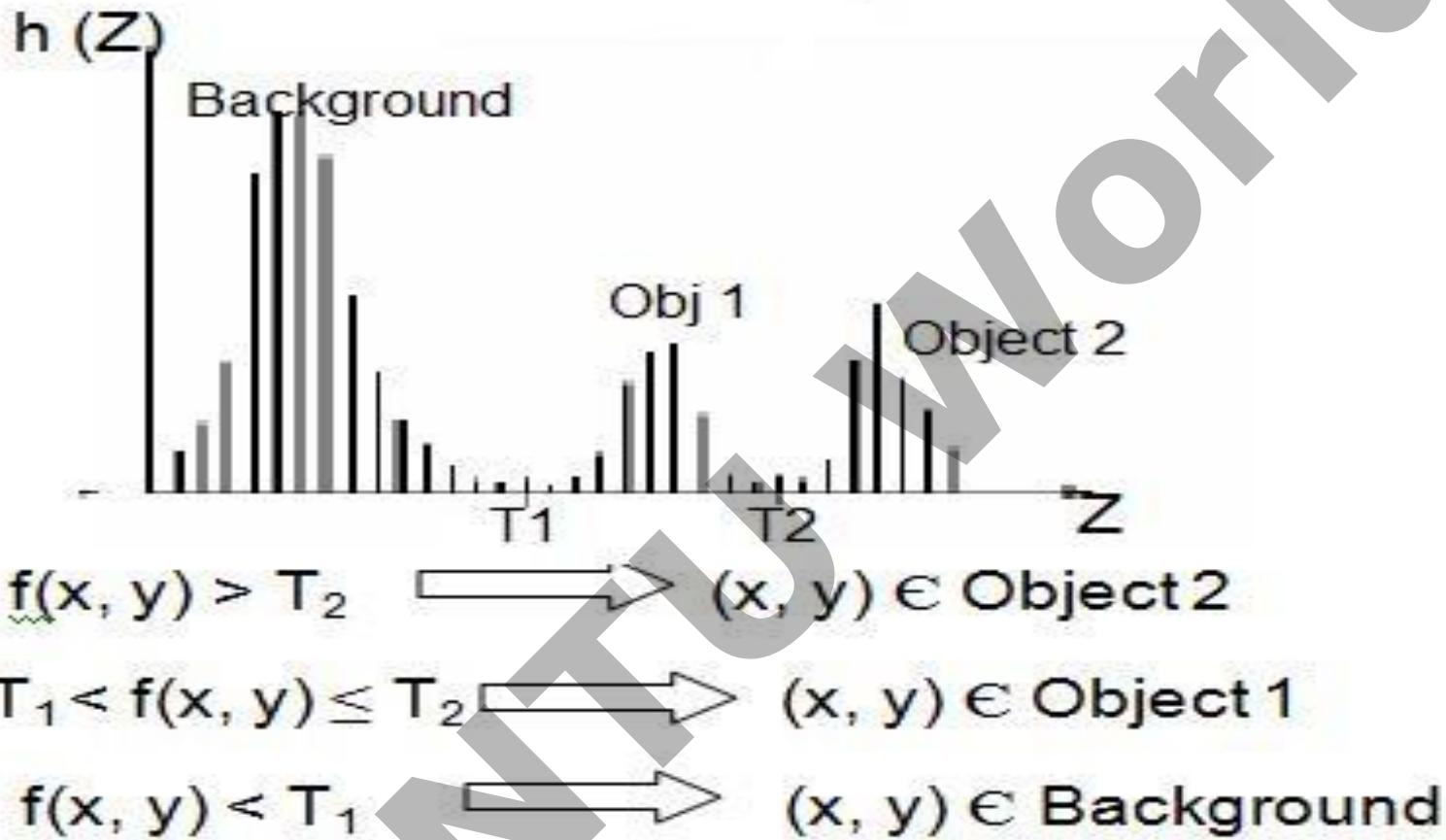
THRESHOLDING

Region based segmentation
operations thresholding
region growing and
the region splitting and merging techniques



So, for such a bimodal histogram, you find that there are two peaks.
Now, the simplest form of the segmentation is, choose a threshold value say T in the valley region
if a pixel at location x,y have the intensity value $f(x, y) \geq T$; then we say that these pixel belongs to object
whereas if $f(x, y) < T$, then these pixel belongs to the background.

Thresholding In a multi modal histogram



So, you will find that the basic aim of this thresholding operation is we want to create a thresholded image $g(x, y)$ which will be a binary image containing pixel values either 0 or 1 depending upon whether the intensity $f(x, y)$ at location (x, y) is greater than T or it is less than or equal to T .

This is called **global thresholding**.

Automatic Thresholding

1. Initial value of Threshold T
2. With this threshold T, Segregate the pixels into two groups G1 and G2
3. Find the mean values of G1 and G2. Let the means be μ_1 and μ_2
4. Now Choose a new threshold. Find the average of the means

$$T_{\text{new}} = (\mu_1 + \mu_2)/2$$

5. With this new threshold, segregate two groups and repeat the procedure. $|T - T_{\text{new}}| > \Delta T'$, back to step.
Else stop.

Basic Adaptive Thresholding is

- Divide the image into sub-images and use local thresholds

But, in case of such non uniform illumination, getting a global threshold which will be applicable over the entire image is very very difficult

So, if the scene illumination is non uniform, then a global threshold is not going to give us a good result. So, what we have to do is we have to **subdivide the image into a number of sub regions and find out the threshold value for each of the sub regions** and segment that sub region using this estimated threshold value and here, because your threshold value is position dependent, it depends upon the location of the sub region; so the kind of thresholding that we are applying in this case is an adaptive thresholding.

Basic Global and Local Thresholding

Simple thresholding schemes compare each pixels gray level with a single global threshold. This is referred to as **Global Thresholding**.

If T depends on both $f(x,y)$ and $p(x,y)$ then this is referred to a **Local Thresholding**

Adaptive thresholding Local Thresholding

Adaptive Thresholding is

- Divide the image into sub-images and use local thresholds,
Local properties (e.g., statistics) based criteria can be used for adapting the threshold.

Statistically examine the intensity values of the local neighborhood of each pixel. The statistic which is most appropriate depends largely on the input image. Simple and fast functions include the *mean* of the *local* intensity distribution,

$$T = \text{Mean}, \quad T = \text{Median}, \quad T = (\text{Max} + \text{Min}) / 2$$

You can simulate the effect with the following steps:

1. Convolve the image with a suitable statistical operator,
i.e. the *mean* or *median*.
2. Subtract the original from the convolved image.
3. Threshold the difference image with C .
4. Invert the thresholded image.

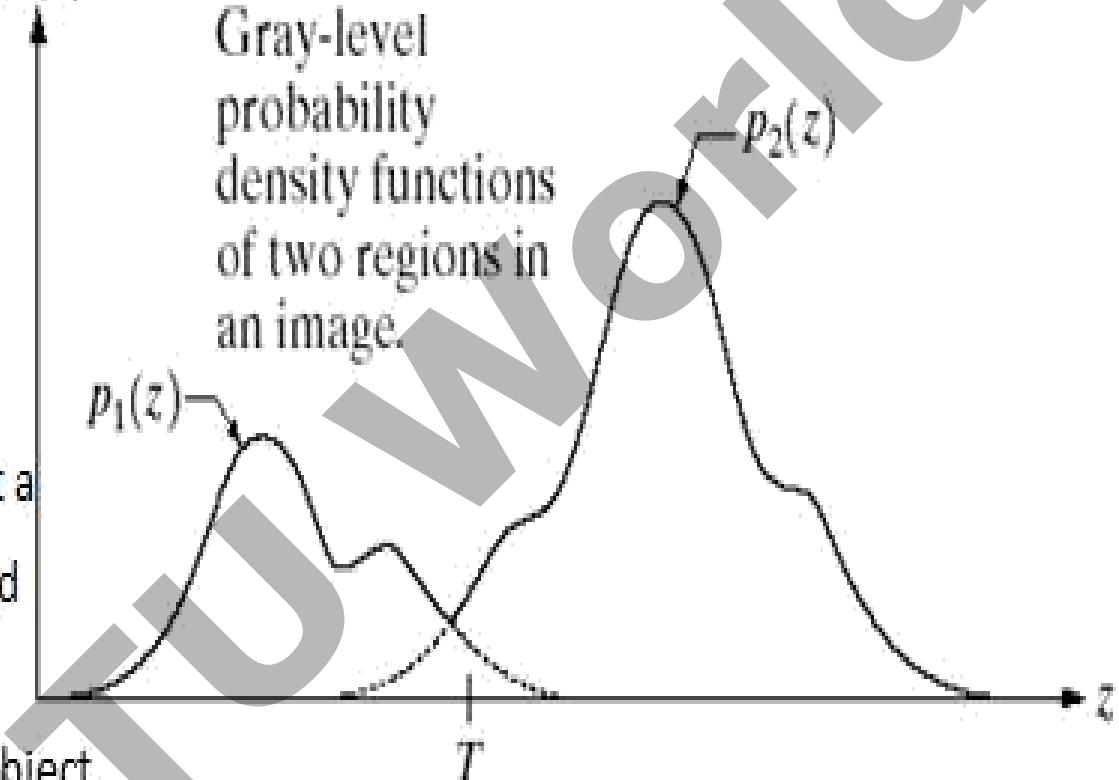
OTIMAL THREOLDING

$P(z)$ is histogram

$$P(z) = P_1 P_1(z) + P_2 P_2(z)$$

$$P_1 + P_2 = 1$$

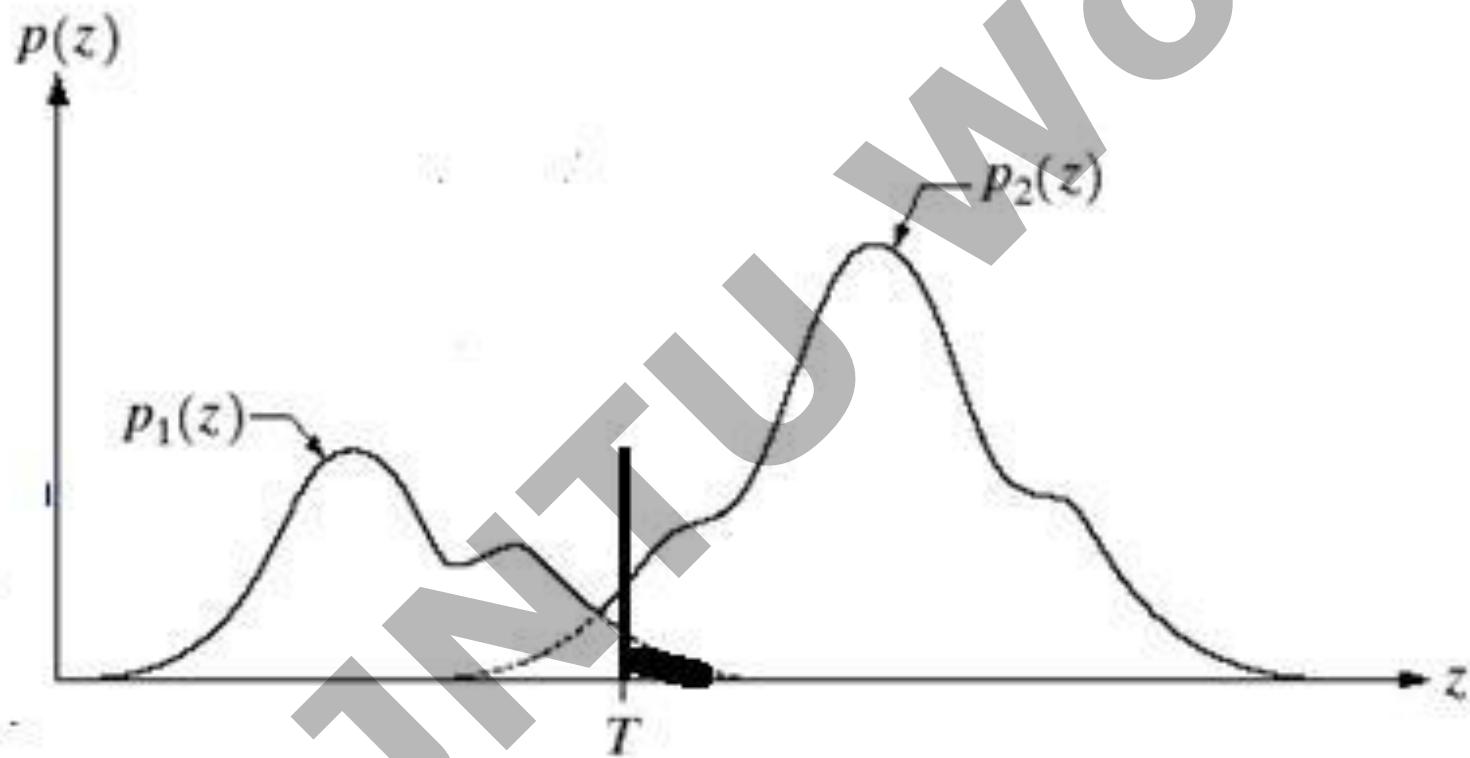
Capital P_1 indicates the probability that a pixel will belong to the background and capital P_2 indicates that a pixel belongs to an object.



The normalized histogram can be viewed as a probability density function $p(z)$ of this random variable z .

The overall histogram that is $p(z)$ can be represented as the combination of $p_1(z)$ and $p_2(z)$

Now, what is our aim in this particular case? Our aim is that we want to determine a threshold T which will minimize the average segmentation error.



$f(x,y) > T, (x,y)$ belongs to Object

If $f(x, y)$ is greater than T , then (x, y) belongs to object but the pixel with intensity value $f(x, y)$ also has a finite probability; say given by this that it may belong to the background. So, while taking this decision, we are incorporating some error. The error is the area given by this probability curve for the region intensity value greater than T . Let us say corresponding error will be given by $E_1(T)$

Similarly, if a background pixel is classified as an object pixel, then the corresponding error will be given by $E_2(T)$ is equal to integral $P_1(z) dz$ where the integral has to be taken from T to infinity.

Overall probability of error is given by:

$$E(T) = P_2 E_1(T) + P_1 E_2(T)$$

Now, for minimization of this error

$$\partial E(T) / \partial T = 0$$

By assuming Gaussian probability density function,

$$E(z) = \frac{P_1}{(\sqrt{2\pi})\sigma_1} e^{\frac{-(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{(\sqrt{2\pi})\sigma_2} e^{\frac{-(z-\mu_2)^2}{2\sigma_2^2}}$$

The value of T can now be found out as the solution for T is given by, solution of this particular equation

$$AT^2 + BT + C = 0$$

$$A = \sigma_1^2 - \sigma_2^2$$

$$B = 2 (\mu_1 \sigma_2^2 - \mu_2 \sigma_1^2)$$

$$C = (\mu_2^2 \sigma_1^2 - \mu_1^2 \sigma_2^2) + 2 \sigma_1^2 \sigma_2^2 \ln \left(\frac{\sigma_2 P_1}{\sigma_1 P_2} \right)$$

$$\sigma^2 = \sigma_1^2 = \sigma_2^2$$

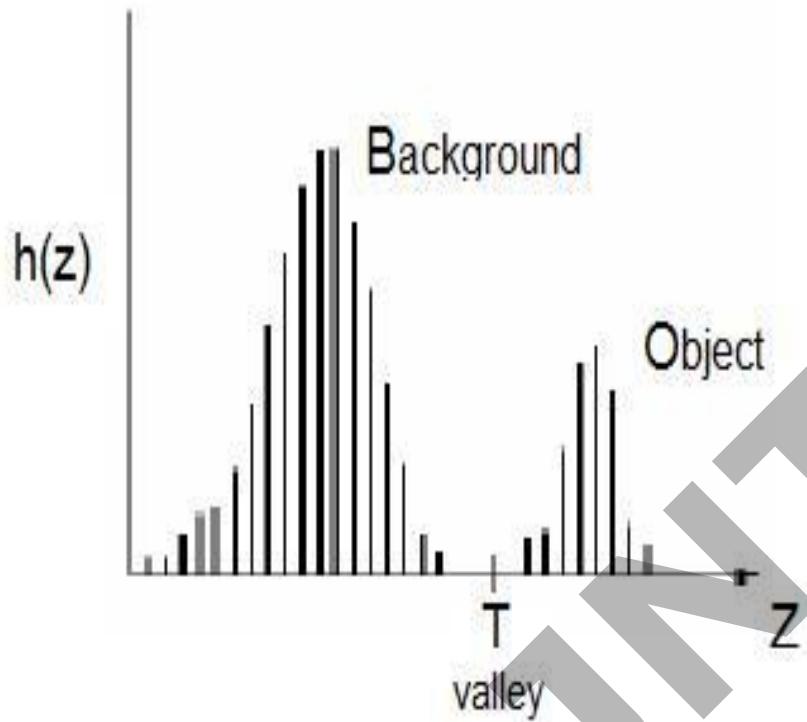
Optimal Threshold is obtained by

$$T = (\mu_1 + \mu_2) / 2 + [\sigma^2 / (\mu_1 - \mu_2)] \ln (P_2/P_1)$$

The capital P_1 and capital P_2 , they are same; in that case, the value of T will be simply μ_1 plus μ_2 by 2 that is the mean of the average intensities of the foreground region and the background region.

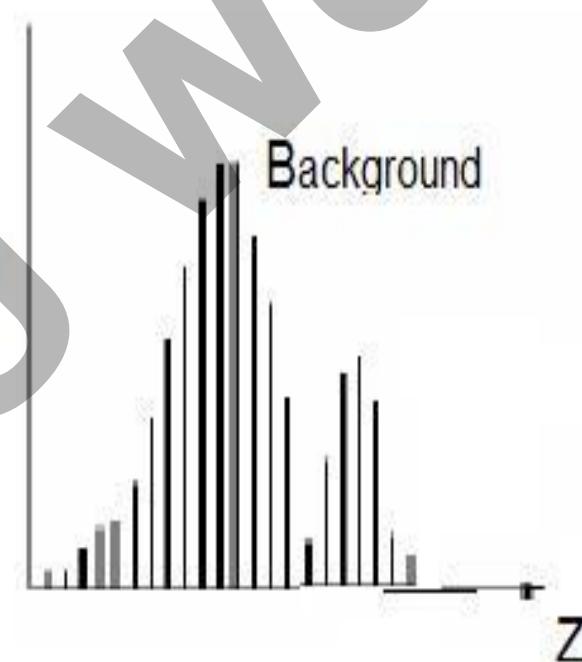
Boundary characteristics for Histogram Thresholding

Use of Boundary Characteristics for Histogram Improvement and Local Thresholding



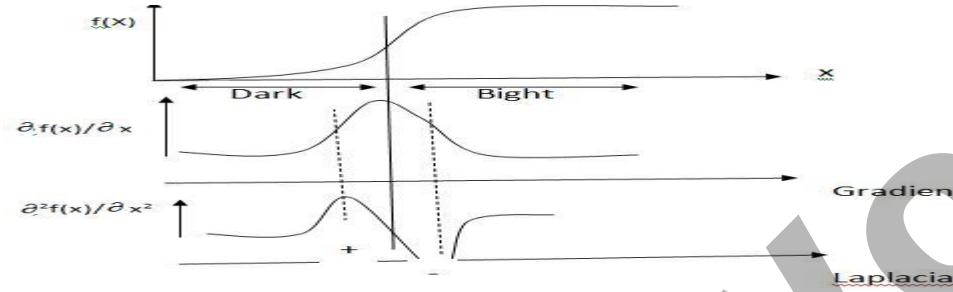
In this histogram, it is easy to find the threshold in the valley

Peaks are separate Peaks are symmetric



So, the bimodal nature of the histogram is not very visible and is dominated by a single mode by the pixels which belong to the background

But, the boundary between the object and the boundary itself is not known



Compute the gradient of intensities (first derivative) and the second order derivative operator, the Laplacian (this will be affected by noise)

So, First derivative will be used for edge position identification and Laplacian for identifying direction. On the bright side of the edge, the Laplacian becomes negative

So, our approach is though we have said that we want to consider only those pixels for generation of the histogram which are lying either on the boundary either on the edge between the object and the background; so, that information can be obtained by using from the output of the gradient because for all the pixels which are lying on the boundary or near the boundary, the gradient magnitude will be quite high

$$s(x, y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \quad \text{and } \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \quad \text{and } \nabla^2 f < 0 \end{cases}$$

So, if the gradient value is less than some threshold T , we assume that this point does not belong to edge point does not belong to an edge or this point is not even within a region near the edge. So, for such points, we are making $s(x, y)$ is equal to 0 and we will put $s(x, y)$ is equal to positive if gradient of f is greater than or equal to T indicating that this is an edge point or this is a point near the edge and at the same time, if del square f is greater than or equal to 0 which indicates that this point is on the dark side of the edge.

Region growing:

starting from this particular pixel, you try to grow the region based on connectivity or based on adjacency and similarity. So, this is what is the region growing based approach

Group pixels from sub-regions to larger regions

- Start from a set of 'seed' pixels and append pixels with similar properties
 - Selection of similarity criteria: color, descriptors (gray level + moments)
 - Stopping rule

Basic formulation

- Every pixel must be in a region
- Points in a region must be connected
- Regions must be disjoint
- Logical predicate for one region and for distinguishing between regions

Region growing operation will start from the seed point.

Choosing a 3 by 3 neighborhood around the seed point and grow the region starting from the seed point, then all the points which include in the same group or in the same partition, these points have to be connected. That means, start growing this region from the points which are **connected to the seed point**. And at the end, what we have is a number of regions which are grown around these seed points.

So, what does this region growing actually mean? The region growing as the name implies that it is a procedure which groups the pixels or sub regions into a larger region based on some predefined criteria , say, **similarity or close in intensities**.

Region splitting & merging –Quadtree decomposition

Region splitting and merging : split the image into a number of smaller size sub images or smaller size components, then you try to merge some of those sub images which are adjacent and which are similar in some sense.

If I have an image say R

**If all the pixels in the image are similar,
leave it as it is**

If they are not similar,
then you break this image into quadrants.

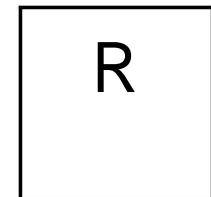
make 4 partitions of this image.

Then, check each and every partition is similar

If it is not similar, again you partition that particular region.

Let us suppose that all the pixels in R are not similar; (say VARIANCE IS LARGE)

Let R
denote the
Full image.

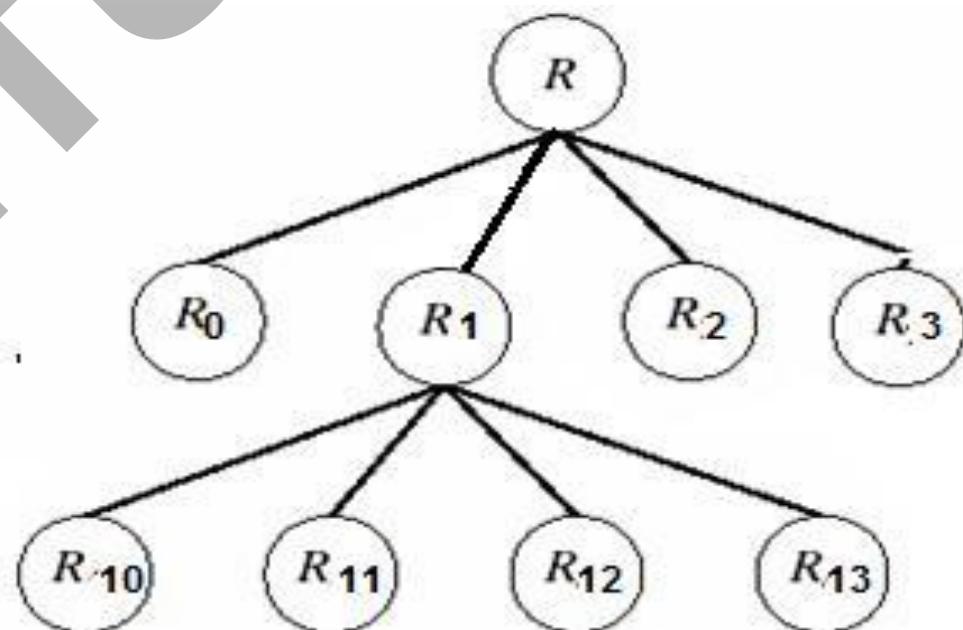
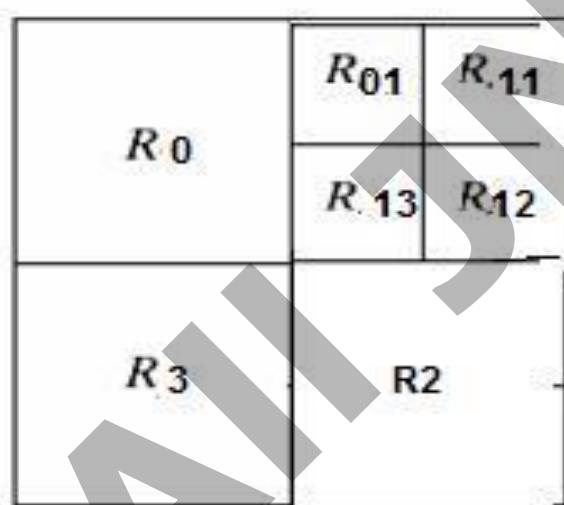


Now say, this R_1 is not uniform,
so partition R_1 region again making it $R_{10} R_{11} R_{12} R_{13}$

and

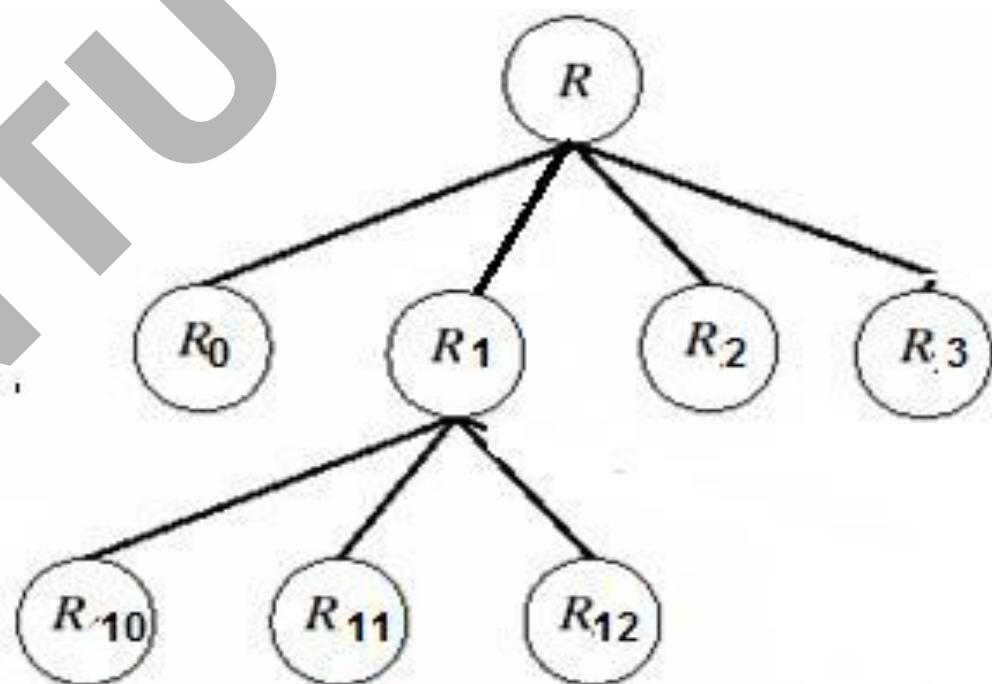
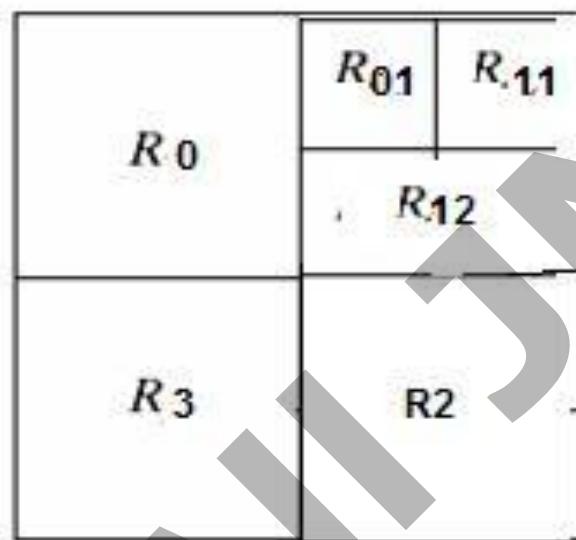
you go on doing this partitioning until and unless you come to a partition size which is the smallest size permissible or you come to a situation where the partitions have become uniform, or so you cannot partition them anymore.

And in the process of doing this, we have a quad tree representation of the image.



So, in case of quad tree representation, if root node is R, initial partition gives out 4 nodes - R₀ R₁ R₂ and R₃. Then R₁ gives again R₁₀ R₁₁ R₁₂ and R₁₃. Once such partitioning is completed, then what you do is you try to check all the adjacent partitions to see if they are similar.

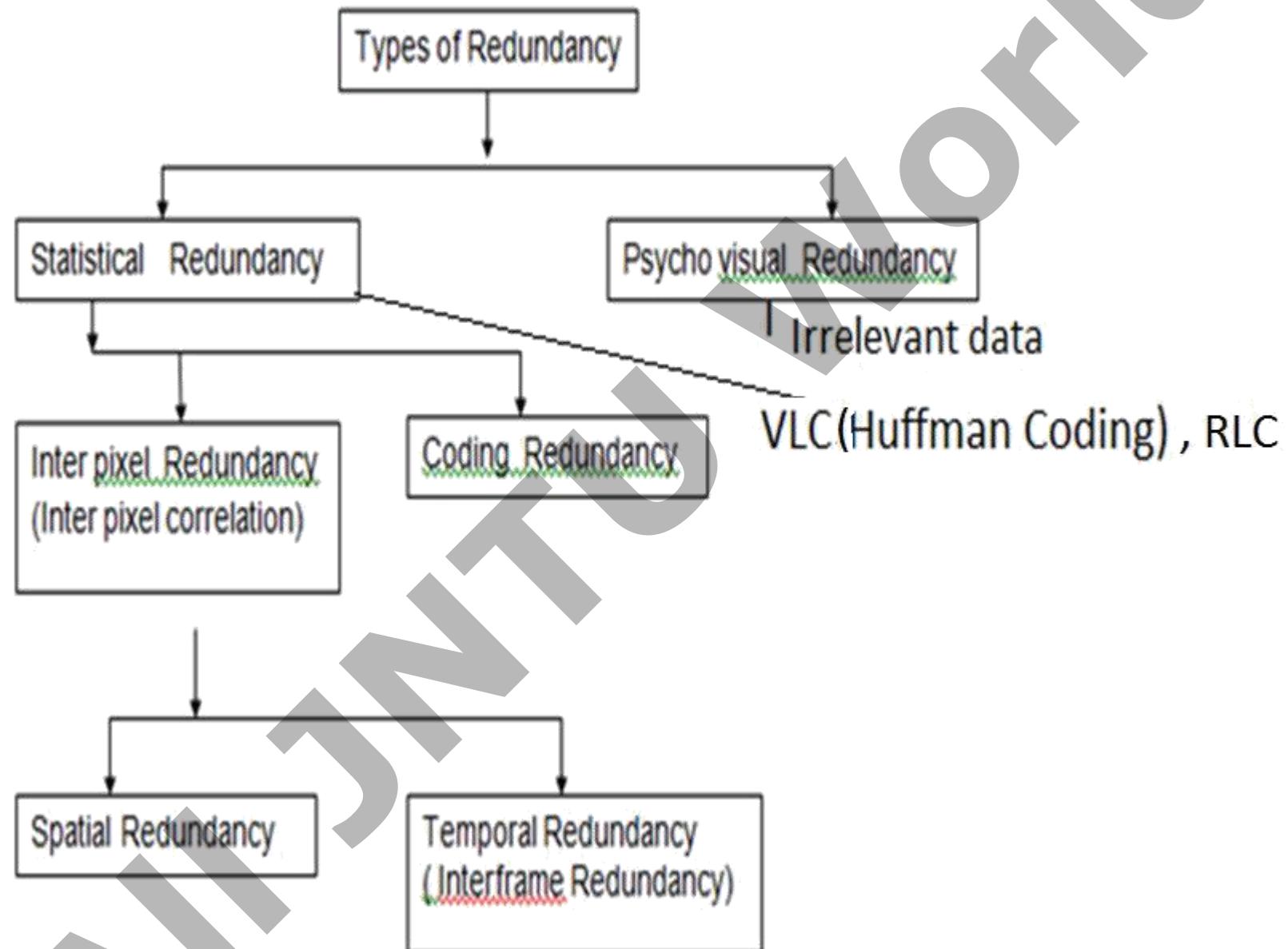
If they are similar, you merge them together to form a bigger segment. Say, if R₁₂ and R₁₃ are similar. Merge them.



So, this is the concept of splitting and merging technique for segmentation.

Now at the end, leave it if no more partition is possible ie. reached a minimum partition size or every partition has become uniform;
then look for adjacent partitions which can be combined together to give me a bigger segment.

Types of Redundancy



Data redundancy is the central concept in image compression and can be mathematically defined.

Data Redundancy

Because various amount of data can be used to represent the same amount of information, representations that contain irrelevant or repeated information are said to contain redundant data.

- The Relative data redundancy R_D of the first data set, n_1 , is defined by:

$$R_D = 1 - \frac{1}{C_R}$$

C_R refers to the compression ratio compression ratio (CR) or bits per pixel (bpp) and is defined by:

Compression Ratio C_R = $\frac{\text{uncompressed file size}}{\text{Compressed file size}} = \frac{n_1}{n_2}$

If $n_1 = n_2$, then $CR=1$ and $R_D=0$, indicating that the first representation of the information contains no redundant data.

Coding Redundancy :

- Code: a list of symbols (letters, numbers, bits , bytes etc.)
- Code word: a sequence of symbols used to represent a piece of information or an event (e.g., gray levels).
- Code word length: number of symbols in each code word

Ex: 101 Binary code for 5, Code length 3, symbols 0,1

The gray level histogram of an image can be used in construction of codes to reduce the data used to represent it. Given the normalized histogram of a gray level image where

$$p_r(r_k) = \frac{n_k}{n} \quad k=0,1,2,\dots,L-1$$

r_k is the pixel values defined in the interval [0,1] and $p_r(r_k)$ is the probability of occurrence of r_k . L is the number of gray levels. n_k is the number of times that k th gray level appears in the image and n is the total number of pixels ($n=M \times N$)

An 8 gray level image has the following gray level distribution.

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6
8 gray levels		Fixed 3 bit code		Variable length code	

Example of Variable Length Coding

The average number of bit used for fixed 3-bit code:

$$L_{avg} = \sum_{k=0}^7 l_1(r_k) p_r(r_k) = 3 \sum_{k=0}^7 p_r(r_k) = 3 \times 1 = 3 \text{ bits}$$

$$L_{avg} = \sum_{k=0}^7 l_2(r_k) p_r(r_k) = 2(0.19) + 2(0.25) + 2(0.21) + \\ 3(0.16) + 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02) \\ = 2.7 \text{ bits}$$

• The compression ratio: $C_R = \frac{3}{2.7} = 1.11$

• The relative Data Redundancy: $R_D = 1 - \frac{1}{1.11} = 0.099 \Rightarrow \sim \%10$

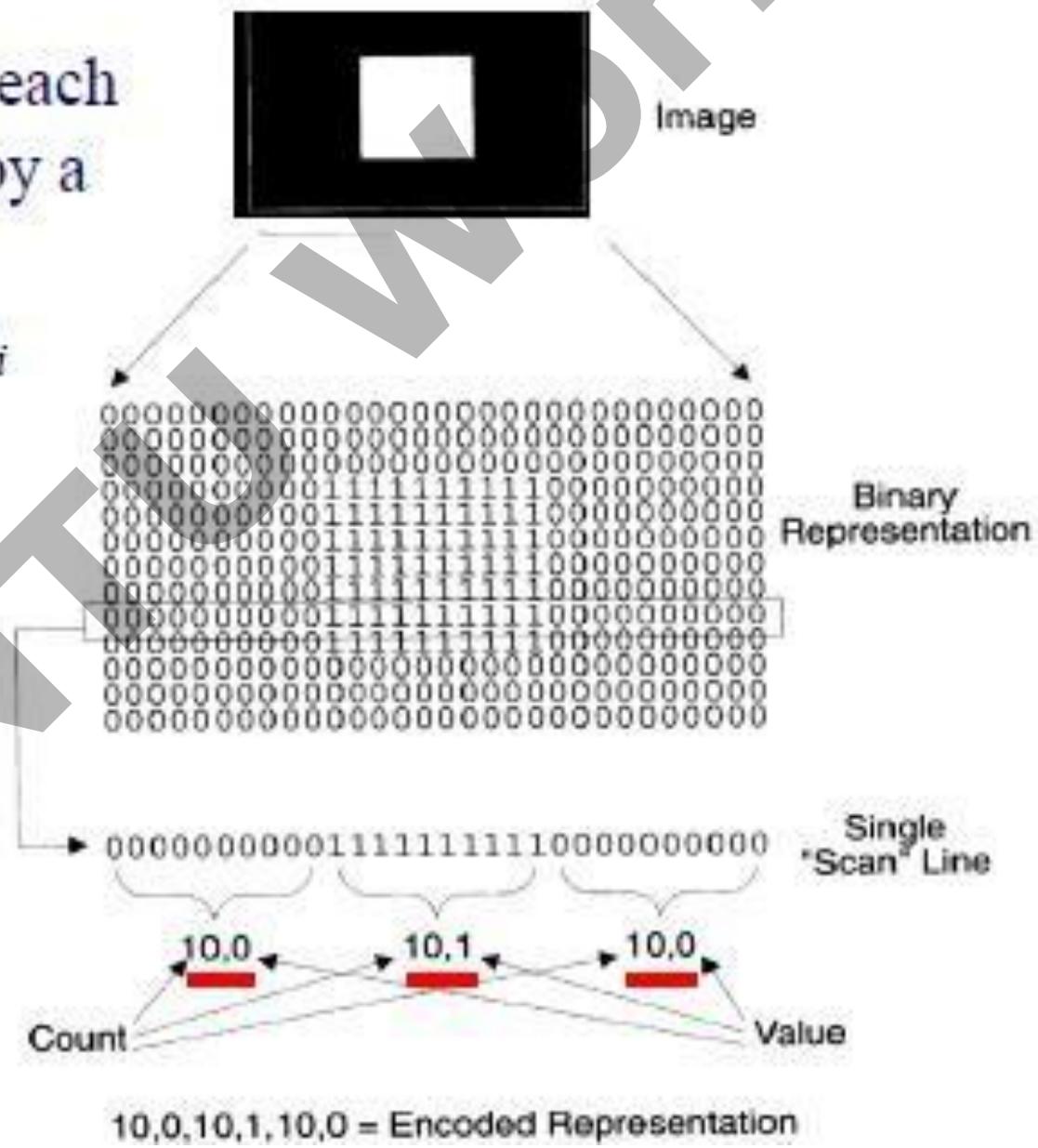
Inter pixel Redundancy or Spatial Redundancy

Pixel redundancy

- In run-length coding, each run R_i is represented by a pair (r_i, g_i) with

g_i = gray level of R_i

r_i = length of R_i



The gray level of a given pixel can be predicted by its neighbors and the difference is used to represent the image; this type of transformation is called **mapping**

Run-length coding can also be employed to utilize inter pixel redundancy in image compression

Removing inter pixel redundancy is lossless

Difference coding

$$f(x_i) = \begin{cases} x_i & \text{if } i=0, \\ x_{i+1} - x_i & \text{if } i>0 \end{cases}$$

Ex original:
code $f(x_i)$:

56	56	56	82	82	82	83	80	80	80	80
56	0	0	26	0	0	1	-3	0	0	0

- The code is calculated row by row.

Irrelevant information

One of the simplest ways to compress a set of data is to remove superfluous data. For images, information that is ignored by human visual system or is extraneous to the intended use of an image are obvious candidate for omission.

The “gray” image, since it appears as a homogeneous field of gray, can be represented by its average intensity alone – a single 8-bit value. Therefore, the compression would be

$$\frac{256 \cdot 256 \cdot 8}{8} = 65,536:1$$

Psychovisual Redundancy (EYE CAN RESOLVE 32 GRAY LEVELS ONLY)

The eye does not respond with equal sensitivity to all visual information. The method used to remove this type of redundancy is called **quantization** which means the mapping of a broad range of input values to a limited number of output values.

Fidelity criteria

$\hat{f}(x, y)$ Fidelity criteria is used to measure information loss and can be divided into two classes.

- 1) **Objective fidelity criteria** (math expression is used):
Measured mathematically about the amount of error in the reconstructed data.
- 2) **Subjective fidelity criteria:** Measured by human observation

Objective fidelity criteria:

When information loss can be expressed as a mathematical function of the input and output of the compression process, it is based on an objective fidelity criterion. For instance, a **root-mean-square** (rms) error between two images.

Let $f(x, y)$ be an input image and $\hat{f}(x, y)$ be an approximation of $f(x, y)$ resulting from compressing and decompressing the input image. For any values of x and y , the mean square error is

$$e_{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2$$

The root mean-square error:

$$e_{RMSE} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

the *mean-square signal-to-noise ratio* of the output image is defined as

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

The rms value of SNR is

$$SNR_{rms} = \sqrt{SNR_{ms}}$$

Subjective criteria:

While objective fidelity criteria offer a simple and convenient way to estimate information loss, images are viewed by humans. Therefore, measuring image quality by subjective evaluations of people is often more appropriate: show two images (original and decompressed) to a number of viewers and average their evaluations.

- **Subjective fidelity criteria:**

- A Decompressed image is presented to a cross section of viewers and averaging their evaluations.

- It can be done by using an absolute rating scale

Or

- By means of side by side comparisons of $f(x, y)$ & $f'(x, y)$.

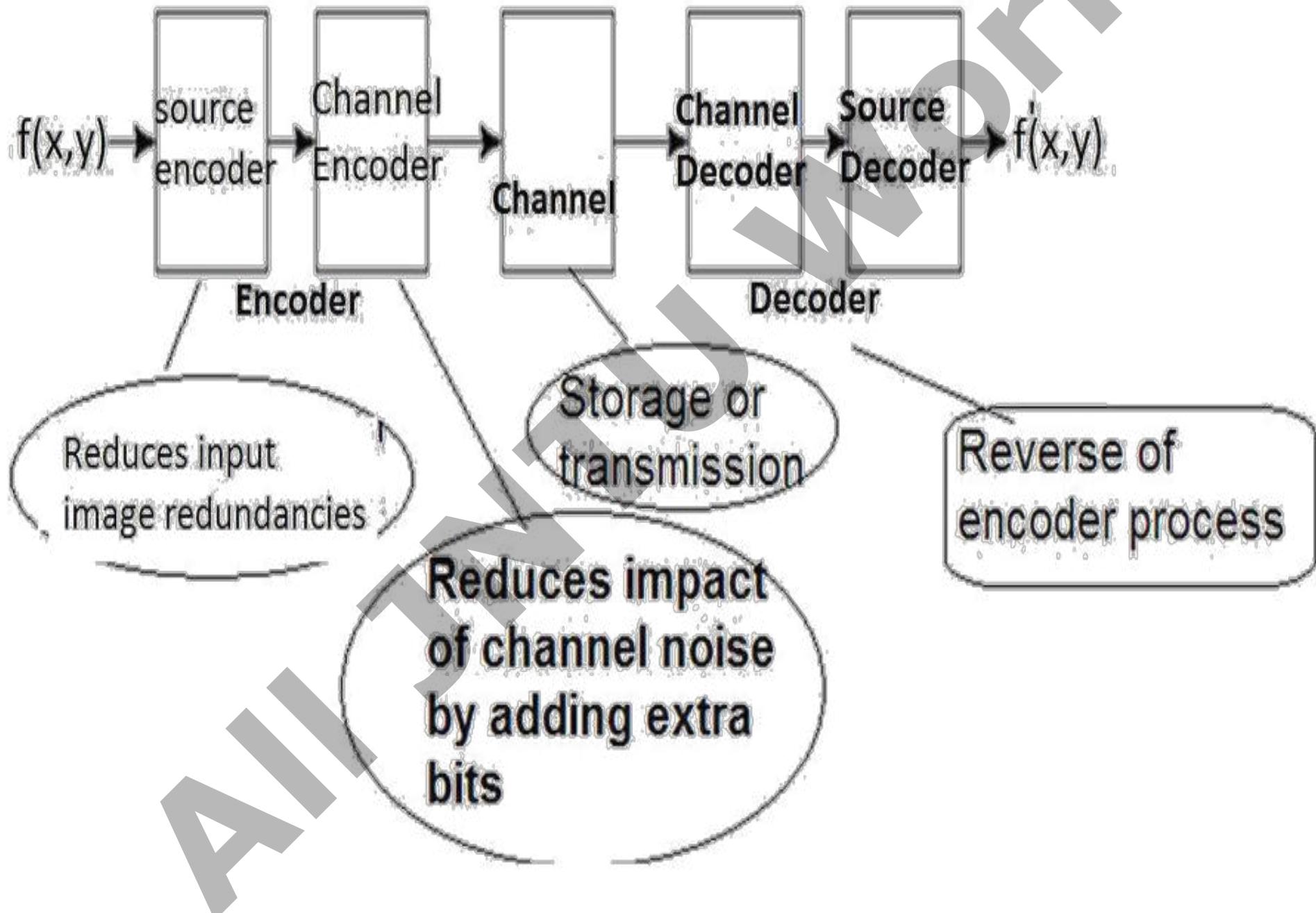
- Side by Side comparison can be done with a scale such as

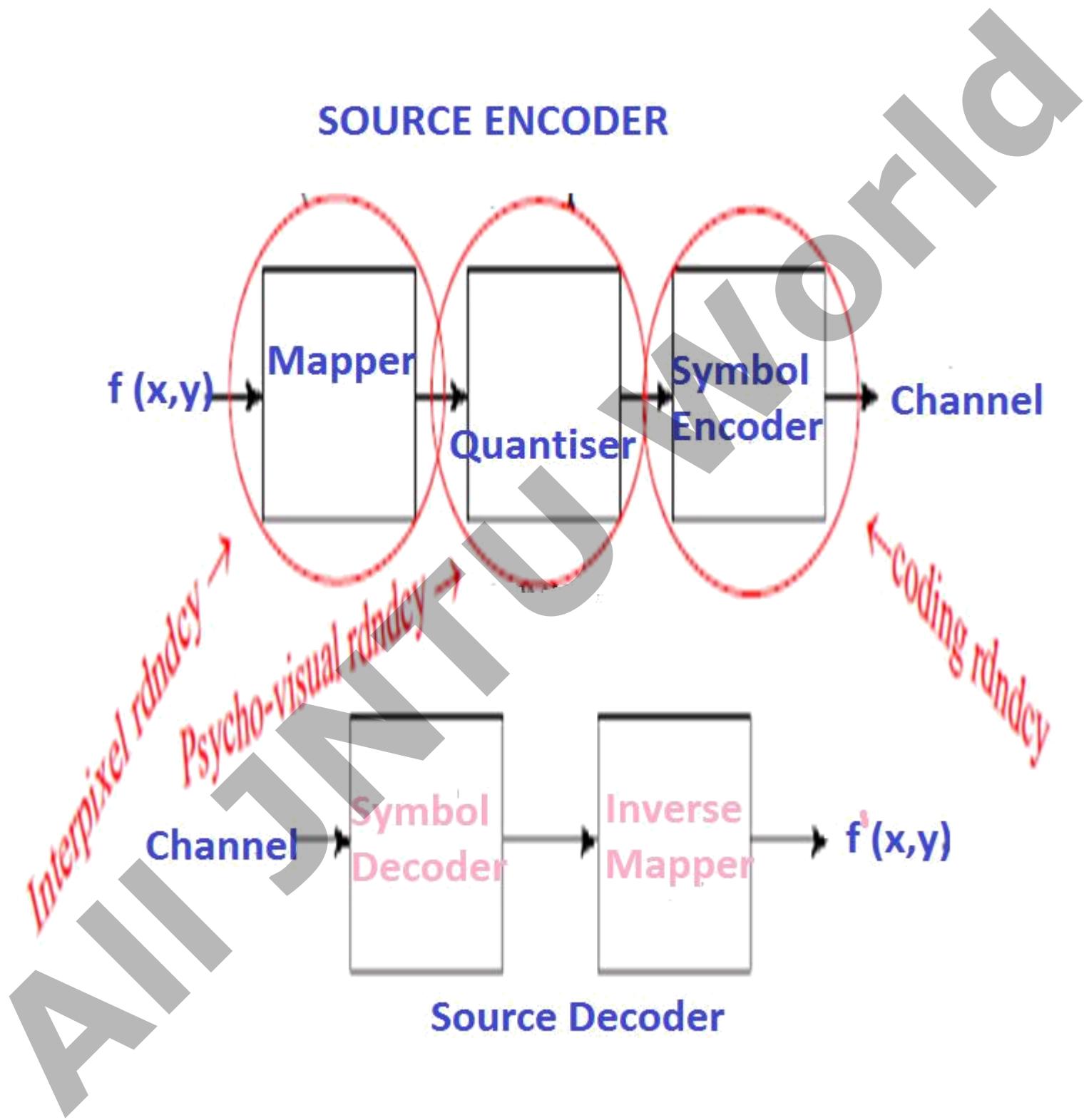
$$\{-3, -2, -1, 0, 1, 2, 3\}$$

to represent the subjective valuations

{much worse, worse, slightly worse, the same, slightly better, better, much better} respectively. One possible absolute rating scale: **Excellent, fine, average, poor**

A General Compression System Model





All the three stages are present in every compression model.

If error free compression is needed, Quantizer part will be omitted. In predict compression system, Quantizer + Mapper = Single Block

Mapper: Transforms the image into array of coefficients reducing inter pixel redundancies. This is a reversible process which is not lossy. Run-length coding is an example of mapping. In video applications, the mapper uses previous (and future) frames to facilitate removal of temporal redundancy.

•Quantizer: This process reduces the accuracy and hence psycho visual redundancies of a given image is irreversible and therefore lossy.

•Symbol Encoder: Removes coding redundancy by assigning shortest codes for the most frequently occurring output values.

Huffman Coding:

Source reduction process:

ORIGINAL SOURCE		SOURCE REDUCTION			
Symbol	Probability	1	2	3	4
a2	0.4	0.4	0.4	0.4	0.6
a6	0.3	0.3	0.3	0.3	0.4
a1	0.1	0.1	0.2	0.3	
a4	0.1	0.1	0.1		
a3	0.06	0.1			
a5	0.04				

Entropy of the source
or Source Reduction

$$E_Z = - \sum_{i=1}^n P(ai) \log P(ai)$$

$$\begin{aligned}
 &= - [0.4 \log (0.4) + 0.3 \log (0.3) + 0.1 \log (0.1) + 0.1 \log (0.1) + \\
 &0.06 \log (0.06) + 0.04 \log (0.04)] = 2.14
 \end{aligned}$$

Codeword construction process:

Original Source		Source Distination				
Sym.	Prob.	Code	1	2	3	4
a_1	0.4	1	0.4 1	0.4 1	0.4 1	0.6 0
a_6	0.3	00	0.3 01	0.3 00	0.3 00	0.4 1
a_1	0.1	011	0.1 011	0.2 010	0.3 01	
a_4	0.1	0100	0.1 0100	0.1 011		
a_3	0.06	01010	0.1 0101			
a_5	0.04	01011				

Pak Iak

Huffman Coding: Note that the shortest codeword (1) is given for the symbol/pixel with the highest probability (a2). The longest codeword (01011) is given for the symbol/pixel with the lowest probability (a5). The average length of the code is given by:

$$\begin{aligned}L_{\text{avg}} &= (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) \\&= 2.2 \text{ bits / symbol}\end{aligned}$$

Transform Coding: (lossy image compression)

In digital images the spatial frequencies are important as they correspond to important image features. High frequencies are a less important part of the images. This method uses a reversible transform (i.e. Fourier, Cosine transform) to map the image into a set of transform coefficients which are then quantized and coded.

Transform Selection: The system is based on discrete 2D transforms. The choice of a transform in a given application depends on the amount of the reconstruction error that can be tolerated and computational resources available.

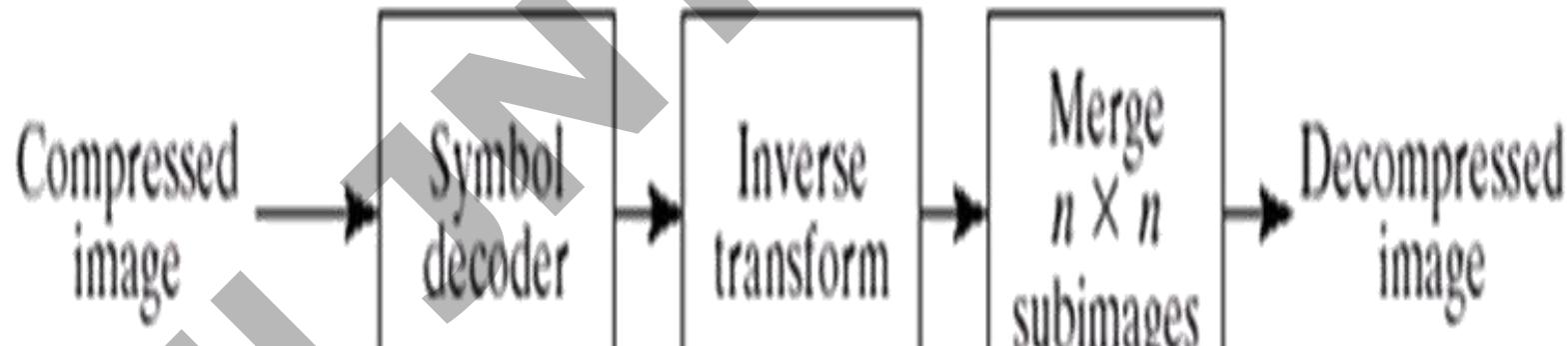
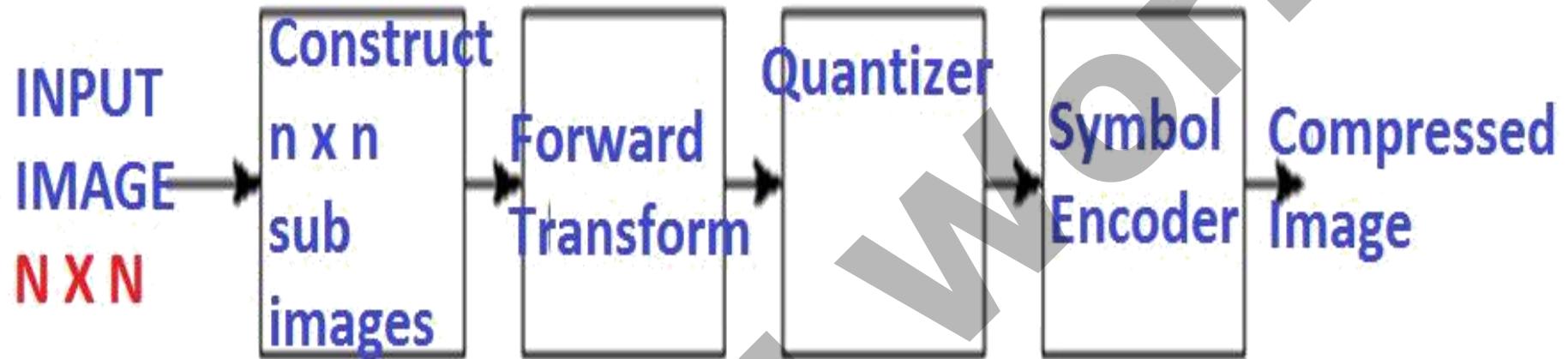
- Consider a sub image NxN image $f(x,y)$, where the forward discrete transform $T(u,v)$ is given by:

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)g(x,y,u,v)$$

• For $u, v=0,1,2,3,..,N-1$

General scheme

- The transform has to decorrelate the pixels or to compact as much information as possible into the smallest number of transform coefficients
- The quantization selectively eliminates or more coarsely quantizes the less informative coefficients
- Variable-length coding eliminates the remained coding redundancy



A Transform Coding System

DCT

$$c(u, v) = \alpha(u) \alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left\{ f(x, y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \right\}$$
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u=0, \\ \sqrt{\frac{2}{N}} & \text{if } u=1, 2, \dots, N-1 \end{cases}$$

Same for $\alpha(v)$

Inverse DCT

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \left\{ \alpha(u) \alpha(v) c(u, v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \right\}$$

JPEG Standard

JPEG exploits spatial redundancy

Objective of image compression standards is to enhance the **interoperability and compatibility** among compression systems by different vendors.

JPEG Corresponds to ISO/IEC international standard 10928-1, digital compression and coding of continuous tone still images.

JPEG uses DCT.

JPEG became the Draft International Standard in 1991 and International standard IS in 1992.

JPEG Standard

Different modes such as sequential, progressive and hierarchical modes and options like lossy and lossless modes of the JPEG standards exist.

JPEG supports the following modes of encoding

Sequential : The image is encoded in the order in which it is scanned. Each image component is encoded in a single left-to-right, top-to-bottom scan.

Progressive : The image is encoded in multiple passes. (web browsers). Group DCT coefficients into several spectral bands. Send low-frequency DCT coefficients first AND Send higher-frequency DCT coefficients next

Hierarchical : The image is encoded at multiple resolutions to accommodate different types of displays.

JPEG(Joint Photographic Experts Group)

Applications : color FAX, digital still camera, multimedia computer, internet

JPEG Standard consists of

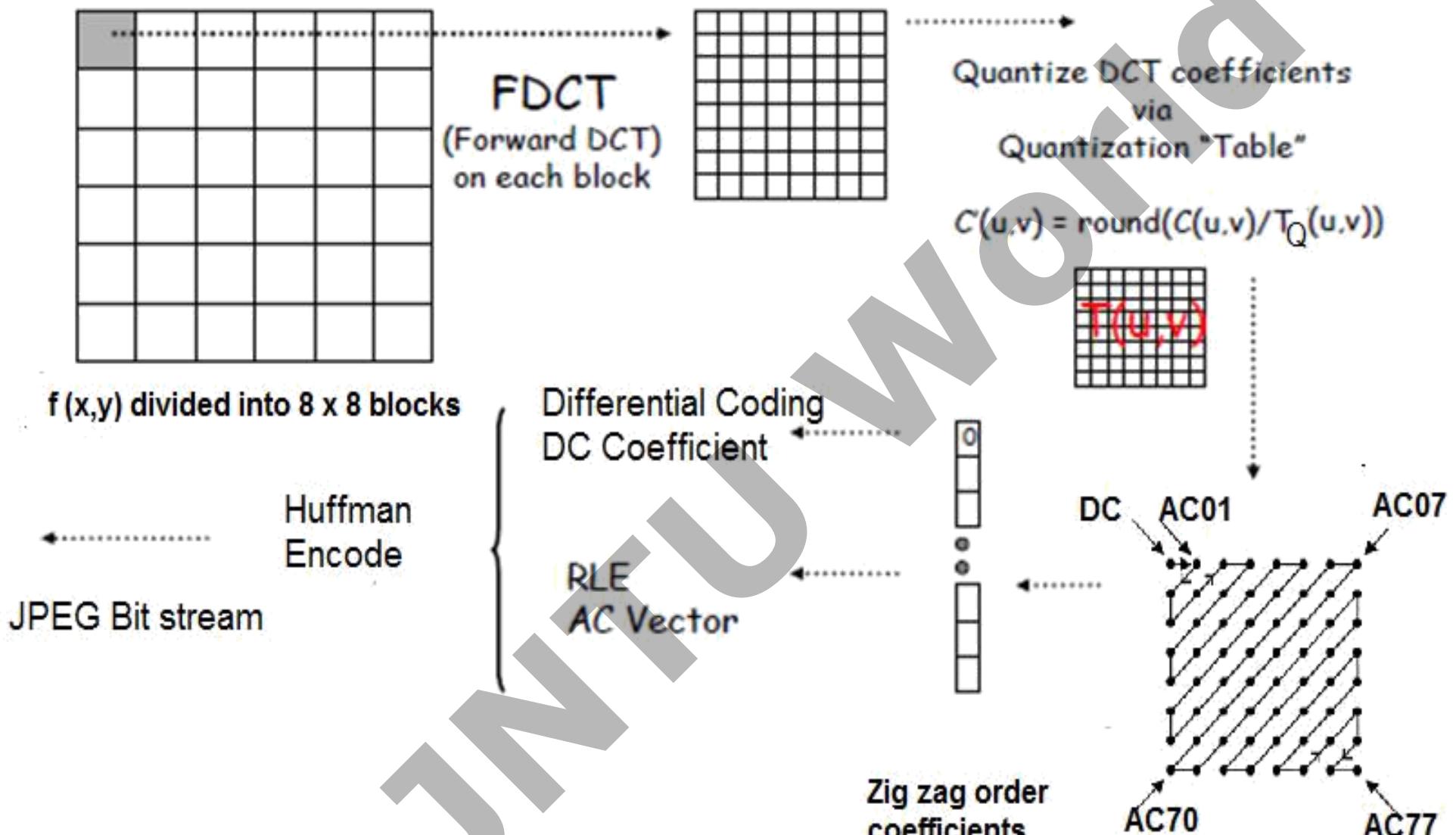
Algorithm: DCT + quantization + variable length coding

Steps in JPEG Compression

- Divide the file into 8 X 8 blocks.
- Apply DCT. Transform the pixel information from the spatial domain to the frequency domain with the Discrete Cosine Transform.
- Each value in the spectrum is divided by the matching value in the quantization table, and the result rounded to the nearest integer.
- Modified spectrum is converted from an 8x8 array into a liner sequence .Look at the resulting coefficients in a zigzag

order. Do a run-length encoding of the coefficients ordered in this manner.

Follow by Huffman coding.



The coefficient with zero frequency is called DC coefficients, and the remaining 63 coefficients are AC coefficients.

For JPEG decoding, reverse process is applied

JPEG2000 (J2K)

In JPEG 2000, Instead of the DCT transformation, JPEG 2000, ISO/IEC 15444, uses the **Wavelet transformation**.

The advantage of JPEG 2000 is that the blockiness of JPEG is removed, but replaced with a more overall fuzzy picture,

H.261/H.263 originally designed for video-conferencing over telephone lines, i.e. low bandwidth

Allows to extract various sub-images from a single compressed image code stream, the so called “*Compress Once, Decompress Many Ways*”.

JPEG2000 (J2K)

Better efficiency, and more functionality

Multiple resolution

Large images

Single decompression architecture

Spatial Scalability:

Multi-resolution decoding from one bit-stream

Blockiness of JPEG is removed,

The compression ratio for JPEG 2000 is higher than for JPEG

Discrete Wavelet Transform (DWT)

Embedded Block Coding with Optimized Truncation (EBCOT)

Applications of JPEG-2000 and their requirements

- Internet
- Color facsimile
- Printing
- Scanning
- Digital photography
- Remote Sensing
- Mobile
- Medical imagery
- Digital libraries and archives
- E-commerce

Each application area has some requirements which the standard should fulfill.

Improved low bit-rate performance: It should give acceptable quality below 0.25 bpp. Networked image delivery and remote sensing applications have this requirements.

Progressive transmission: The standard should allow progressive transmission that allows images to be reconstructed with increasing pixel accuracy and resolution.

Region of Interest Coding: It should preferentially allocate more bits to the regions of interest (ROIs) as compared to the non-ROI ones.

Content based description: Finding the desired image from a large archive of images is a challenging task. This has applications in medical images, forensic, digital libraries etc. These issues are being addressed by MPEG-7.

- Image Security: Digital images can be protected using watermarking, labeling, stamping, encryption etc.

DWT for Image Compression

2D Discrete
Wavelet
Transform

Quantization

Entropy
Coding

2D discrete wavelet transform
(1D DWT applied alternatively to horizontal and vertical direction line by line) converts images into “sub-bands” Upper left is the DC coefficient
Lower right are higher frequency sub-bands.

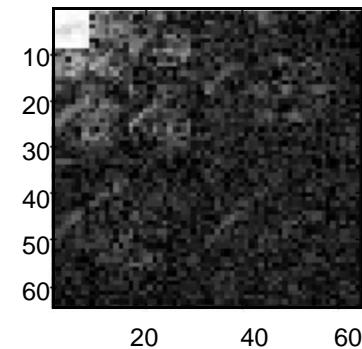
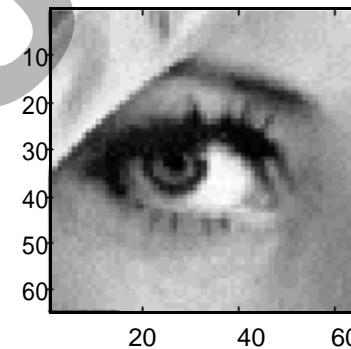


Image decomposition Scale 1

4 subbands : LL1, LH1, HL1, HH1

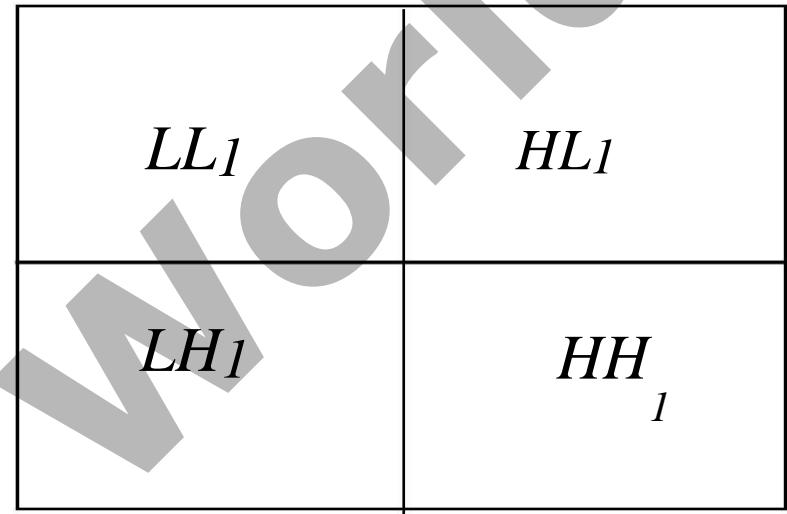


Image decomposition Scale 2

4 subbands : LL2, LH2, HL2, HH2

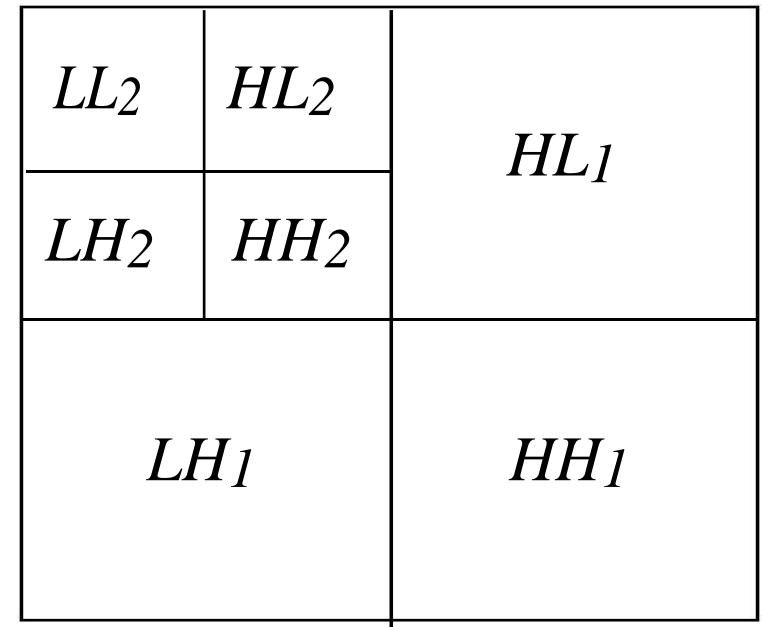


Image Decomposition

Parent

Children

Descendants: corresponding
coeff. at finer scales Ancestors:
corresponding coeff. at coarser
scales

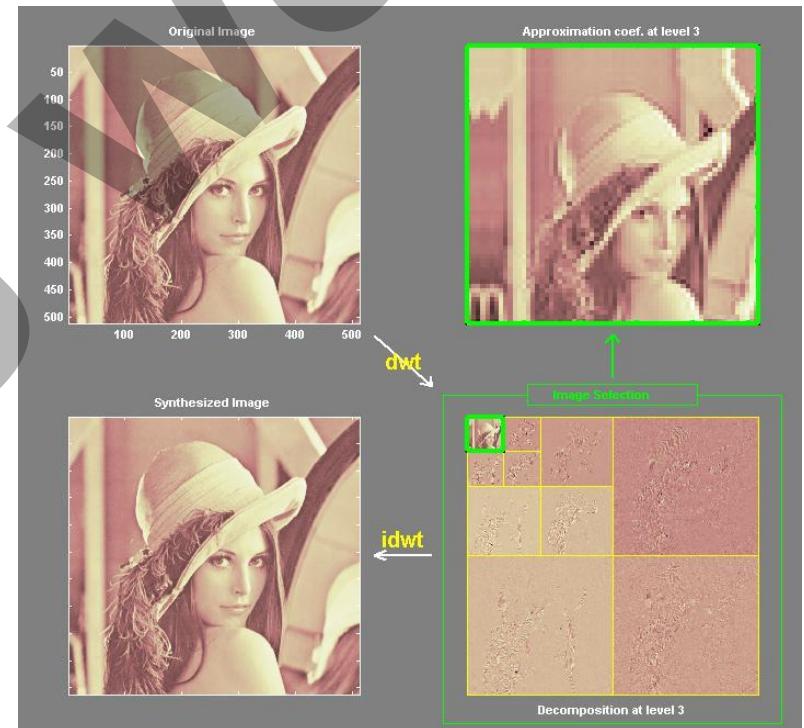
Image Decomposition

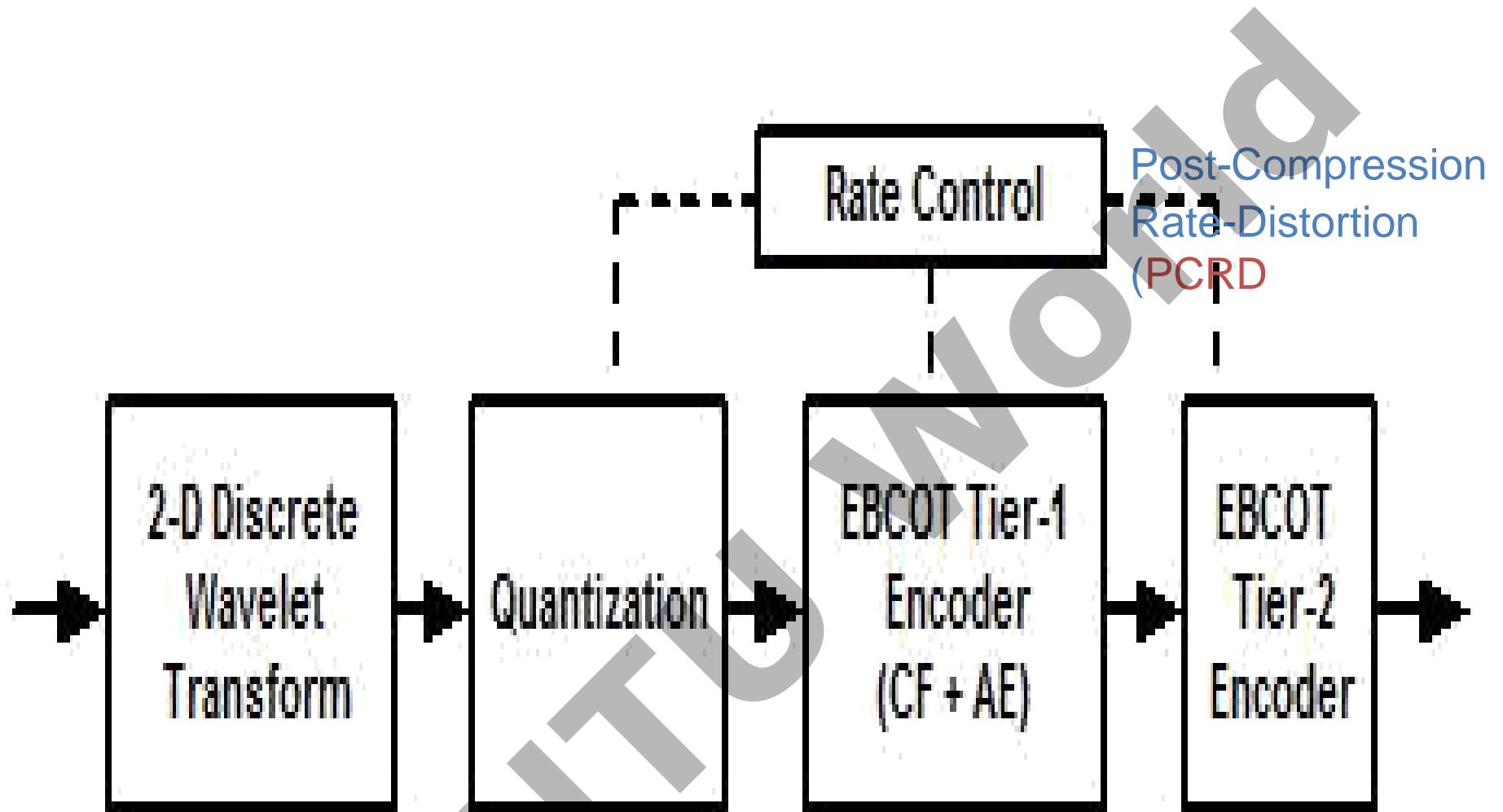
Feature 1:

Energy distribution similar to other
TC: Concentrated in low frequencies

Feature 2:

Spatial self-similarity
across subbands





transform

quantize

coding

J2K uses 2-D Discrete Wavelet Transformation (DWT)

Embedded Block Coding with Optimized Truncation of bit-stream (EBCOT), which can be applied to wavelet packets and which offers both resolution scalability and SNR scalability.

Each sub band is partitioned into small non-overlapping block of samples, known as *code blocks*. *EBCOT generates an embedded bit-stream for each code block. The bit-stream associated with each code block may be truncated to any of a collection of rate-distortion optimized truncation points*

Steps in JPEG2000

Tiling:

Smaller non-overlapping blocks of image are known as tiles

The image is split into *tiles*, rectangular regions of the image. Tiles can be any size.

Dividing the image into tiles is advantageous in that the decoder will need less memory to decode the image and it can opt to decode only selected tiles to achieve a partial decoding of the image.

Wavelet Transform: Either CDF 9/7 or CDF 5/3 bi-orthogonal wavelet transform.

Quantization: Scalar quantization

All operations, such as component mixing, DWT, quantization and entropy coding are therefore done independently for each tile.

Coding:

The quantized subbands are split into *precincts*, rectangular regions in the wavelet domain. They are selected in a way that the coefficients within them across the sub-bands form approximately spatial blocks in the image domain. Precincts are split further into *code blocks*. Code blocks are located in a single sub-band and have equal sizes. The encoder has to encode the bits of all quantized coefficients of a code block, starting with the most significant bits and progressing to less significant bits by *EBCOT* scheme.

'Compress Once, Decompress Many Ways'



A Single Original
Codestream

By resolutions



By layers



Region of Interest



MPEG1 MOVING PICTURE EXPERT GROUP

MPEG exploits temporal redundancy. Prediction based.

Compare each frame of a sequence with its predecessor and only pixels that have changed are updated,

MPEG-1 standard is for storing and retrieving video information on digital storage media.

MPEG-2 standard is to support digital video broadcasting, HDTV systems.

H.261 standard for telecommunication applications

MPEG1 COMPRESSION ALGORITHM: MPEG Digital Video Technology

Temporal compression algorithm: Temporal compression algorithm relies on similarity between successive pictures using prediction in motion compensation

Spatial compression algorithm: relies upon redundancy within small areas of a picture and is based around the DCT transform, quantization and entropy coding techniques.

MPEG-1 was up to 1.5 Mbit/s. MPEG-2 typically over 4MBit/s but can be up to 80 Mbit/s.

MPEG-1(ISO/IEC 11172) and MPEG-2(ISO/IEC 13818), MPEG-4(ISO/IEC 14496)

MPEG-1 : Digital Storage Media (CD-ROM...)

MPEG-2 : Higher bit rates and broader generic applications

(Consumer electronics, Telecommunications, Digital Broadcasting, HDTV, DVD, VOD, etc.)

Coding scheme :

Spatial redundancy : DCT + Quantization

Temporal redundancy : Motion estimation and compensation
Statistical redundancy : VLC

Applications :

Internet Multimedia, Wireless Multimedia Communication

Multimedia Contents for Computers and Consumer Electronics
Interactive Digital TV

Coding scheme :

Spatial redundancy : DCT + Quantization, Wavelet Transform

Temporal redundancy : Motion estimation and compensation

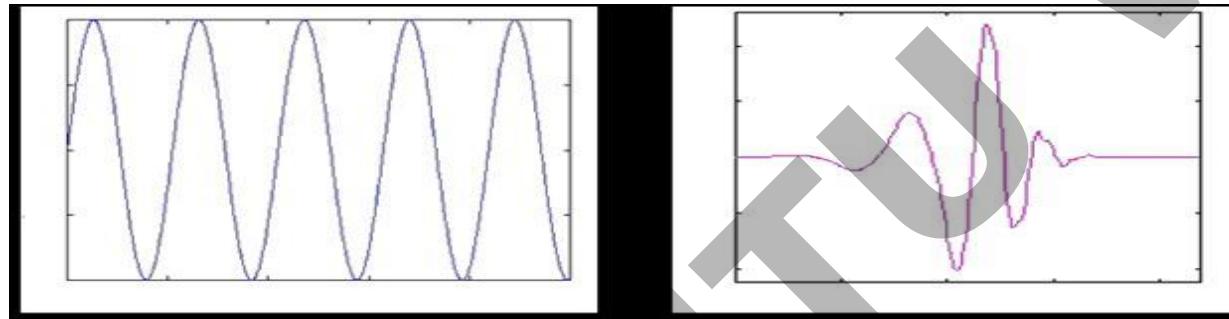
Statistical redundancy : VLC (Huffman Coding, Arithmetic Coding)
Shape Coding : Context-based Arithmetic Coding

Wavelets are functions that “wave” above and below the x-axis,

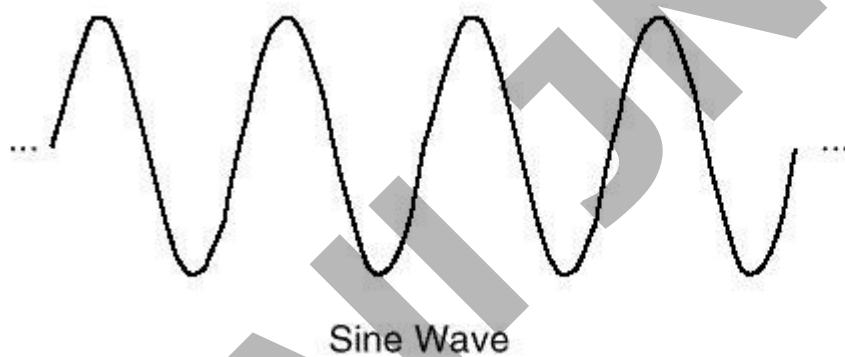
(1) varying frequency, (2) limited duration, and (3) an average value of zero.

This is in contrast to sinusoids, used by FT, which have infinite energy.

Like sines and cosines in FT, wavelets are used as **basis** functions $\psi_k(t)$



COMPARISON OF THE
SINE WAVE AND THE
WAVELET



- A wavelet is a waveform of effectively limited duration that has an average value of zero.

Wavelet functions

Wavelet Ψ (mother wavelet),

Scaling Function (Father wavelet): Translation, Dilation

These two functions generate a family of functions that can be used to break up or reconstruct a signal

The scaling function is given as

$$(x, y) \underset{j,mn}{2^{j/2}} (2^j x m, 2^j y n)$$

The orthonormal basis or wavelet basis is $\underset{i}{\{j,mn, (x, y) 2^{j/2} \underset{i}{(2^j x m, 2^j y n)}\}}$

$$\underset{i}{\{HV,, D\}}$$

where ψ is called the wavelet function and j and m, n are integers that scale and dilate the wavelet function. The factor ' j ' is known as the scale index, which indicates the wavelet's width. The location index m,n provides the position. The wavelet function is dilated by powers of two and is translated by the integer m,n

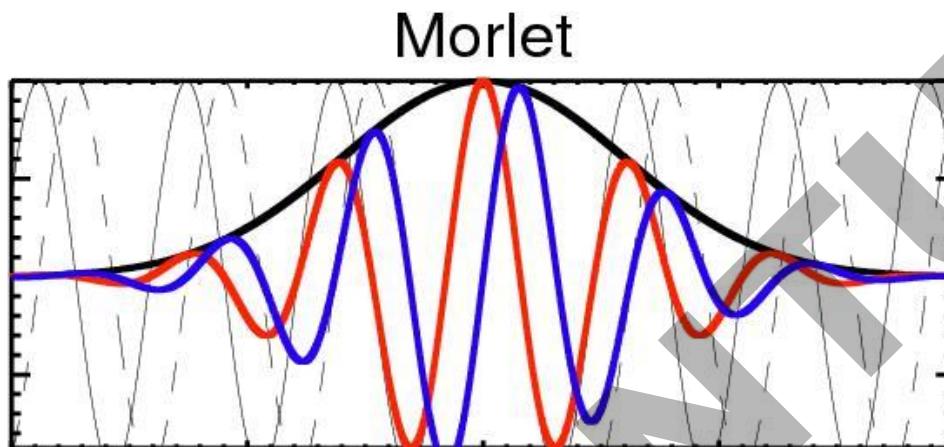
HVD stands for Horizontal vertical and Diagonal

Some Continuous Wavelets

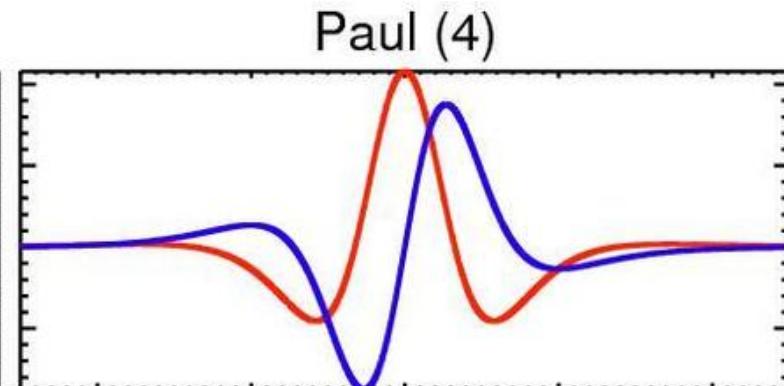
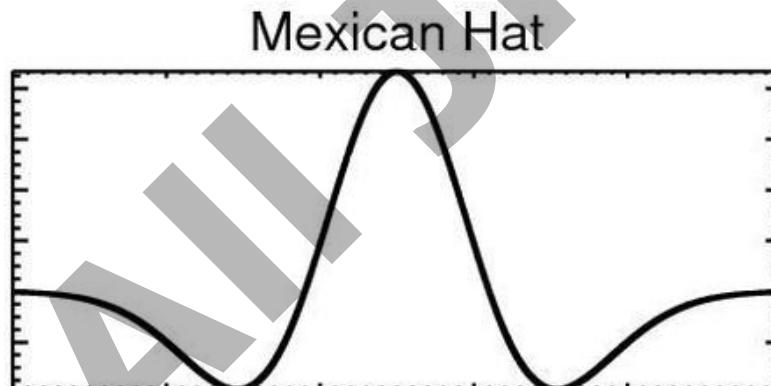
$$\text{Morlet} \quad \Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} e^{i\omega t} e^{-\frac{(t-\tau)^2}{s}}$$

Gabor

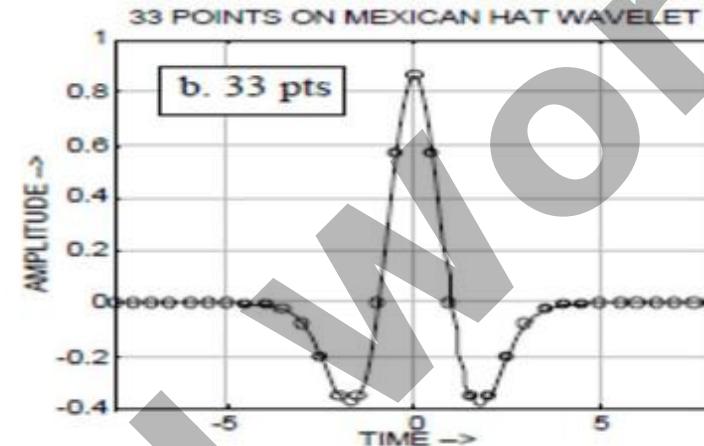
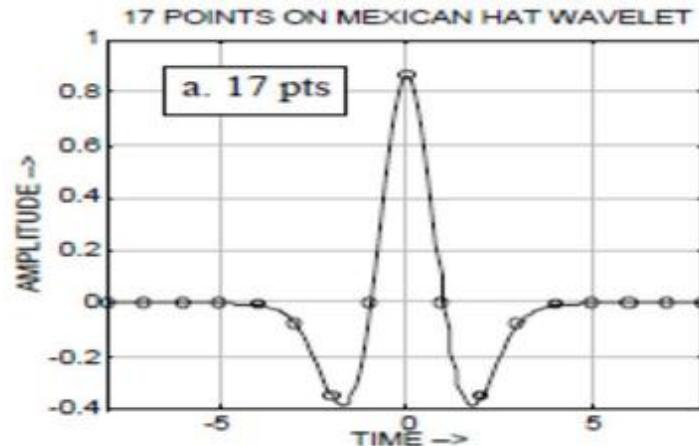
$$g_{a,\tau}(t) \equiv g(t-\tau) e^{-i2\pi a t}$$



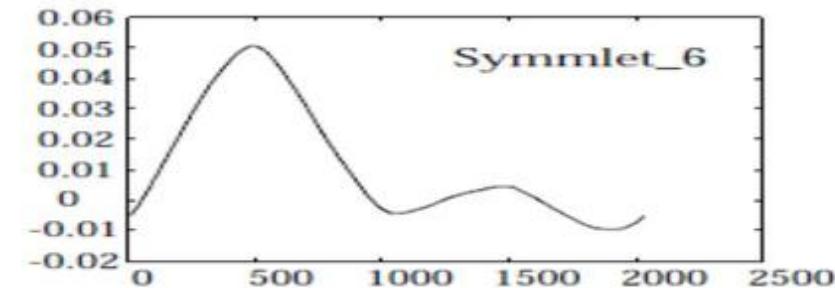
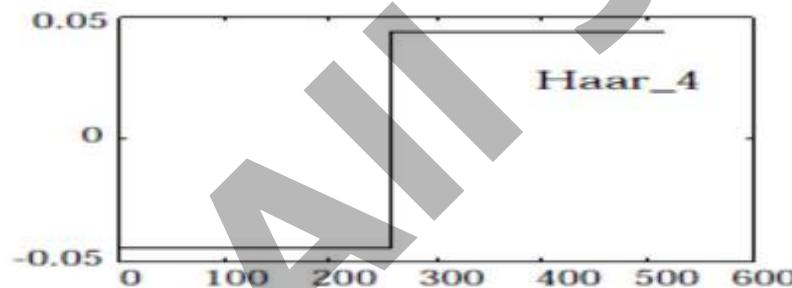
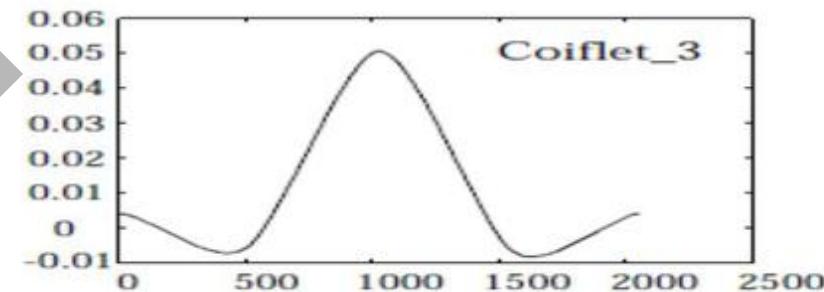
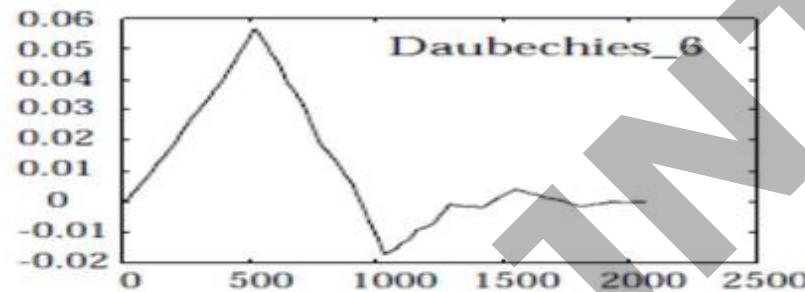
- Haar or Daubechies 1
- Daubechies 2, 3, 4, 5, 6, and 10
- Symmetric 2, 3, 4, and 5
- Biorthogonal 3.3, 3.5, 3.7, 3.9, 4.4, 5.5, and 6.8
- Coiflet 3, 4, and 5 • Morlet
- Meyer
- Mexican hat
- chirplet



Crude wavelets are generated from mathematical expression.
 To use them with digital signal, the crude wavelets are to be converted to wavelet filters having number of discrete equal distant points

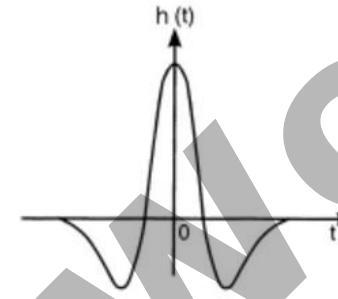


"Crude" Mexican Hat Wavelet with 17 points (a) then 33 wavelet filter points superimposed on the continuous (crude) representation (a then b). Although the defining equation describes an infinite, continuous waveform, by using equispaced discrete points we have created discrete, finite-length filters ready for use with digital computers.



Two popular wavelets for CWT are the Mexican hat and Morlet.
The Mexican hat wavelet is the second derivative of the Gaussian function,
given as

$$mexh(t) = \left\{ 2 / (\sqrt{3} \pi^{-1/4}) \right\} (1-t^2) e^{-t^2/2}$$



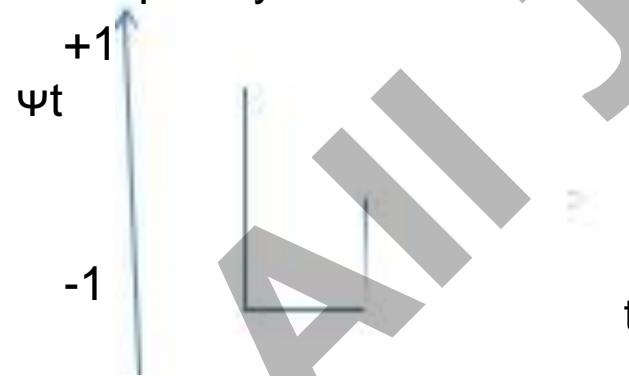
The two-dimensional Mexican hat wavelet is well known as the Laplacian operator, widely used for zero-crossing image edge detection.

The Morlet wavelet is given by

$$\Psi(t) = \pi^{-1/4} e^{-i w_0 t} e^{-t^2/2}$$

Haar Wavelet

The Haar wavelet is a bipolar step function, localised in time domain, poorly localised in frequency domain.



$$h(t) = \begin{cases} 1 & \text{when } 0 < t < 1/2 \\ -1 & \text{when } 1/2 < t < 1 \\ 0 & \text{otherwise} \end{cases}$$

The wavelet and scaling coefficients are related by the quadrature mirror relationship,

$$g^n = (-1)^n h_{1-n+N}$$

The term N is the number of vanishing moments

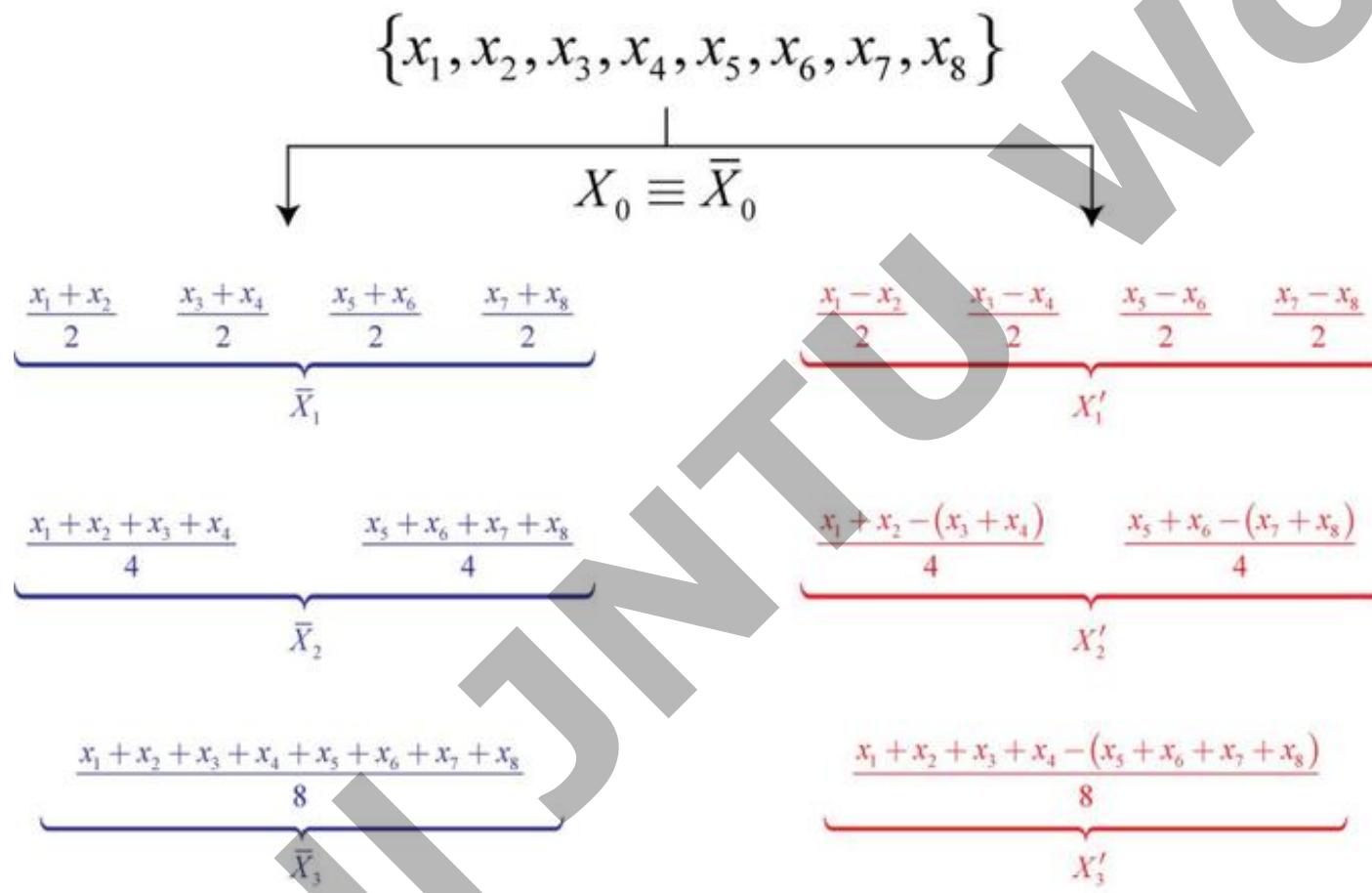
Wavelet families and their properties

Wavelet Family	Filters length	Number of vanishing moments, N
Haar	2	1
Daubechies M	$2M$	M
Coiflets M	$6M$	$2M-1$
Symlets	$2M$	M

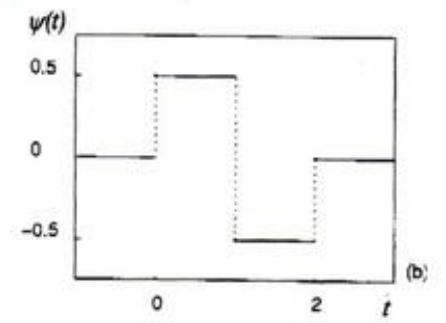
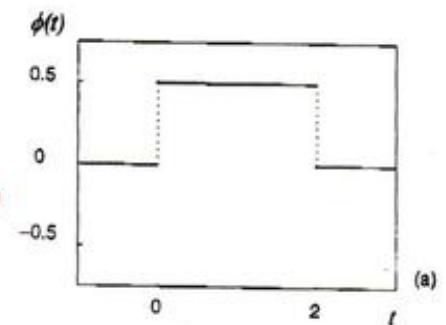
Wavelets can come in various shapes and sizes by stretching and shifting



Multiscale analysis – 1D example



Haar Wavelet
Multiscale Filter:



Example

$$\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$$

$$c^2$$

$$d^2$$

Multiplication by $\sqrt{2}$ is needed to ensure energy conservation

$$\sqrt{2}$$

Energy Concerns

Energy of signals

$$\mathcal{E}_f = f_1^2 + f_2^2 + \cdots + f_N^2$$

$$\mathcal{E}_f = 4^2 + 6^2 + \cdots + 5^2 = 446$$

- The 1-level Haar transform conserves energy

$$\mathcal{E}_{(c^2 | d^2)} = 25 \cdot 2 + 121 \cdot 2 + \cdots + 2 + 0 = 446$$

$$\mathcal{E}_{c^2} = 25 \cdot 2 + 121 \cdot 2 + 49 \cdot 2 + 25 \cdot 2 = 440$$

$$\mathcal{E}_{d^2} = 2 + 2 + 2 + 0 = 6$$

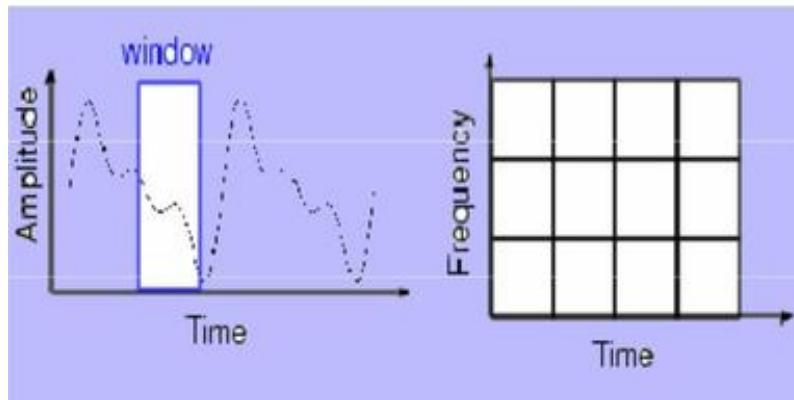
Proof of Energy Conservation

$$\begin{aligned} c_1^2 + d_1^2 &= \left[\frac{f_1 + f_2}{\sqrt{2}} \right]^2 + \left[\frac{f_1 - f_2}{\sqrt{2}} \right]^2 \\ &= \frac{f_1^2 + 2f_1f_2 + f_2^2}{2} + \frac{f_1^2 - 2f_1f_2 + f_2^2}{2} \\ &= f_1^2 + f_2^2. \end{aligned}$$

SHORT TIME FOURIER TRANSFORM (STFT)

To analyze only a small section of the signal at a time -- a technique called *Windowing the Signal* is used.

The Segment of Signal is Assumed Stationary



$$\text{STFT}_X^{(\omega)}(t', f) = \int [x(t) \cdot \omega^*(t-t')] \cdot e^{-j2\pi ft} dt$$

$\omega(t)$: the window function

A function of time and frequency

- Time/Frequency localization depends on window size.

Once you choose a particular window size, it will be the same for all frequencies

Use **narrower** windows at **high frequencies** for better time resolution.

Use **wider** windows at **low frequencies** for better frequency resolution.

Wavelet transforms overcomes the preset resolution problem of the STFT by using a **variable length window**:

Wavelet Transform – Multi Resolution Analysis

Wavelet transform is capable of providing the time and frequency information simultaneously,

- Similar to STFT: signal is multiplied with a function
- “The Forest & the Trees”
Notice gross features with a large “window” Notice small features with a small “window”

Width of the Window is Changed as the Transform is Computed for Every Spectral Components

Split Up the Signal into a Bunch of Signals Representing the Same Signal, but all Corresponding to Different Frequency Bands Only Providing What Frequency Bands Exists at What Time Intervals

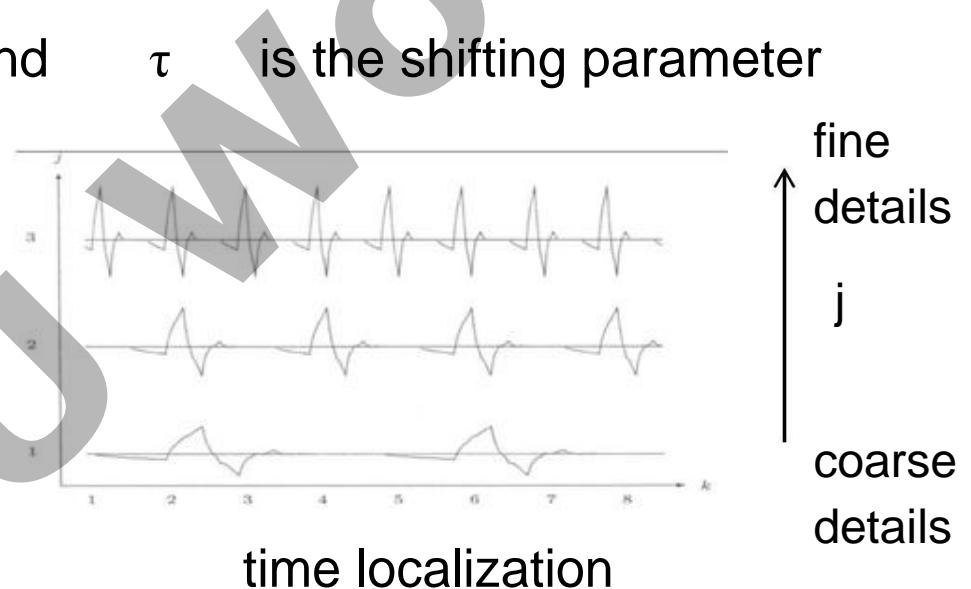
What is wavelet transform?

- Wavelet transform decomposes a signal using a set of basis functions (wavelets)
- Wavelets are obtained from a single prototype wavelet $\Psi(t)$ called mother wavelet by dilations and shifting:

where s is the scaling parameter and τ is the shifting parameter

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \Psi\left(\frac{t-\tau}{s}\right).$$

scale/
frequency
localization



• 2D function

$$\Psi_{s,\tau_x,\tau_y}(x,y) = \frac{1}{|s|} \Psi\left(\frac{x-\tau_x}{s}, \frac{y-\tau_y}{s}\right)$$

$$s^{-2_j}$$
$$k^{-2_j}$$

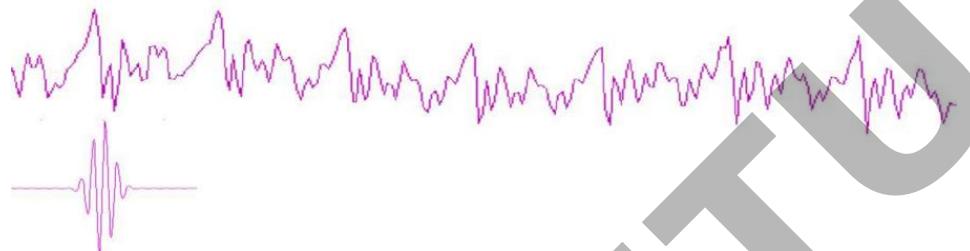
For easier calculation we can have discrete continuous signal

Scale = $1/j = 1/\text{Frequency}$

Wavelet has pseudo frequency in the sense that the frequency varies over its length. **Scaling, Stretching, Dilation (opposite is compression, Shrinking)** all refer to the frequency (pseudo) for expanding or shrinking the wavelet in time axis.

In Digital Signal Processing, scaling means changing amplitude.

Low scale or small scale → a Compressed wavelet → Rapidly changing details → High frequency → detailed $S < 1$: compress the signal



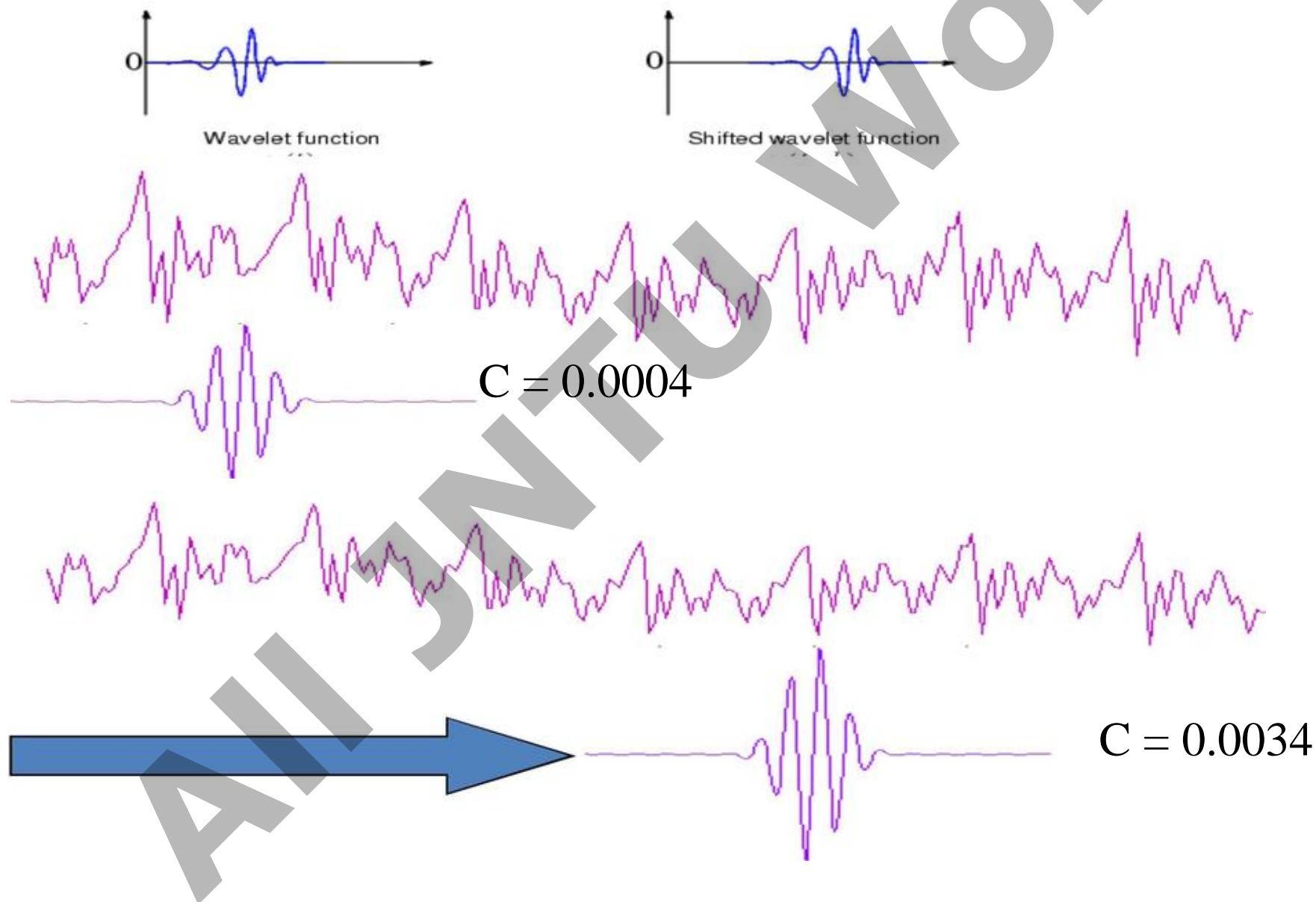
High scale or large scale → a Stretched wavelet → Slowly changing, coarse features → Low frequency → Non detailed $S > 1$: dilate the signal



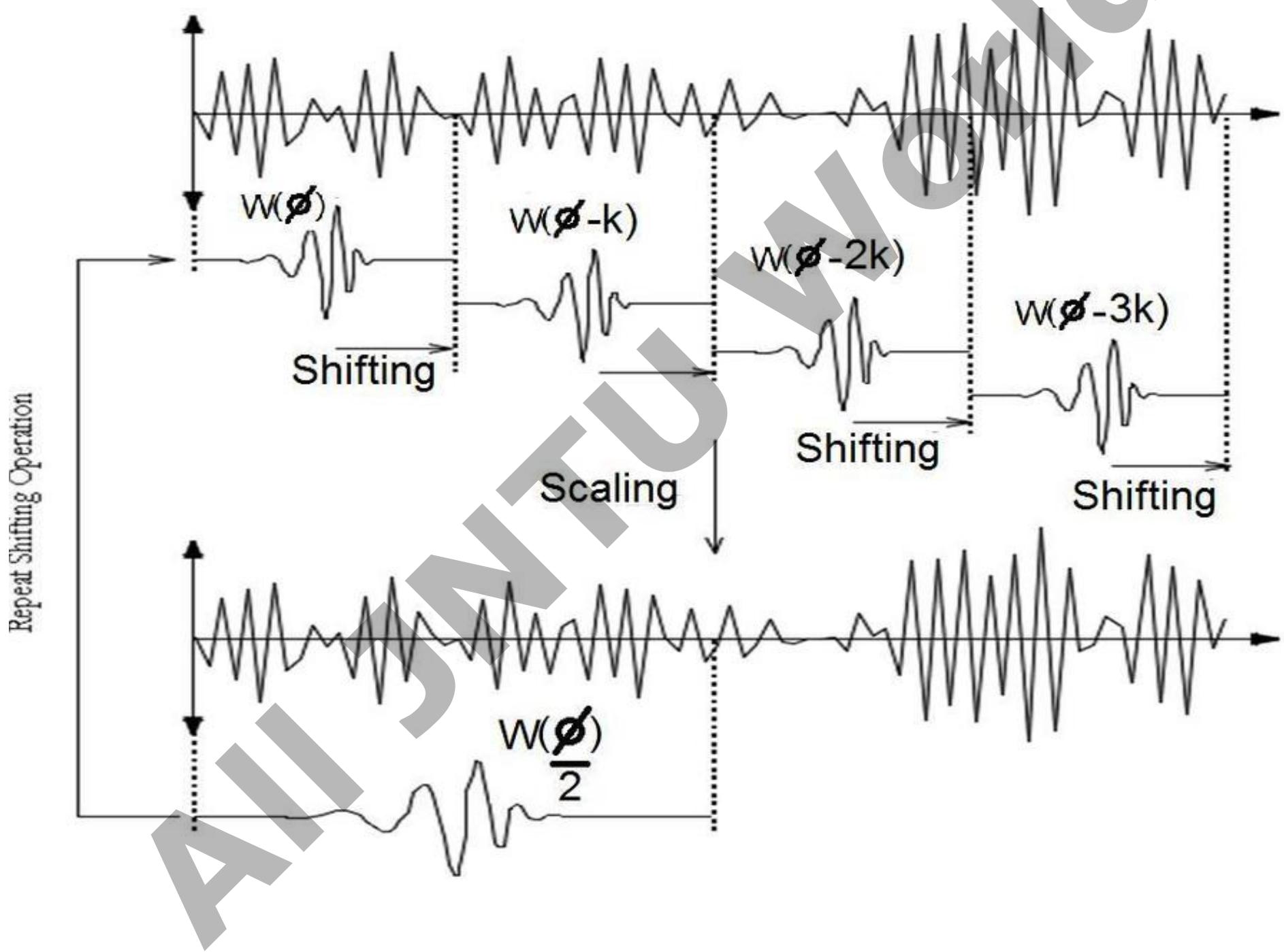
Shifting, Sliding or Translation: shifting the wavelet in time axis

Shifting a wavelet simply means delaying (or hastening) its onset.

Mathematically, delaying a function $f(t)$ by k is represented by $f(t-k)$



SCALING AND SHIFTING PROCESS OF THE DWT



DEFINITION OF CONTINUOUS WAVELET TRANSFORM

There are two main differences between the STFT and the CWT:

The width of the window is changed as the transform is computed for every single spectral component, which is probably the most significant characteristic of the wavelet transform.

The Continuous Wavelet Transform (CWT) is
the scalar product of the original signal with a translated and
dilated version of a locally confined function, called wavelet .

Therefore, the CWT of a function depends on two parameters,

ONE for translation , shifts the wavelet for local information and
OTHER for dilation controls the window size in which the signal analysis
must be performed .

2D Continuous Wavelet Transform

$$\Psi_{s,\tau_x,\tau_y}(x,y) = \frac{1}{\sqrt{s}} \iint f(x,y) \Psi \left[\frac{x-\tau_x}{s}, \frac{y-\tau_y}{s} \right]$$

The wavelets are generated from a single basic wavelet $\psi(t)$, the so-called *mother wavelet*, by scaling and translation

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \Psi \left(\frac{t-\tau}{s} \right)$$

where $*$ denotes complex conjugation. Scale = $1/j =$

s is the scale factor, τ is the translation factor and the factor s is for energy normalization across the different scales.

CWT can be regarded as the inner product of the signal with a basis function

$$\Psi_{s,\tau}^*(t)$$

As seen in the above equation, the transformed signal is a function of two variables, τ and s , the **translation** and **scale** parameters, respectively. $\Psi(t)$ is the transforming function, and it is called **the mother wavelet**.

In terms of frequency, low frequencies (high scales) correspond to a global information of a signal (that usually spans the entire signal), whereas high frequencies (low scales) correspond to a detailed information of a hidden pattern in the signal (that usually lasts a relatively short time).

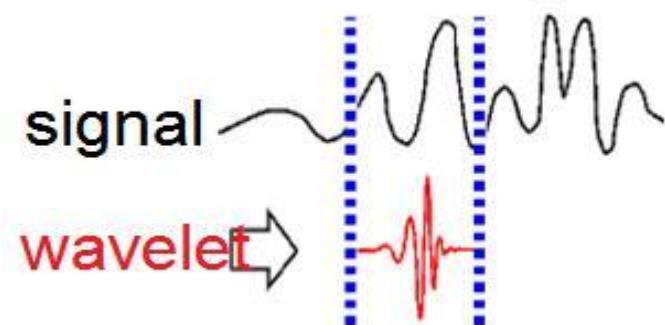
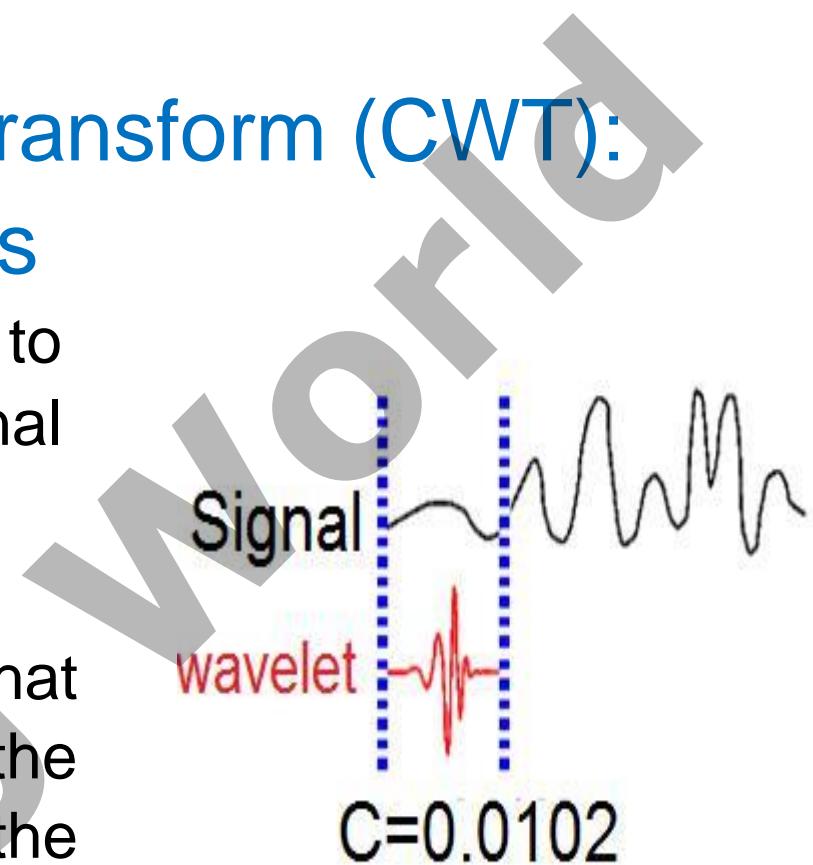
Scales $s > 1$ dilates the signals whereas

Scales $s < 1$, compresses the signal.

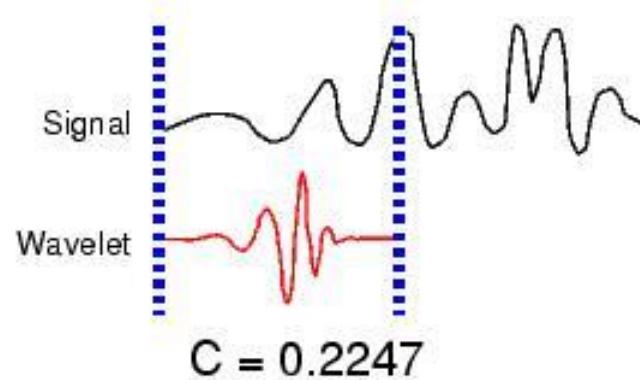
Continuous Wavelet Transform (CWT):

Main Steps

1. Take a wavelet and compare it to a section at the start of the original signal.
2. Calculate a number, C , that represents how closely correlated the wavelet is with this section of the signal. The higher C is, the more the similarity.
3. Shift the wavelet to the right and repeat steps 1 and 2 until you've covered the whole signal.



4. Scale the wavelet and repeat steps 1 through 3.



5. Repeat steps 1 through 4 for all scales

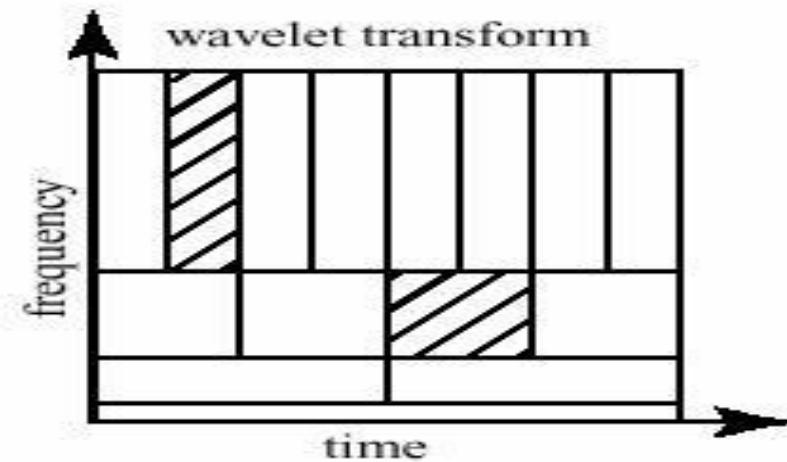
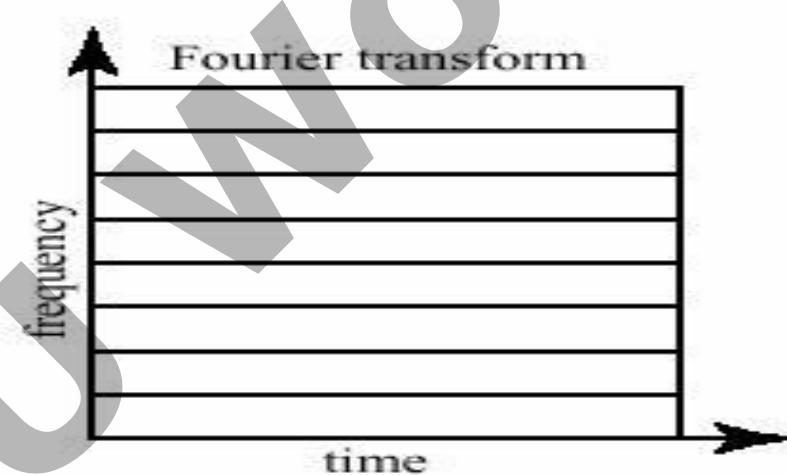
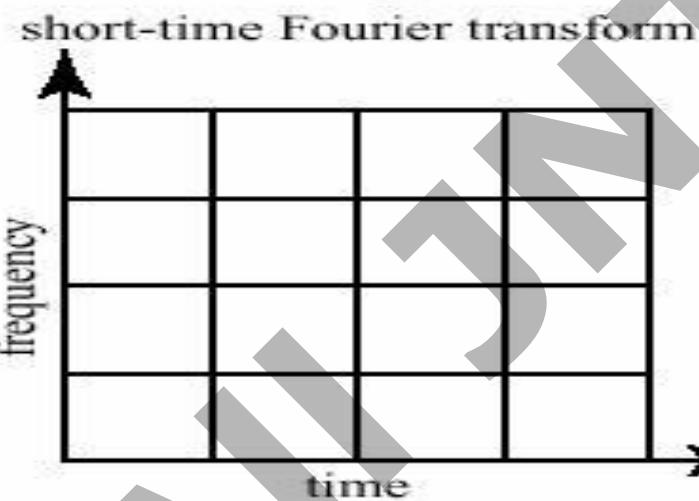
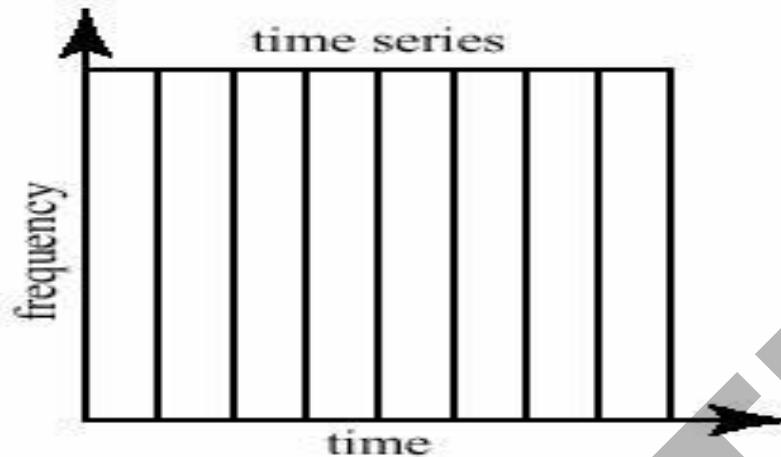
- Wavelet analysis produces a time-scale view of the input signal or image.

•Inverse CWT:

$$f(t) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} c(\tau, s) \psi\left(\frac{t-\tau}{s}\right) d\tau ds$$

double integral!

COMPARISON OF TRANSFORMATIONS





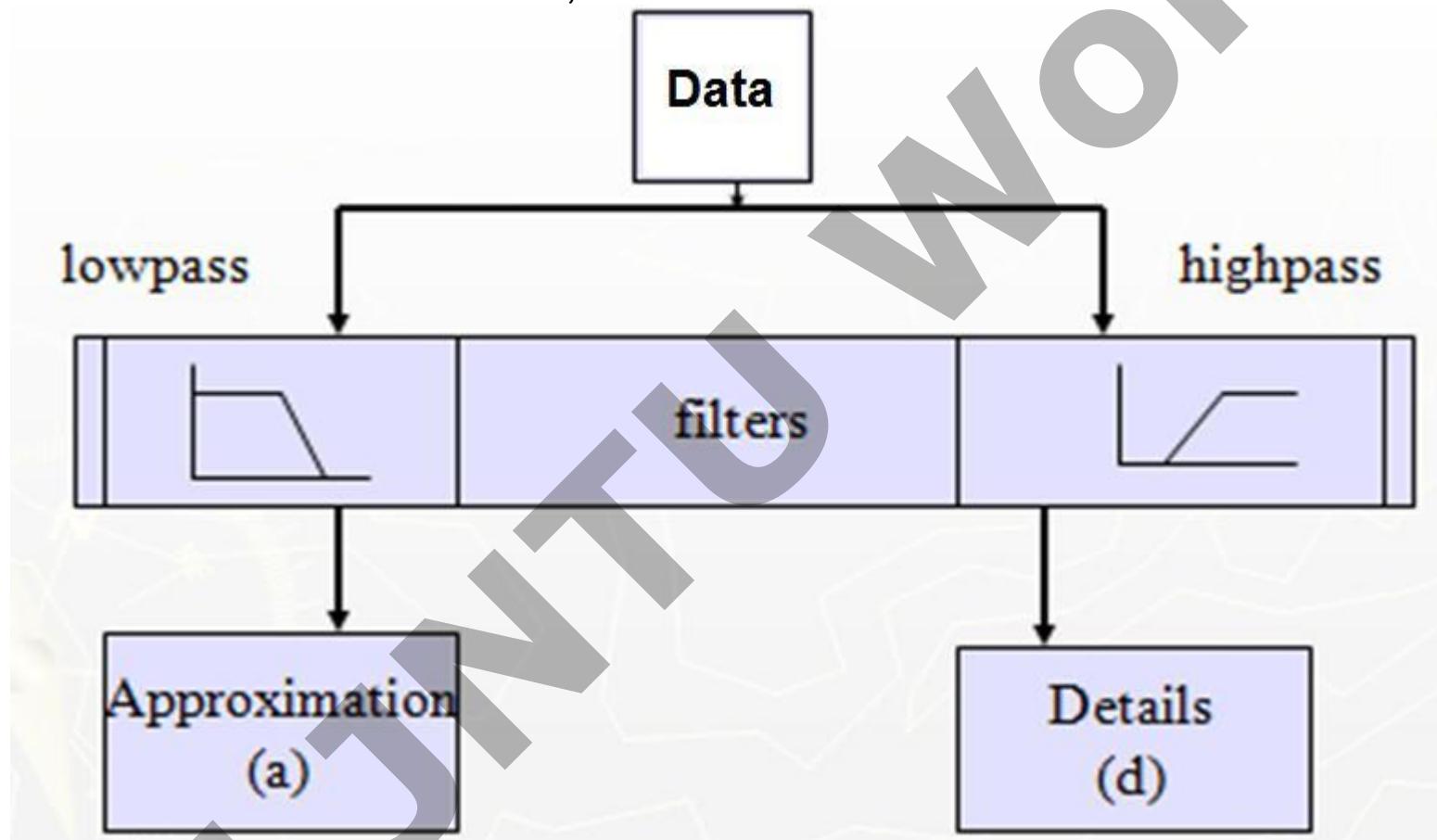
Each box represents an equal portion or equal area

COMPARISON OF CWT And Discrete Wavelet Transform

	CWT	DWT
Scale	At any scale	Dyadic scales
Translation	At any point	Integer point
Wavelet	Any wavelet that satisfies minimum critteria	Orthogonal, bi orthogonal
Computation	Large	Small
Detection	Easily detects Direction, Orientation	Can not detect minute objects if not finely tuned
Application	Pattern recognition, feature extraction, detection	Compression, De noising, Transmission, Characterisation

Discrete Wavelet transform

DWT is a fast linear operation on a data vector, whose length is an integer power of 2. Behaves like a filter bank: Data in, coefficients out



The lower resolution coefficients can be calculated from the higher resolution coefficients by a tree-structured algorithm (**filter bank**).

Discrete Wavelet Transform (DWT)

- When the signal is in vector form (or pixel form), the discrete wavelet transform may be applied.
- Decompose the signal into a coarse approximation and detail information

Remember that our discrete wavelets are not time-discrete, only the translation and the scale step are discrete.

Discrete Wavelet Transform is computed by
successive low pass and high pass filtering of the input image data.

This is called the Mallat algorithm or
Mallat-tree decomposition.

- Mallat was the first to implement this scheme, using a well known filter design called “two channel sub band coder”, yielding a ‘*Fast Wavelet Transform*’

Mallat's Algorithm

The input data is passed through two convolution functions, each of which creates an output stream that is half the length of the original input. This procedure is referred to as down sampling . The convolution functions are filters.

The low pass outputs contain most of the information of the input data and are known as “**coarse**” coefficients. The outputs from the high pass filter are known as “**detail**” coefficients.

The coefficients obtained from the low pass filter are used as the original signal for the next set of coefficients.

This procedure is carried out recursively until a trivial number of low pass filter coefficients are left. The final output contains the remaining low pass filter outputs and the accumulated high pass filter outputs.

This procedure is termed as decomposition.

Effectively, the DWT is nothing but a system of filters.

There are two filters involved,

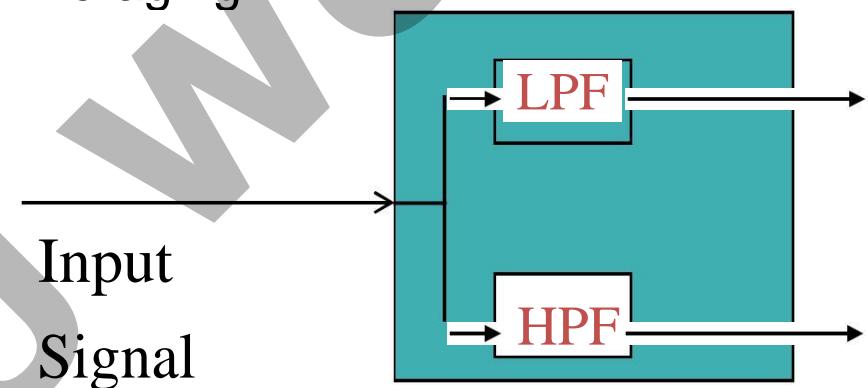
one is the “wavelet filter”,

other is the “scaling filter”.

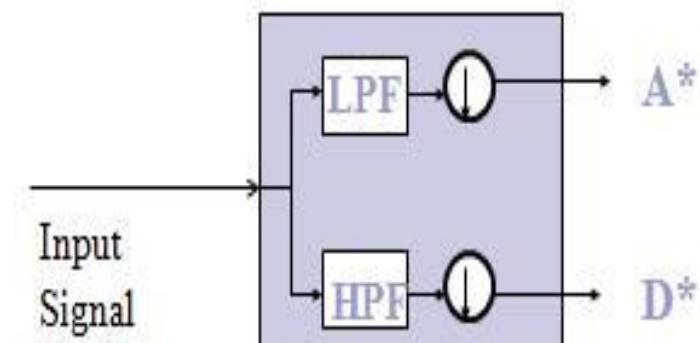
HIGH PASS
FILTER Details

LOW PASS
FILTER Averaging

- **Approximations:** High-scale, low-frequency components of the signal
- **Details:** low-scale, high-frequency components
- This process produces twice the data it began with: N input samples produce N approximations coefficients and N detail coefficients.
- To correct this, we *Down sample* (or: *Decimate*) the filter output by two, by simply **throwing away** every second coefficient.



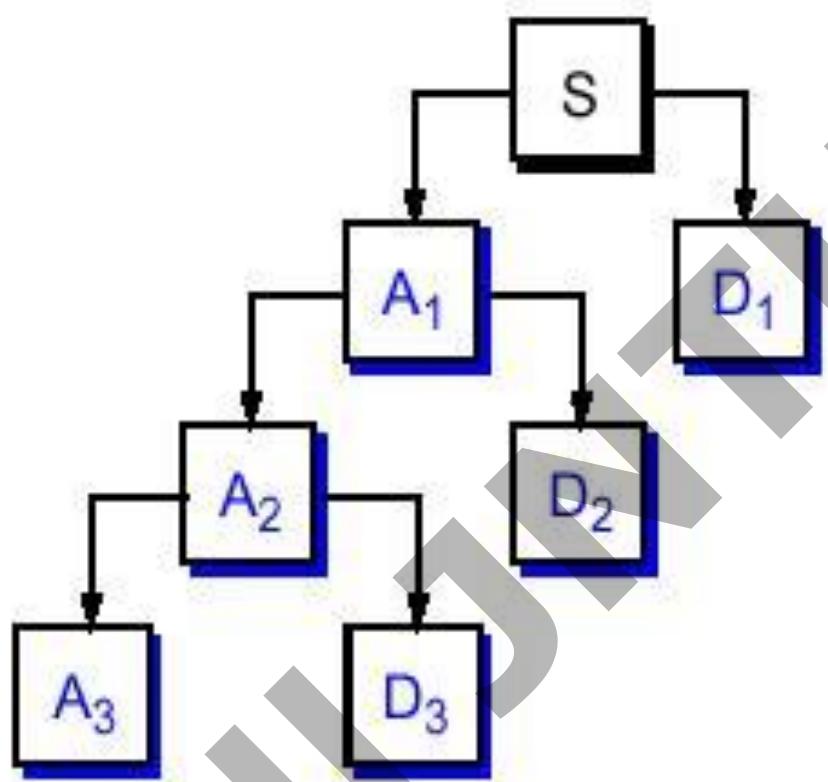
So, a complete one stage block looks like:



Multi-level Wavelet Analysis

Multi-level wavelet decomposition tree

Reassembling original signal



$$S = A_1 + D_1$$

$$= A_2 + D_2 + D_1$$

$$= A_3 + D_3 + D_2 + D_1$$

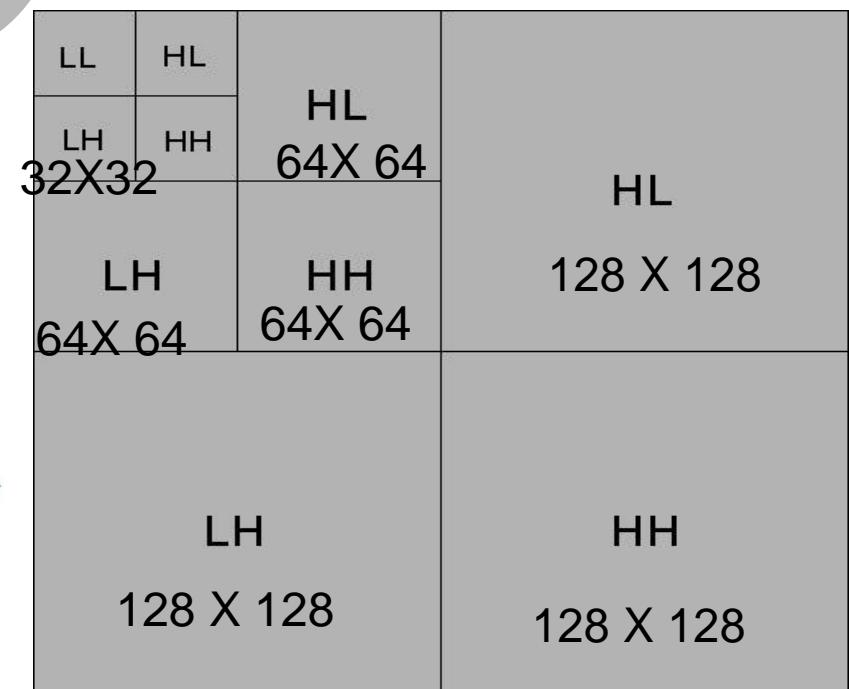
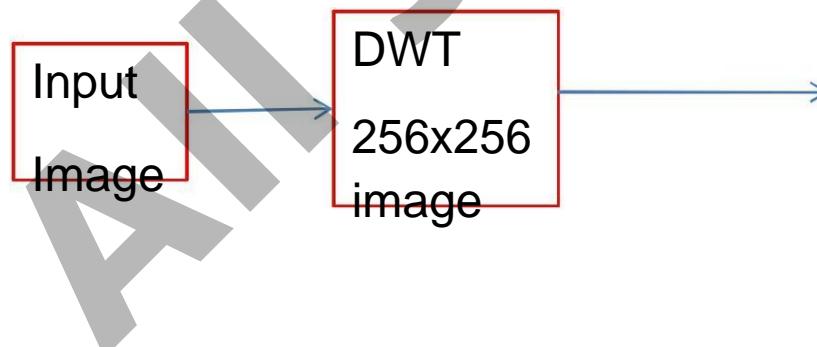
2D (Separable) Wavelets and Filter Banks

The 1D wavelet transform can be applied twice to form a 2D wavelet transform.

First, the rows of the input are transformed, then the columns. This approach only works if the filters are separable: that is, if the filter transfer functions are of the form $H(z_1, z_2) = H_1(z_1)H_2(z_2)$.

Thus in 2D, the wavelet transform has 4 stages for every scale: filtering and decimation along the rows and then along the columns. To extend the transform to color, the image can be decomposed into its RGB components, essentially reducing the image to 3 separate grayscale images.

The figure shows how an image is successively decomposed into high- and low frequency bands horizontally and vertically. The first letter refers to the horizontal component, the second to the vertical.



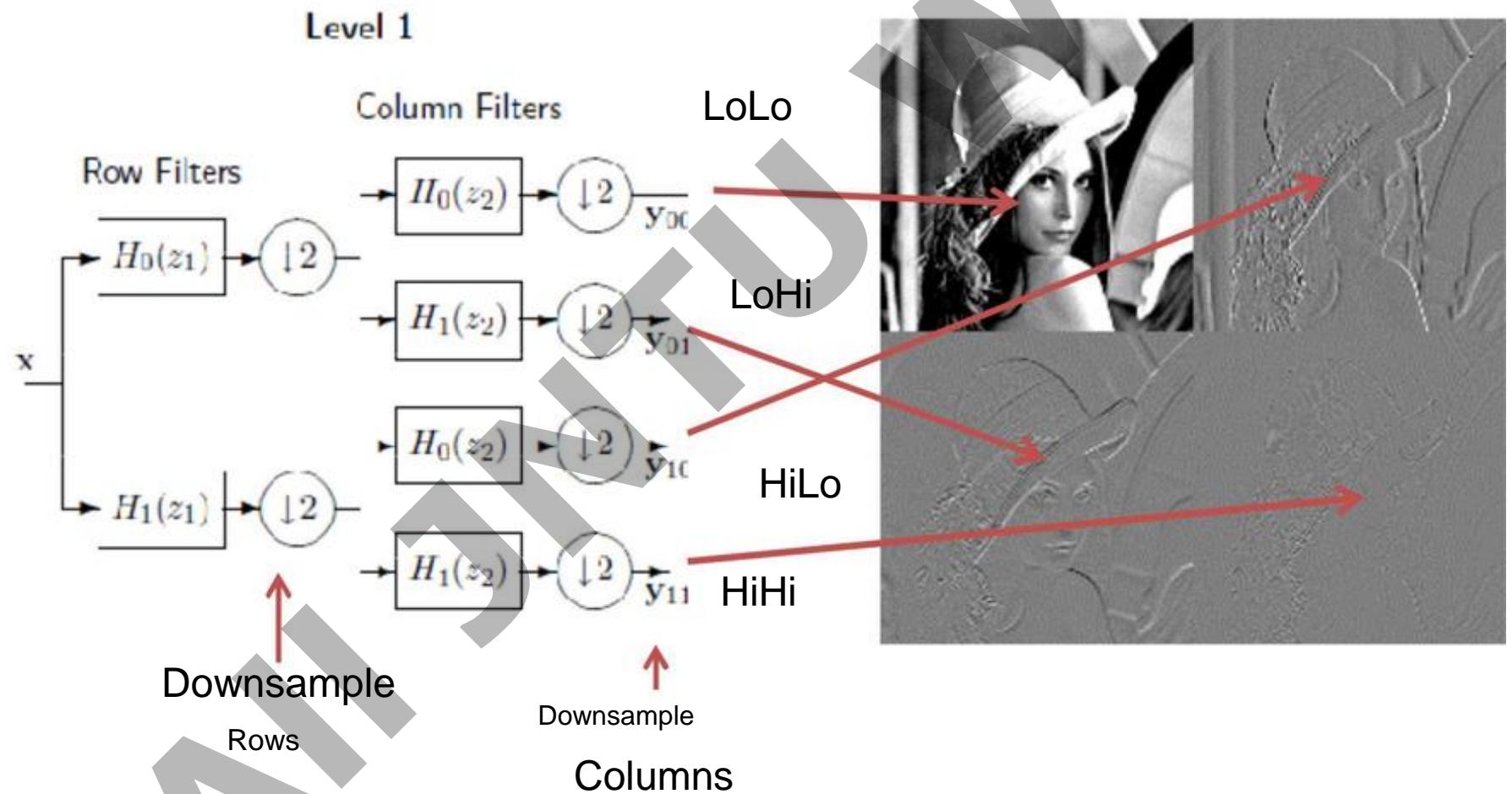
Wavelet Transforms in Two Dimensions

$$W(j_0, m, n) \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) j_0, m, n(x, y)$$
$$j, m, n(x, y) 2^{j/2} (2^j x m, 2^j y n)$$
$$i j, m, n(x, y) 2^{j/2} i (2^j x m, 2^j y n) \quad i \{H, V, D\}$$
$$W^i(j, m, n) \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) i j, m, n(x, y)$$
$$i \quad H, V, D$$

Inverse Wavelet Transforms in Two Dimensions

$$f(x, y) = \frac{1}{\sqrt{MN}} \sum_{m,n} W(j_0, m, n) \sum_{j,m,n} (x, y)$$
$$= \frac{1}{\sqrt{MN}} \sum_{i \in \{H, V, D\}} \sum_{j \neq j_0} W^i(j, m, n) \sum_{m,n} (x, y)$$

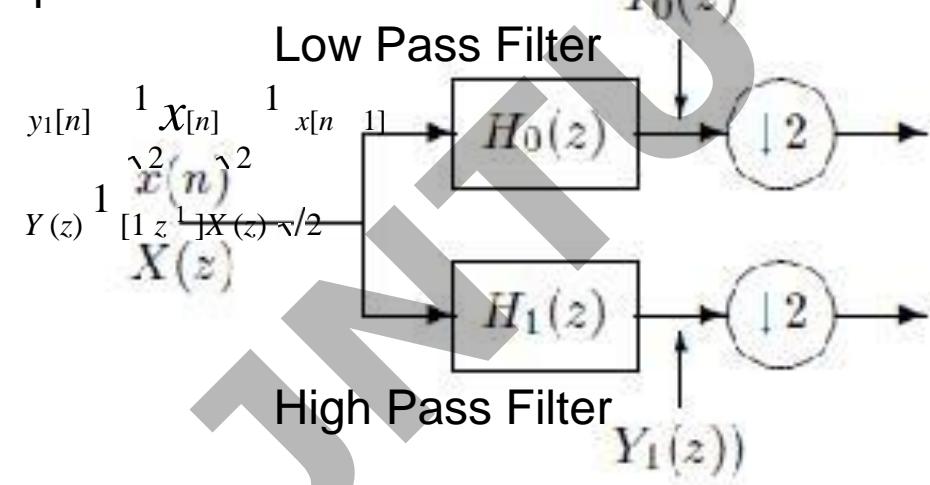
2D Wavelet Transform



Analysis Filter Bank

Analysis filter bank has common Inputs;
Synthesis filter bank has common outputs

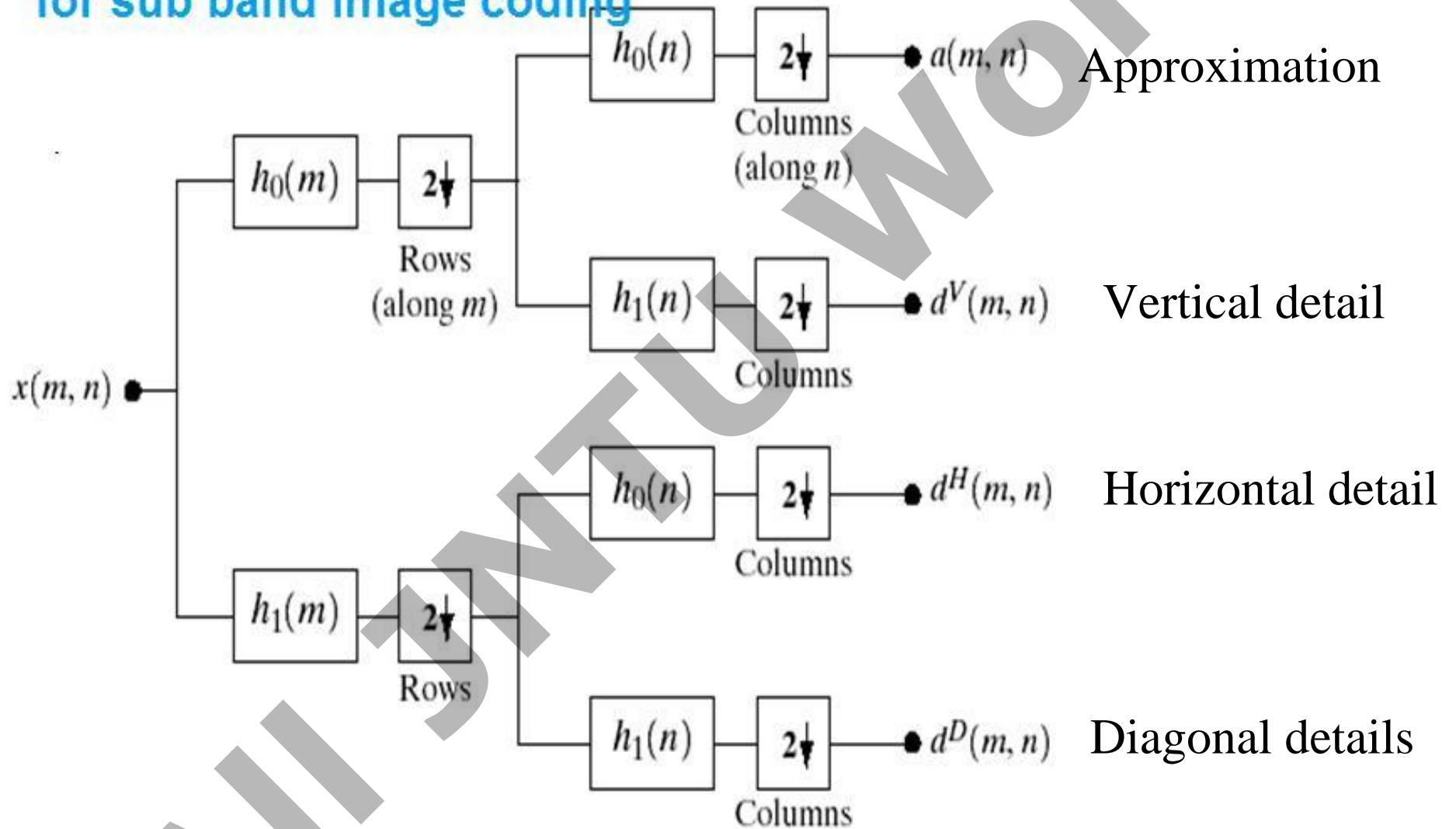
$$y_0[n] = \frac{1}{\sqrt{2}}x[n-1] + \frac{1}{\sqrt{2}}x[n]$$
$$Y(z) = H_0(z)X(z) + \frac{1}{\sqrt{2}}[z^{-1} - 1]X(z)$$



$$y_1[n] = \frac{1}{\sqrt{2}}x[n-1] - \frac{1}{\sqrt{2}}x[n]$$
$$Y(z) = H_1(z)X(z) + \frac{1}{\sqrt{2}}[z^{-1} - 1]X(z)$$

2-D 4-band filter bank

A two dimensional four band filter bank
for sub band image coding



SUBBABD CODING ALGORITHM

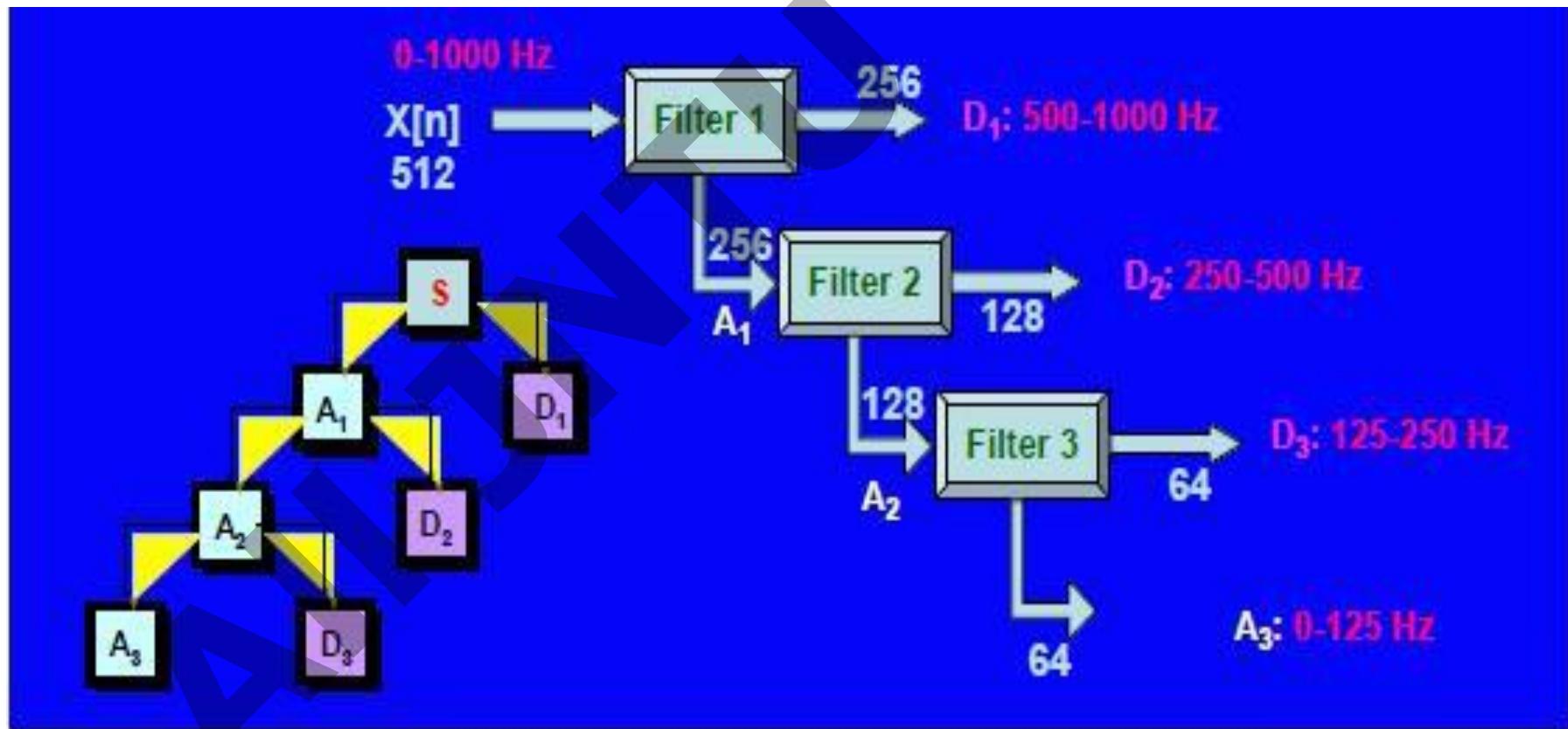
If we regard the wavelet transform as a filter bank, then we can consider wavelet transforming a signal as passing the signal through this filter bank. The outputs of the different filter stages are the wavelet- and scaling function transform coefficients

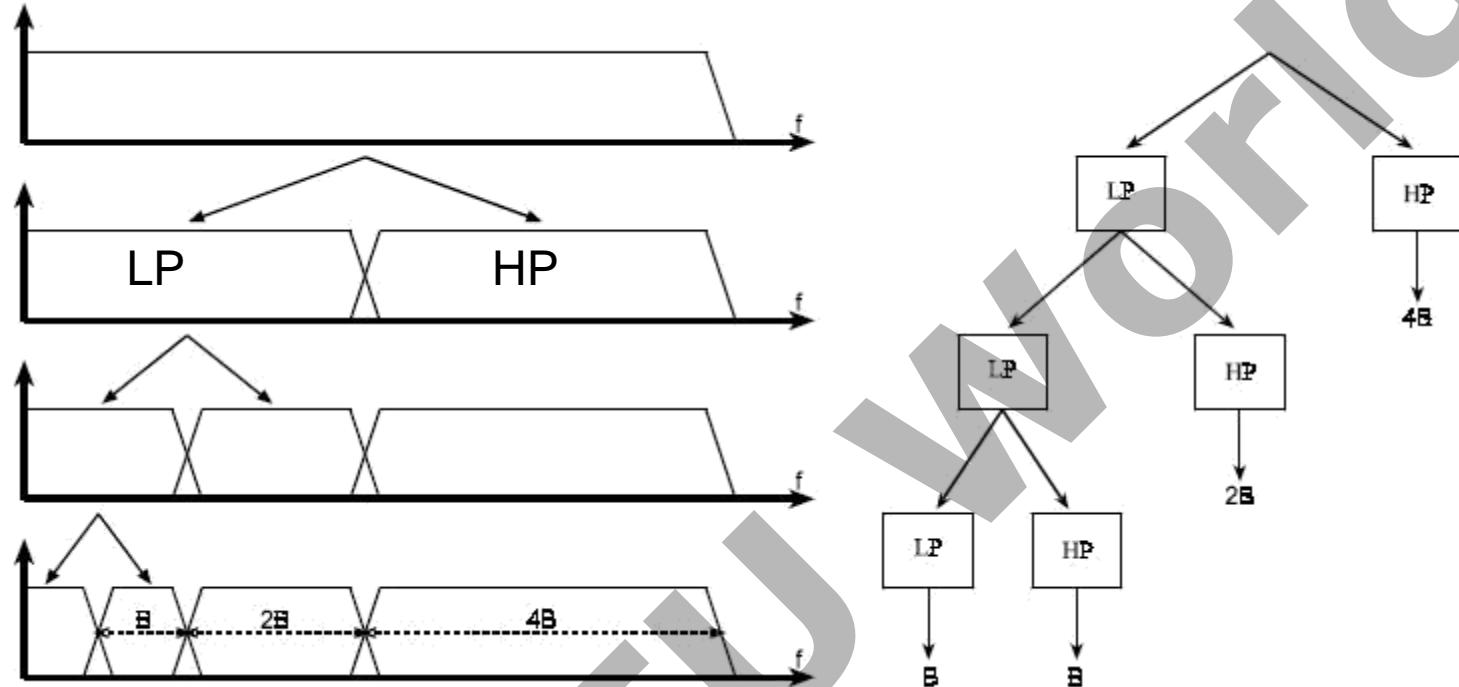
Halves the Time Resolution

Only half number of samples resulted

Doubles the Frequency Resolution

The spanned frequency band halved





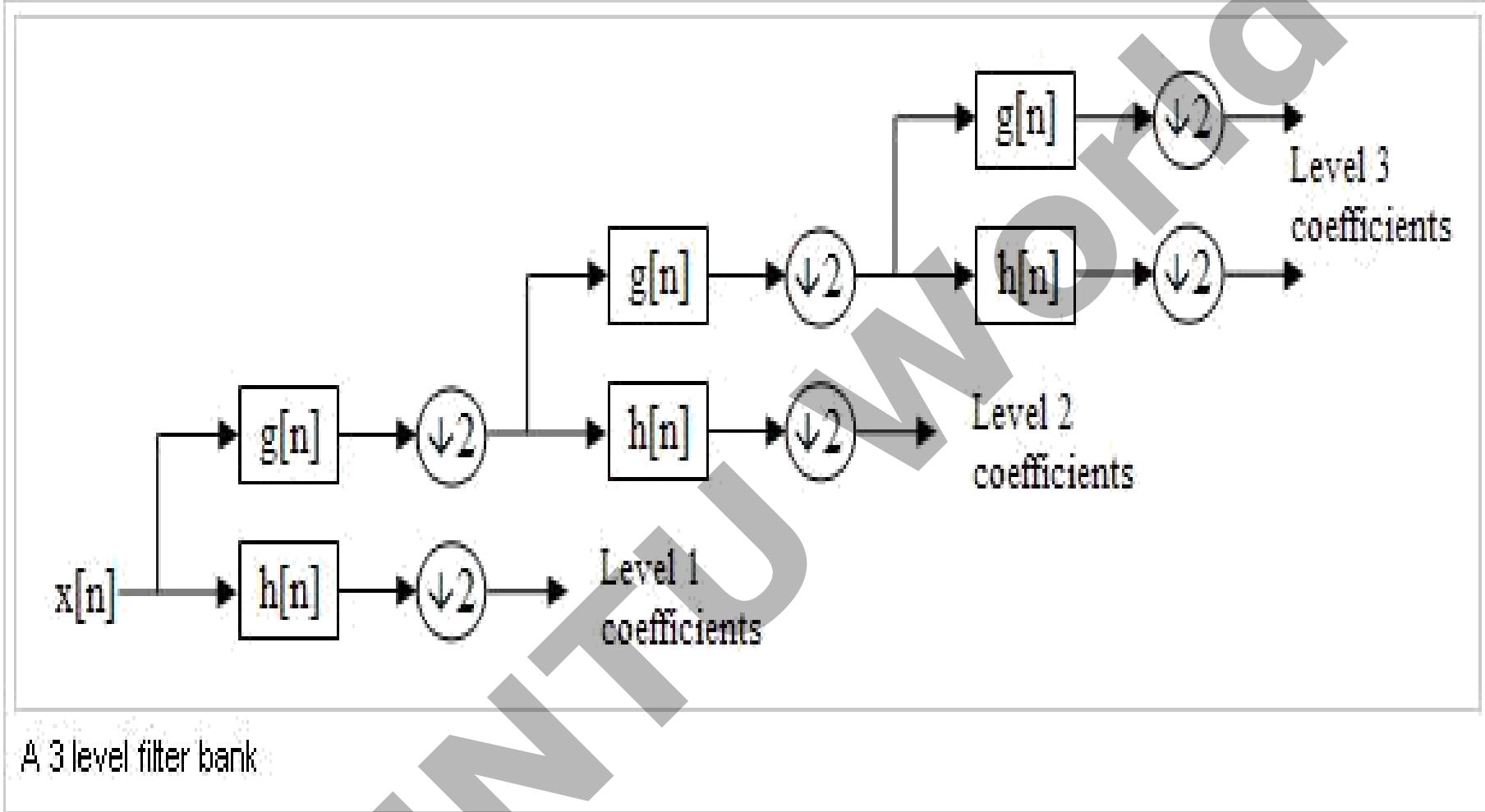
Another way is to split the signal spectrum in two (equal) parts, a low-pass and a high-pass part. The high-pass part contains the smallest details we are interested in and we could stop here. We now have two bands. However, the low-pass part still contains some details and therefore we can split it again. And again, until we are satisfied with the number of bands we have created. In this way we have created an *iterated filter bank*. *Usually the number of bands is limited by for instance the amount of data or computation power available.*

The wavelets give us the band-pass bands with doubling bandwidth and the scaling function provides us with the low-pass band.

From this we can conclude that a wavelet transform is the same thing as a subband coding scheme using a constant-Q filter bank.

In general we will refer to this kind of analysis as a multiresolution analysis.

Summarizing, if we implement the wavelet transform as an iterated filter bank, we do not have to specify the wavelets explicitly!



Haar Transform

Averages and differences

- two neighboring samples

$$\begin{array}{ccc} & s = (a+b)/2 & \\ a \quad b & \nearrow & \searrow \\ & d = b - a & \end{array}$$
$$a = s - d/2$$
$$b = s + d/2$$

Properties

- exploits correlation
- better encoding possible

$$s = \frac{a+b}{2},$$
$$d = a - s.$$

and the inverse $(d, s) \rightarrow (a, b)$ by

$$a = s + d,$$
$$b = s - d.$$

Averaging and differencing at multiple scales can be done efficiently via a discrete orthogonal wavelet transform (WT), e.g., the Haar wavelet

Haar Wavelet

Resolution	Averages	Detail Coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4] [2, 1, 4, 4] [1.5, 4] [2.75]	----- [0, -1, -1, 0] [0.5, 0] [-1.25]

wavelet decomposition(wavelet transform): [2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

The diagram illustrates the Haar wavelet decomposition of a signal across four resolution levels. The signal starts at resolution 0 with a value of 2.75. At each subsequent resolution, the signal is split into two averages (sums of pairs) and one detail coefficient (the difference between the pairs). The averages are shown in red boxes, and the detail coefficients are shown in blue boxes.

Resolution 0: Value [2.75]

Resolution 1: Averages [1.5, 4], Detail Coefficient [-1.25]

Resolution 2: Averages [2, 1, 4, 4], Detail Coefficient [0.5, 0]

Resolution 3: Averages [2, 2, 0, 2, 3, 5, 4, 4], Detail Coefficients [0, -1, -1, 0]

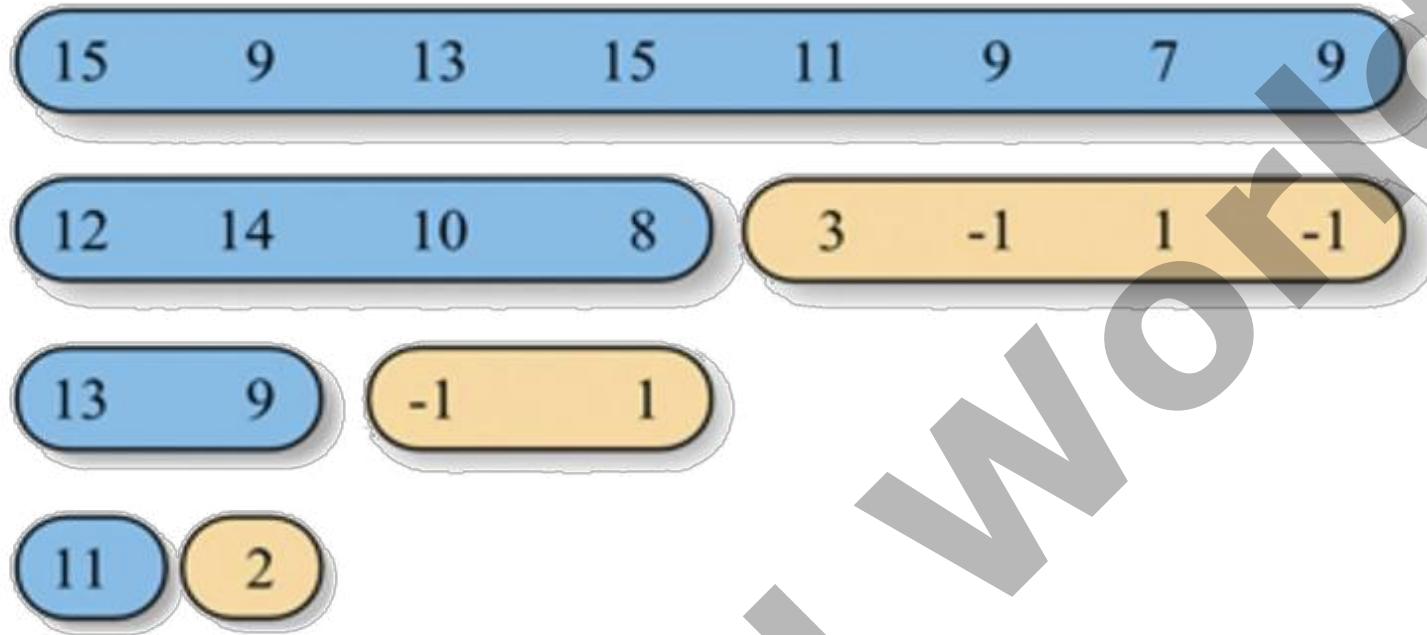
Haar Wavelet Coefficients

- Using wavelet coefficients one can pull the raw data
- Keep only the large wavelet coefficients and pretend other coefficients to be 0.

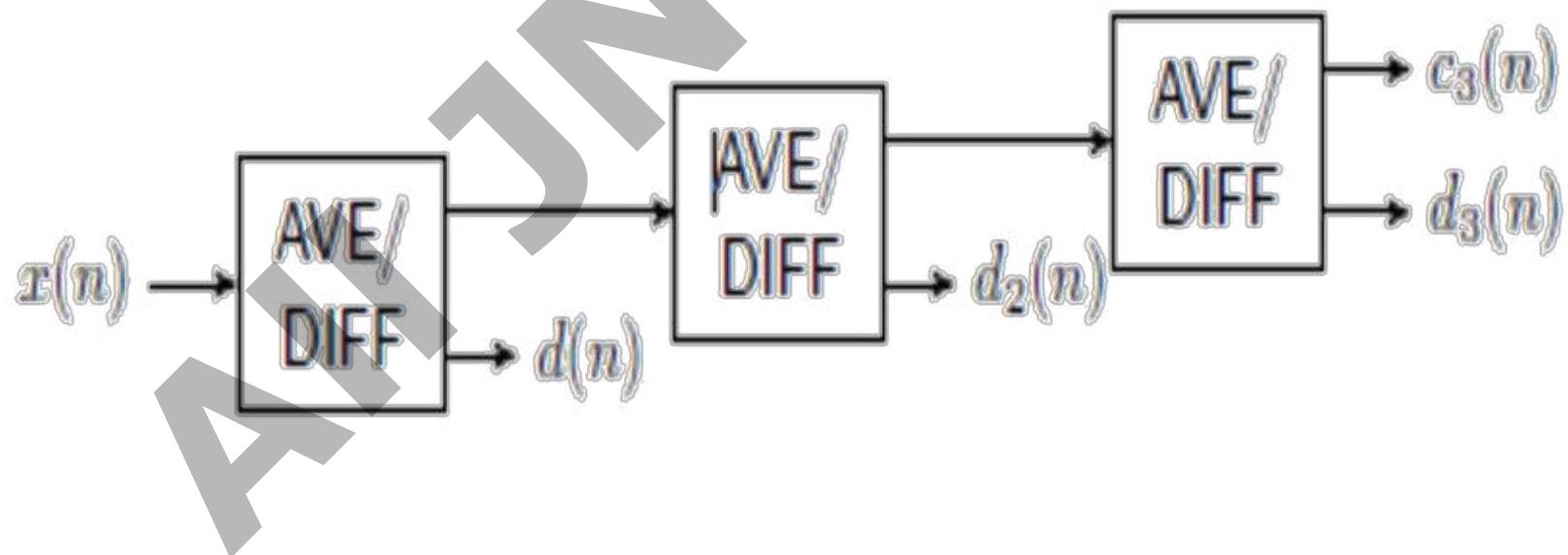
[2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

[2.75, -1.25, 0.5, 0, 0, 0, 0, 0]-synopsis of
the data

- The elimination of small coefficients introduces only small error when reconstructing the original data



. Haar wavelet example.



AUJNTU World

Why wavelets for denoising?

Noise removal or noise reduction can be done on an image by filtering, by wavelet analysis, or by multi-fractal analysis.

Wavelet techniques consider thresholding

During thresholding, a wavelet coefficient is compared with a given threshold and is set to zero if its magnitude is less than the threshold; otherwise, it is retained or modified depending on the threshold rule.

Types

Universal or Global Thresholding

Hard

Soft

SubBand Adaptive Thresholding

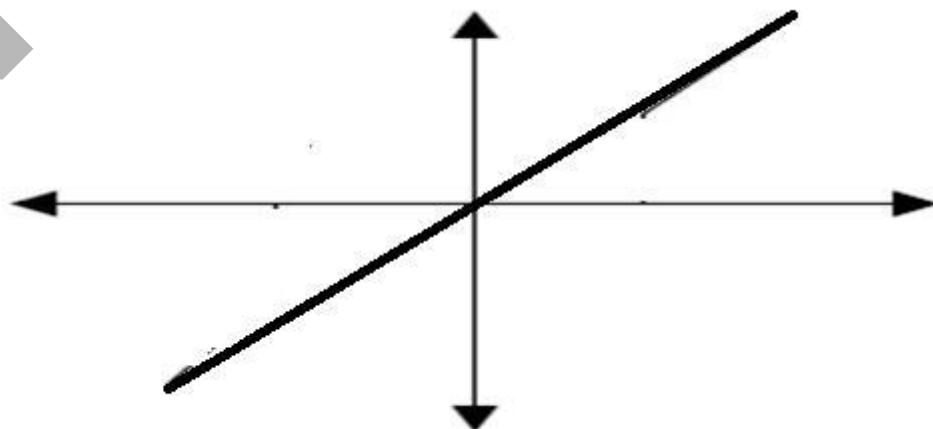
The choice of a threshold is an important point of interest.

Care should be taken so as to preserve the edges of the denoised image.

There exist various methods for wavelet thresholding, which rely on the choice of a threshold value.

Some typically used methods for image noise removal include
VisuShrink, Level shrink, SureShrink and BayesShrink

Linear thresholding

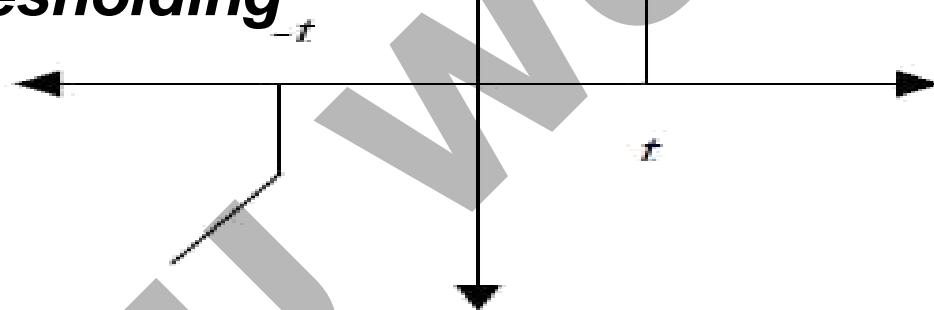


$$T_H = \begin{cases} x & \text{for } |x| \geq t \\ 0 & \text{in all other regions.} \end{cases}$$

Here t is the threshold value. A plot of T_H is shown in Figure

The hard thresholding rule is usually referred to simply as **wavelet thresholding**

Hard thresholding

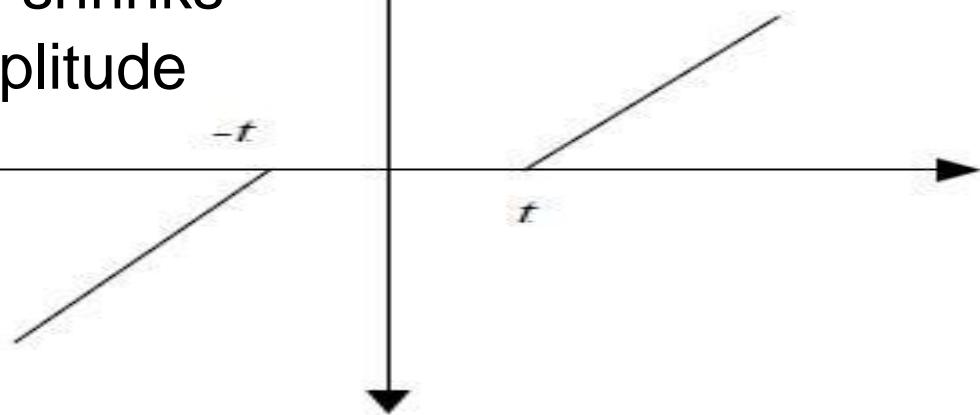


$$T_S = \begin{cases} \text{sign}(x)(|x| - t) & \text{for } |x| > t \\ 0 & \text{in all other regions.} \end{cases}$$

soft thresholding is referred to as **wavelet shrinkage**, since it "shrinks" the coefficients with high amplitude towards zero.

Soft thresholding

$$\begin{aligned} T_S &= x - t, \text{ if } x > t \\ &= x + t, \text{ if } x < t, \\ &= 0, \text{ otherwise} \end{aligned}$$



hard- thresholding and soft-thresholding

Hard thresholding sets zeros for all wavelet coefficients whose absolute value is less than the specified threshold limit. It has shown that hard thresholding provides an improved signal to noise ratio.

Soft thresholding is where the coefficients with greater than the threshold are shrunk towards zero after comparing them to a threshold value

The thresholding of the wavelet coefficients is usually applied only to the detail coefficients $d_{j,k}$ of y rather than to the approximation coefficients $a_{j,k}$ since the latter ones represent 'low-frequency' terms that usually contain important components of the signal, and are less affected by the noise.

In practice, it can be seen that the soft method is much better and yields more visually pleasant images. This is because the hard method is discontinuous and yields abrupt artifacts in the recovered images. Also, the soft method yields a smaller minimum mean squared error compared to hard form of thresholding.

Signal denoising using the DWT consists of the three successive procedures, namely,

signal decomposition,
thresholding of the DWT coefficients,
and signal reconstruction.

IMAGE NOISE ALGORITHM

It has following steps:

1. Perform multi scale decomposition of the image corrupted by gaussian noise using wavelet transform.
2. Estimate the noise variance σ^2
3. For each level, compute the scale parameter
4. Compute the standard deviation
5. Compute threshold t ,
6. Apply semi soft thresholding to the noisy coefficients.
7. Denoise high frequency coefficient
8. Merge low frequency coefficient with denoise high frequency coefficient
9. Invert the multiscale decomposition to reconstruct the denoised image .

VisuShrink

It uses a threshold value t that is proportional to the standard deviation of the noise.

It follows the hard thresholding rule. It is also referred to as universal threshold and is defined as

$$t = \sigma \sqrt{2 \log N}$$

σ^2 is the noise variance present in the signal and N represents the signal size or number of samples.

Universal or Global Thresholding

$$t_{Univ} = \sqrt{2 \ln N}$$

Level shrink: uses level dependent thresholds

$$t_j = \sqrt{2 \ln N} \sigma 2^{-(J-j)/2}$$

J is total decomposition levels , j is scale level

SureShrink

A threshold chooser based on Stein's Unbiased Risk Estimator (SURE) and is called as SureShrink.

It is a combination of the universal threshold and the SURE threshold. This method specifies a threshold value t_j for each resolution level j in the wavelet transform which is referred to as level dependent thresholding.

Let wavelet coefficients in the j -th subband be $\{ \hat{X}_i : i = 1, \dots, d \}$

For the soft threshold estimator $\hat{X}_{it} (\hat{X}_i)$

$$SURE(t; X) = \sum_{i=1}^d \min_{t \geq 0} |\hat{X}_i|^2 - t^2$$

Select threshold t^S by $t^S = \text{argmin} SURE(t; X)$

The goal of SureShrink is to minimize the mean squared error

$$\text{MSE} = \frac{1}{n^2} \sum_{x,y=1}^n (z(x,y) - s(x,y))^2,$$

where $z(x,y)$ is the estimate of the signal while $f(x,y)$ is the original signal without noise and n is the size of the signal. SureShrink suppresses noise by thresholding the empirical wavelet coefficients.

SureShrink

The SureShrink threshold t^* is defined as

$$t^* = \min(t, \sigma\sqrt{2\log n}),$$

where t denotes the value that minimizes Stein's Unbiased Risk Estimator, σ is the noise variance computed from equation, and n is the size of the image.

SureShrink follows the soft thresholding rule. The thresholding employed here is adaptive

BayesShrink

The goal of this method is to minimize the Bayesian risk, and hence its name, BayesShrink. It uses soft thresholding and is **subband-dependent**, which means that thresholding is done at each band of resolution in the wavelet decomposition. Like the SureShrink procedure, it is smoothness adaptive.

The Bayes threshold, tB , is defined as where σ^2 is the noise variance and σ_s is the signal variance without noise

$$tB = \sigma^2 / \sigma_s$$

Thus, this threshold choice adapts to both the signal and the noise characteristics as reflected in the parameters σ and σ_s

When $\sigma^2 / \sigma_s \leq 1$, *the signal is much stronger than the noise*

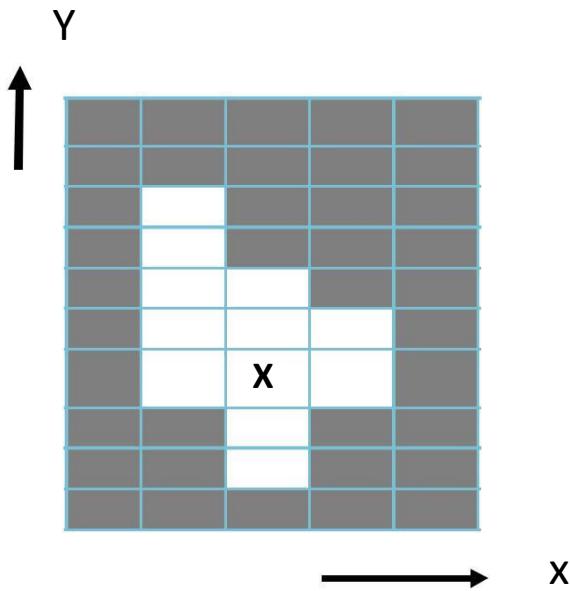
When $\sigma^2 / \sigma_s \geq 1$, *the noise dominates*

Morphological (shape-based) Image processing

Morphology is an image processing techniques that discusses about the structure and shape of objects. It is a tool for extracting image components that are useful for representation and description of region shape, boundary, skeleton, convex hull etc.

In image morphology, the basic assumption is an image can be presented by a point set

Binary morphology is applied to binary images,
Gray level morphology is applied to gray level images
It can be applied to color images also.



X is a set of points belonging to object, The pixel values of **object (Foreground)** is 1 and **back ground** is zero for this binary image.

$A \in z^2$

$a = (a_1 \ a_2);$

$a \in A$

$b \notin A$

$A \subset B$

$A \cup B$

$A \cap B$

$A^c = \{ b \mid b \notin A \}$ **A complement is equal to b, and b does not belong to set A**

$A - B = \{ w \mid w \in A \text{ and } w \notin B \}$ **Difference of A and B**

set of points A in two dimensional space z^2 .

Point a is represented by an order pairs $(a_1 \ a_2);$

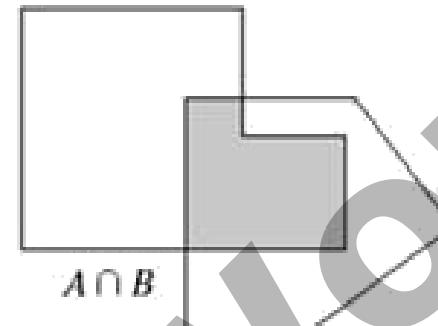
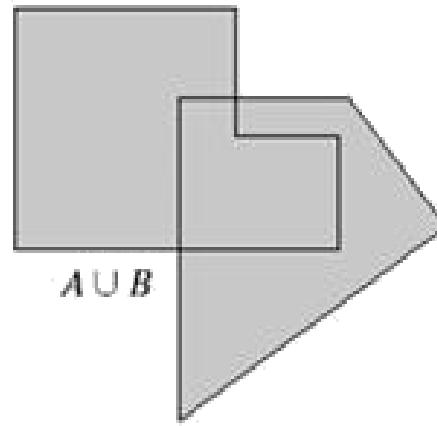
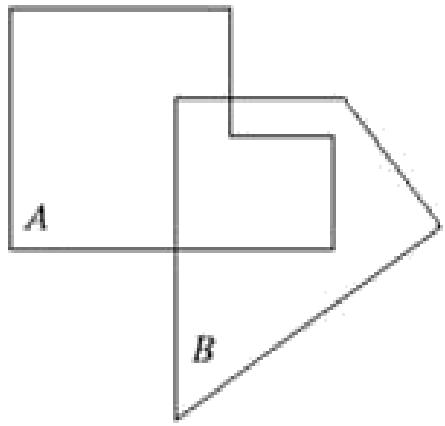
a belongs to set A

b does not belongs to set A

point set A is a sub set of point set B

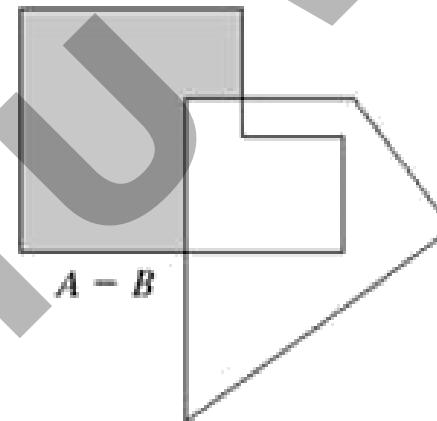
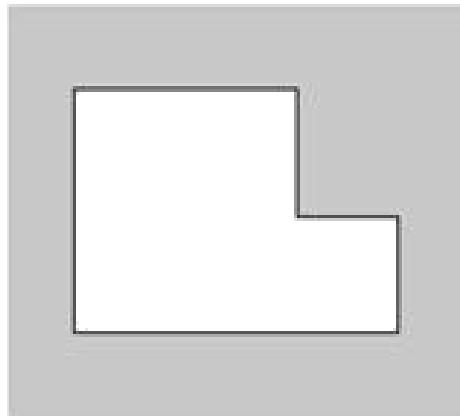
A union B. All the points in A and B

A intersection B. all common elements of A and B



a b c
d e

- (a) Two sets A and B . (b) The union of A and B .
(c) The intersection of A and B . (d) The complement of A .
(e) The difference between A and B .



Reflection of B :

$$B = \{ w \mid w = -b \text{ for } b \in B \}$$

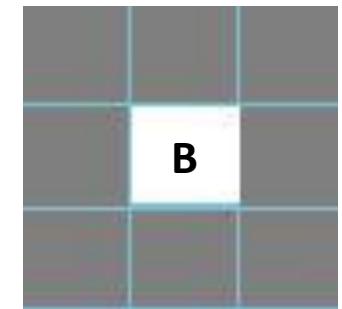
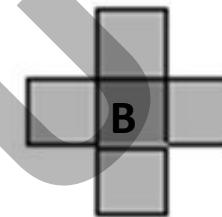
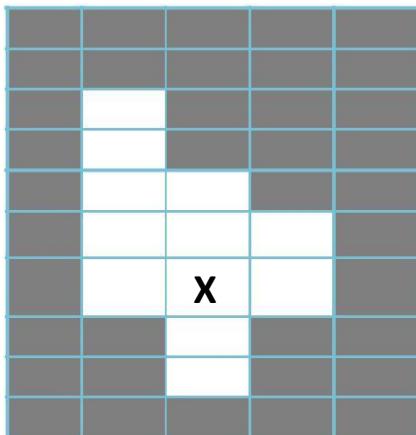
Translation operation:

$$A z = \{ c \mid c = a + z \text{ for } a \in A \}$$

Morphological Transformation:

X

Morphological transformation gives a relation of the image X with another small point set or another small image say capital B which is called a **structuring element**



Structuring Element is a small set to probe the image under study , for each SE, define origin and shape and size must be adapted to geometric properties for the objects

There are basically four morphological transformations:

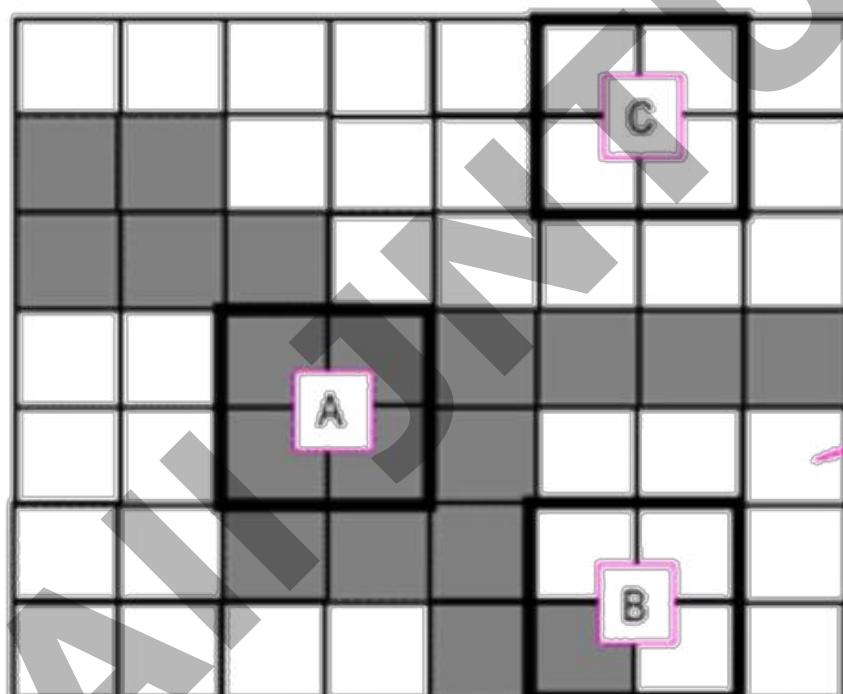
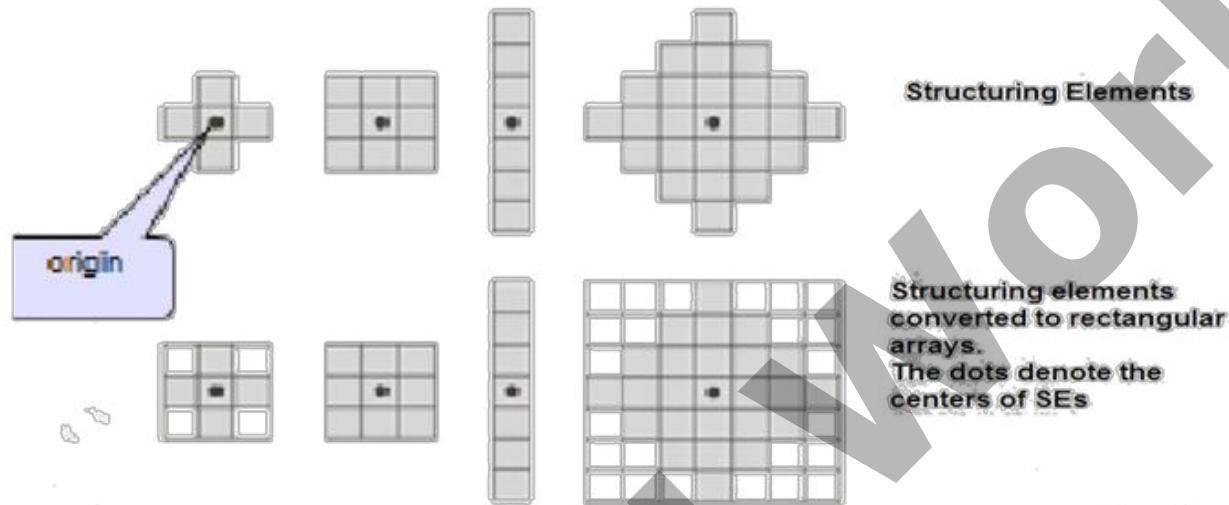
- Dilation,
- Erosion,
- Opening and
- Closing

Structuring Element (Kernel or Template)

- Structuring Elements can have varying sizes
- Usually, element values are 0,1 and none(!)
- Structural Elements have an origin
- For thinning, other values are possible
- Empty spots in the Structuring Elements are *don't care's!*

By applying these structuring elements to the data using different algebraic combinations, one performs morphological transformations on the data.

Examples: Structuring Elements (1)



- A - the structuring element fits the image
- B - the structuring element hits (intersects) the image
- C - the structuring element neither fits, nor hits the image

Structuring element

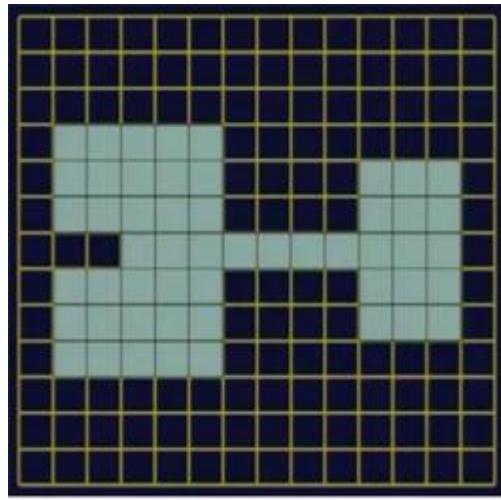


Simple morphological transformations

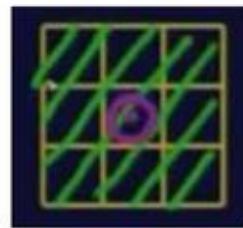
Dilation	$A \oplus B = \{ p \mid p = a + b, a \in A, b \in B \}$ A dilation B (vector addition) object area gets expanded , Grow filling of holes of certain shape and size, given by SE internal noise within objects removed,.
Erosion	$A \ominus B = \{ p \mid p + b \in a, \text{ for every } b \in B \}$ the object area gets reduced along its boundary, Shrink removal of structures of certain shape and size, given by SE
Closing=Dilation + Erosion	$A \ominus B = (A \oplus B) \ominus B$ <p style="text-align: center;">or</p> $A \bullet B = (A \ominus B) \oplus B$ <p>Dilation removes internal noise within objects, Objects get expanded</p> <p>Erosion removes All additional pixel boundaries introduced by dilation , except the filled up internal pixels in the body.</p>
Opening = Erosion + Dilation	$A \ominus B = (A \oplus B) \ominus B$ <p>Erosion Removes boundary pixels in the bodies including the external noise. Shrinks the objects</p> <p>Dilation Expands the removed pixels except the external noise</p>
Hit and Miss operation	<p>Two structuring elements, fore ground and back ground</p> $A \otimes B = (A \ominus B_1) \cap (A^c \oplus B_2)$

Dilation transformation on a image

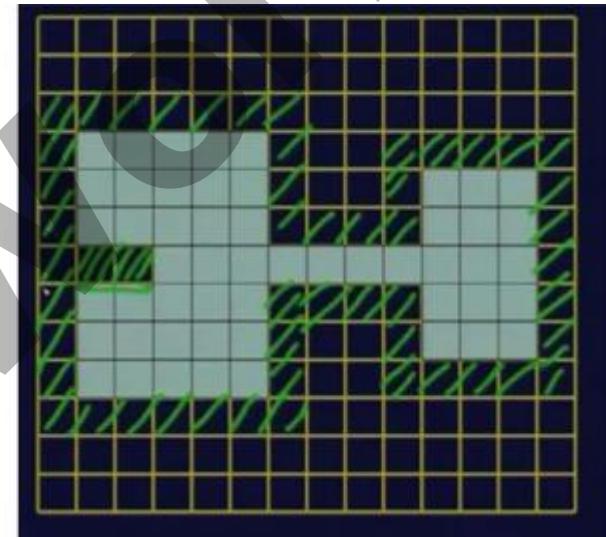
image A



The structuring element is shown by B:

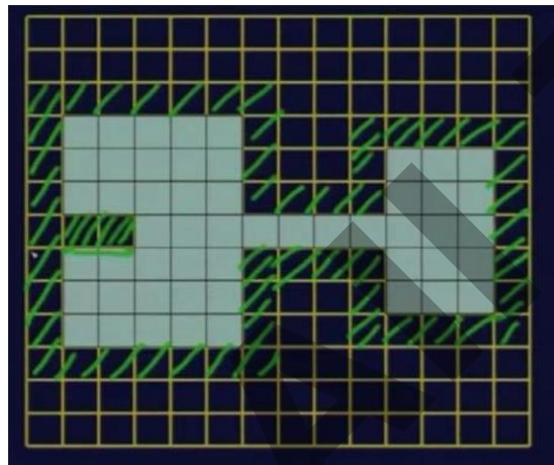


The dilation of B over X

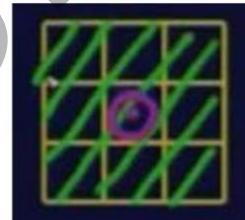


Erosion transformation on a image

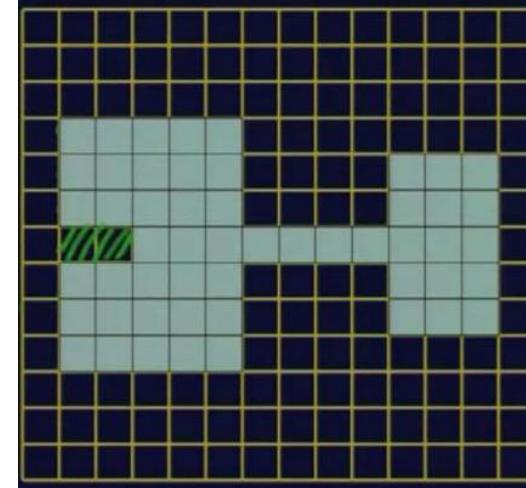
image A



Structuring Element B



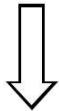
The erosion of B over X



Closing operation

$$A \bullet B = (A - B) \Theta B$$

Closing involves one or more dilations followed by erosion.



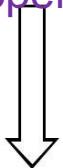
Dilation : Dilation enlarges foreground, shrinks background all pixels turnout to be object pixels, internal noise within objects removed, Objects get expanded.

The Dilation of an Image 'A' by a structuring element 'B' is written as $A \bullet B$

FOLLOWED BY

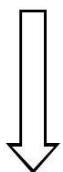
Erosion : Erosion shrinks foreground, enlarges Background All additional pixel boundaries, introduced by dilation will be removed, except the filled up internal pixels in the body.

Opening operation $A \ominus B = (A \ominus B) \oplus B$



Erosion

Removes boundary pixels in the bodies including the external noise. Shrinks the objects

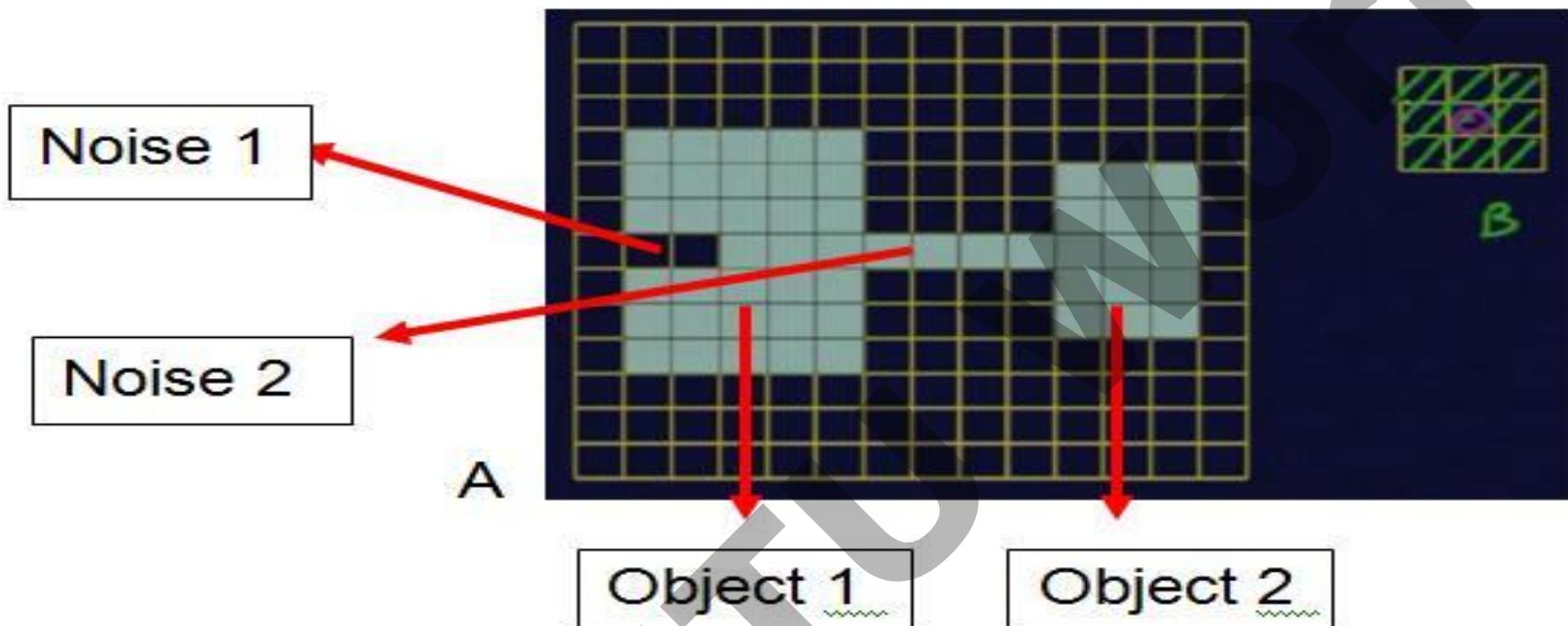


Dilation

Expands the removed pixels except the external noise

Opening is **idempotent**: Repeated application has no further effects!

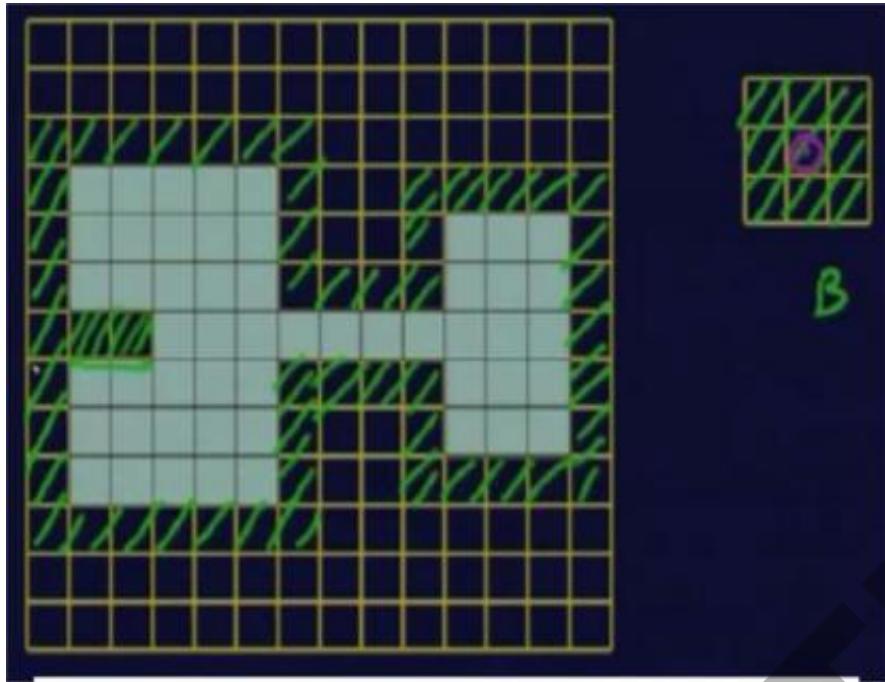
Closing and Opening Operation



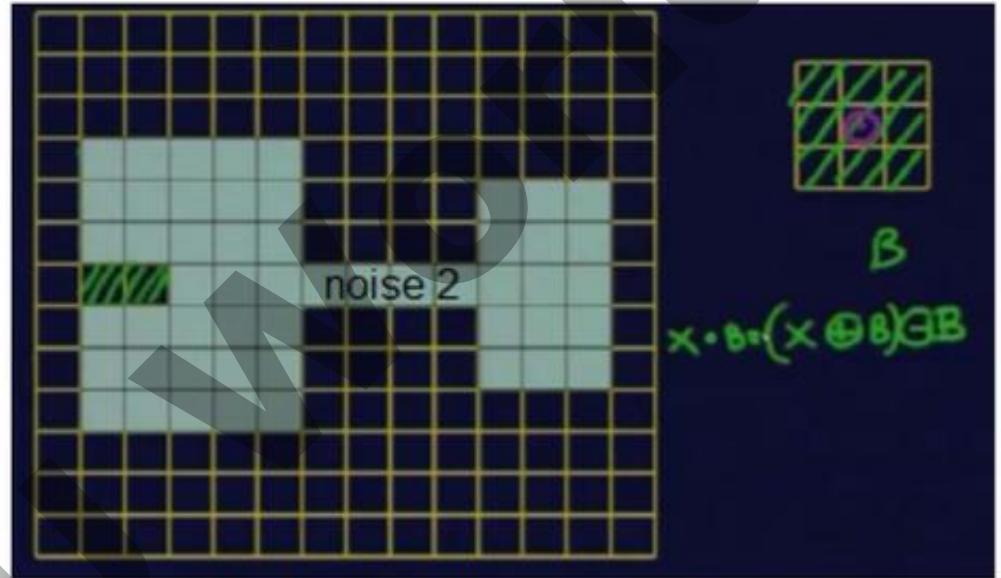
Take this particular binary image. This image contains two different object regions and there are two noisy regions in this image.

One is that the inter region should have been object, but two object pixels have turned out to be background pixels and second one is that we do not have any noise but two object regions are connected by a thin line which is just one pixel wide. So generally, we can assume that this connection is nothing but because of the presence of noise

Closing Operation involves Dilation and Erosion



Dilation increased one pixel boundary, but filled up the internal object noise gap



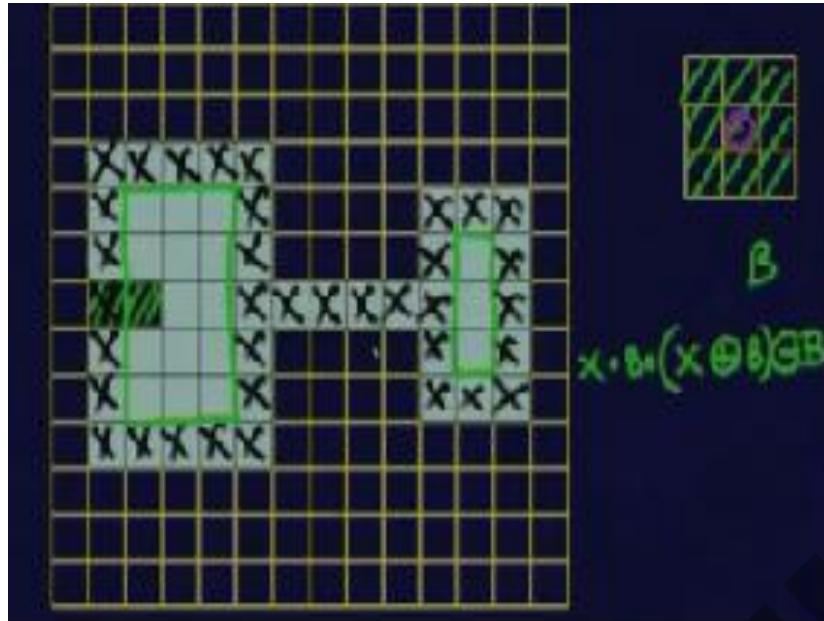
First dilation has filled up internal noise, expanded boundary by a pixel. Erosion on that has removed the expanded portion. But external noise 2 is still existing

So, the effect is quite clear. After doing the morphological closing operation, we have removed the internal noise 1

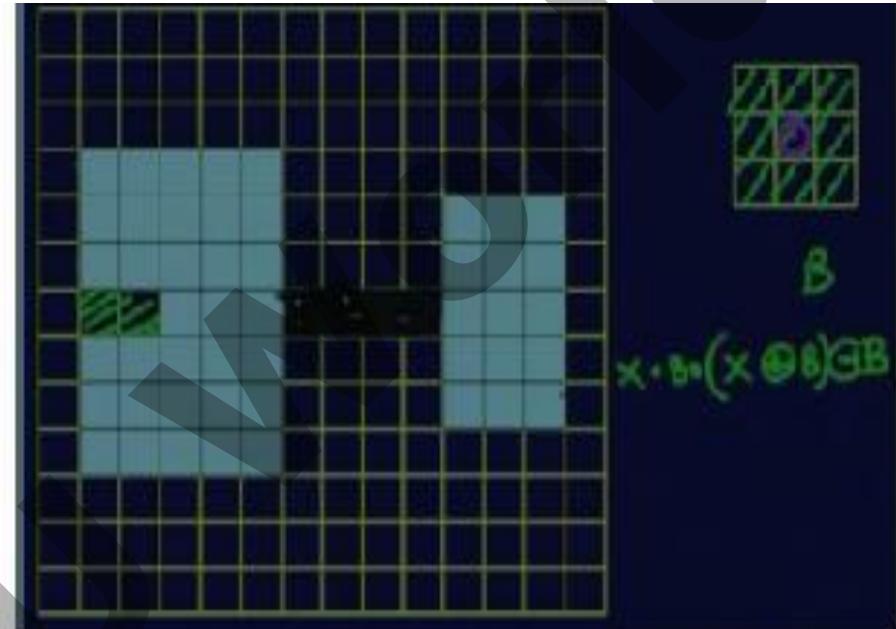
Noise 2 is still present, which is joining the two objects.

This can be removed by opening operation (EROSION AND DILATION)

Opening operation involves erosion and dilations

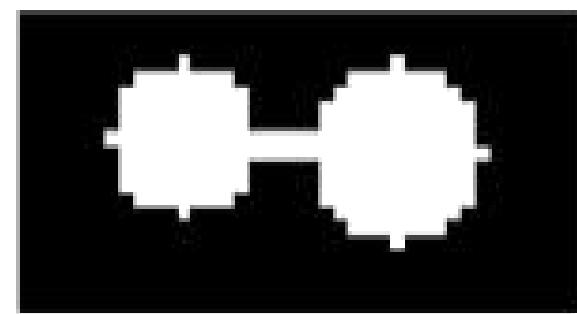
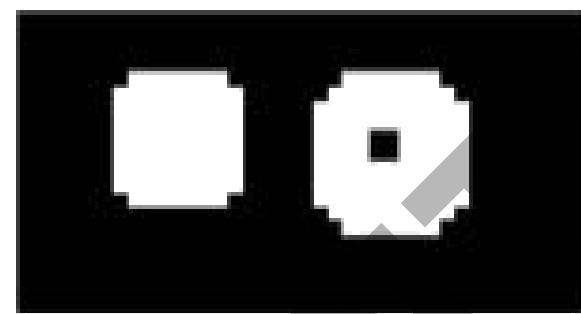
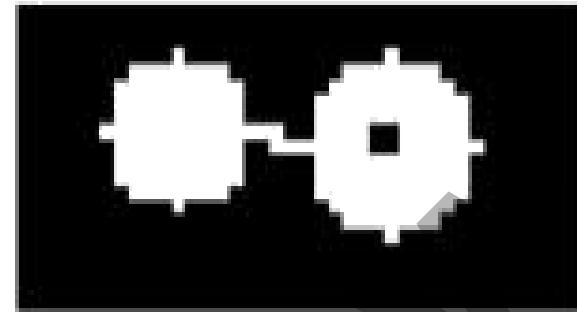


Erosion opening operation has reduced one pixel and so the noise 2 is removed, but images are shrinked.



Dilation has reverted back the shrunked pixel boundary but noise 2 is removed

All world



opening of A

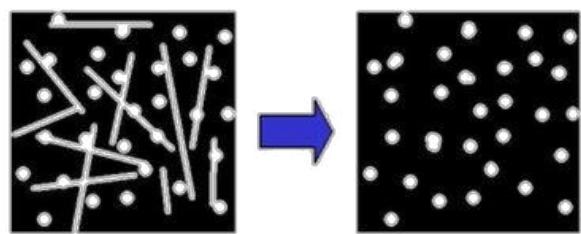
→ removal of small protrusions, thin
connections, ...

closing of A

→ removal of holes

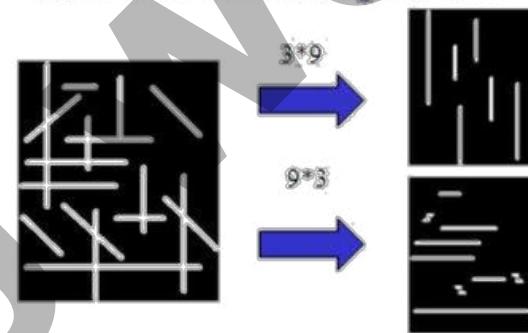
Opening Example

- Opening with a 11 pixel diameter disc



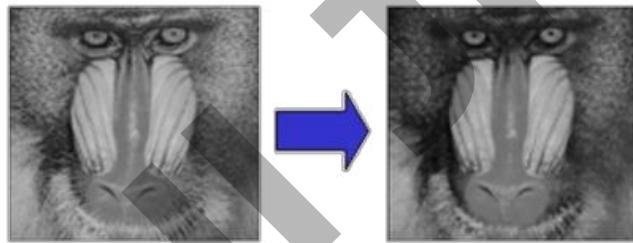
Opening Example

- 3x9 and 9x3 Structuring Element



Opening on Gray Value Images

- 5x5 square structuring element



All JNTU World

Hit-and-miss Transform

- Used to **look for particular patterns** of foreground and background pixels
- Very simple **object recognition**
- All other morphological operations **can be derived from it!!**
- Input: Binary Image
 - Two Structuring Elements, containing 0s and 1s and don't cares

Hit-and-miss Transform

- Similar to Pattern Matching:
- If foreground and background pixels in the structuring element *exactly match* foreground and background pixels in the image, then the pixel underneath the origin of the structuring element is set to the foreground value.
If it doesn't match, then that pixel is set to the background value.

Hit-or-Miss Transform is normally used to detect or locate an object of a given shape and size in an image. The morphological hit-or-miss transform is a basic tool for shape detection.

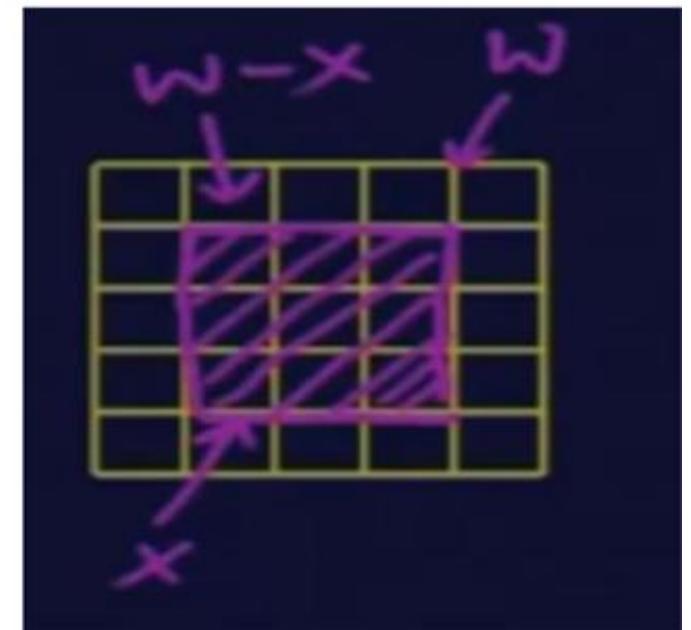
$$A \otimes B = (A \Theta B_1) \cap (A^c \Theta B_2)$$

Where B1 is the shape to detect (foreground), and B2 is the set of elements associated with the corresponding background

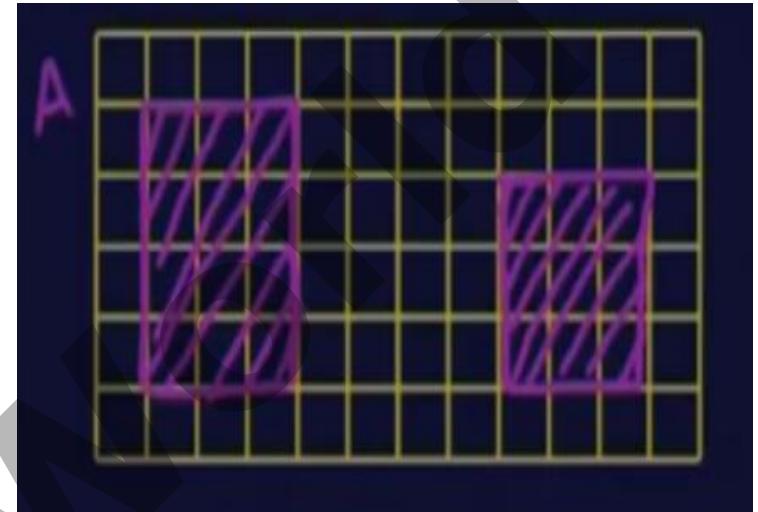
The Hit-or-Miss Transform is given by

$$A \circledast B = (A \Theta X) \cap [A^c \Theta (W - X)]$$

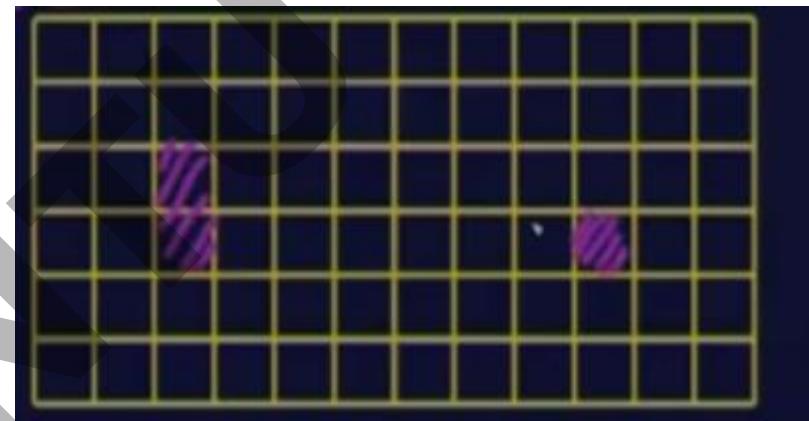
X is the object



And the image in which
the object is to be detected is:



Perform $A \Theta X$:
the output obtained is:

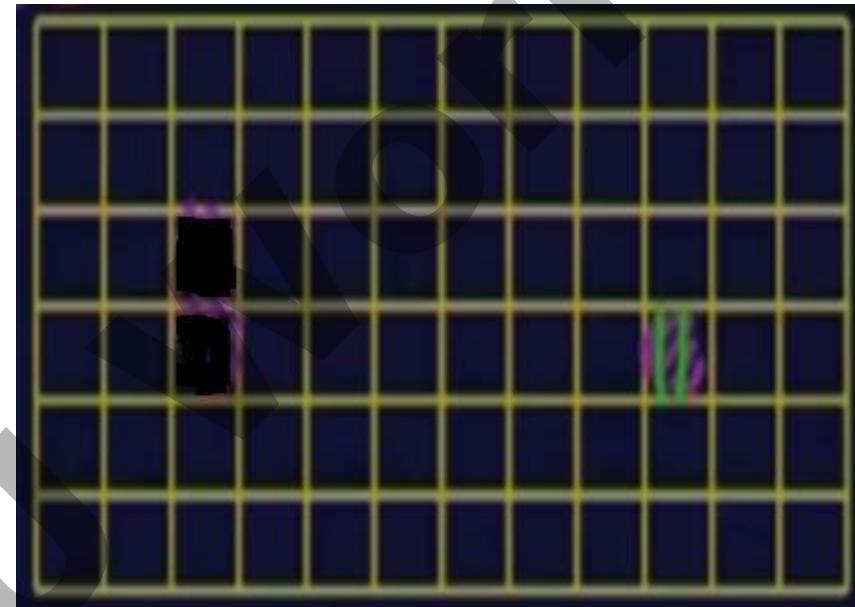


AUJAN

Now, perform $A^c \Theta (W-X)$

We obtain only 1 pixel shown in yellow:

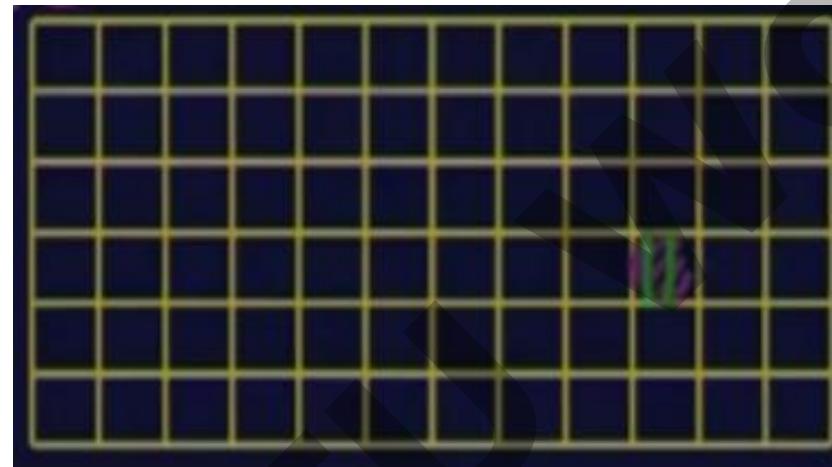
We get, the location of the object in the image



Taking the intersection between the two results: i.e.:

$$A \odot B = (A \Theta X) \cap [A^c \Theta (W - X)]$$

We get, the location of the object in the image



Applications of morphology

is extracting image components that are useful in the representation and description of shape.

Medical image analysis: Tumor detection, measurement of size and shape of internal organs, Regurgitation, etc.

Robotics: Recognition and interpretation of objects in a scene, motion control and execution through visual feedback.

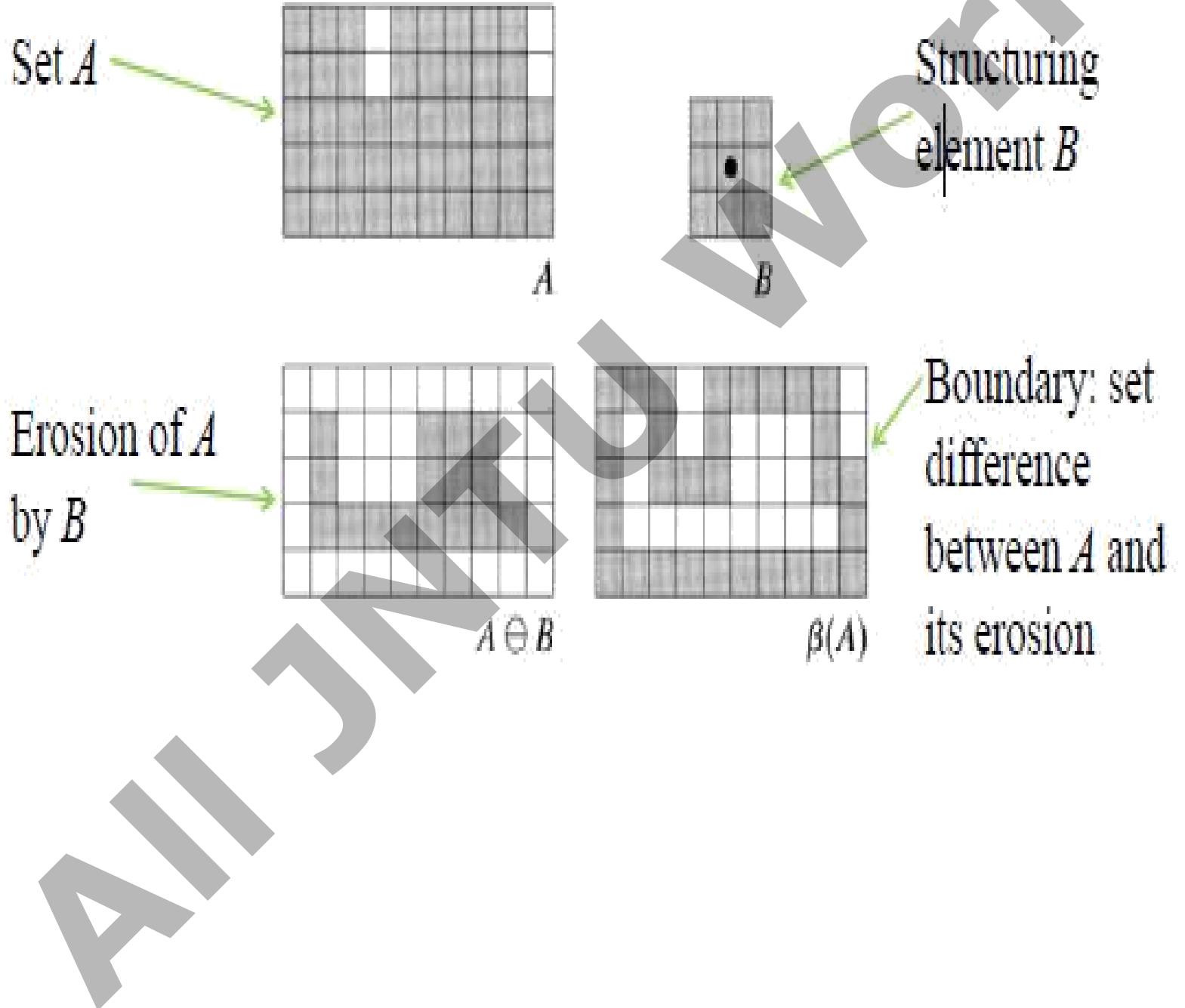
Radar imaging: Target detection and identification.

Boundary Extraction

The ***boundary*** of a set A , denoted as $\beta(A)$, can be obtained by first eroding A by B and then performing the set difference between A and its erosion as follows:

$$\beta(A) = A - (A \ominus B)$$

where B is a suitable structuring element.



Corner Detection with Hit-and-miss Transform

Structuring Elements representing four corners. Usually 1s at the center

	1	
0	1	1
0	0	

	1	
1	1	0
0	0	

	0	0
1	1	0
1	0	

	0	
0	1	1
1		

Four structuring elements used for corner finding in binary images using the hit-and-miss transform. Note that they are really all the same element, but rotated by different amounts.

The operations of [erosion](#), [dilation](#), [opening](#), [closing](#), [thinning](#) and [thickening](#) can all be derived from the hit-and-miss transform in conjunction with simple set operations

Example: Apply HMT

original

HMT

Thinning

1. Used to **remove** selected **foreground pixels** from binary images
2. After edge detection, lines are often **thicker than one pixel**.
3. Thinning can be used to thin those lines to **one pixel width**.

- Let K be a kernel and I be an image

$$\text{thin } I, K \quad I \quad \text{HitAndMiss } I, K$$

with $0 \cdot 1 = 0!!$

- If foreground and background **fit** the structuring element exactly, **then** the pixel at the origin of the SE is set to 0
- Note that the value of the SE at the origin is 1 or *don't care*!

Thickening

Used to grow selected regions of foreground pixels

E.g. applications like approximation of *convex hull*

- Let K be a kernel and I be an image

$\text{thicken } I, K$

$I \text{ HitAndMiss } I, K$

with $1+1=1$

- If foreground and background match exactly the SE, then **set the pixel at its origin to 1!**
- Note that the value of the SE at the origin is 0 or *don't care*!

Digital Watermarking

- A watermark is a pattern of bits inserted or embedded into a digital image, audio or video file that identifies the file's copyright information (author, rights, etc) which can be later extracted or detected for variety of purposes including identification and authentication purposes.
- Water marking is a secret key (string or integer) produced using a random number. This water marking is embedded redundantly over the whole image, so that every part of the image is protected.

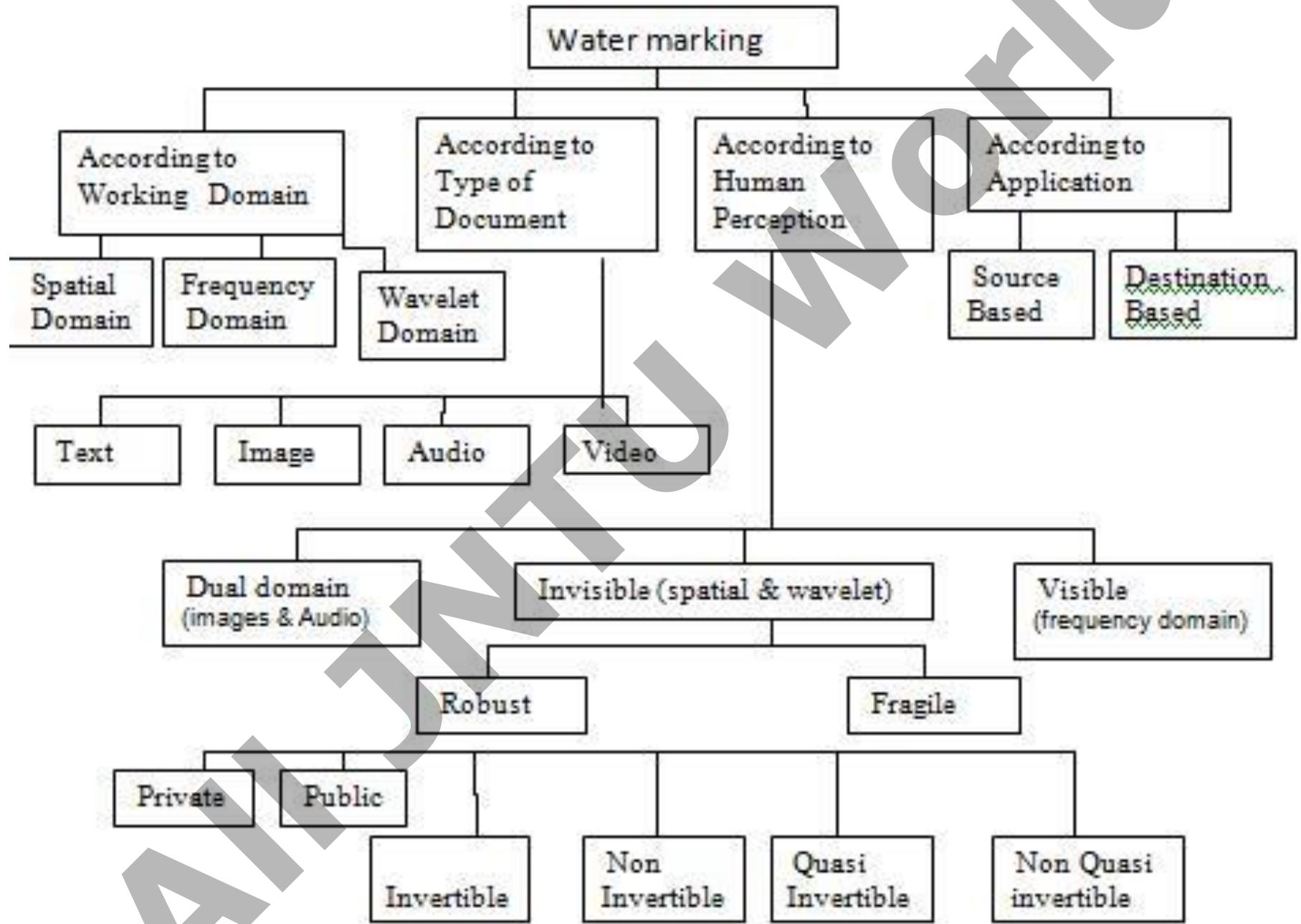
Water marked image $f_w = (1 - \alpha) f + \alpha w$

α will be between 0 and 1

If $\alpha = 1$, $f_w = \alpha w$, image becomes opaque and gets completely filled up
and image becomes obscured with watermark

If $\alpha=0$, $f_w = f$, no water marking happens.

Types of watermarking techniques



Spatial Domain Approach:

- Embeds the watermark into least significant bits (LSB) of the image pixels. This technique has relatively low information hiding capacity and can be easily erased by lossy image compression.
- “Patchwork” Approach: This technique uses a random number generator to select n pairs of pixels and slightly increases or decrease their luminosity (brightness level). Thus the contrast of this set is increased without any change in the average luminosity of the image. With suitable parameters, Patchwork even survives compression using JPEG.

Frequency domain: water mark is added to DCT coefficients

Wavelet domain: Water mark is embedded into the high frequency coefficients
Audio : Jitter :

- For audio watermarking scheme, jitter is added to the signal. The signal is split into chunks of 500 samples, either duplicated or deleted a sample at random in each chunk (resulting in chunks of 499 or 501 samples long) and stuck the chunks back together.

Robust watermark : It sticks to document (image, audio, video or text) to which it is embedded. Removing it destroys the quality of signal. It is used for copyright protection.

Fragile Watermark : It breaks very easily on modifying host signal. It is used for temper detection, finger printing and digital signature.

The dual watermarking : is combination of a visible watermark and an invisible watermark. The invisible watermark is used as protection or back up for the visible watermark.

Video watermarking The basic idea of *watermarking for raw video* is addition of pseudo-random signal to the video that is below the threshold of perception that can't be identified and thus removed without knowledge of the parameters of the watermarking algorithm.

An invisible robust private watermarking scheme requires the original or reference image for watermark detection; whereas the public watermarks do not.

Source-based watermark are desirable for ownership identification. The watermark could also be destination based where each distributed copy gets a unique watermark identifying the particular buyer. The destination -based watermark could be used to trace the buyer in the case of illegal reselling.

In order to achieve the copyright protection, the Digital Water Marking algorithm should meet few **basic requirements**

- i) Imperceptibility: The watermark should not affect the quality of the original signal, thus it should be invisible/ inaudible to human eyes/ ears.
- ii) Robustness: The watermarked data should not be removed or eliminated by unauthorized distributors, thus it should be robust to resist common signal processing manipulations such as filtering, compression, filtering with compression.
- iii) Security: watermark should only be detected by authorized person.
- iv) Watermark detection should be done without referencing the original signals.
- v) The watermark should be undetectable without prior knowledge of the embedded watermark sequence
- vii) The watermark is directly embedded in the signals, not in a header of the signal.

Digital watermarking applications

- Copyright protection: Visible watermarking is used for copyright protection which is the most important watermarking application
- Finger Printing: Finger printing is similar to giving serial number to any product. Each distributed multimedia copy is embedded with a different watermark. The objective is to convey the information about the legal recipients.
- Content Authentication (integrity protection): Invisible watermark is an evidence of ownership. The objective of this application is to detect modification in data.
- d) Broadcast Monitoring: Watermark is embedded in commercial advertisements. Automated monitoring system can verify whether the advertisements are broadcasted as contracted or not. The main use of broadcast monitoring is to protecting the valuable TV products like news items from illegal transmission.
- Indexing: Comments and markers or key information related to the data is inserted as watermark. This watermark information is used by a search engine for retrieving the required data quickly and without any ambiguity.
- Medical Applications: Patient's information is inserted as watermark in medical Images. It helps in avoiding ambiguities in searching the medical records.

watermarking scheme (algorithm)

General Framework for Watermarking

In general, any watermarking scheme (algorithm) consists of three parts.

- _ The watermark.
- _ The encoder (insertion algorithm).
- _ The decoder and comparator (verification or extraction or detection algorithm).

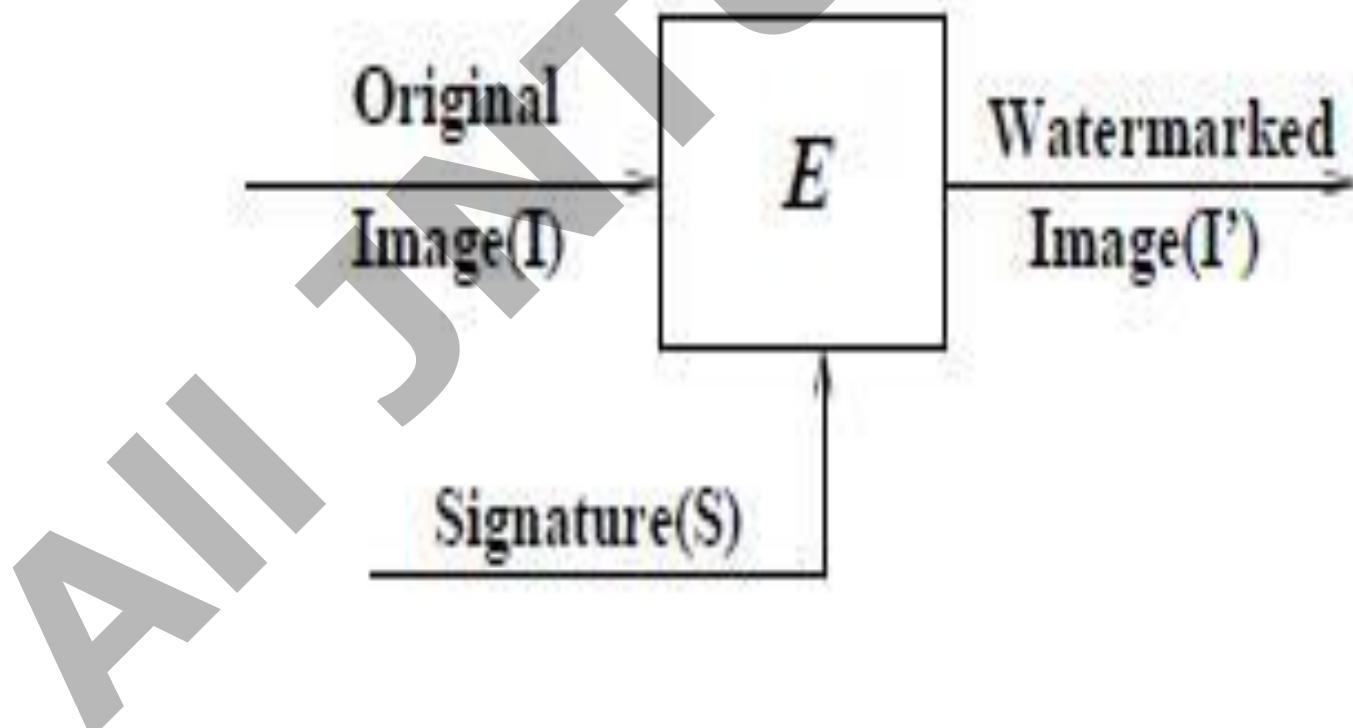
Encoding Process

Let us denote an image by I , a signature by $S = s_1, s_2, \dots$ and the watermarked image by I' . E is an encoder function, it takes an image I and a signature S , and it generates a new image which is called watermarked image I' , mathematically,

$$E(I, S) = I'$$

It should be noted that the signatures S may be dependent on image I

ENCODER



Decoding Process

A decoder function D, I' is the water marked image, I is the original image, S' is the recovered signature.

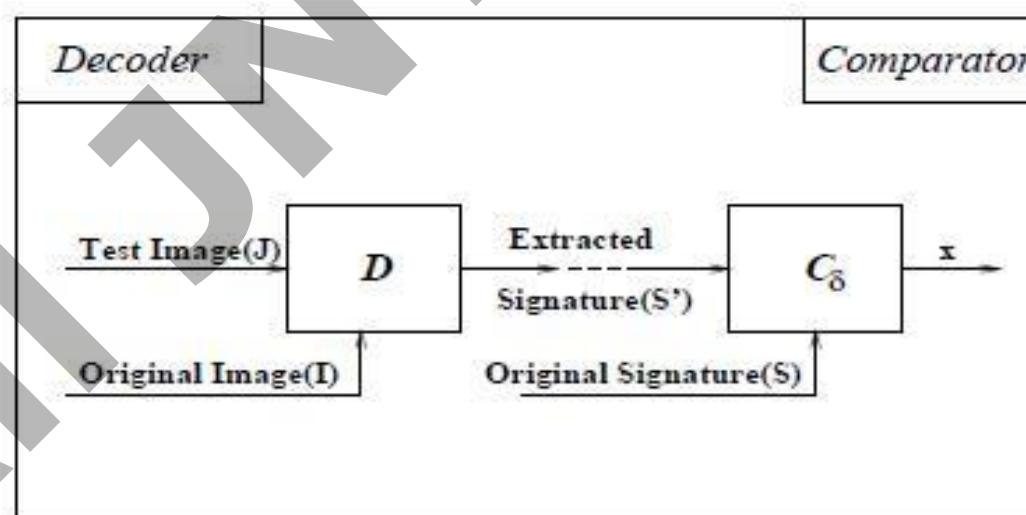
$$D(I', I) = S'$$

S' is compared with owners signature,

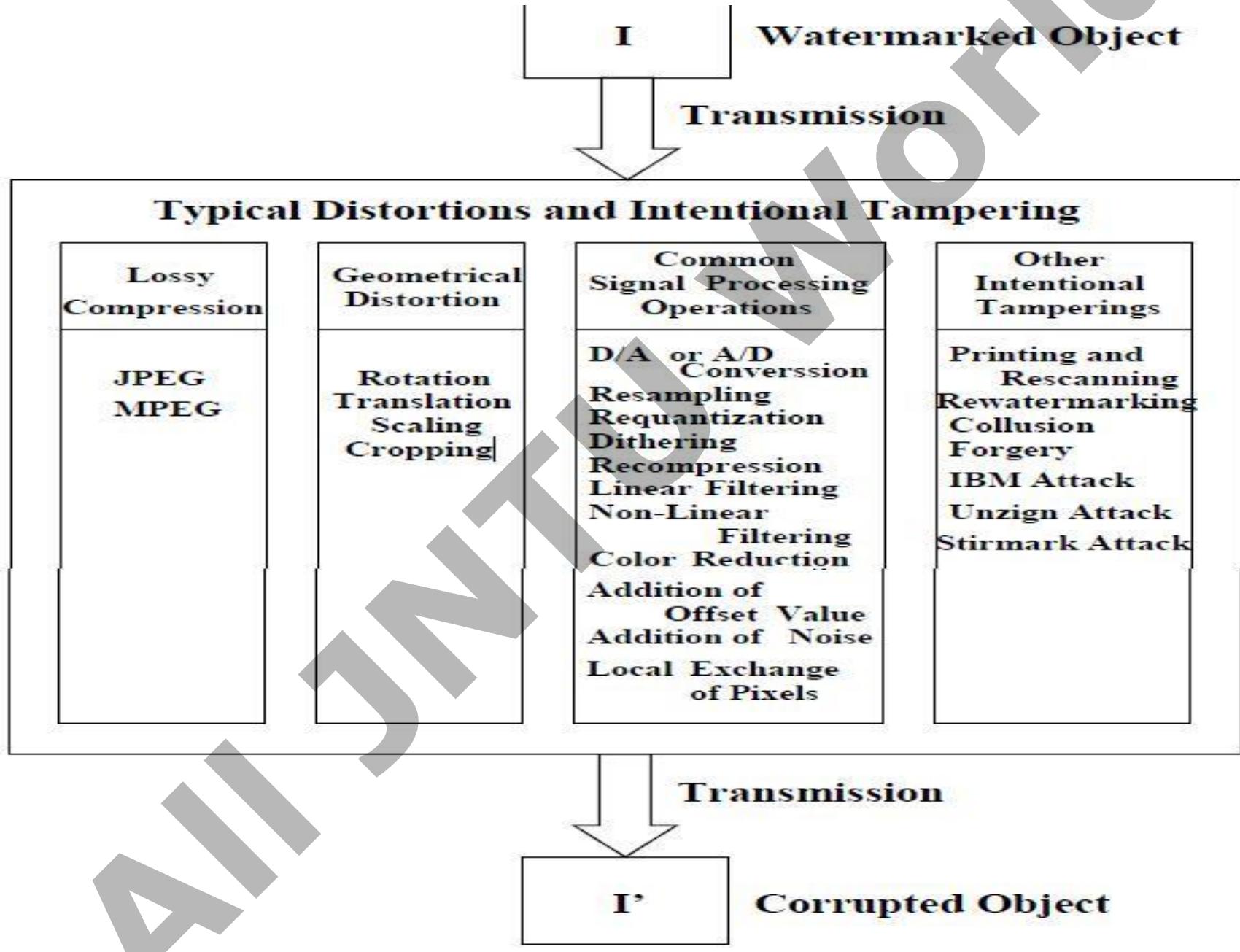
C_δ is a comparator function , δ is some threshold

Extracted water mark S' is compared with original water mark S using similarity function Sim (S , S')

$$\text{sim}(S, S') = \frac{SS'}{\sqrt{SS'}}$$



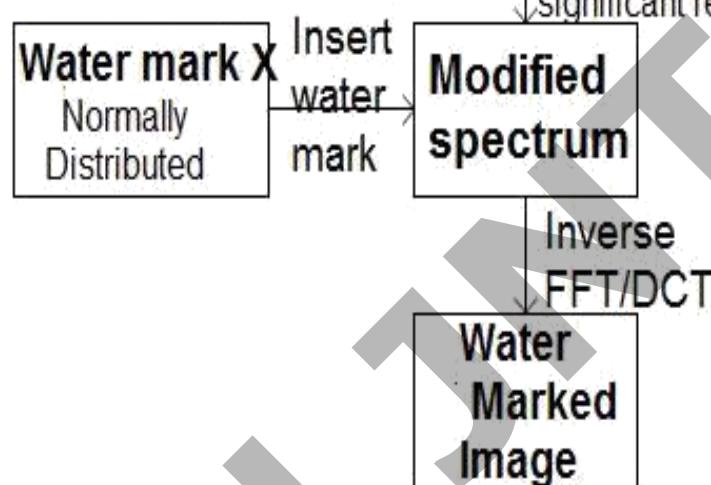
Attacks on Watermarks



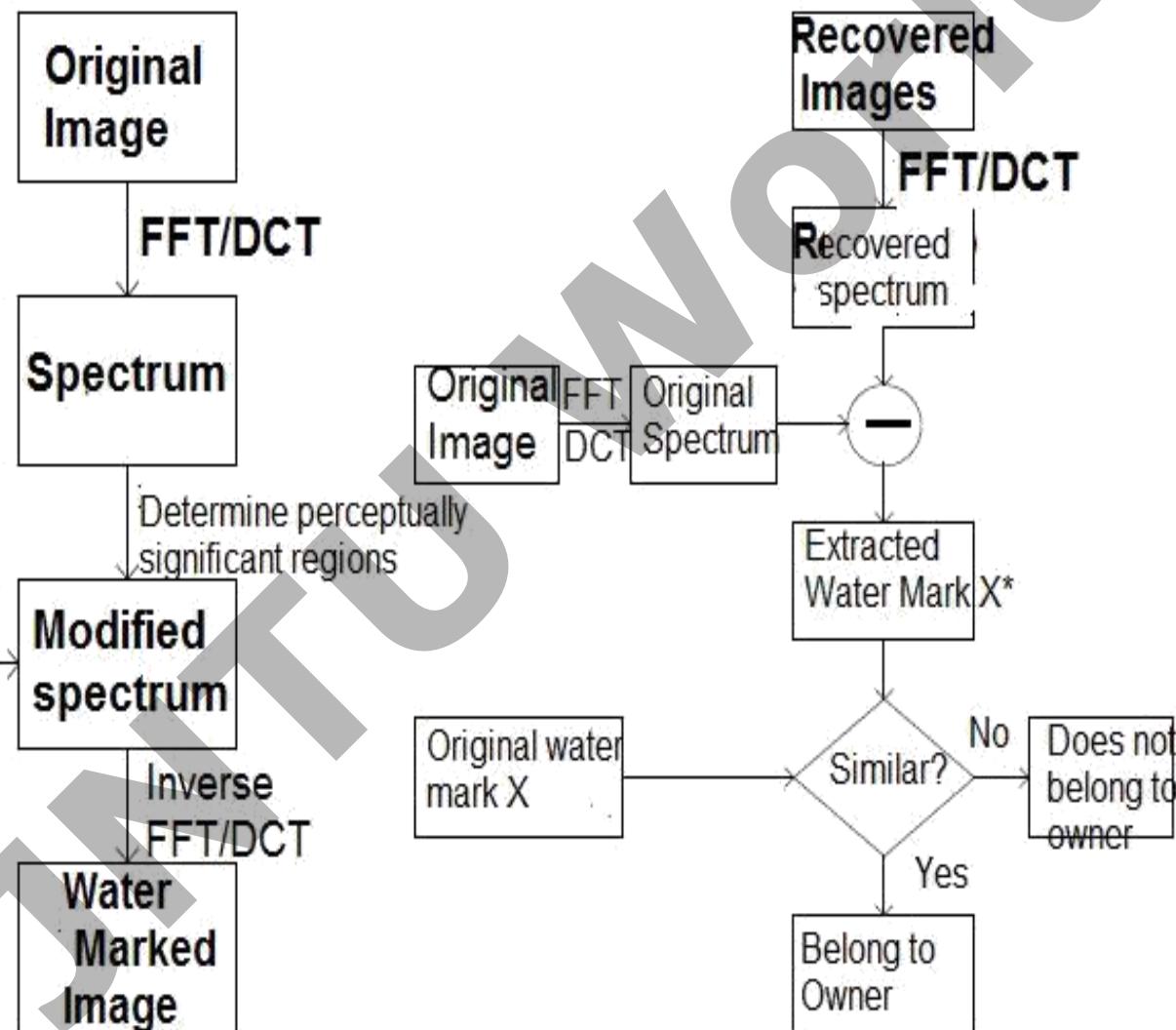
The watermark is inserted in the DCT domain of the image by setting the frequency components v_i in the image to v_i'

$$f_w = f (1 + \alpha x)$$

where α is a scale factor (say 0.1)



Insertion Process



Extraction Process

Insertion of watermark in frequency domain (invisible)	Extraction of watermark
Compute DCT for entire image	Compute DCT of the entire watermarked
Find out perceptually significant coefficients (say 1000 coeffts)	Compute DCT of the entire original image
Compute water mark $X = x_1, x_2, x_3, \dots, x_n$ Where x_i is chosen as per $N(0,1)$; N is normal distribution with mean zero and variance of 1.	The difference of the above two is the extracted watermark X^* Extracted water mark X^* is compared with original water mark X using similarity function $\text{Sim}(X, X^*)$
The watermark is inserted in the DCT domain of the image by setting the frequency components v_i in the image to v'_i $v'_i = v_i (1 + \alpha x_i)$ where α is a scale factor (say 0.1)	
IDCT to get the water marked image	