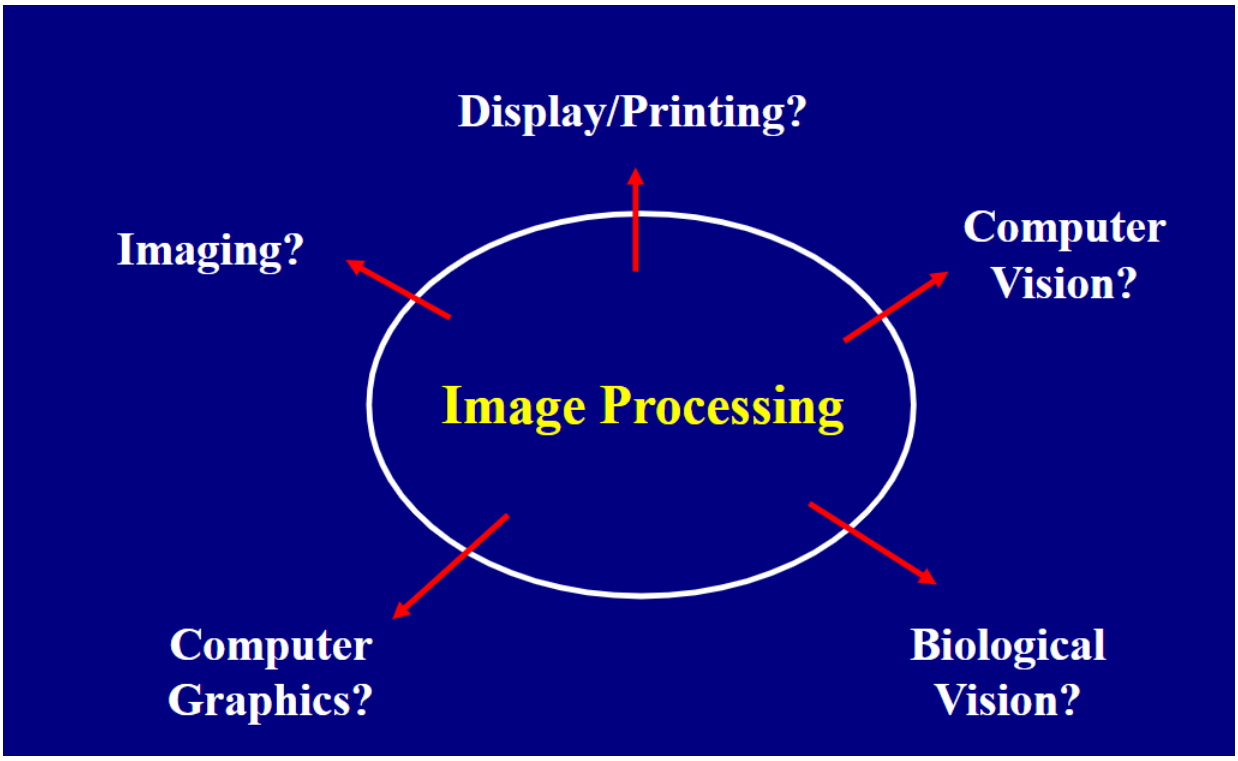


Imaging concerns on how we are going to acquire images. For example, how are we going to design cameras, CT Scans, MRI images, etc… **Display/Printing** deals with the representation of the image on some medium (paper, computer display, etc…) Once the image has been processed then it will show us the method to present image with the help of some displays or printing technologies. **Image Processing** deals with the image pixels. We firstly understand the image pixels and after that we will process that image. In the field of **Computer Graphics,** we start with objects. **Objects** are the central idea for both Computer Graphis and Computer Vision. In Computer Graphics, we are given certain objects and may be the relation between them and we only need to find is how are we going to render the objects. You basically create a picture that reflects view in the natural environment. In **Computer Vision**, we will start with the picture that will alter the objects rendered. We basically need to find the what are the objects, what is the relation between the objects, how to manipulate them, and many more.

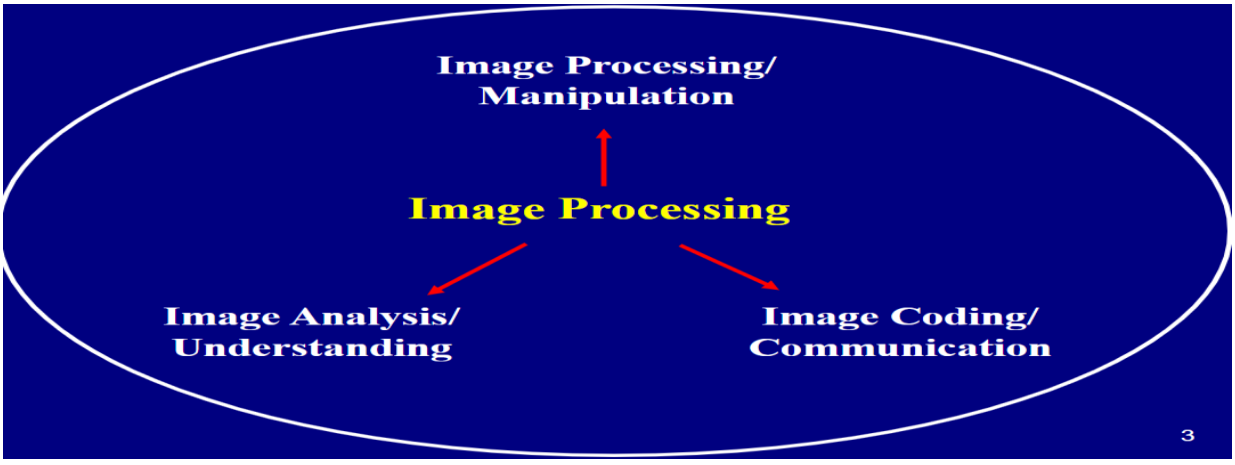
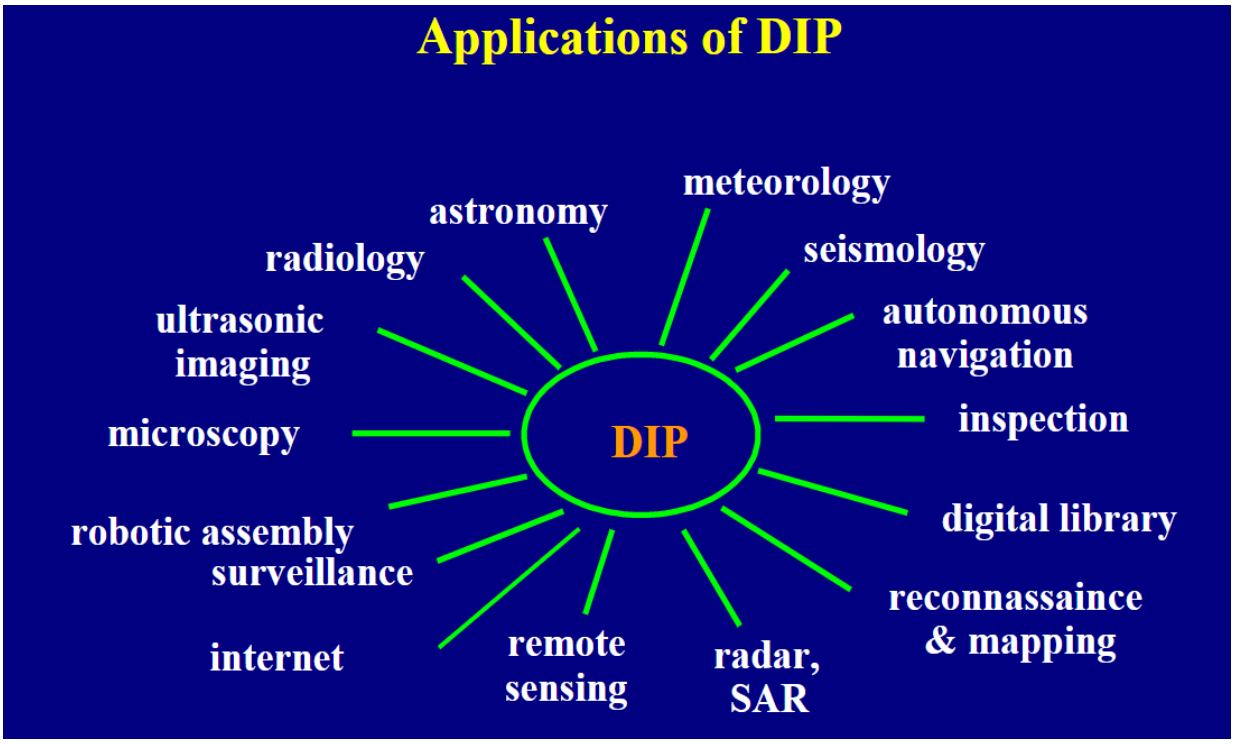


Image Processing/Manipulation is a process in which we take some image as input, process that image and output some new modified image. For example, removing the noise from an image. **Image Analysis/Understanding** is also a process in which we take some image as input, and we produce certain output that gives certain interpretation of input image. For example, if we have given an input image then we can detect some object or retrieve some meaningful information from it, another example can also be said as human face recognition, building detection, and many more. **Image Coding/Communication** is way in which we deal with the transmission of image from one location to the other. That is also known as **Image and Visual Communication**. Image coding and Image Compression is typically the same thing.

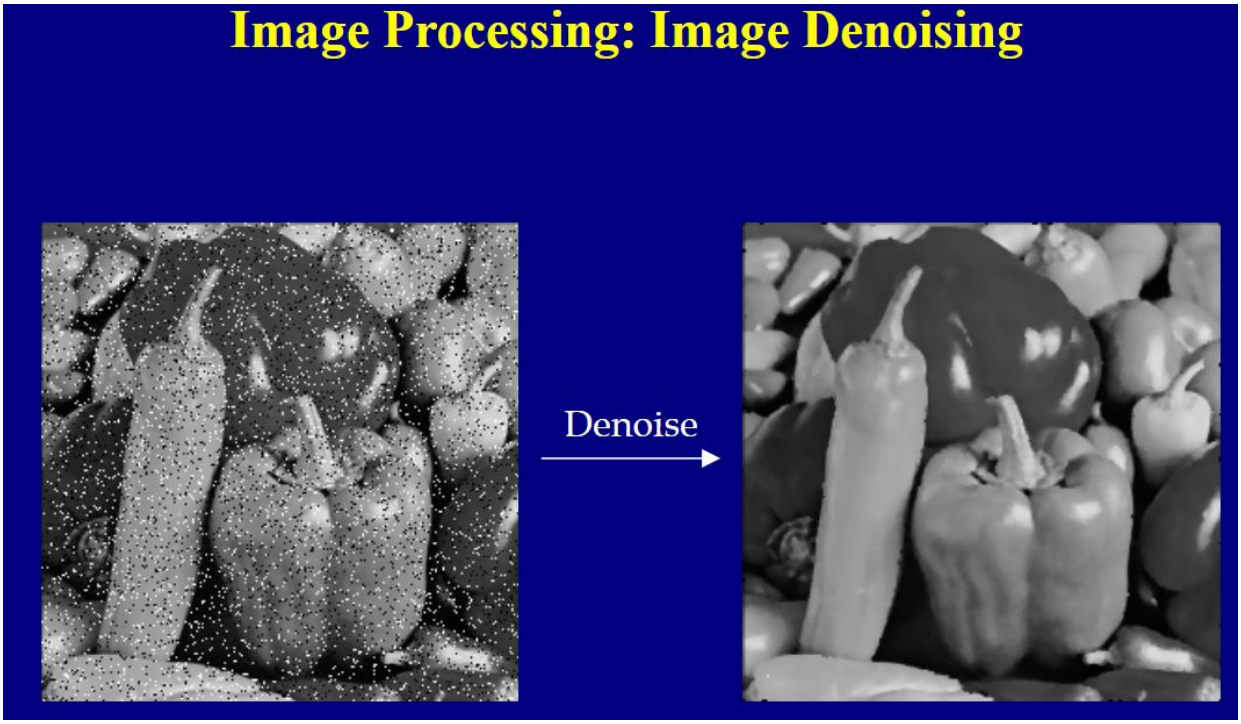


Applications of Digital Image Processing (DIP)

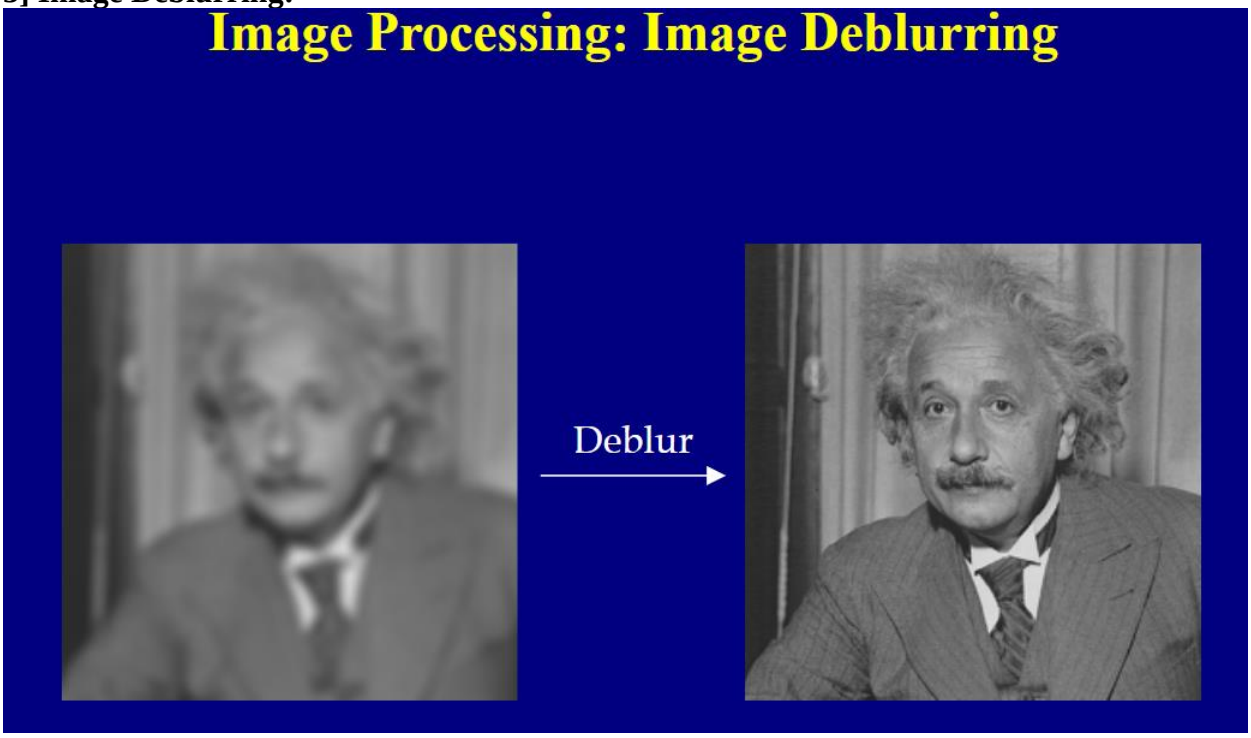
Image Processing/Manipulation:

1] **Image Enhancements:**

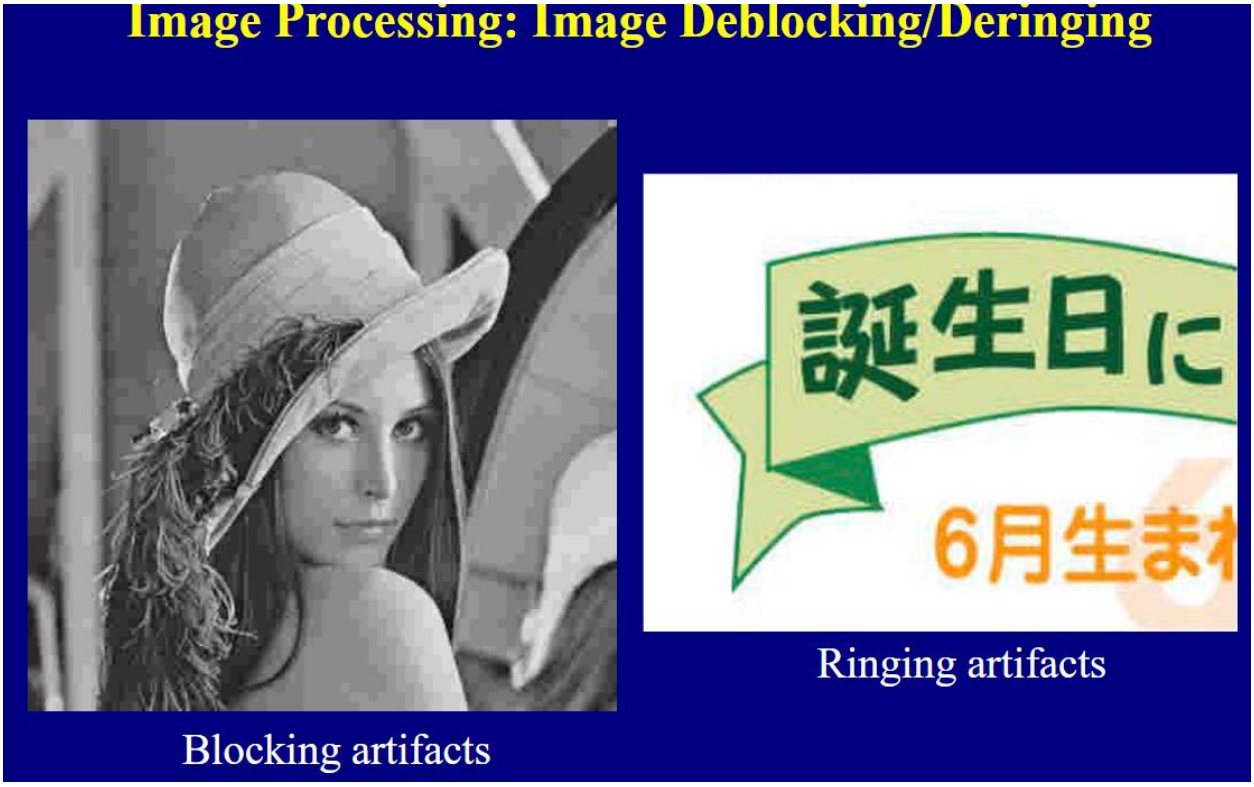
2] **Image Denoising:**



3] Image Deblurring:

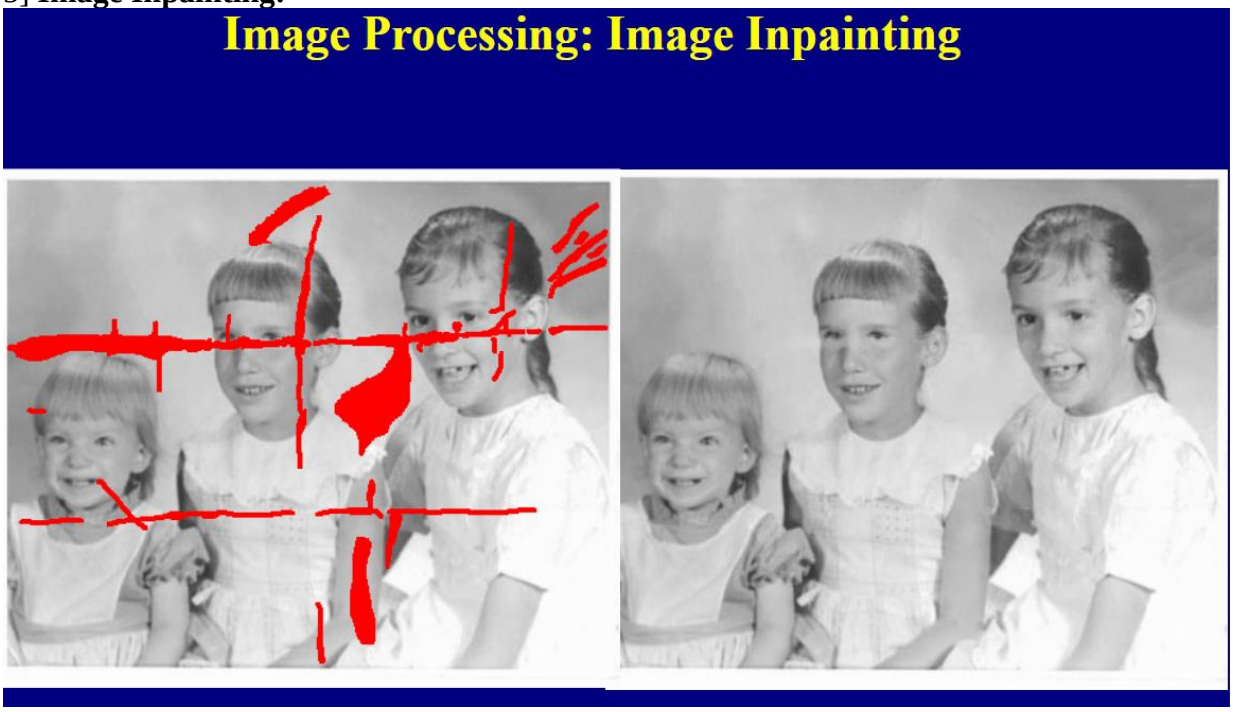


4] **Image Deblocking/Deringing:**

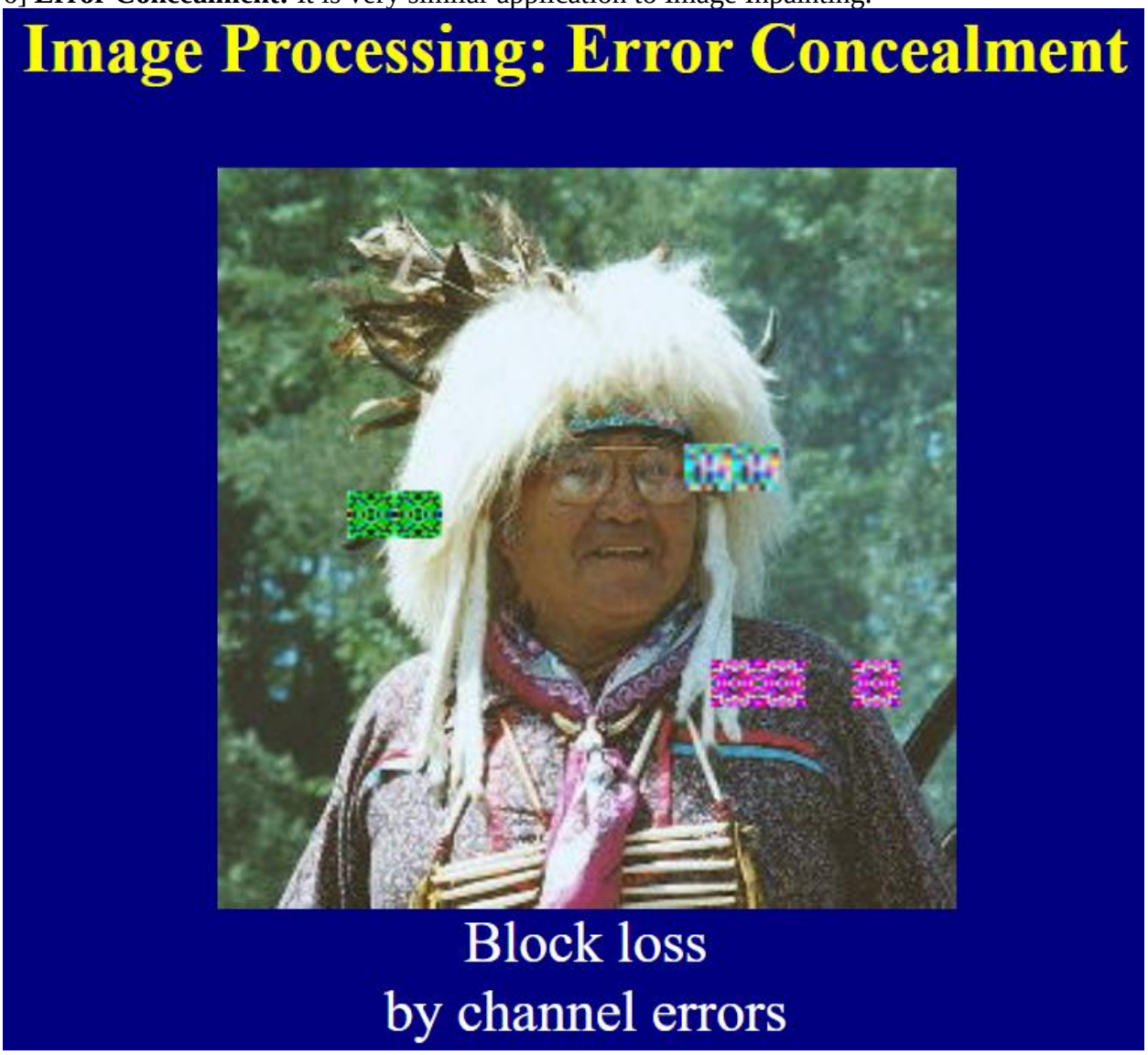


Blocking artifects and Ringing artifects are very common. If we use JPEG image and use JPEG Compresser for image compression then we will be able to see blocking artifects. If we use some other type of methods for compression then we will see Ringing artifects.

5] **Image Inpainting:**

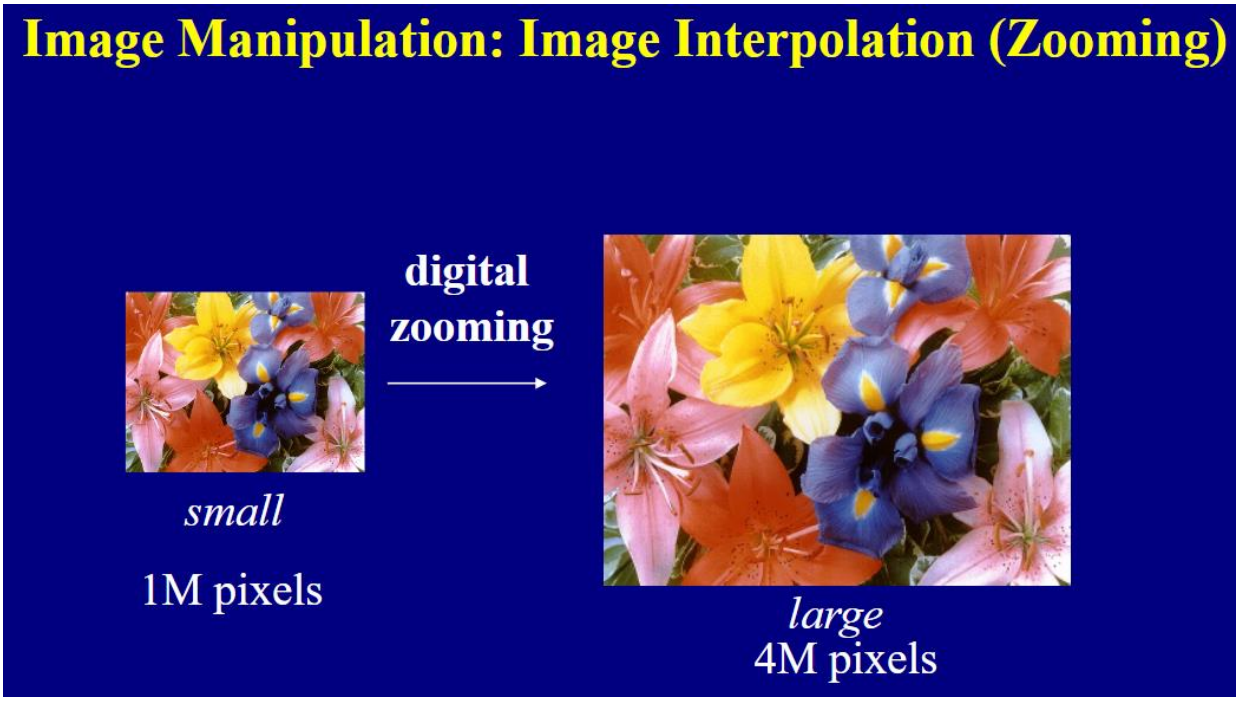


6] **Error Concealment:** It is very similar application to Image Inpainting.

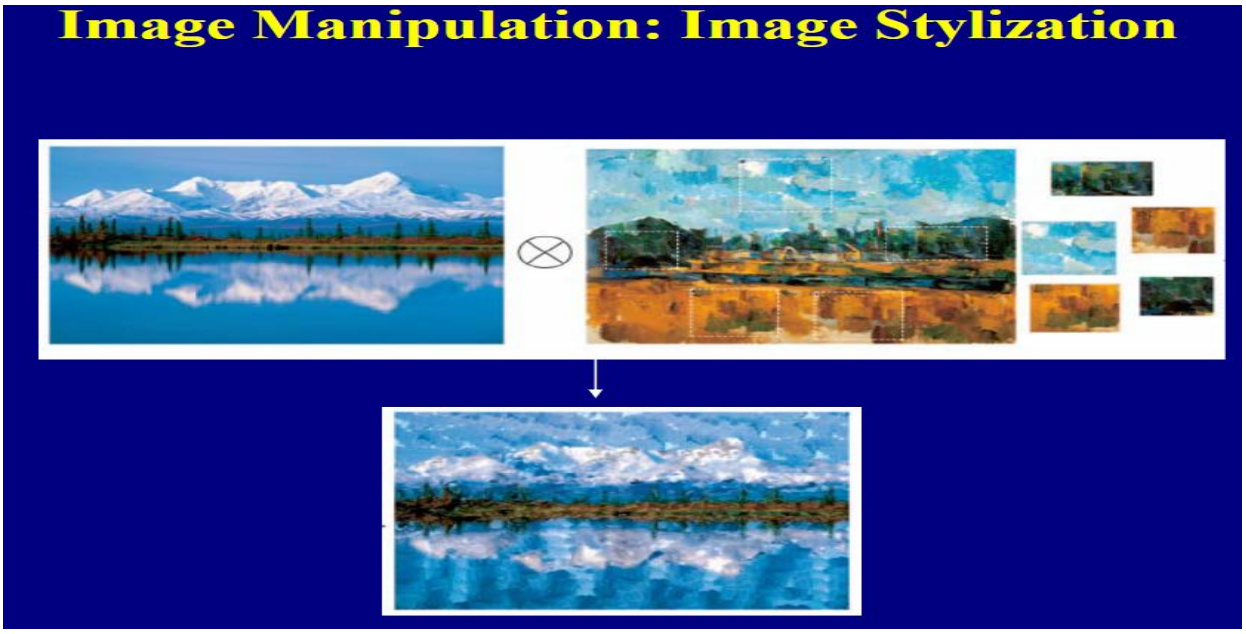


It occurs at the time of processing an image. At the time of image processing if due to some error 1 becomes 0 or visa versa then during reconstruction we will not be able to obtain our desired image.

7] **Image Interplotation:** You have to design an algorithm that can design or zoom the picture.



8] **Image Stylization:** Here we have to combine some images to produce an image that looks similar to the other.



9] **Image Fusion:**

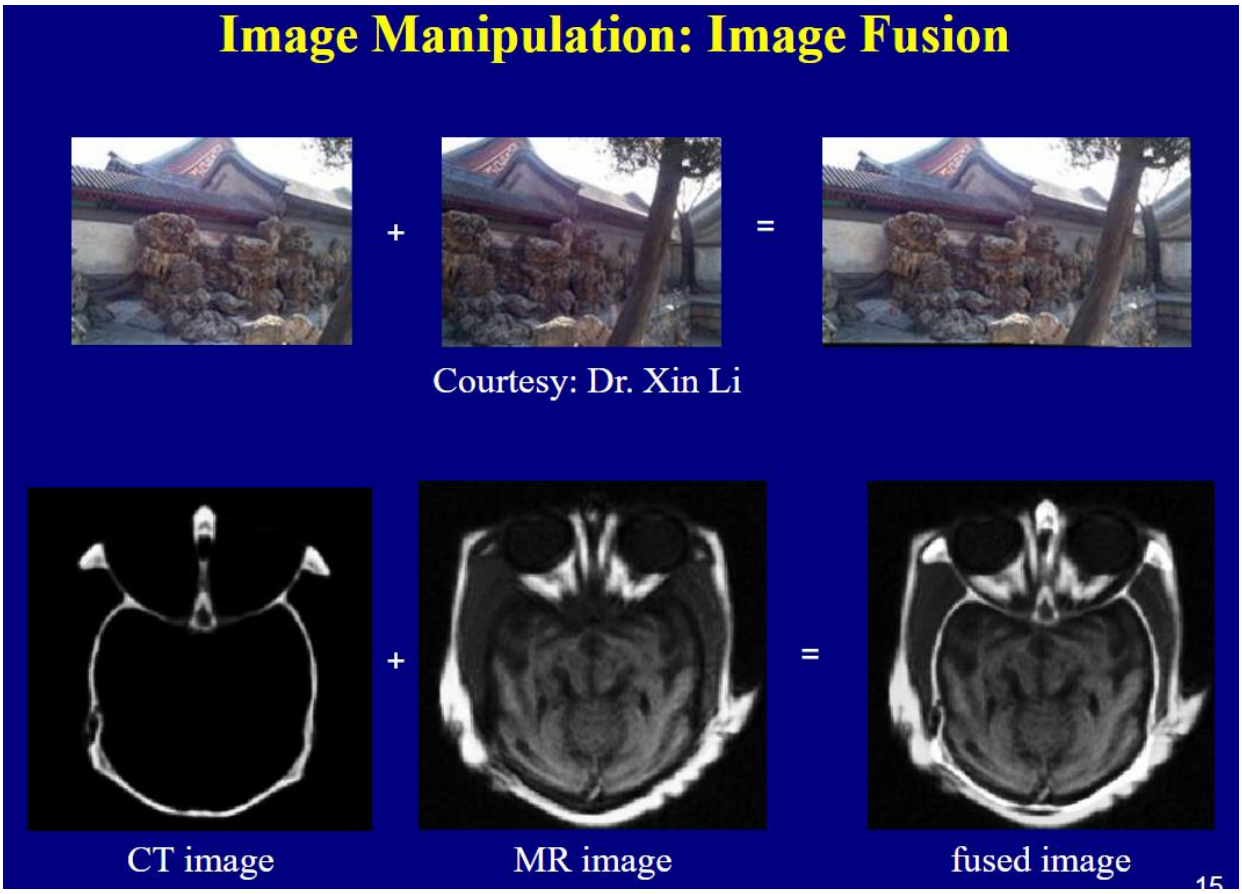
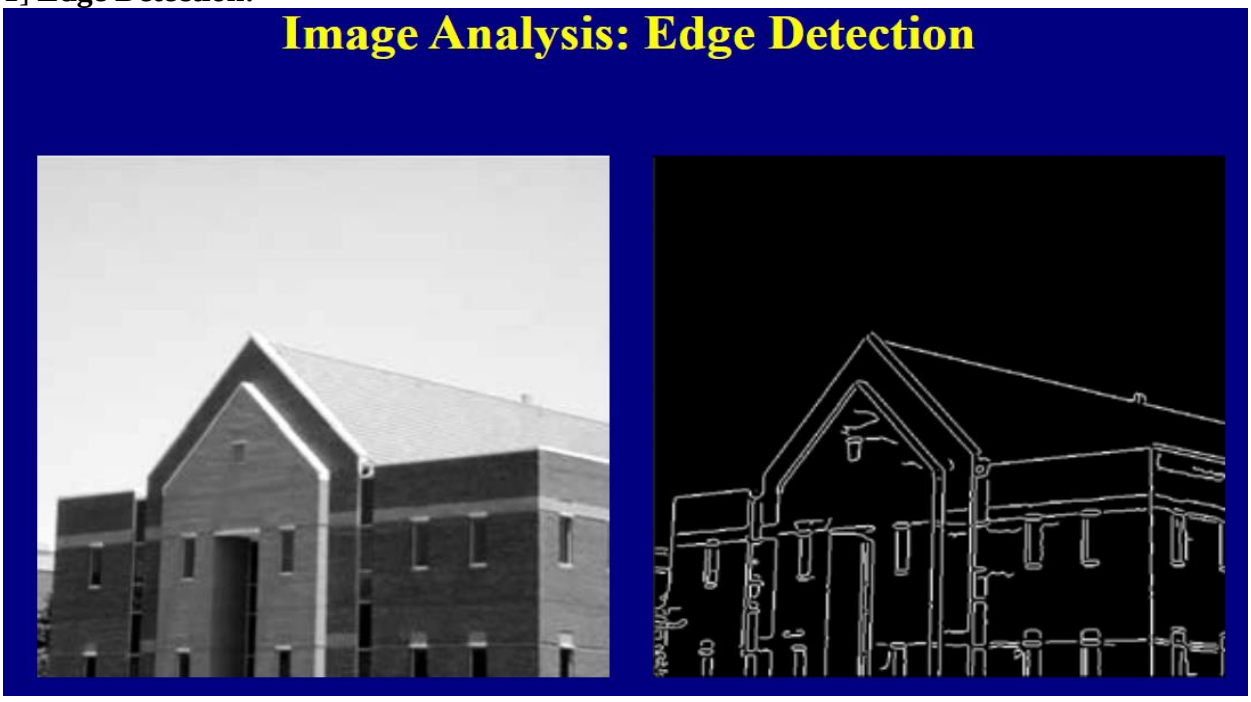
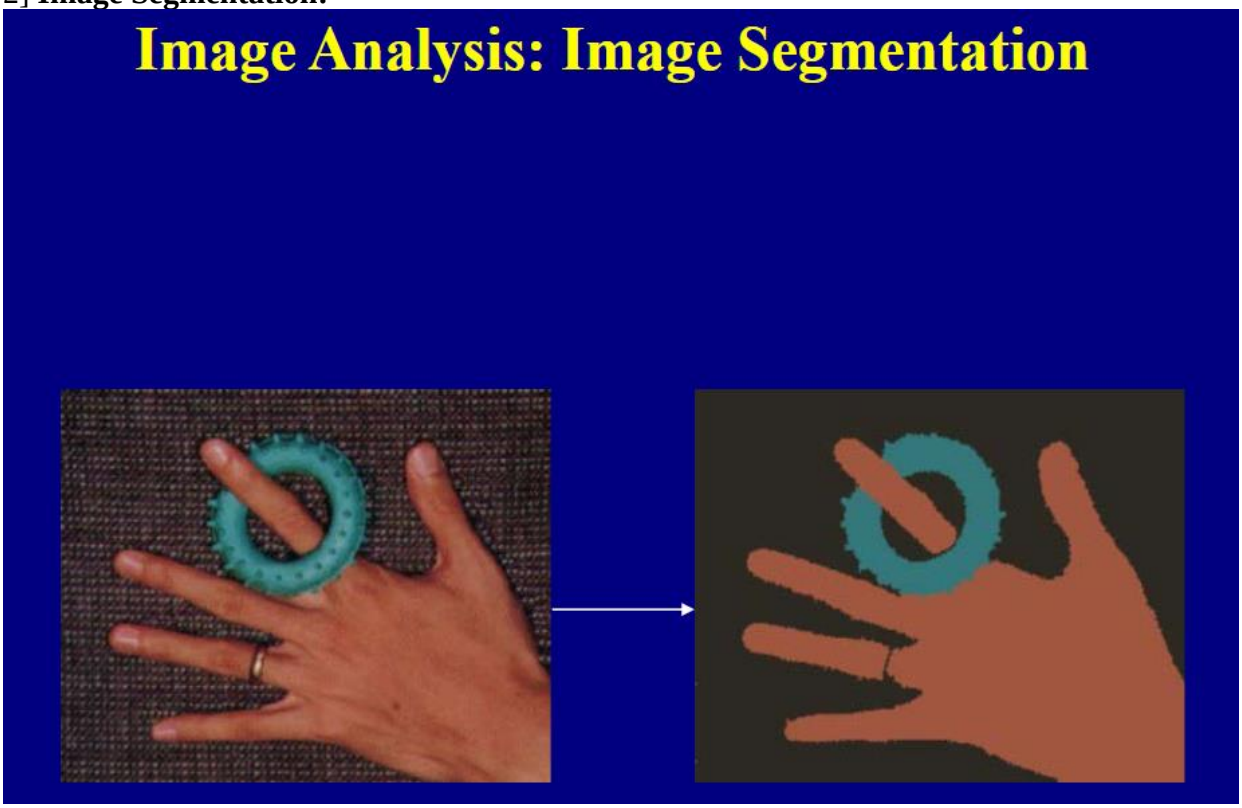


Image Analysis/Understanding:

1] **Edge Detection:**

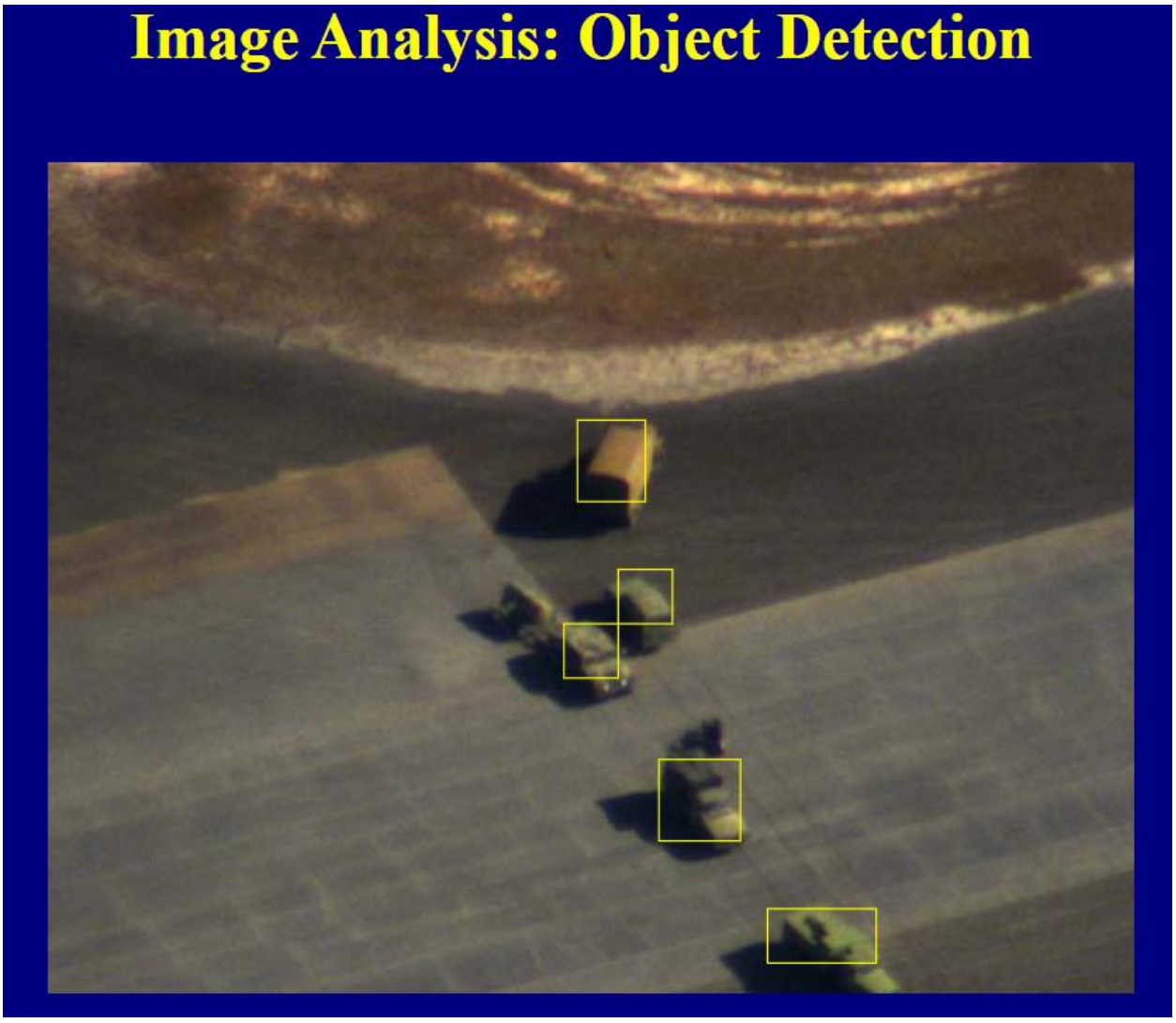


2] **Image Segmentation:**



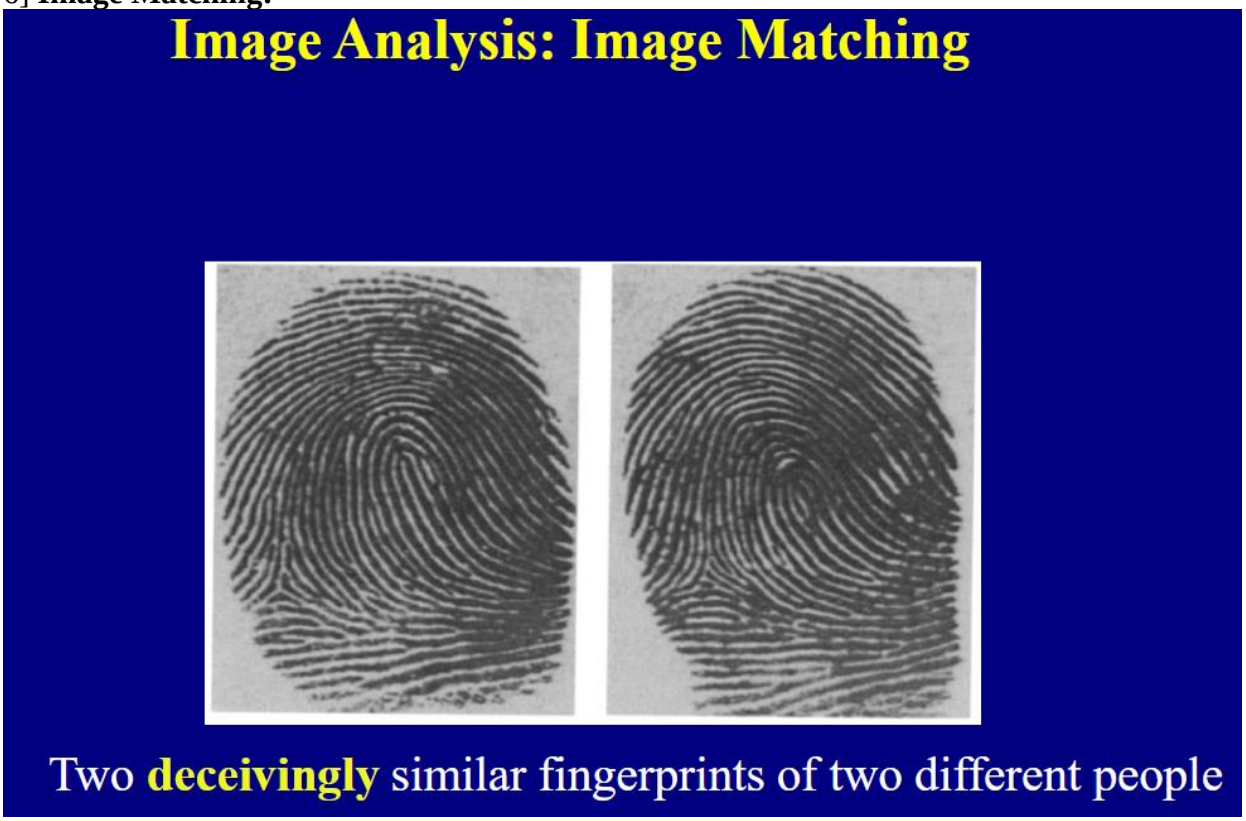
3] **Face Detection:**

4] **Object Detection:**



5] **Change Image:**

6] **Image Matching:**



7] **Image Retrival:**

8] **Video Summirization:**

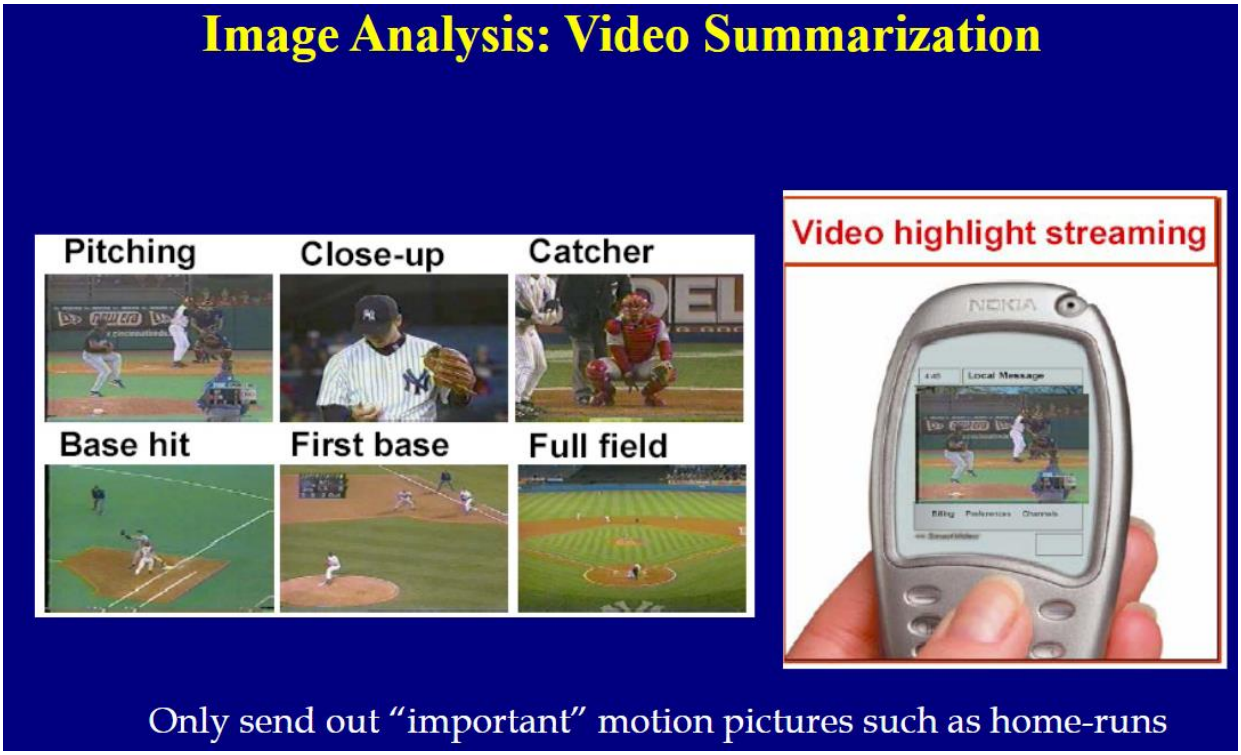


Image Transformation: It will basically take an image as an input and then transform that image into another domain, which is also known as transform domain. In the transform domain we get the new representation of image which we call them as transform coefficients.

Statistical Image Modelling: There will be a large space of images in which all the images are kept. In this large space of images, we can see some particular type of images grouped together. Grouping of typical images. These typical images will occupy extremely small or tiny amount of space in the entire domain. In this case, we are able to build strong PRIOR model in statistical sense.

What is the difference Knowledge-Driven and Data-Driven approaches?
Knowledge-driven approach depends on the domain or the specific problem you are working on and then you convert your knowledge to make computational model to solve problems. There are two disadvantages because of which we cannot use Knowledge-driven approaches, which are mentioned below:

- Require deep domain knowledge.
- Too difficult for complex problems.

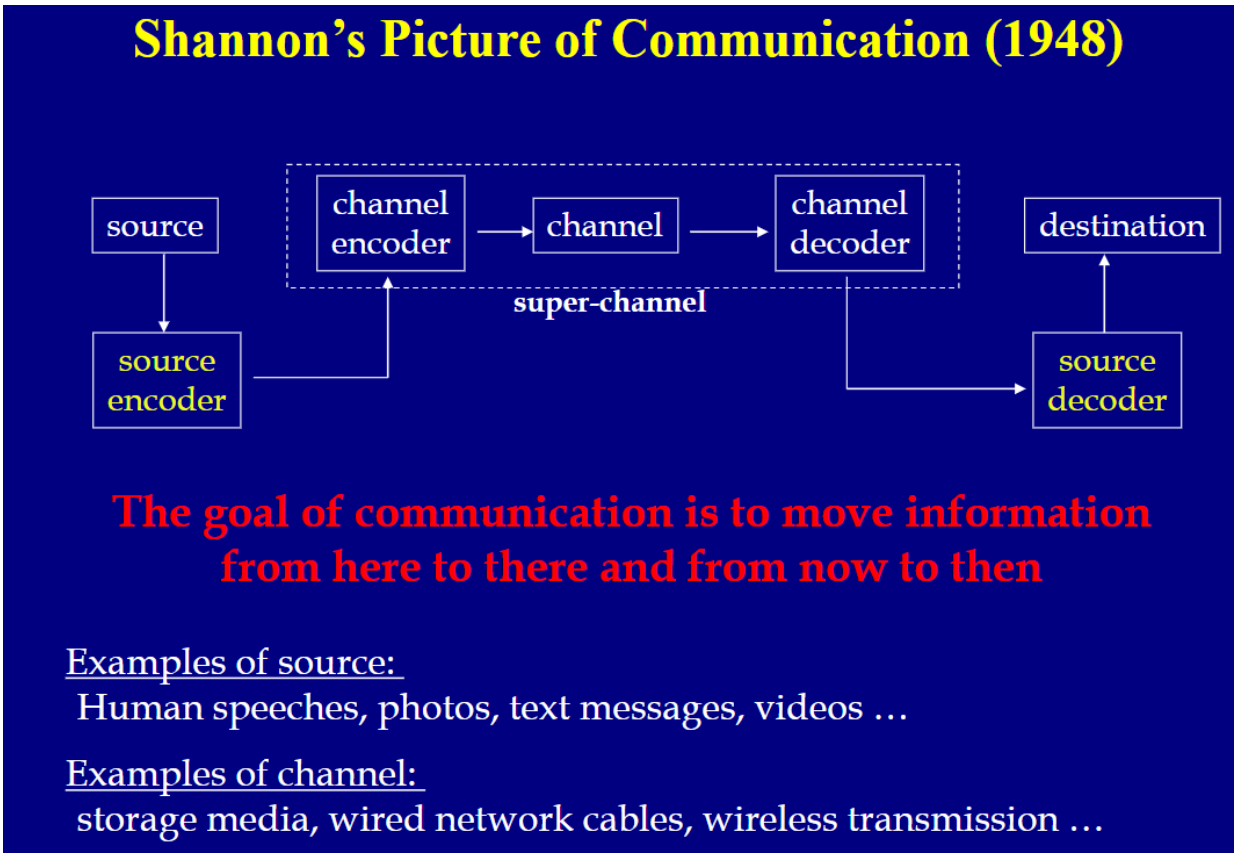
Because of these 2 reasons we use Data-driven approaches. Data-driven approaches will collect sufficient data and learn rules from the data. We do not require any domain knowledge. All we need is the data.

With the help of DNN (Deep Neural Networks) we are able to solve all the problems related to image processing. Here are a few limitations of Data-driven approaches:

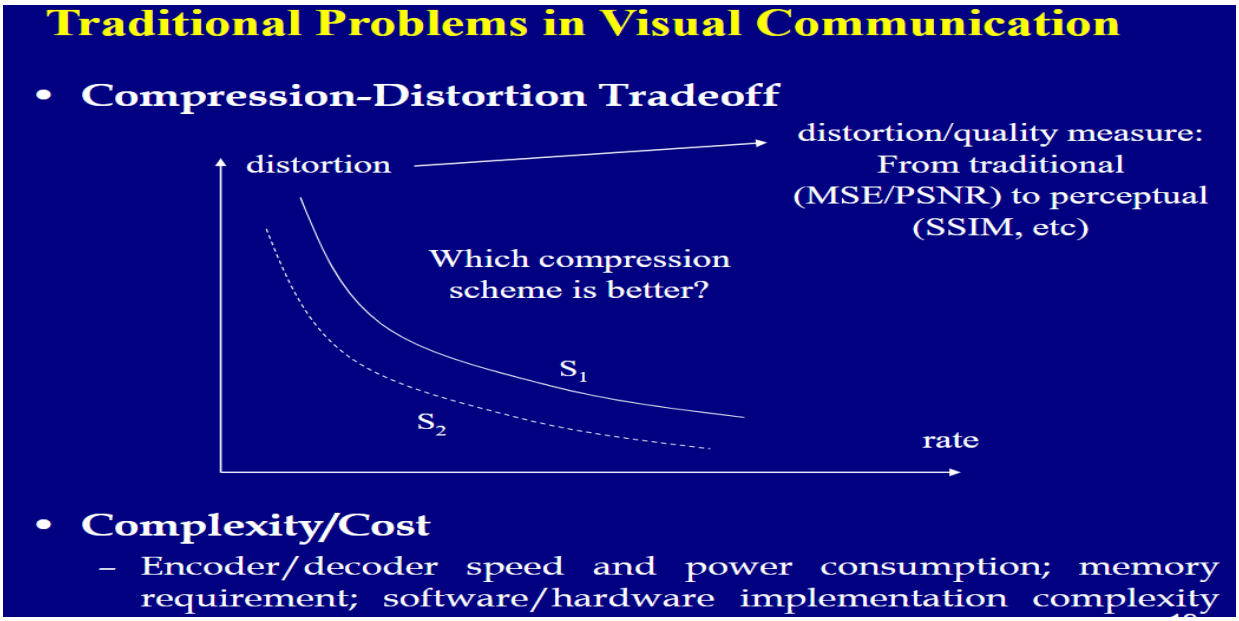
- Weak interpretability
- Weak transferability

- The data challenge

Visual Communications:



The above picture is self-explanatory.



Here is the rate-distortion graph shown which tells us that what process we need to follow to compress the image. Here, the rate is the bit-rate for the compressed image and the distortion is the quality of the image compressed. Rate-Distortion curve (RD curve). Here among S1 and S2, S2 is the better compression algorithm because of 2 reasons:

- If we consider same rate for S1 and S2 then S2 has less distortion as compared to S1.
- If we consider same distortion for S1 and S2 then S2 has less rate as compared to S1.

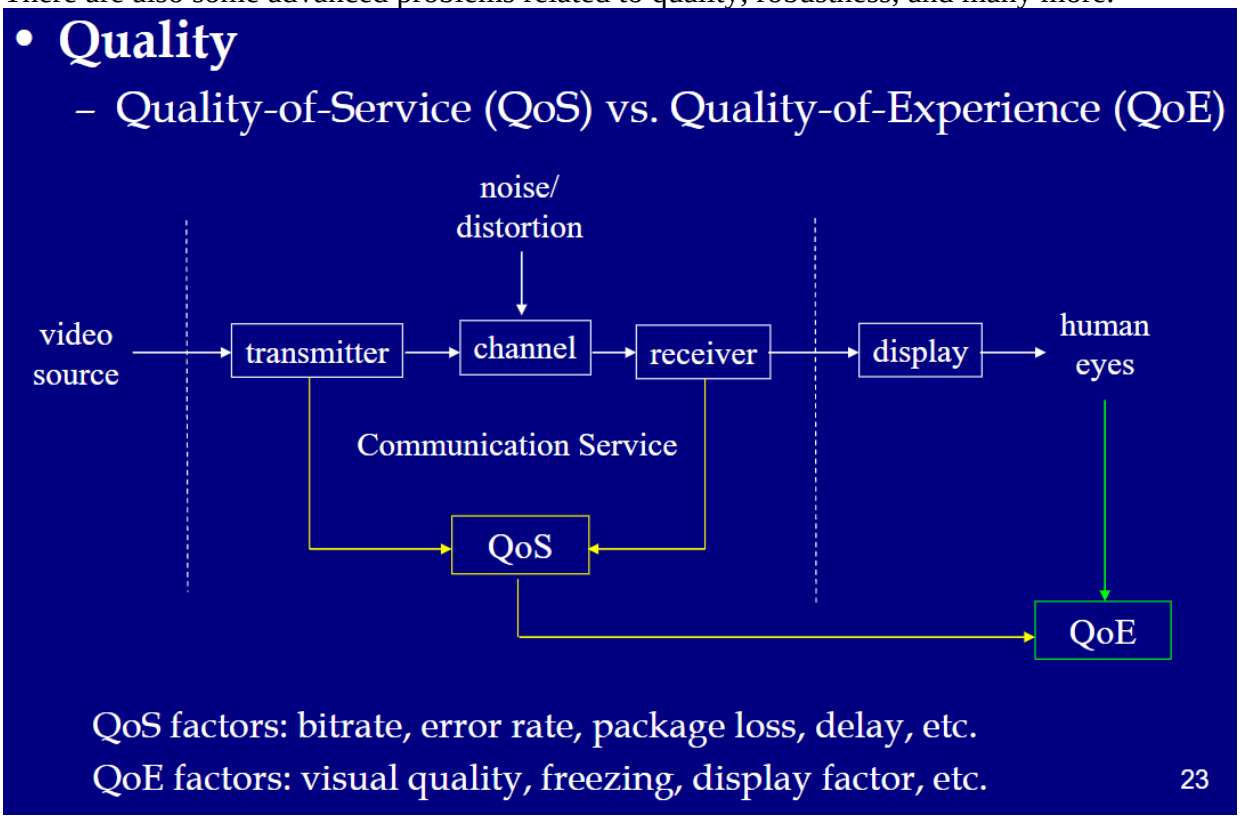
Another problem is also of complexity/cost. We need to look what resources are available for compression that will eventually determine the cost or complexity.

There are two types of image compression: a] Lossy compression b] Lossless compression

- Lossless image compression: Here, we will have all the infiormation preseerved meaning that after compression and decompression you will have 100% original image preserved by every single pixel value. Here we have low compression ratio.
- Lossy image compression: Here, we basically lose information. We have original stream of image and we are able to recover almost all of them after reconstruction, but it will not be exactly same as the original image. Here, we have the high compression ratio. If you compare pixel by pixel you will not notice a major difference.

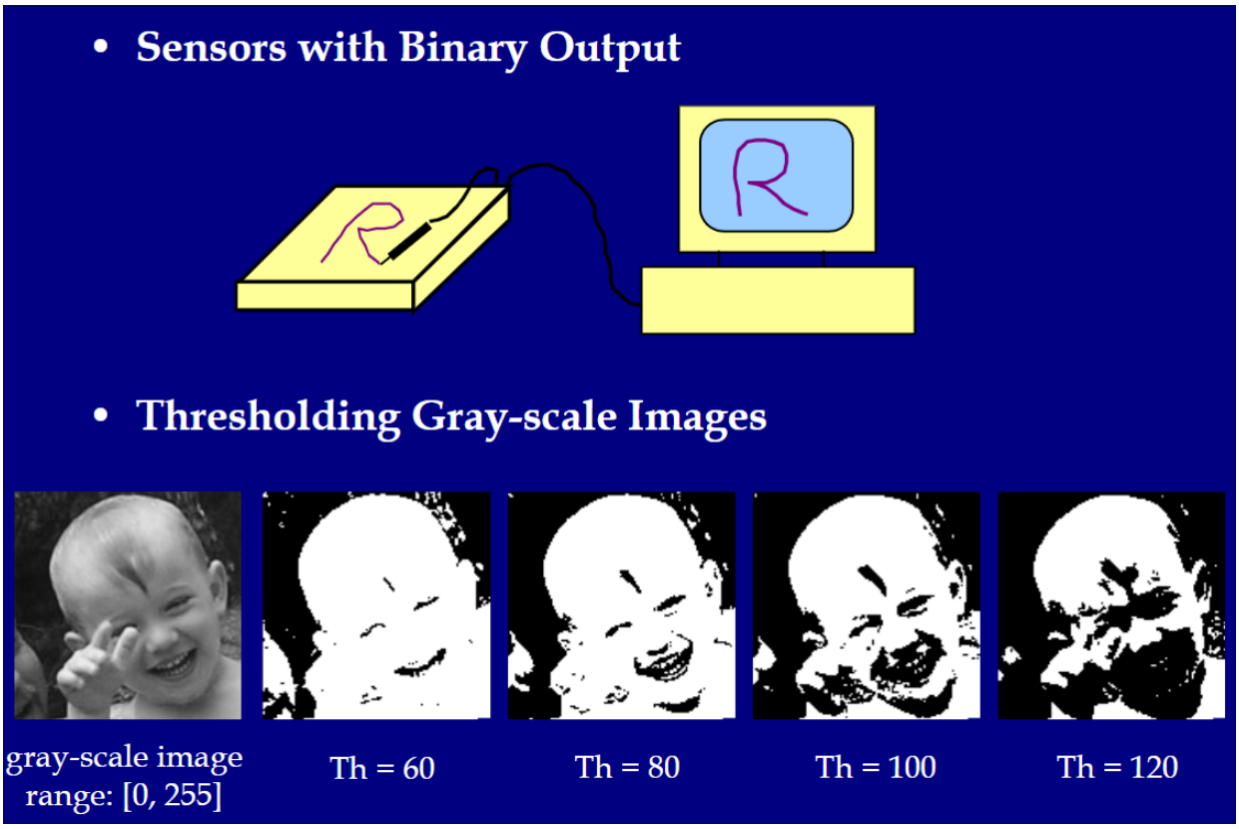
JPEG compression algorithm is not better than JPEG 2000. Having the same compression ratio for both of the algorithms JPEG 2000 performs better than JPEG.

There are also some advanced problems related to quality, robustness, and many more.

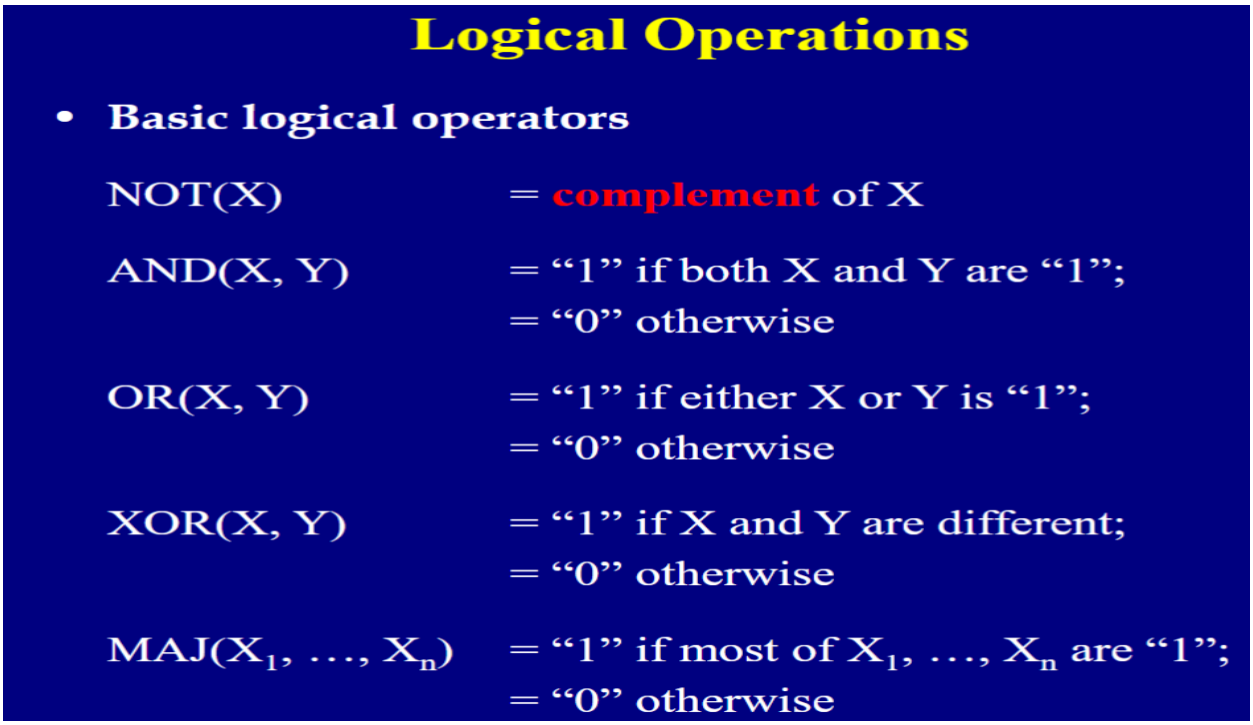


Binary Image Processing:

It is the simplest form of images. The pixel value of an image contains the binary value. It can have either ‘0’ or ‘1’ as the values. You can use ‘0’ to represent a white pixel color, and ‘1’ to represent a black pixel color. It can also be said as 1 bit/pixel. The Binary Images are generated in many different ways. You can generate binary images by sensors with binary output and also you can apply threshold on grey-scale images to produce binary images.

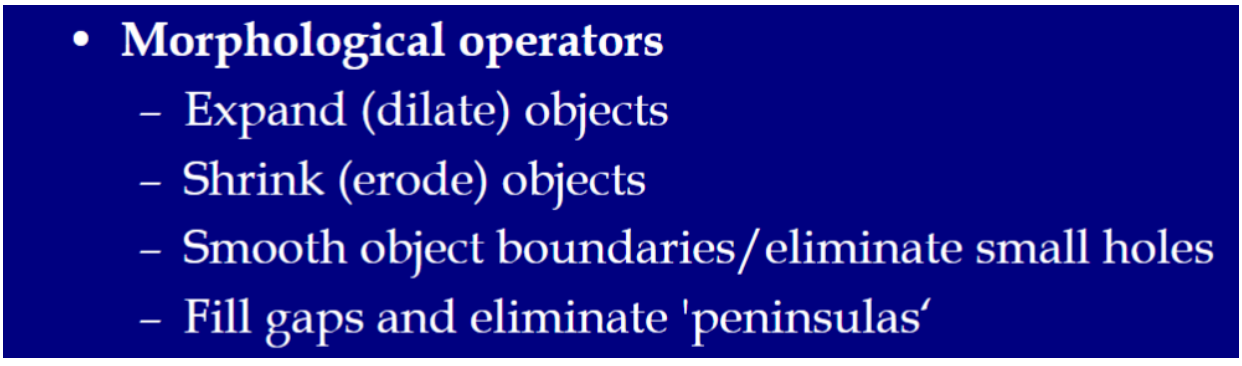


Binary images with different threshold levels.

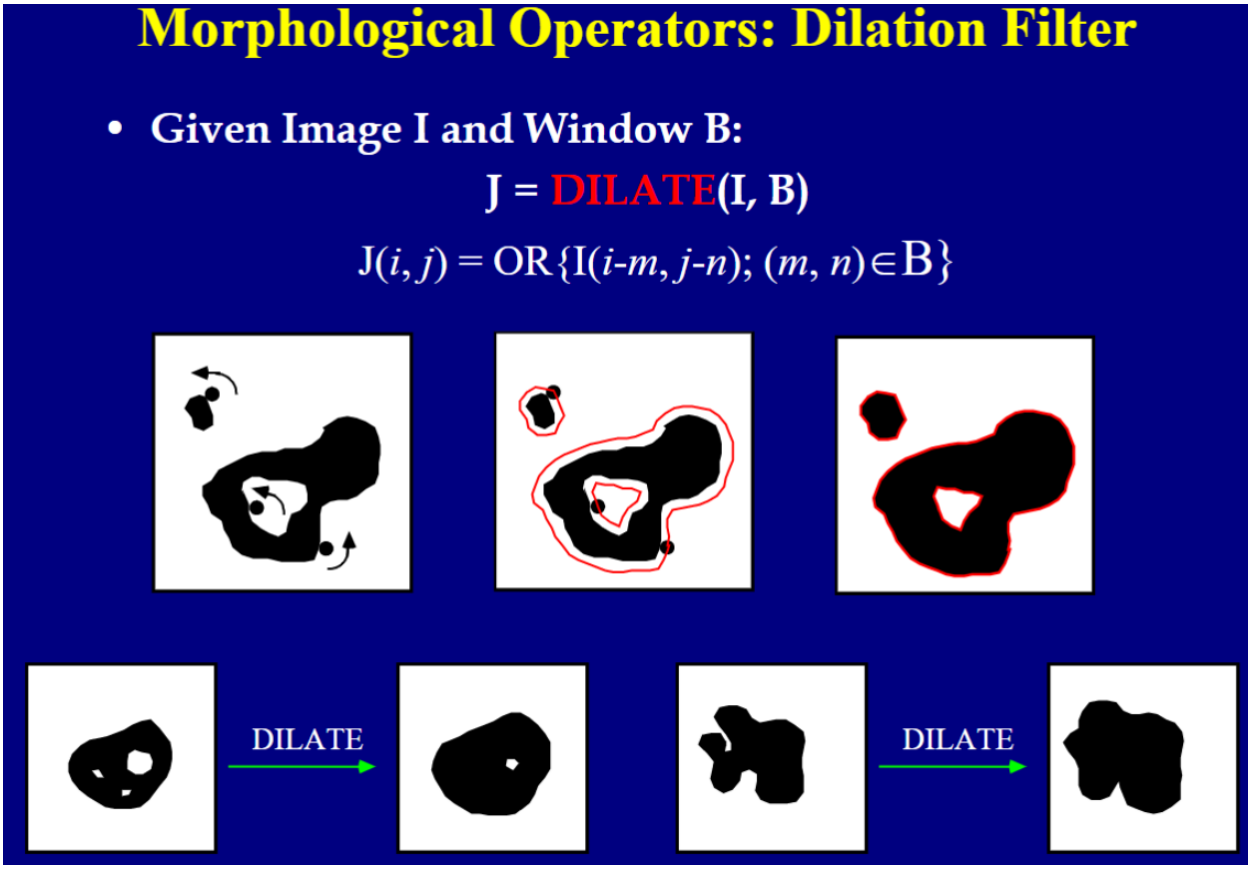


We can also perform the logical operation on the pixel value of an image as shown in the above image. On single bit pixel value, we can perform NOT operation. In 2 bit pixel value, we can perform AND, OR, XOR operation. If we have a group of pixels, then we can perform MAJ operation.

Morphological Operation:
We use Morphological operators to process binary images. In order to define a morphological operator, we have to define a window. The window can be of any shape (square, rectangle, diamond, etc…). Once we define a window then we have to slide this window step by step. By moving this window step by step we also perform some logical operation and update the value of the center pixel. We will perform this for the entire image. Now this image will be the processed binary image.

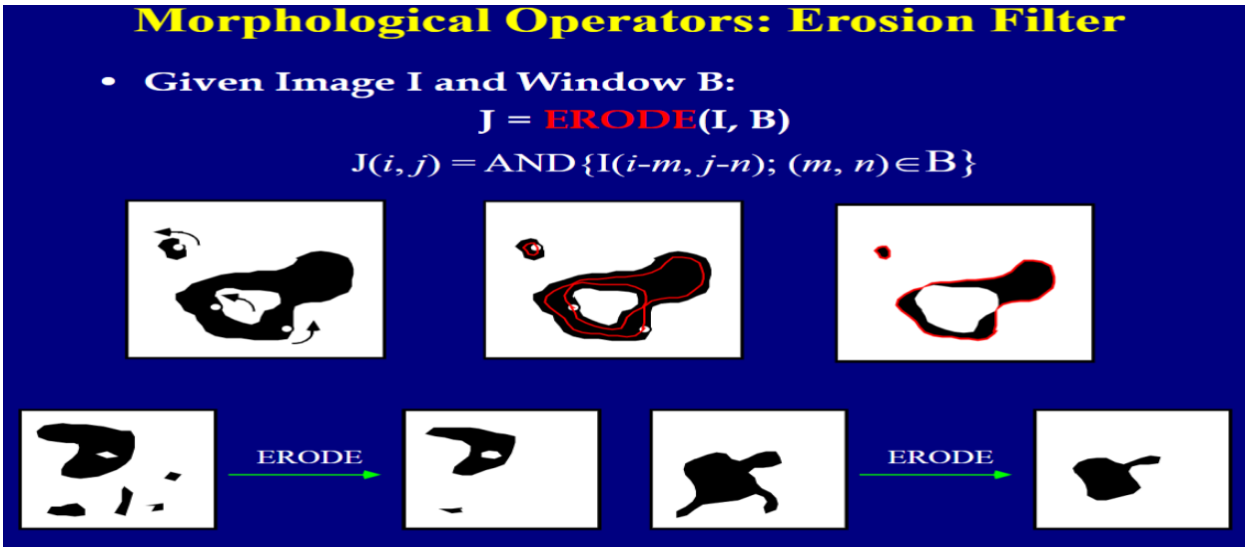


Example of Dilation Filter:
We are defining a window B which is applied to Binary Image I as shown. This window will slide through the whole image I and it will perform OR operation.

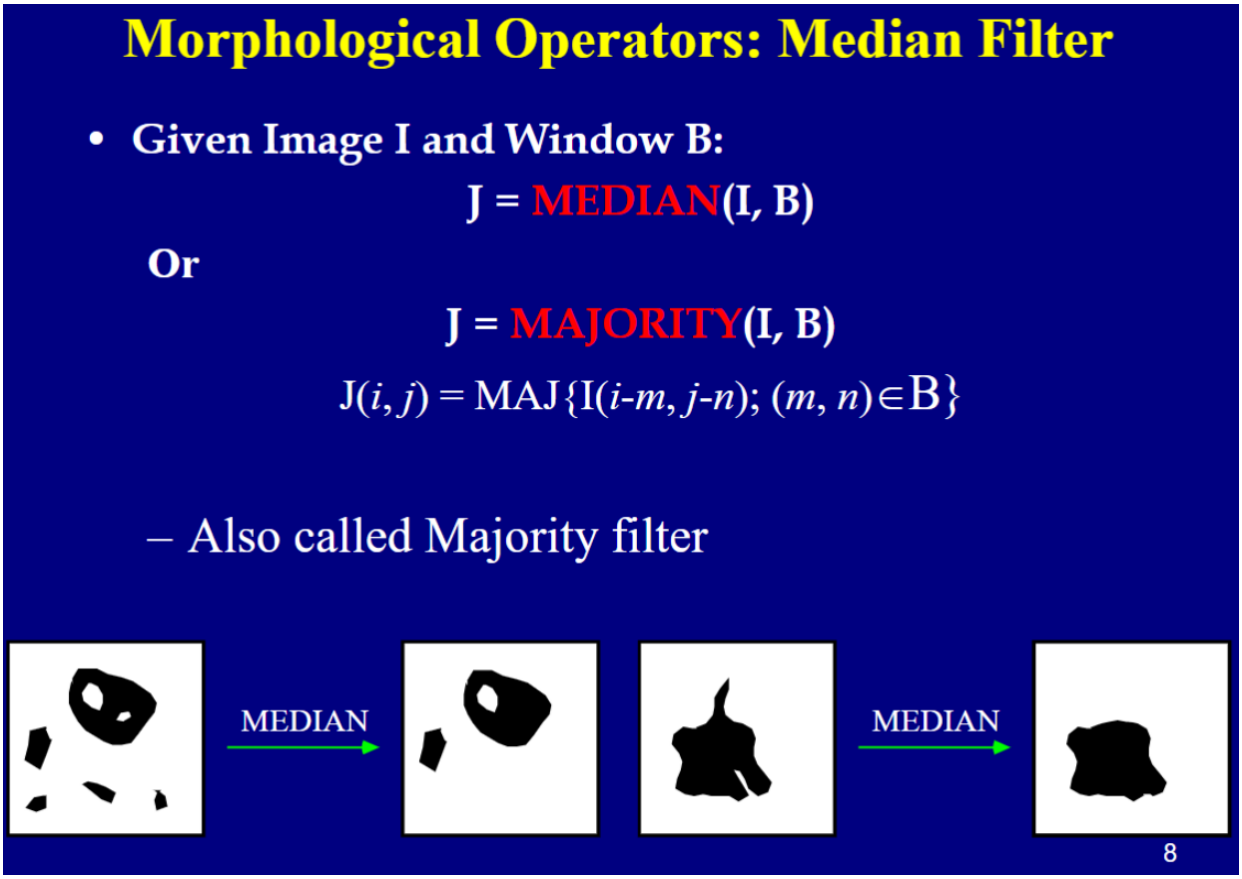


The 3rd image is the final output after applying the dialtion filter. It will basically fill the holes for the object in the image and it will also expand the boundaries of the objects. It will also fill the gaps.

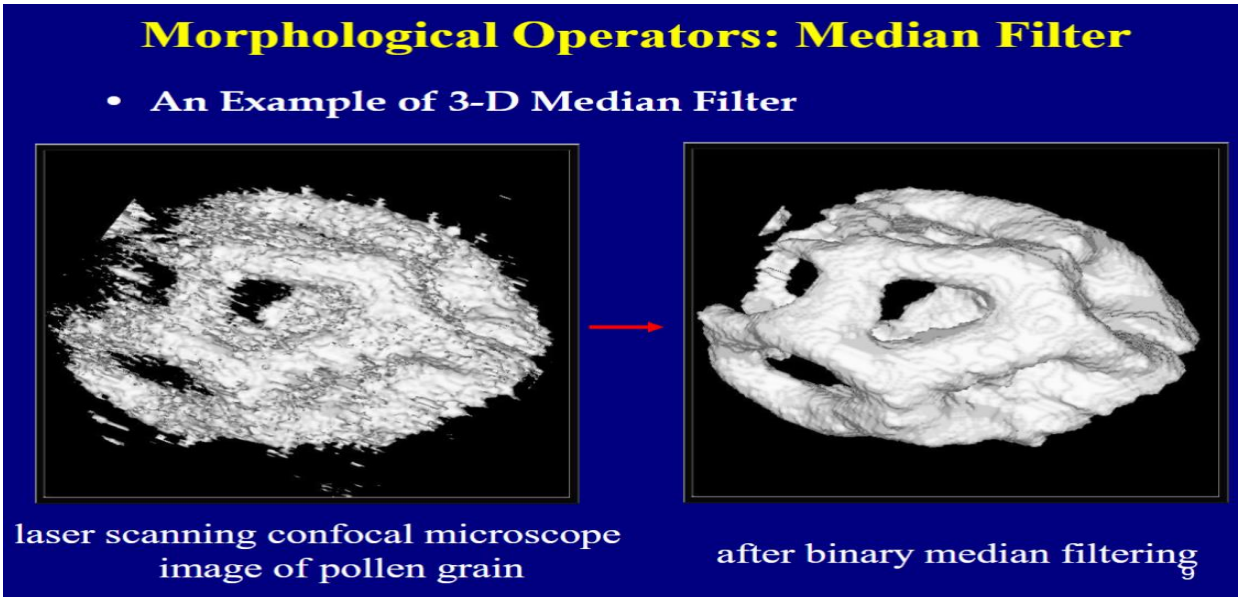
Example of the Erosion Filetr:
It is similar to the Dilation Filter. The only difference is that the logical operation between the pixel is AND operation instead of the OR operation. Een if one of the pixel value is 0 then the whole outcome will be 0. If all the pixels are 1 then the final outcome will be 1. It will make the boundaries smaller.



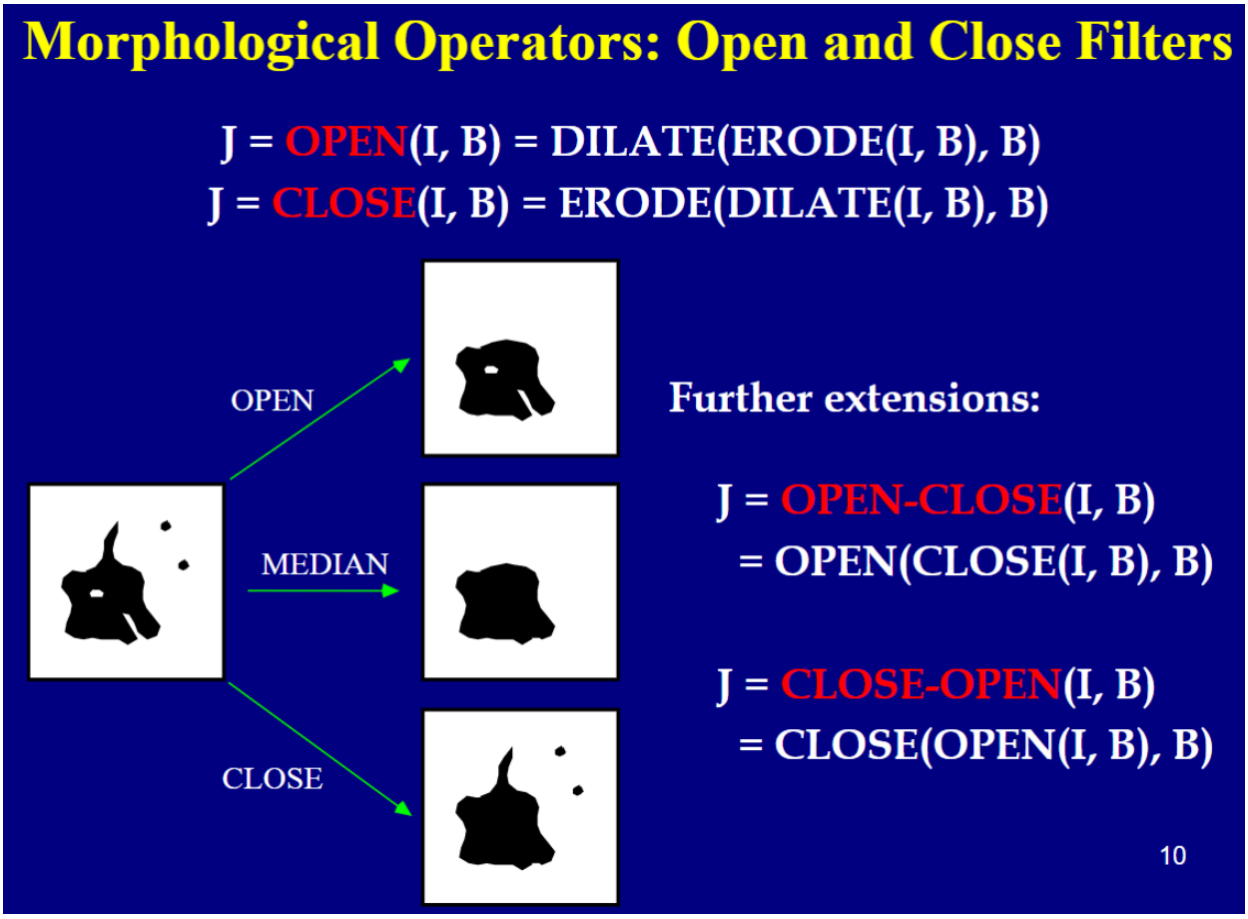
Example of the Median Filter:
When the window is applied to the image I then we will apply the majority operator. If we have more 1's then 0's then we will have the output 1. And Visa Versa. Majority Operation is also known as the Median Operation.

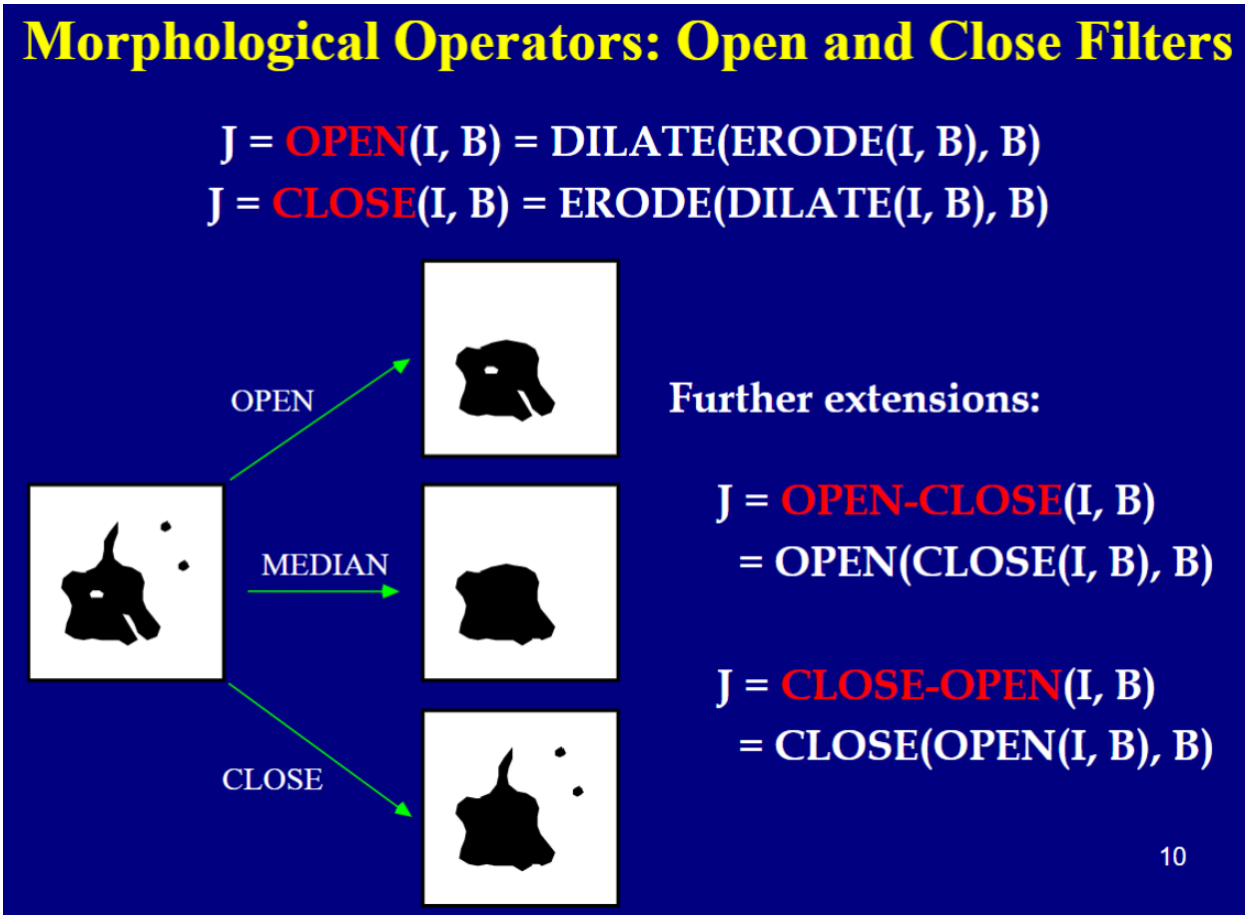


We have 2 ways to define this operator because majority and the median are the same thing.It is also used to smoothen the images.



Open and Close Filters:
OPEN Filter: First we use ERODE filter and then the DILATE filter.
CLOSE Filter: First we use DILATE filter and then the ERODE filter.





Here, the OPEN filter will remove the island and keep the pond. The MEDIAN filter will remove the island and also remove the pond. The CLOSE filter will remove the pond and keep the island. All these effect will be as compared to the original image on the left most side.

Binary Images are the simplest form of images and the most comman way to process this kind of image is to use morphological operations.

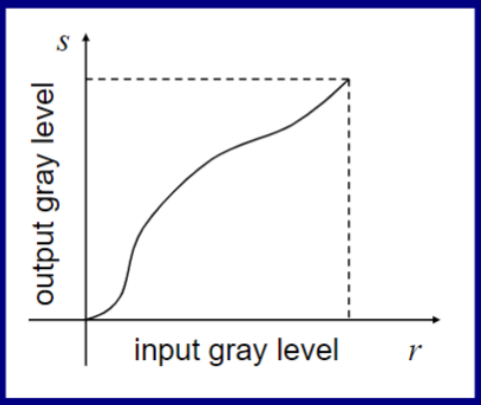
Intensity Transformation for Image Enhancement

Here, the images we use will be the grey-scale images. Intensity Transformation is the simplest transformation we can apply to a grey-scale image. Let supppose we have a 8 bit/pixel value then the range values will be from 0 to $2^8 - 1$ (255). Any number between 0 and 255. Intensity transformation will take one value which is between 0 and 255 and it will map this value to another value which is also between 0 and 255. And this operation will be applied to each individual pixel value. After you apply this to all the pixel then we get a new image and that is called a processed image. Because this operation is applied to each point independently, it is also called **point operation**. It is also called as Zero-memory operation because after the processing of each pixel we can release the memory. So, if we are processing the next pixel then we do not require to remember the previous pixel value. All the Intensity Transformation can be represented as one-dimensional function. This Intensity function is typically monotonically increasing (but it is not always true).

- **Map a given gray level to another level**

$$s = T(r)$$

typically monotonically increasing (but not always)



Here the mapping function T will map the grey-scale value ‘r’ into the value ‘s’.

Intensity Histogram is a very usefull way to measure the effect of the Intensity Transformation.

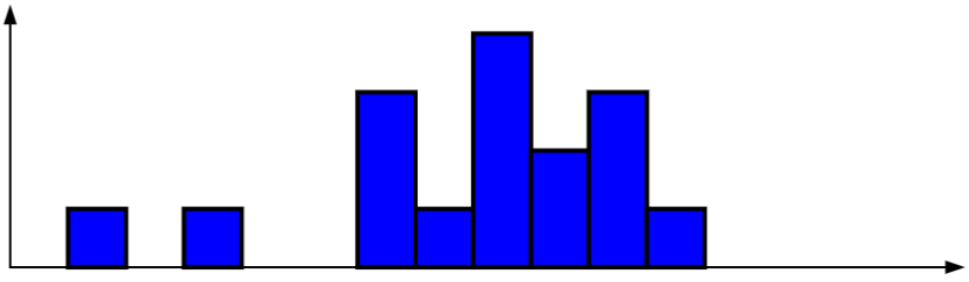
Intensity Histogram

- **Example**

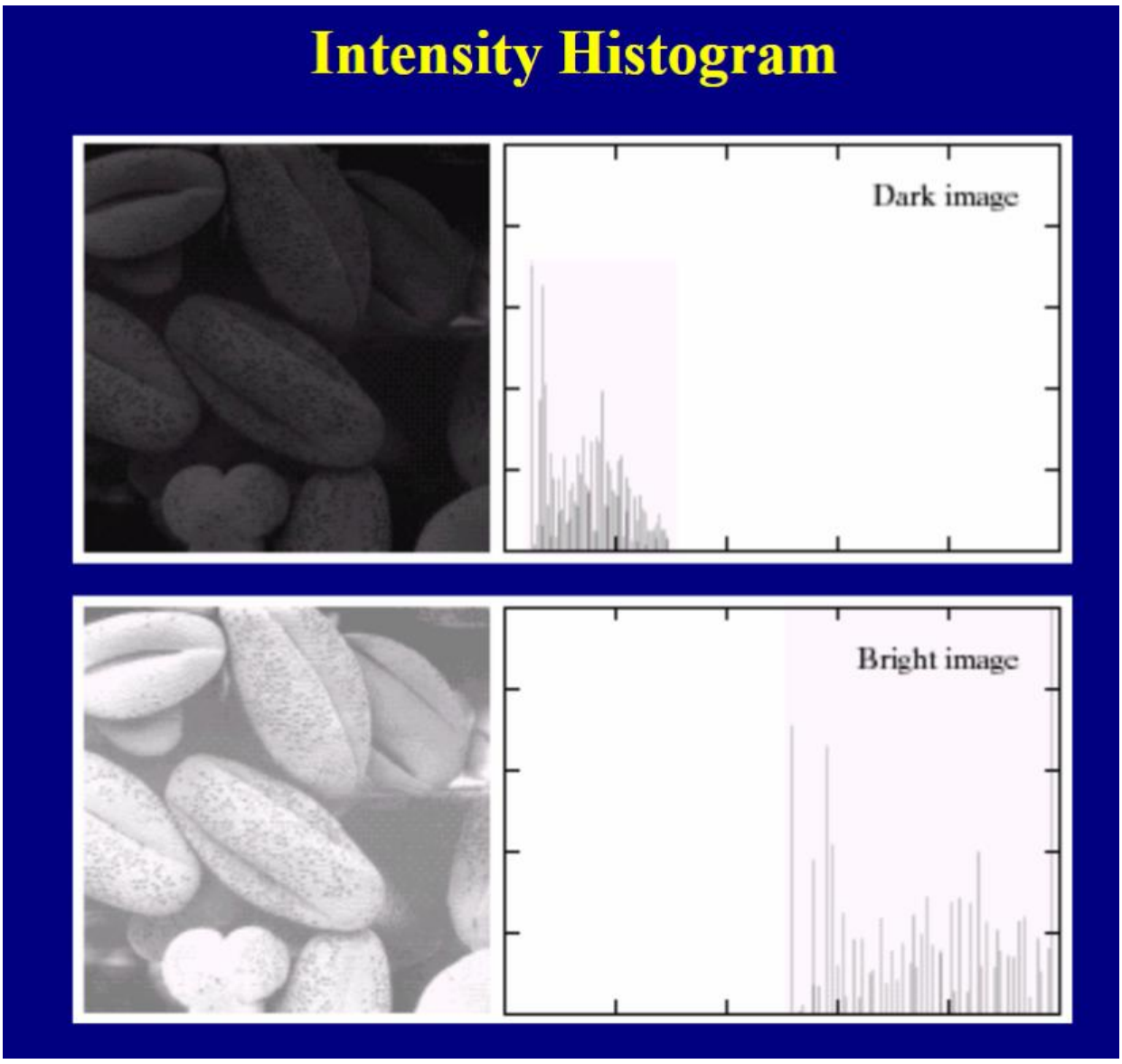
  a 4x4, 4bits/pixel image →

1	8	6	6
6	3	11	8
8	8	9	10
9	10	10	7

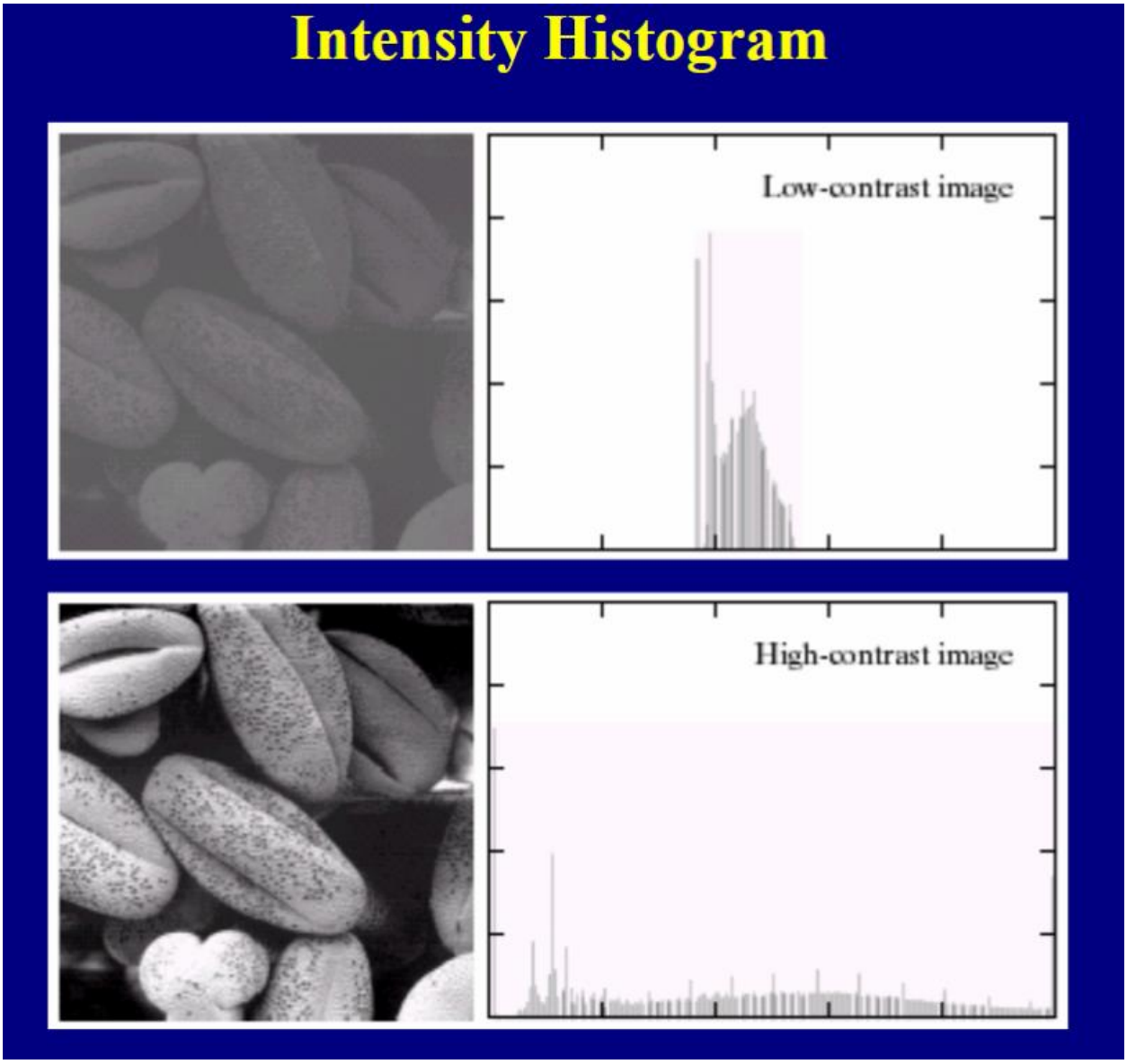
k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H(k)	0	1	0	1	0	0	3	1	4	2	3	1	0	0	0	0



In this above example, we have given a 4x4 image and each pixel in this image is going to be 4 bits. In other words, we can only take the pixel value which is between the range of 0 to $2^4 - 1$ (15). All togather we can have 16 possible distinct values. So, in the above image the value of k varies from 0 to 15. So the value 6 occurs 3 times as shown from the image. The value 8 occurs 4 times. So the histogram for the value 6 has a height of 3. While for the value 8 has a height of 4. It is shown by H(k). Similarly we can create a histogram for a particular image. This is just an example. In real world we have to deal with a lot of data and the histogram size will be much bigger. If we have 8 bits the the value range would be 0 to $2^8 - 1$ (255). We have 256 different possiblr values.



Here, is more example of this Intensity Histogram. They are also 8 bit grey-scale image. So it will also have values between 0 to 255. In the first image, 0 is the black image and 255 represents the white image or very bright image. Most of the histograms are concentrated to the left side or towards 0. So, the first image is pretty dark. While in the second image, most of the histograms are concentarted to right side or towards 255. So, the image apperas to be very bright.



This picture contains the images which are not too dark or not too bright. The forst image looks greyish and also have low contrast. The same thing can be observed from the histogram. Most of the spikes are concentrated in the middle of the graph. While in the second image, the spikes in the histogram are distributed uniformly all over the place. There are significant number of dark values as well as significant number of bright values. So this is a high contrast image.

Basic Transformations

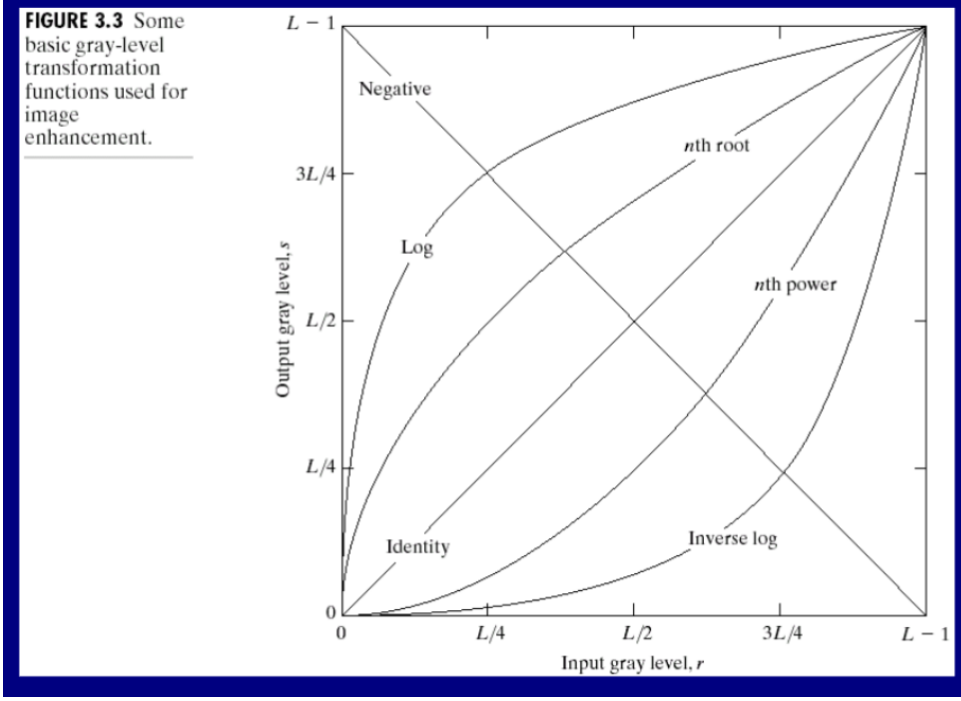


FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.

Negative:

$$s = L - 1 - r$$

Log:

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

Inverse Log:

$$s = e^{cr} - 1$$

Power-law:

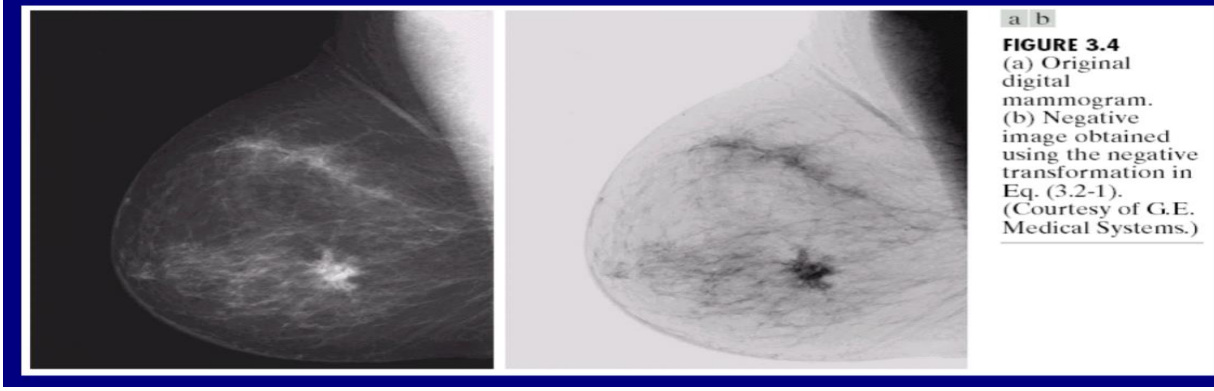
$$s = cr^{\gamma}$$

.....

Here there are few examples of the intensity transformation function. First is the Negative intensity transformation function. In this function the negative value of 'r' will be mapped to 's'. So, lower the value of 'r', the higher the value of 's'. So, the max possible value of 'r' will bw mapped to min possible value of 's' and visa versa. Another we have is the Log mapping. Here, the log is applied to (1 + r) and not directly to 'r' because we have to map 0 = 0 on both the sides. So, log(0) not equals 0. Log(1) = 0.

Negative Transformation

$$s = L - 1 - r$$



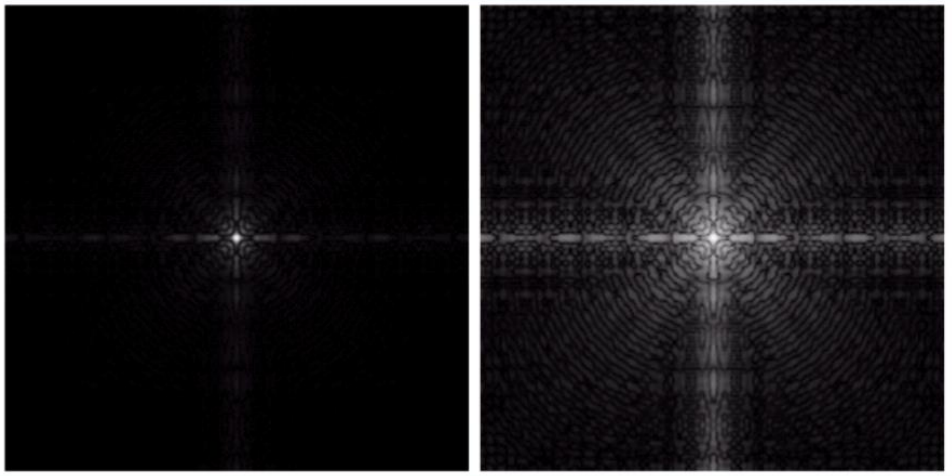
a b

FIGURE 3.4 (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

The above image shows the before and after image of the negative intensity transformation.

Log Transformation

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



a b

FIGURE 3.5 (a) Fourier spectrum. (b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.

This is the example of LOG transformation. It has a very practical use. If you apply 2D discrete fourior transform then you will most likely to get an image on the left side. The second image is after applying the log transformation.

Power-law (Gamma) Transformation

$$s = cr^{\gamma}$$

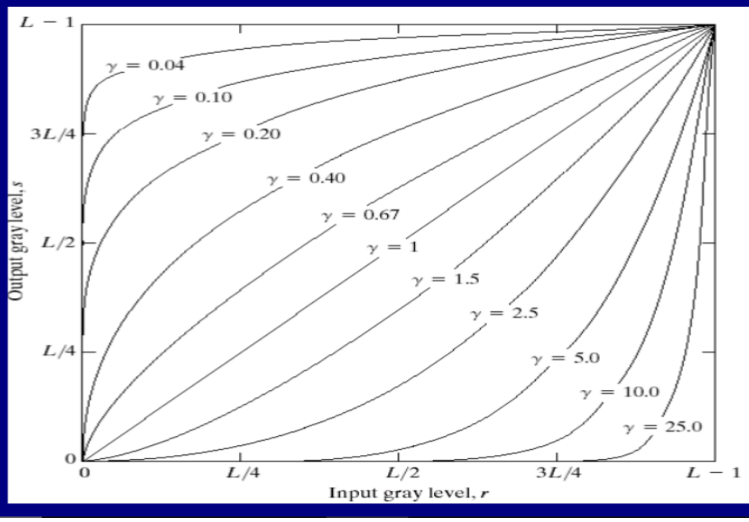
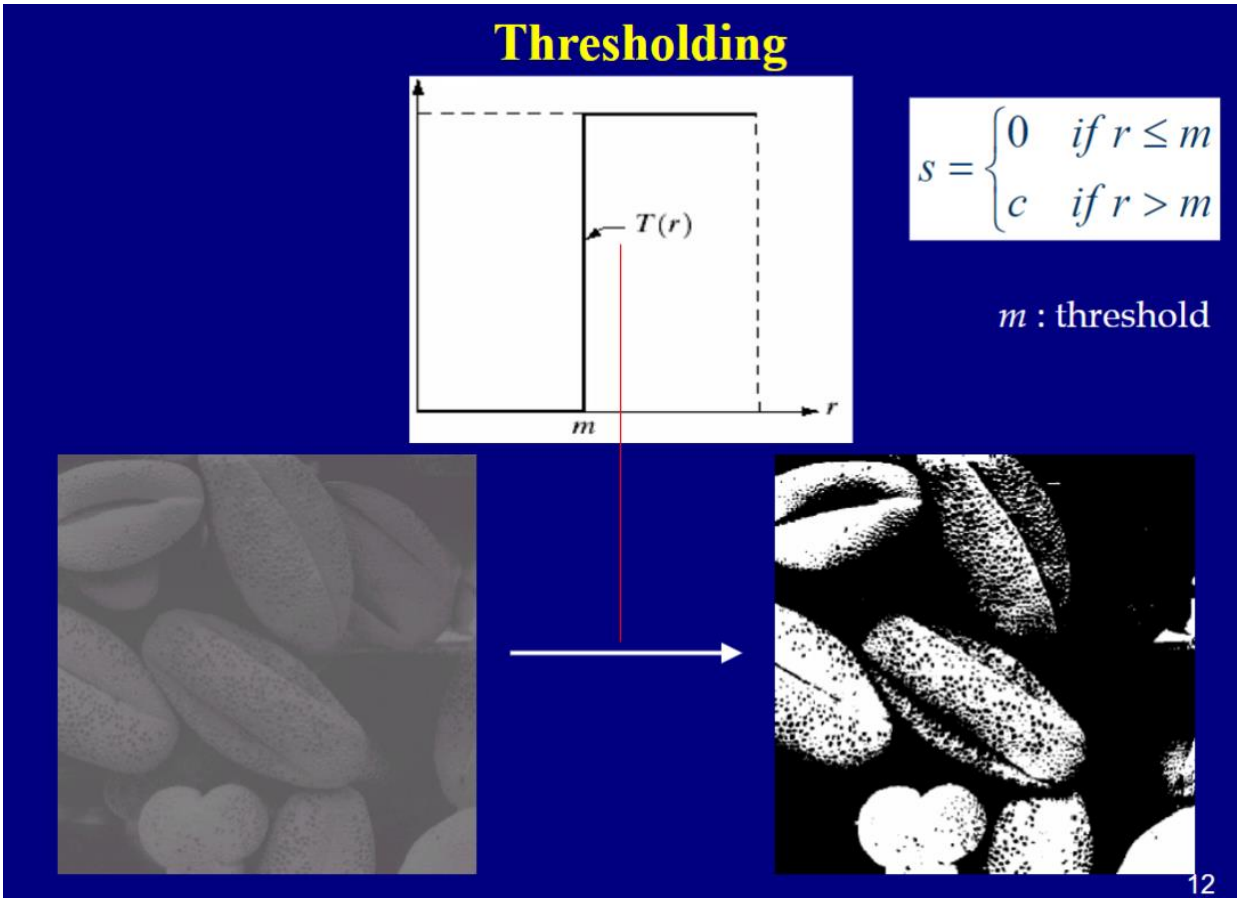


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Power-Law transform is also sometimes called as Gamma transform. Here, for different gamma values we are getting different curves as shown. When gamma = 1 there is no transformation. When gamma > 1, then we get a convex curve. We get much higher contrast and high intensity values. When gamma < 1 then we get a concave curve. We get low contrast and low intensity values.

Thresholding is a special case of Intensity Transformation. In thresholding the mapping function is the step function as shown.



The step function is 0 if r <= m else it has some value c. After applying thresholding to a grey-scale image we will get a binary image which can be seen from the above example.

Example: Fixed Intensity Transformation

- A 4x4, 4bits/pixel image

1	8	6	6
6	3	11	8
8	8	9	10
9	10	10	7

passes through an intensity transformation

$$s = T(r) = round\left(\frac{1}{15}r^2\right)$$

1 → round(0.0667) = 0;
3 → round(0.6) = 1;
6 → round(2.4) = 2;
7 → round(3.2667) = 3;
8 → round(4.2667) = 4;
9 → round(5.4) =5;
10 → round(6.6667) = 7;
11 → round(8.0667) = 8;

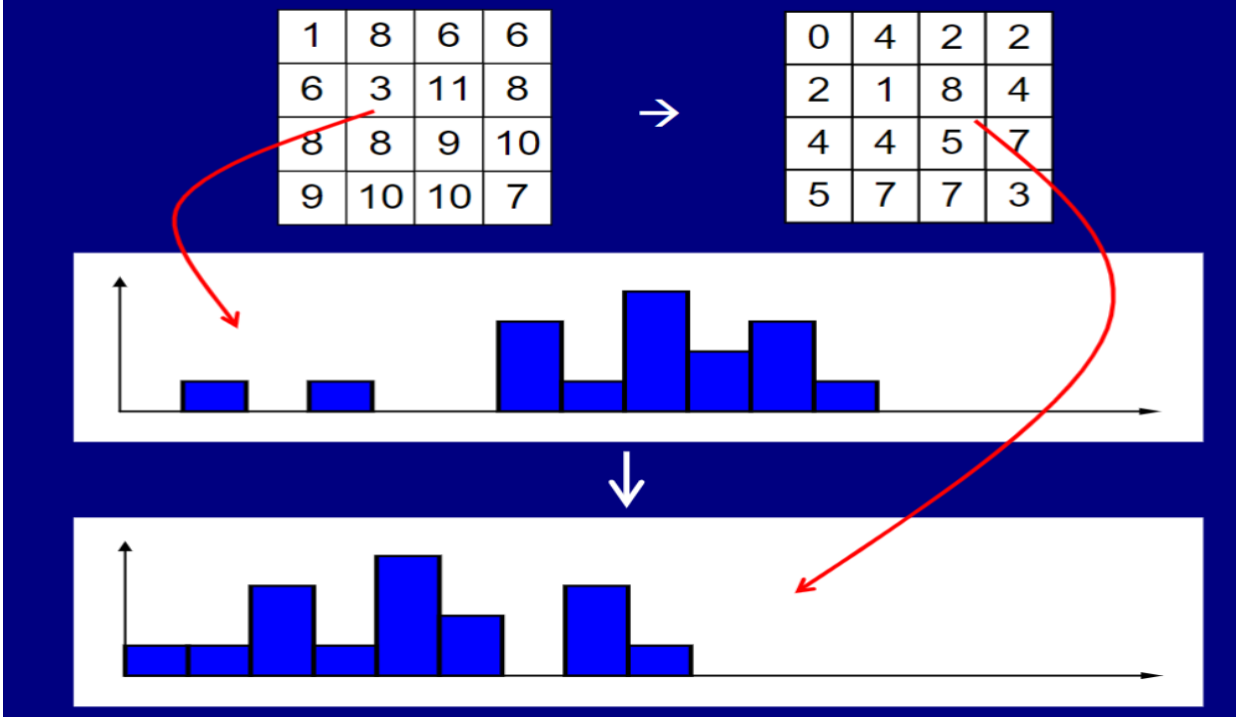
The resulting image is:

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

13

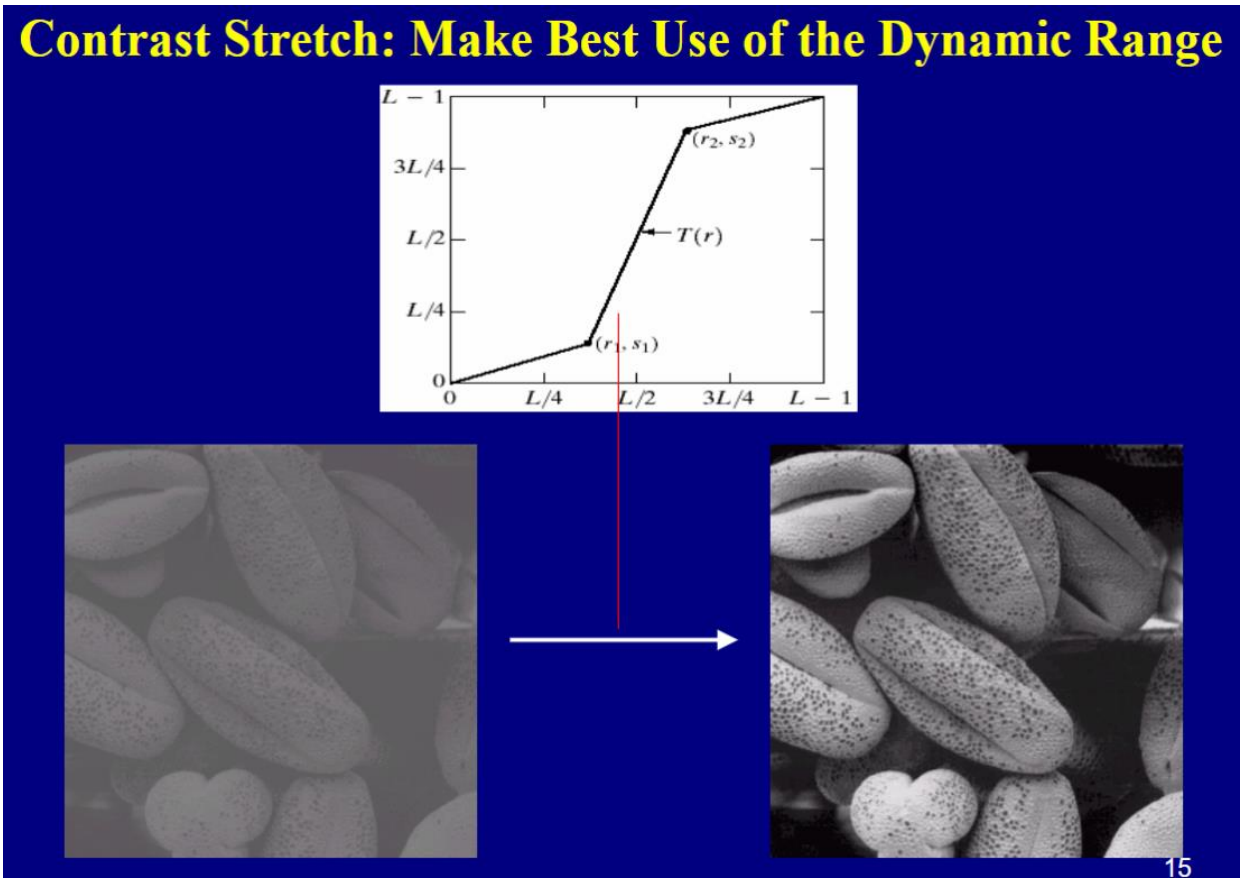
Suppose, we have given a 4x4, 4 bits/pixel image and we have also given the Intensity Transformation function as shown. So, here all we need to do is to look at the pixel values in our input image. The pixel values is the 'r' values. So, we will take each pixel value and compute the 's' value for each pixel as shown in the above image.

Example: Histogram Change



The Histogram representation is also shown. The Intensity Transformation is pushing the values to the low end as shown in th ebefore and after histogram.

The most comman use of the Intensity Transformation is to enhance the contrast of an image. So we are given a low contrast image and we have to output an image with high conatrst. It is called cnotrast stretch. We have to make the best possible use of the dynamic range.



Contrast Stretch

General form:

$$s = \begin{cases} \frac{s_1}{r_1} \cdot r & 0 \le r < r_1 \\ \frac{s_2 - s_1}{r_2 - r_1} \cdot r + \frac{s_1 r_2 - s_2 r_1}{r_2 - r_1} & r_1 \le r \le r_2 \\ \frac{2^B - 1 - s_2}{2^B - 1 - r_2} \cdot r + (2^B - 1) \cdot \frac{s_2 - r_2}{2^B - 1 - r_2} & r_2 < r \le 2^B - 1 \end{cases}$$

Special case → Full-scale contrast stretch:

$r_1 = r_{\min}$ $\quad s_1 = 0$

$r_2 = r_{\max}$ $\quad s_2 = 2^B - 1$

$$\longrightarrow \quad s = (2^B - 1) \cdot \frac{r - r_{\min}}{r_{\max} - r_{\min}}$$

Typically used: $$s = round\left((2^B - 1) \cdot \frac{r - r_{\min}}{r_{\max} - r_{\min}} \right)$$

Consider the following example and we have to do a full scale contrast stretch.

Example: Full-Scale Contrast Stretch

- Full-scale contrast stretch of a 4x4, 4bits/pixel image

4	8	6	6
6	4	11	8
8	8	9	10
8	11	10	7

- Find $r_{\min} = 4$ $\quad r_{\max} = 11$ $\quad 2^B - 1 = 15$

$$s = round\left((2^B - 1) \cdot \frac{r - r_{\min}}{r_{\max} - r_{\min}} \right) = round\left(15 \cdot \frac{r - 4}{11 - 4} \right) = round\left(\frac{15}{7}(r - 4) \right)$$

4 → round(0) = 0;
6 → round(4.29) = 4;
7 → round(6.43) = 6;
8 → round(8.57) = 9;
9 → round(10.71) = 11;
10 → round(12.86) = 13;
11 → round(15) = 15;

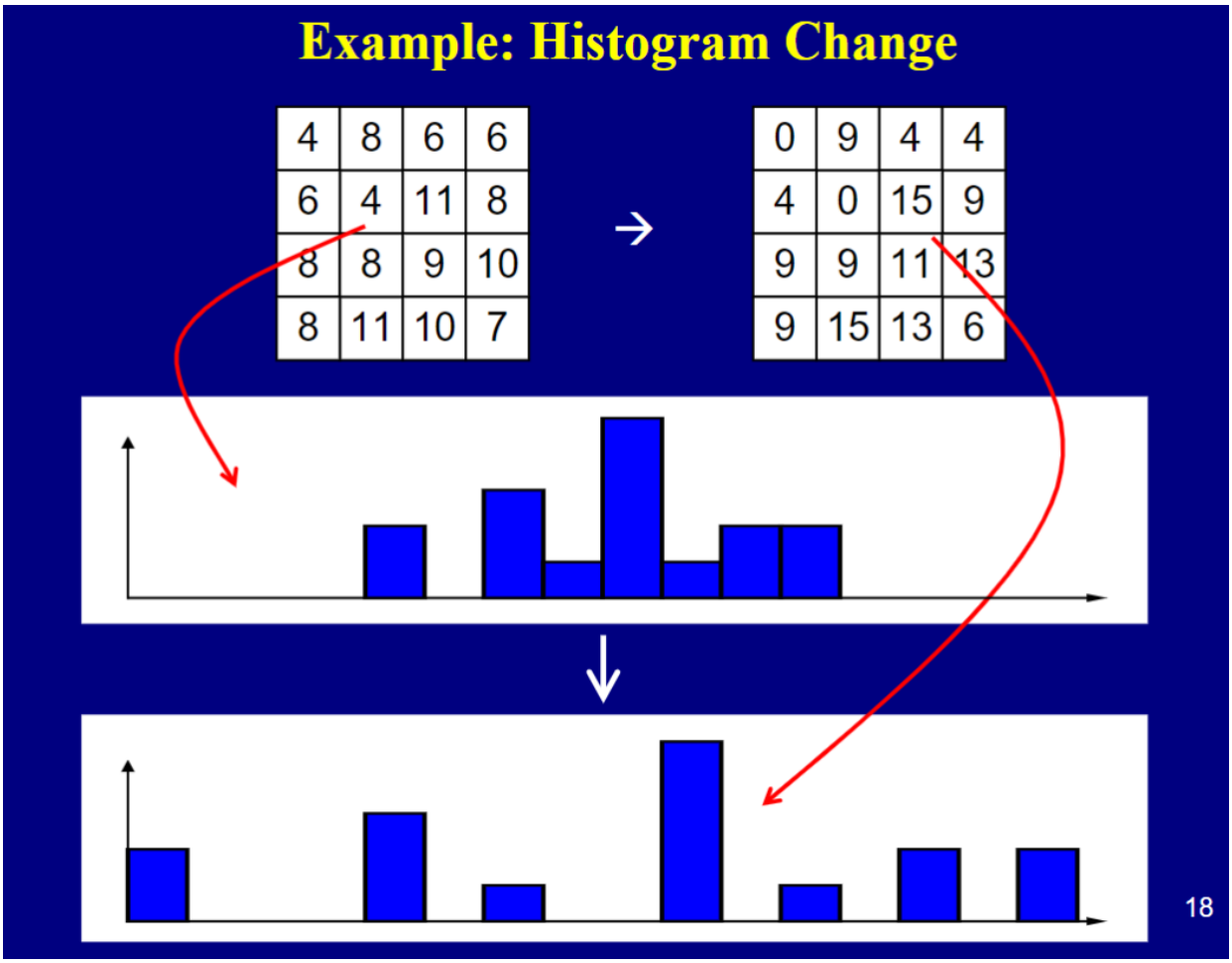
The resulting image is:

0	9	4	4
4	0	15	9
9	9	11	13
9	15	13	6

17

First thing we are going to do is to find the minimum value and the maximum value form the matrix of the image that is given to us. So the min value is 4 and the max value is 11. We know $2^4 - 1 = 15$ that is

the fixed number. So from the previous slide we take the formula of ‘s’ and apply it since we have the values of ‘r’. we will apply this formula for each pixel and we will do it for all the whole image. So we now have our resulting image.

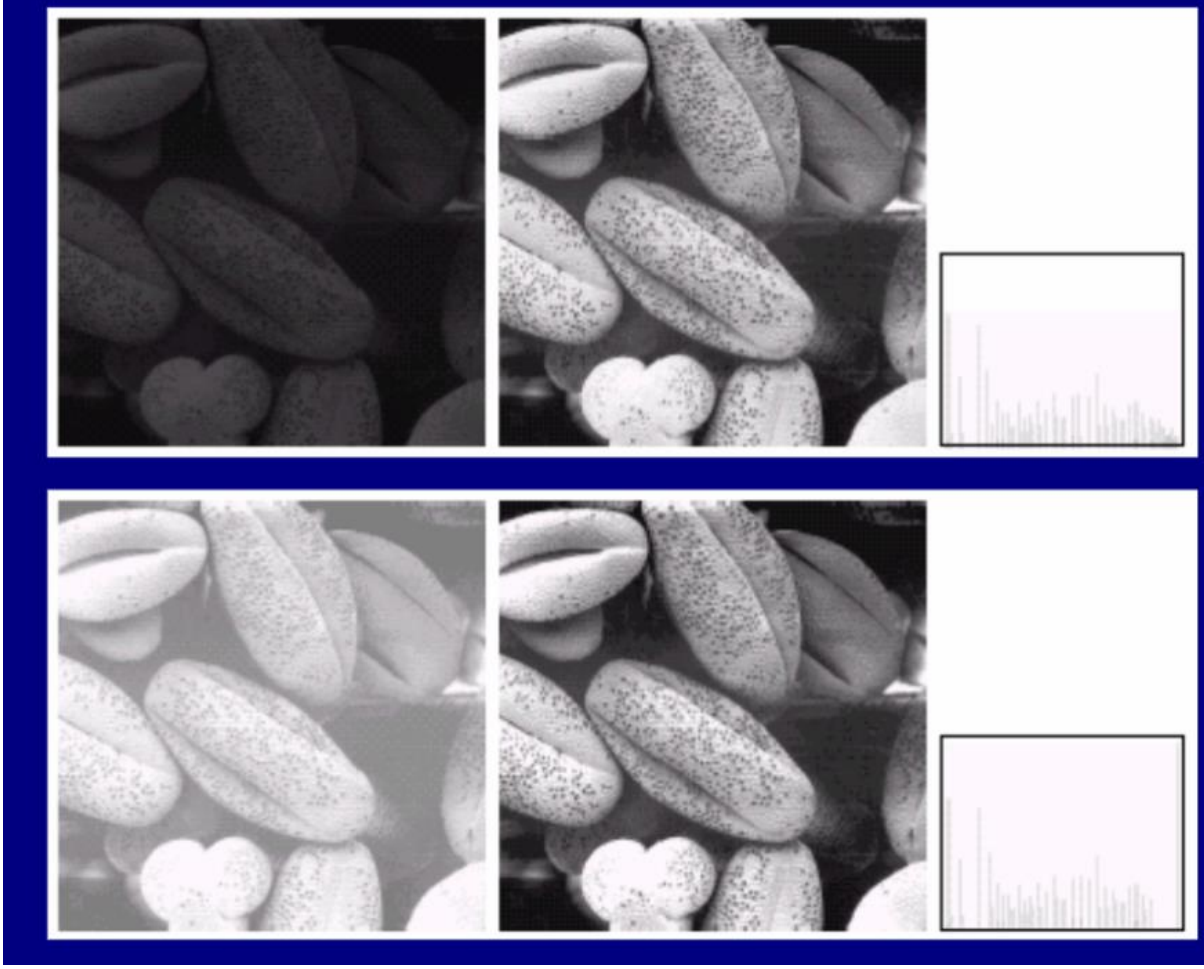


This is the before and after Histogram representation. It shows the effect of the intensity transformation. After full scale contrast stretch the min value is garenteed to hit 0 and the max value is gaureenteed to hit 255 as shown. So overall the histogram will be uniformly distributed. The new image will have a high contrast.

Histogram Equalization

The main idea of the histogram equalization is that we can have the histogram as flat as possible. The important thing with the histogram equalization is that no matter what input we have whether we have a high contrast or a low contrast image, the output will always be the same. So the input image may be different but the output image will always be similar.

Histogram Equalization



Example

- A 4x4, 4bits/pixel image

2	8	9	9
2	3	10	9
8	3	3	11
8	3	10	11

- First try: full-scale contrast stretch $r_{\min} = 2$ $r_{\max} = 11$

$$s = round\left((2^B - 1) \cdot \frac{r - r_{\min}}{r_{\max} - r_{\min}} \right) = round\left(15 \cdot \frac{r-2}{11-2} \right) = round\left(\frac{5}{3}(r-2) \right)$$

2 → round(0) = 0;
3 → round(1.67) = 2;
8 → round(10.00) = 10;
9 → round(11.67) = 12;
10 → round(13.33) = 13;
11 → round(15) = 15;

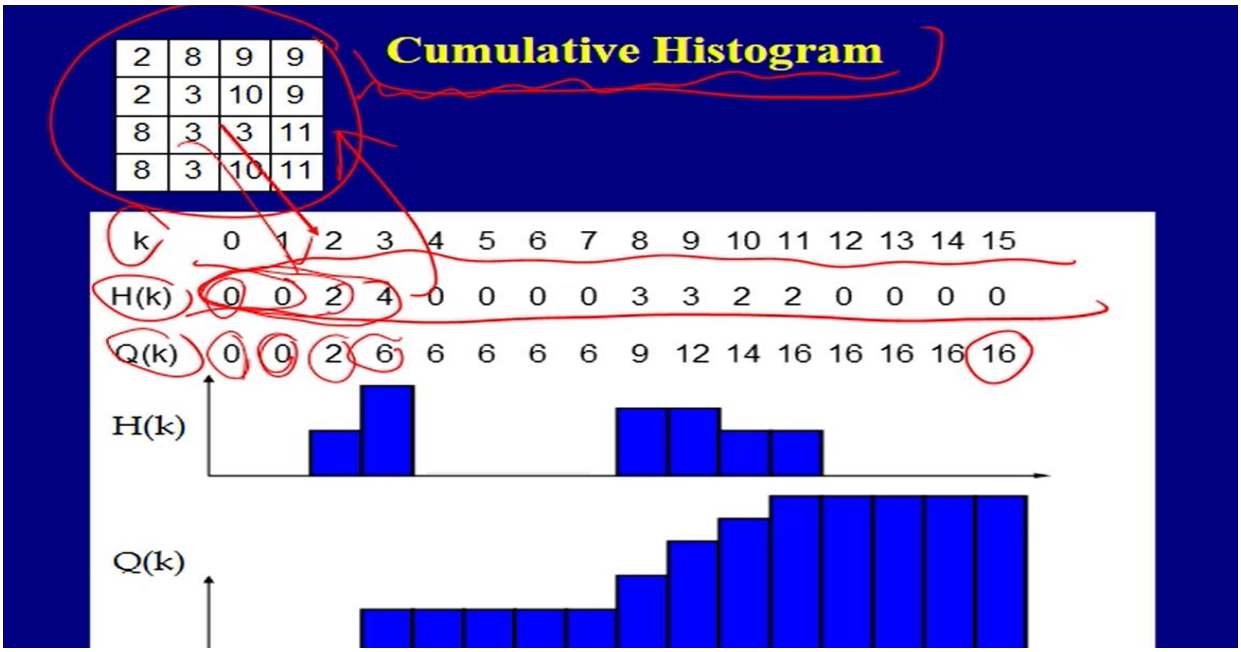
The resulting image is:

0	10	12	12
0	2	13	12
10	2	2	15
10	2	13	15

21

We have given the input image. So we will find out the min value and the max value. Min value is 2 and the max value is 11. We will apply to the same formula and also to each and every pixel so that we can get a new processed image.

Cumulative Histogram is the specific case of the Histogram Equalization. Here we will divide the histogram equalization into several steps.



We know how to get k, H(k). If we want to obtain Q(k) then we simply need to add the previous values of the H(k) as shown. After we have created a cumulative histogram then we will have to create an intermediate image. The intermediate image can be converted from an original image. See below example you will understand it.

Intermediate Image

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
H(k)	0	0	2	4	0	0	0	0	3	3	2	2	0	0	0	0
Q(k)	0	0	2	6	6	6	6	6	9	12	14	16	16	16	16	16

original

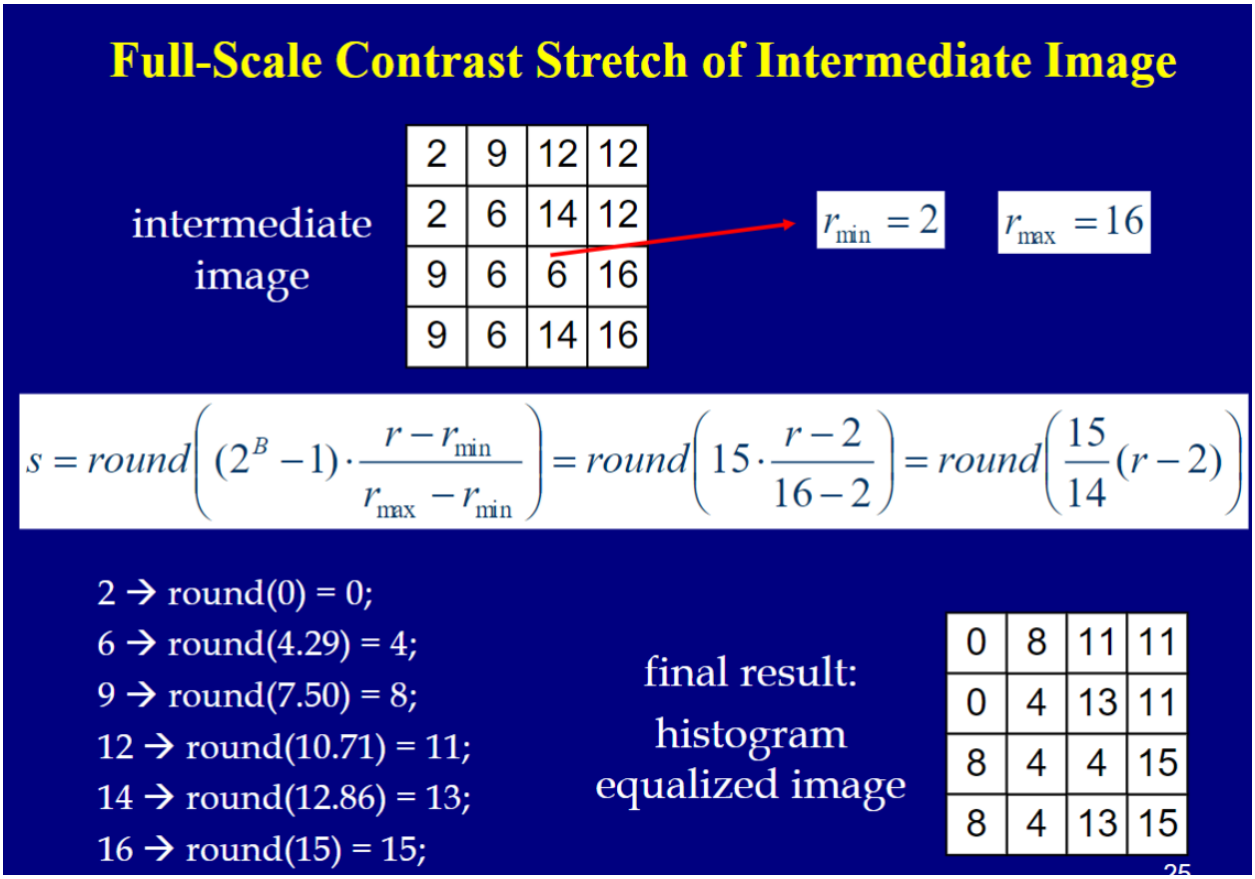
2	8	9	9
2	3	10	9
8	3	3	11
8	3	10	11

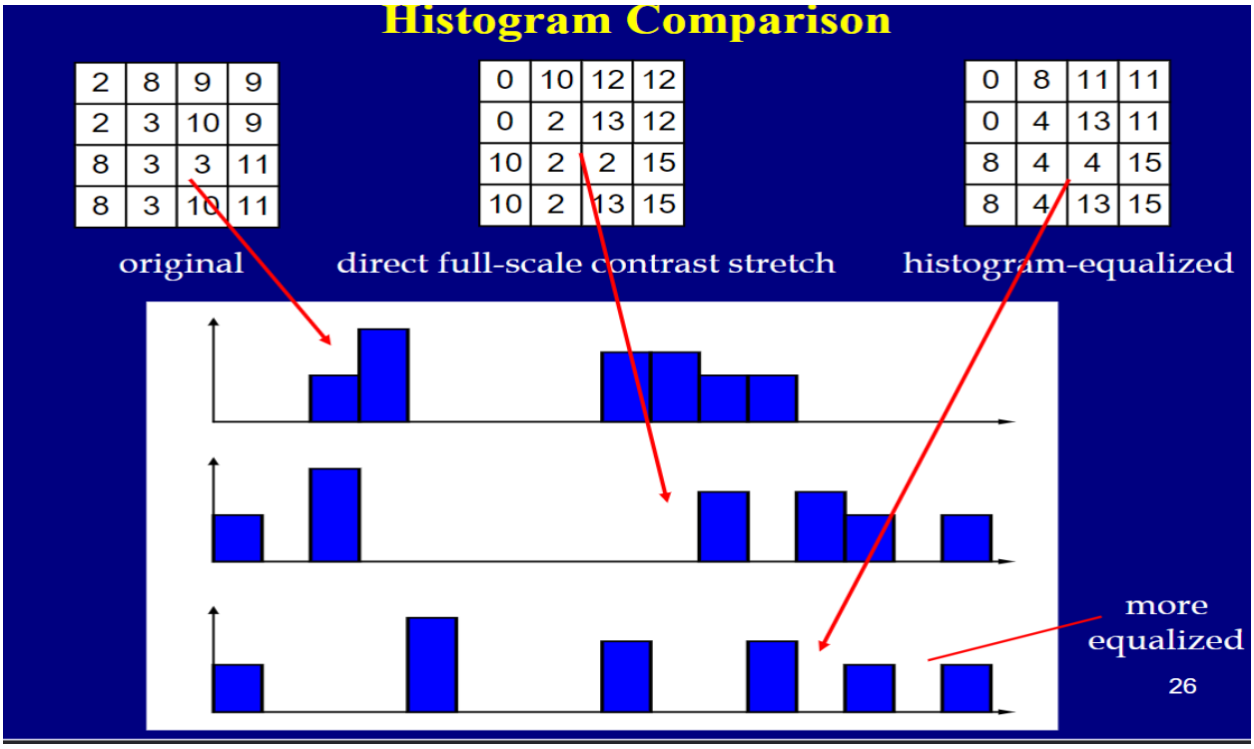
→

2	9	12	12
2	6	14	12
9	6	6	16
9	6	14	16

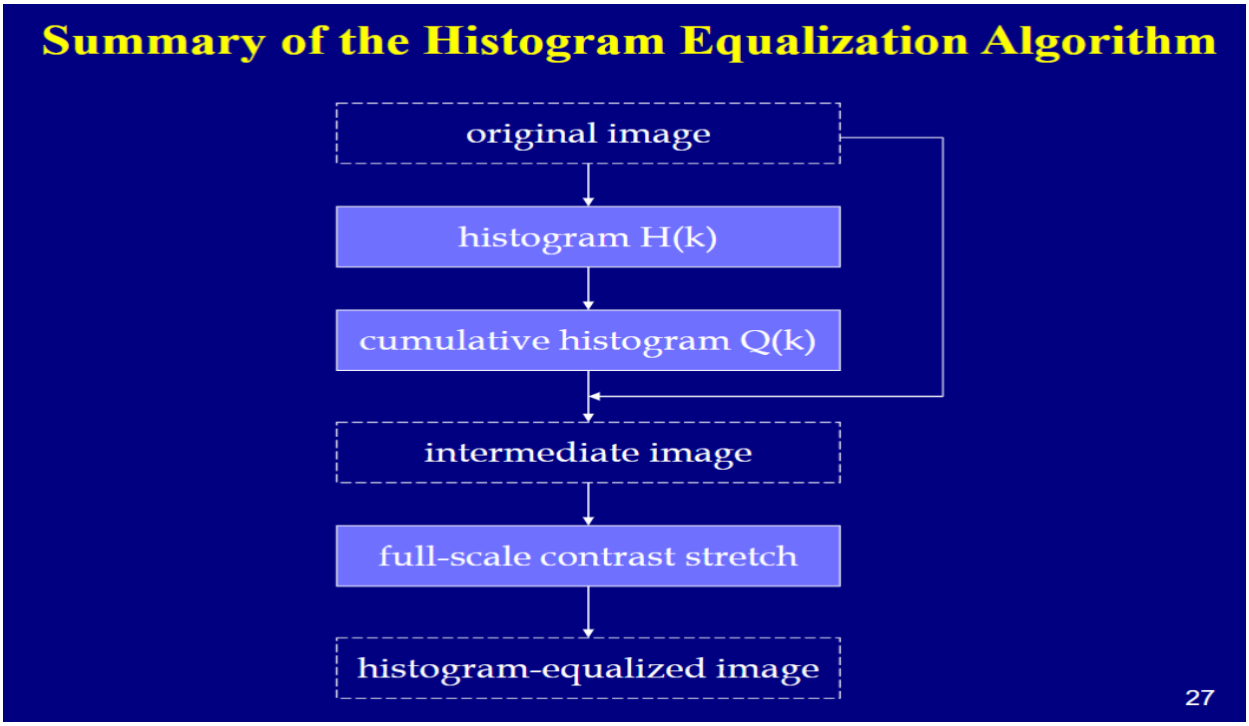
intermediate image

Consider the k value of the original image and consider the Q(k) value corresponding to the k value and construct the intermediate image. After obtaining the intermediate image, we can have a full-scale contrast stretch image as done previously.



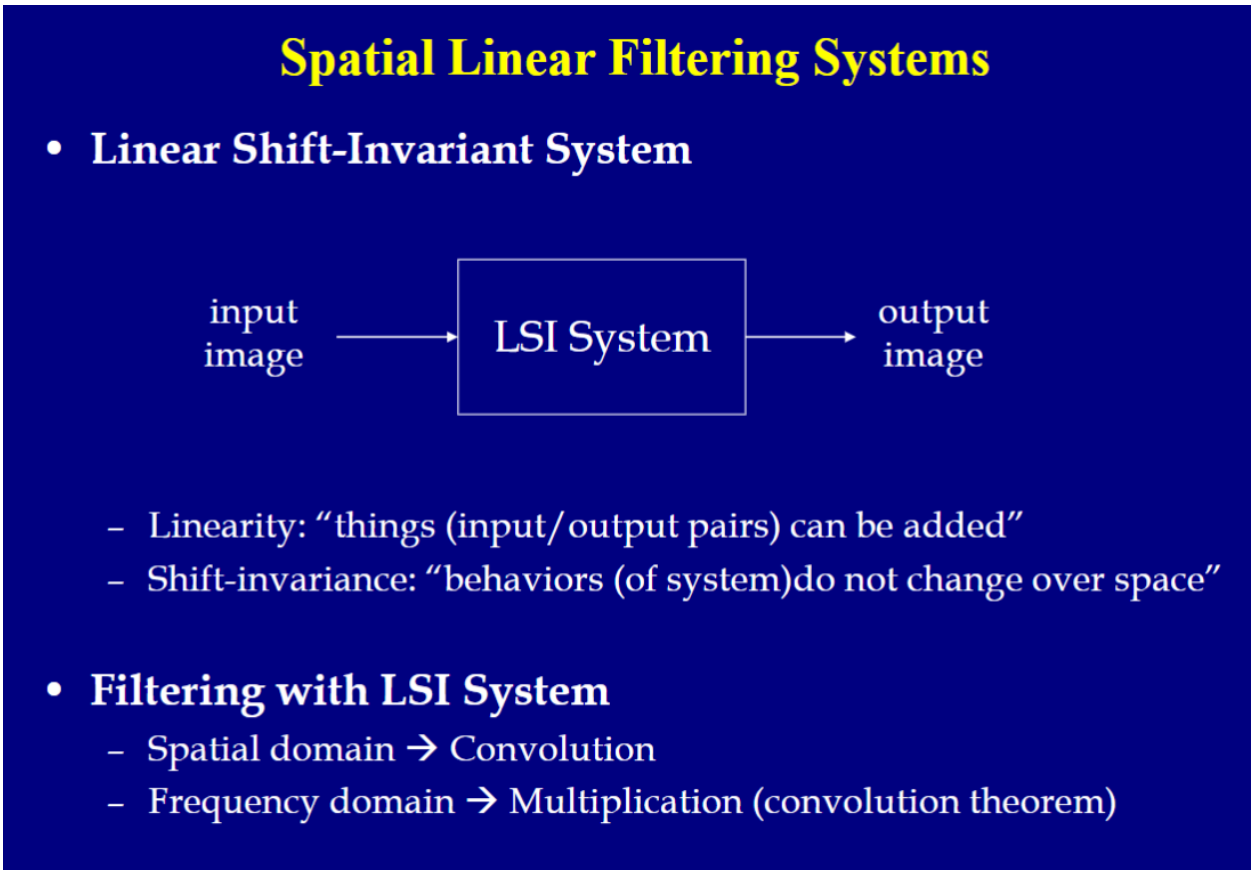


The first histogram is of the original image. It has low contrast. The second histogram is for the direct full-scale contrast stretch. But the problem is that we get a big gap in the histogram that means that we are not using the full dynamic range. The third image has more evenly spread histogram as shown.



Spatial Linear Filtering of Images

To understand about Linear Filtering we first need to be clear about Linear shift-Invariant system, In general, a system is a blackbox that is taking some input image and ouput some processsed image. LSI (Linear Shift-Invariant System) is a specific type of system that follows two additional properties. A] Linearity: input/output pairs can be added. Additive input produces the additive output. B] Shift-invariance: behaviours of system do not change over space.



Impulse Response means that if you put delta function or an impulse as an input to an LSI system then your output is going to be an impulse response.An LSI system can be fully characterised by an impulse system. Given the impulse response of an LSI system, togather with the input to the system, the output can be uniquely determined.

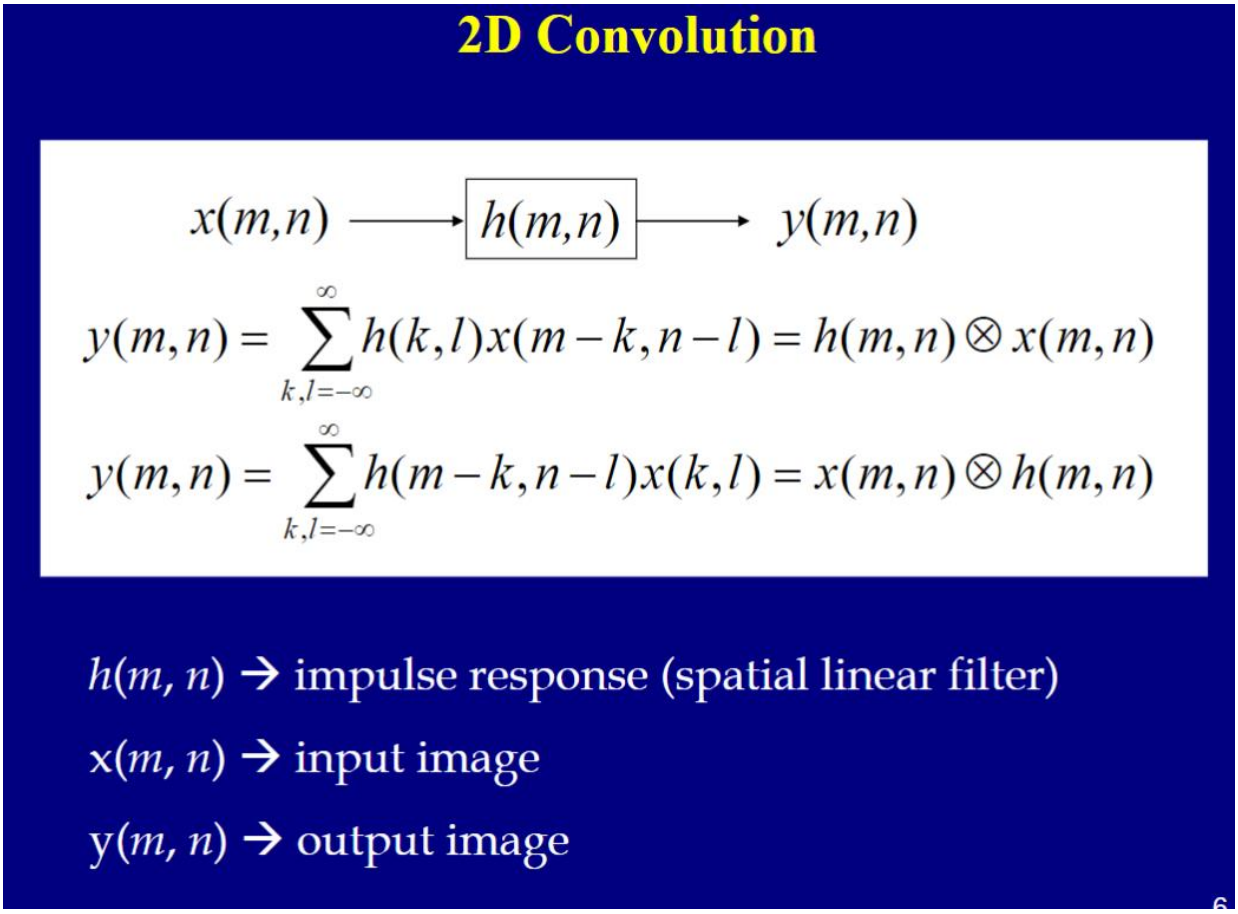
Spatial Neighbourhood

When we work on spatial domain linear image processsing, the first thing we need to define is the Spatial Neighbourhood. It is typically defined in a local wnidow. And this local window can be of any shape (square, rectangle, diamond, etc…) When we apply linear filtering then we will take this widow and slide this whole window to each and every pixel of the input image. And at each step, we will perform some operation on that particualr pixel and then we move forward. We will repeate this for the whole image.

Once we define the windows for the linear filter, we will now define the values to be kept in the window. These values are called the coefficients. These values are the pixel values of the window. They can also be refered to as filter coefficients. When we slide the window to the image then we will simply do the inner product of the window coefficients and the image pixel values. And then we will add them all

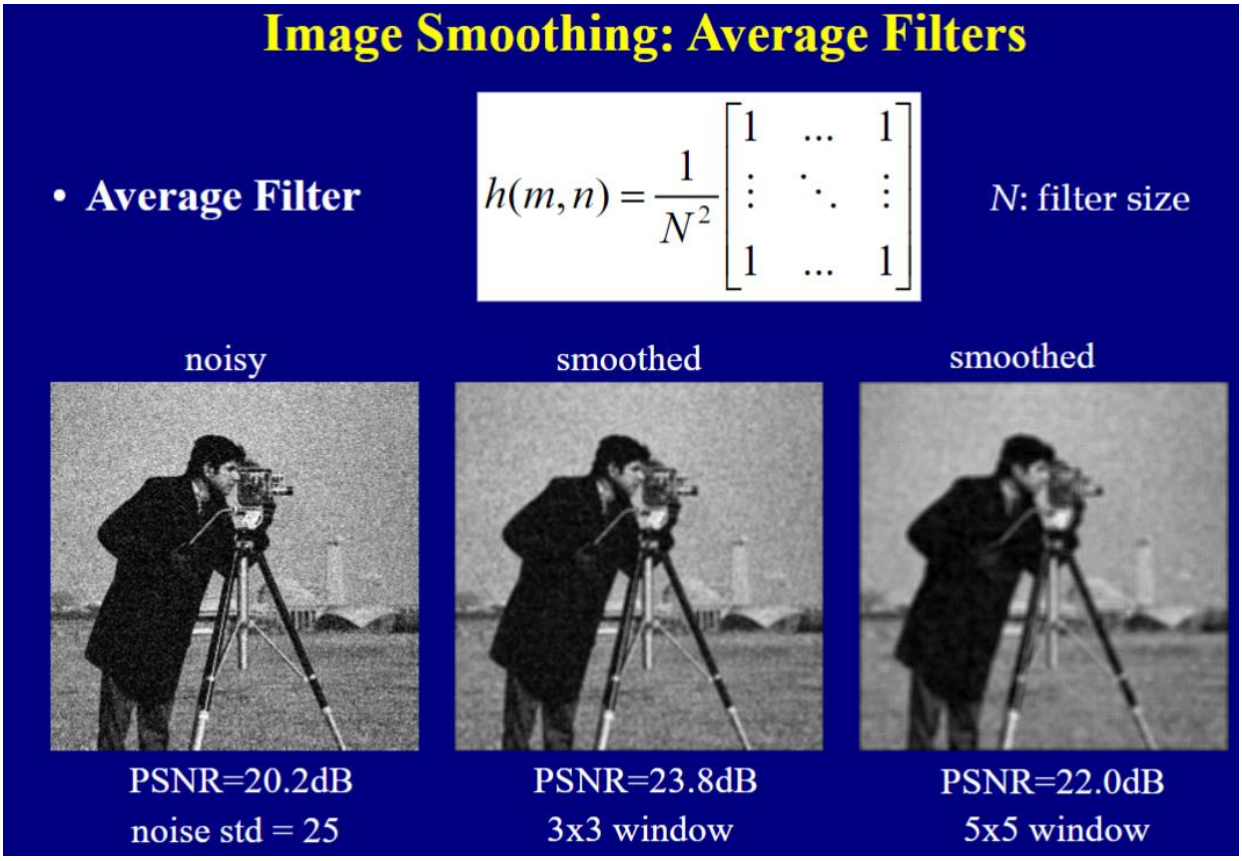
togather so that it will produce some number and that number will be the final pixel value of the processed image. So, this particular window has many different names such as **Window**, **Mask**, **Filter**, **Impulse Response**. Symetri filter have some of the values same like the diagonal value, opposite end value, etc…

The linear filtering process can be expressed using convolution. So basically we can say that the output y is the convolution of the input x and the impulse response h.

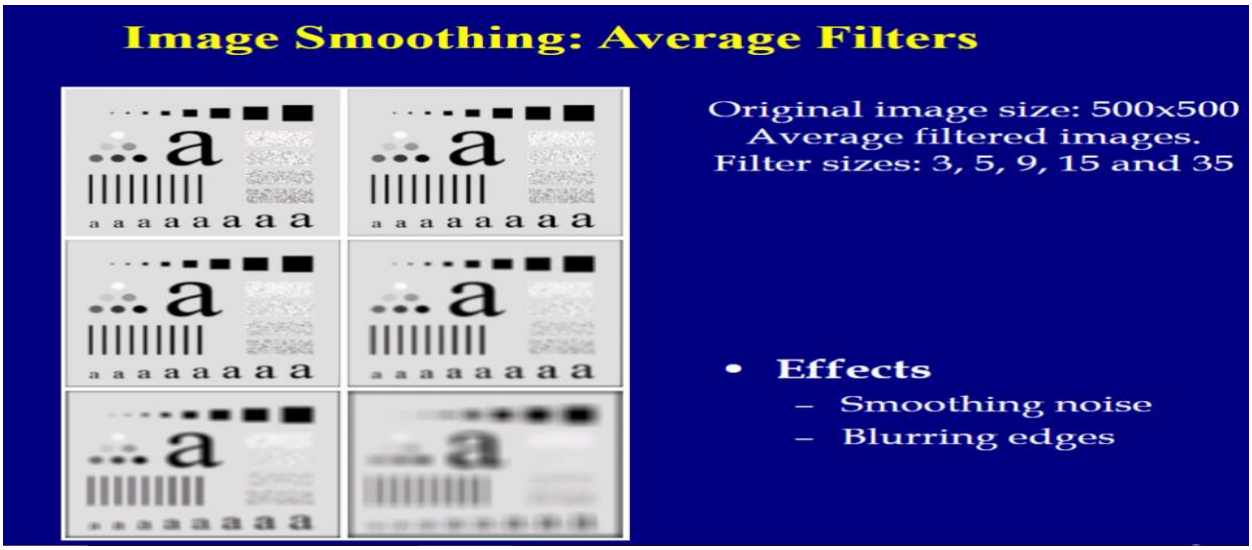


Here, h convolve with x is the same as the x convolve with h.
Spatial Linear filtering has many many applications such as image smoothing, image enhancement (make an image sharper), image restoration (image denoising, image deblurring), edge detection, filter bank (image transformation, frequency analysis).

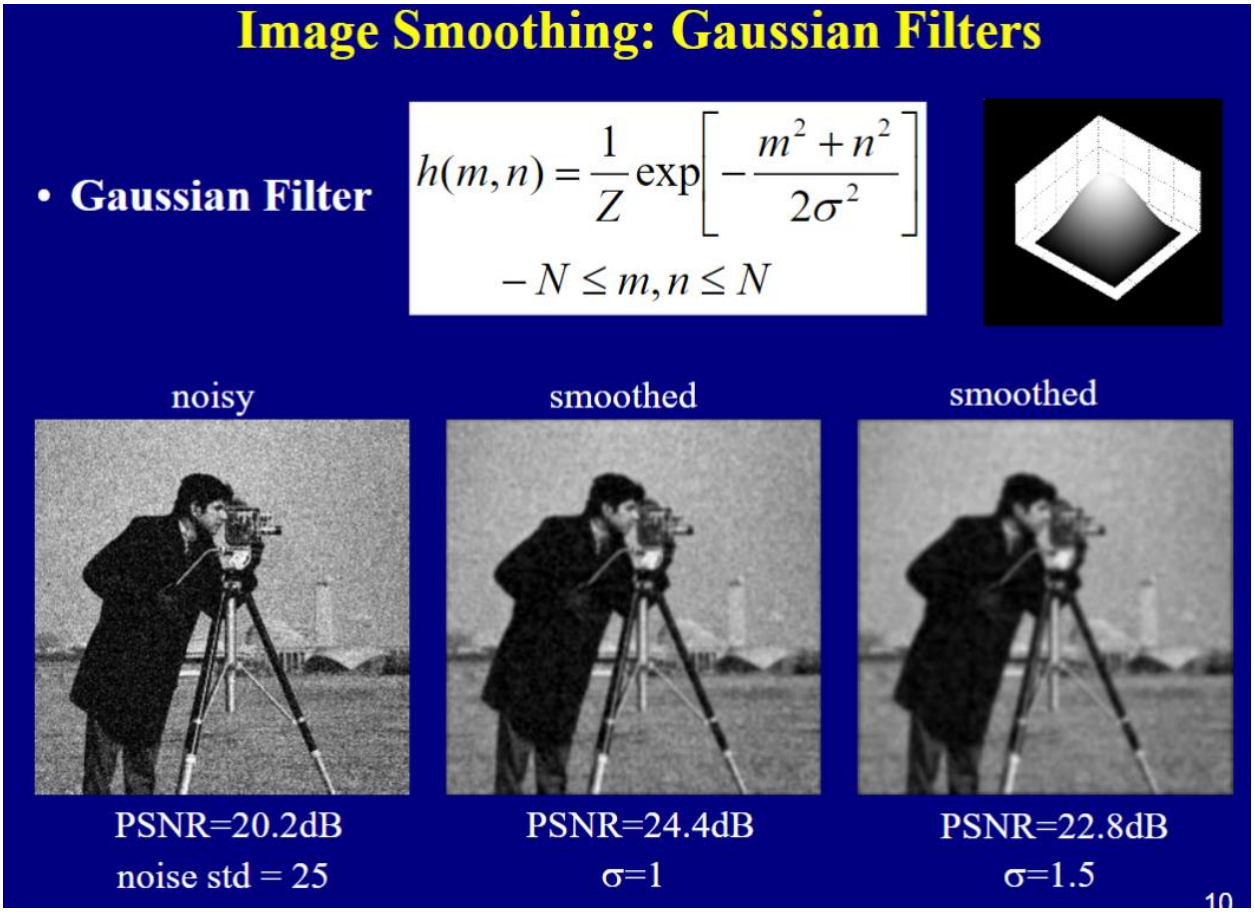
Image Smoothing
We will take a filter h as shown in the image below. It will be a NxN filter.



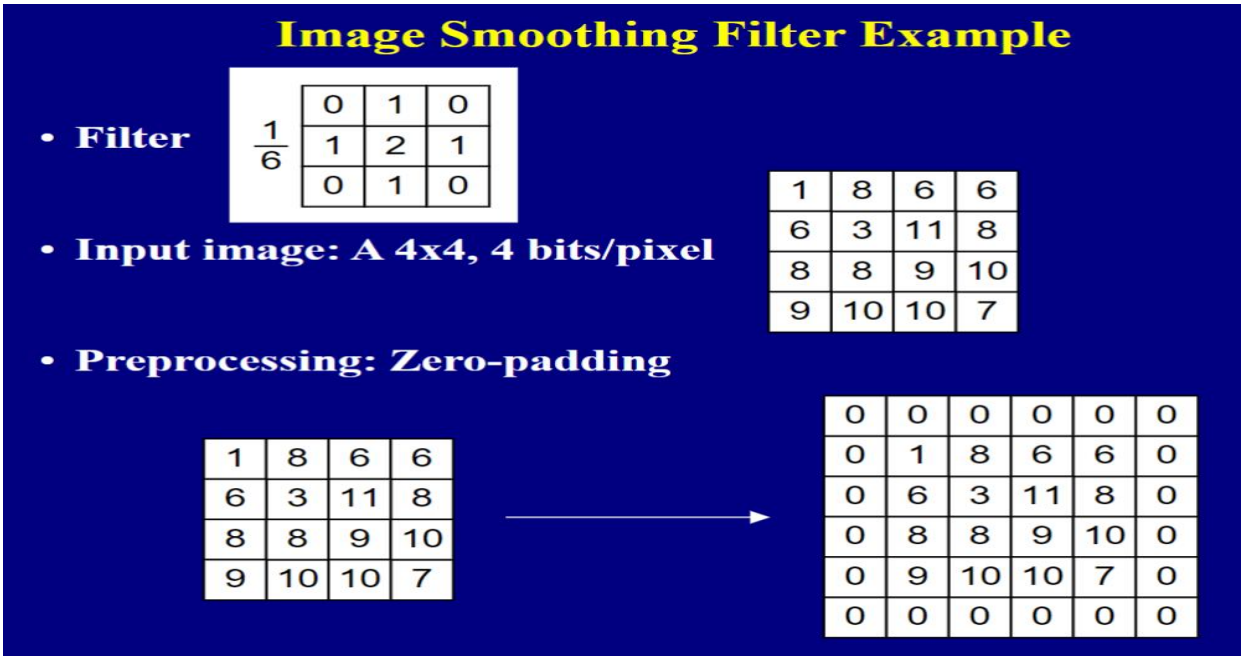
Here, we use a NxN filter and the we are also dividing it by N^2 so it will act like an average operator. So if this h is a block then we will move this block throughout the image and we will average them. It will reduce the noise but at the same time it will also make the image blur.



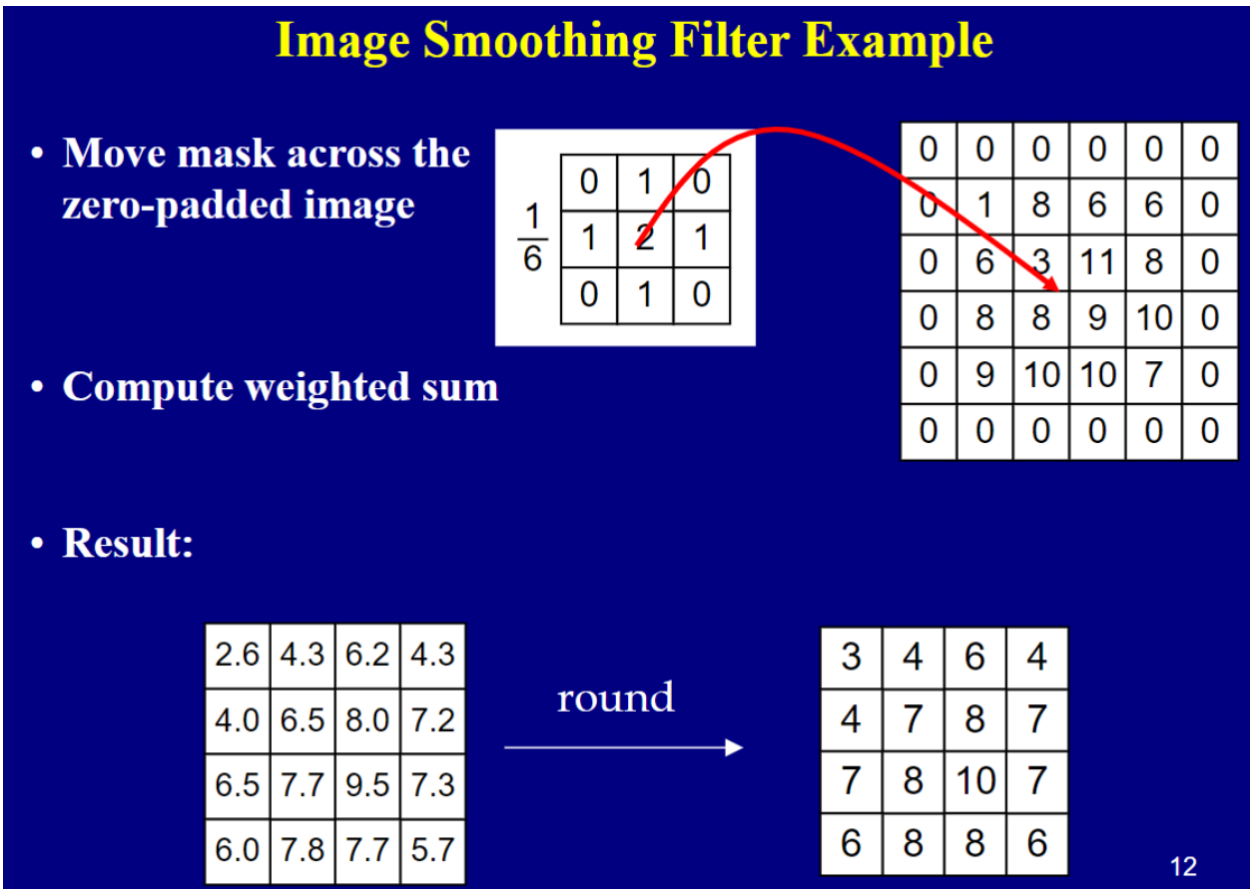
In the average filter scenario we use the square shape filter. Now, we will use some other filter and that is called the gaussian filter. We will use the gaussian shape as shown.



Here, the sigma will define the width of the gaussin distribution. With the help of sigma we can make the filter bigger or smaller. Here you can see that as we increase the sigma value the smoothing effect on the image becomes stronger and stronger. The noise gets significantly reduced but at the same time the image becomes blurry.



Here zero-padding is added because we have to cover all the pixel values for the image. If we apply direct filtering on the image then we might not be able to apply this filter on each and every pixel. But with the help of zero padding we can cover all the pixel values.



Spatial Linear filters can not only be used to smoothing the image, but it can also be used to sharpening the image. We will use the Laplacian operator to sharpening the image.

Sharpening Linear Filters

- **Discrete Approximation of Laplacian**

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

  - Zero at smooth regions
  - Sensitive to image details

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

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

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

The Laplacian operator will help us to sharpen the edges in the image.

Sharpening Linear Filters

- **Image Sharpening Idea: combining Laplacian with the image itself**
  - Case 1: Center coefficient of the Laplacian mask is positive

$$g(x,y) = f(x,y) + \nabla^2 f(x,y)$$

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

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

  - Case 2: Center coefficient of the Laplacian mask is negative

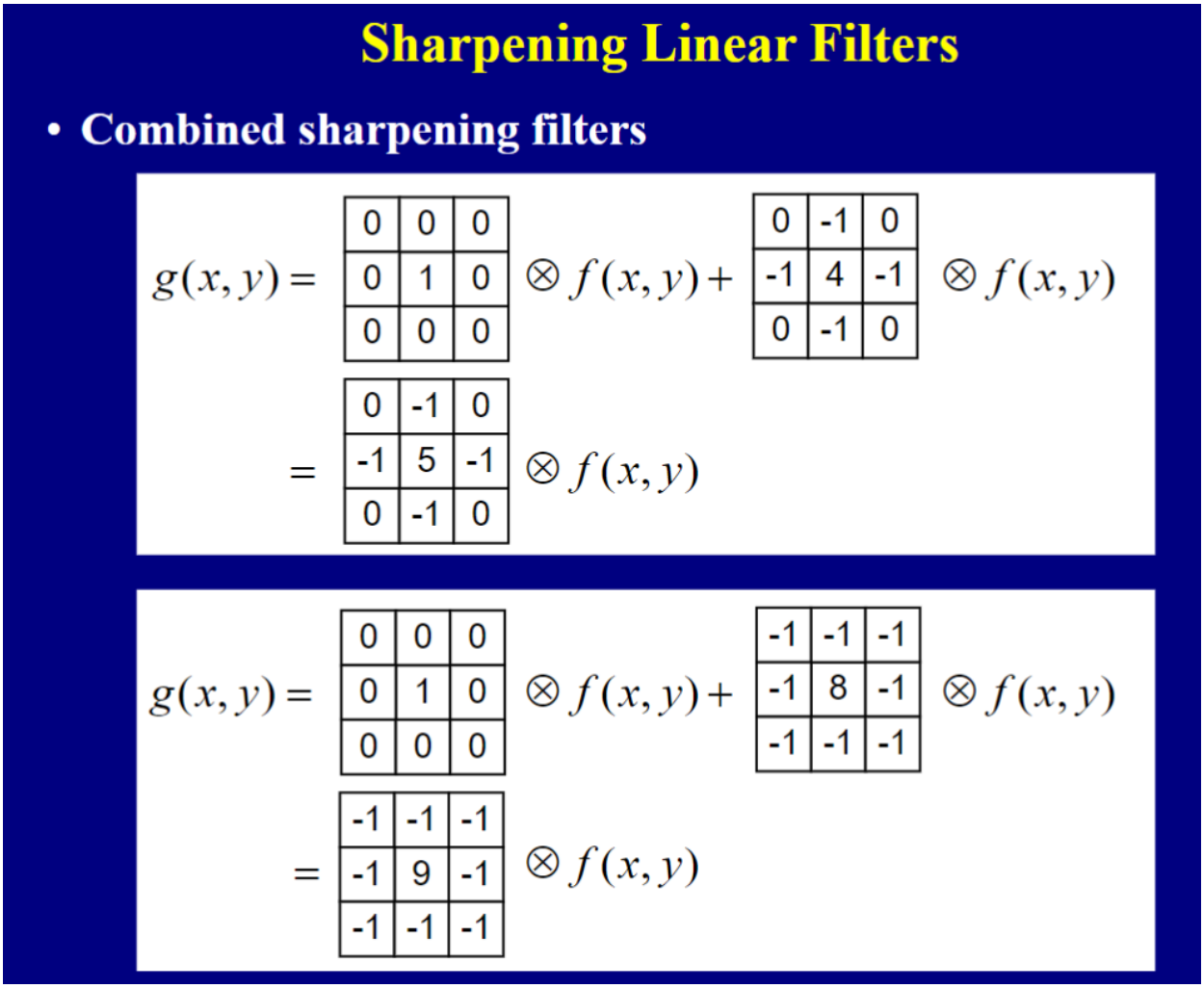
$$g(x,y) = f(x,y) - \nabla^2 f(x,y)$$

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

15

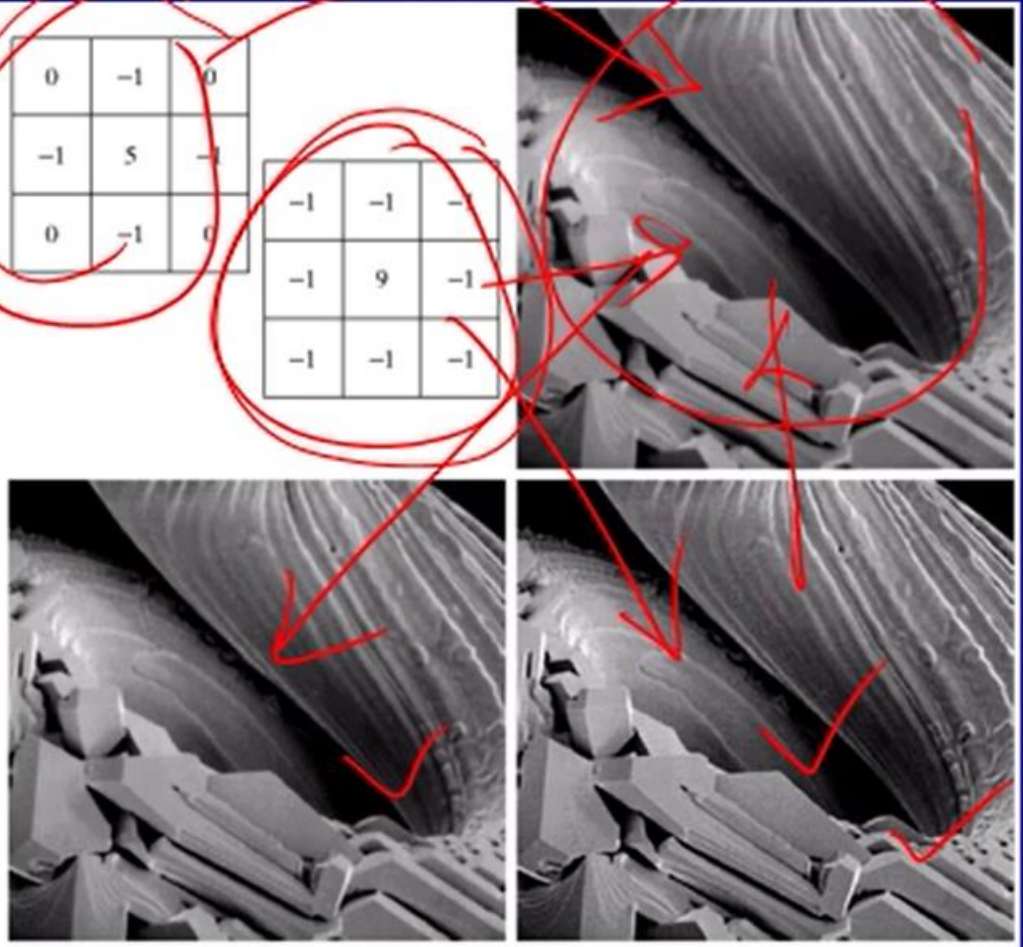
In case 1, there is + sign so it will be positive inside and negative outside as shown in 3x3 boxes. While in case 2, there is – sign so it will be negative inside and positive outside. Here, f is the origianl image and delta is the laplacian operator.



Sharpening Linear Filters

0	−1	0
−1	5	−1
0	−1	0

−1	−1	−1
−1	9	−1
−1	−1	−1

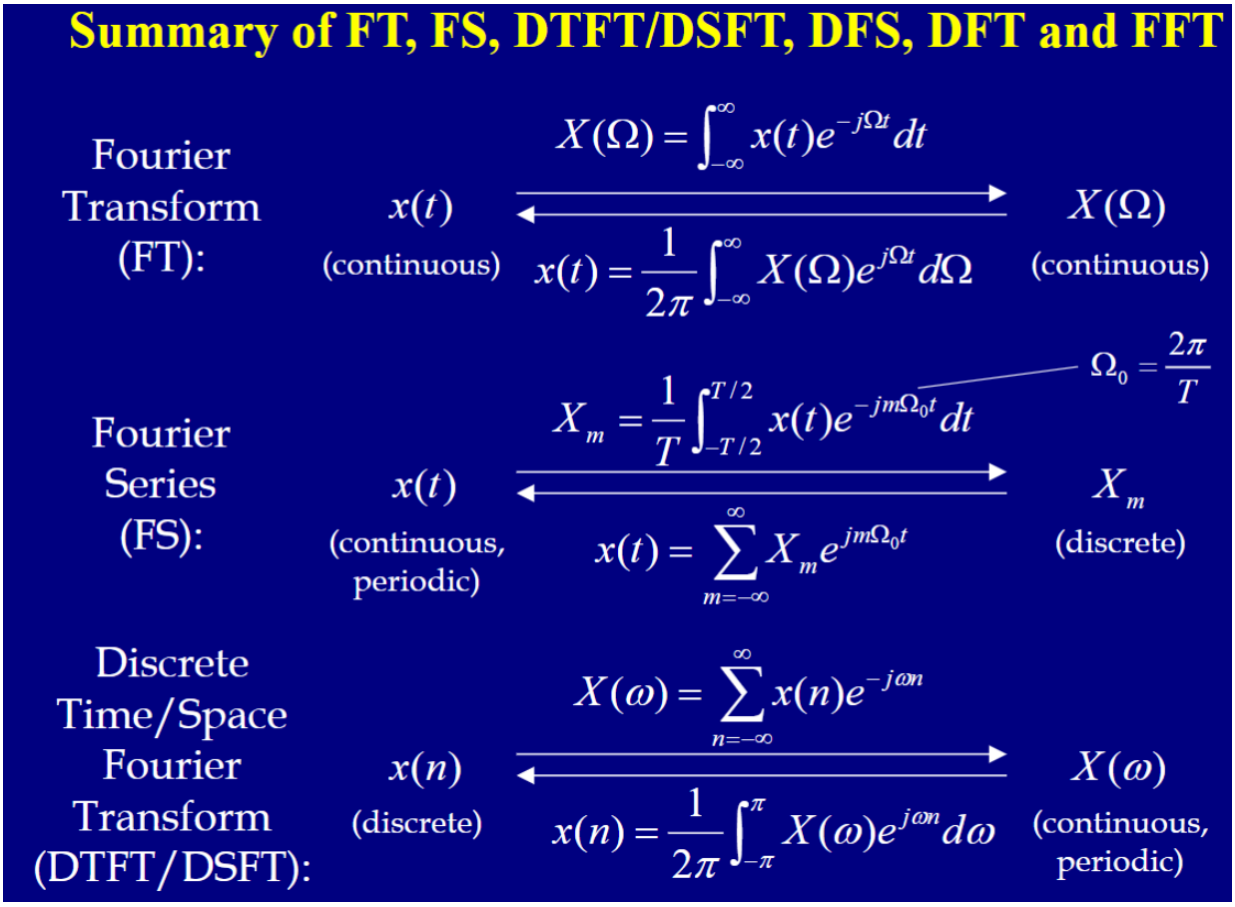


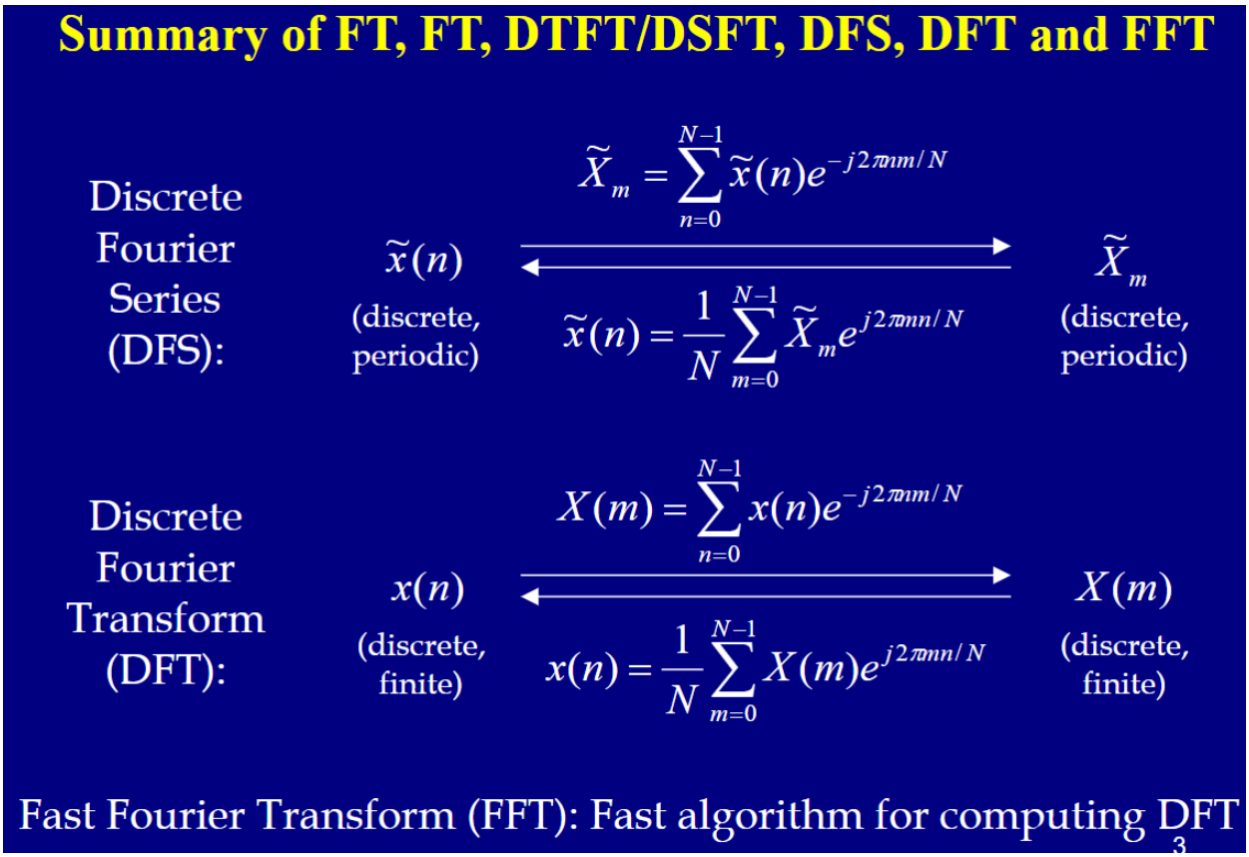
a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

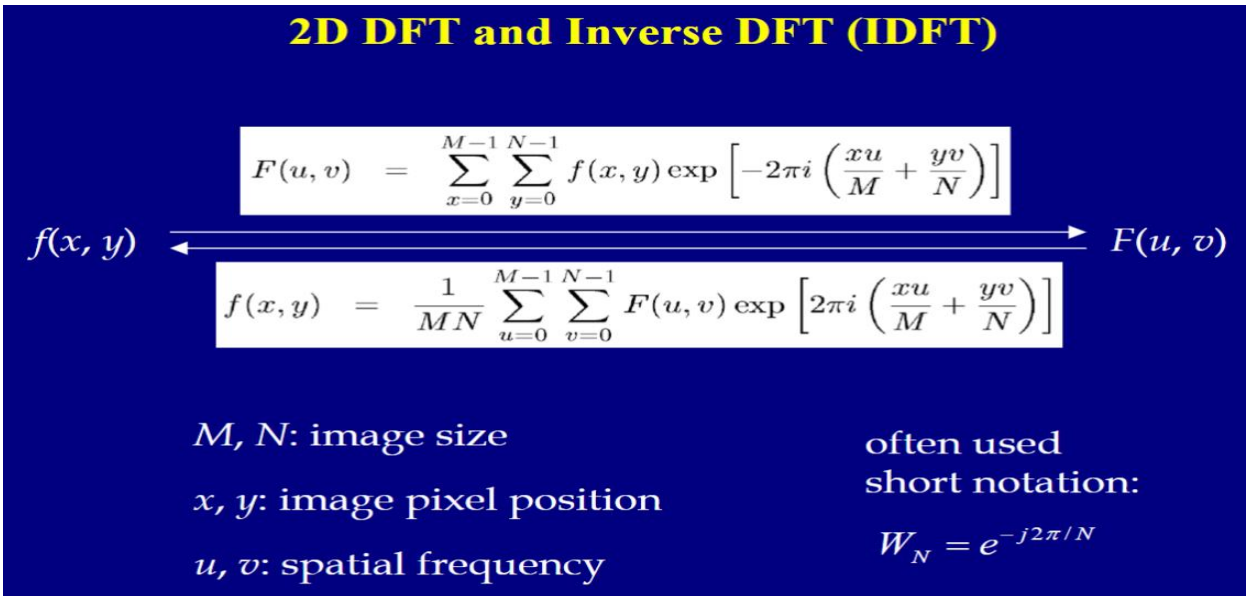
2D Discrete Fourior Transform

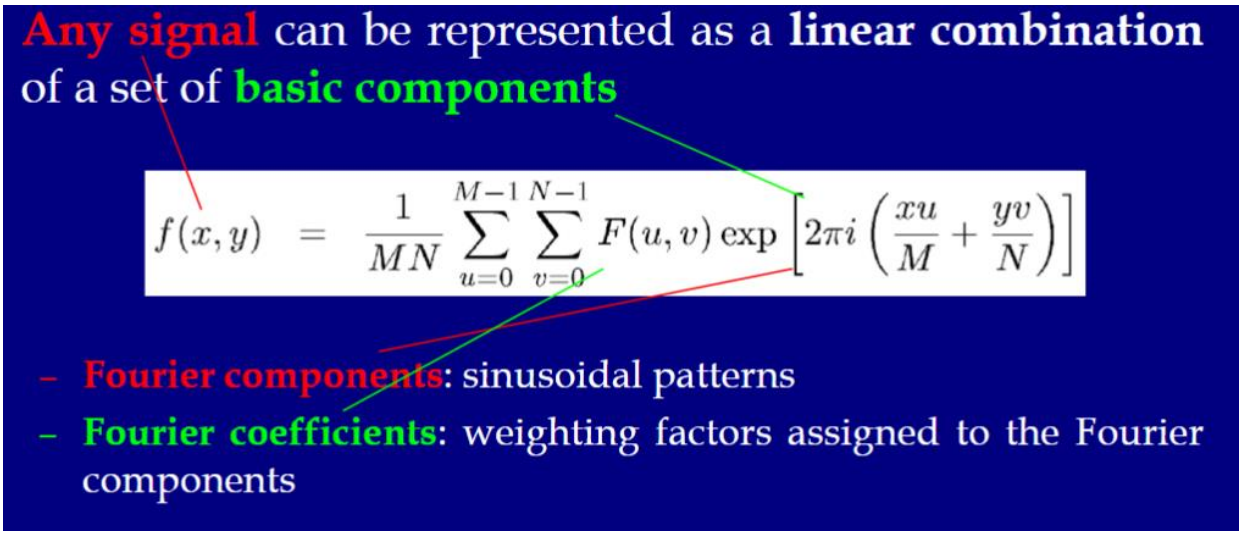
There are few terms Fourier Transform (FT), Fourier Series (FS), Discrete Time Fourier Transform (DTFT), Discrete Space Fourier Transform (DSFT), Discrete Fourier Series (DFS), Discrete Fourier Transform (DFT), Fast Fourier Trasnform (FFT).





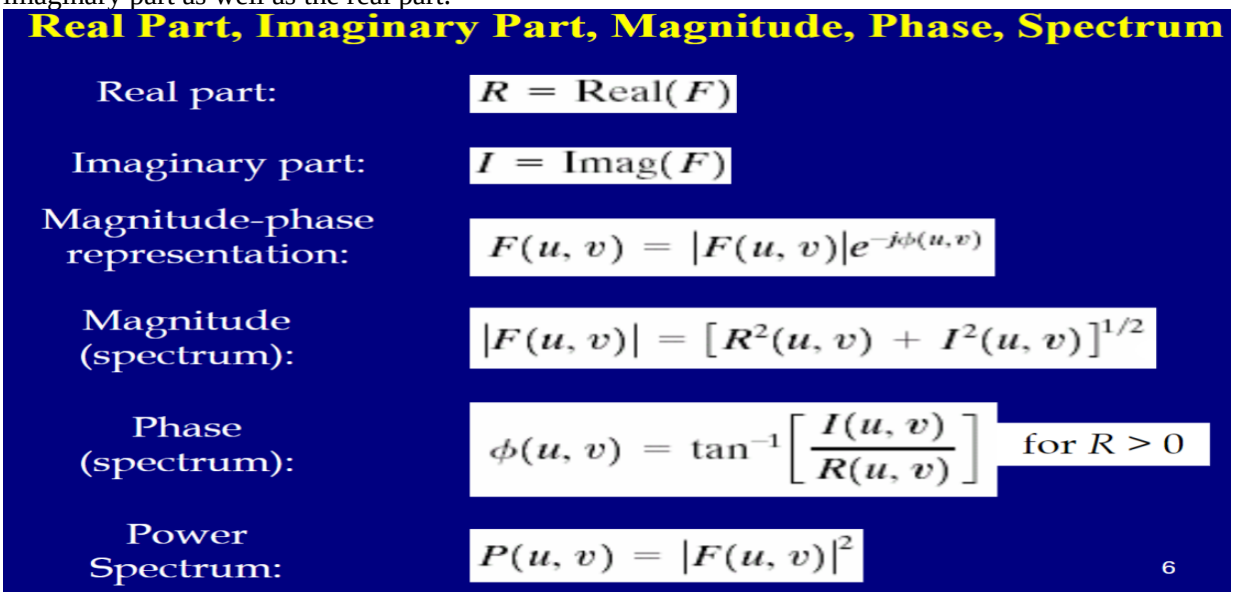
Try to understand the figures all of them are self explanatory.





Here looking at the equation depending upon the values of u and v we can observe that it will form the sinosoidial wave. It can be horizontal, vertical or may be at a given angle sinosoidial wave. F(u,v) is the coefficient assigned to the exponential part. Then at the end we have a product of M and N in the denominator. Togather it will constitute the DFT. It will be the linear combination of a set of basic components. Basic components consist of fourier components, fourier coefficients.

The **Spatial Frequency** is the frequency of the fourier component. Here, depending upon the values of u and v, frequency is defined. Do not confuse with the electromagnetic frequencies. Generally all the fourier coefficients are the complex numbers. And we all know that the complex numbers include the imaginary part as well as the real part.



These are the basic formulas of the above mentioned terms.
Now consider just like 1D DFT we also have a list of propertiesof 2D DFT as shown in the below image.

2D DFT Properties

Mean of image/ DC component:

$$\bar{f}(x, y) = F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

Highest frequency component:

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

"Half-shifted" Image:

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

Conjugate Symmetry:

$$F(u, v) = F^*(-u, -v)$$

Magnitude Symmetry:

$$|F(u, v)| = |F(-u, -v)|$$

7

Mean of an image/DC component simply represent the average of an image, which can also be seen from the formula. F(x, y) is the original image. Here, u – M/2 is shifting u over M/2, and v – N/2 is the shifting v over N/2. We also have the symmetry property as shown as the conjugate symmetry and the magnitude symmetry.

2D DFT Properties

Spatial domain differentiation:

$$\frac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$$

Frequency domain differentiation:

$$(-jx)^n f(x, y) \Leftrightarrow \frac{\partial^n F(u, v)}{\partial u^n}$$

Distribution law:

$$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$$

Laplacian:

$$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2)F(u, v)$$

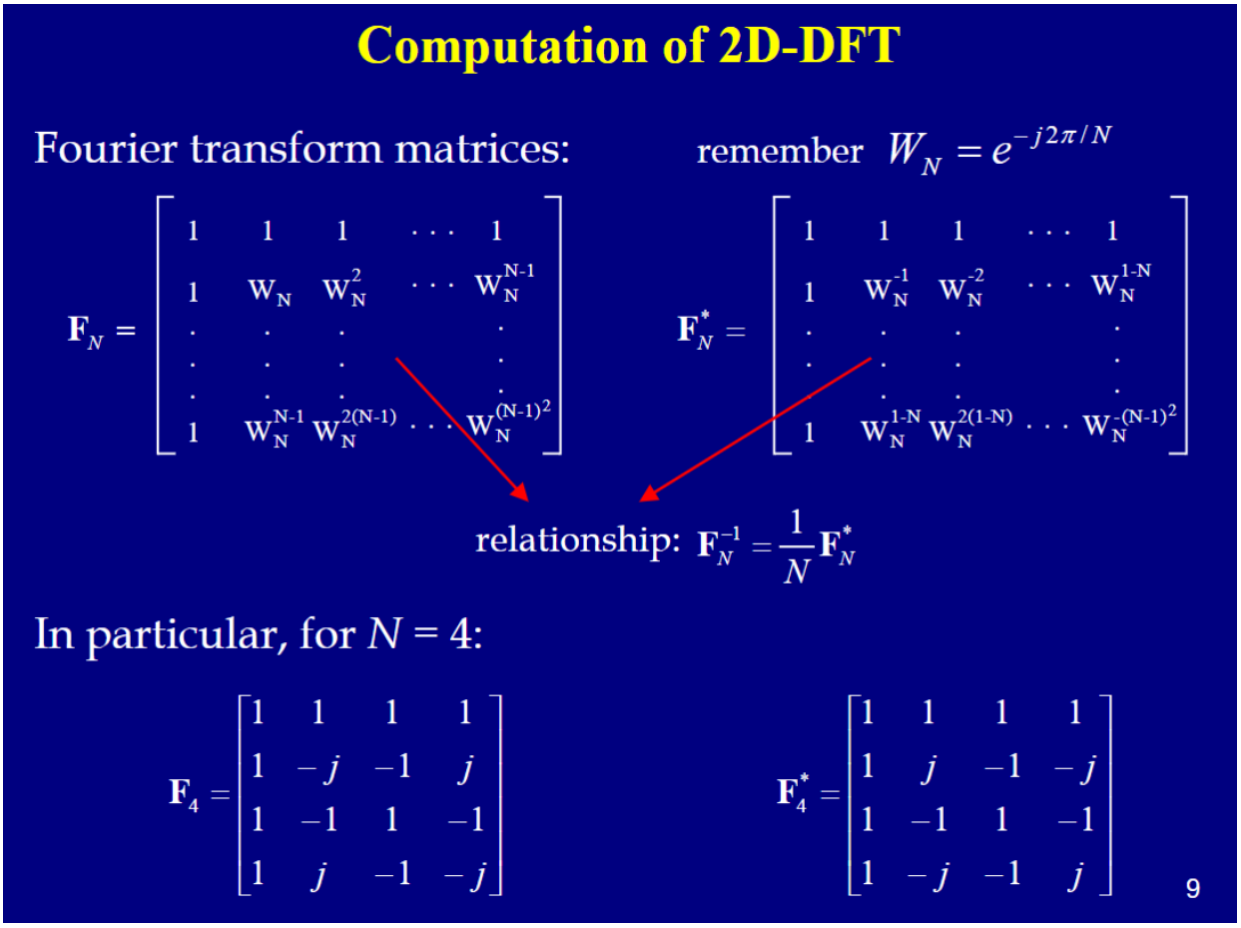
Spatial domain Periodicity:

$$f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$$

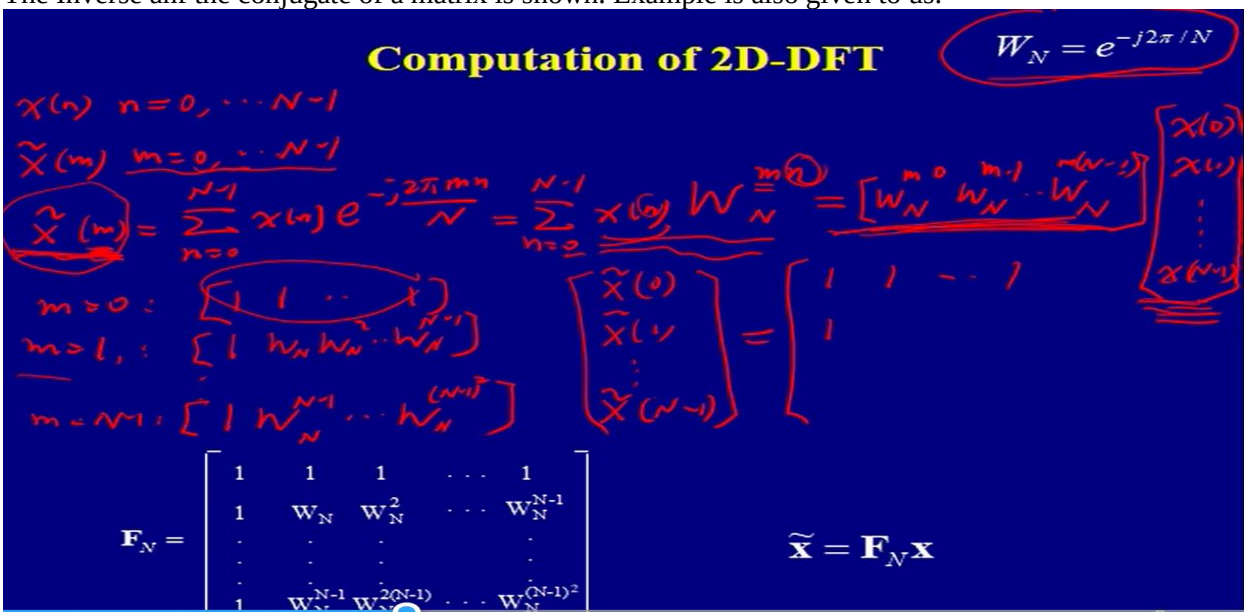
Frequency domain periodicity:

$$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$$

DFT and the DFS both are the same.



The Inverse anf the conjugate of a matrix is shown. Example is also given to us.



Computation of 2D-DFT

- To compute the 1D-DFT of a 1D signal **x** (as a vector):

$$\tilde{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$$

To compute the inverse 1D-DFT:

$$\mathbf{x} = \frac{1}{N} \mathbf{F}_N^* \tilde{\mathbf{x}}$$

- To compute the 2D-DFT of an image **X** (as a matrix):

$$\tilde{\mathbf{X}} = \mathbf{F}_N \mathbf{X} \mathbf{F}_N$$

To compute the inverse 2D-DFT:

$$\mathbf{X} = \frac{1}{N^2} \mathbf{F}_N^* \tilde{\mathbf{X}} \mathbf{F}_N^*$$

Here, 1/N and 1/ N^2 are simply called the scale factor.

Computation of 2D-DFT: Example

- A 4x4 image
- Compute its 2D-DFT:

$$\mathbf{x} = \begin{bmatrix} 1 & 3 & 6 & 8 \\ 9 & 8 & 8 & 2 \\ 5 & 4 & 2 & 3 \\ 6 & 6 & 3 & 3 \end{bmatrix}$$

$$\tilde{\mathbf{X}} = \mathbf{F}_4 \mathbf{X} \mathbf{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 & 3 & 6 & 8 \\ 9 & 8 & 8 & 2 \\ 5 & 4 & 2 & 3 \\ 6 & 6 & 3 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

MATLAB function: *fft2*

$$= \begin{bmatrix} 21 & 21 & 19 & 16 \\ -4-3j & -1-2j & 4-5j & 5+j \\ -9 & -7 & -3 & 6 \\ -4+3j & -1+2j & 4+5j & 5-j \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

lowest frequency component

highest frequency component

$$= \begin{bmatrix} 77 & 2-5j & 3 & 2+5j \\ 4-9j & -11+8j & -4-7j & -5-4j \\ -13 & -6+13j & -11 & -6-13j \\ 4+9j & -5+4j & -4+7j & -11-8j \end{bmatrix}$$

Here, we have given a 4x4 image and we have to conpute a 2D-DFT for it. Since it is a 4 x4 image we know the value of F_4 as shown in the example. So by simply putting the values we get X bar. This whole computation can be done faster using the MATLAB fft2 sunction. So this is how the 2D-DFT is calculated. The top-left corner always represents the DC term and it will give the lowest frequency. If we want to find the highest frequency component then we have to divide the 4x4 matrix by 2 so that we can get the 4 2x2 matrices. Noe the top-left element or the first element of this last 2x2 matrix will give us the highest frequency.

Computation of 2D-DFT: Example

$$\tilde{\mathbf{X}} = \begin{bmatrix} 77 & 2-5j & 3 & 2+5j \\ 4-9j & -11+8j & -4-7j & -5-4j \\ -13 & -6+13j & -11 & -6-13j \\ 4+9j & -5+4j & -4+7j & -11-8j \end{bmatrix}$$

Real part:

$$\tilde{\mathbf{X}}_{real} = \begin{bmatrix} 77 & 2 & 3 & 2 \\ 4 & -11 & -4 & -5 \\ -13 & -6 & -11 & -6 \\ 4 & -5 & -4 & -11 \end{bmatrix}$$

Imaginary part:

$$\tilde{\mathbf{X}}_{imag} = \begin{bmatrix} 0 & -5 & 0 & 5 \\ -9 & 8 & -7 & -4 \\ 0 & 13 & 0 & -13 \\ 9 & 4 & 7 & -8 \end{bmatrix}$$

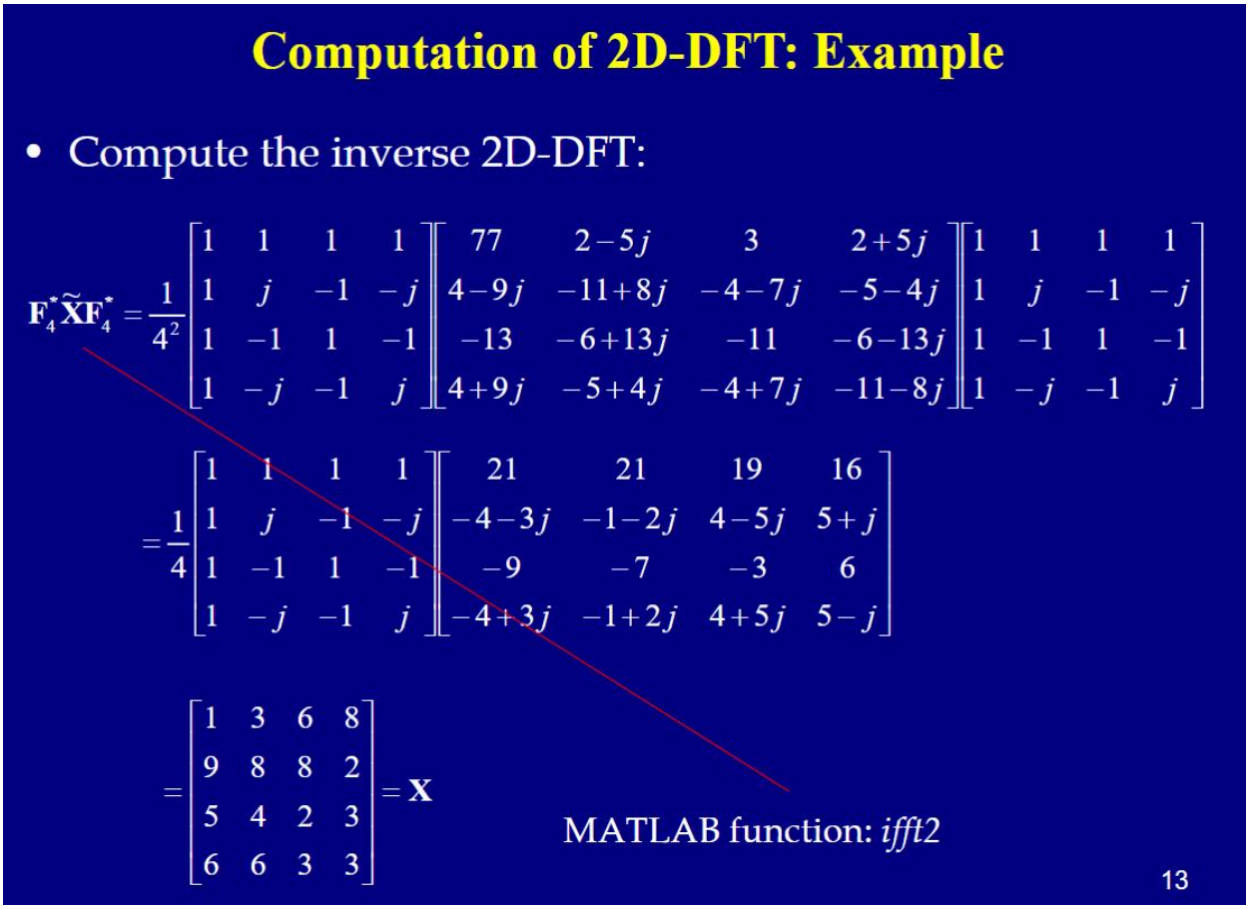
Magnitude:

$$\tilde{\mathbf{X}}_{magnitude} = \begin{bmatrix} 77 & 5.39 & 3 & 5.39 \\ 9.85 & 13.60 & 8.06 & 6.4 \\ 13 & 14.32 & 11 & 14.32 \\ 9.85 & 6.40 & 8.06 & 13.60 \end{bmatrix}$$

Phase:

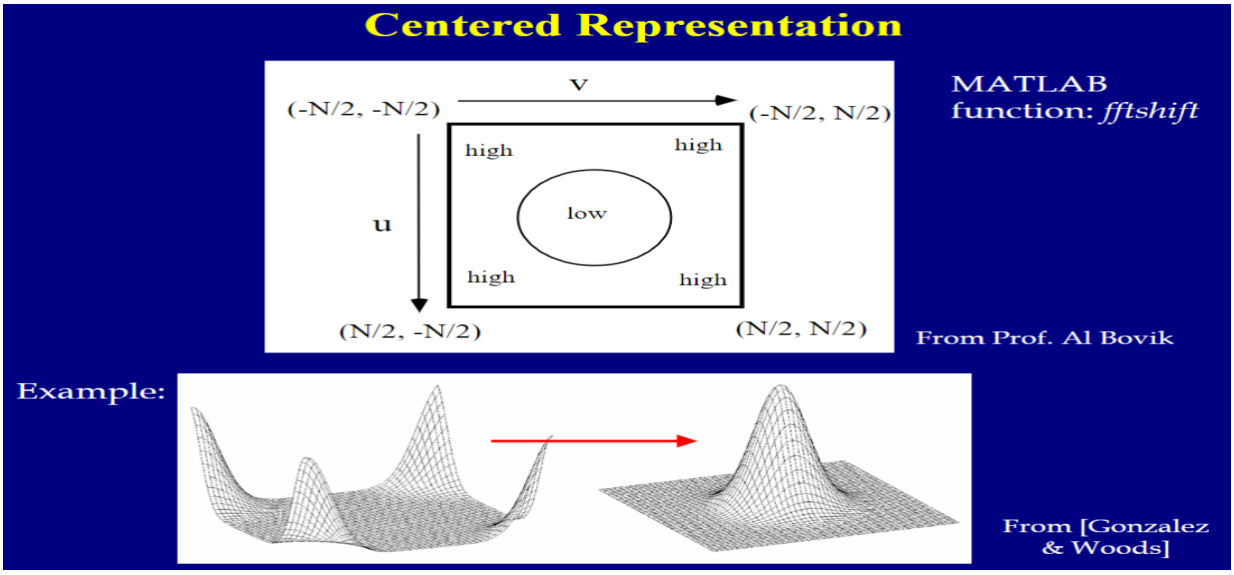
$$\tilde{\mathbf{X}}_{phase} = \begin{bmatrix} 0 & -1.19 & 0 & 1.19 \\ -1.15 & 2.51 & -2.09 & -2.47 \\ 3.14 & 2.00 & 3.14 & -2.00 \\ 1.15 & 2.47 & 2.09 & -2.51 \end{bmatrix}$$

We can calculate the real part, imaginary part, Magnitude and phase from the given X bar matrix.

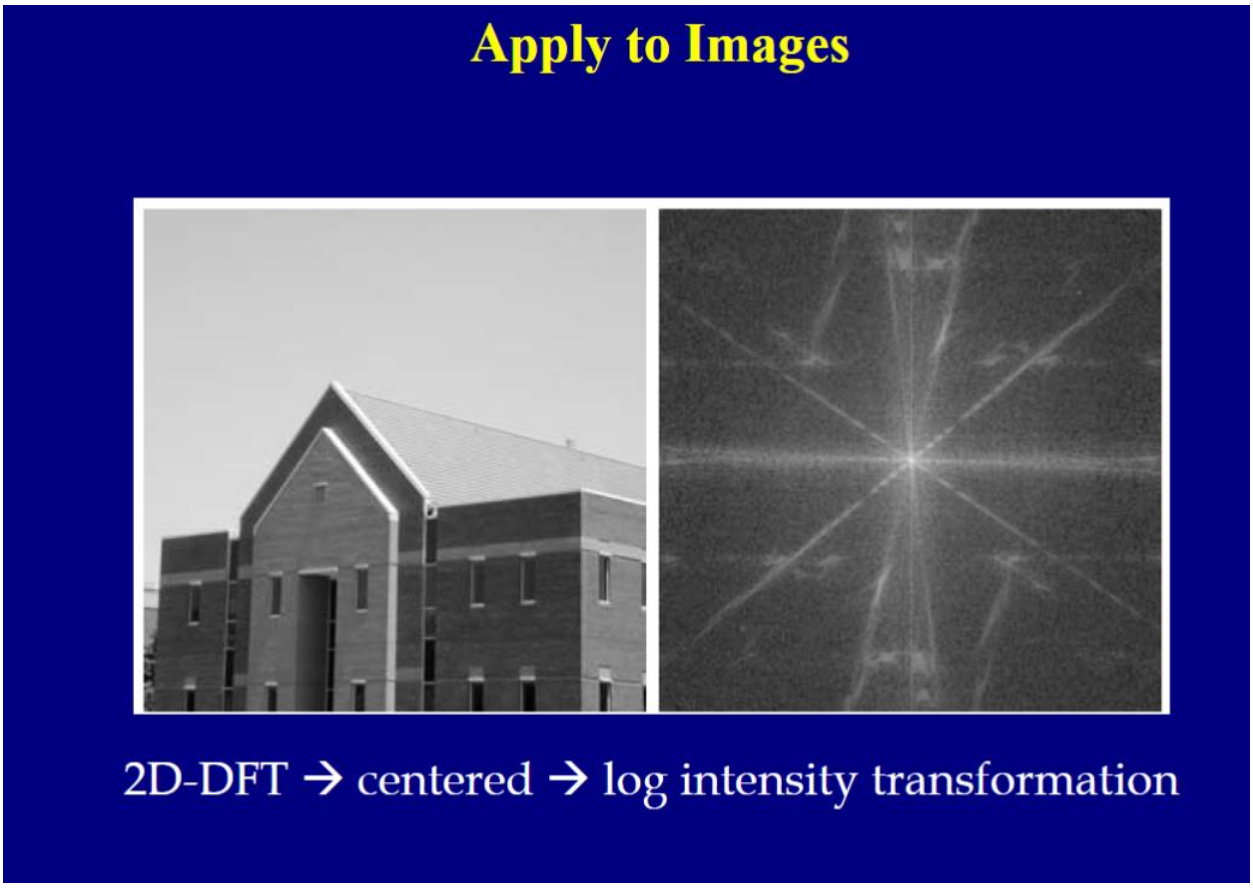


This above figure shows how to calculate the inverse 2D-DFT.

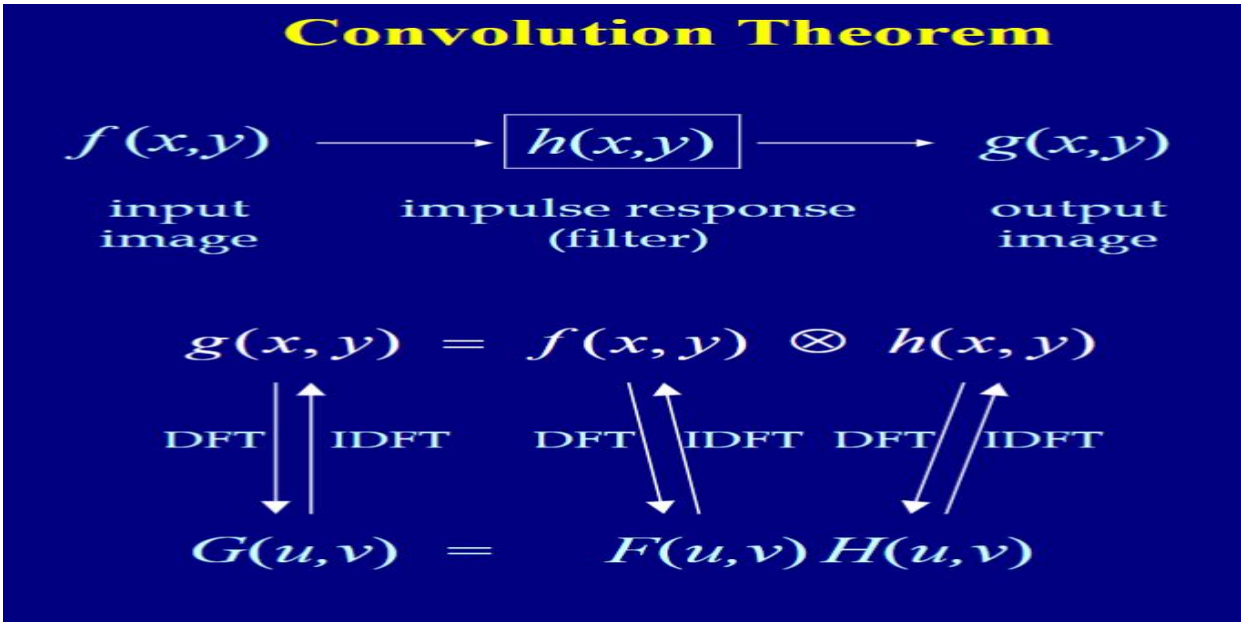
In most of the images the low frequency is at the center and the high frequency will be concentrated at the cornors.



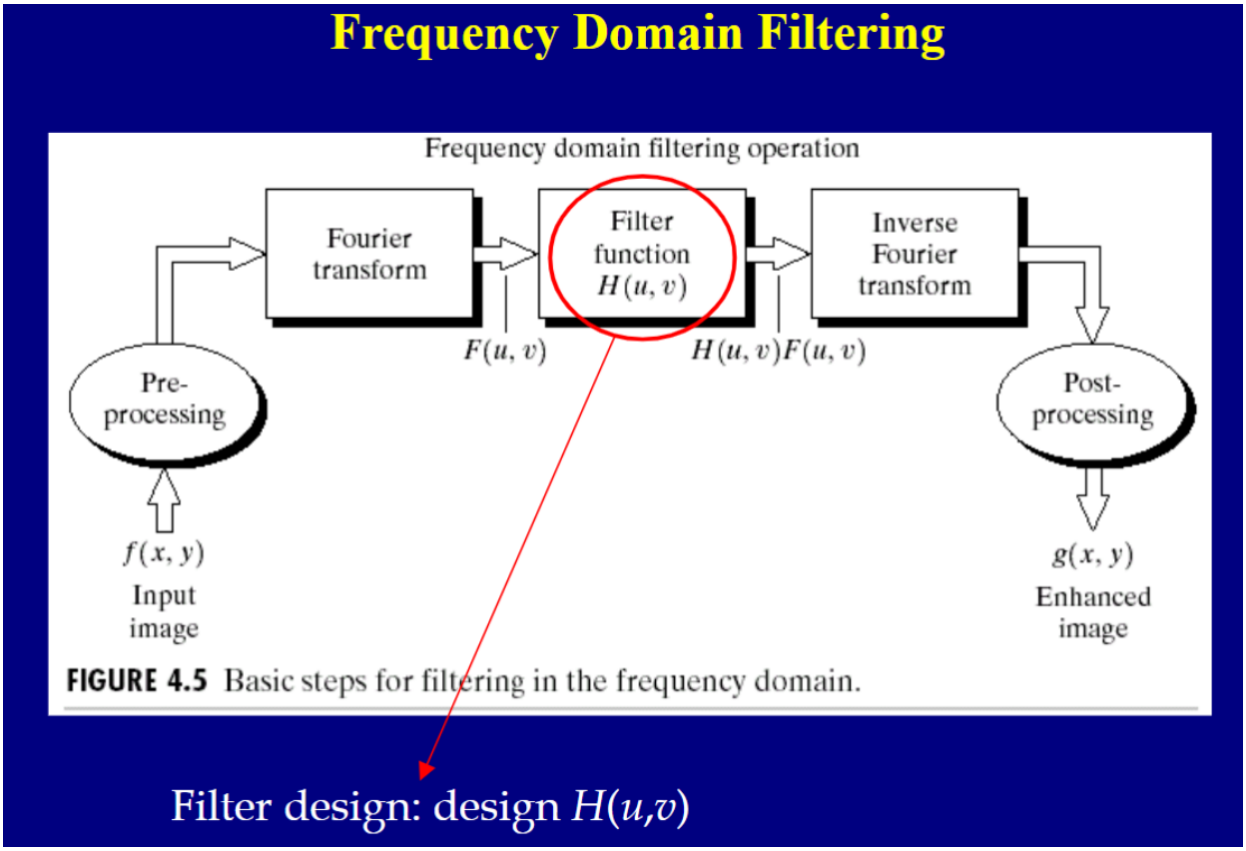
fftshift will try to move the low frequency shift to the center. fftshift is a MATLAB function.



Here in the image on the right side you can see the dot at the center and there are several lines passing through the center dot. The white dot represents the energy concentration and the lines are created because of some perpendicular edge in the original image. We can see that there are many sharp edges in th eoriginal left side image.



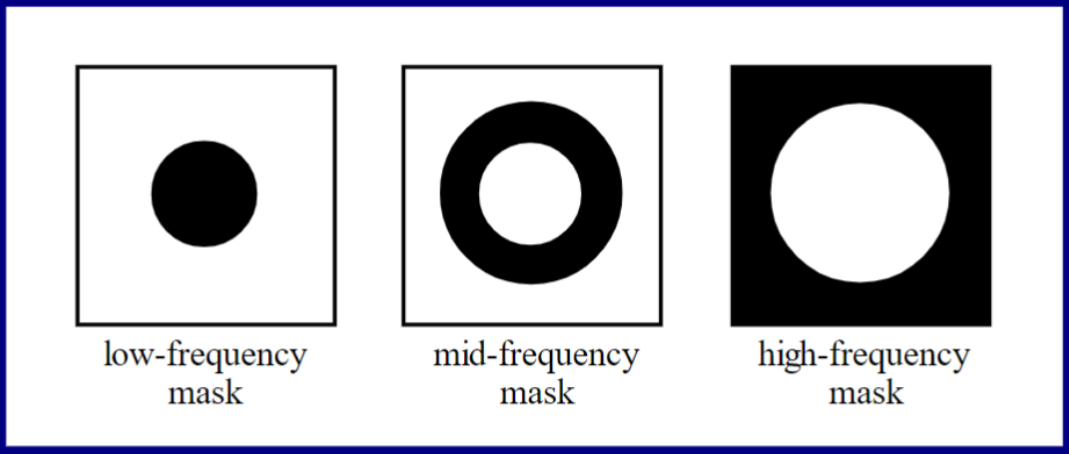
It is very important for image processing because in image processing we frequently perform image filtering operation. Let consider we have a filter h and an input image f. So If the input image f is passed to the concolve h then we will get the output as g. The complexity depends upon the dimensions of h. The convolutional theorm says that you can apply fourier transform to any specific individual component.



This is the diagram for standard frequency domain filtering. If we have a spatial domain image then we first need to apply the fourier transform. So that we will get F(u,v). In frequency we simply do the dot product of the F(u,v) with the impulse response (H(u,v)). So, now we have G(u,v) and we apply inverse fourier transform to get the original signal back.

2D-DFT Domain Filter Design

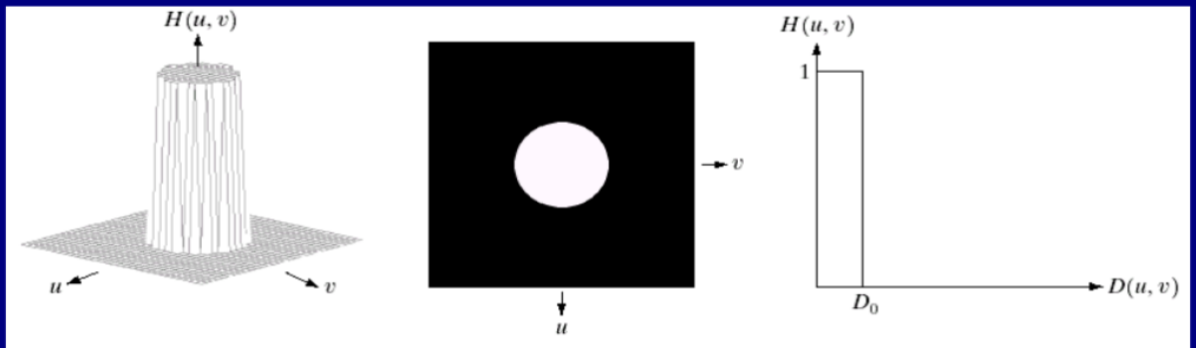
- Ideal lowpass, bandpass and highpass



Instead of designing H(x,y), people started to design their own filters in the frequency domain. So you can easily define the ideal low-pass, ideal bandpass and the ideal high-pass filters. Suppose the black means 0 and white represents 1. So, we know that the low frequency is concentrated at the center. so the left most image shows the similar represenation.

2D-DFT Domain Filter Design

- Ideal lowpass, bandpass and highpass

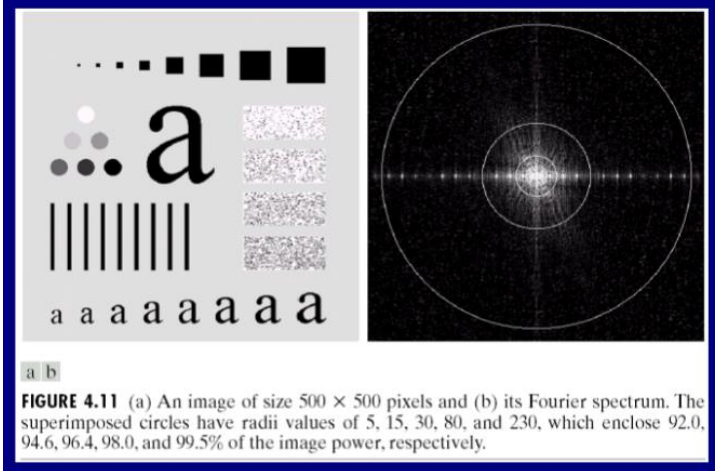


a b c

FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

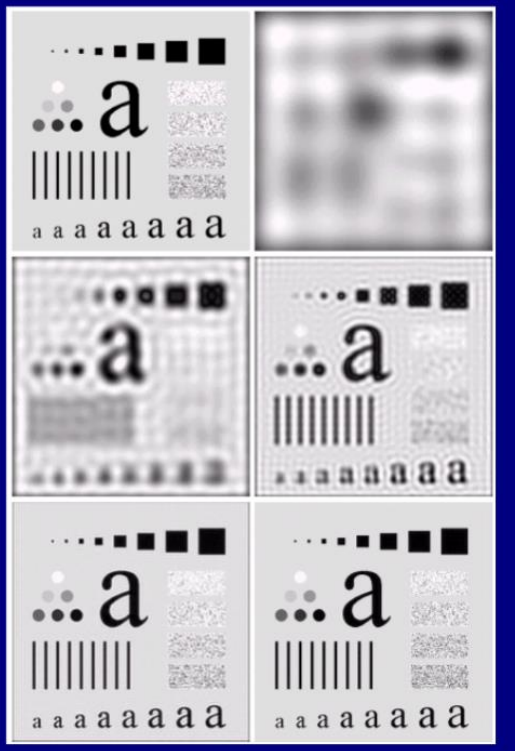
The above image shows the different approaches for the ideal low pass filter.

2D-DFT Domain Filter Design



a b

FIGURE 4.11 (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.

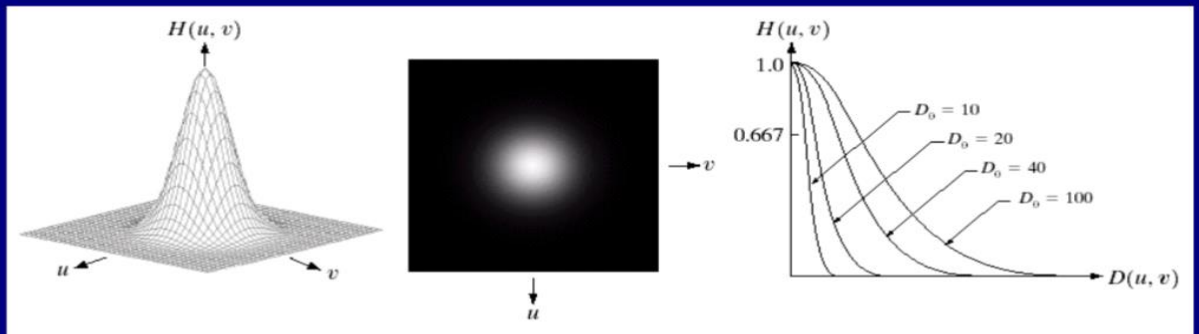


Ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, respectively

From [Gonzalez & Woods]

2D-DFT Domain Filter Design

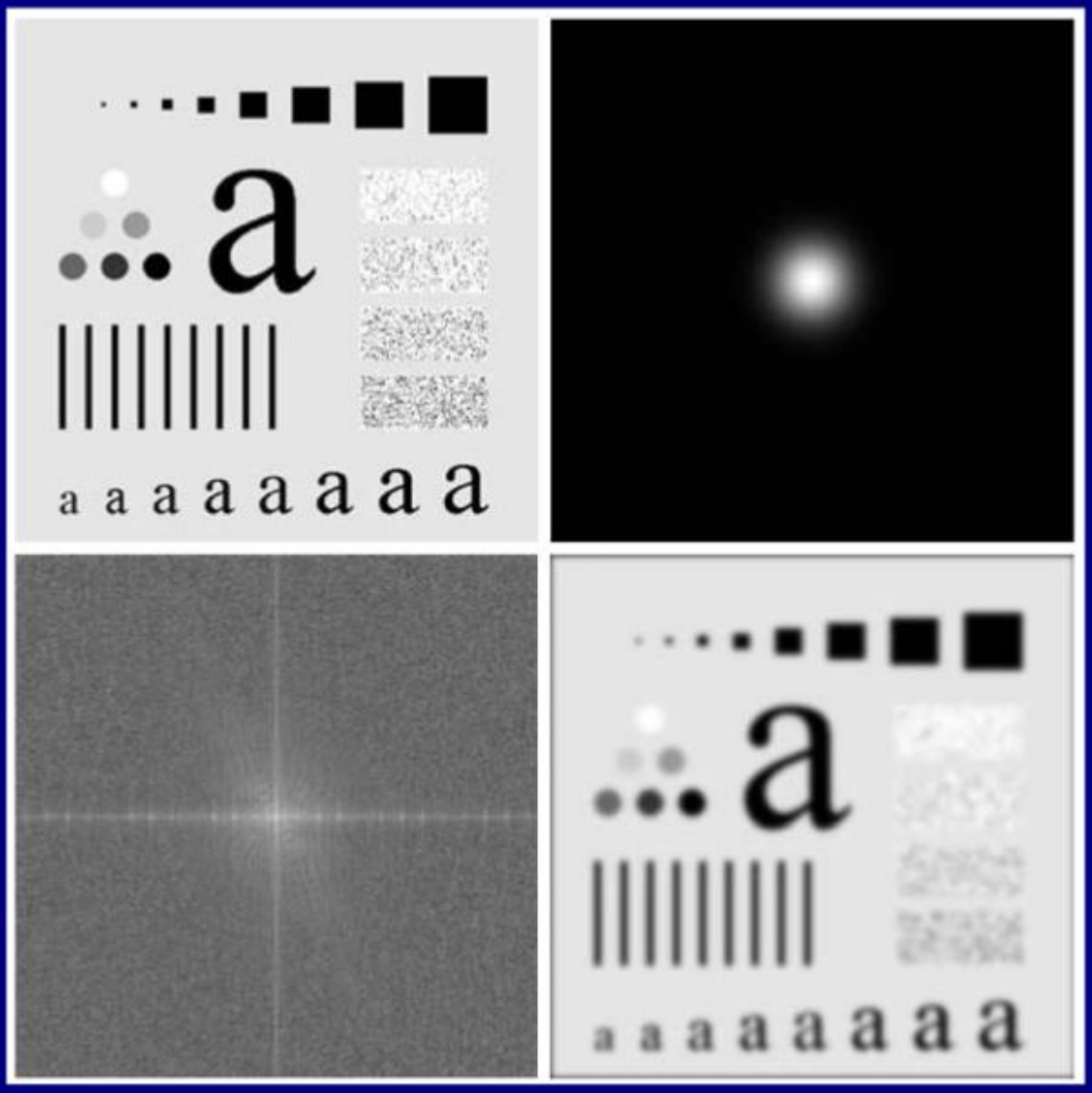
- Gaussian lowpass



a b c

FIGURE 4.17 (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.

2D-DFT Domain Filter Design



Effect of Gaussian lowpass filter

2D-DFT Domain Filter Desig

Effect of Gaussian lowpass filter

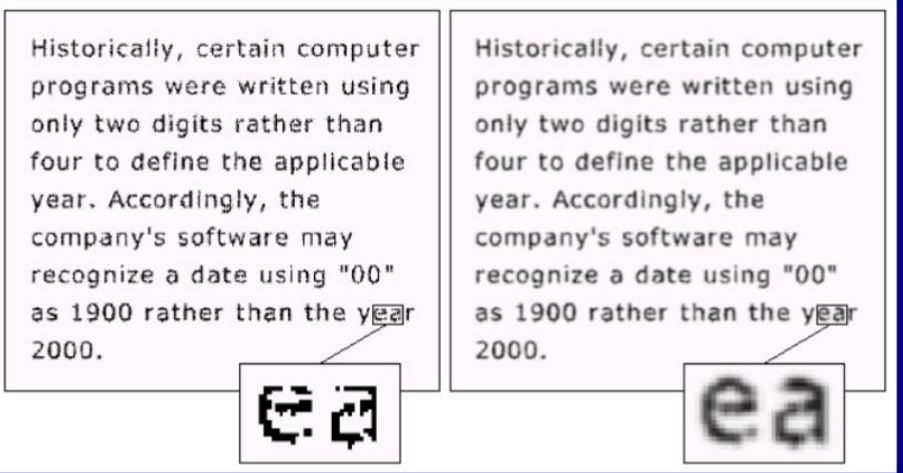
FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

a b
c d
e f

2D-DFT Domain Filter Design

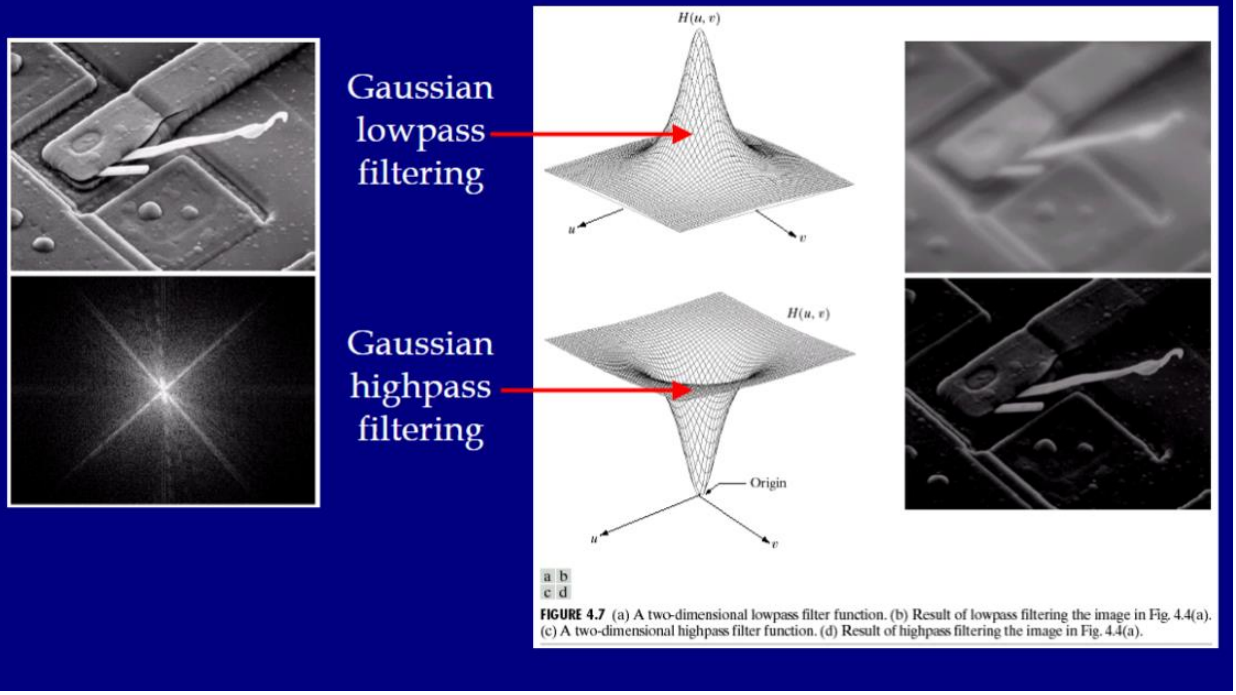
a b

FIGURE 4.19
(a) Sample text of poor resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).



Effect of Gaussian lowpass filter

2D-DFT Domain Filter Design



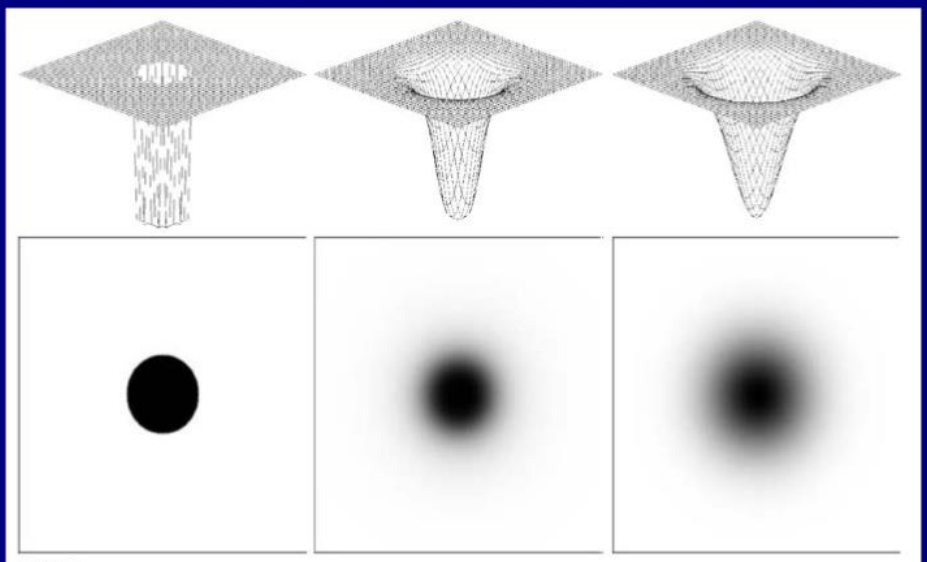
a b
c d

FIGURE 4.7 (a) A two-dimensional lowpass filter function. (b) Result of lowpass filtering the image in Fig. 4.4(a). (c) A two-dimensional highpass filter function. (d) Result of highpass filtering the image in Fig. 4.4(a).

2D-DFT Domain Filter Design

- Choices of highpass filters

Ideal Butterworth Gaussian



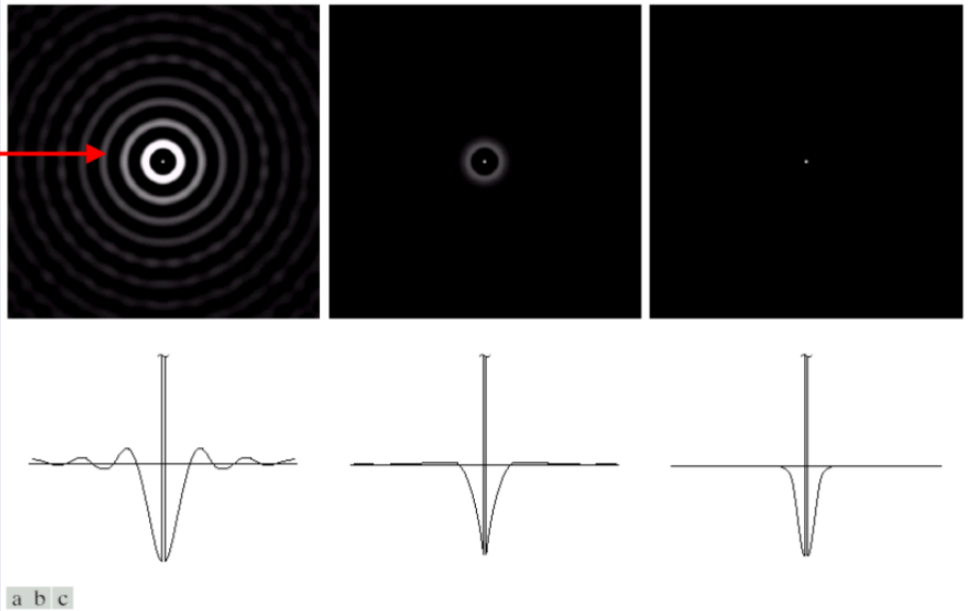
a b c
d e f

FIGURE 4.17 Top row: Perspective plots of ideal, Butterworth, and Gaussian highpass filters. Bottom row: Corresponding images.

2D-DFT Domain Filter Design

Ideal Butterworth Gaussian

Obtained by applying inverse 2D-DFT to the corresponding frequency domain filters

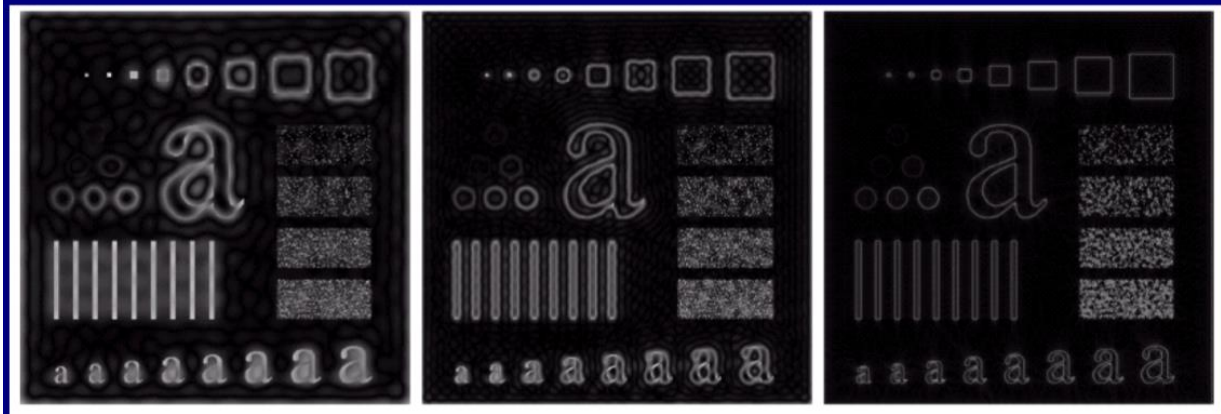


a b c

FIGURE 4.23 Spatial representations of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding gray-level profiles.

2D-DFT Domain Filter Design

Ideal Butterworth Gaussian



a b c

FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15$, 30, and 80, respectively. Problems with ringing are quite evident in (a) and (b).

2D-DFT Domain Filter Design

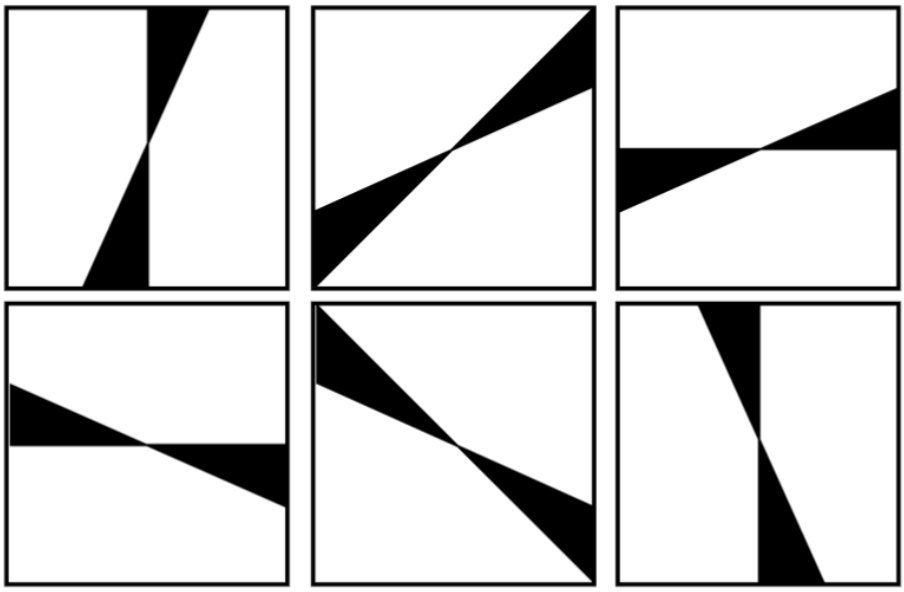
Gaussian filter with different width

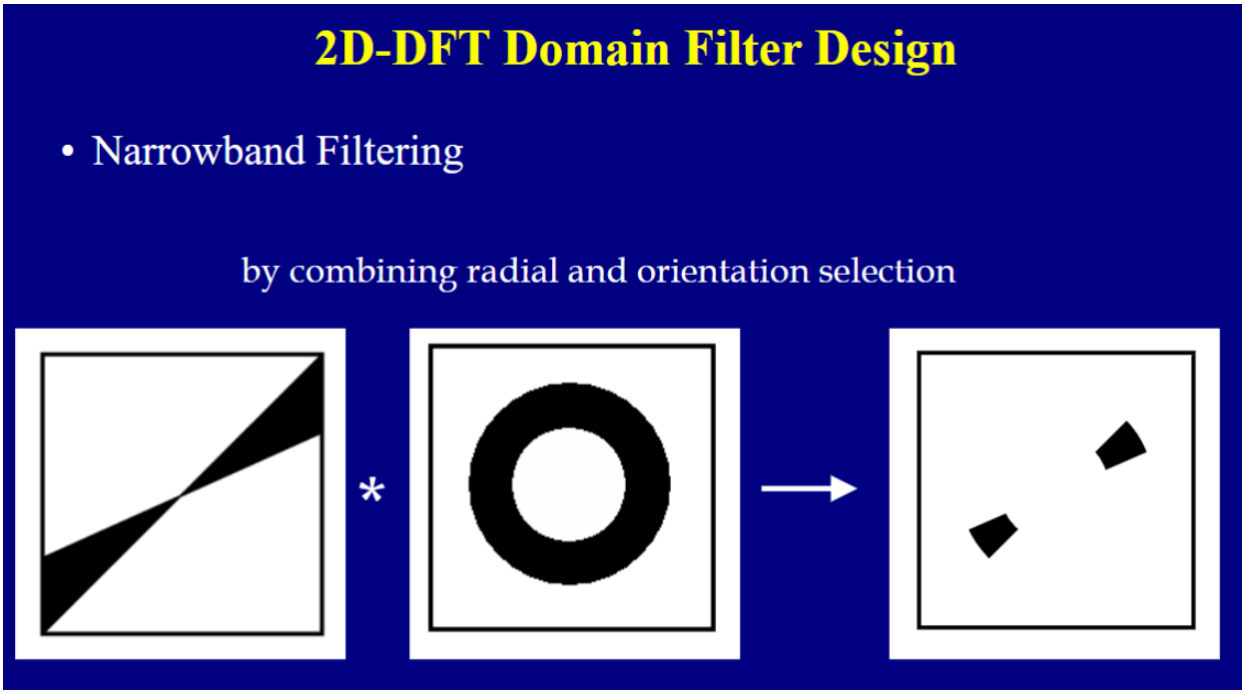
a b c

FIGURE 4.26 Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. Compare with Figs. 4.24 and 4.25.

2D-DFT Domain Filter Design

- Orientation selective filters





Frequency Domain Image Restoration

In the field of image processing, by the term Linear Filtering then bydefault we mean that applying the Linear Shift Invarient system to an input image and produce an output image, and this thing can be implemeted in the **Spatial Domain** and in the **Frequency Domain** as well. In the Spatial Domain, the output is the convolutional of the input and impulse Response.

Linear Filtering

- **Spatial Domain Filtering: Convolution**

$$x(m,n) \longrightarrow \boxed{h(m,n)} \longrightarrow y(m,n)$$

$$y(m,n) = \sum_{k,l=-\infty}^{\infty} h(k,l)x(m-k,n-l) = h(m,n) \otimes x(m,n)$$

- **Frequency Domain Filtering: Multiplication**

$$X(u,v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(m,n)e^{-j(um+vn)}$$

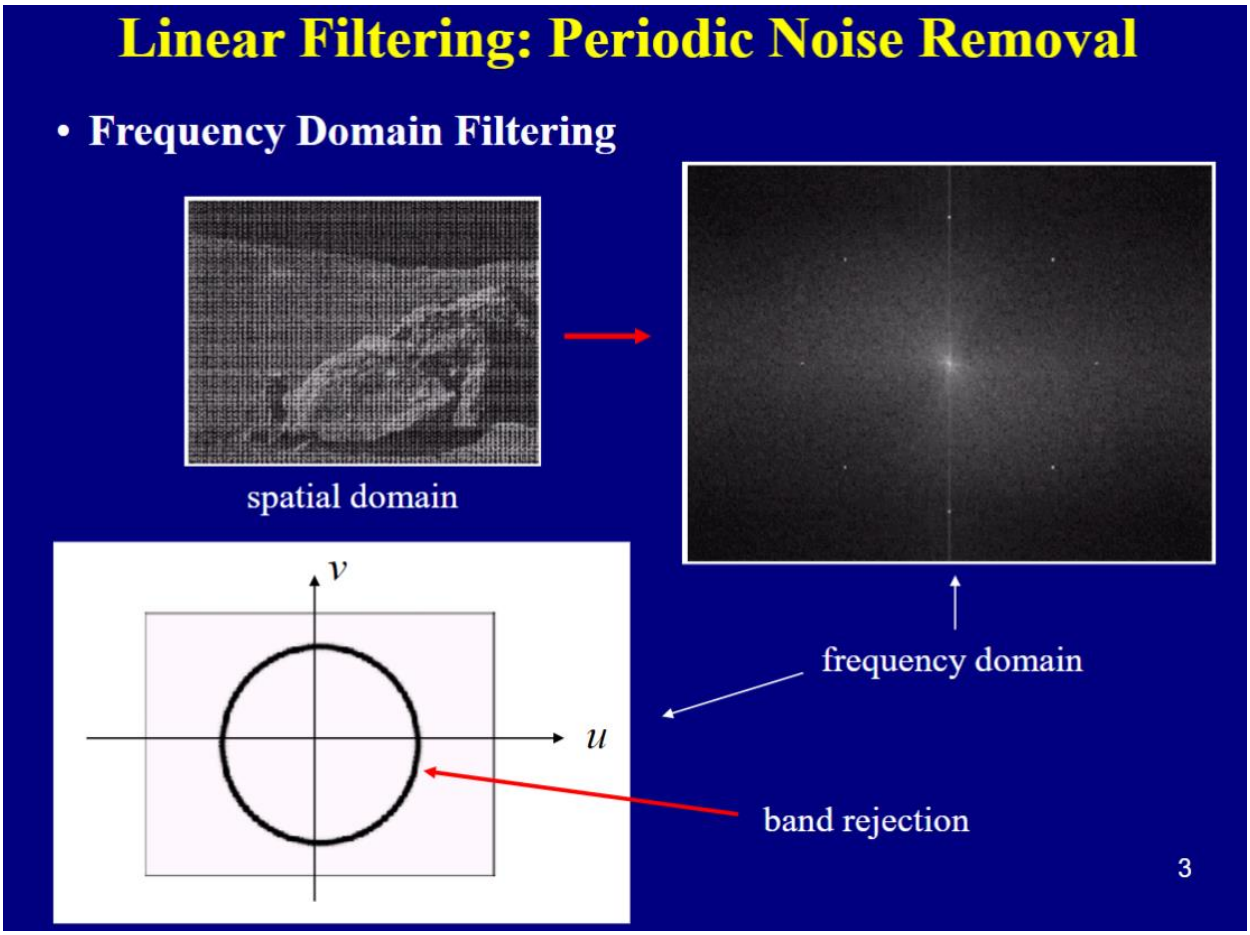
$$x(m,n) \otimes h(m,n) \underset{F^{-1}}{\overset{F}{\rightleftharpoons}} X(u,v)H(u,v)$$

$H(u, v)$: low-pass

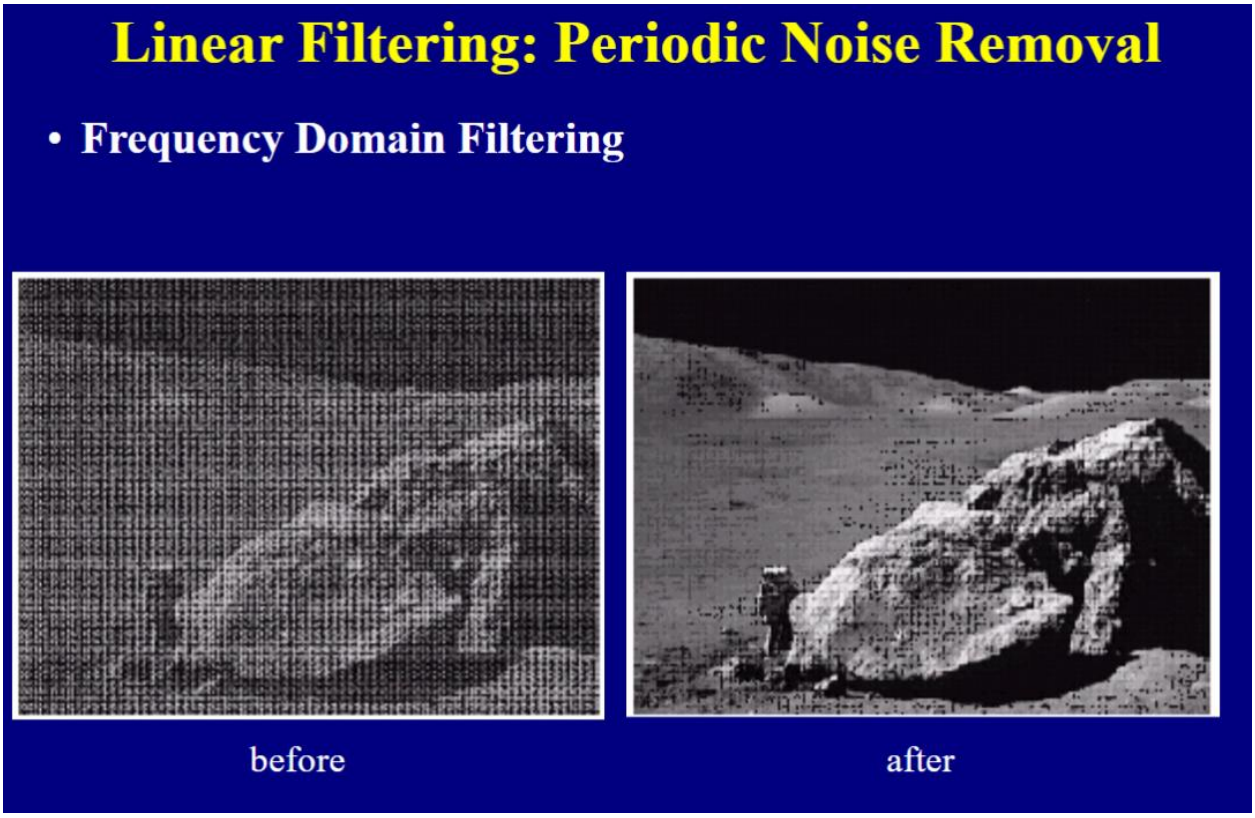
2

Suppose here in the spatial domain x is the input image, h is the impulse response then y, which is the ouput will be convolution of x and h as shown in the figure. And this convolved ouput is given input to the frequency domain, where the convolution becomes simply the dot product ot the multiplication. The convolution becomes multiplication based on the convolution Theorem. And this multiplication is the dot product between Fourier Transform of the impulse response and the fourier transform of the input image.

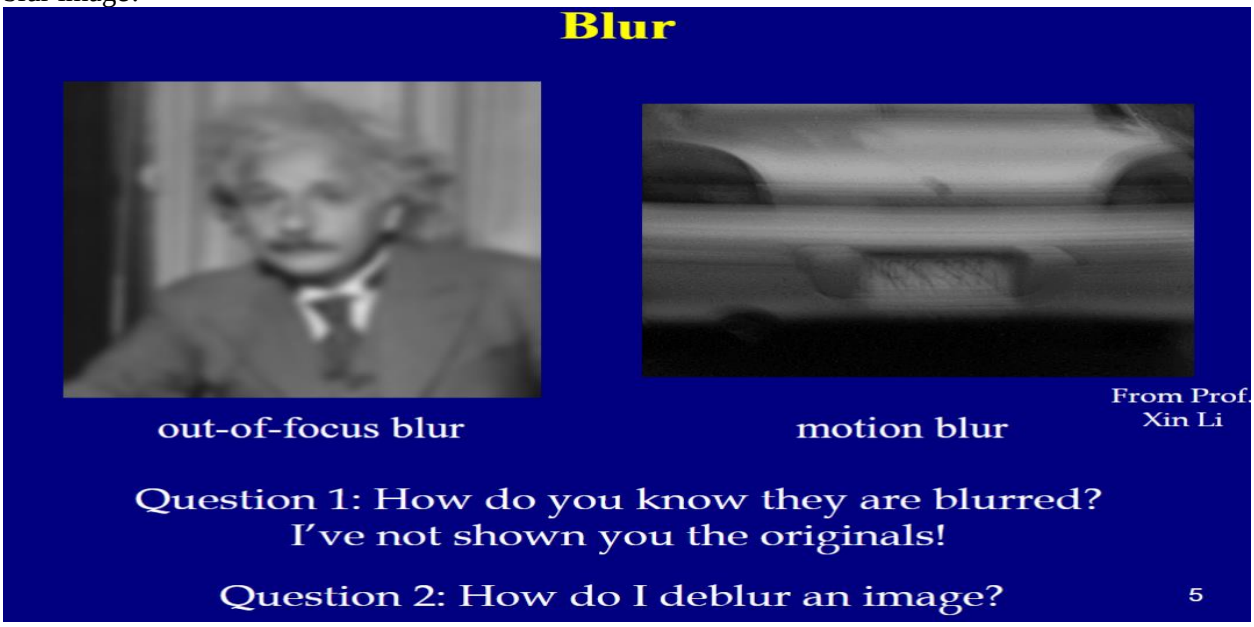
There is an equivalence filter in the spatial domain as well as in the frequency domain. This means that if we design a filter in the spatial domain then there exist a filter in the frequency domain which is similar to that of in the spatial domain and visa versa.
Question: Is there any advantage to do frequency domain filtering? Why not just design all the filters in the spatial domain?
Answer: The frequency domain representation of a signal sometimes give you more intuitive and this intuitive understanding is much easier to design filters in the frequency domain.



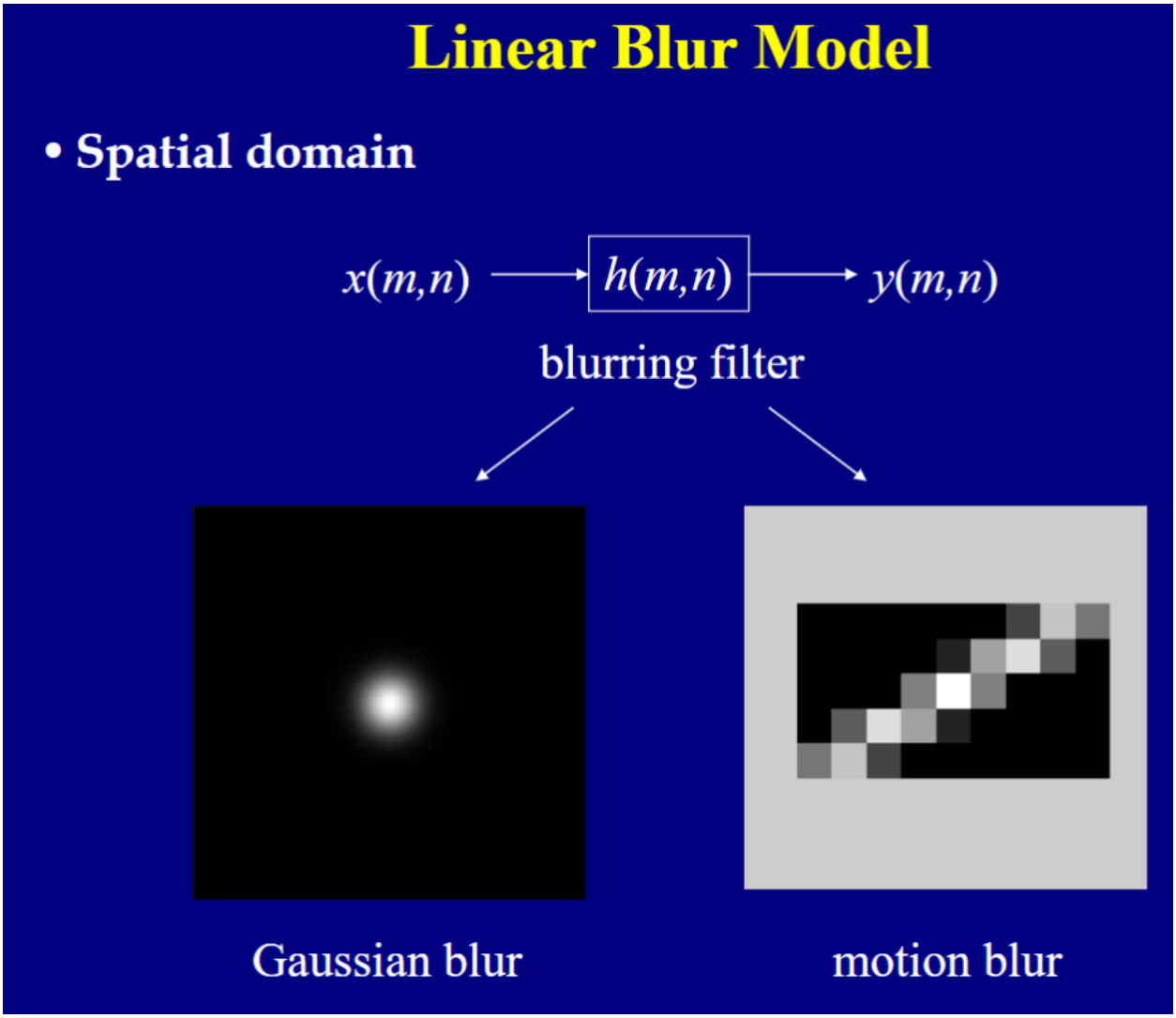
Here, we have given a picture in the spatial domain which consist of some noise and the noise is in the form of verticle and the horizontal patterns as shown. In spatial domain it is really hard to remove this patterns with any filter design. But if we apply 2D-DFT to the image in the spatial domain, we will get the image in the frequency domain as shown in the above picture. In the frequency domain the things become much cleaner. You can see the periodic dots of the image in the frequency domain. So these periodic dots are in circular manner and they are just the noisy patterns from the picture in the spatial domain. It is very difficult to remove this patterns in the spatial domain but in the frequency domain this is not the case. We only need to get rid of these dots in order to remove noise. So we have designed a ring shaped band rejection filter as shown. Here white means value 1 and black means value 0. So if we apply frequency domain filter (band rejection filter) on our image in the frequency domain then we can easily be able to remove the noise. It will be simple product between the fourier transform of the input image and the filter. So after the removal of these dots in the image in frequency domain we can apply the inverse of fourier transform and get our original image back into the spatial domain. The result is shown in the below image.



Another problem of the image restoration is that how are we going to recover the original image from the blur image?

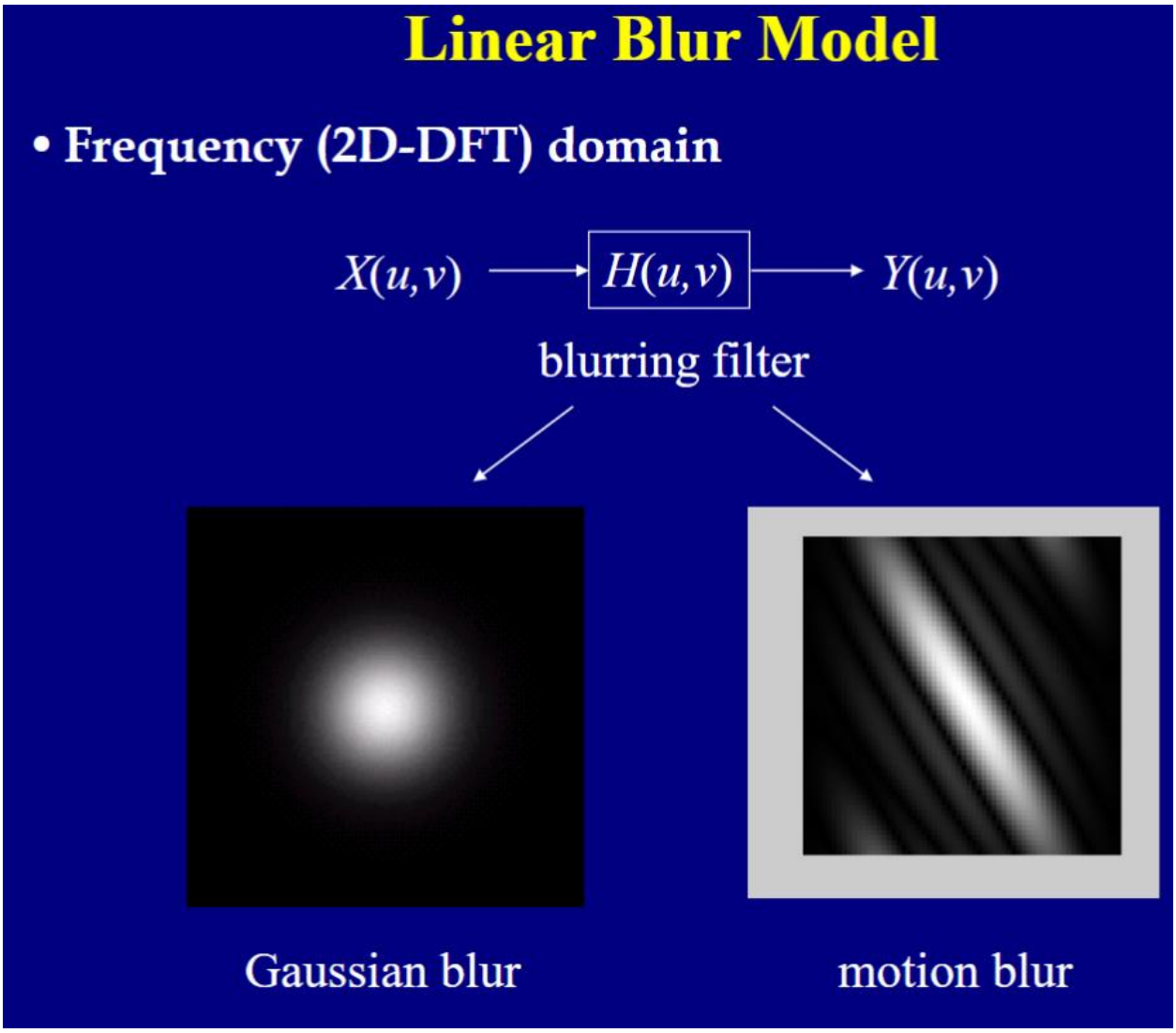


There are two kinds of blur. One is **picture blur** and the other is the **motion blur**. Both the terms are self explanatory from the above image. The real question is that how are we going to design an algorithm that deblur these images?

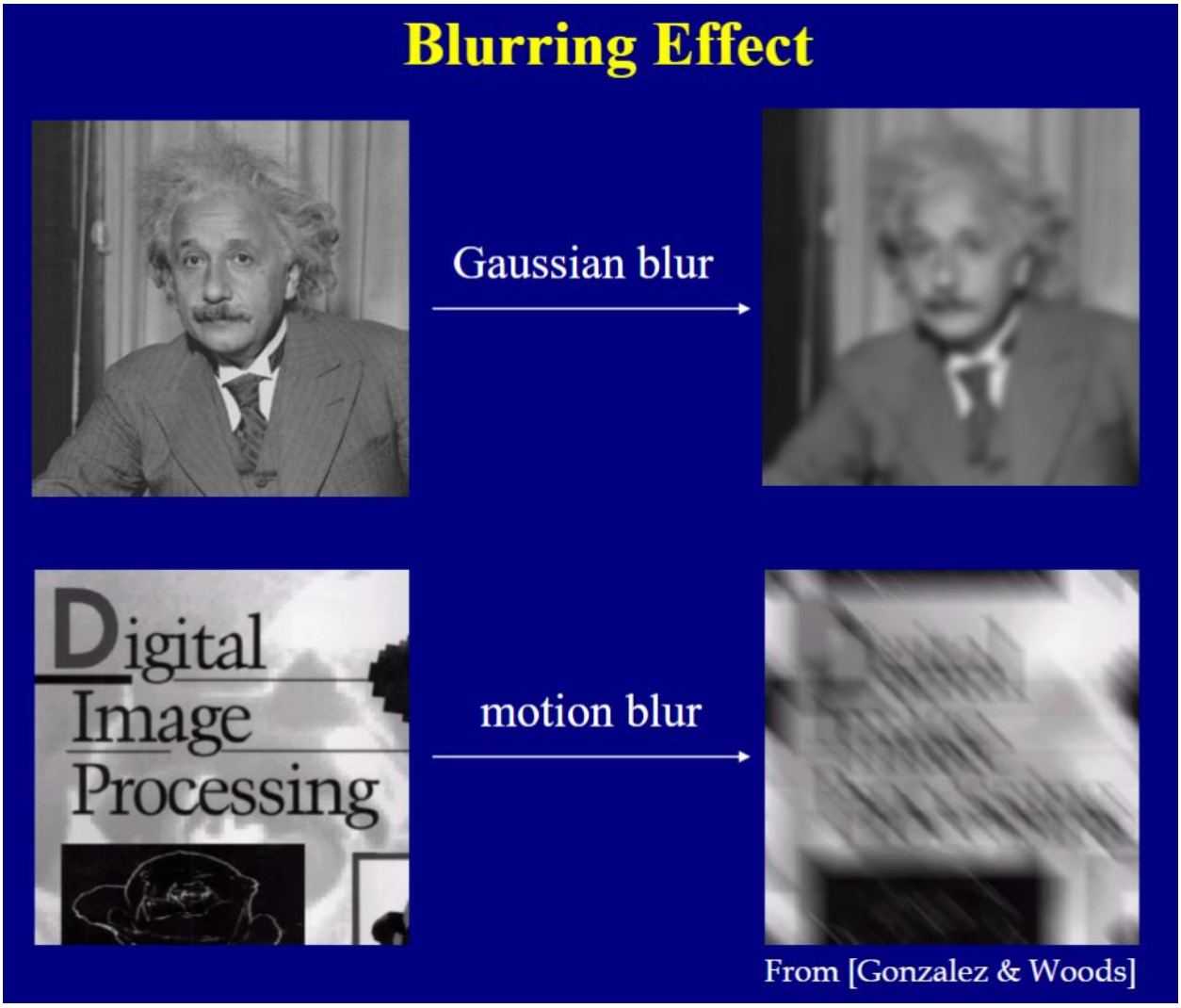


The first step to the above question is to create a model that can be used to describe these blurring process. The blurring process can be modeled by using the linear shift invarient system. From the above image suppose x is the original image and y is the blur image we observe. So here we can model different blurring process using kernel blurring filter as the impulse response of the linear shift invarient system. So in the first case we can say it is a gaussian blur and in the second case we have a motion blur. Both these gaussian blur and the motion blur are nothing but the filters applied to the original image. You can apply this blurring filters to a sharp image and after that you can observe the results whether they are blur or not.

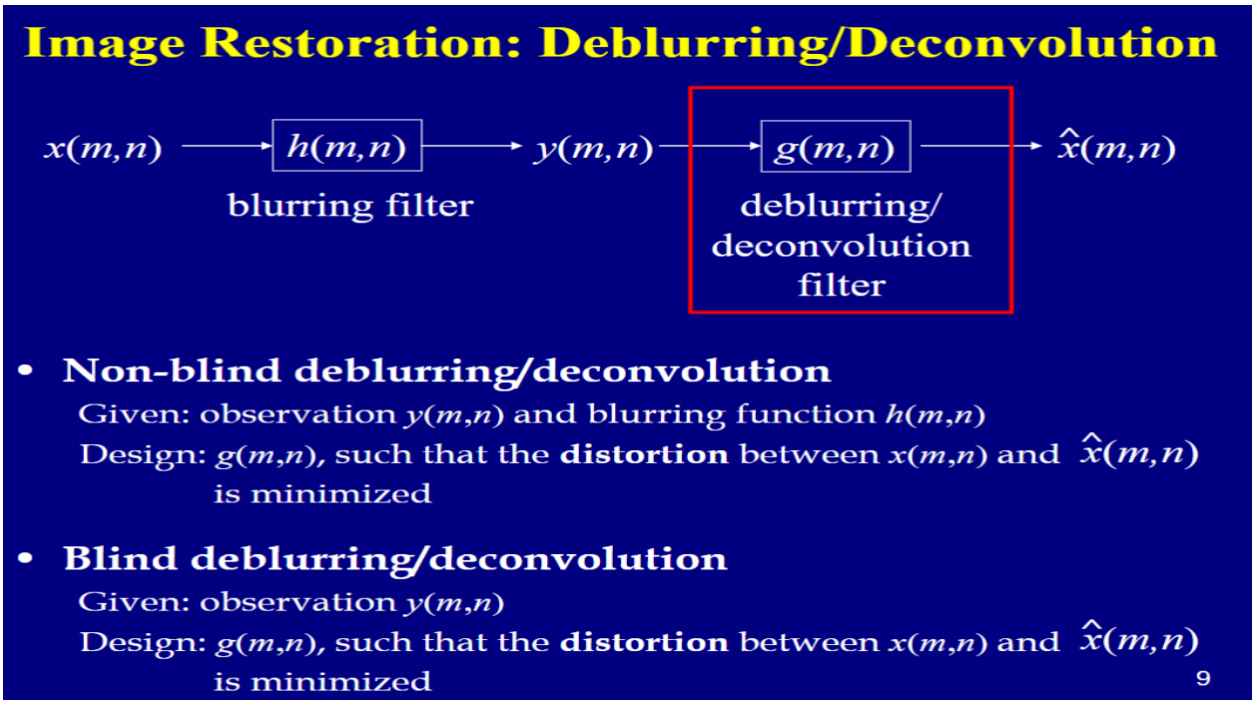
So now we only have the blur image and we need to find any deblurring filter which can be applied to the image y so that we can get an image which is similar to the original image. Here in the abive image the blur filter is in the spatial domain and in the below image the blur filter is in the drequency domain.



In the frequency domain also we can apply these filters similar to the spatial domain.
Below is the examples after applying the blur model to the sharp image.

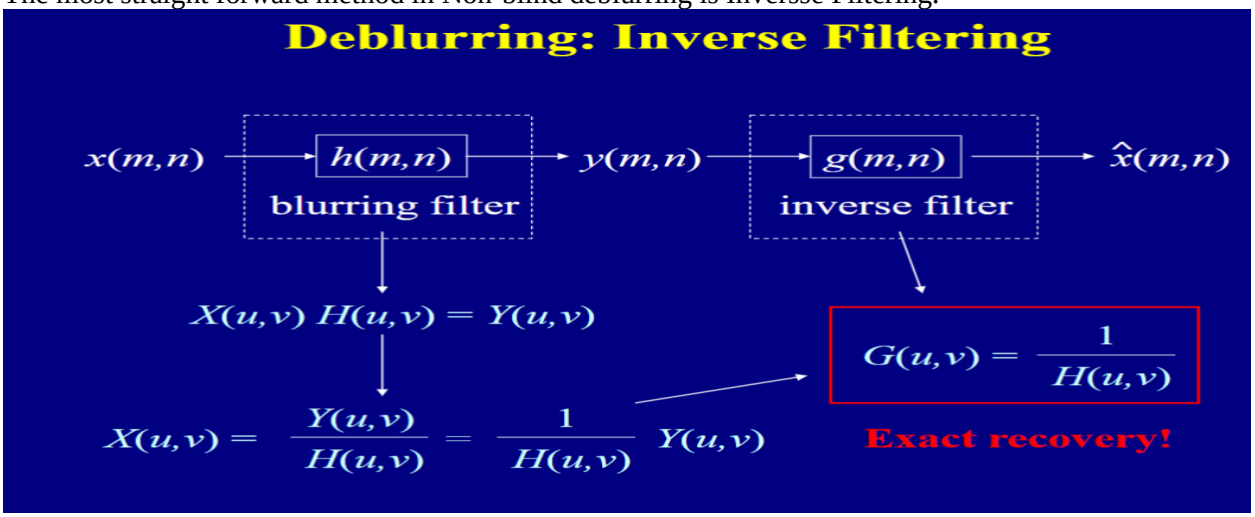


So now we have the blur model so we can now talk about the image Restoration model. Here, we basically need to find the debulrring filter. Deblurring is also called Deconvolution. There are basically two types of image deblurring filters. One is **Non-blind deblurring/deconvolution** and the other is **Blind Deblurring/Deconvolution**.



In Non-blind deblurring we have y and h. So the goal is given these 2 thing and we have to design g that can produce x hat which ios similar to x. In Blind deblurring we have given y and we have to design g such that the gap between x and x hat is minimized.

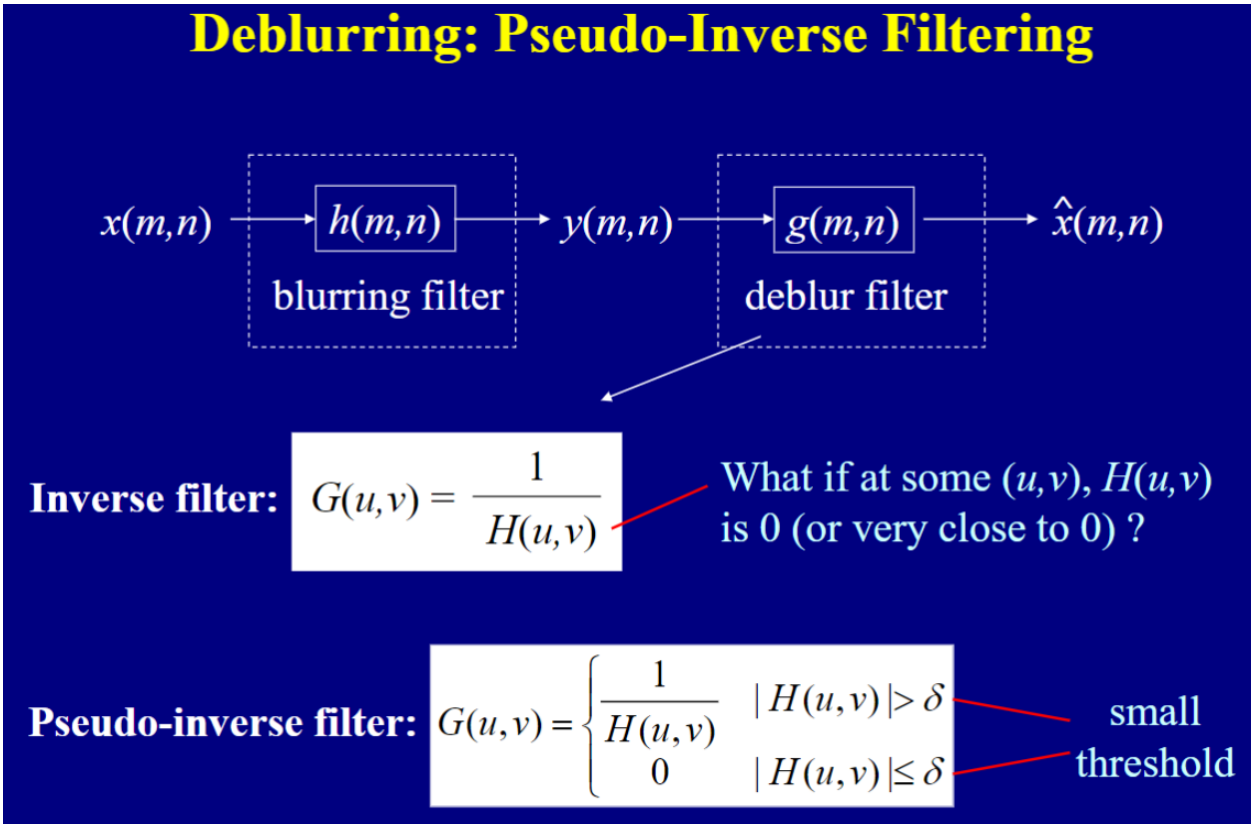
The most straight forward method in Non-blind deblurring is Inversse Filtering.



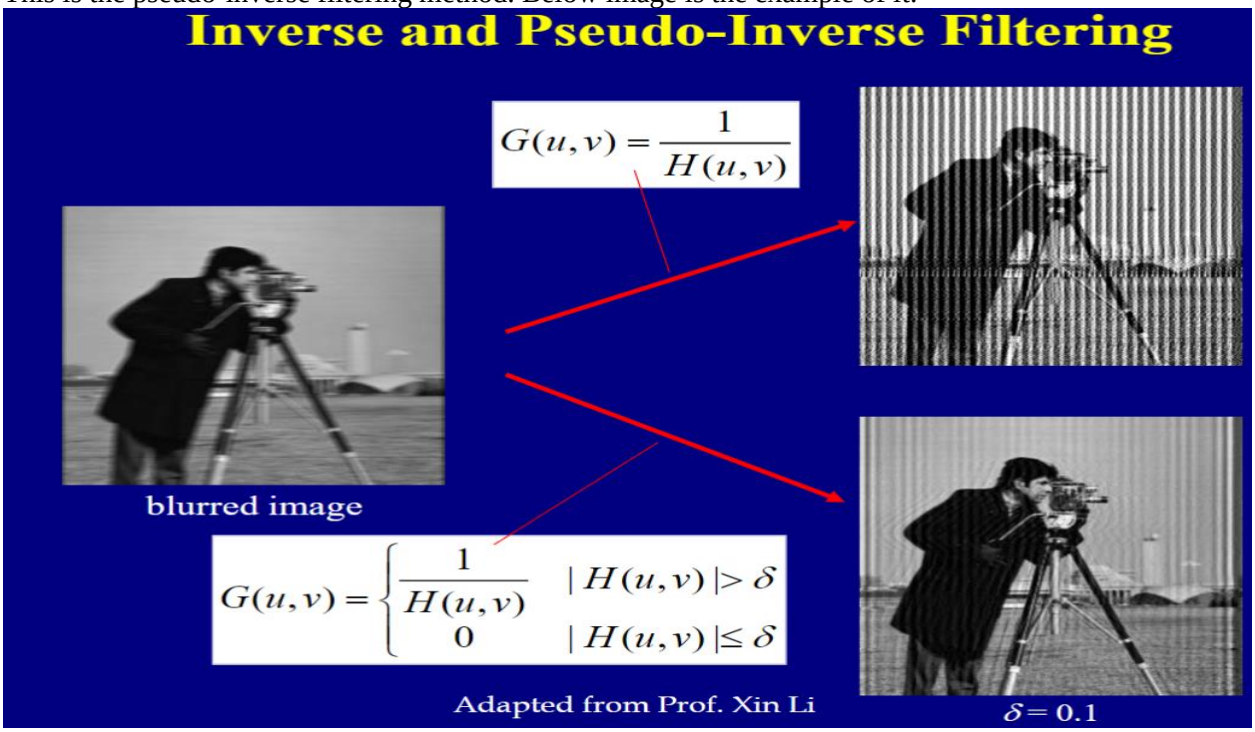
After applyingforiertransformtox we will get X, after applying fourier transform to h we will H, after applying fourier transform to y we get Y as shown in the above figures. In the frequency domain convolution becomes multiplication. Tha calculation in the above image is self explanatory. Here after obtaining G we are perfectly able to recover the x hat. This 1/H is known as the **inverse filter**.

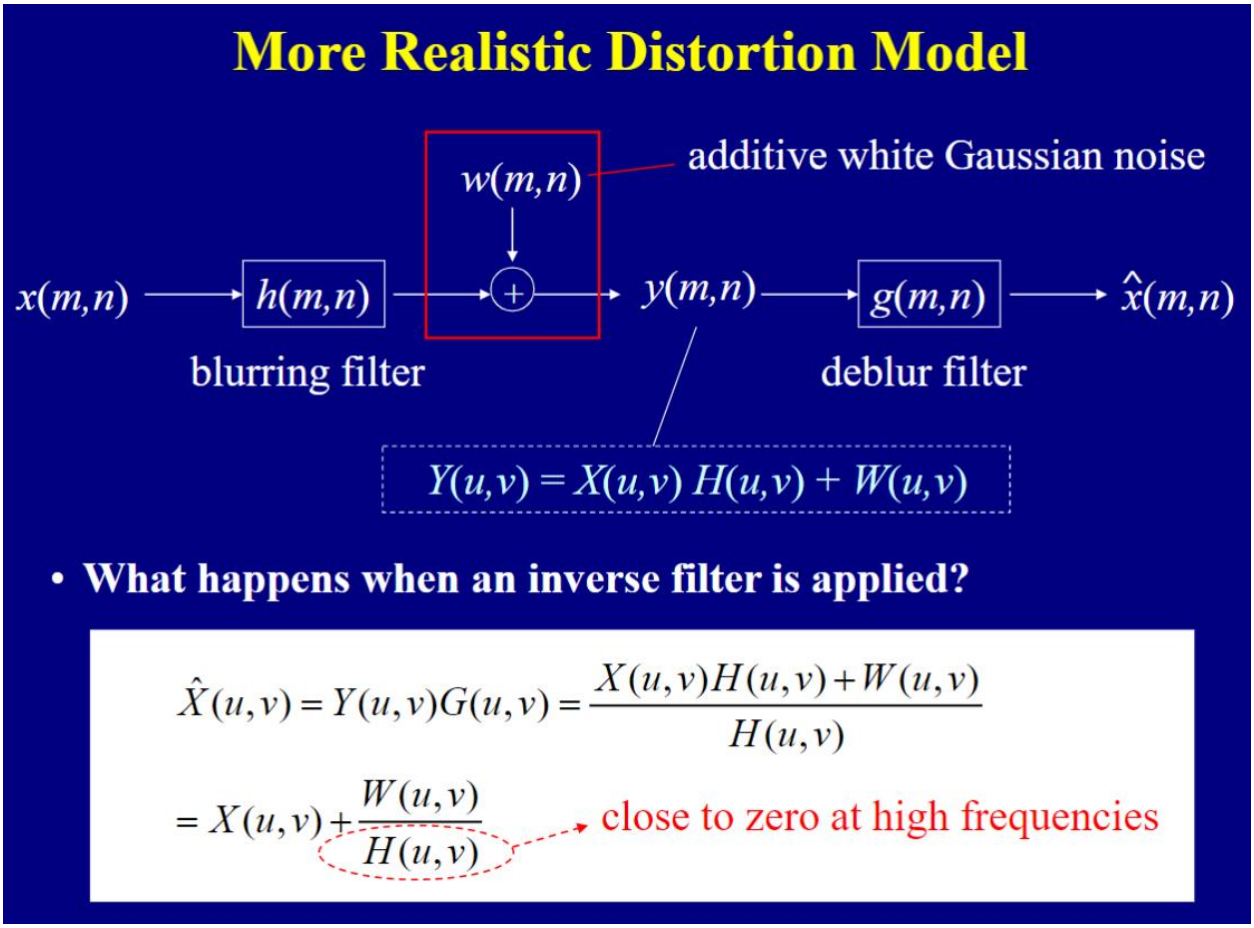
There are some problems with this inverse filter. What if at some (u,v), H(u,v) is 0 (or very close to 0)? That will not going to make us a valid filter. So we introduce the concept of pseudo-inverse filter. If the

magnitude of H(u,v) is greater than threshold (delta) then we use 1/H(u,v) else we will use 0 as shown in the below image.



This is the pseudo-inverse filtering method. Below image is the example of it.



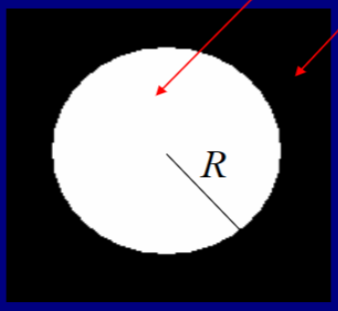


Here there is a problem of boosting noise. So to avoid the problem of boosting noise another approach is used which is called **Radially Limited Inverse Filtering**.

Radially Limited Inverse Filtering

Radially limited inverse filter:

$$G(u,v) = \begin{cases} \dfrac{1}{H(u,v)} & \sqrt{u^2+v^2} \le R \\ 0 & \sqrt{u^2+v^2} > R \end{cases}$$



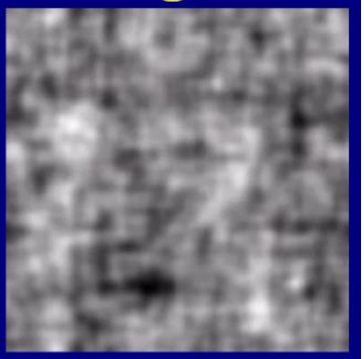
- **Motivation**
  - Energy of image signals is concentrated at low frequencies
  - Energy of noise uniformly is distributed over all frequencies
  - Inverse filtering of image signal dominated regions only

Radially Limited Inverse Filtering

Image size: 480x480

Original

Blurred

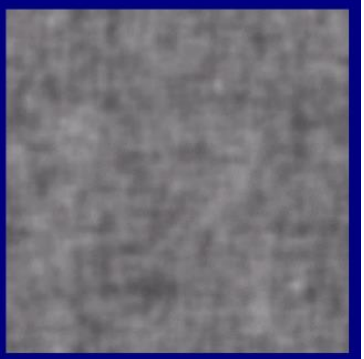


Inverse filtered

Radially limited inverse filtering:

$R = 40$

$R = 70$



$R = 85$

Wiener (Least Square) Filtering

Wiener filter: $$G(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K}$$

$$K = \frac{\sigma_W^2}{\sigma_X^2}$$

σ_W^2 → noise power

σ_X^2 → signal power

- Optimal in the least MSE sense, i.e. $G(u, v)$ is the best possible linear filter that minimizes

$$error\ energy = E\left\{\left|\hat{X}(u,v) - X(u,v)\right|^2\right\}$$

- Have to estimate signal and noise power

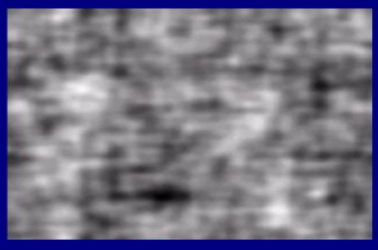
16

Weiner tells us that the error energy is given as the square magnitude of the difference between the restored image and the origonal image. So in order to minimize the error he gave a filter according to him as shown in the above image. Here in this case we have the error energy which is our cost function and we have to optimize it. This Weiner filter is similar to the inverse filtering. Here, H*(u,v) is the conjugate of H(u,v).

Wiener Filtering

Blurred image

Inverse filtering

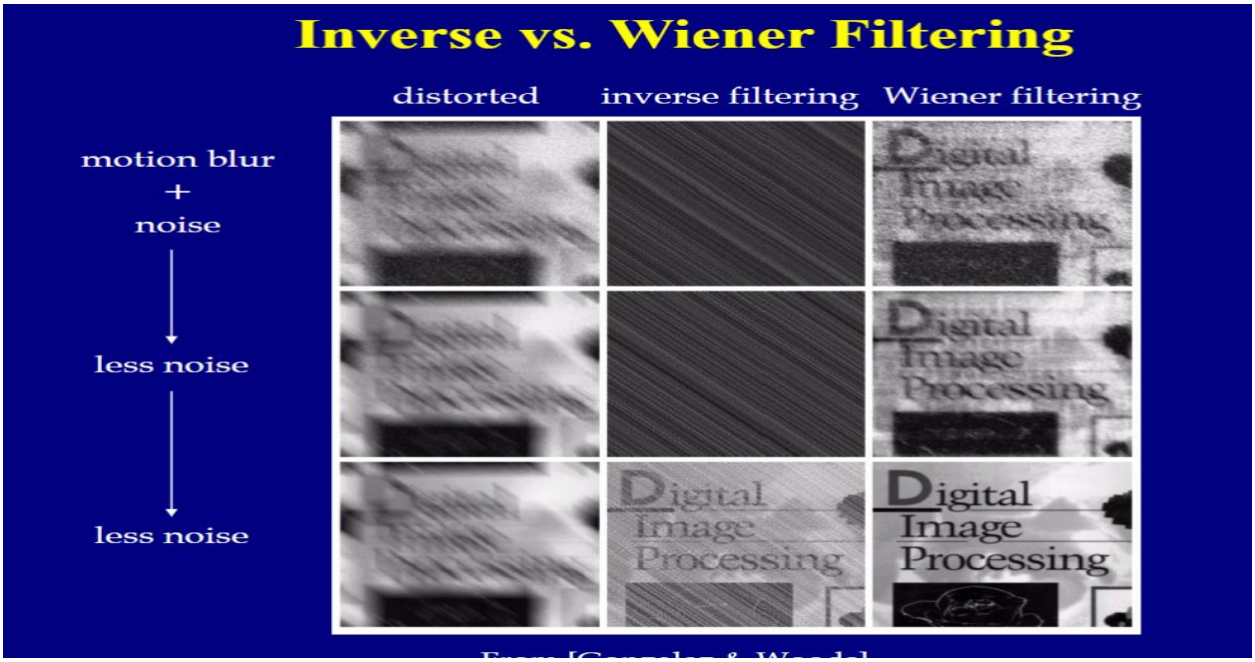


Radially limited inverse filtering $R = 70$

Wiener filtering

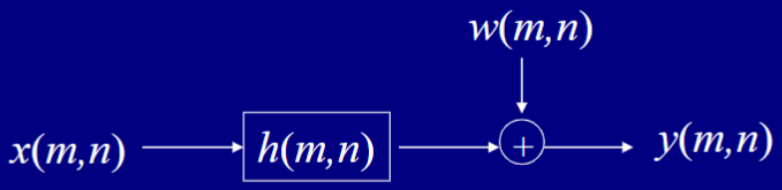
17

Here we are comparing the result of different filtering methods. The Weiner filtered image has way better quality than any other image filtering technique. We can get the best result in Weiner filtering.



Here are some more examples of Weiner filter comparision with the other filters. In this case we are actually adding different level of noise at each stage.
We can extend the implication of Weiner filtering to another problem of image denoising.

Wiener Image Denoising



- What if no blur, but only noise, i.e. $h(m,n)$ is an impulse or $H(u, v) = 1$?

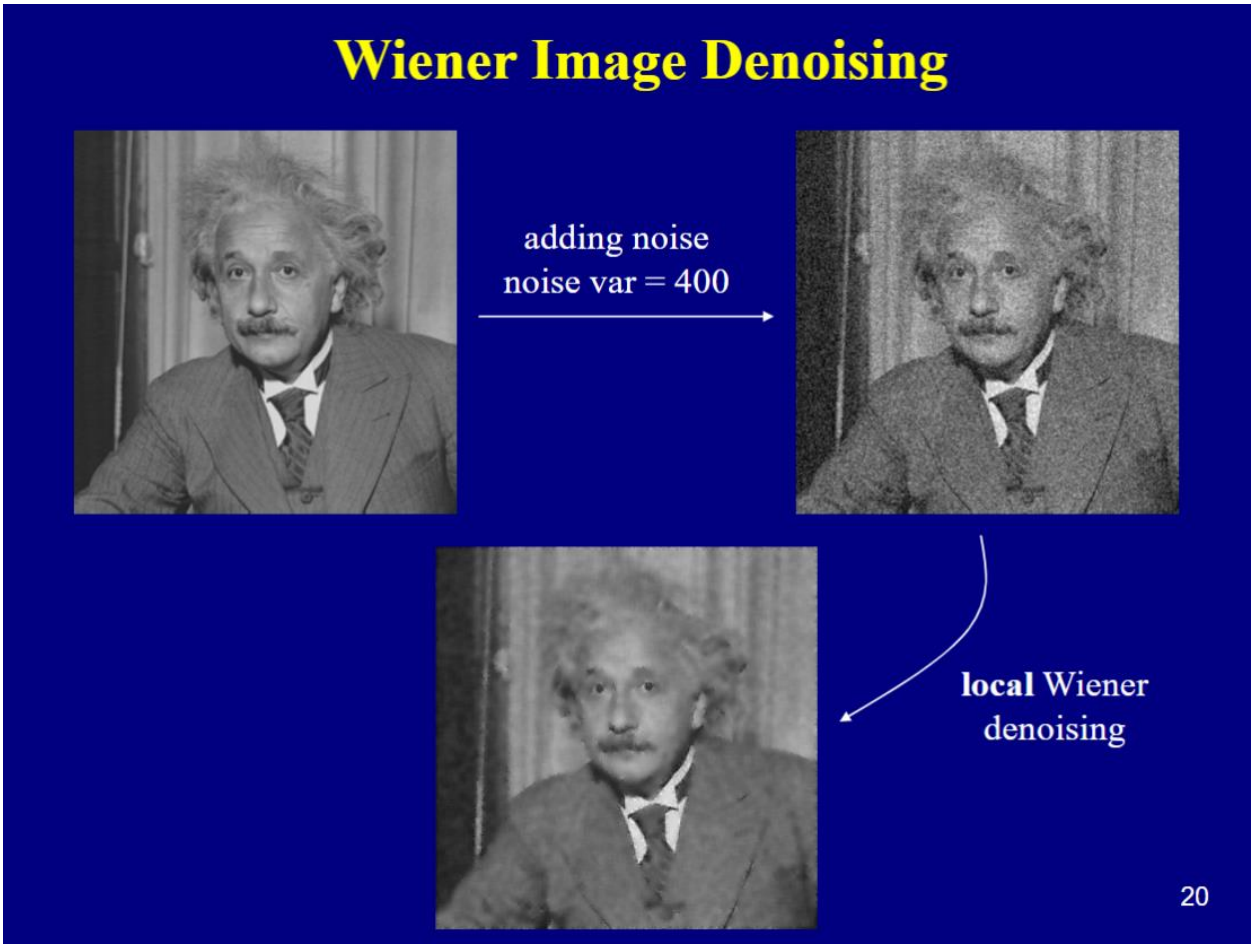
Wiener filter: $G(u,v) = \dfrac{H^*(u,v)}{|H(u,v)|^2 + K}$ where $K = \dfrac{\sigma_W^2}{\sigma_X^2}$

for $H(u,v) = 1$

Wiener denoising filter: $G(u,v) = \dfrac{1}{1+K} = \dfrac{1}{1+\sigma_W^2/\sigma_X^2} = \dfrac{\sigma_X^2}{\sigma_X^2+\sigma_W^2}$

19

Image Denoising is the special case of the image restoration problem. Here, w is the white noise added to the picture.x is the origianl image. H is the impulse response and y is the output image. Here what if we do not blur the image so in that case y will only be the noisy image and nothing else. And the restoration becomes how we can remove the noise from the image. In the case of denoising the blur function will became the function of 1 as shown in the image above. In the above image they have also shown the calculation for weiner denoising filter. So in the frequency domain the H and H* (conjugate) will become 1. Weiner filter is the best solution for image denoising.



Weiner image denoising example. If the final goal is to minimize mean squared error then we must use weiner filter.