Biometrics

Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

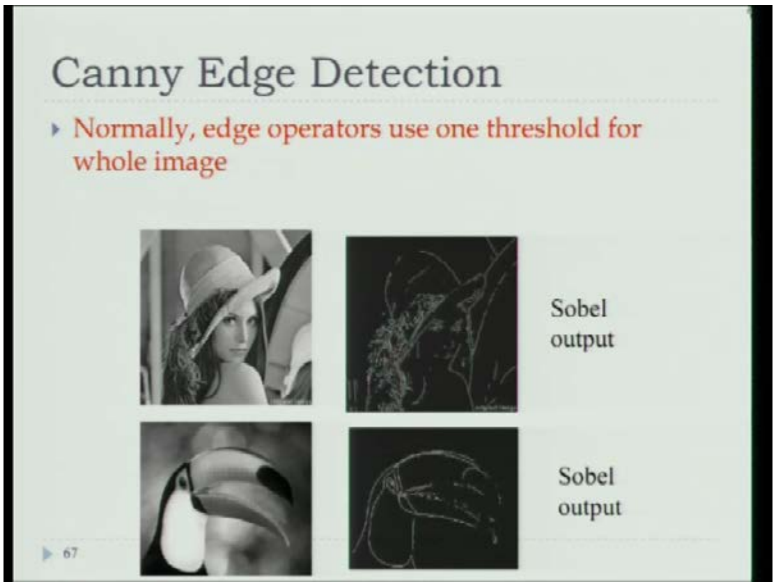
Lecture No. # 07

Canny Edge Detection

Say up to which process will be in the syllabus for whatever I will be covering till may be another two lectures that will come. And then whatever we will be covering I will start on biometrics that will come final final exam. In the last class, I was discussing about the edge detection, and we discussed different types of edge detection algorithm, one of them was Prewitt, Sobel, Roberts, and also we mentioned that there are certain issues in front of you while writing the edge detection algorithms.

The problem is that you have to decide a threshold in such a way, because you will be using the gradient, magnitude and direction. But sometimes what happens due to some reasons that your threshold is not catching or you will not be able to catch the edge. Because, you know if you if you make the threshold very less then, you may find that too many false edges are coming. But if you make the threshold very high then, there are certain genuine edge you will be missing. So, because of various reasons, you may find that some part of the edges are very weak, and your threshold is not able to detect those edges. So, you have to be very careful while detecting the threshold that is the issue.

(Refer Slide Time: 02:10)



Canny Edge Detection

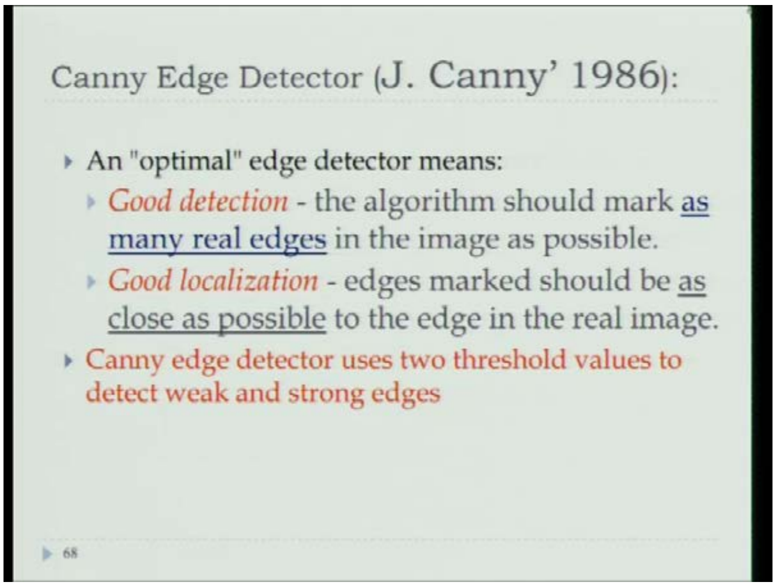
- Normally, edge operators use one threshold for whole image

Sobel output
Sobel output

Now, that edge detector whatever you are using they are using only one threshold value. And, if it is exceeding that value gradient magnitude along that direction if you find it is exceeding the threshold value you tell it is a edge point otherwise it is not. Then set of those points which are continuous, you will tell that this is an edge or true edge. So, you observe I do not know whether it is visible. See in this this is a Sobel operators output and you will find there are you will find that there are several edges here, which are not connected which are not connected, just it has broken there is no links between the two edge.

Say for example, this is the this is the this is I think this is visible for you see there is this is not connected. But it should have been connected this is an edge it should have been connected, because of your threshold proper improper selection of threshold this has come out like this.

(Refer Slide Time: 03:16)



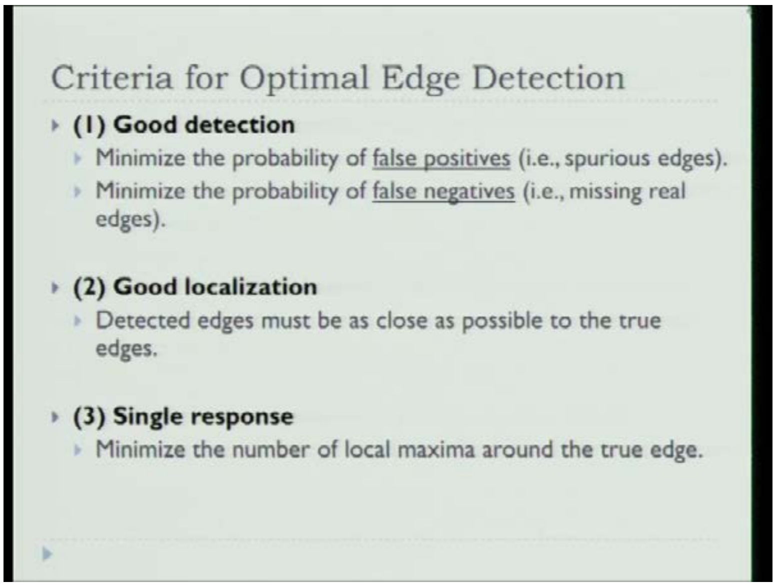
Canny Edge Detector (J. Canny' 1986):

- An "optimal" edge detector means:
  - *Good detection* - the algorithm should mark as many real edges in the image as possible.
  - *Good localization* - edges marked should be as close as possible to the edge in the real image.
- Canny edge detector uses two threshold values to detect weak and strong edges

68

Now once you design an edge detection algorithm, what are the things you are looking for? First one: you are looking that you your algorithm must be able to detect the true edges as many as possible. This is the first condition. So, good detection the basic one we are able to detect all the true edges as far as possible. Second one is that good localization that whatever way you are selecting the edge it should be nearer to the true edge.

(Refer Slide Time: 03:50)



Criteria for Optimal Edge Detection

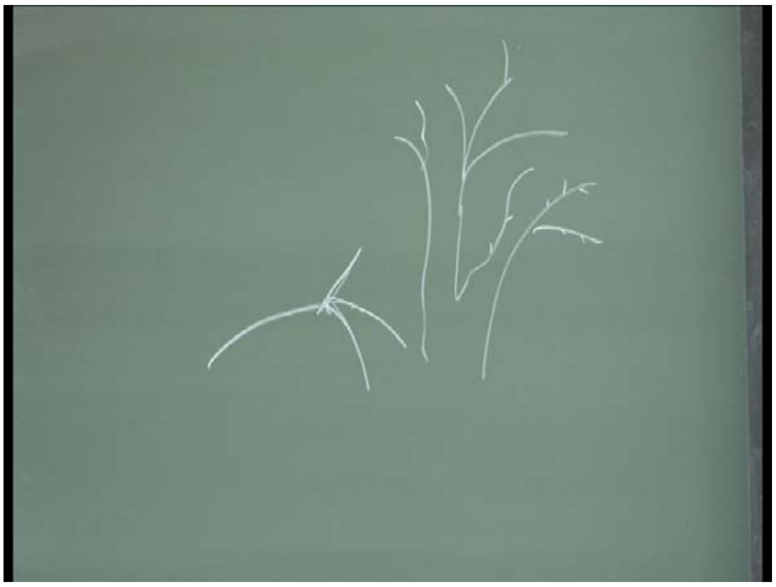
- **(1) Good detection**
  - Minimize the probability of false positives (i.e., spurious edges).
  - Minimize the probability of false negatives (i.e., missing real edges).
- **(2) Good localization**
  - Detected edges must be as close as possible to the true edges.
- **(3) Single response**
  - Minimize the number of local maxima around the true edge.

Because edge is like this and they are true edge is that way it should be nearer to that as far as possible. These are the two conditions you should look into while designing an edge detection algorithm. Now, by good detection what we mean? We mean that you are minimizing the probability of false positive that if whatever is the noise it should not look like an edge or you should you should not consider it is an edge. So, false positive should be as minimum as possible and the next one is that minimize the probability of false negative.

That means that you should not tell a true edge as not an edge. So, your threshold value should be such that, this should be taken into account. Now remember one thing that they are inversely related. That if you want to make a very high threshold, yes you will not be able to get the false edges because, you have given a very high value of threshold you will not be able to get the false edges. But at the same time sum of the true edges also will be missing sum of the true edges also will be missing or if you give very low threshold very low threshold then what happens? Then you will be getting all edges or say looks like an edge all edges you will be considering in your system. And, as a result some of the false or some of the noises also you will be considering as edge and it will create a problem.

And one thing you remember that, it is very difficult to get a threshold such a threshold which satisfy both of them. So, this is the important thing, this will create the two types of error also. We will discuss while we will be covering the biometric system that what are the different types of error and how we should minimize it? Good localization that means the detected edges must be as close as possible to the true edges and another thing is their single response. What it means? That if this is a pre edge point from there there is a direction you obviously you will be getting the gradient direction and gradient magnitude and towards that direction only you should get another local maxima. You should not get too many local maxima.

(Refer Slide Time: 06:51)

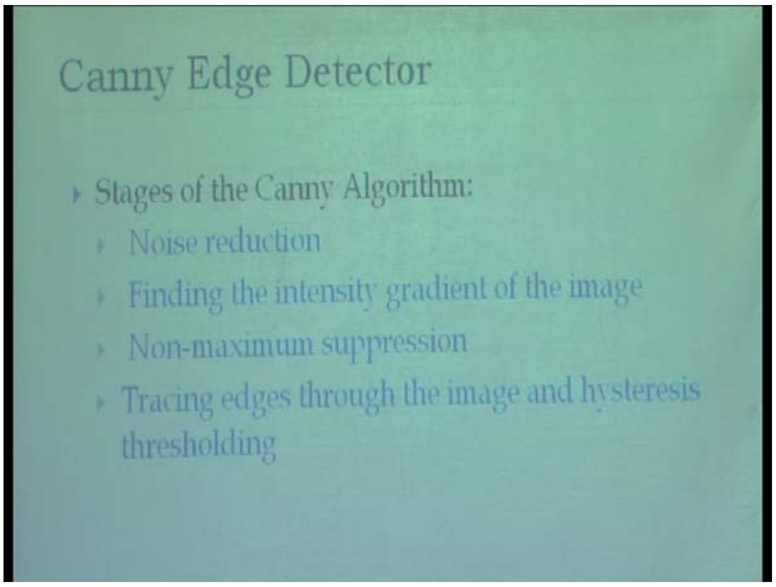


Suppose this is an edge, now while I am looking for whether there is another whether there is another neighboring pixel is there edge point or not, it should not give me several edge points, several local maxima. Because, if it gives you the several local maxima, so what will happen? It will create an edge this way, it will create an edge this way, this way and so on and most probably generally does not happen. Because, we assume the image is such that, between the two neighboring pixel distance is not too much. So, symmetry is maintained.

So, your threshold should be such that these types of scenarios does not occur. And, if it occurs there should be a way out how to remove some of these false edges. For example, what we get? If say vein pattern. Because, of if i take the vein pattern photographs and you may scan image you have that veins veins it will look like like this. It may look like this; this is your vein pattern here. Now, because you are here, even though we are using the infrared camera because of the here you may find like this type of edges you may find. Now but these are not edges you it is a very easily everyone would tell this is edges these are all false edges.

So there is a thing what we are looking for that your threshold should be such that, there are not too many local maxima around the true edge. What we have done in our case? We have assumed that if it is this length of this edge is more than some number then only you consider it is an edge, otherwise it is not. Now, Canny Edge Detector uses the true thresholds to detect the weak edge and the strong edge. That one is genuine edge by if you put the higher threshold they are genuine edge. If anything any gradient magnitude is greater than that threshold it is an edge point. Another one that I have put a lower threshold and we checked it is probably end with edge point but we are not able to take the decision at this moment.

(Refer Slide Time: 09:48)

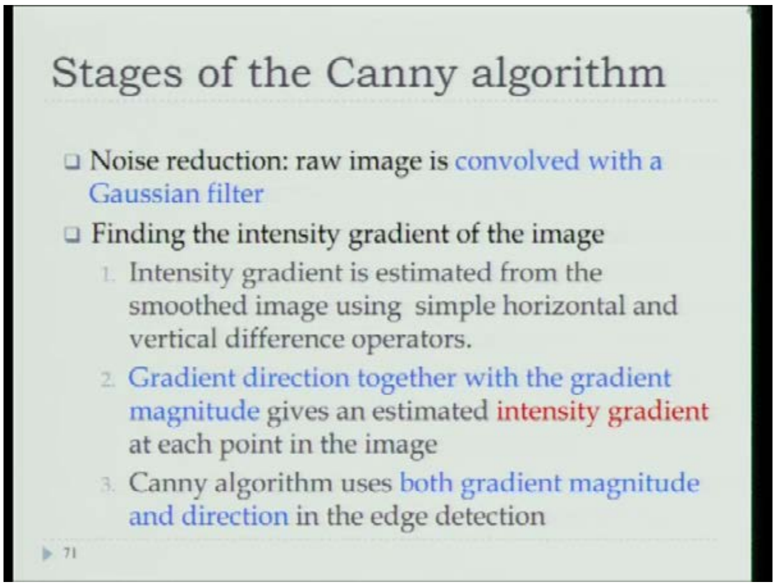


Canny Edge Detector

- Stages of the Canny Algorithm:
  - Noise reduction
  - Finding the intensity gradient of the image
  - Non-maximum suppression
  - Tracing edges through the image and hysteresis thresholding

And if it is less than the lower threshold then it cannot be an cannot be an edge point. Now there are four stages of Canny's algorithm. Because, any algorithm for edge detection we need to first remove the noise, so same thing here also we have removed the noise. Second is, then you need to compute like Sobel and others you need to know what is the intensity gradient of each pixel points? And, third step is the non maximum suppression. That is the thing that I am telling that this part non maximum suppression and once you have suppressed the non maximum points, then use the image and hysterisis thresholding to determine the edges. Now, hysterisis thresholding is the major part what Canny has designed.

(Refer Slide Time: 10:57)



Stages of the Canny algorithm

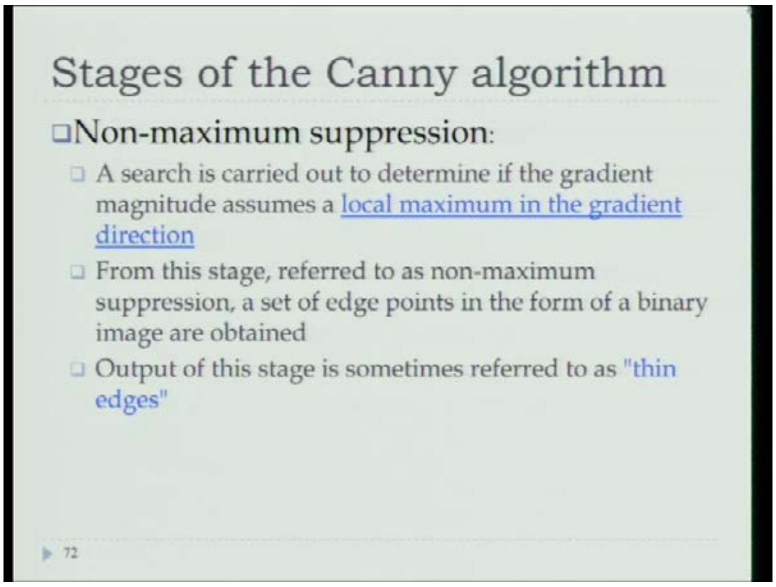
- Noise reduction: raw image is convolved with a Gaussian filter
- Finding the intensity gradient of the image
  1. Intensity gradient is estimated from the smoothed image using simple horizontal and vertical difference operators.
  2. Gradient direction together with the gradient magnitude gives an estimated intensity gradient at each point in the image
  3. Canny algorithm uses both gradient magnitude and direction in the edge detection

71

Now, noise removal algorithm here he has suggested that you use that Gaussian filtering algorithm. So, raw image you have just convolved with a Gaussian filter to get the removed noise removed image; now that image is used to estimate to get the estimated intensity gradient of that image. Now you would have other intensity gradient, we can use the Sobel operator, to get one is horizontal direction another one is vertical direction.

And, using the horizontal and direction vertical direction, you can get the gradient magnitude and also you can get the direction, am I right? Because you remember the direction those things that you have already converted tan inverse of y by x same thing. So you know now gradient direction and also you know the magnitude these two forms your estimated intensity gradient. And, Canny has used both of them to determine whether a point is an edge point or not.

(Refer Slide Time: 12:12)



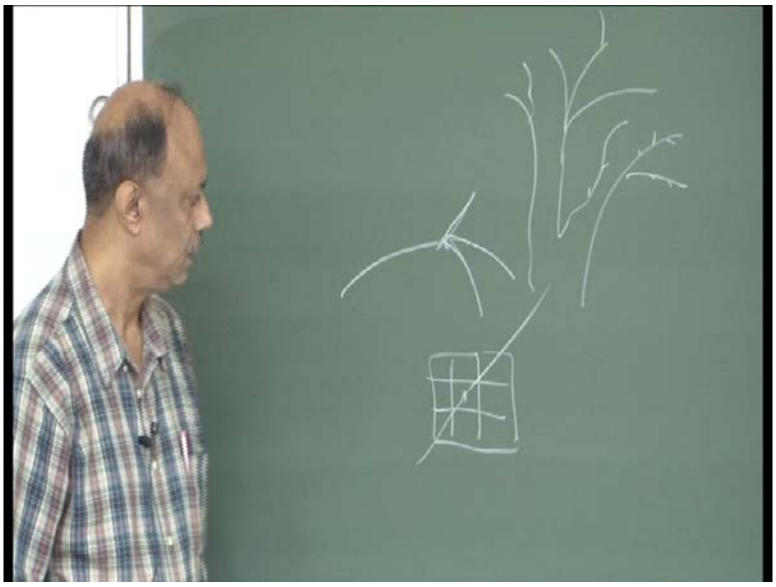
Stages of the Canny algorithm

- Non-maximum suppression:
  - A search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction
  - From this stage, referred to as non-maximum suppression, a set of edge points in the form of a binary image are obtained
  - Output of this stage is sometimes referred to as "thin edges"

72

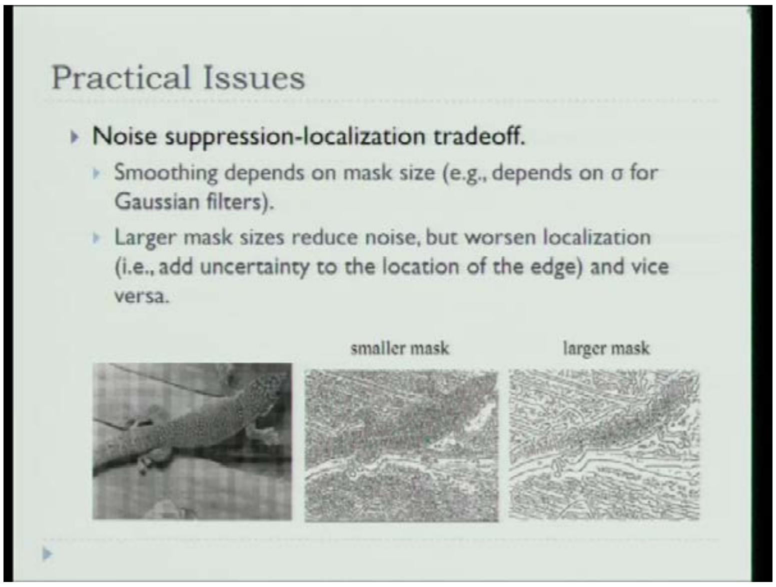
So, you have now the gradient magnitude and the gradient direction. So, non maximum suppression what it does that he looks for a pixel for each pixel whether that gradient magnitude gives you the local maxima along the direction on not.

(Refer Slide Time: 12:38)



Say0 you have a pixel point, what it looks? And you know that direction gradient direction say this way. So he looks whether this point is a local maxima or not. That is the thing only he is looking, if it is then he retains, otherwise he is throwing that no it is not an edge point that is the thing he suppressed if it is not a local maxima and this obviously since he is suppressing he will get some thin edges. But it is not at zero one value it is a thin edges with some gradient magnitude values.

(Refer Slide Time: 13:24)



Practical Issues

- **Noise suppression-localization tradeoff.**
  - Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
  - Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.

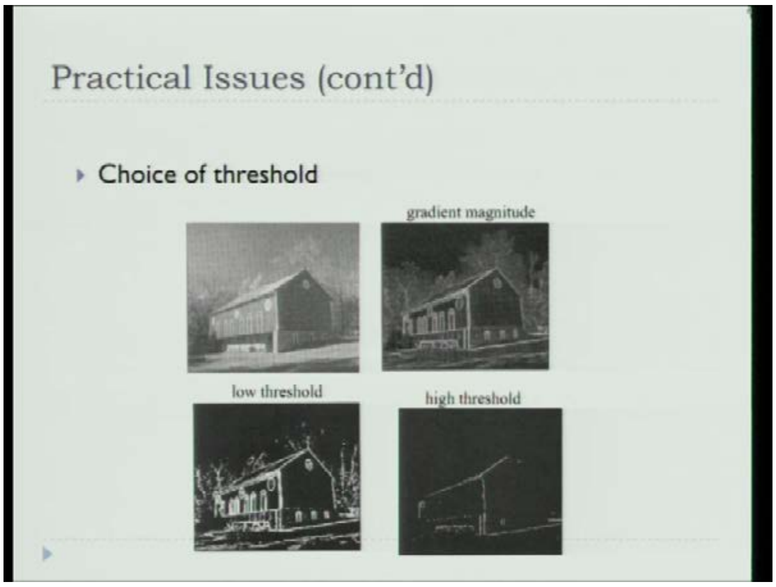
smaller mask
larger mask

We will show you how to compute later on. Now, see the issue is first problem we are facing the noise suppression by your filtering technique. Now, you remember in the case of Gaussian filtering technique, sigma plays a major role. That you remember that larger value of sigma it becomes the main filtering. But if you put larger value of sigma or your mass size will be increased, once you are increasing the mass size you are reducing the noise. No problem, because it is coming to the smoothing effect so, you will reduce the noise there is no problem.

The problem will come out that localization edge localization problem. That it will worsen the edge localization. Say for this example if you see, this is the very smaller mass you have taken and it is looking the same image with edges only. But too many false things are also there but if I use a larger mass it has suppresses several things.

And some of the edge lines see these edge lines they have been not in a perfect way, perfect position. it has been deviated. So, localization will be a problem so, you have to select your sigma in such a way it should not be small at the same time it should not be very large.

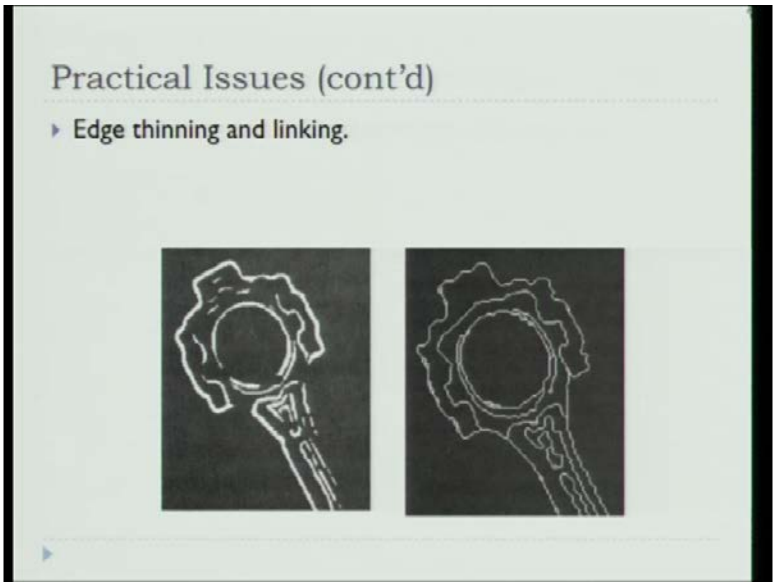
(Refer Slide Time: 15:06)


Practical Issues (cont'd)

- Choice of threshold

This is the effect of low threshold and high threshold. If you put the low threshold, you see some of the noise they are visible. You are not getting exactly whole edge proper edges but if you put the high threshold only the boundary you can see the rest you cannot see.

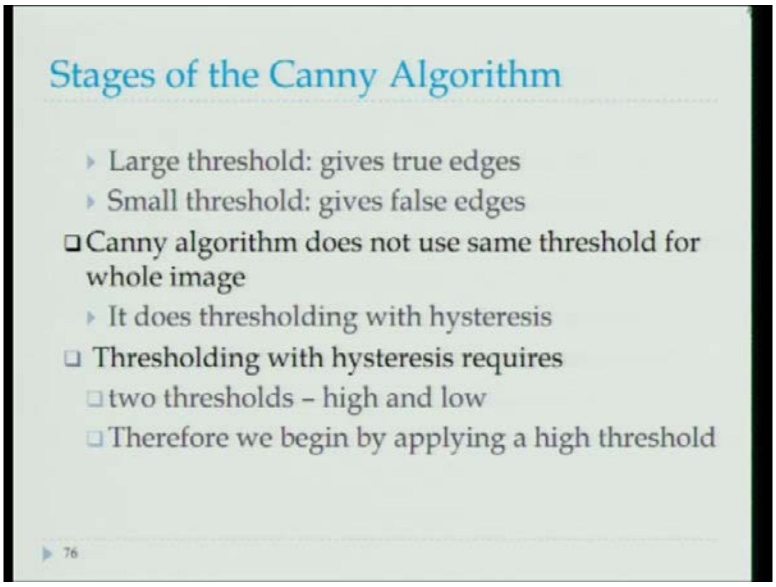
(Refer Slide Time: 15:29)


Practical Issues (cont'd)

- Edge thinning and linking.

The next example is that this is the true edge detection algorithm you have got. This image but you see they are not linked. They should be so you need the linking operator to link the edges, broken edges.

(Refer Slide Time: 15:45)



Stages of the Canny Algorithm

- Large threshold: gives true edges
- Small threshold: gives false edges

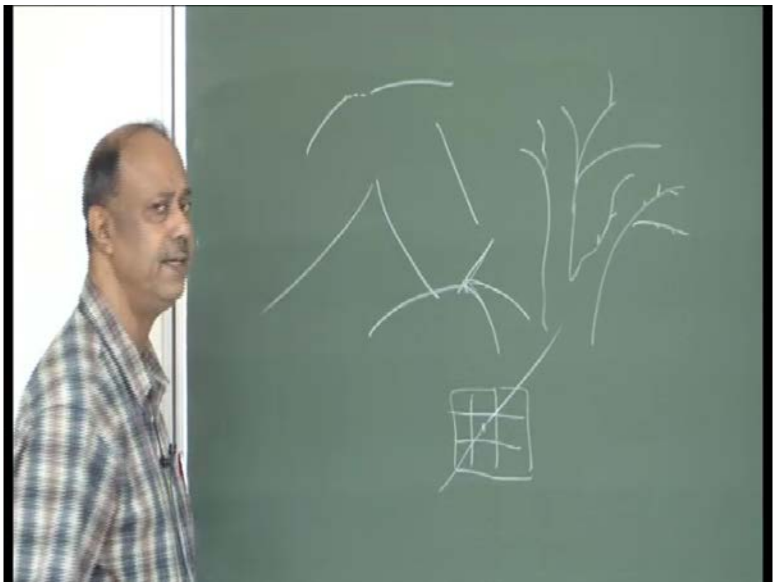
❑ Canny algorithm does not use same threshold for whole image
- It does thresholding with hysteresis

❑ Thresholding with hysteresis requires
- ❑ two thresholds – high and low
- ❑ Therefore we begin by applying a high threshold

76

Now, we have early mentioned that if you have last threshold you will be getting the true edges. And, if you take the small threshold value then, it will give some false edges also. So, these none of them are suitable if you use only one threshold value will not give you the correct edge map. So what Canny is suggesting that Canny has suggested that you use two thresholds. One is low threshold another is high threshold. Now, what happens? First you use high threshold. You will be getting the true edges these are genuine edges, but as you may get like this.

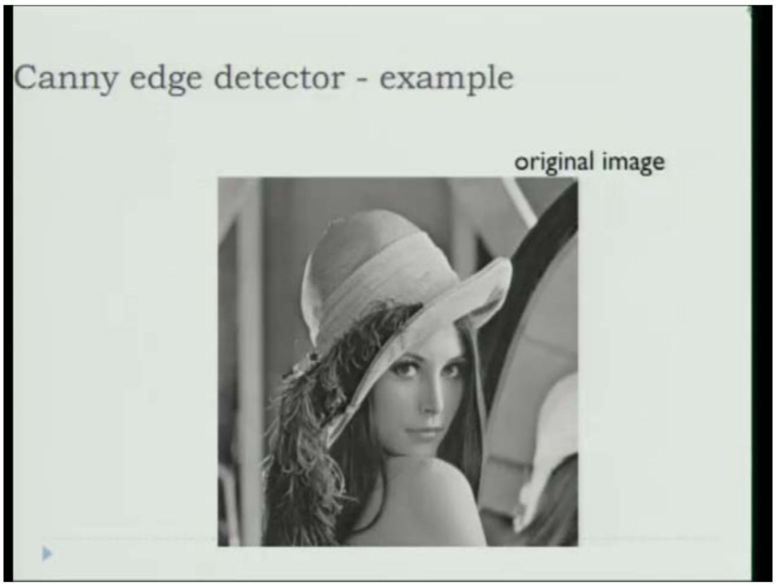
(Refer Slide Time: 16:31)



You will get this type of edges. But they will not be connected. Of course see now you traverse that edge, you make use of control of this edge. And, you use now here the lower threshold based edge map and you look for whether there exists any any weak edge towards the direction towards this direction. So, what I am looking that in this direction in this direction whether there exists any weak edge or not? If there exists some weak edge you consider that weak age but you remember that you have to follow the direction.

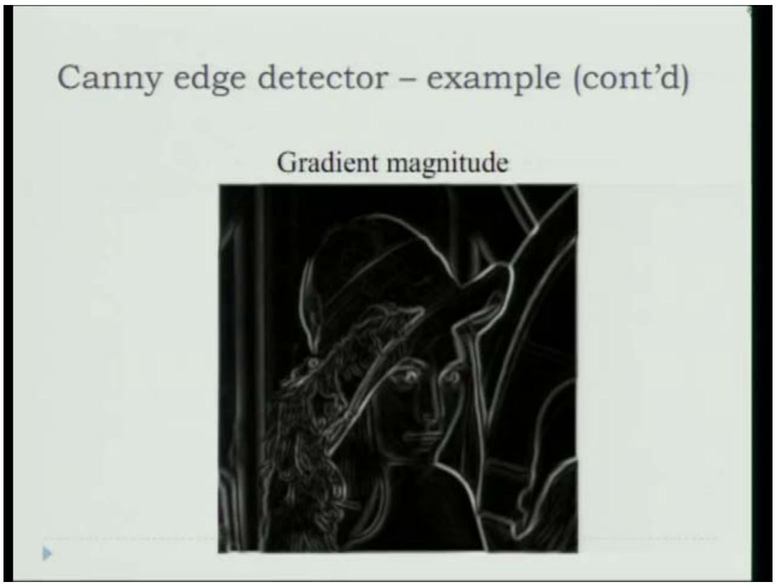
In that direction whether there exist any weak edge or not? Weak edge you will be getting through the through the lower threshold one small threshold value. And, if there exist some weak edge towards that direction, you may consider this is an edge. By that process you will be able to generate the good edge map.

(Refer Slide Time: 17:51)


Canny edge detector - example

So this is an original image.

(Refer Slide Time: 17:57)

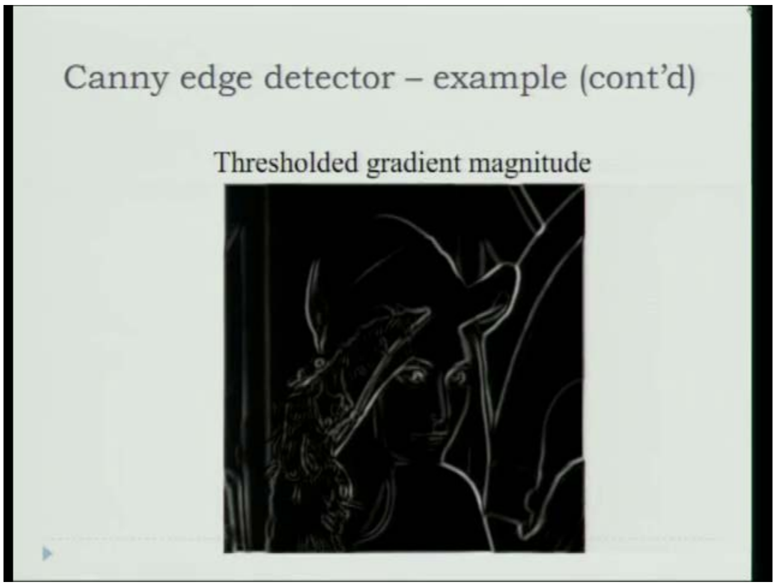


Canny edge detector – example (cont'd)

Gradient magnitude

This is a gradient magnitude.

(Refer Slide Time: 18:00)

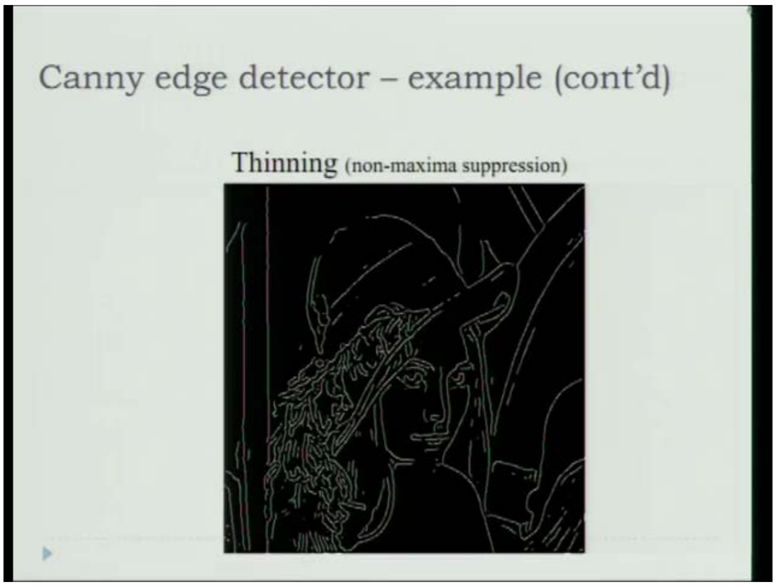


Canny edge detector – example (cont'd)

Thresholded gradient magnitude

Now if we put some threshold then the image will look like this.

(Refer Slide Time: 18:04)

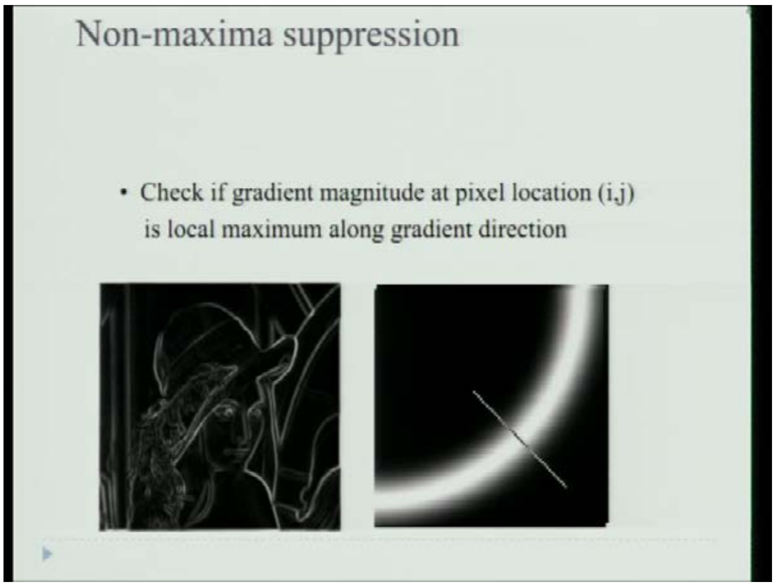


Canny edge detector – example (cont'd)

Thinning (non-maxima suppression)

Then the non maxima suppression things.

(Refer Slide Time: 18:10)

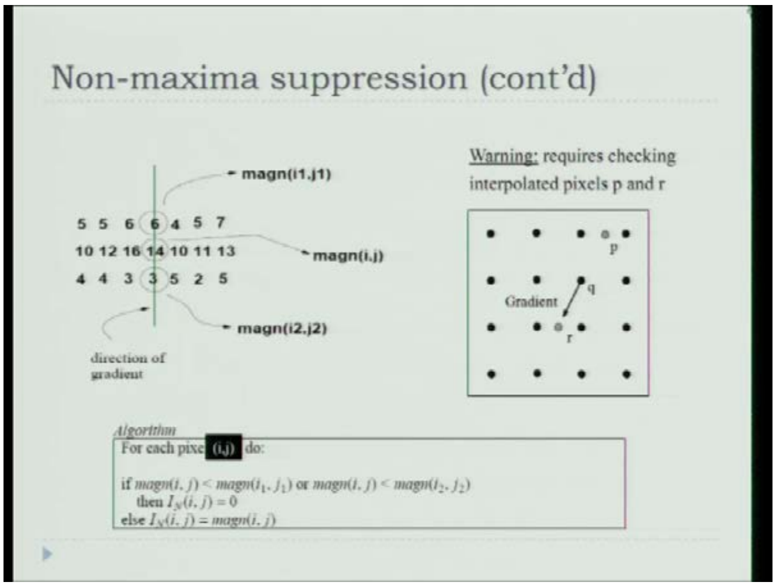


Non-maxima suppression

- Check if gradient magnitude at pixel location (i,j) is local maximum along gradient direction

Here, there is the line the direction vector has to be taken into account for seeing that local maxima.

(Refer Slide Time: 18:17)



Non-maxima suppression (cont'd)

```
 5  5  6  6  4  5  7
10 12 16 14 10 11 13
 4  4  3  3  5  2  5
```

magn(i1,j1)

magn(i,j)

magn(i2,j2)

direction of gradient

Warning: requires checking interpolated pixels p and r

p
q
r
Gradient

Algorithm

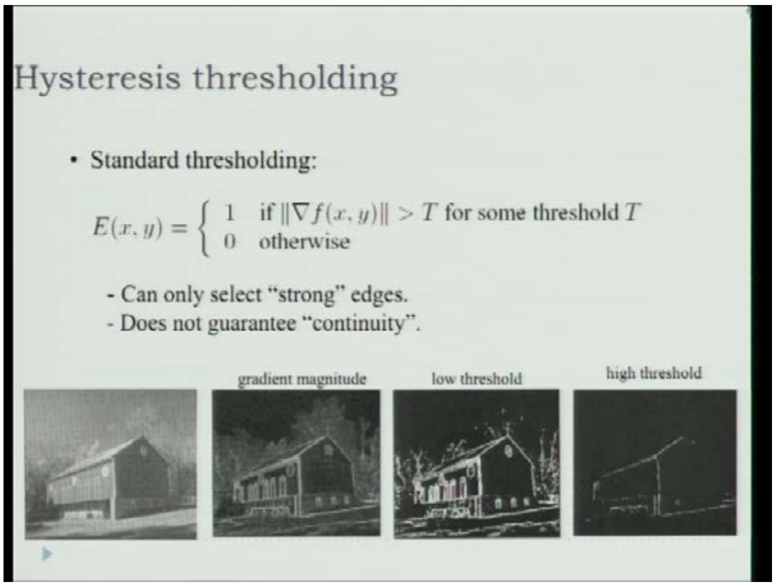
```
For each pixe (i,j) do:

if $magn(i, j) < magn(i_1, j_1)$ or $magn(i, j) < magn(i_2, j_2)$
   then $I_N(i, j) = 0$
else $I_N(i, j) = magn(i, j)$
```

So, this is an example suppose that your direction is this way. So, what I am looking for? If magnitude of i j is less than magnitude of this or magnitude of this lesser then you set it 0. Otherwise you consider this it is an edge part. But it looks simple when your gradient direction is vertical or horizontal. But what happens if it is of this type? You have gradient direction, you have p and here is r and you are looking for q.

So, there complexity is increasing, because direction is not either vertically or horizontally. So, you need to know the interpolated value, you need to know the interpolated value of in between this and then only you will be able to take the decision. Now, how to get that it is not required to consider the bilinear interpolation. Because, line will be if you think in this way the line will go this way.

You need either this side or this side horizontal or vertical. But you do not have to go to the center of this point. Bilinear interpolation is not required, it is not that complicated one, just one directional interpolated value you have to check and then you take the decision whether it should be set to zero or it should be retained.

(Refer Slide Time: 20:05)



Hysteresis thresholding

- Standard thresholding:

$$E(x, y) = \begin{cases} 1 & \text{if } \|\nabla f(x, y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

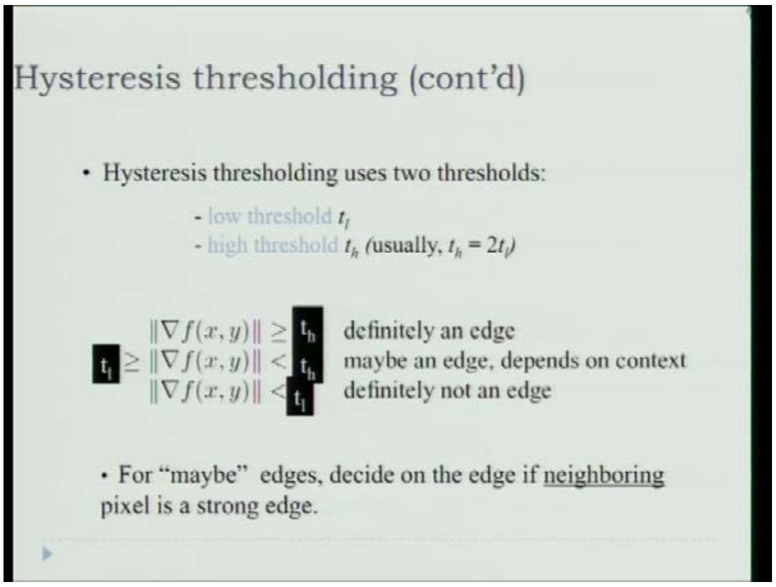
- Can only select "strong" edges.
- Does not guarantee "continuity".

gradient magnitude
low threshold
high threshold

Now standard thresholding technique is like that if you get the difference operator, if you remember that there is a reverse or inverted delta is the difference operator which is nothing but f of x plus 1 y of plus 1 minus f of x y. That is the thing, if that difference value the difference of the value if we define that it is greater than T, then you tell that this is an edge point. So, you tell that e x y is 1 otherwise it is set to 0. So, through that you can get only the strong edges or you can get edges and also the weak edges.

So there there is no guaranty of continuity if the T is very high, because you will be considering only the strong edges. So, there will be discontinuity between their two sub edges but in case of Canny what we have? One low threshold value another high threshold value.

(Refer Slide Time: 21:22)



Hysteresis thresholding (cont'd)

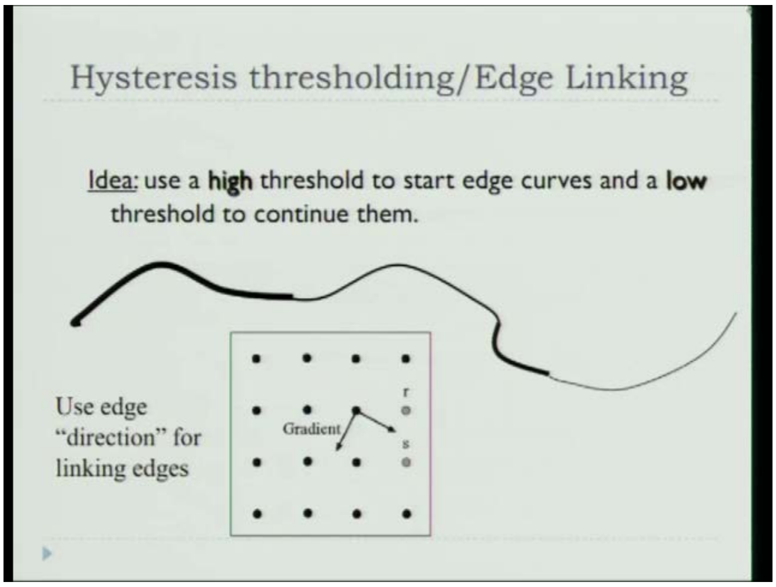
- Hysteresis thresholding uses two thresholds:
  - low threshold t_l
  - high threshold t_h (usually, $t_h = 2t_l$)

$$\begin{array}{rl} \|\nabla f(x,y)\| \geq t_h & \text{definitely an edge} \\ t_l \geq \|\nabla f(x,y)\| < t_h & \text{maybe an edge, depends on context} \\ \|\nabla f(x,y)\| < t_l & \text{definitely not an edge} \end{array}$$

- For "maybe" edges, decide on the edge if neighboring pixel is a strong edge.

Generally high threshold value is almost equals to the twice of the low threshold value. If you use then the difference operator if you get get of them the high threshold value, you will get 100 percent true edges. And, because they are strong edges if it is less than the low threshold value then it cannot be it cannot be an edge point. But if is lying between the low threshold value and high threshold value, then it may be an edge point or may not be an edge point. Now, these may be edge point you will be deciding based on the neighboring pixels, intensity gradient value, magnitude and the direction.

(Refer Slide Time: 22:16)



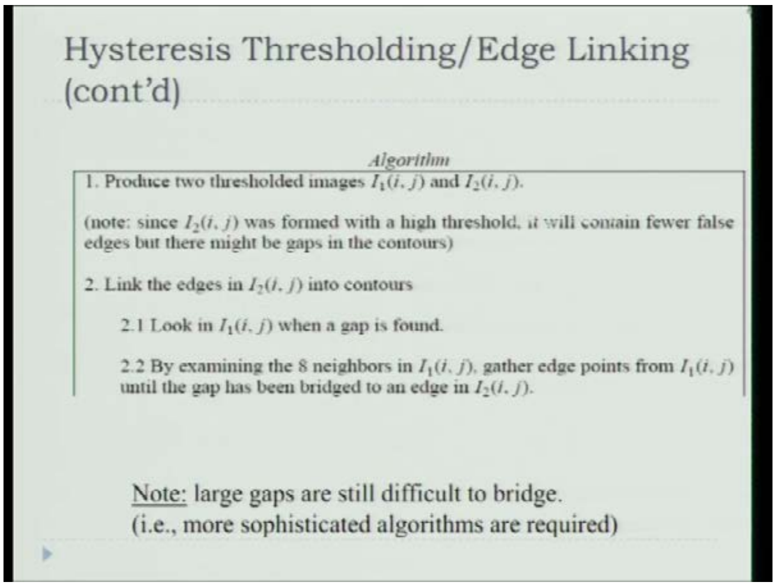
Hysteresis thresholding/Edge Linking

Idea: use a **high** threshold to start edge curves and a **low** threshold to continue them.

Use edge "direction" for linking edges

So, this is the thing that it is a high to a peak this is the weak edges and you have to collect them, that is the idea. Now if I I think this is an algorithm.

(Refer Slide Time: 22:32)



Hysteresis Thresholding/Edge Linking (cont'd)

Algorithm

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$.

(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I_2(i, j)$ into contours

    2.1 Look in $I_1(i, j)$ when a gap is found.

    2.2 By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $I_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

Note: large gaps are still difficult to bridge.
(i.e., more sophisticated algorithms are required)

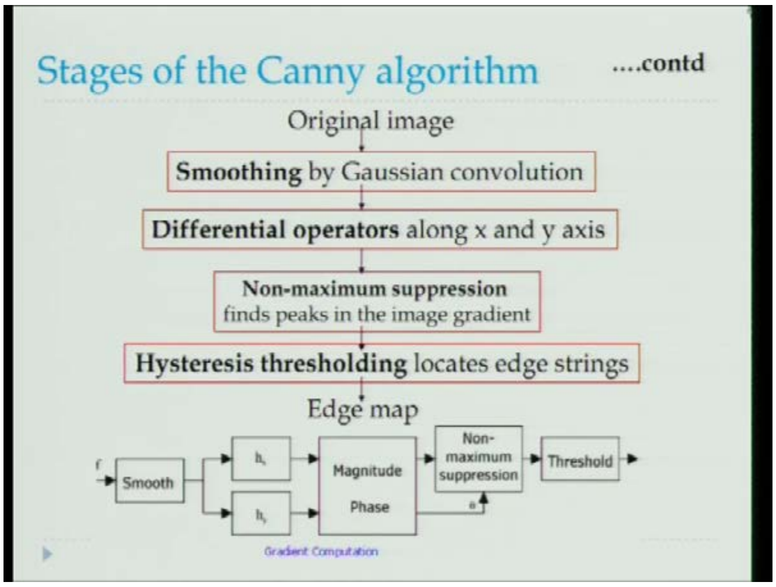
So, if I define it algorithmically, I have the two edge map you have. One is with the high thresholded image another one is low thresholded image. So you have I 1 and I 2. Let us assume that i 1 is the high thresholded image and i 2 is the low thresholded image. Now link the edges in I 2 x I 2 x so I to i j into the contours. Look in I 1 i j this is the low threshold value when gap is found. If there is no gap then it is any way true edge, you do not have to do anything. If gap is found there is I 2 i j is 0 there, then you examine the 8 neighbors of I 1 i j. Whether there exists a weak edge or not whether there exist a non zero value or not. If it exists then you consider I 2 i j as a edge pixel.

However, there is a possibility that if there is a large gap you may not be able to connect that. Because, you know after sometime this is a large gap between two true edges.

(Refer Slide Time: 23:54)

So connecting is very sometimes will become very difficult, because you may get several several sevaral weak edges. Now there is no guarantee that you will be following a path which will touch this one, because you may start going this way, which will lead you to some other place. You may start following this path which will take some other place, it will not come here. There is the pause. So if there is a long gap between the two strong sub edges connected to a (( )) problem you may not get that connected one.

(Refer Slide Time: 24:37)



Stages of the Canny algorithm ....contd

Original image

Smoothing by Gaussian convolution

Differential operators along x and y axis

Non-maximum suppression
finds peaks in the image gradient

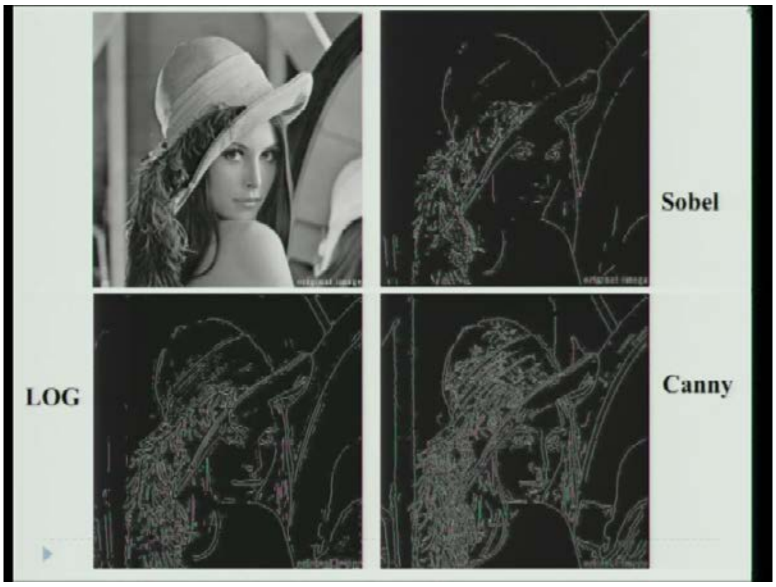
Hysteresis thresholding locates edge strings

Edge map

f
Smooth
h_x
h_y
Magnitude
Phase
θ
Non-maximum suppression
Threshold
Gradient Computation

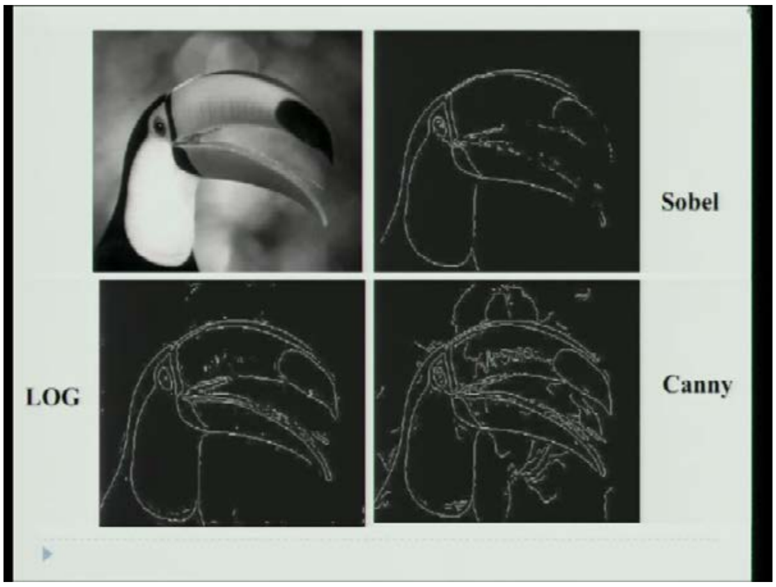
So this is overall algorithm first smoothing the differential operator you have taken, then you have consider non maximum and finally, hysteresis thresholding.

(Refer Slide Time: 24:50)



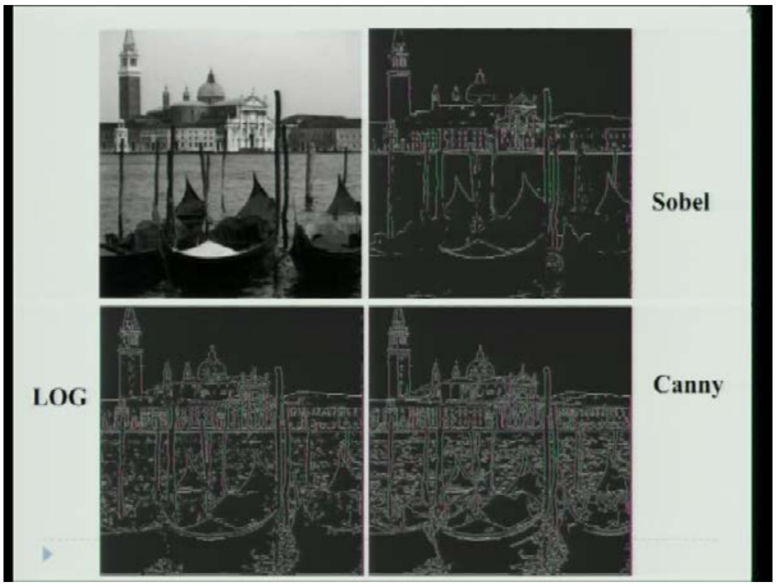
So, that one is the Sobel and see that the Sobel it looks very neat and clean. But lot of edge breaks are there but it is not that case with the Canny. But you will find that Canny is one contains lot many edges compared to any others.

(Refer Slide Time: 25:11)



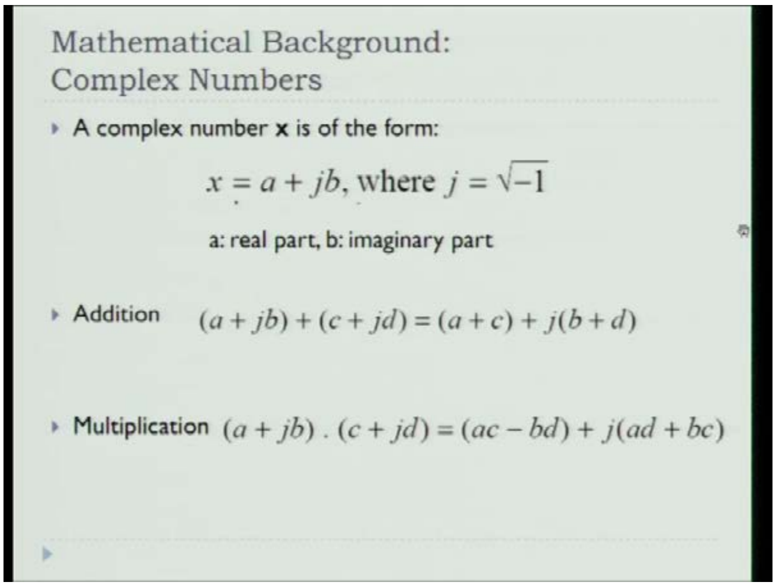
This is the parrots one; so you you see that this will give you the better one. See this, lot of broken edges are there. But this is not the case with the Canny. I think this is another example.

(Refer Slide Time: 25:25)



So, this is all about Canny Edge Detection or the edge detection algorithms. Now, next one is the Fourier transform, how many of you know the Fourier transform? So this terminology is unknown to you. Let us start let us start from this scratch.

(Refer Slide Time: 25:40)



Mathematical Background: Complex Numbers

- A complex number **x** is of the form:

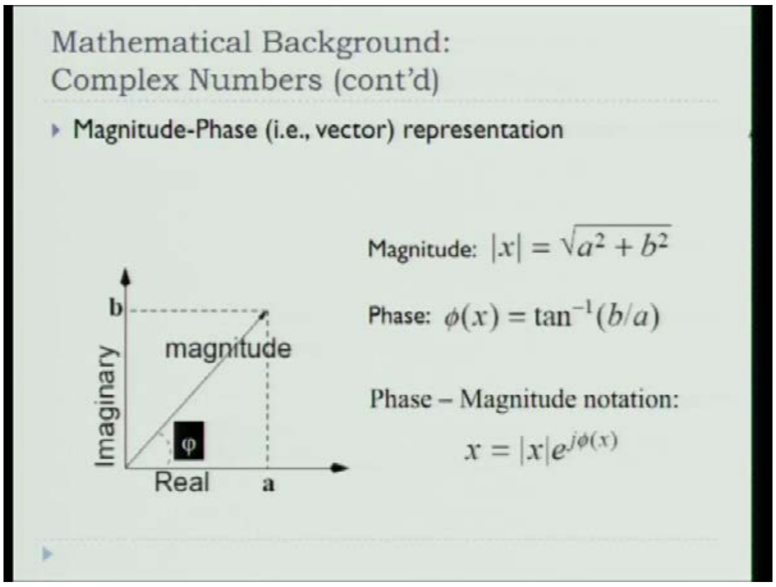
$$x = a + jb, \text{ where } j = \sqrt{-1}$$

a: real part, b: imaginary part

- Addition $(a + jb) + (c + jd) = (a + c) + j(b + d)$
- Multiplication $(a + jb) \,.\, (c + jd) = (ac - bd) + j(ad + bc)$

A complex number x is of the form x equals to a plus j b; j is square root of minus one and a is the real part and b is the imaginary part; and that addition of the two complex number is also a complex number; similarly, the case with the multiplication.

(Refer Slide Time: 26:05)



Mathematical Background: Complex Numbers (cont'd)

- Magnitude-Phase (i.e., vector) representation

Magnitude: $|x| = \sqrt{a^2 + b^2}$

Phase: $\phi(x) = \tan^{-1}(b/a)$

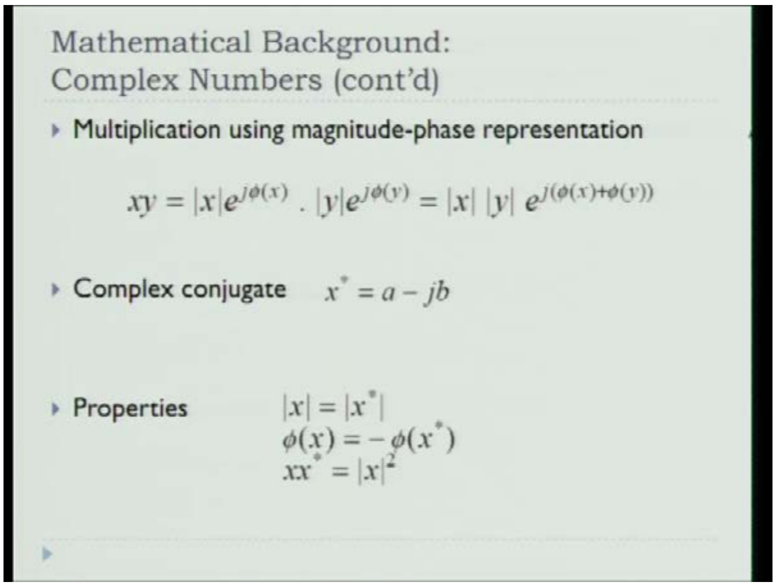
Phase – Magnitude notation:

$$x = |x|e^{j\phi(x)}$$

b
magnitude
Imaginary
φ
Real
a

Now there is a term magnitude and phase representation; magnitude is nothing but square root of a square plus b square and phase is tan inverse of b minus a and phase magnitude notation is absolute value of x e to the power j phi x.

(Refer Slide Time: 26:24)



Mathematical Background: Complex Numbers (cont'd)

- Multiplication using magnitude-phase representation

$$xy = |x|e^{j\phi(x)} \,.\, |y|e^{j\phi(y)} = |x|\,|y|\; e^{j(\phi(x)+\phi(y))}$$

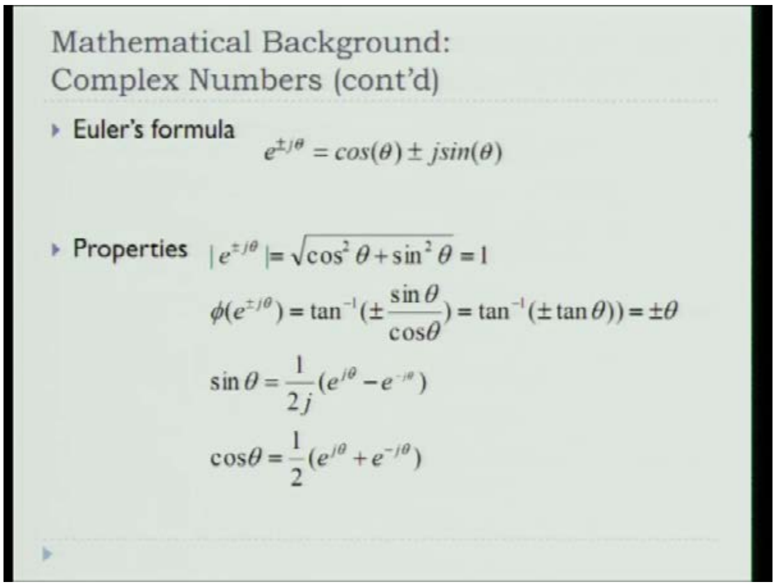
- Complex conjugate $x^* = a - jb$

- Properties

$$|x| = |x^*|$$
$$\phi(x) = -\phi(x^*)$$
$$xx^* = |x|^2$$

Now if I write multiplication using the two magnitude phase representation representation x y I can obtain mod absolute magnitude of x dot magnitude of y e to the power j phi x plus phi y. And complex conjugate conjugate x star is a minus j b properties are x absolute value of x equals to or magnitude of x equals to magnitude of x star and phi x is equal to minus phi x star and x x star is absolute value of square root of the absolute value of x.

(Refer Slide Time: 27:02)



Mathematical Background: Complex Numbers (cont'd)

- Euler's formula $e^{\pm j\theta} = cos(\theta) \pm jsin(\theta)$
- Properties $|e^{\pm j\theta}| = \sqrt{\cos^2\theta + \sin^2\theta} = 1$

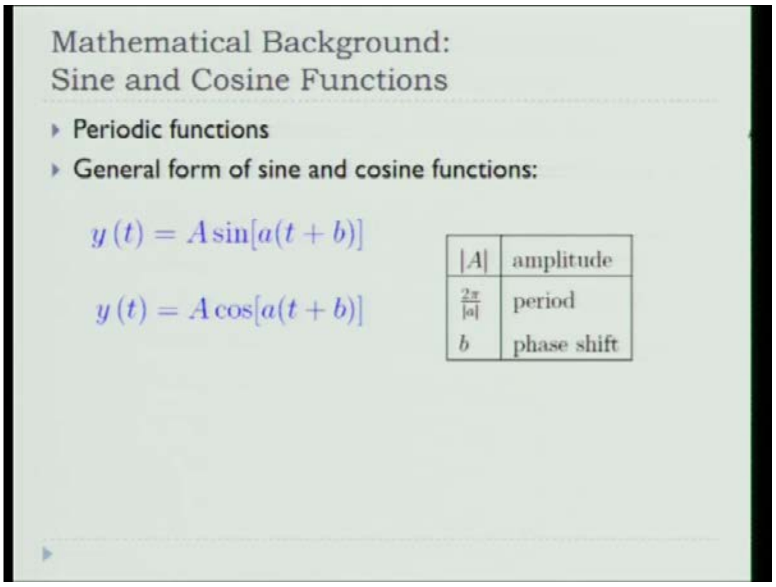
$$\phi(e^{\pm j\theta}) = \tan^{-1}(\pm\frac{\sin\theta}{\cos\theta}) = \tan^{-1}(\pm\tan\theta)) = \pm\theta$$

$$\sin\theta = \frac{1}{2j}(e^{j\theta} - e^{-j\theta})$$

$$\cos\theta = \frac{1}{2}(e^{j\theta} + e^{-j\theta})$$

This is Euler's formula e to the power plus minus j theta is equal to cos theta plus minus j sine theta its property holds good very, very simple properties.

(Refer Slide Time: 27:14)



Mathematical Background: Sine and Cosine Functions

- Periodic functions
- General form of sine and cosine functions:

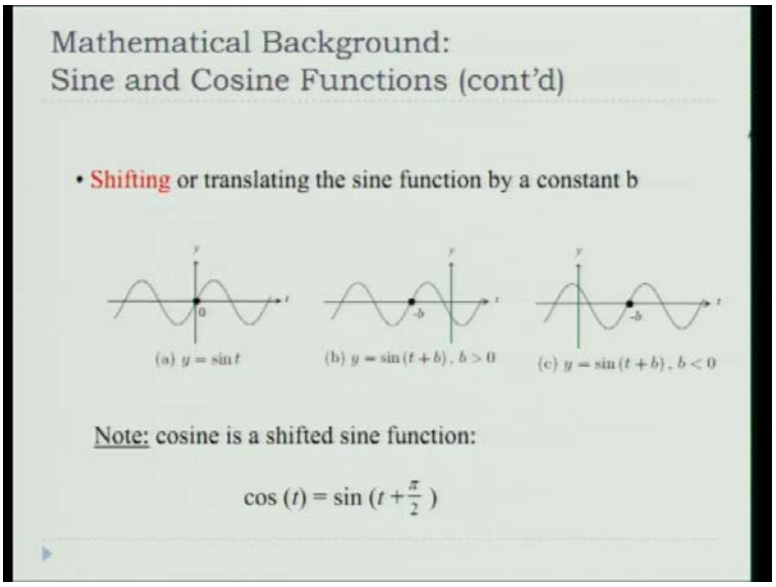
$$y(t) = A\sin[a(t+b)]$$

$$y(t) = A\cos[a(t+b)]$$

$\lvert A \rvert$	amplitude
$\frac{2\pi}{\lvert a \rvert}$	period
b	phase shift

And now, sine function and cosine function they are periodic functions. And, general form of sine and cosine functions are A which is amplitude sine a t b t plus b t is time domain it is time and b is the what? Phase shift and 2 pi by absolute value of a gives you the period.

(Refer Slide Time: 27:47)



Mathematical Background: Sine and Cosine Functions (cont'd)

- Shifting or translating the sine function by a constant b

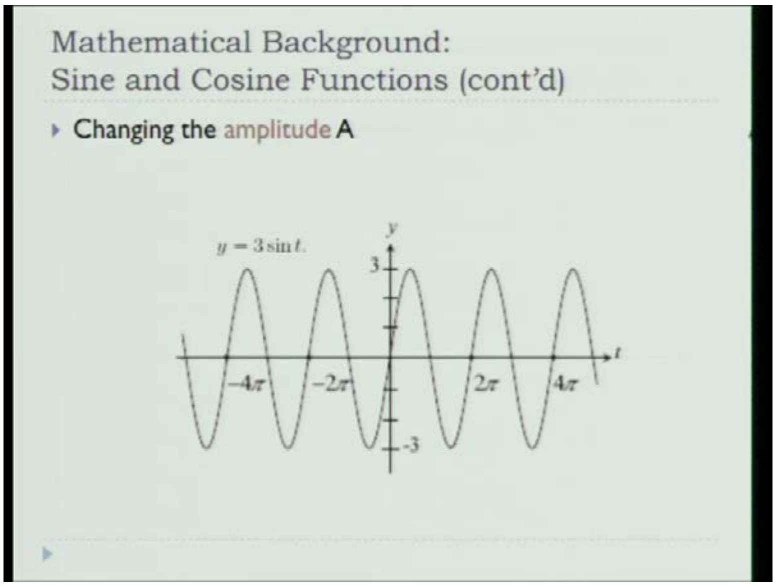
(a) $y = \sin t$ (b) $y = \sin(t+b)$, $b > 0$ (c) $y = \sin(t+b)$, $b < 0$

Note: cosine is a shifted sine function:

$$\cos(t) = \sin(t + \frac{\pi}{2})$$

This is the well known sine and cosine function for different values of a and a. Shifting if you can shift by constant b phase shift and you can take from this you can draw the conclusion cosine is a shifted sign function.

(Refer Slide Time: 28:11)



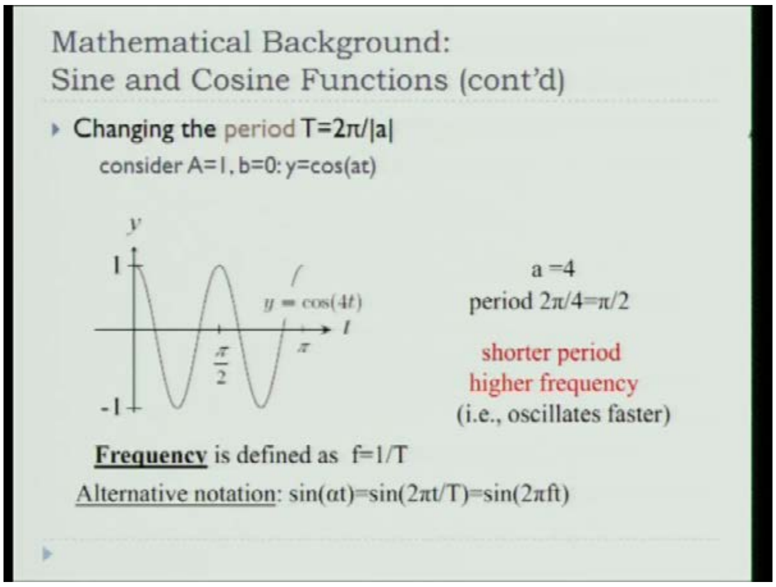
Mathematical Background: Sine and Cosine Functions (cont'd)

- Changing the amplitude A

$y = 3\sin t$
y
3
-3
t
-4π
-2π
2π
4π

Changing of amplitude just you put the value A by 3, you will be getting this type of graph.

(Refer Slide Time: 28:19)



Mathematical Background: Sine and Cosine Functions (cont'd)

- Changing the period T=2π/|a|

  consider A=1, b=0: y=cos(at)

y
1
-1
$y = \cos(4t)$
t
$\frac{\pi}{2}$
π

a =4

period 2π/4=π/2

shorter period

higher frequency

(i.e., oscillates faster)

Frequency is defined as f=1/T

Alternative notation: sin(αt)=sin(2πt/T)=sin(2πft)

Changing the period period is I told you that is T; T is nothing but 2 pi by the absolute value of a and considering A equals to one, b equals to zero and y is equals to cos of a t.

A if you consider 4 period becomes pi by 2 and you observe that it has become the shorter period and higher frequency, though it becomes first oscillation and frequency is defined by one by t. So, sine of a t can be expressed as sine of 2 pi f t, if f is the frequency.

(Refer Slide Time: 28:54)

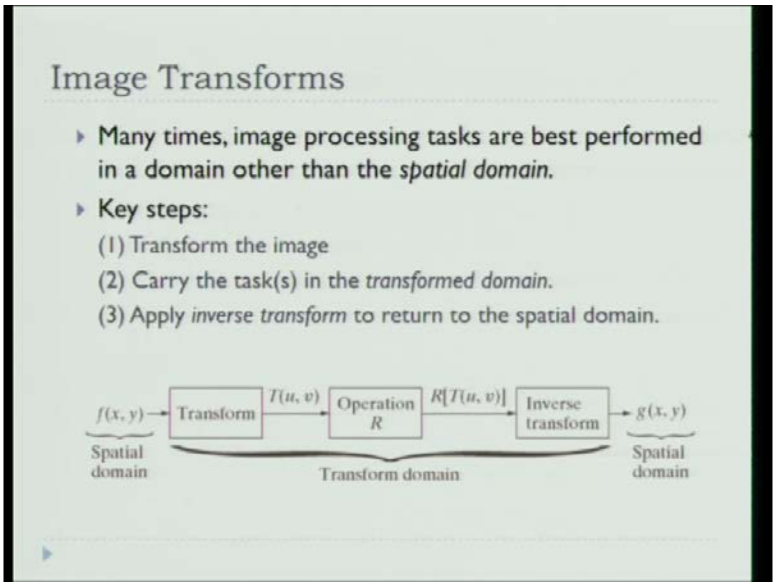


Image Transforms

- Many times, image processing tasks are best performed in a domain other than the *spatial domain.*
- Key steps:

  (1) Transform the image

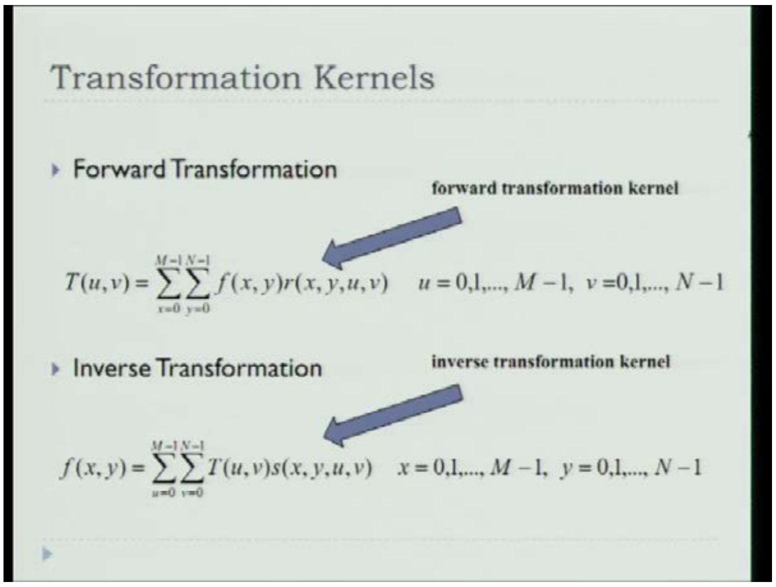
  (2) Carry the task(s) in the *transformed domain.*

  (3) Apply *inverse transform* to return to the spatial domain.

$$\underbrace{f(x,y)}_{\text{Spatial domain}} \rightarrow \underbrace{\boxed{\text{Transform}} \xrightarrow{T(u,v)} \boxed{\text{Operation } R} \xrightarrow{R[T(u,v)]} \boxed{\text{Inverse transform}}}_{\text{Transform domain}} \rightarrow \underbrace{g(x,y)}_{\text{Spatial domain}}$$

So image transform so, sometimes you know what happens that you want to perform an operations on image it becomes very difficult it is you are not able to visualize the things. What you do that we convert or transform from the special domain to another domain, perform the operation on that domain and then they take the inverse of the transformation that is the thing. So, first is transform the image into another domain. Then carry out what operation you want to perform then perform again inverse transformation on it.

(Refer Slide Time: 29:28)



Transformation Kernels

- Forward Transformation

forward transformation kernel

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

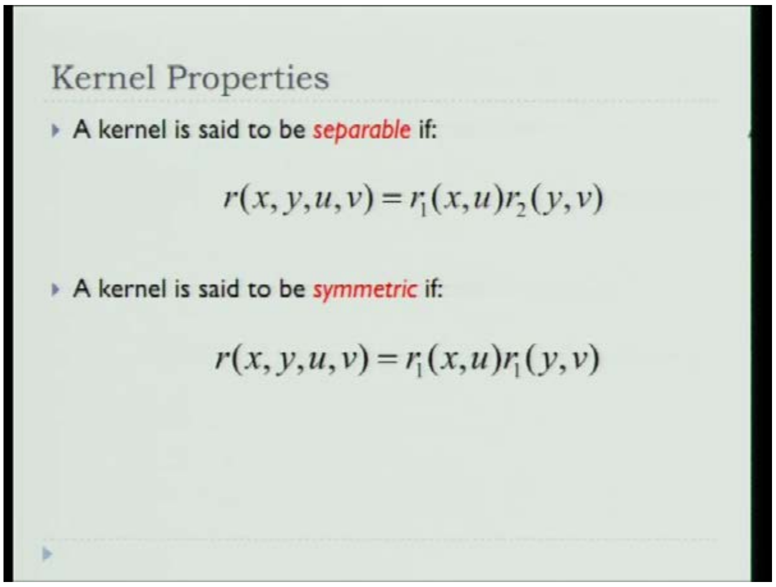
- Inverse Transformation

inverse transformation kernel

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

So f x y was your originally pixel value of a image and this is the forward transformation and resultant you get the forward transform image and inverse one is just it is inverse of it.

(Refer Slide Time: 29:47)



Kernel Properties

- A kernel is said to be separable if:

$$r(x,y,u,v)=r_1(x,u)r_2(y,v)$$

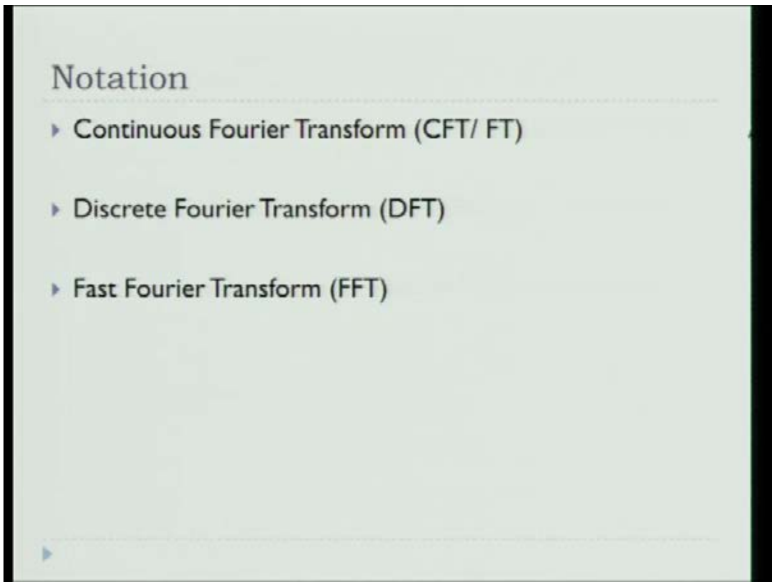
- A kernel is said to be symmetric if:

$$r(x,y,u,v)=r_1(x,u)r_1(y,v)$$

Now, the Kernel is said to be separable. If you can write r x y u v that is the kernel you can write r 1 x u dot r 2 y v. Now, r 1 x 1 along the x axis you can have some Kernel and along the y axis you can have another Kernel.

Kernel is said to be symmetric, if you can write r 1 r x y u v equals to r 1 x u dot r 1 y v. Now these are the common terminology we will be using in our discussion one is that Fourier transform.

(Refer Slide Time: 30:32)

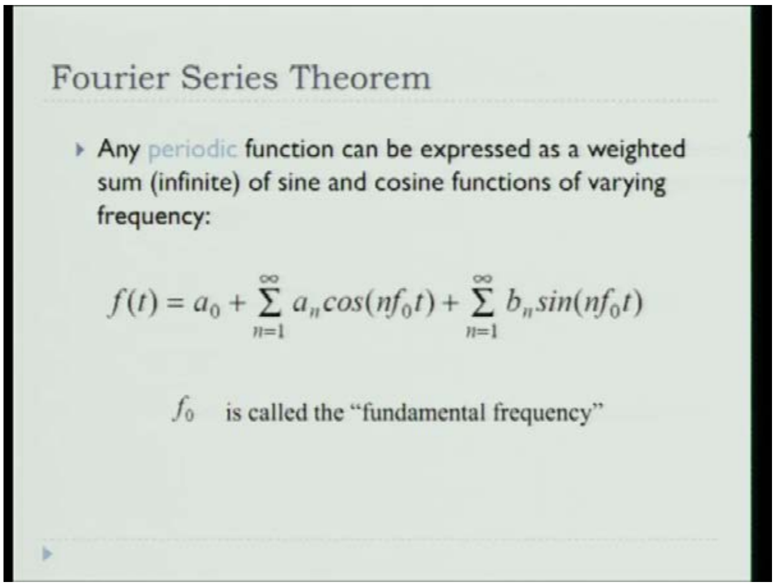


Notation

- Continuous Fourier Transform (CFT/ FT)
- Discrete Fourier Transform (DFT)
- Fast Fourier Transform (FFT)

Generally, we do not write CFT writer FT and then Discrete Fourier Transform because we will be performing the operation on discrete domain and then Fast Fourier Transform.

(Refer Slide Time: 30:44)



Fourier Series Theorem

- Any periodic function can be expressed as a weighted sum (infinite) of sine and cosine functions of varying frequency:

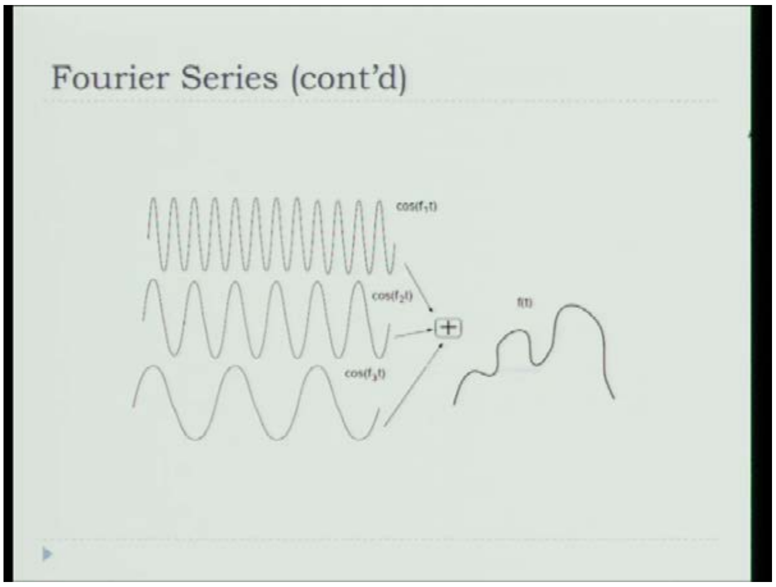
$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n cos(nf_0 t) + \sum_{n=1}^{\infty} b_n sin(nf_0 t)$$

f_0 is called the "fundamental frequency"

But we will be discussing the algorithm how to do this all available just I want to browse the things so that, while discussing the algorithms in biometric system you should think that this is the thing we have done.

Fourier Series Theorem is like any periodic function can be expressed as the summation of or infinite summation of with respect to cosine function and sine function which is defined by f t equals to a 0 summation n equals to 1 to infinity a n cos n f 0 t plus summation n equals to 1 to infinity b n sine n f 0 t, where f 0 is called the fundamental frequency.

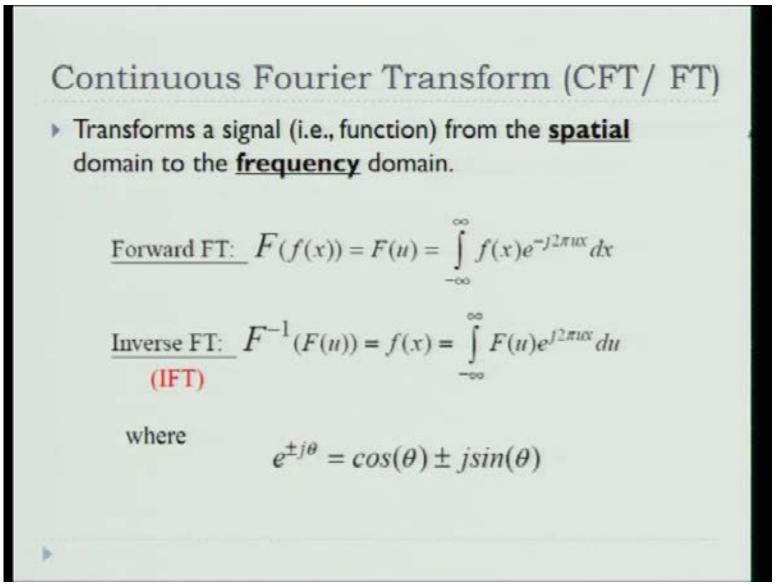
(Refer Slide Time: 31:25)



Fourier Series (cont'd)

This is one example the sum of some function you can get on FTS.

(Refer Slide Time: 31:32)



Continuous Fourier Transform (CFT/ FT)

- Transforms a signal (i.e., function) from the **spatial** domain to the **frequency** domain.

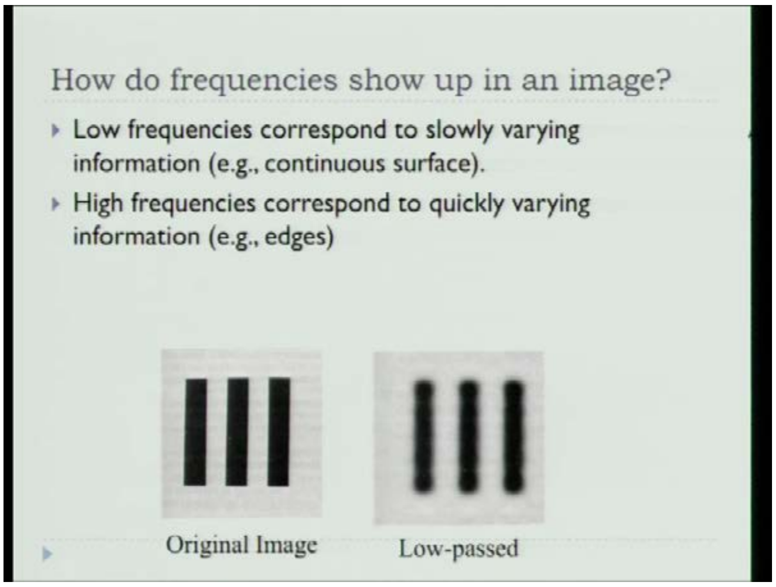
Forward FT: $F(f(x)) = F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$

Inverse FT: (IFT) $F^{-1}(F(u)) = f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$

where $e^{\pm j\theta} = cos(\theta) \pm jsin(\theta)$

So, continuous Fourier Transform; this transform a signal it transform a function from the special domain to the frequency domain. And this is the F f (x) is equals to F u equals to minus infinity to plus infinity f (x) e to the power minus j 2 pi u x, where e to the power plus minus j theta is defined by cos theta plus minus j sine theta. Similarly, inverse is defined.

(Refer Slide Time: 31:59)



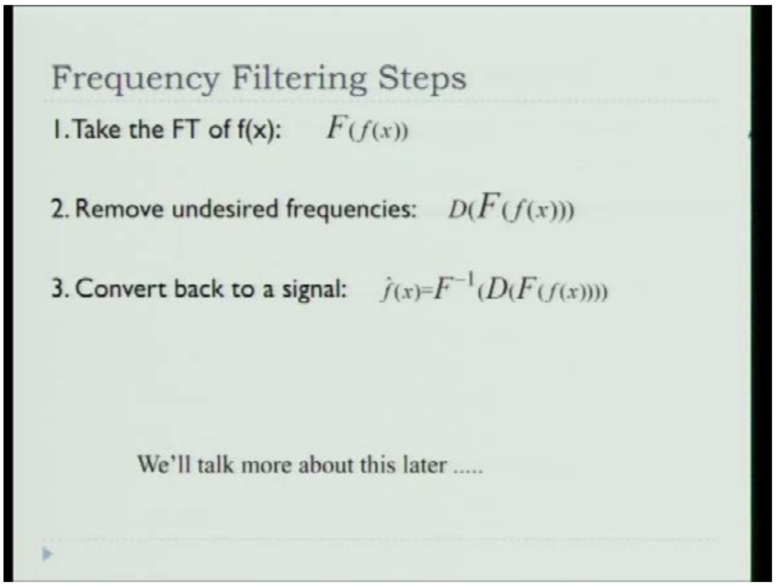
How do frequencies show up in an image?

- Low frequencies correspond to slowly varying information (e.g., continuous surface).
- High frequencies correspond to quickly varying information (e.g., edges)

Original Image Low-passed

So, the low frequency corresponds to that corresponds to that it is continuity of the intensity level. That means, there is no change or gradually it is changing the intensity value. But high frequency corresponds to the sudden change or you can get the edge information there.

(Refer Slide Time: 32:26)



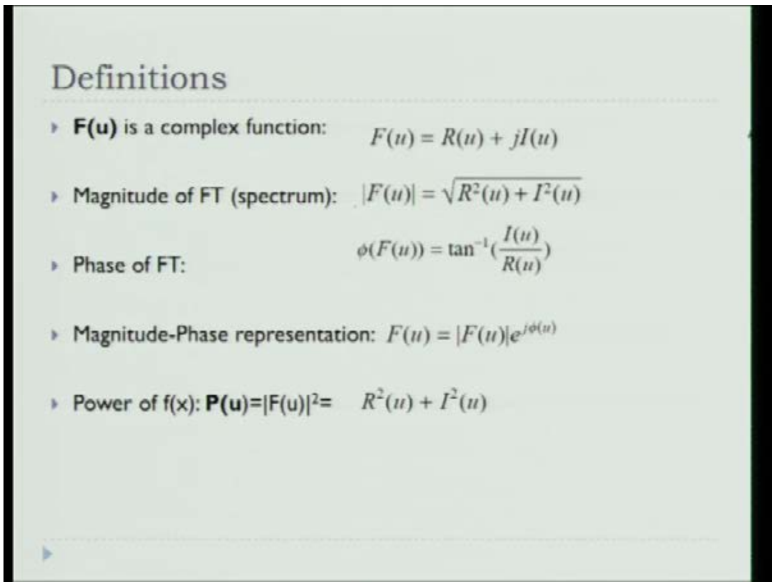
Frequency Filtering Steps

1. Take the FT of f(x): $F(f(x))$
2. Remove undesired frequencies: $D(F(f(x)))$
3. Convert back to a signal: $\hat{f}(x) = F^{-1}(D(F(f(x))))$

We'll talk more about this later .....

So, suppose you want to do the filtering operations on it then take the Fourier transform. And, then you perform some operations to remove the undesired component and then you just take the inverse of it you will be getting back, your original image.

(Refer Slide Time: 32:47)



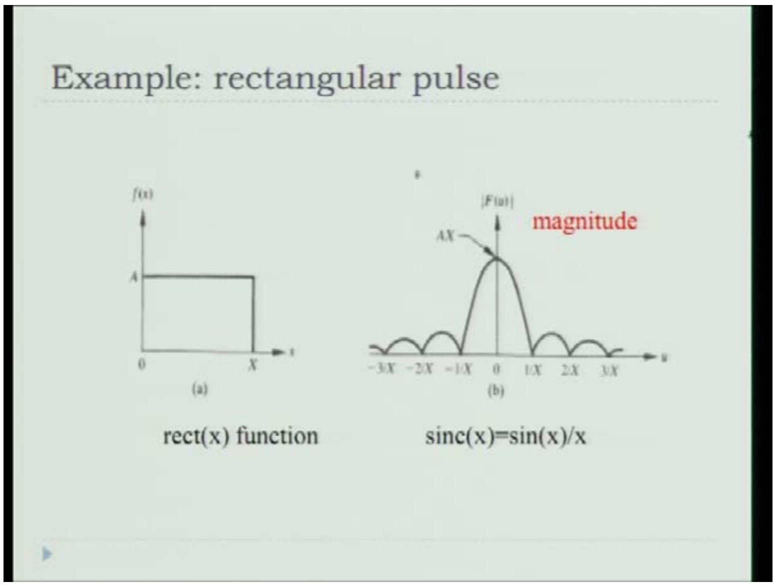
Definitions

- **F(u)** is a complex function: $F(u) = R(u) + jI(u)$
- Magnitude of FT (spectrum): $|F(u)| = \sqrt{R^2(u) + I^2(u)}$
- Phase of FT: $\phi(F(u)) = \tan^{-1}(\frac{I(u)}{R(u)})$
- Magnitude-Phase representation: $F(u) = |F(u)|e^{j\phi(u)}$
- Power of f(x): **P(u)**=|F(u)|²= $R^2(u) + I^2(u)$

Now, F u is a complex function; so F u can be expressed as R u plus j I u and magnitude is defined by square root of R u square plus I so I I square u and phase is nothing but phi

F u is equal to tan inverse I u by R u and magnitude phase representation is magnitude of F u into the power j phi u and power of f x also R square u plus I square u.

(Refer Slide Time: 33:19)

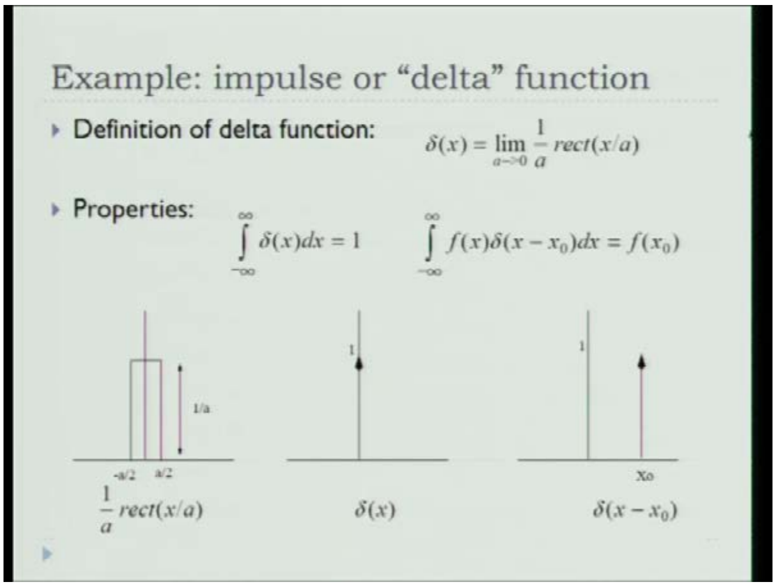

Example: rectangular pulse

rect(x) function

sinc(x)=sin(x)/x

These are two functions defined as rect x and see sinc x.

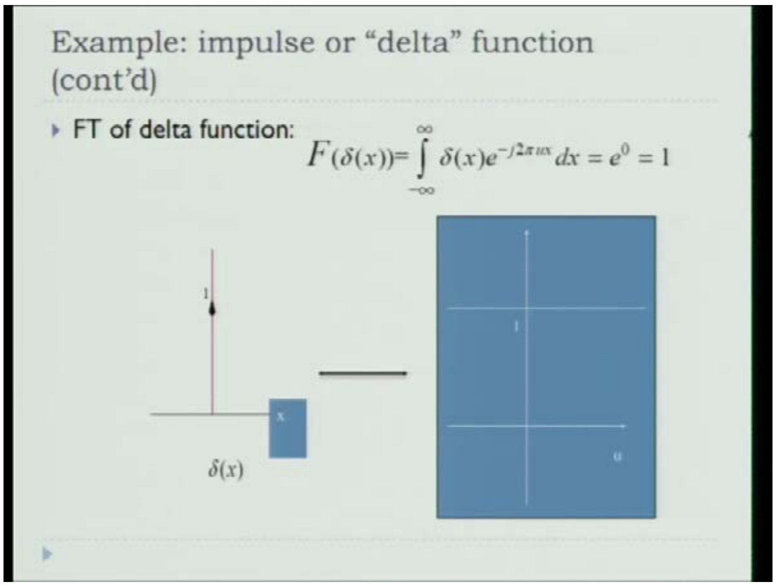
(Refer Slide Time: 33:19)


Example: impulse or "delta" function

- Definition of delta function: $\delta(x) = \lim_{a \to 0} \frac{1}{a} rect(x/a)$
- Properties: $\int_{-\infty}^{\infty} \delta(x)dx = 1$ $\int_{-\infty}^{\infty} f(x)\delta(x-x_0)dx = f(x_0)$

And delta function is defined I think that is the thing important. Delta function is defined by delta x equals to limit a tends to 0 1 by a rect x by a, this is the function and property is minus infinity to plus infinity delta x f d x is equal to 1 and shifting thing is that minus infinity to plus infinity f (x) delta x minus x 0 d s equals to f (x 0).

(Refer Slide Time: 33:51)



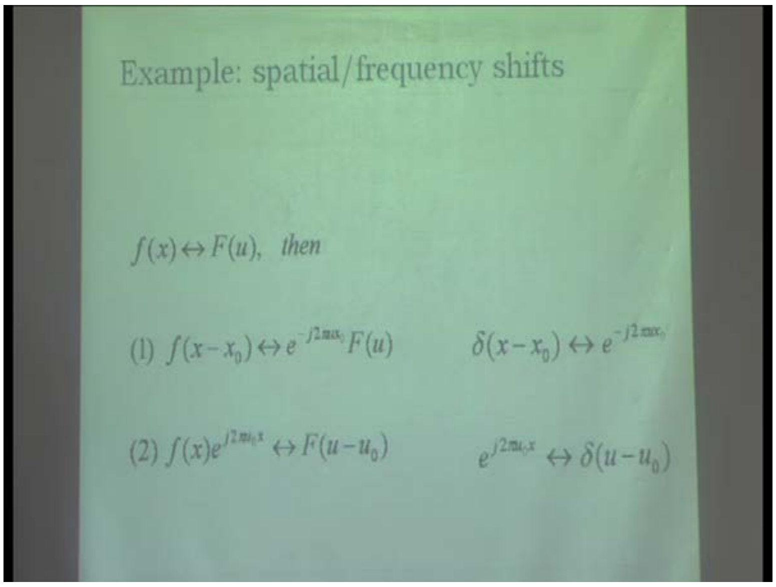
Example: impulse or "delta" function (cont'd)

- FT of delta function: $F(\delta(x)) = \int_{-\infty}^{\infty} \delta(x) e^{-j2\pi ux} dx = e^0 = 1$

1
x
$\delta(x)$
1
u

So Fourier transform of delta function is given by minus infinity to plus infinity delta x e to the power j 2 pi u x d x it gives you the e to the power 0 which is 1.

(Refer Slide Time: 34:06)



Example: spatial/frequency shifts

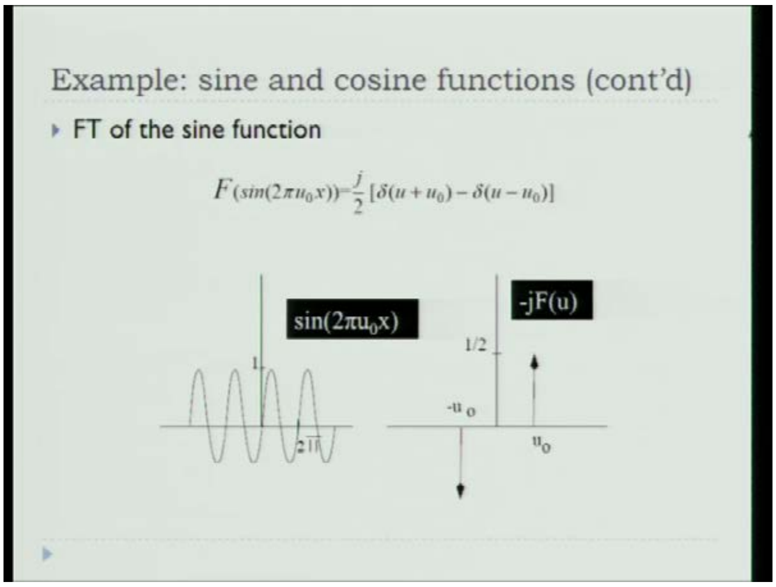
$f(x) \leftrightarrow F(u)$, *then*

(1) $f(x-x_0) \leftrightarrow e^{-j2\pi ux_0} F(u)$ $\delta(x-x_0) \leftrightarrow e^{-j2\pi ux_0}$

(2) $f(x)e^{j2\pi u_0 x} \leftrightarrow F(u-u_0)$ $e^{j2\pi u_0 x} \leftrightarrow \delta(u-u_0)$

So this is a simple shifting properties it maintains this is f of x minus x 0 you will be getting its corresponding transformation will be given that e to the power j 2 pi u x 0 F (u).

(Refer Slide Time: 34:25)



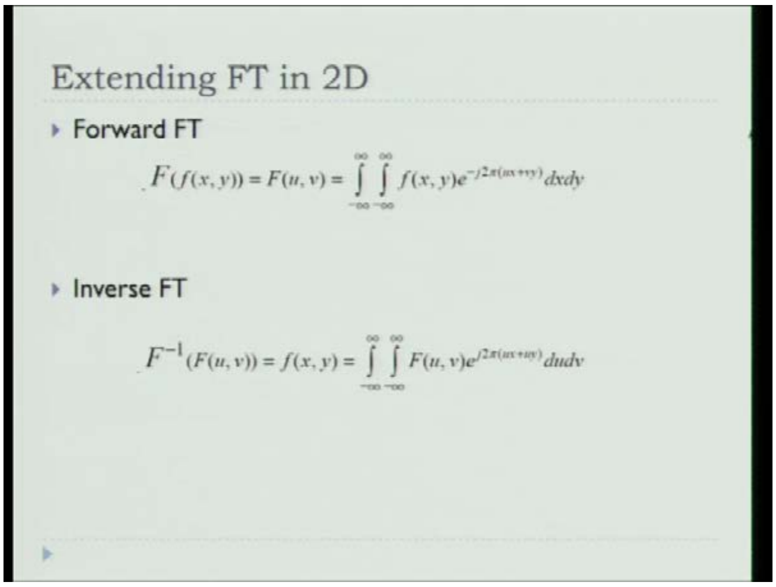
Example: sine and cosine functions (cont'd)

- FT of the sine function

$$F(\sin(2\pi u_0 x)) = \frac{j}{2}[\delta(u+u_0) - \delta(u-u_0)]$$

Then, Fourier transformation of cosine function and also the sine function are also given this also you can easily find, because just you have to browse your book you will be able to get it these are I do not want to cover this.

(Refer Slide Time: 34:36)



Extending FT in 2D

- Forward FT

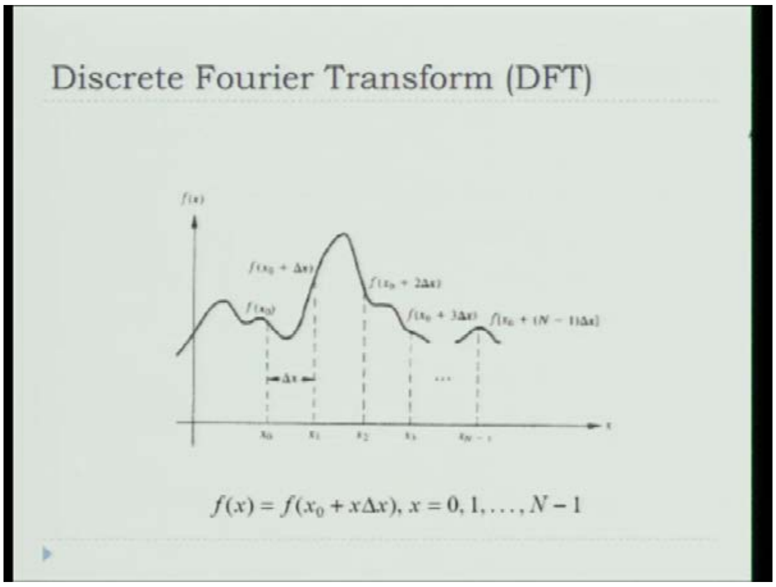
$$F(f(x,y)) = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy$$

- Inverse FT

$$F^{-1}(F(u,v)) = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv$$

Fourier transform in two dimension is that obviously bind with that will be two integration there for u and v and so you will be getting this and inverse of this that is f inverse F of u v is equal to f (x,y) minus infinity to plus infinity minus infinity to plus infinity f of u v e to the power j 2 pi u x plus v y du dv.

(Refer Slide Time: 35:03)

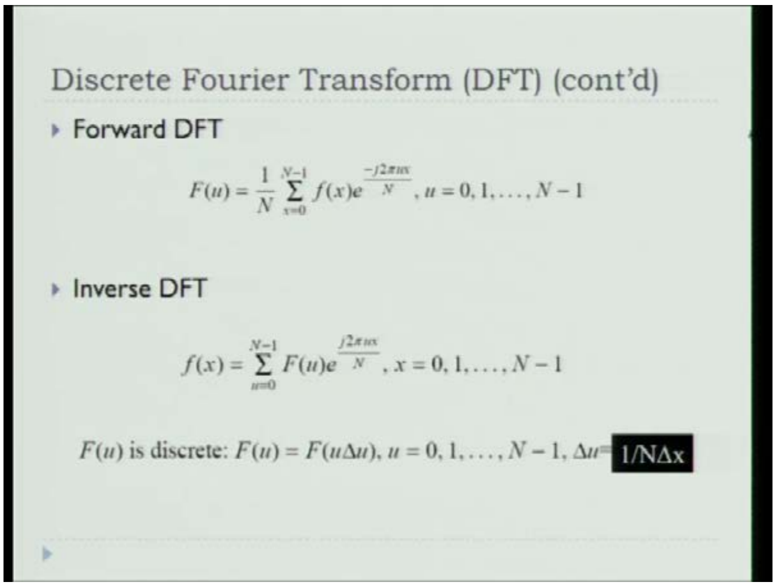


Discrete Fourier Transform (DFT)

$$f(x) = f(x_0 + x\Delta x),\ x = 0, 1, \ldots, N-1$$

Now, the question is given that was on continuous domain it is difficult to it will not possible for us to use them, we have to go to our discrete domain. There is one slight problem. Anyway, f (x) equals to so I have divided as you know you do in the case of numerical methods same form things f x equals to f of x plus x 0 plus this is not x. This is sum other I let us consider i delta x and i is 0 1 to n minus one. So, I have partitioned them then I get these values, it should not be x delta x it should be i delta x.

(Refer Slide Time: 35:56)



Discrete Fourier Transform (DFT) (cont'd)

- Forward DFT

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{\frac{-j2\pi ux}{N}},\ u = 0, 1, \ldots, N-1$$

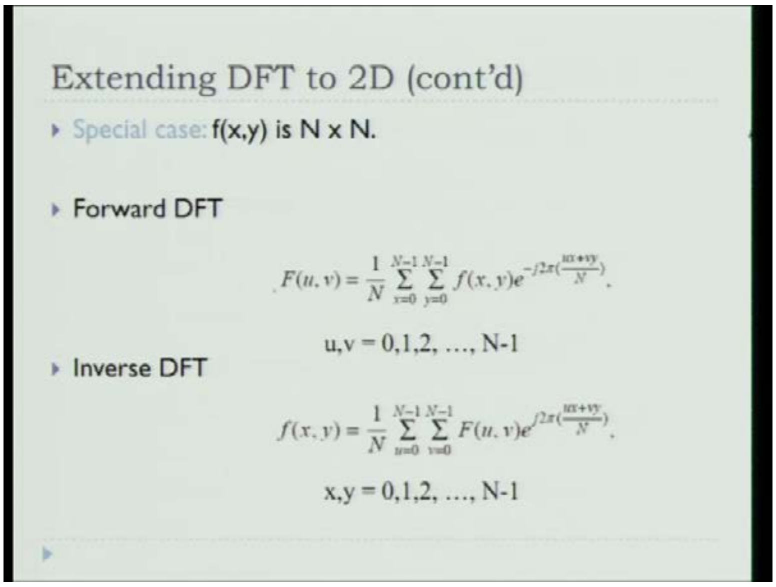
- Inverse DFT

$$f(x) = \sum_{u=0}^{N-1} F(u)e^{\frac{j2\pi ux}{N}},\ x = 0, 1, \ldots, N-1$$

$F(u)$ is discrete: $F(u) = F(u\Delta u),\ u = 0, 1, \ldots, N-1,\ \Delta u = 1/N\Delta x$

And forward DFT formula is F of u one by N summation f of x this is a one dimensional things e to the power minus j 2 pi u x by n. And, inverse is coming summation over u equals to 0 to n minus 1 F of u e to the power j 2 pi u x by n. This should be again i delta u and delta u is equals to one by N delta x.

(Refer Slide Time: 36:22)



Extending DFT to 2D (cont'd)

- Special case: f(x,y) is N x N.
- Forward DFT

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

u,v = 0,1,2, …, N-1

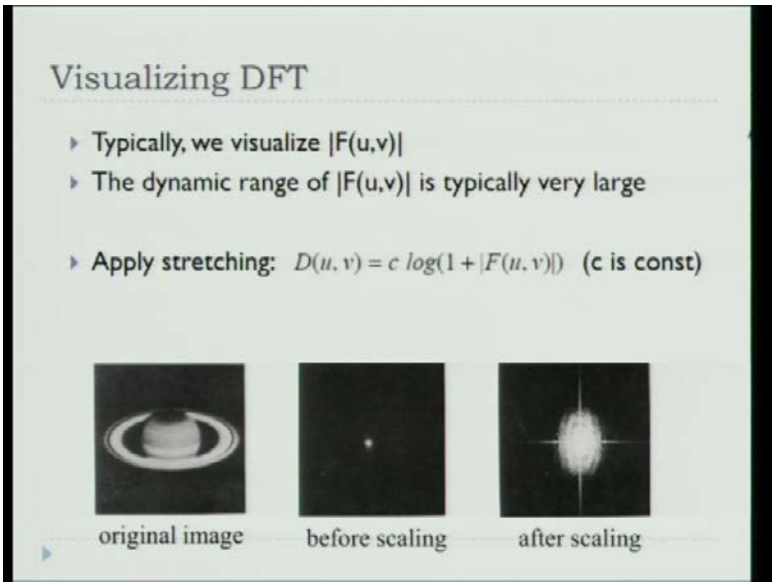
- Inverse DFT

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

x,y = 0,1,2, …, N-1

So in the case of two dimensional f x y is m cross n if you assume that F of u v becomes 1 by M N and you have the same formula, only thing is that it considers the two dimensional effect. And, inverse is equals to summation over u equals to 0 to m minus 1 v equals to 0 to n minus 1 F (u,v) e to the power 2 pi u x by m and v y by n. Now, if m equals to n if m equals to n the formula becomes very simplified one, just I put n equals to m nothing else.

(Refer Slide Time: 37:01)



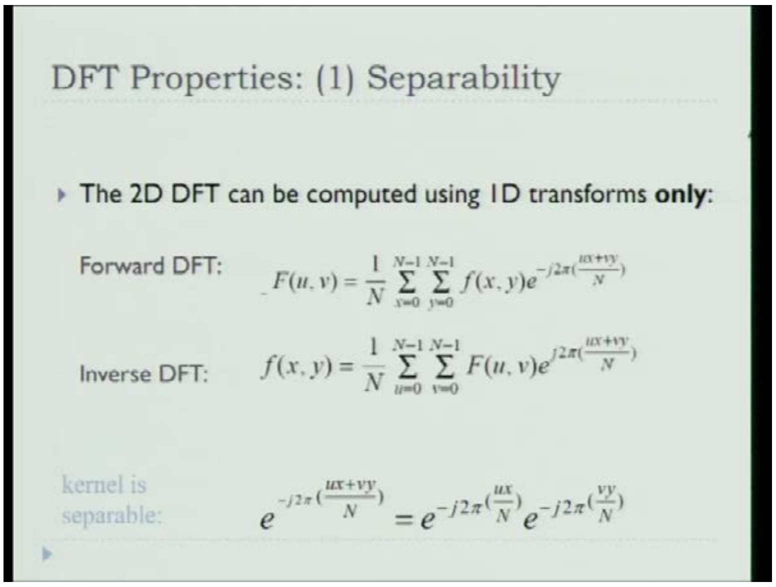
Visualizing DFT

- Typically, we visualize |F(u,v)|
- The dynamic range of |F(u,v)| is typically very large
- Apply stretching: $D(u,v) = c\,log(1+|F(u,v)|)$ (c is const)

original image
before scaling
after scaling

Now if you visualize that F of u v you observe that range domain for this u v becomes very, very large. Now, if you use the sum scanning factor on it, then you can realize the effect.

(Refer Slide Time: 37:19)



DFT Properties: (1) Separability

- The 2D DFT can be computed using 1D transforms **only**:

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

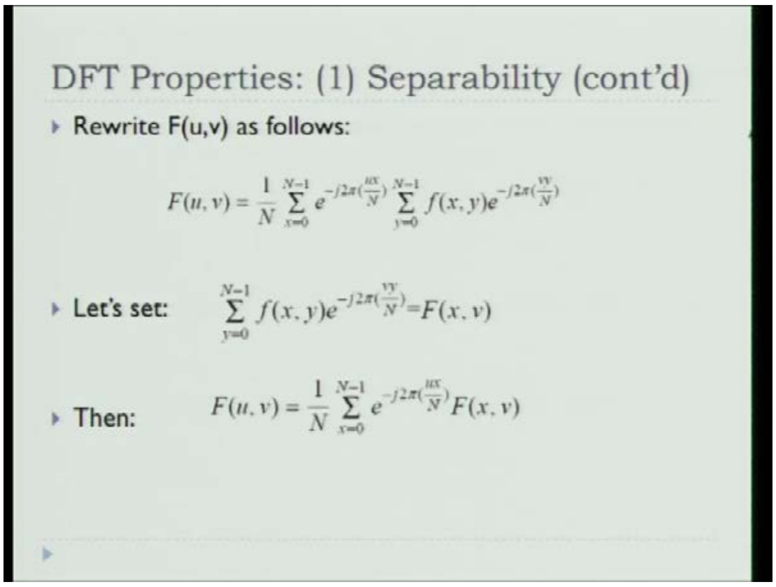
Inverse DFT: $$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux+vy}{N})}$$

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

Now, certain properties that 2 D DFT can be computed using 1 D DFT. Say, you have F of u v is equals to 1 by N summation over x 0 to N minus 1 y 0 to N minus 1 f of x y e to the power minus 2 pi u x plus v y by N and inverse is also correspondingly written.

Now, you look this one can be expressed into this, the product of the 2 e functions if it is the case.

(Refer Slide Time: 38:00)



DFT Properties: (1) Separability (cont'd)

- Rewrite F(u,v) as follows:

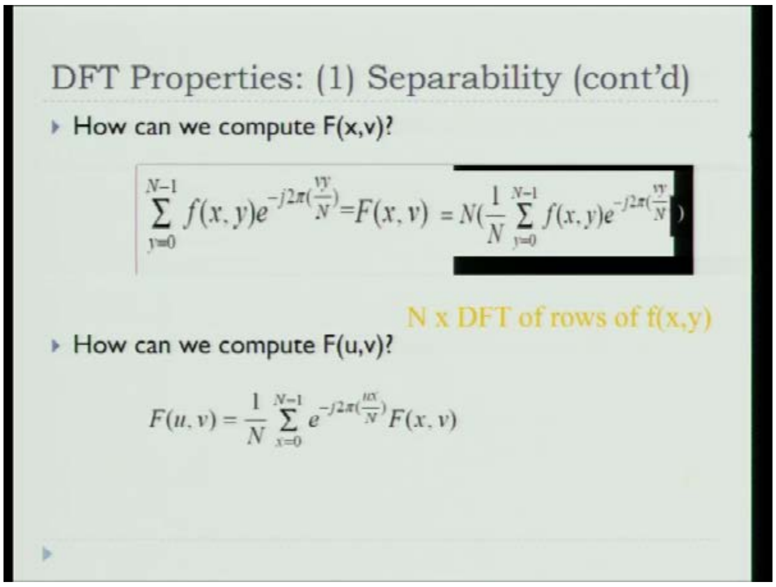
$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} e^{-j2\pi(\frac{ux}{N})}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{vy}{N})}$$

- Let's set: $\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{vy}{N})} = F(x,v)$

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

So we can rewrite your F of u v as the product of the two components. Let us assume, that this is equals to f of x v then F of u v is nothing but 1 by N summation over x equals to 0 to N minus 1 e to the power minus 2 j pi u x by N F of x v.

(Refer Slide Time: 38:29)



DFT Properties: (1) Separability (cont'd)

- How can we compute F(x,v)?

$$\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{vy}{N})} = F(x,v) = N\left(\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{vy}{N})}\right)$$

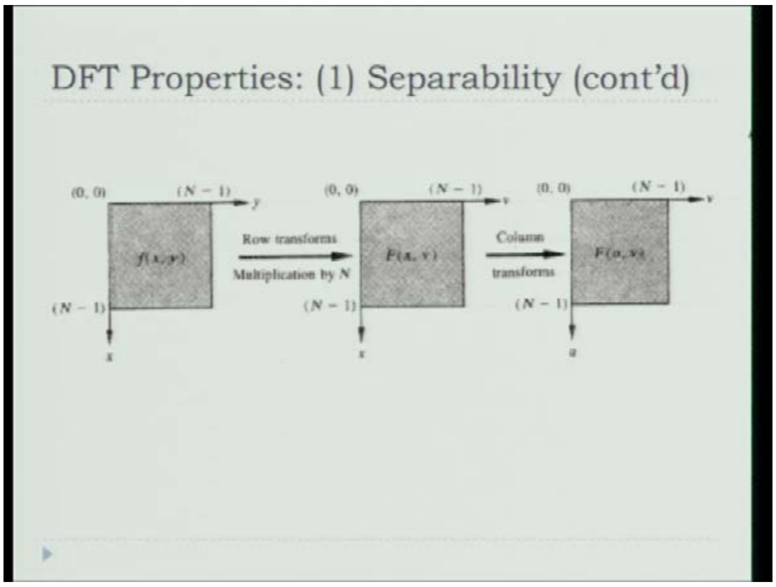
N x DFT of rows of f(x,y)

- How can we compute F(u,v)?

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

So, so you have summation y equals to 0 to N minus one f of x y e to the power minus j 2 pi v y by N, which can be written as N times of 1 by N summation over y is 0 to N minus one f x y e to the power j 2 pi v y by N that means what N times you will be doing the row wise row wise DFT N times you will be doing the DFT of row wise of x y and once you have done it then, 1 by N summation over e to the power j 2 pi u x by N f of x v and there is only one time DFT will be computing along column wise.

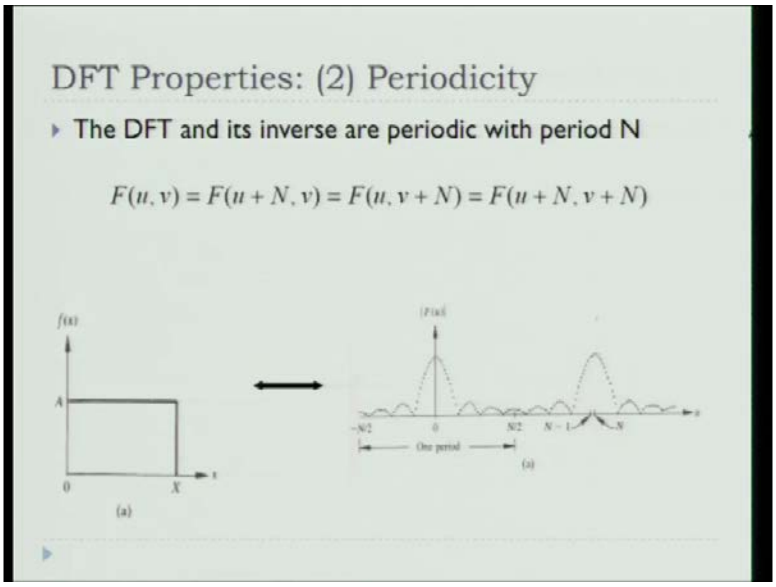
(Refer Slide Time: 39:19)



DFT Properties: (1) Separability (cont'd)

So, this is the important property of this and as a result you can have the very good or first algorithms to design or to open the value of discrete Fourier transform.

(Refer Slide Time: 39:32)



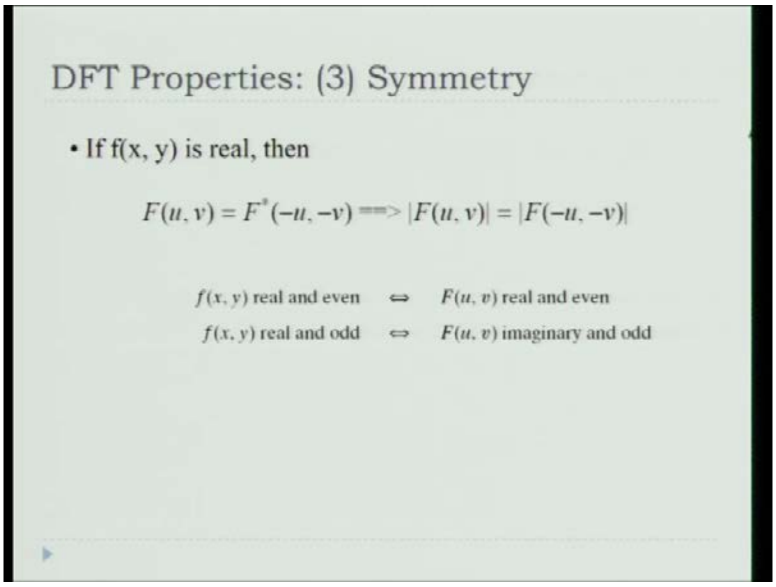
DFT Properties: (2) Periodicity

- The DFT and its inverse are periodic with period N

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

Now, DFT and its inverse are periodic with period of N. That property also you can easily visualize F of u v equals to f of u plus N v and if F of u and v plus N and or equivalent to F of u plus N v plus N.

(Refer Slide Time: 39:51)



DFT Properties: (3) Symmetry

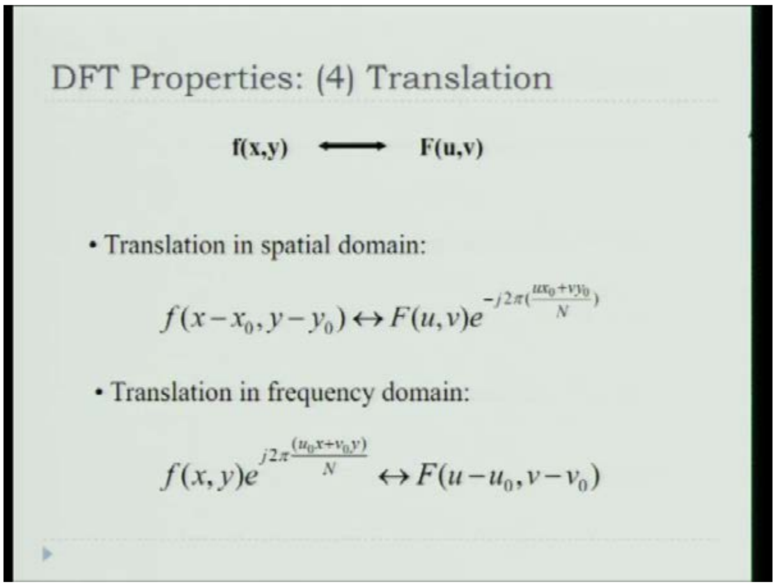
- If f(x, y) is real, then

$$F(u,v) = F^*(-u,-v) \Longrightarrow |F(u,v)| = |F(-u,-v)|$$

$$f(x,y) \text{ real and even} \Leftrightarrow F(u,v) \text{ real and even}$$
$$f(x,y) \text{ real and odd} \Leftrightarrow F(u,v) \text{ imaginary and odd}$$

It obeys the property of symmetry, we have already find what is a symmetry property. And, f of x y it is real c than f of u v you can easily show that F star minus u minus v and which indicated that magnitude of F of u v is equals to magnitude of F of u v is equal to minus u minus v. If further you can see that f of x y if it is a real and even then F of F of x u v is also real and even. Similarly, is the case with odd F of u v is imaginary and odd.

(Refer Slide Time: 40:28)



DFT Properties: (4) Translation

f(x,y) $\longleftrightarrow$ **F(u,v)**

- Translation in spatial domain:

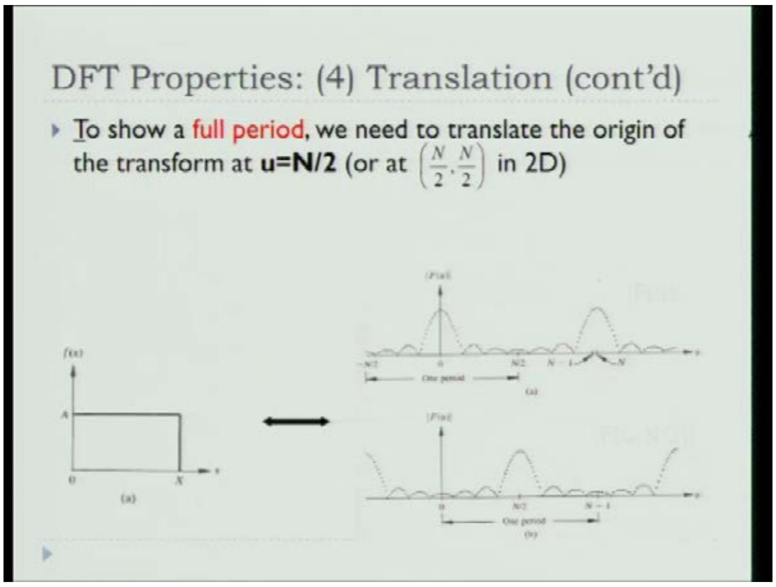
$$f(x-x_0, y-y_0) \leftrightarrow F(u,v)e^{-j2\pi(\frac{ux_0+vy_0}{N})}$$

- Translation in frequency domain:

$$f(x,y)e^{j2\pi\frac{(u_0x+v_0y)}{N}} \leftrightarrow F(u-u_0, v-v_0)$$

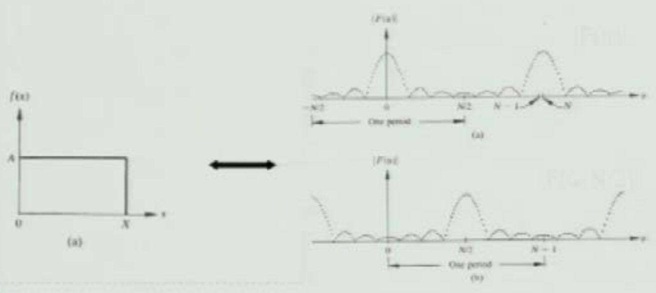
So this translation is an important property in DFT. See, in the translation in spatial domain f of x minus x 0 y minus y 0 you will be getting F of u v e to the power j 2 pi just you put the value in your original algorithm, then you will find some part of e to the power you will be there, that is a constant because it is u 0 and v 0 so you can take x 0 implies y 0 so you can take out. So, if you can do it reverse one inverse one also would be the same satisfying the same property that you have f if x y you multiply this you find the formula for F u minus x u minus u 0 and v minus v 0.

(Refer Slide Time: 41:21)



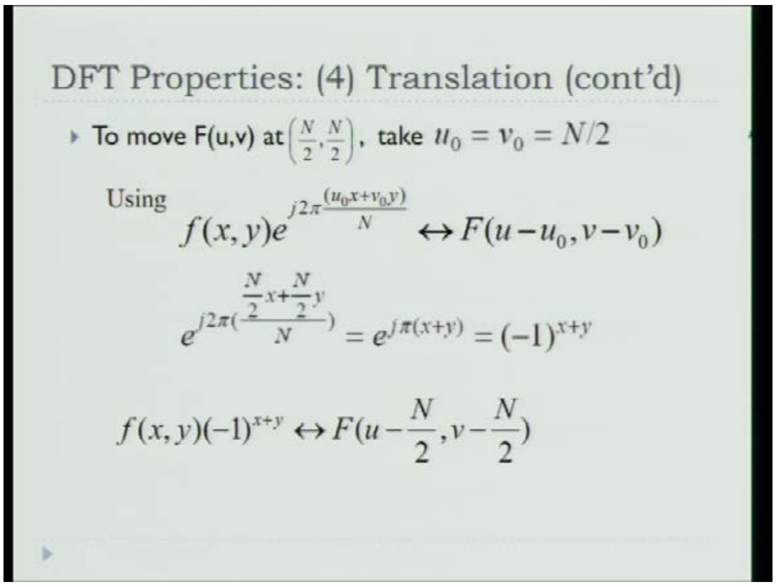
DFT Properties: (4) Translation (cont'd)

- To show a full period, we need to translate the origin of the transform at **u=N/2** (or at $\left(\frac{N}{2}, \frac{N}{2}\right)$ in 2D)



Now, to show a full period; you need to translate the origin of the transform at N by 2. This is the full period that minus N by 2 plus N by 2. Now, if you want to show the full period after shifting, after transform then you need to show the full period. So that means, you have to shift from N by 2 to N by 2. So, minus N by 2 to N by 2 that is the thing so you need to shift from minus N by 2 to N by 2 here it is 0 to N minus 1. So, that shifting is required. That means, u will be N by 2 and v will be N by 2 that has to be u 0 instead of v 0 you have to put that.

(Refer Slide Time: 42:06)



DFT Properties: (4) Translation (cont'd)

- To move F(u,v) at $\left(\frac{N}{2},\frac{N}{2}\right)$, take $u_0 = v_0 = N/2$

Using

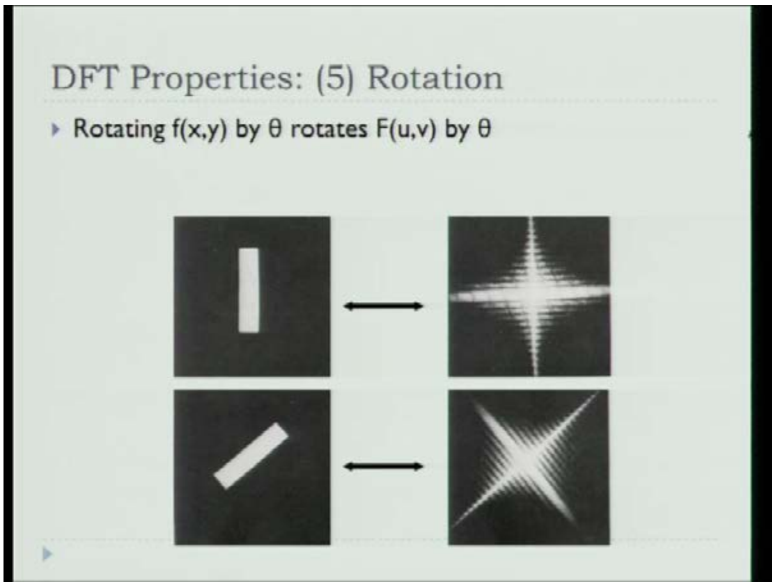
$$f(x,y)e^{j2\pi\frac{(u_0x+v_0y)}{N}} \leftrightarrow F(u-u_0,v-v_0)$$

$$e^{j2\pi\left(\frac{\frac{N}{2}x+\frac{N}{2}y}{N}\right)} = e^{j\pi(x+y)} = (-1)^{x+y}$$

$$f(x,y)(-1)^{x+y} \leftrightarrow F\left(u-\frac{N}{2},v-\frac{N}{2}\right)$$

So, to move F of u v at N by 2 N by 2 so, you will be taking u 0 equals to v 0 equals to N by 2. So, using f of x y e to the power j 2 pi u 0 x plus v 0 y by N and is a transform is f of u minus u 0 v minus v 0. Now, if you consider this part and u 0 you put N by 2 and v 0 you put N by 2 and then you are getting that it is nothing but e to the power j pi x plus y which is nothing but minus 1 to the power x plus y. So, basically you have f of x minus 1 to the power x plus y is nothing but, F of u minus N by 2 and v minus N by 2.

(Refer Slide Time: 43:01)

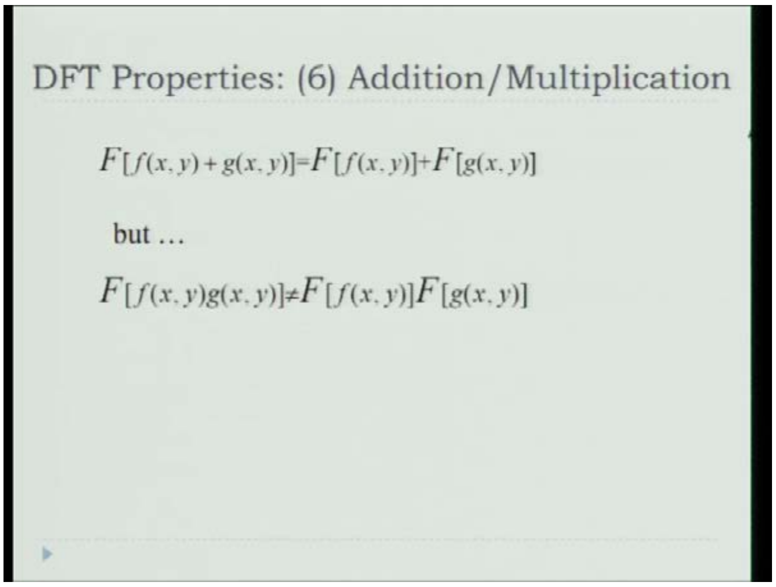


DFT Properties: (5) Rotation

- Rotating f(x,y) by θ rotates F(u,v) by θ

So that is the translation property. And now rotation, so rotating f (x,y) by theta rotates also F (u,v) by theta

(Refer Slide Time: 43:12)



DFT Properties: (6) Addition/Multiplication

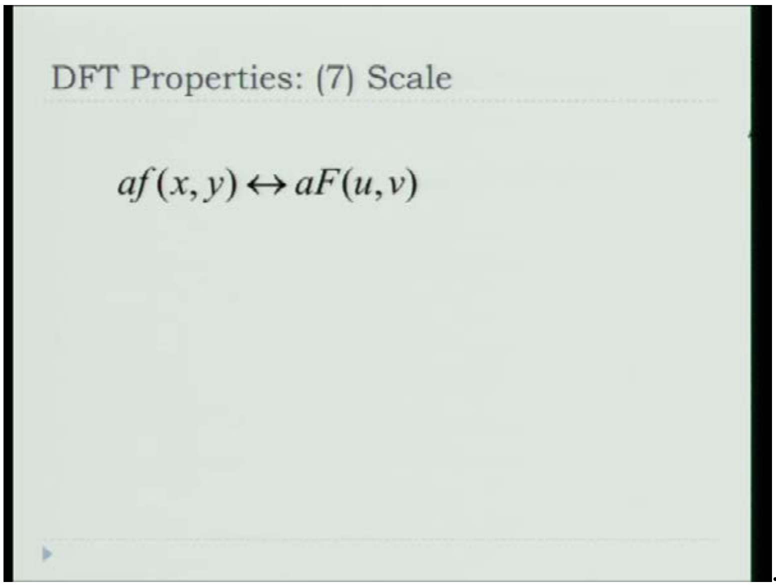
$$F[f(x,y)+g(x,y)]=F[f(x,y)]+F[g(x,y)]$$

but …

$$F[f(x,y)g(x,y)]\neq F[f(x,y)]F[g(x,y)]$$

So addition and multiplications on DFT that if you add the 2 DFT the result is also DFT and but it does not hold good in the case of in the case of multiplication.

(Refer Slide Time: 43:29)

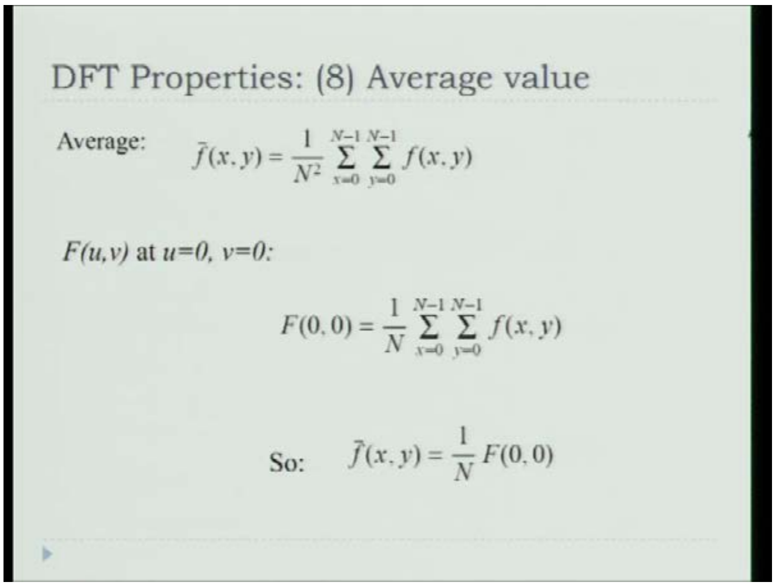


DFT Properties: (7) Scale

$$af(x,y) \leftrightarrow aF(u,v)$$

Scaling is the another one f of x y is equals to is the is the in the transform of a into F of u v, that is also a is the scaling factor if this is scaling factor. It is very straight forward if you can also put in the formula you will be easily getting it.

(Refer Slide Time: 43:45)



DFT Properties: (8) Average value

Average: $\bar{f}(x,y) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$

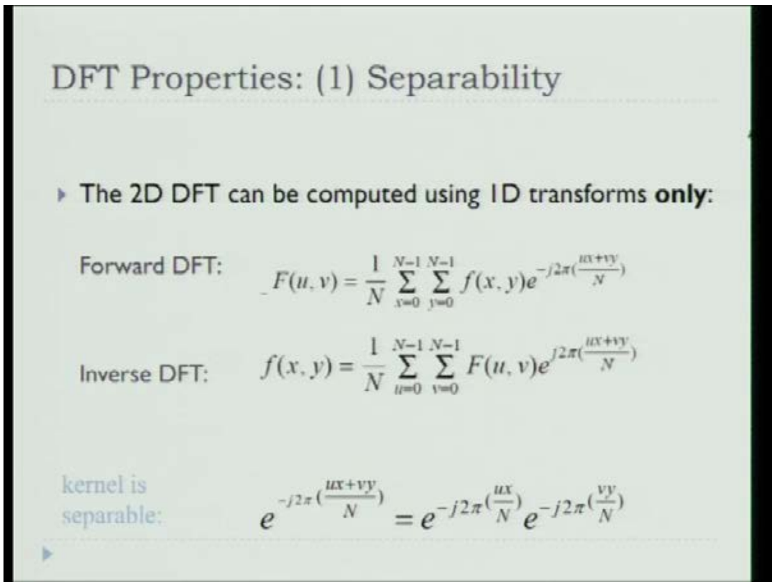
$F(u,v)$ at $u=0,\ v=0$:

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

So: $\bar{f}(x,y) = \frac{1}{N}F(0,0)$

But finally, you have the property of average f of x y is equals to 1 by f of average f of x y or f bar x y is equal to one by N square summation over x 0 to N minus 1 y 0 to N minus 1 f of x y.

(Refer Slide Time: 44:04)



DFT Properties: (1) Separability

- The 2D DFT can be computed using 1D transforms **only**:

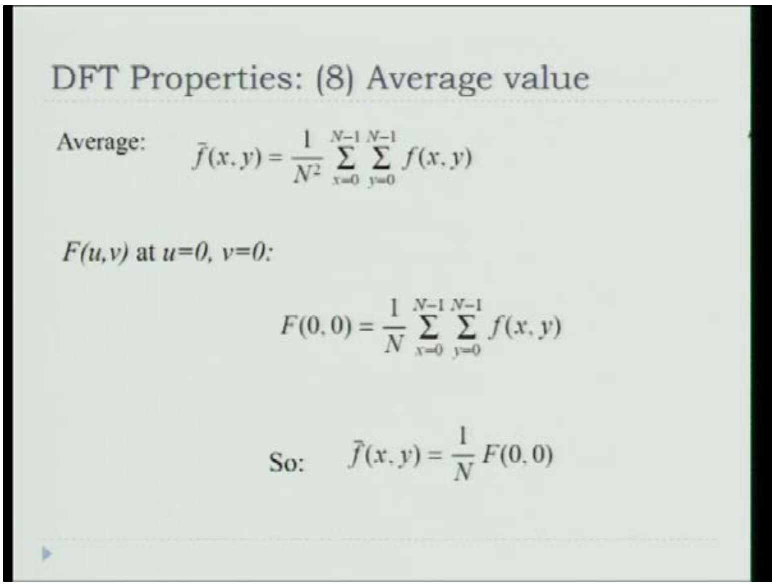
Forward DFT: $$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi(\frac{ux+vy}{N})}$$

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

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

Now, what happens that if i put u 0 and v 0? So this becomes 1 so, it becomes 1 by N summation F of 0 F of 0 0 is 1 by N summation x equals to 0 to N minus 1 0 to y is 0 to N minus 1 f of x y.

(Refer Slide Time: 44:29)



DFT Properties: (8) Average value

Average: $$\bar{f}(x,y)=\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)$$

F(u,v) at *u=0, v=0:*

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

So: $$\bar{f}(x,y)=\frac{1}{N}F(0,0)$$

So, that is the thing that F (0,0) is equals to 1 by N summation over 0 to N minus 1 0 to N minus one f of x y. If you compare these two, what you get? That f bar (x,y) is nothing but 1 by N of F (0,0). So, that is another property you get. So, these are the 8 properties you have on pure DFT.