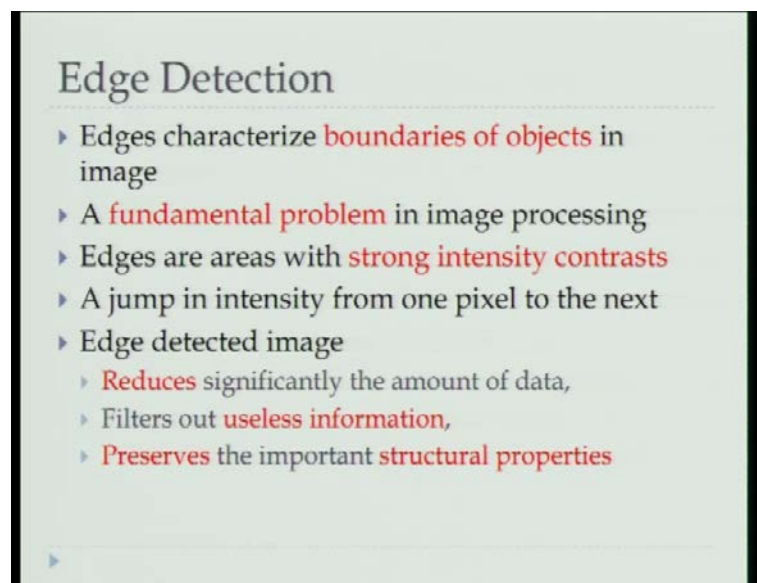


Biometrics
Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture No. # 06

We also define what is edge, actually what edge; edge is gives you the boundary of an objects in an image.

(Refer Slide Time: 00:25)



And by boundary means; suppose, you have a (()) in the (()) you can see what is eye, what is nose and but, if you see the edge; directly of all component of nose, you will not get **right**. Wherever you will find that, there is a sharp change of intensity level those part only will be visible.

So, it is the fundamental problem in which crossing it occurs, this type of problem occur very or we will be needing to solve in many occasions. Edges are areas, where you will find that strong intensity contrast or the in the boundary line you will find, there is a vast difference between the pixel value or pixel intensity value on the edge and its neighbors.

So, that edge detected image it will be a binary image type; binary image. The wherever there is an edge **wherever there is an edge** you will get say, dark pixel and rest of the

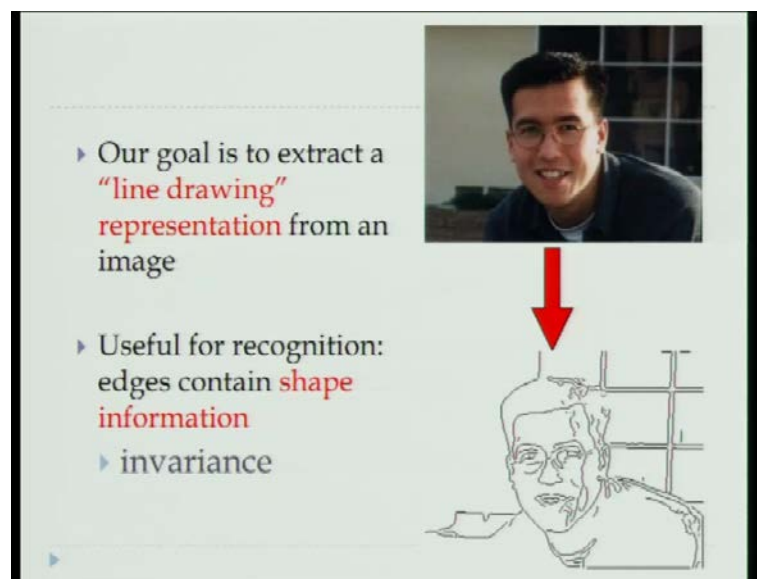
thing you can make it white to get the skeleton of the image. So, if since it is a 0 1 type thing; it reduces the space **you need to** you do not need to store all the information, only you can keep **you can keep** the dark pixels.

Now, you remember that, you have a image 1000 cross 1000 say image size and there are dark pixels and white pixels. So, you want to store I am claiming that, volume of data will be reduced substantially; only one way you can reduce, because you can use a big vector, so 8 pixel positions you **(())** can **put** put in **(())**; so space will be reduced by one-eighth.

So, another way you can think you are shown that is a **(()) right**, because 1000 cross 1000 means say 10 to the power 6 locations and may be 2 percent percentage is there. So, why to store all the 0 values there only one value along with along with row major information row and column information.

Since you are retaining only the edges, there will be certain noise and other things in any image, those things will be suppresses by your method that has to be taken into account. And it retains the structural property that is the most important thing in the edge, that structure will not get changed, because you are keeping the boundary informations. So, shape of the image is retained.

(Refer Slide Time: 03:22)



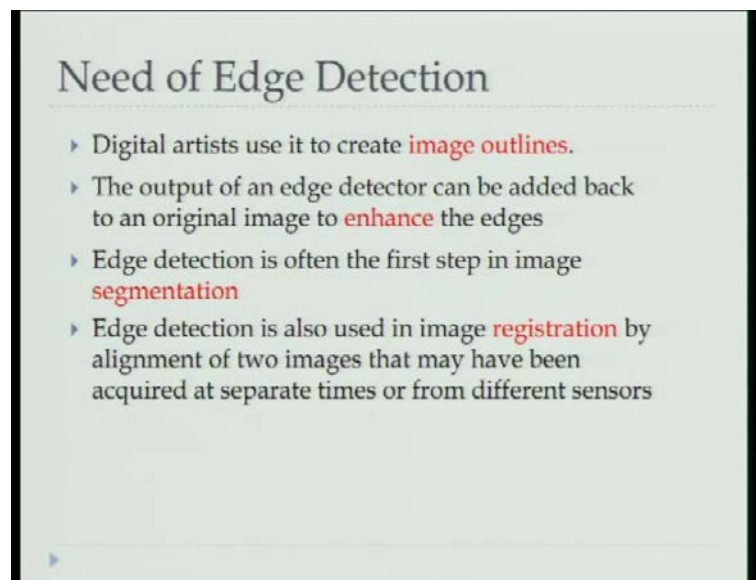
- ▶ Our goal is to extract a "line drawing" representation from an image
- ▶ Useful for recognition: edges contain shape information
- ▶ invariance

So, what is your aim? Your aim is to draw a line or line drawing representation of an image **right**. Now, once I give you an image and if I tell you that, draw a line over the image; only the boundary area only you will be drawing the line, rest of the things you will not be touching **right**.

And as you are drawing the lines, so you are retaining the shape informations. Once you are retaining the shape informations, whether it is a geometrically you increase decrease whatever you do; the shape will be remain would be kept same. So, shape is invariance; shape invariance. Another thing is that give, if I have the line drawing representations I can put the color on it to get back to my original image.

Now, one of course, you will not get back the exactly same image, but similar looking image you can find out **right**. Say, suppose I have a line drawing image and now I want to put the how I look if I put the red color shirt on it. So, you can put the shirt design and color it and you can find out, how it looks. So, all those things possible on the edge detected image.

(Refer Slide Time: 04:49)



Now, why you **you** need the edge detection algorithms. So, we need from the edge detection you will be getting what? Image outlines. Now, once image outline is available as I told you that, you can do lot many work; one is that, as I want to put the texture color on it or if I want to put some other information on this suppose, I want to put a mole and check, how it looks or you can do also **(())** things.

Then, you can enhance the image how what I do, I will put the boundary lines, I put the very dark color on it and then, I super impose on my previous original image, then the boundary will be more sharpen, more highlighted. Second one is that, it is very useful. This one example I can give you recently, what we are trying to solve I have the vein data; vein data means hand vein I have taken the data.

Now, you have the flash on it and the veins are inside the flash, it is not visible. So, but small lines are available you check that lines are available. Now, I want to get the features of these veins; how can we do it, because whatever way you do it, that (()) also coming in between, that this features of this skin also coming in between, I do not want the features of the skin, because I want the feature of the vein.

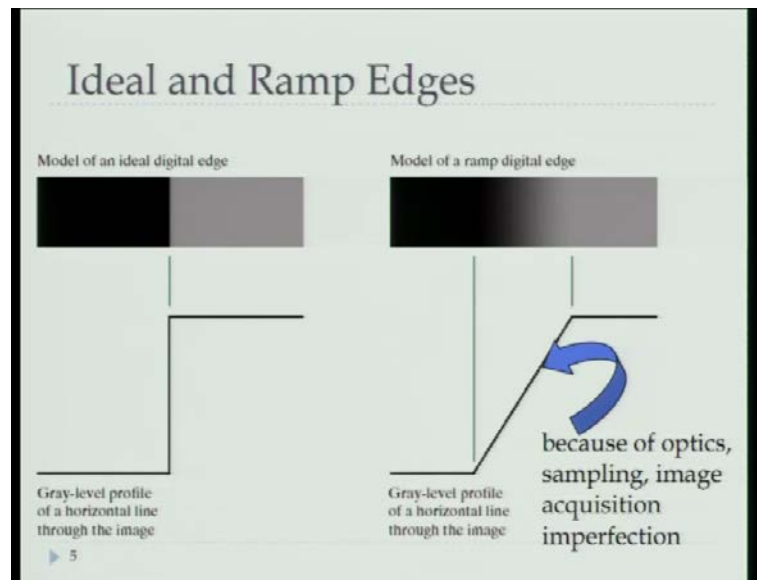
So, what we are doing that on the gray scale image; we obtain the edge of this veins and that super imposed on it. Now, the vein information is more highlight and then, your skin and other (()) other things. Now, we are extracting the feature from there, now there are some different feature extraction techniques. So, we are trying to get some feature extraction from there and we are getting it very good way.

Next one is that segmentation. See I think in the last class also I gave one example that if I have the edge of be boundary of some objects then, I want to get only the object I do not want the background or I want to get the partial objects; say it is a composition of two objects, I want one objects; the other objects can be eliminated. So, I can separate them out through the edge detected from the edge detected image.

And next one is more useful one is the registration image registration; there I have the two structural property or two structure. Now, I want to register them register means I want to super impose some part of it, so the structure is there. So, what I do, that I have the two traced image; just I put one on the top of the, another one to see that whether they are matched or not, that is known as registration.

So, these three are very important of potentials. We generally perform on image and if I have the edge detected image, then it is becoming very easy to implement.

(Refer Slide Time: 08:14)



So, this is an ideal image or this is the ramp image, last class also we have shown this image, that this is a very dark image. As suppose, this is the white part, because to show the white part I can put, so I put some gray scale here.

So, or you can have the but, this color is same and this gray value is same. So, the gray values are same here suddenly jump and the another gray values will be there; they are same at intensity level. So, this is the ideal way to find out the edge of the of the object or the boundary of the object. But, in reality you will not get; in reality you will be getting like this, that you have the dark image suppose one color, then suddenly it gets changes finally, it achieve to the next color intensity level.

So, you will find there is a ramp, slope; slope indicating the changes of intensity levels. Now, these changes may not be due to the optical characteristics may be due to the (()) you have done, due to say you have done the average filtering on it to remove the noise. And as you know that as you are doing the average filtering that, there is a (()) of blurring would be coming and that blur will give you will be effected in your edge also. And that edge will become blur; that means, the texture; that means, there will be a the effect will be here as as a result this ramp will be created.

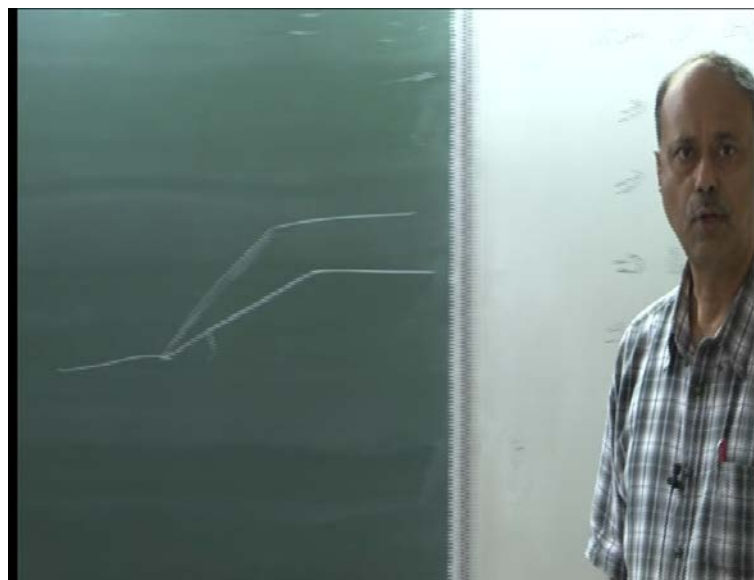
Now, the (()) of once it is a black; you cannot think that, it is a single (()) change of intensity; one picture has intensity x immediately you will not get the another pixel, which is having y intensity its slowly change. So, once slowly change with that image,

edge will be thick ((C)) it is not on one pixel thick image or thin image. Now, what is the claim? The slope of the ramp is inversely related with the degree of blur, if the slope is very high that 90 degree; that means, sharp change one pixel will get immediately sharp change from the previous one and there is no blurring effect, it is a sharp change.

And similarly, if the slope is very less, then the pixel intensity also will be changing slowly. So, that is why it is telling that that, it is slowly inversely proportional. So, more the slope more the slope, less the effect of the blur and vice-versa. And as you have the slope; obviously, you will not get the thin edge. So, what is the edge point? Edge point is nothing but, a point on the slope or on the thick edge, that is your edge point on the ramp. Edge point is a point, which is lying on the ramp.

Now, the set of such points will form an edge, provided they are connected right. Set of these points will form an edge, provided they are connected. Now, thickness of an edge is depending on the length of the ramp. Now, again length of the ramp is dependent on the slope of the ramp, because and slope of the ramp is dependent on the blur blur blurring factor.

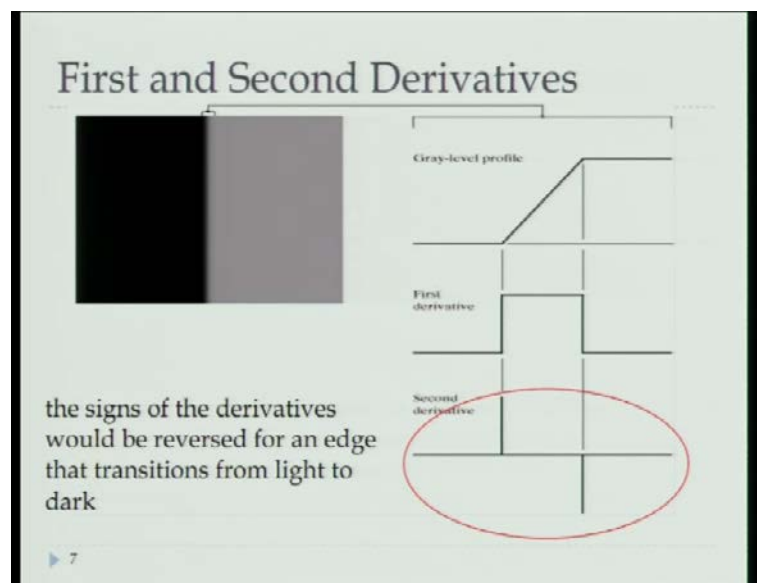
(Refer Slide Time: 12:04)



What it means that, if this is the structure; what it means that, this thickness of the edge here from here to here that is the ramp blank.

And this is this is nothing but, **the** that amount of slope you have. And as we have mentioned previously, that it is inversely related with the blurring effect. So, it depends on the blur also, suppose this slope is like this; that means, thickness of the edge will be this much. And thickness of this edge will be this much means; that slope of this edge will be creating that information as slope is reducing; **thickness is** length is increasing and as length is increasing; thickness is increasing, is it clear.

(Refer Slide Time: 13:11)



So, the blurred edge tend to thick, while the thin edge; indicating the sharp edge; indicating the thin, again this I do not know why it is happening. See now, if I take suppose this is an slope, then first derivative will look like this.

(Refer Slide Time: 13:34)

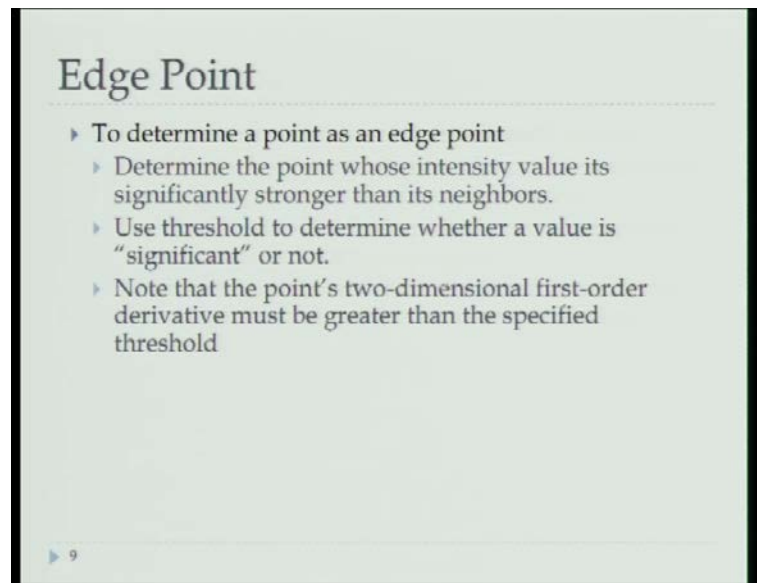


And while I take the second derivative; something like that **right** (Refer Slide Time: 13:48). So, basically in the second derivative case; you see that you get the two points; one is towards positive, it need not be this is a positive, this can be negative and this can be positive depending upon the slope downwards slope or upwards slope. So, you get the two points.

Now, if I draw an imaginary line. So, this point is known as zero crossing points and this is very powerful, that will give you the center of the edge thickness. See, because I **I** cannot have generally if I take the whole thick image edge **edge** image, then **you know** you will be losing the effect of **effect of** visual edge. What you **you** need; you need a thin edge; line on the image. How to get the thin edge line, then you take the center of **the center of the center of** this point center of this edge thick edge, that will give you the thin edge **that will give you the thin edge**.

So, zero crossing will help you to determine the center of the thick edge.

(Refer Slide Time: 15:09)



Now, how to get an edge point so; obviously, you will be looking for a point in the image, which has the higher intensity value compared to the neighboring one; that is the basic criteria. That I will tell this an edge point provided it has the intensity value, which is higher than **higher than** the neighboring pixels. Now, once you determine that higher than the neighboring pixels; you need to know, how much significant it is.

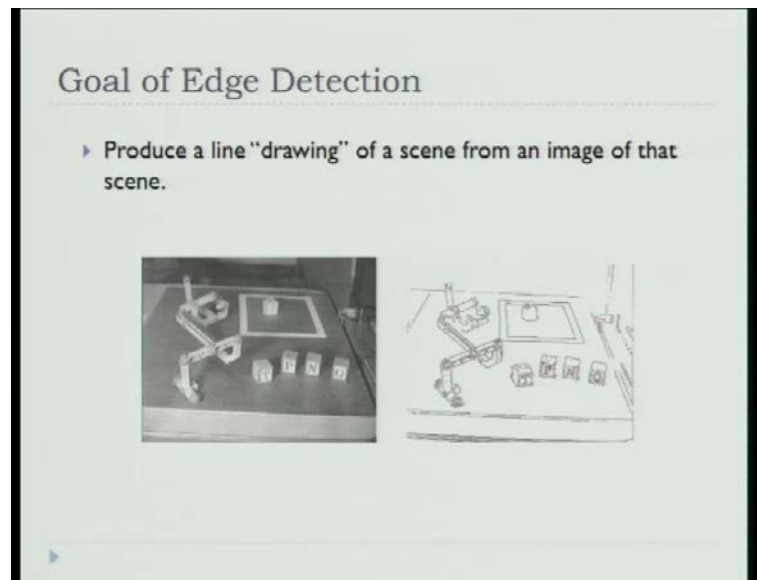
Significantly higher word I will be use I think I have used the word **yes** significantly stronger. Now, this significant stronger will be dependent on a threshold value; you have to decide a threshold value by which you will be telling, if it is higher than that one, then you will be considering **it is a** it is a edge point **it is an edge point**.

Now, how to determine the threshold value; one possible way is that, you take the average of the image intensity values, that can be your threshold value. And if it is above the threshold average and you tell that this is an edge point, this is one way you can think. But, issue is that since you are taking the average one; there is a possibility there is a lot many points may come out as an edge point, which may not be true.

So, there are several ways you can determine the whether that is a significantly larger than the threshold value or not. Besides that, if I see think about the first derivative. Now, you get this type of thing will be there, any point higher than this; you can tell that this is an edge point. So, that is the thing the points two dimensional word I will be coming later on points first order; first order derivative must be greater than some

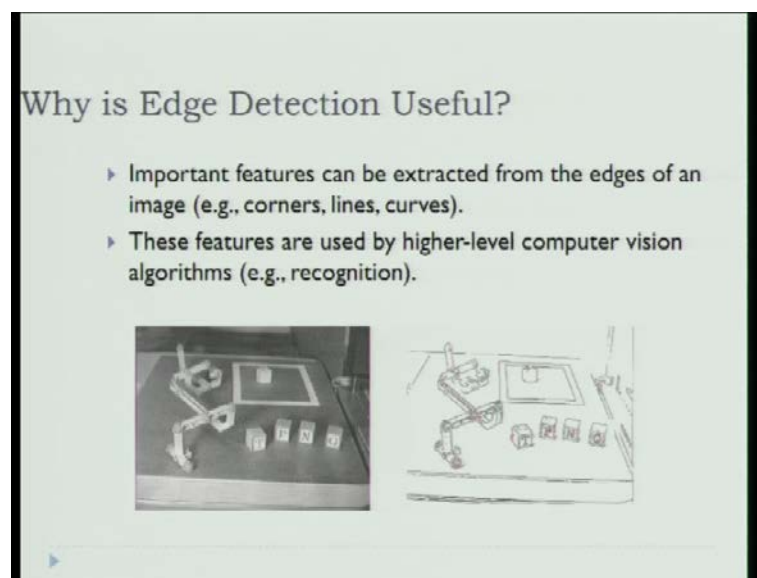
specified threshold. So, if you take the derivative then, this must be higher than the threshold value, that is the thing. Now, why two dimension, we will come to the discussion, while we will be discussing later, that two dimensional effect another time.

(Refer Slide Time: 17:34)



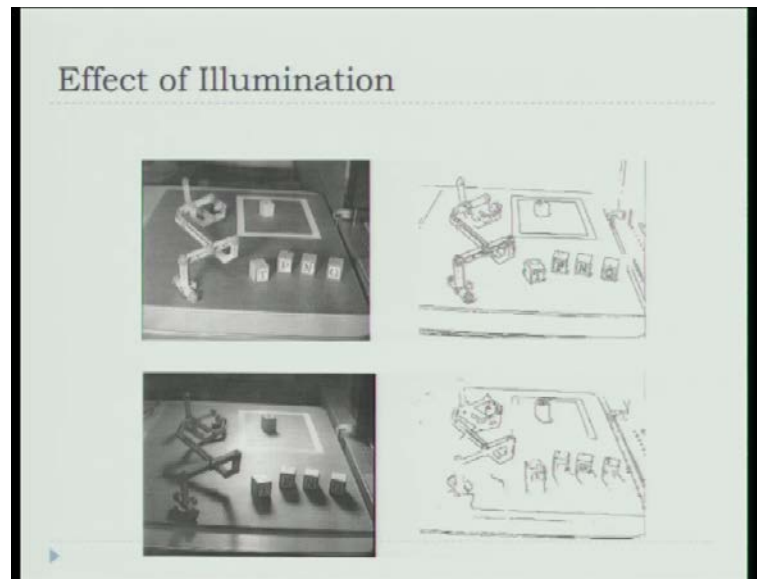
So, this is an edge detection structure that you have an image and you see that, if I use determine these boundaries, then I can easily visualize all the objects. There are some cases, you will find that object boundaries are not visible at all. But, while I will be using the edge detected image, then it is possible to visualize all the things.

(Refer Slide Time: 18:03)



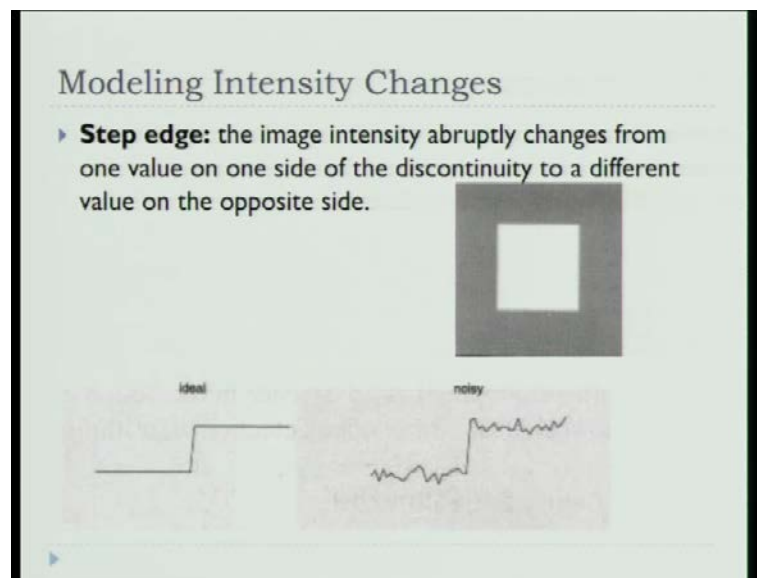
Let us see here corner points are creating problem.

(Refer Slide Time: 18:07)



See, this is the case where there is a illumination effect is there and since it is a dark image; say here, the object boundaries you are missing. So, you have to do certain operations to see that, no all objects should boundary should be available to you.

(Refer Slide Time: 18:25)

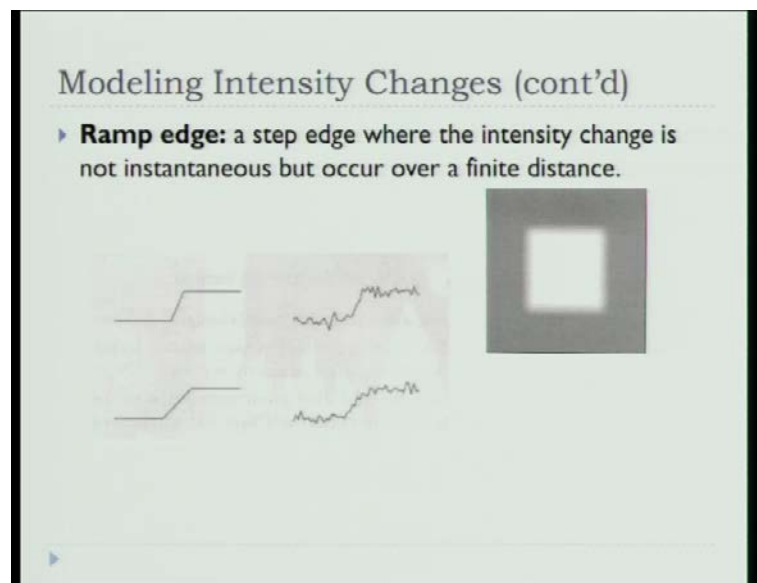


Now, what are the different types of edges you can visualize? One is known as the step edge step; edge is nothing but, that you have the constant intensity value and suddenly it

jumps to the higher or lower intensity value and again it is retaining the constant, almost constant intensity values.

So, this is the an ideal situation; constant intensity value suddenly jump to the higher one and again retaining. But, in reality you will not get this one, there will be a noise in your image, so it will look like that, one if you have the noise. Say, example is that a binary image you have 0 and 1 only, then you will get this type of 0 and 255 say, then you will get this type of sharp step edge.

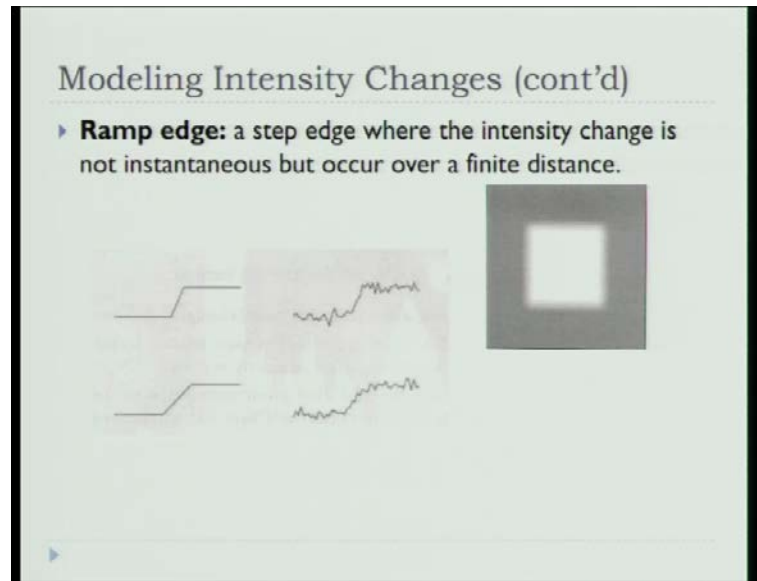
(Refer Slide Time: 19:20)



Ramp edge gray scale image I gave you one example in the first class or second class of my lecture that, you have the **gray** gray scale image and in the gray scale image you will not find that this is 255 and this is 0; you will get there a certain slowly changing the color from this to this and as a result the ramp will come out and if you have the noise; the structure will look like this.

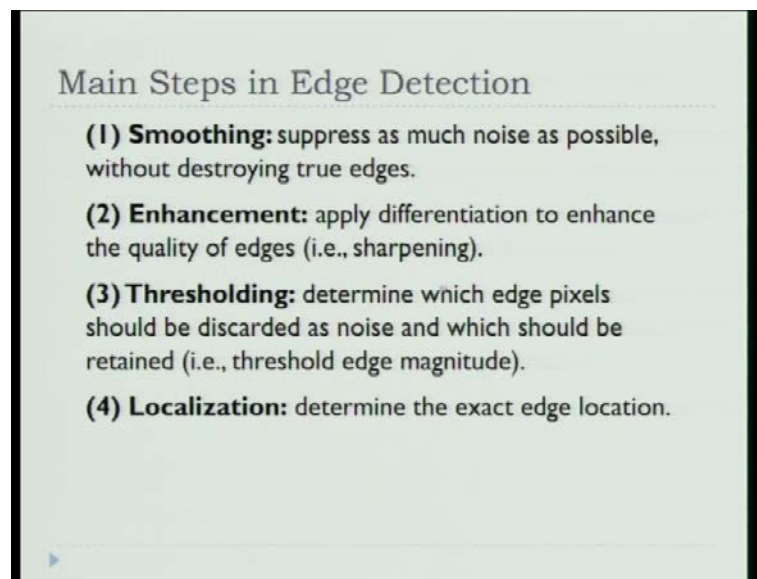
Then, the ridge edge is you will get very frequently, what happens you have the constant intensity; suddenly it jumps and again come back and again **again** it is retaining the constant intensity level.

(Refer Slide Time: 20:11)



So, it may look like this one or this one and with noise you will have the structure similar to this, there is a term roof edge it is also ridge edge, but it is not suddenly sharp small time it comes back; it is it retains for a little longer duration then, comes back the again constant intensity level. So, roof edge belongs to ridge edge only. This is thing only issue is that here, it **it** is little longer period.

(Refer Slide Time: 20:43)



Now, how to detect the edge that is the most important thing. If I know the steps of detecting the edge; rest of the things will be taken care. You have an image it must have

an must have noise. So, you must be able to first remove that noise otherwise, if you do not remove the noise and if you introduce a next step, which is known as enhancement; the noise also will get enhanced. And if noise get enhanced; the noise also will show as an edge. So, I do not want noise should be consider.

So, first thing is that, you smoothen the image; that means, you suppress the noise. Now, if you use any noise removal algorithm or any filtering algorithm to suppress the noise; obviously, image will become blur; once image will become blur, then you need to enhance the image. So, you will be using some enhancement technique to enhance the image. And once you have the enhanced image now you have to you know that by seeing the enhanced image; you will find the boundaries are more highlighted compared to the background one, that is the reason of enhancement.

So, through thresholding; you can get the boundary and rest you can suppress right. Now, you get only the thresholded elements and these elements will form the your edge map. Now remember that, once you do the thresholding; there is a chance there is a chance of missing link. What happens? That, you have suppose you have an object like that, and once you do the edge detection after processing the first, second, and third steps; you may find that, these are missing; these links are missing because, it is dependent on the (()) threshold. And (()) threshold you cannot satisfy, if you bring down the threshold level; you will find several other noise are coming some other, several small small lines will be coming.

So, this link will be missing, because of the threshold value. If threshold I take lower than the desired one, you may find extra lines will be there, which you do not want, but if you increase the threshold level higher than that; then there is a possibility some lines will be missing. So, there is a need of introducing something which is known as edge linking that, if there is a small difference between the link, no link between this distance very small then, you add it; some technique is there line fitting or some there several examples are there.

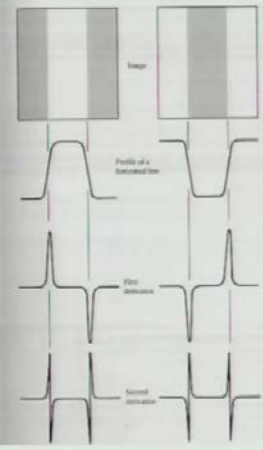
So, these are the four steps involve in it. First one is smoothing, second one is enhancement, third is thresholding and fourth is localization.

(Refer Slide Time: 24:08)

Edge Detection Using Derivatives

► Often, points that lie on an edge are detected by:

- (1) Detecting the local **maxima** or **minima** of the first derivative.
- (2) Detecting the **zero-crossings** of the second derivative.



So, the points edge point can be determined, either by the local maxima or minima through or maxima minima of first derivative or the through zero crossing **right**. That, as you know that, this is first derivative effect will be there. So, this local maxima and minima will give you the whether it is **whether it is a** edge point or not. And similarly, if I have the second derivative then, just a line artificial line if you draw between these two; you will be getting a zero crossing point, that will become your edge point.

(Refer Slide Time: 24:52)

Edge Detection Using First Derivative

1D function

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

➡ mask: $[-1 \ 1]$ (not centered at x)

mask $M = [-1, 0, 1]$ (centered at x)

Now, let us consider that it is an one dimension edge, because image is a two dimensional picture, but let us consider it is an one dimensional image. And then, in that case, I have to obtain the first derivative, which is the formula is a straight forward formula; and you get **you get** $f(x+1) - f(x)$, if h is equal to 1. So, now mask becomes minus 1 and 1. Here x is not at center that for x is that the point, where you will be computing the whether it is an edge point or not. To make it an x center, then it becomes minus 1, 0, 1 **right**, that is your mask. So, these are the different types of edge we have.

(Refer Slide Time: 25:41)

Edge Detection Using Second Derivative (cont'd)

(upward) step edge

S_1			12	12	12	12	24	24	24	24	24
S_1	\otimes	M	0	0	0	0	-12	12	0	0	0

(downward) step edge

S_2			24	24	24	24	12	12	12	12	12
S_2	\otimes	M	0	0	0	0	12	-12	0	0	0

ramp edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	\otimes	M	0	0	0	-3	0	0	0	3	0	0

roof edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	\otimes	M	0	0	0	-12	24	-12	0	0	0	0

This is an example do not consider as an image. Say, values are 12, 12, 12, 12, 12, 24, 24, 24; that means, you have an edge 12, 12, 12 and then 24, 24, 24, like that it is there. So, this is the intensity level 12, 12 and this is for 24. And if I use that minus 1, 0, 1 on this mask is minus 1, 0, 1 then, you will find this is your first derivative effect. And this is the downwards step that is 24, 24, 24 suddenly it has come down, so it is a downward step edge and resultant will be 0, 0, 0, 0, minus 12, minus 12 and then 0, 0, 0, 0.

This is the ramp edge where it is increasing 12, 15, 18, 21, 24, 24, 24. And this is an ridge edge, where it is 12, 12, 12, 12, 24 and again 12, 12, 12 and so on. And the corresponding convolution or convoluted image becomes S_1 , convoluted with M it gets this result **right**.

(Refer Slide Time: 26:51)

Edge Detection Using Second Derivative

- ▶ Approximate finding maxima/minima of gradient magnitude by finding places where:
$$\frac{d^2f}{dx^2}(x) = 0$$
- ▶ Cannot always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

Now, think about the second derivative one that approximate finding the maxima and minima of the gradient magnitude by second derivative is d^2f/dx^2 , this should be I will correct it equals to 0, but finding the 0 or finding that point, discrete pixel point it may not be always possible.

So, what we do? We obtain the approximate value through zero crossing, because just these two points and I am adding it, but it may not be exactly satisfying this pixel, it will be **(())** one.

(Refer Slide Time: 27:31)

Edge Detection Using Second Derivative (cont'd)

1D function

$$f''(x) = \lim_{h \rightarrow 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) = \frac{f(x+2) - 2f(x+1) + f(x)}{h^2} \quad (\text{centered at } x+1)$$

Replace $x+1$ with x (i.e., centered at x):

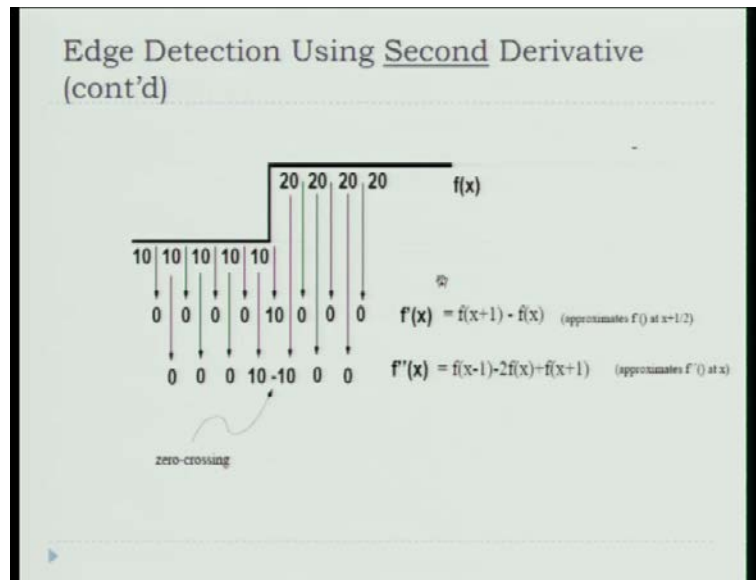
$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

↓

mask: [1 -2 1]

And if I have the one dimensional image, you will be getting the value $f(x+2) - 2f(x+1) + f(x)$. Now, you observe that is centered $x+1$. So, I just shift it, $f(x+1) - 2f(x) + f(x-1)$, now it is centered at x and the mask becomes that 1, minus 2, 1.

(Refer Slide Time: 27:59)



So, this is an example you have the all 10 and suddenly all 20's, this is the first derivative all 0's, then 10, then 0, 0, 0. But, this is $f''(x)$ derivative one and that zero crossing is lying between these two **right**; one of them you have to pick up as your zero crossing. Now, because it is not as I told you that, it is not at $f''(x)$ it will give you that in between something in between is not possible you have to select one of the integer. So, one of the things you will be considering.

So, here the mask I have used that 1, minus 2, 1 or what **yes**. So, this mask I have used then, if I use this mask; that means, this plus this minus two times of this, this becomes 0. Then, this plus this minus 2 times of this, this becomes 0. Similarly, if I consider this one, this plus this plus this, this becomes this **right** and so on.

(Refer Slide Time: 28:58)

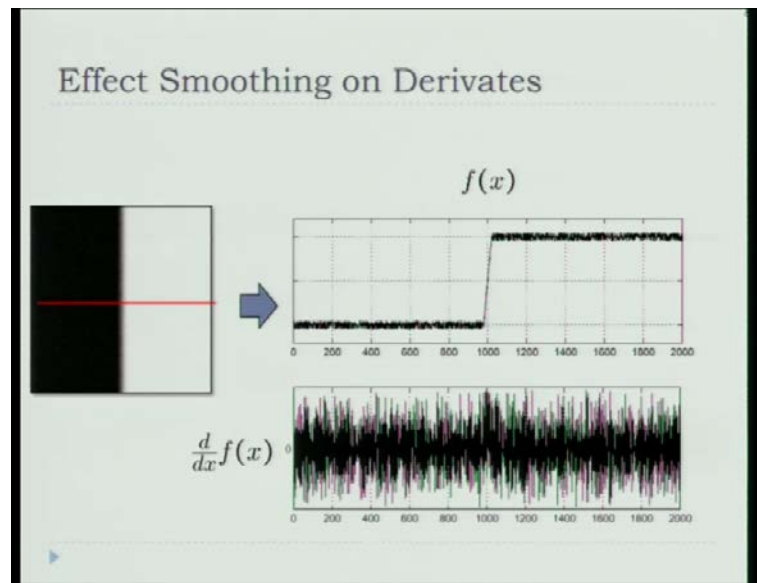
Edge Detection Using Second Derivative
(cont'd)

- ▶ Four cases of zero-crossings:
 $\{+,-\}, \{+,0,-\}, \{-,+\}, \{-,0,+\}$
- ▶ **Slope** of zero-crossing $\{a, -b\}$ is: $|a+b|$.
- ▶ To detect "strong" zero-crossing, threshold the slope.

So, if you analyze these things, thus you will be finding that zero crossing area will come at this structure, either it is plus minus or minus plus or plus 0 minus or minus 0 plus.

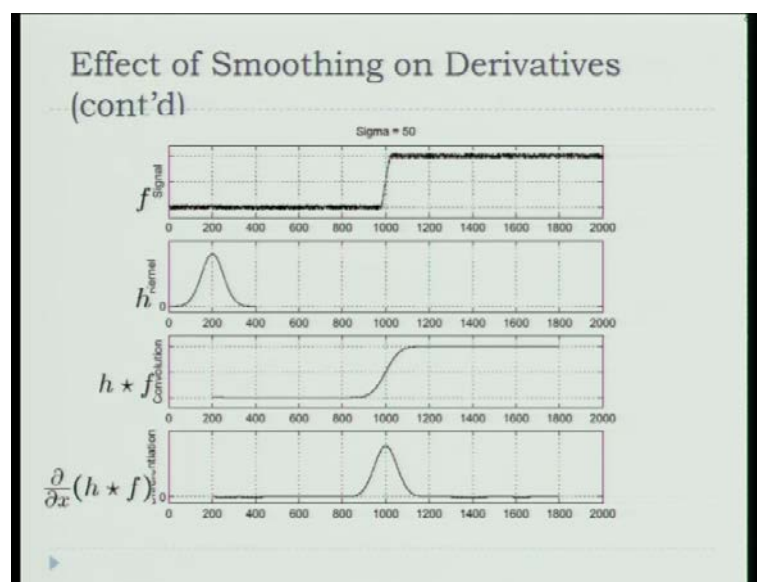
For example, if you go back to the previous one; the zero crossing is minus and plus or plus minus area or minus 0 plus or similarly, the reverse one. So, these are the possible area, where you will be getting your zero crossing. Now, how to get the zero crossing is the because **you now** you know **the now** area, where your zero crossing is lying **right**. The slope of zero crossing you can obtain, the a and minus b just you add it, that will be your slope and that to you will be using a threshold value to detect, what is the pixel zero crossing areas, from zero crossing point.

(Refer Slide Time: 29:57)



So, the standard way I will come to that later on. Now, suppose you have the effect of smoothing the derivatives; you have that image you have though you are looking you are feeling that, this is a dark and white, but in reality the image is like this; that you have it is lot of noises and pixel changes are there. Now, if I do the derivative of this, the derivative image will look like this.

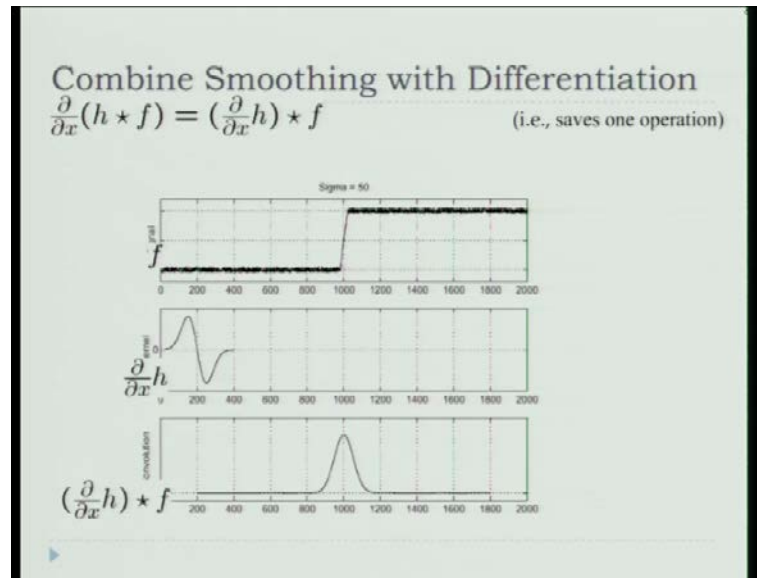
(Refer Slide Time: 38:28)



Instead of doing that, that your image like that. So, I am using some filter, Gaussian filter here. So, h into f the image will like like this, this is a smooth one, but obviously, blur

will be there, because here it was a little short, but here it is a blurring effect is there, but you will be getting it. Now, if I since I have use this also, if I take the derivative one I must get back this one. So, I get a smooth image normal image or Gaussian image.

(Refer Slide Time: 31:01)



But, if I change little, now you have the original image like this, instead of using $\frac{\partial}{\partial x} f$ effect it is equivalent sign approximate sign, then this I am doing into f , what I am doing that this is $(\frac{\partial}{\partial x}) f$, this is an original image. And I have taken the derivative of your filter, this will look like this, I have multiplied this one with f then, it looks like that it is similar Gaussian image, but it is an approximate one; it will not get back exactly, because you know obviously, you have not considered this one and you have right you have some part you have you have not added. So, this is the original one and changed one, but it also; obviously, the similar properties.

(Refer Slide Time: 31:45)

Edge Detection Using First Derivative (Gradient)

2D function:

- ▶ The first derivative of an image can be computed using the gradient:

$$\nabla f$$
$$\text{grad}(f) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Now, this is an one dimensional image you have considered, but in reality; it is not the case, because the image is a two dimensional one and image is a two dimensional one and so you have to do along x axis and along the y axis **right**. So, del f is del f by del x and del f by del y, you have to do or you have to obtain.

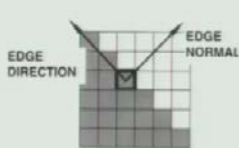
(Refer Slide Time: 32:10)

Gradient Representation

- ▶ The gradient is a vector which has **magnitude** and **direction**:

$$\text{magnitude}(\text{grad}(f)) = \sqrt{\frac{\partial f^2}{\partial x} + \frac{\partial f^2}{\partial y}}$$
$$\text{direction}(\text{grad}(f)) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- ▶ **Magnitude:** indicates **edge strength**.
- ▶ **Direction:** indicates **edge direction**.
 - ▶ i.e., perpendicular to edge direction



And using this you will be defining the gradient, which is a vector and **which compose of** which compose of the two components; one is known as magnitude, another is known as slope. The magnitude is nothing but, that del f square with respect to x and del f by del x

square plus del f by del y whole square and then, you take the square root that will give you the magnitude. And slope is nothing but, tan inverse the **del y by del f del** del f by del y slash del f by del x. And this magnitude will give you the edge strength and direction will be obtained by the slope **right**, edge direction is obtained by the slope.

(Refer Slide Time: 32:55)

Approximate Gradient

▶ Approximate gradient using finite differences:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h}$$

$$\frac{\partial f}{\partial y} = \lim_{h \rightarrow 0} \frac{f(x, y+h) - f(x, y)}{h}$$

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

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

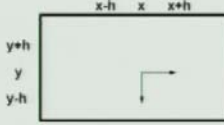
Now, how to obtain this **(())** because, **you know** this is **an** not an **a inte** integral part, this is your discrete domain. So, your value will that I have we have already shown that, the value will be f of x plus 1, y minus f x y with respect to del x and similarly, with respect to del y; it is f of x y plus 1 minus f of x y.

(Refer Slide Time: 33:20)

Approximate Gradient (cont'd)

► Cartesian vs pixel-coordinates:

- j corresponds to x direction
- i to y direction


$$f(x+1, y) - f(x, y) \rightarrow \frac{\partial f}{\partial x} = f(i, j+1) - f(i, j)$$
$$f(x, y+1) - f(x, y) \rightarrow \frac{\partial f}{\partial y} = f(i, j) - f(i+1, j)$$

But, that is in the cartesian domain. Now, you need to know what happens to the pixel domain, because pixel domain there is no x and y ; you have an i and j and x is in your case, x is this direction and y is because you are reading the image from this (()) y is here, y minus 1, y , y plus 1.

But in the case of image; it is first row, second row, third row, fourth row and so on. So, it is a and row is pointing to your y and column is pointing to your x . So, correspondingly you have to change this one; it was here x plus 1 y minus $f(x, y)$, it is i j plus 1 and i j right similarly, is the case with respect to y .

(Refer Slide Time: 34:13)

Approximate Gradient (cont'd)

sensitive to horizontal edges!

$$\frac{\partial f}{\partial y} = \frac{f(x_3, y_2) - f(x_3, y_1)}{y_2 - y_1}$$

or $\frac{f(x, y+Dy) - f(x, y)}{Dy} = \frac{f(x, y) - f(x, y+1)}{1}$ (grad in the y-direction)

($y_3=y_2+Dy, y_2=y, x_3=x, Dy=1$)

edge in the x-direction
(0,0)

sensitive to vertical edges!

edge in the y-direction

$$\frac{\partial f}{\partial x} = \frac{f(x_2, y_1) - f(x_1, y_1)}{x_2 - x_1}$$

or $\frac{f(x+Dx, y) - f(x, y)}{Dx} = \frac{f(x+1, y) - f(x, y)}{1}$ (grad in the x-direction)

($x_2=x+Dx, x_1=x, y_1=y_2=y, Dx=1$)

Now, this is the more detailed we can see later on.

(Refer Slide Time: 34:16)

Approximating Gradient (cont'd)

▶ We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:

-1	1
----	---

$\frac{\partial f}{\partial x}$ good approximation at $(x+1/2, y)$

1
-1

$\frac{\partial f}{\partial y}$ good approximation at $(x, y+1/2)$

Now, what happens in that case, if $\frac{\partial f}{\partial x}$ if you see that one $f(x+1, y) - f(x, y)$ we have obtained that we have obtained $f(x+1, y) - f(x, y)$ and $f(x, y) - f(x+1, y)$. So, basically $\begin{pmatrix} -1 & 1 \end{pmatrix}$ is or the mask is minus 1 and 1 and this side is 1 and minus 1 agreed, because that is the thing we have obtained.

So, in reality it is finding that, this point $f(x, y+1/2)$ of $f(x+1, y) - f(x, y)$. So, it is finding basically it is trying to find out the value of these two points to estimate the

because this square plus this square the value of this square plus value of this square will give you the thing, the difference between these points and difference between these points will give you the change.

The difference gives you what? Difference gives you the change of intensity with respect to x direction and change of intensity with respect to y directions that square **right** but, it is not taken into account the change of, because this direction also there will be a change. It is not bidirectional thing here has taken unidirectional with respect to x and with respect to y.

So, the better approximation becomes, if you consider this point **right** and this point will be considered only when, if you take this minus this and this minus this. So, that is the thing $f(x, y)$ and $f(x+1, y)$ and here you have taken $f(x+1, y)$ and $f(x, y+1)$. So, that will give you the effect or the change in the intensity here is it ok. So, that is known as a bidirectional approximation.

(Refer Slide Time: 36:34)

Robert's Method

► The simplest approximations to a first-order derivative that satisfy the conditions stated in that section are

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$G_x = (z_9 - z_3)$ and $G_y = (z_8 - z_6)$

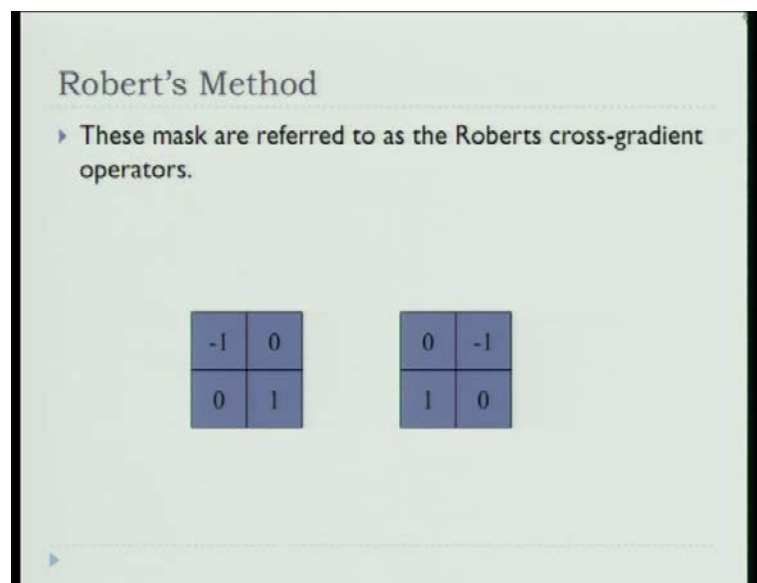
$\nabla f = \sqrt{(z_9 - z_3)^2 + (z_8 - z_6)^2}$

$\nabla f \approx |z_9 - z_3| + |z_8 - z_6|$

So, using this bidirectional approximation, you have obtained several edge detected algorithms. First one is the robert's method and it is considering, suppose you have the image of this 3 cross 3 image; it is estimating this area say, it is considering this z_5 , z_6 , z_8 and z_9 and with respect to x; gradient is this minus this and with respect to y; it is z_8 minus z_6 .

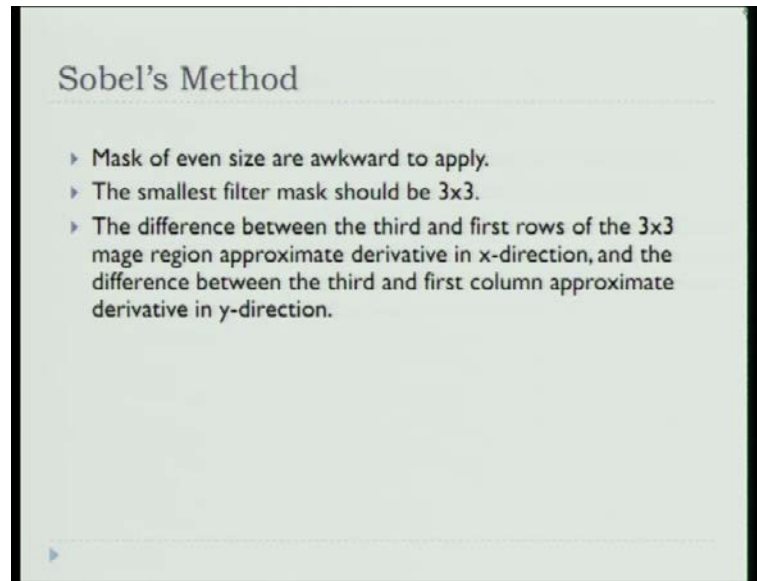
So, as **you know** the magnitude is nothing but, square root of $G_x^2 + G_y^2$; that means, square root of $(z_9 - z_5)^2 + (z_8 - z_6)^2$ **right** that will give you the edge strength or magnitude. And you can because **you know** it involves lot of square and square root, you can also estimate by $z_9 - z_5$ difference plus difference of $z_8 - z_6$, that also will give you the difference operator value.

(Refer Slide Time: 37:54)



Using this the Robert's Mask becomes minus 1, 0, 0, 1, 0. So, because you see that one, that z_9 becomes 1 this is minus 1 and this is minus 1, this is **yes**, this is minus 1, this is 1 and this is again with respect to y; this is 1, this is minus 1. So, 1 and minus 1.

(Refer Slide Time: 38:27)



Now, the problem here it is that 2 cross 2 and it is a very awkward way of estimating the whether a pixel is a edge pixel or not; reason is that, there is no center pixel here. So, for any what we are doing to estimate whether it is a pixel or not, edge pixel or not? You will be doing minus 1 of this plus of this **right** with respect to j x and similarly, with respect to this also you obtain and then, you determine and if it is greater than some threshold value; you take the decision whether it is edge pixel or not.

But, since it is a 2 cross 2 and it is a very awkward way of estimating the edge better way is that, is it possible the odd **sub** sub image size; that odd sub image size will be smallest one is 3 cross 3. It is not required that you have to consider 3 cross 3, you can go for 5 cross 5 and so on. And as you are increasing the 5 cross 5 or so on, that you convolution parameters that **you know that** you have to obtain certain operation on this. You have to increase the computational cost further, that you will be losing boundary results.

So, you have to be careful to select the size of the **(())**, the smallest one is 3 cross 3. Now, in order to obtain the whether it is a edge pixel or not, it is considering that with respect to x direction; it will be considering the difference of the third and **second** first column, first row and with respect to y , first derivative y direction, first derivative you will be considering the difference between the third column and first column **right**; that means, you have a 3 cross 3 sub image.

(Refer Slide Time: 40:29)



So, in order to get the first estimated value of first derivative; you will be obtaining the difference between these two. And in order to get the first derivative approximation with respect to y direction; you will be getting the difference between these two columns; first and third columns; that means what? No weightage is given to the center pixel, it is talking care whether **whether** central pixel is an edge pixel or not, that is determined by the other pixels, not himself. So, in general the structure is like that.

(Refer Slide Time: 41:22)

Another Approximation

- ▶ Consider the arrangement of pixels about the pixel (i, j) :
$$\begin{array}{ccc} a_0 & a_1 & a_2 \\ 3 \times 3 \text{ neighborhood: } & a_7 & [i, j] & a_3 \\ & a_6 & a_5 & a_4 \end{array}$$
- ▶ The partial derivatives $\frac{\partial f}{\partial x}$ $\frac{\partial f}{\partial y}$ can be computed by:
$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$
- ▶ The constant c implies the emphasis given to pixels closer to the center of the mask.

If you have 3 cross 3 neighbors; these are the pixel values and this is the center and you want to know whether this is a edge pixel or not, what we are telling that we have to obtain this partial derivative of this, instead of doing we are defining the mask now. The mask is nothing but, a 2 plus you give some weight to this a 3 c into a 3 plus a 4 minus a 0 c into a 7 c some constant plus a 6. So, initially we started with the idea that, difference between the third and the first columns or difference between the third row and the first row now we are changing **we are**. What we are doing telling, that not only difference will give little extra weight on the row or column, where your pixel is lying.

So, M_x is a 2 plus some value here, third column plus this minus this multiplied by this c times and so on. Now here you observe that I have given the same constant multiplication multiplier, why same constant? That it is expected that, if there is the constant weight you are giving here on this intensity same weight, because there is only one pixel distance. So, change will not be **(())** very large significant. So, same constant will be there.

Similarly, this direction, this direction also we are giving the same constant. Now, if **you know** as I change the value c, what it indicates? The c gives you that, how much weight is you want to give on the vertical and horizontal pixel intensities. Basically you see, that I am giving the weight extra weight on this am I right. So, this extra weights give you the indication, how much weight I want to give to the vertical and horizontal directions and how much less weight is you want to give towards the diagonal direction.

(Refer Slide Time: 43:50)

Prewitt Operator

▶ Setting $c = 1$, we get the Prewitt operator:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

M_x and M_y are approximations at (i, j)

So, as if I put c equals to 1, you get 1 edge detector which is known as prewitt operator. And if I put c equals to 2, then you get the sobel operator.

(Refer Slide Time: 44:08)

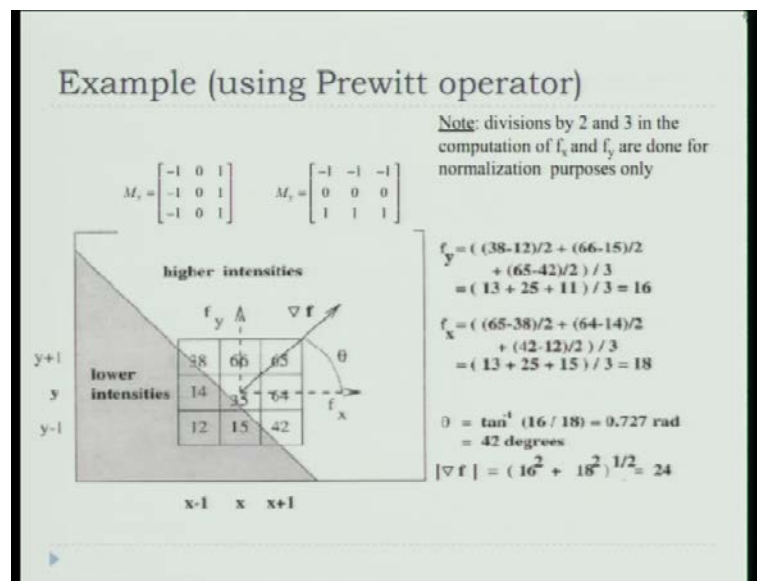
Edge Detection Steps Using Gradient

- (1) Smooth the input image ($\hat{f}(x, y) = f(x, y) * G(x, y)$)
- (2) $\hat{f}_x = \hat{f}(x, y) * M_x(x, y) \quad \text{---} \quad \frac{\partial f}{\partial x}$
- (3) $\hat{f}_y = \hat{f}(x, y) * M_y(x, y) \quad \text{---} \quad \frac{\partial f}{\partial y}$
- (4) $magn(x, y) = |\hat{f}_x| + |\hat{f}_y|$ (i.e., sqrt is costly!)
- (5) $dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$
- (6) If $magn(x, y) > T$, then possible edge point

So, come to that original algorithm what I told that, how to detect the edge. So, first is that you use some filter to smoothen it $G \times y$ is your filter **right**. So, you have used the filter formulate with formulation $f \times y$ with $G \times y$ to get the smoothen image. Now, this smoothen image you use your mask that, whatever sobel operator or prewitt operator you use it; to get $\text{del } f$ by $\text{del } x$ similarly, you get $\text{del } f$ by $\text{del } y$.

So, once you know the Δf by Δy then, I am just adding them with absolute value right I could have taken the square root of because I should have taken square root of this square plus this square, but it is a very costly thing, that is why we have taken just absolute value and then, add it and the direction is given by \tan^{-1} of this. Now, if this magnitude is greater than threshold, then you tell this is an edge point right. And from that edge point; you can use your direction you can determine in which direction your edge is moving.

(Refer Slide Time: 45:34)



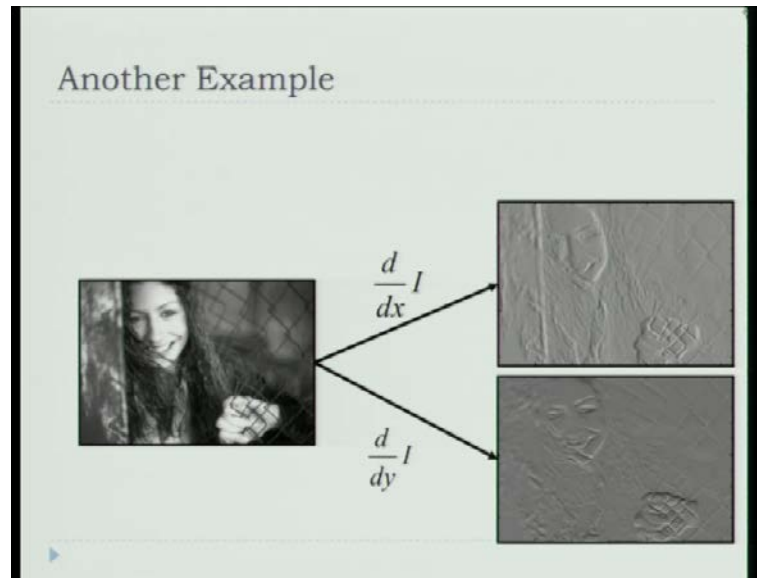
So, this is an example how to compute the prewitt through prewitt operator, this is a 3 cross 3 and the numbers are 38, 66, 65, 14, 35, 64, 12, 15, 42. Now, I have deviated see you observe that, I have taken what 38, minus 12; 38, minus 12 divided by 2 plus 66 minus 15 divided by 2 plus 65 minus 42 divided by 2 and then, total divided by 3. Or I have taken this minus this plus this minus this plus this minus this divided by 6 (Refer Slide Time: 46:16).

Now, why I am dividing by this, because I need to normalize also, because the intensity values; if I do that otherwise, if you do not have the normalize effect, then it will be (()) right. You do not know where it will be increasing, where it will be then threshold will not work. So, to use the proper threshold, you need to normalize them.

So, normalizing factor is by 6, because here you are taking the two values, two intensity value and finally, you are adding them by 6 three values; so again I am dividing by 3. So,

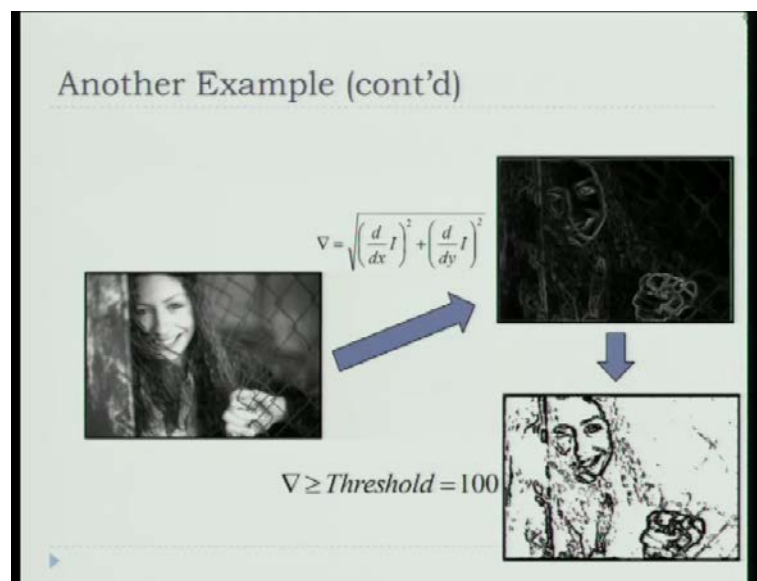
that is why it is divided by 6 and theta is as usual tan inverse 16 by 18 and you get the forty two degree this direction is 42 degree and the other difference or edge strength is 24.

(Refer Slide Time: 47:17)



So, this is an image with respect to derivative with respect to d x you get the the image like that and with respect to y, you get another one.

(Refer Slide Time: 47:30)




Now, if I add that and after that, that through thresholding you get your image looks like this. Now, you observe there are several noises are also there, even though you have used


the smoothing algorithm, whatever the noise will be there, then edges **edges** are disconnected; it should not be there, it should be connected edges and other thing **right**. You can find out in the case of here, they are disconnected edges or that is not correct one. So, we have to do something we will discuss later on, how to do it.


(Refer Slide Time: 48:02)


Isotropic property of gradient magnitude

- The magnitude of the gradient detects edges in all directions.

$\frac{d}{dx} I$


$\frac{d}{dy} I$


$\nabla = \sqrt{\left(\frac{d}{dx} I\right)^2 + \left(\frac{d}{dy} I\right)^2}$




We have discussed about the isotropic property (()) maintaining in all direction, the same similar structure. So, this is one example.

(Refer Slide Time: 48:11)

Diagonal edges with Prewitt and Sobel Masks

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

Prewitt

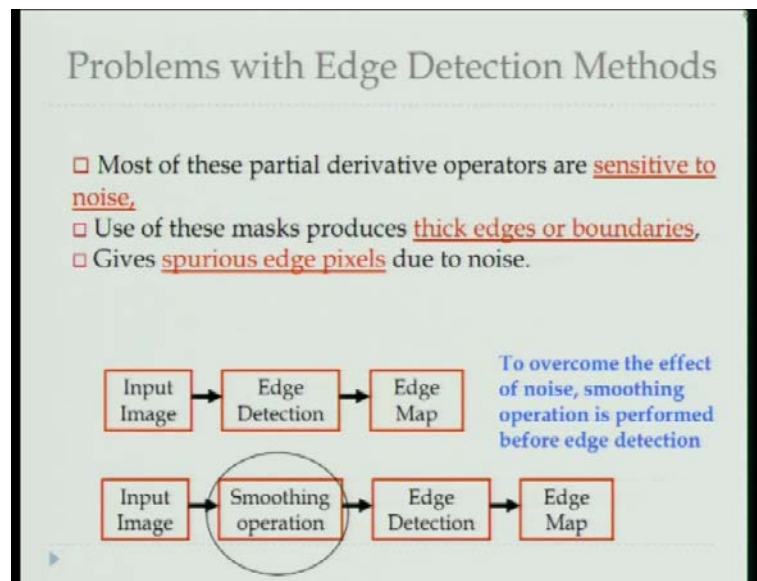
0	1	2	-2	-1	0
-1	0	1	-1	0	1
-2	-1	0	0	1	2

Sobel

Sobel masks have slightly superior noise-suppression characteristics which is an important issue when dealing with derivatives.

Now, what we have done, if you see that in the sobel or prewitt where operators it is your values are with respect to x direction or y direction **right** vertically or horizontally; it is not taken into account the diagonal effects. So, if I consider the diagonal direction, then it is the mask becomes 1, 1, 1, minus 1, minus 1, minus 1. And similarly, this side. Similarly, it is the case with the sobel, you have the several examples.

(Refer Slide Time: 48:43)



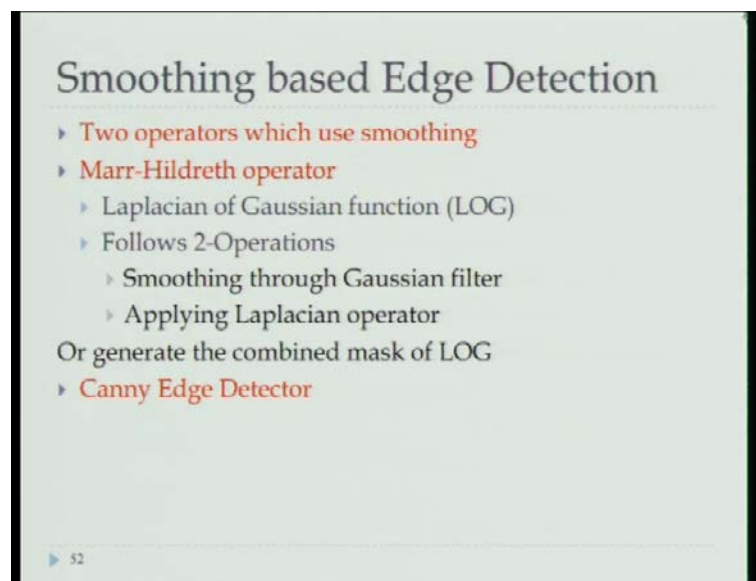
This will be this examples are available actually, because it is not showing here. So, those things will be added.

So, what you get with the edge detection major problem. So, that it is very sensitive to the noise; as noise is increasing **right** the edge also start edge is also you will find that, your system is not able to detect the edge properly. Because, the noise will also the edge also will be shown on noise. So, you need to design an algorithm, which will suppress the noise **right** then, it will this mask will be producing the thick edges, because if you see that, **you have** it is considering then minus 1, minus 1, minus 1, 1, 1, 1 that type of things, further you have smoothen the image.

So, thickness will be there, because **you have** you wanted to suppress the noise and as you wanted to suppress the noise; it is creating, what? That blurred image and as blurred image is there, edge thickness will be there, agreed.

And also due to noise, you will get some (()) edge pixels. So, original idea was that, you have input image then, you have to use the detection edge detection you will get the edge map. You realize that, no this will not work, because there exist noise and you need to suppress it. And in order to do that, you have added smoothing operations right. And by smoothing operations, you are you are suppressing the noise, but you are introducing another problem, which is in giving you the edge (()) right.

(Refer Slide Time: 50:35)



But, once using their through zero crossing, you will be getting the now the thin image through zero crossing, approximate thin image and from there, you have to determine your edge map.

Now, how to the smooth the image first how to smooth the image right and then, you get the edge detected map. So, there exist the two operators, (()) will be discussing this one, this will not be discussing today, because he still in different mode. So, the operator suppose I give you (()) 5 o' clock will will there be any change, no change. So, there is some problem with the system, some system there is some problem.

So, there exist the two operators, which use the smoothing one is that Marr-Hildreth operator, it is a very simple one, what it does? First I will use the Gaussian filter to remove the (()) and then, I will use the laplacian on the Gaussian filtered image and that is why its name is Laplacian of Gaussian, LOG right.

So, laplacian of gaussian function is nothing but, first you use the Gaussian filter to smooth it and then, use the simple laplacian operator on the Gaussian filtered image. The another one is the Canny Edge Detector.

(Refer Slide Time: 52:06)

Laplacian

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

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

Laplacian masks

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

▶ 53

You remember the laplacian operator in last class we have discussed this thing for to sharp the image and **the** it is nothing but, del square f x square by del x square. If it is the two dimensional one and del square f f x y del y del x y square. And the equation is, if I expand it, you will be getting that f of x plus 1 y plus f of x minus y y f of x y plus 1 plus f of x y minus 1 minus 4 times of f x y that, we derived in the last class, that it is coming the that 1, 0, 1, 0, minus 4, 0, 1, 0, 1 remember that was there.

(Refer Slide Time: 52:58)

The slide is titled "Laplacian of Gaussian (LOG)". It contains a bullet point: "Laplacian combined with smoothing as a precursor to find edges via zero-crossing." Below this, there are two equations in blue boxes. The first equation is $h(r) = -e^{-\frac{r^2}{2\sigma^2}}$ with the text "where $r^2 = x^2 + y^2$, and σ is the standard deviation" to its right. The second equation is $\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}$. At the bottom left of the slide, there is a small icon and the number "54".

Now, we also discussed what is Gaussian filter? Gaussian filter is nothing but, **e to the** minus e to the power minus r square; r square is nothing but, **x** x square plus y square divided by 2 sigma square. Now, this sigma is the filter parameter, how much filter you want to perform. Now, if I take the second difference on this, then I will be getting this, is it correct. Just take the derivative and then, second derivative it should be correct, just not this one I have used this one. Just first derivative of this and then, second time **you know** derivative of the function of f x y and then, y c you have to perform.

Then obviously, this will come sigma to the power 4 term **sigma to the power 4 term**, because in the book, it was 2 sigma square. So, I was in so most probably it will come sigma to the power 4 minus r r square by minus r square by sigma square sigma to the power 4 e to the power minus r by 2 sigma square.

So, now what happen that, **this is the** this will give you a mask; that mask if you use, then it will first filter the Gaussian with using the Gaussian filter and then, laplacian operator is used to sharpen the image.

this through filtering and then, zero crossing. Now, how to obtain the zero crossing we will discuss later on.

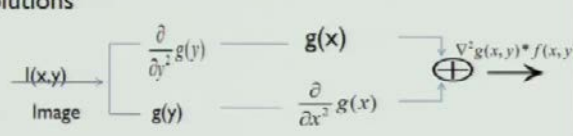
(Refer Slide Time: 55:15)

Decomposition of LoG

▶ It can be shown that LoG can be written as follows:

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

▶ 2D LoG convolution can be implemented using 4, 1D convolutions



So, see what happens that, this laplacian gaussian del square g x y can be defined as del of d y square g y into g x plus g y into **this** this is simple standard formula of derivative. Now, I am passing through because you have to convolve with I x y, because original image is your I x y. So, this I x y I am convolving with this one with this one then, I am convolving the result with g x y. So, I get the result here.

And similarly, I x y I will be convolving with g y and then, resultant will be convolved with **g** del del x y g x. The resultant these two resultant images you will be adding, because this is the formula I am writing the g x plus g y. So, that is the thing I am adding to get to get my LOG image.

(Refer Slide Time: 56:17)

Decomposition of LoG (cont'd)

Steps

1. Convolve the image with a second derivative of Gaussian mask ($g_{yy}(y)$) along each column.
2. Convolve the resultant image from step (1) by a Gaussian mask ($g(x)$) along each row. Call the resultant image I^x .
3. Convolve the original image with a Gaussian mask ($g(y)$) along each column.
4. Convolve the resultant image from step (3) by a second derivative of Gaussian mask ($g_{xx}(x)$) along each row. Call the resultant image I^y .
5. Add I^x and I^y .

So, this is the steps involved: first what we did, convolve the image with the second derivative of gaussian mask along with the column. And then, you are convolving the resultant with the gaussian mask, $g \times$ the resultant image is I^x similarly, you are doing with respect to y and then I^x plus I^y we are obtaining to obtain the laplacian of gaussian image. This is an example (Refer Slide Time: 56:46).

(Refer Slide Time: 56:48)

Zero Crossing & LoG

- ▶ Approximate the zero crossing from LoG image
 - ▶ Threshold the LoG image by setting all its positive values to white and all negative values to black.
 - ▶ Zero crossings occur between positive and negative values of the thresholded LoG.

▶ 60

Now, the zero crossing approximation, basically that is the important thing, how to estimate now zero crossing **right**. So, this what you can do that, there will be some

negative value, some positive value that is the thing in the case of second derivative, you will be getting the second derivative; you will be getting some positive value. And then, you are suppose to draw the lines, then the point which is touching your x axis that will be your zero crossing (()).

So, what instead of doing that, where all negative values I am making it dark and all positive values I am making it white right. So, you will have the dark, dark, dark, dark and then white, white, white and then, dark, dark. dark and so on. Now this is the area region; this is the region, where it is a dark; there is a change of oppositions; that is the reason is you have the zero crossing area right. And the middle of that, we consider one of the point as the pixel point, which is known as edge pixel point right. That zero crossing occurs between the positive and negative values of the thresholded LOG.

So, all negative you put dark and all positive you put white. Then, it will you will get the dark, dark, dark, white, white, white, dark, dark, dark and so on. So, in between the all white some position we consider that is the zero crossing element.

(Refer Slide Time: 58:13)

Gradient vs LoG

- ▶ Gradient works well when the image contains sharp intensity transitions and low noise.
- ▶ Zero-crossings of LOG offer better localization, especially when the edges are not very sharp.

step edge

2	2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8	8
2	2	2	2	8	8	8	8	8
2	2	2	2	8	8	8	8	8
2	2	2	2	8	8	8	8	8
2	2	2	2	8	8	8	8	8

0	0	0	0	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0

ramp edge

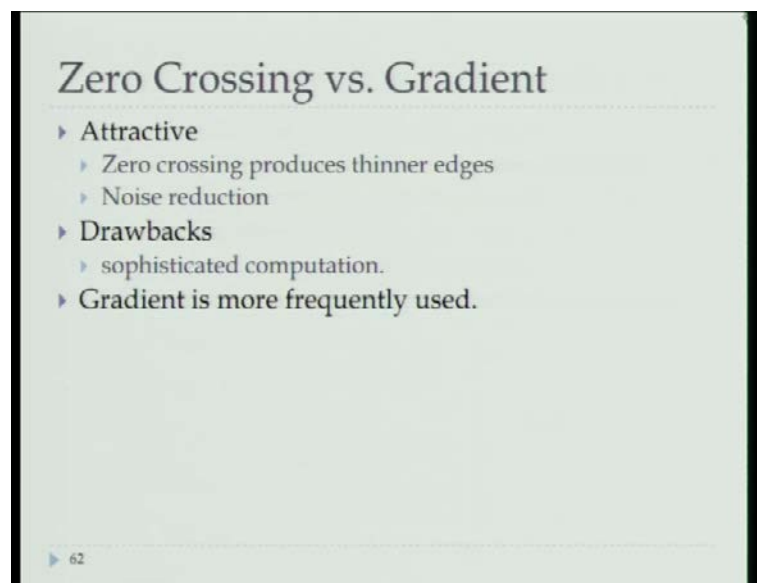
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8

0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0

See, now you know the two method; one is the gradient method, another one is your zero crossing way. Now, in the case of gradient method you see that, it was very well, when there is a sharp change sharp change of intensity level; the gradient method was very well, but when there is a ramp; slowly it is changing the intensity you will find that, zero crossing helps you very well to detect the to detect the edge points.

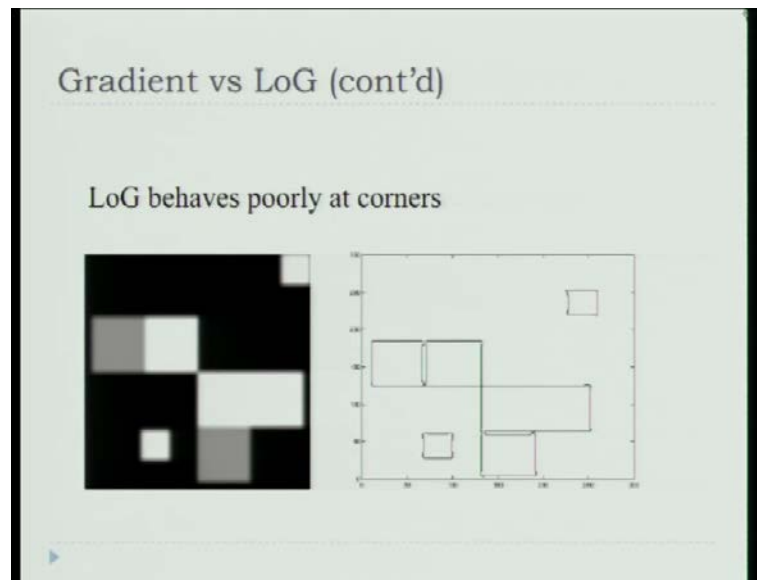
See here, it is a step edge the sharp change from 2 to 8 and you see that, this gives you the area 6 6 6 6 and minus 6 minus 6 is straightway you are getting. But, in this case; it is **it is not** it is a ramp image and in that case your value will be 3 0 minus 3 and you will not be able to detect straightway through gradient method, which one is your pixel edge pixel. But, through zero crossing, because this is the zero crossing area, you will be getting the your pixel, edge pixel point very easily.

(Refer Slide Time: 59:27)



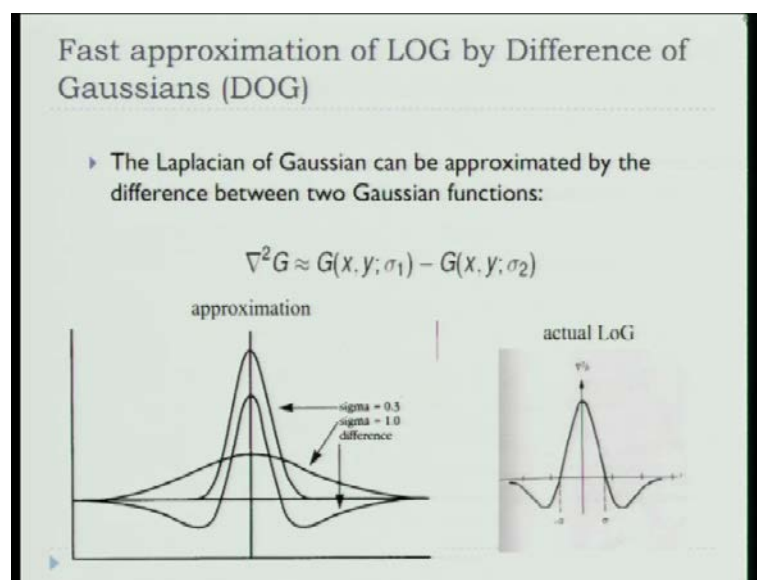
So, zero crossing gives you thinner edges, because it is the center point you are taking you will be getting the thinner edges. It reduces noise, but it takes time, because it has lot of computational thing involves. And in reality, we use the prewitt gradient method.

(Refer Slide Time: 59:49)



This is an example, where LOG behaves poor in the corner. See, this corner effects **right**. So, if sometimes you may not need **you may not need** the corner components then, LOG gives you the better, but it needs lot of because **you know** obviously, your mask will be size 5 cross 5 minimum. In the case of sobel and others you need 3 cross 3, here it is 5 cross 5; it needs more computational compare to that prewitt or sobel.

(Refer Slide Time: 1:00:29)




Now, L O G can be estimated through Difference Of Gaussian, which is known as D O G. So, the here L O G that del square G is nothing but, the difference with the two

gaussian filters. Now, these gaussian filters are dependent on the value of sigma as you told, the sigma gives you the filtering operations. So, this is the sigma is 1 and this is sigma is 0.3 and this gives you the sigma is the difference of these two operators and then, you see the behavior looks alike.

(Refer Slide Time: 1:01:07)

Difference of Gaussians (DoG) (cont'd)

$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2)$$


(b)-(a)

Ratio (σ_1/σ_2) for best approximation is about 1.6.
(Some people like $\sqrt{2}$.)

So, generally that ratio of sigma and sigma 2 is generally its we found that 1.6 gives you the good estimate for the approximating **approximating** that del square G.

(Refer Slide Time: 1:01:28)

Edge Linking

- ▶ Edge detection algorithm are followed by linking procedures to assemble edge pixels into meaningful edges.

▶ 66

And obviously that, whatever edge detector we have discussed till today, all of them are having the problem of linking between the two, there is a chance of gap between the same edge, but in between there may be a edge will be missing. So, there is a need of some edge linking algorithm.