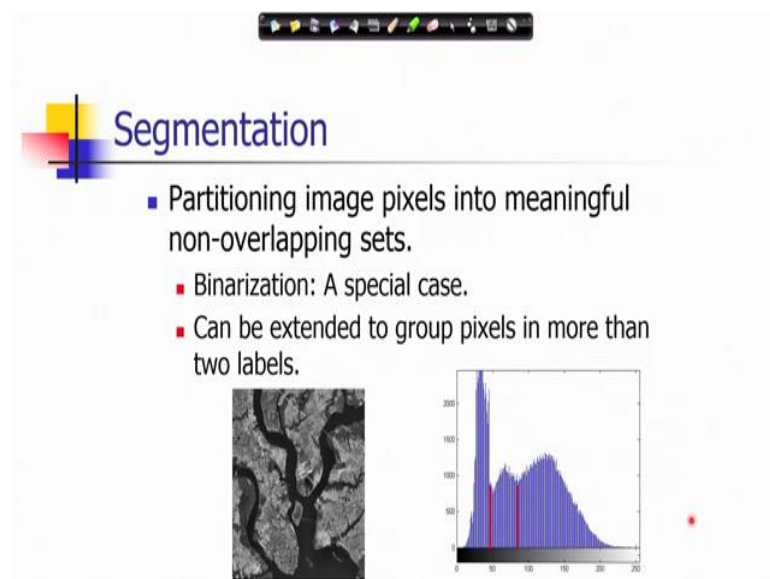


**Computer Vision**  
**Prof. Jayanta Mukhopadhyay**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 02**  
**Fundamentals of Image Processing – Part II**

We will move to the next part of this particular topic. We are still over viewing some of the Fundamentals of Image Processing.

(Refer Slide Time: 00:25)



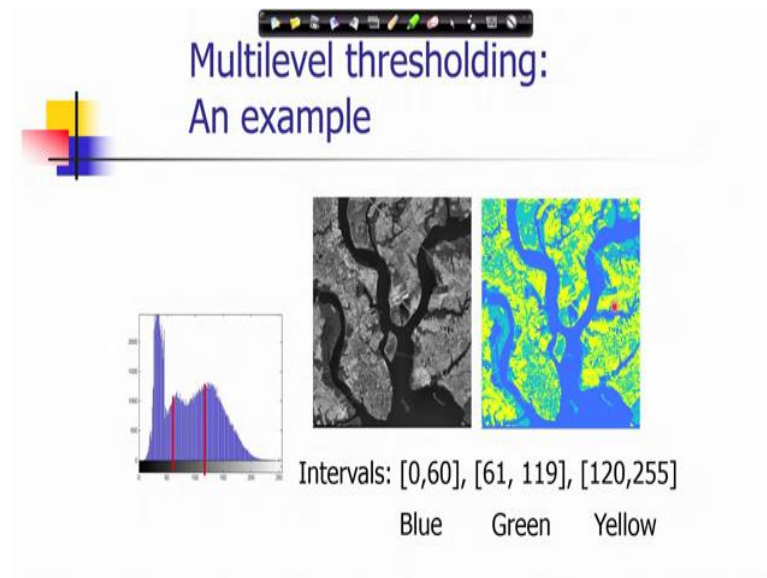
**Segmentation**

- Partitioning image pixels into meaningful non-overlapping sets.
  - Binarization: A special case.
  - Can be extended to group pixels in more than two labels.

The slide includes a grayscale image of a river network on the left and a histogram on the right. The histogram shows a bimodal distribution with two distinct peaks, one at a lower intensity value and one at a higher intensity value, illustrating the concept of segmentation based on intensity differences.

And we will start with the concept which is called Segmentation. Now, a segmentation is partitioning of image pixels into a meaningful non-overlapping sets. The binarization process what we discussed earlier, it is a special case of this segmentation. In that case, we have only two such sets that we discussed; one is of the background, another one is foreground. But it could be extended to groups of pixels, which are more than two levels.

(Refer Slide Time: 01:01)



For example we can consider a multilevel thresholding scheme. In the previous case, we have just only single thresholds. In a multilevel thresholds, we can have more than one threshold. Say consider this particular image and its histogram and let us consider we have selected these two thresholds. We can use a similar computation what we discussed earlier, particularly the Bayesian classification methods that can be extended further and we can determine this thresholds automatically also or manually by visualizing the modes, you can also select this thresholds.

In this particular case, let us consider there are three such intervals and for each interval, we assign a particular colour to those pixels. There are 3 colours you can look at this particular image and they are blue and green and yellow and each colour represents a segment of the image. As we can see from the intervals say there are 0 to 60 and 61 to 119 and 120 to 255.

*Intervals: [0,60], [61,119], [120,255]*

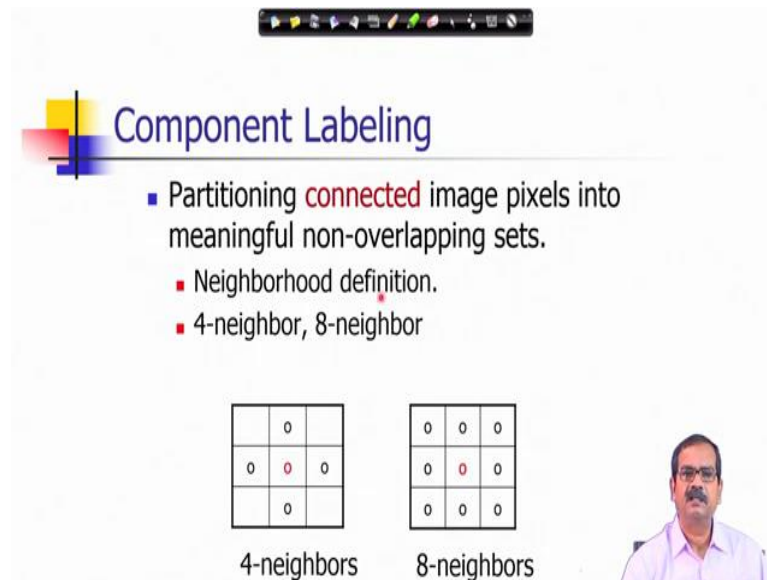
*for Blue, Green, Yellow*

These are the definitions of those non overlapping intervals and this is how the pixels are grouped into this particular image into 3 different segments.

You can note that 0 to 60 corresponds to the darker values which is the water channel here. 61 to 119 is a middle values. Usually, these are the points which corresponds to this

interval and 120 to 255, they are the brighter values and they corresponds to the yellow portions of the image. So, this is the blue zone. This is the green zone and this is the yellow zone.

(Refer Slide Time: 02:57)



**Component Labeling**


- Partitioning **connected** image pixels into meaningful non-overlapping sets.
- Neighborhood definition.
- 4-neighbor, 8-neighbor

	0	
0	0	0
	0	

4-neighbors

0	0	0
0	0	0
0	0	0

8-neighbors



So, one of the task of segmentation even after grouping the pixels into different intervals that is still remaining and that is called component labelling. So, in this case, we need to also partition the image pixels in the plane itself. So, it is a connected image pixels into meaningful non overlapping sets that is what is our objective. So, there we can use various neighbourhood definition to define this connectedness.

For example, we can consider the 4-neighbourhood definition or 8-neighbourhood definition in this respect. Say 4-neighbours are defined in this fashion. Say given a pixel, these are the positions of the neighbouring pixels and out of them, these are the pixels which are called 4-neighbours. Suppose, this is a location  $x$  and  $y$ . This is a coordinate location. So, this could be  $x$ , this should be  $x + 1$   $y$ . So, this is  $x$  right neighbour. Similarly,  $x - 1$   $y$ , this is its left neighbour.  $x$   $y + 1$ , its top neighbour.  $x$   $y - 1$ , its 4-neighbour, its bottom neighbour.

So, these are the 4-neighbours of this pixel and this neighbourhood is called 4-neighbourhood whereas, now if I consider also the diagonal pixels, diagonal positions are also their neighbours. So, then this definition is an 8-neighbourhood definition. So, you

can have different such neighbourhood definitions and accordingly, this connected, connectivity will also be defined between 2 pixels.

(Refer Slide Time: 04:35)

**Component Labeling**

- Form graph with edges between neighboring pixels having same labels.
- Compute connected components.
- Graph traversal algorithms

20	20	50	20
20	20	50	100
50	50	50	100
100	100	20	20

Do you require an explicit graph representation?  
Can you compute using only the image array?

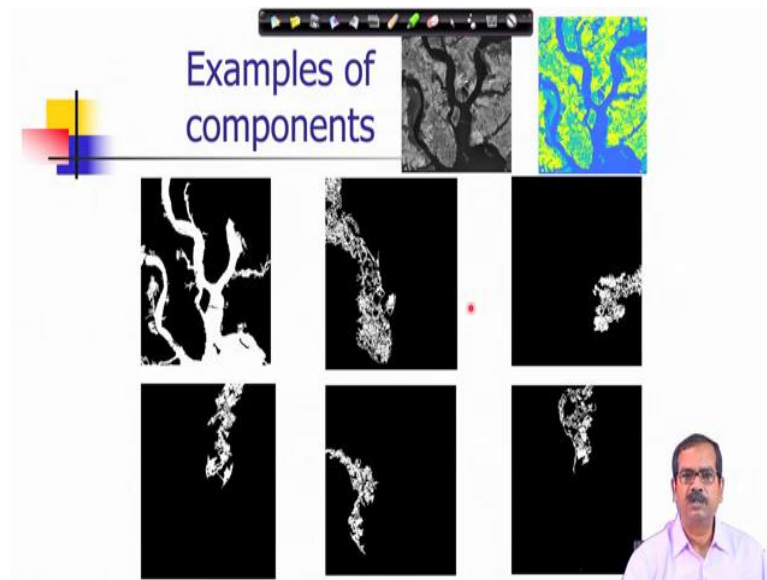
So, let us discuss a particular case when you want to get the components preserving the 8 connectivity of pixels. So, let us take this particular example. So, this is a small image and each pixel value is represented by these numbers. So, this is a 4 x 4 image for example. So, what we do, we can form a graph with edges between neighbouring pixels having same labels which say you consider every pixel is a node of this graph and you form the edge when their neighbour, the neighbours have the same pixel values.

So, for example, this is 20, 20, 20, 20. So, you can see that all these pixels, they form edges among between themselves and similarly, we can form edge between 50, 50 and this pixels in this form and also say for 100, 100 this is a edge. For another case this 20, 20 form an edge and 100, 100 another edge. So, it gives graphical representation of this particular image following the neighbourhood definitions.

So, now your task would be to find out the components of these graphs, you can use any graph traversal algorithms like breadth first search or depth first search and by which you can find out each component and declare them as a segment. So, I have displayed here by different colours, those segments. They are showing different components in this case. One of the interesting question, what could be asked here that whether you required explicit graph representation in this case.

Because we have already the image array and the neighbourhood definitions are merely précised, where they have a well organised structure. So, actually do not require any graph representation. You can compute everything on this array itself to get these components. So, you can compute using only one image array itself.

(Refer Slide Time: 06:59)



So, some of the examples of this processing let me show you. Say these are the two images which we earlier displayed. So, this is the original image and this is the segmented image into 3 segments, 3 classes of intervals. Now, we are forming the connected components from each segments. So, I will show you some of the prominent components in order of their number of pixels in those components. So, this is a largest component and you can see that in this component, we could recover we could get the river channel into one component itself.

Similarly, you can have various other components. So, this should be corresponding to this part of the zone. This is a green segment. So, connected green segments are shown here. This is mostly showing you connected regions in this part, yellow segment part. So, you get brighter pixels, yellow segment parts from here and in this way there are various other components could be also retrieved. I have shown only 6, but there are many other components which are possible in this image.

(Refer Slide Time: 08:15)

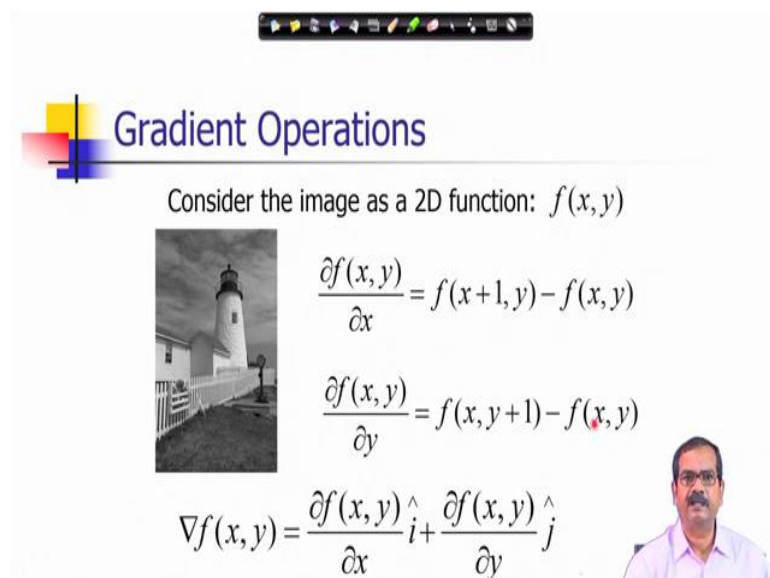


Why a part of river channel is missing?

One of the interesting question that I can ask you from this results itself; you need to be carefully observing particular, you should observe this results. See you can find out that though in the original image, there is a river channel, this is a river channel and in the largest segment also you can get the corresponding pixels to river channel, but part of the river channel is missing in this segment.


So, can you tell why this part is missing? So, this is a question you need to find out by yourselves and you check why this information is missing in this part.

(Refer Slide Time: 09:05)



### Gradient Operations

Consider the image as a 2D function:  $f(x, y)$


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

So, let us now discuss another operations for this with the images and we called this operations are computation of gradient values in the images. So, a gradient operation could be is defined in this way. So, image is a 2-dimensional function. So, in a 2-dimensional functions a gradient vector is defined by the corresponding partial derivatives along two principle directions. So, this is a partial derivative with x

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

and partial derivative of function with y

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

and they form a vector

$$\frac{\Delta f(x, y)}{\Delta x} = \frac{\partial f(x, y)}{\partial x} \hat{i} + \frac{\partial f(x, y)}{\partial y} \hat{j}$$

and that is a gradient vector.

So, computation of this gradient is quite simple and direct. We can use finite difference method by which you can compute gradient along the x direction which is simply the difference between the right neighbour and the pixel itself and similarly, we can compute the gradient along y direction. It is a difference between the top neighbour and the pixel itself. So, functional values at those positions is a difference. So, with these simple computations, you can compute these gradients and you can get the gradient vector.

(Refer Slide Time: 10:21)

**Computation with mask**

$\begin{bmatrix} -1 & 1 \end{bmatrix}$   
 $(x,y) \quad (x+1,y)$

Weights

$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$   $(x,y+1)$   
 $(x,y)$

$g(x,y) = (1) \cdot f(x,y+1) + (-1) \cdot f(x,y)$

1. Scan the image top to bottom and left to right.
2. At every point  $(x,y)$  place the mask and compute the weighted sum.
3. Write the value  $g(x,y)$  at  $(x,y)$  pixel position of the processed image.

These computations, so let me discuss in a more general framework. There is a motivation behind this that would be understood later on. Let us consider that these difference operations could be performed by a computation with mask. So, let us define a mask. In this case its a mask 1 x 2 elements. Mask is a kind of array. In this case, it happens to be 1-dimensional array. So, the dimension is that there are 2 rows and number of column is 1 in this case.

And these values in the mask, they represent the weights. So, what you need to do for computing the gradient all on the vertical direction, you place this mask at every pixel positions. So, this particular one of the points in the mask is corresponds to the central pixel positions and that is shown here by this particular coordinate  $(x, y)$ . So, you are placing this mask in the  $(x, y)$  positions and then, what you are doing you are performing the computation. So, you are performing the weighted sum of the pixel values of the corresponding with its neighbours, where this mask is also present.

So, once you place a mask here  $(x, y)$ . So, this should be my multiplied by - 1. Then, the functional value at  $(x, y + 1)$  should be multiplied by 1. So, effectively if you take the sum that would be the difference between  $(x, y+1)$  and  $(x,y)$ . So, the algorithm for these mask would be that you scan the image top to bottom and left to right, then at every point  $(x, y)$  place the mask and compute this weighted sum which means that as I mentioned that you multiply the weight with a corresponding functional values at those locations and then,



you take the addition. Take the sum of them and that would give you the weighted sum and then, you replace the central value by this weighted sum.

$$g(x, y) = 1 \cdot f(x, y + 1) + (-1)f(x, y)$$

So, in this way you can compute the finite differences along the vertical directions. Same computations can be carried out with this mask also which computes that difference along the horizontal directions.

(Refer Slide Time: 12:37)

**Robust gradient computation**

Averaging neighboring gradient values

-	-	1
-1	-1	1
-	1	1
-1	1	1
-	-	1
-1	-1	1
-	1	1
-1	1	1

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

So, to generalize this computation for making a robust gradient computation, what we can do instead of computing the difference only with the right neighbour or top neighbour, we can consider a neighbourhood region and find out this statistics of this finite differences in that region and take the mean of those know mean of that distribution.

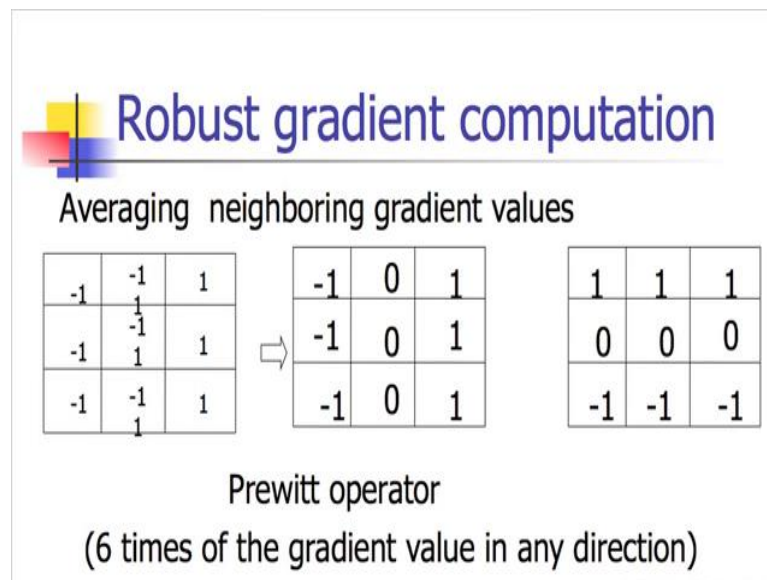
So, in this case particularly we are representing the mask value. So, we are taking a 3 x 3 mask and this is a central positions, these (x, y) positions. So, with respect to these position, so we can compute finite differences along using these two pixels also. So, , we are computing these finite differences 6 times around its neighbourhood and each one would not be the same; they are different values.

But it is expected that if there is a gradient directions statistically I mean they if you take the average, then the noise or the error would be reduced, if you take the average of all these values. So, which is equivalent this computations since these are all linear

combinations, this computations can be carried out by single mask computation itself, when the mask is designed in this fashion.

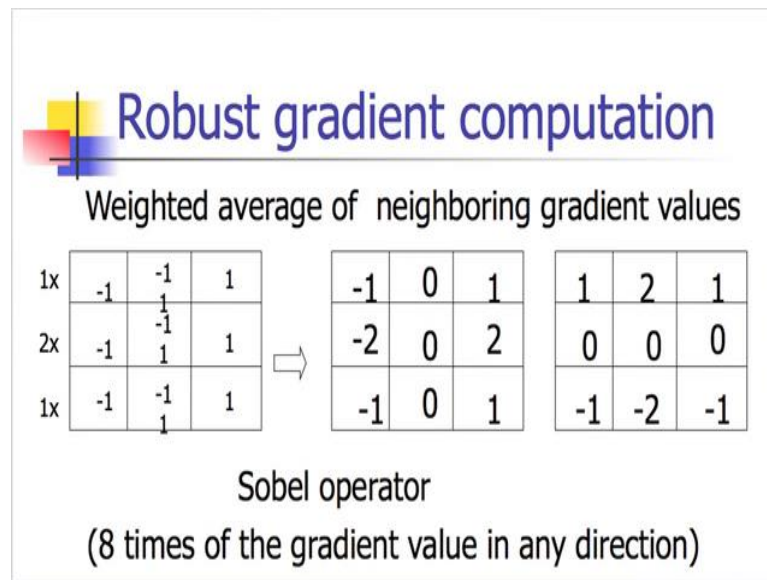
So, we can note that what we are doing at every cell of the mask, we are simply adding the weights. So,  $1 - 1$  becomes 0 here. This one is 1. So, in this way this is a distribution of weights in the mask and we can carry out the same computations with this mask and compute the corresponding. So, horizontal gradients in this particular case, in the gradient in the horizontal direction.

(Refer Slide Time: 14:37)



Similarly, gradient in the vertical direction also could be computed by this mask. Incidentally this masks two masks are called Prewitt operator in image processing and as you understand that 6 times you have computed this gradients in a particular direction.

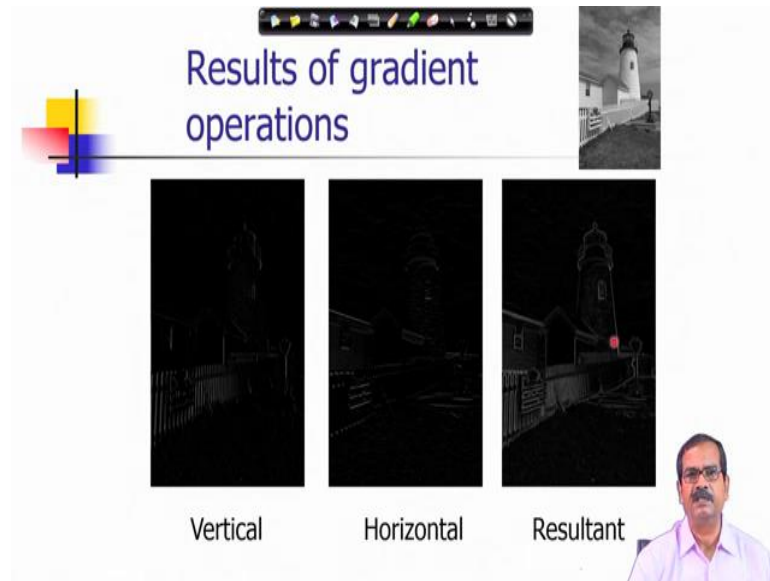
(Refer Slide Time: 14:59)



There is another example of robust gradient computation. In this case, what we are doing we are giving more weights to the central gradients than the off centric gradients. So, for example, in the horizontal directions when we are computing this gradient, we are giving weights of 2 here.

So, weights are doubled compared to the off centric rows and once again as I mentioned that every value of the cell can be computed by the sum of these corresponding values and if I multiply them with the weights. So, the distribution of the weights in the mask will look like this. This is for the computation on the horizontal directions and similarly, for the vertical directions, the distribution will be here in this case and so, this is a Sobel operator and this gradient has been computed 8 times in this case.

(Refer Slide Time: 15:57)



So, the results of these gradient operations, let me show you here. Consider the image on right top is your original image and if I compute the vertical gradients, I am showing those pixels by vertical gradients are very prominent. You can see those vertical bars of the fences; they become prominent in this particular image. For horizontal gradient again the horizontal directional fences which are parallel to the ground in those directions, those fences are becoming more prominent.

But if I consider the resultant of this two vectors by taking the magnitude of the vector itself, we will get all the edge pixels in every direction and the corresponding edge points or boundary points of different object services, they become prominent in this particular image. This is how you compute that gradients.

(Refer Slide Time: 16:53)

**More on computation with mask**

$w_1$	$w_2$	$w_3$
$w_4$	$w_c$	$w_5$
$w_6$	$w_7$	$w_8$

Convolution operation  
Filtering  
Mask  $\rightarrow$  Filter Response ( $h(x,y)$ )

$f(x,y) \rightarrow h(x,y) \rightarrow g(x,y)$

$$g(x,y) = w_1f(x-1,y+1) + w_2f(x,y+1) + w_3f(x+1,y+1) + w_4f(x-1,y) + w_c f(x,y) + w_5f(x+1,y) + w_6f(x-1,y-1) + w_7f(x,y-1) + w_8f(x+1,y-1)$$

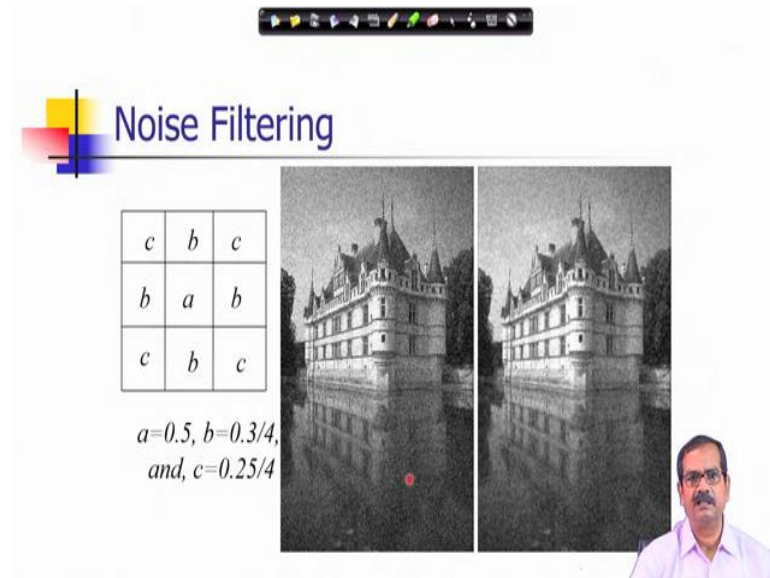
Let us generalize these computations with any arbitrary mask; that means, we consider any arbitrary weight distribution and this is an example of a 3 x 3 mask, but it could be of any dimension itself. So, if I consider weighted sum of the pixel values around the neighbourhood of the central pixel and we can represent this computations in the form

$$g(x,y) = w_1f(x-1,y+1) + w_2f(x,y+1) + w_3f(x+1,y+1) + w_4f(x-1,y) + w_c f(x,y) + w_5f(x+1,y) + w_6f(x-1,y-1) + w_7f(x,y-1) + w_8f(x+1,y-1) +$$

So, we are assuming that we are considering the functional values at the corresponding locations and then, multiplying its weights and we are taking the sum of all these values and that gives you the corresponding weighted sum and which would replace the functional values in the processed image. So, this computation is nothing but a convolution operation, where at every pixel you are placing this mask and you are computing this weights and then, replacing that functional value with this processed value.

So, it is the convolution operation. It is an operation meant for a computing outputs given an input of a linear shift invariant system in this case and this is an impulse response. So, these weights they define the discrete impulse response in this case, which is also called filtering because there is a frequency domain, transform domain interpretations which i will discuss in the next lecture. So, this mask is sometimes called also filters. So, this is the filter response  $h(x,y)$ .

(Refer Slide Time: 18:39)



**Noise Filtering**

$c$	$b$	$c$
$b$	$a$	$b$
$c$	$b$	$c$

$a=0.5$ ,  $b=0.3/4$ ,  
and,  $c=0.25/4$

One example of this kind of filtering is noise filtering, where the mask values are shown here. You can see that every value is positive and its weighted combination of the neighbouring values. Finally, sum of all these weights is equal to 1. So, this gives you a kind of low pass filtering action and if I apply this operation, these convolution operations on this image with the same mask computations what we discussed that is weighted sum of the weighted sum of pixel values around its neighbourhoods.

And from there, replace the original pixel value by the processed pixel value, we will get an image something like this, where you can see that this noisy structures look little reduced in this case..

(Refer Slide Time: 19:33)

**Gaussian Smoothing**

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{((x-x_c)^2 + (y-y_c)^2)}{2\sigma^2}}$$
$$g(x, y) = f(x, y) * G(x, y)$$

$\sigma=2$   
Mask size: 9x9

A special case of this kind of filtering operation often used in the image processing tasks is called Gaussian smoothing.

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

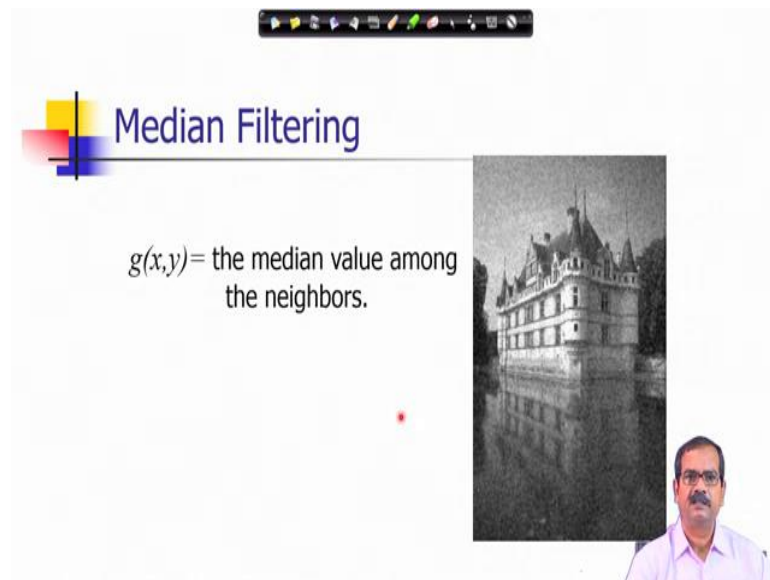
When the filter or the mask is computed, the weights of the mask is computed following a Gaussian distribution. So, you consider the origin of this distribution at the central pixels and there is a width of the Gaussian distribution or standard deviation of the Gaussian distribution. Accordingly, the mask size also has to be determined to capture the Gaussian most of the functional values, significant functional values around its mask.

So, \* is a symbol which is usually used for convolution operation. We denote this computation of with the mask this is what is a convolution operation.

$$g(x, y) = f(x, y) * G(x, y) \text{ with } \sigma = 2, \text{ mask size: } 9 \times 9$$

So, in this particular case, if the mask size is 9 x 9 and your  $\sigma$  is 2; then, using this know Gaussian mask by using this convolution operation, you can get a result like this.

(Refer Slide Time: 20:43)



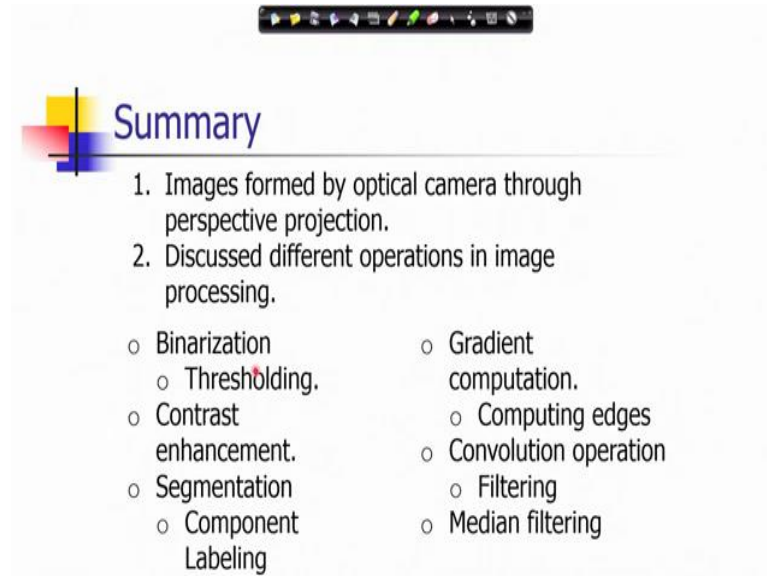
The slide features a title 'Median Filtering' with a decorative graphic of overlapping colored squares (yellow, red, blue) to its left. Below the title, the text reads:  $g(x,y)$  = the median value among the neighbors. To the right of this text is a grayscale image of a large, ornate building with a reflection in water, which is heavily corrupted with salt-and-pepper noise. A small red dot is visible on the slide, and a small inset photo of a man is in the bottom right corner.

Median filtering is another kind of computation which is not which does not use convolution rather it is a very simple to describe. What it does, it computes the median value among the neighbours among the neighbouring pixels and replaces original value by this median value.

It is a kind of non-linear filtering operations, but this is a very effective method to reduce a noise like spot noise kind or salt and paper noise kind of noise which are randomly the random dots which can be scattered over the images and those could be minimized, those could be reduced using mitigated using this kind of filtering.



(Refer Slide Time: 21:35)



The slide features a title 'Summary' in a blue font, preceded by a decorative graphic of overlapping colored squares (yellow, red, blue) and a vertical line. Below the title is a numbered list of two main points, with the second point having a sub-list of ten items arranged in two columns.

- 1. Images formed by optical camera through perspective projection.
- 2. Discussed different operations in image processing.
  - o Binarization
  - o Thresholding.
  - o Contrast enhancement.
  - o Segmentation
  - o Component Labeling
  - o Gradient computation.
  - o Computing edges
  - o Convolution operation
  - o Filtering
  - o Median filtering

So, here we are at the end of our lecture on this fundamentals on image processing and so, as a summary what we discussed in this particular lecture that we can summarize it. That images they are formed by optical camera and we see that the mapping of 2-dimensional, 3-dimensional points, 2-dimensional points that takes place through a kind of projection which is known as perspective projection.

So, we discussed rule of projection. That in this case, a point from a scene point a 3-dimensional point, you have to draw a straight line which passes through a particular point which is known as centre of projections and which hits the image plane that is point of intersection defines the corresponding image point that is what is perspective projection. Then, we have discussed different operations in image processing. Some of them are say binarization and in binarization particularly, we discussed the thresholding operation. Then, we discussed also contrast enhancement, we discussed segmentation and special task of segmentation say component labelling.

We also discussed gradient computation where we have discussed how edges could be computed and convolution operations. In the convolution operation, we also introduced the concept of filtering which is equivalently called filtering and it is a computation with a mask; a mask consist of weights in its cells and that weighted sum gives you corresponding distribution and then, we also discussed median filtering at the end. So, here let me stop.

Thank you very much for your listening.

Keywords: Binarization, segmentation, gradients, convolution, filtering.