Computer Vision and Image Processing - Fundamentals and Applications

Professor Dr. M. K. Bhuyan
Department of Electronics and Electrical Engineering,
Indian Institute of Technology, Guwahati

Lecture: 16

Image Filtering

Welcome to NPTEL MOOCs course on Computer Vision and Image Processing: Fundamentals and Applications. Last class I discussed the concept of image enhancement, the objective is to improve the visual quality of an image. Today, I am going to discuss the concept of image filtering to remove noises. Image filtering operations can be implemented in spatial domain or in frequency domain. In spatial domain I can manipulate the pixel values directly, so for this I can consider neighborhood operations. In case of a frequency domain I can modify the Fourier Transform of the image.

So first I have to determine the Fourier Transform of the image and after this I can do processing in the frequency domain. And after this I have to do Inverse Fourier Transformation to get the processed image. So, before discussing the image filtering concept, I will first discuss some image processing operations. So, already you have understood these operations, I am going to explain it again some of the operations like zooming and some neighborhood operations.

(Refer Slide Time: 01:46)

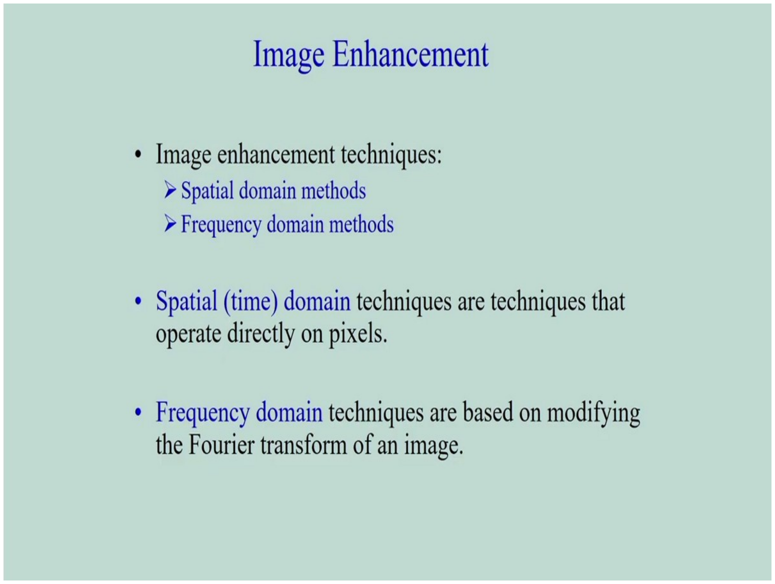


Image Enhancement

- Image enhancement techniques:
  - Spatial domain methods
  - Frequency domain methods
- Spatial (time) domain techniques are techniques that operate directly on pixels.
- Frequency domain techniques are based on modifying the Fourier transform of an image.

So, first one is the image enhancement techniques that can be implemented in spatial domain or in frequency domain. In spatial domain I can operate directly on pixels so that means I can change the pixel values directly. And in the frequency domain I can modify the Fourier transform of an image. So, this concept already I have explained.

(Refer Slide Time: 02:13)

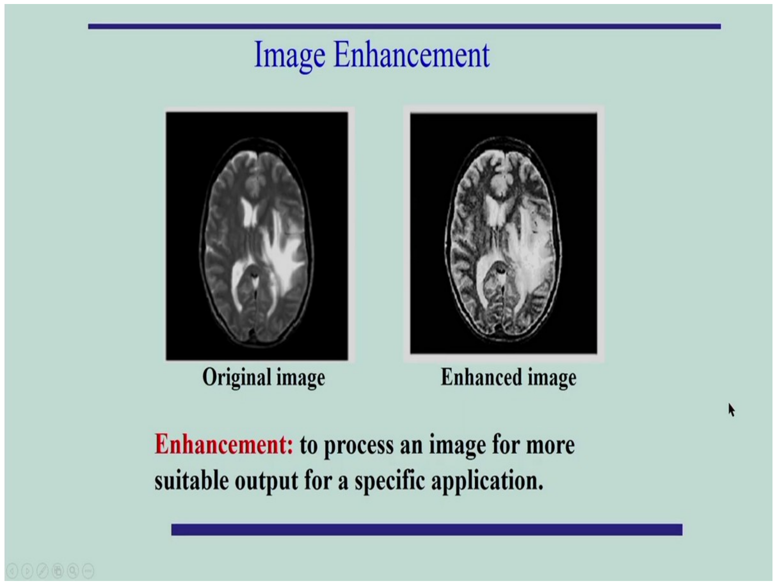


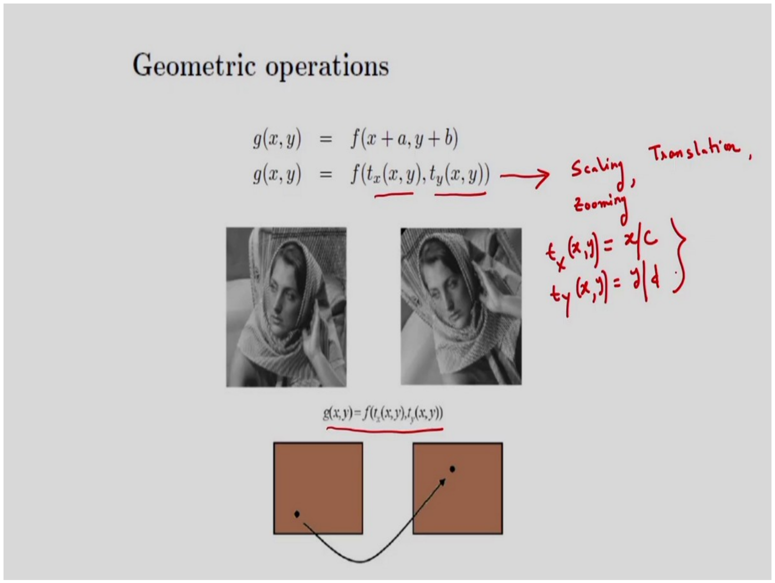
Image Enhancement

Original image
Enhanced image

Enhancement: to process an image for more suitable output for a specific application.

And in this case I have given one example of image enhancement. I have the original image and you can see the enhanced image.

(Refer Slide Time: 02:23)



Geometric operations

$$g(x,y) = f(x+a, y+b)$$
$$g(x,y) = f(\underline{t_x(x,y)}, \underline{t_y(x,y)}) \longrightarrow \text{Scaling, Translation, Zooming}$$

$$\left.\begin{array}{l} t_x(x,y) = x/c \\ t_y(x,y) = y/d \end{array}\right\}$$

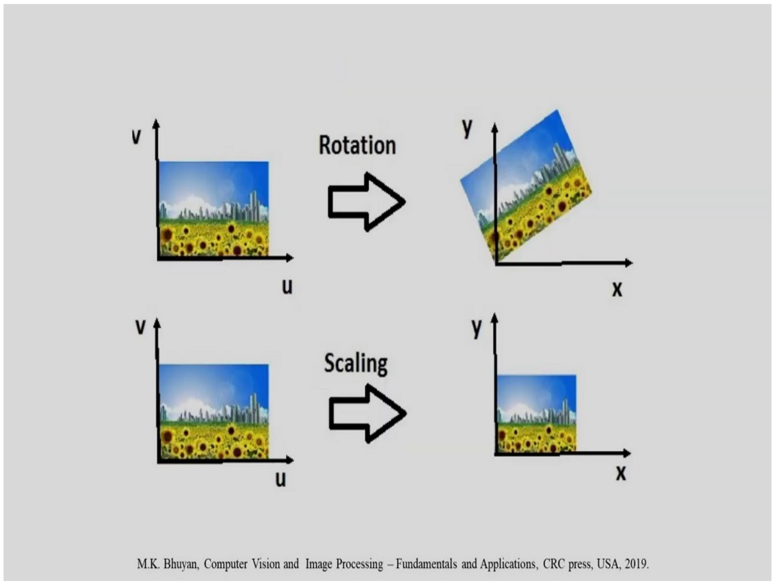
$$\underline{g(x,y) = f(t_x(x,y), t_y(x,y))}$$

And I have considered some geometric operations. So, if you see here g x y is equal to f x plus a comma y plus b. That means g x y that is the output image f tx y, tx y is the transformation for the x coordinate ty x y, that is the transformation for the y coordinate. That means I am changing the domain of an image.

So, if you see here that g x y is equal to f tx x y comma ty x y that is the I am changing the domain of an image. That is the spatial coordinates I am changing. So, based on this equation I can do scaling operation, I can do translation operation or I can do the operation like the zooming operation zooming operation also I can show.

The image zooming either magnifies or minifies the input image. So, if I consider suppose from transformation tx x y and in this case suppose x coordinate is divided by c and if I consider ty x y the y coordinate is divided by d. So, by this transformation I can do zooming. Zooming in and the zooming out.

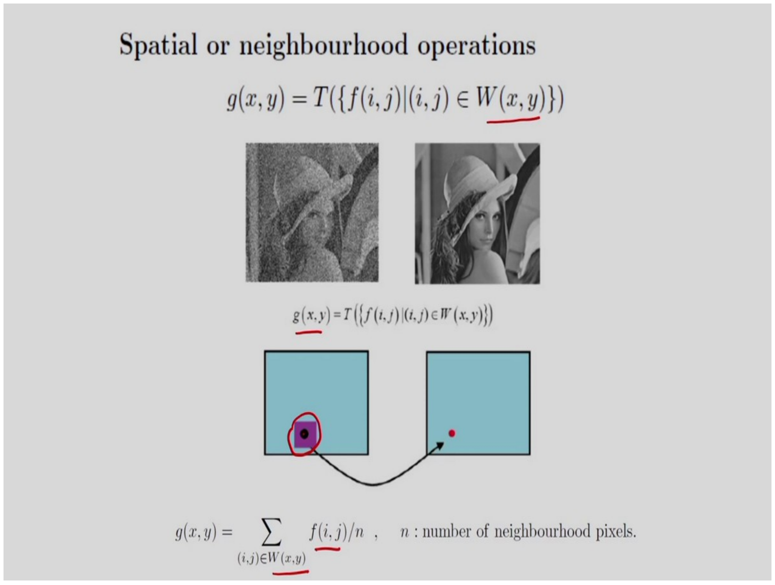
(Refer Slide Time: 03:59)



M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

And in this case I have shown some operations like rotation I can do and also I can do the scaling. That is I am changing the domain of an image.

(Refer Slide Time: 04:10)



Spatial or neighbourhood operations

$$g(x,y) = T(\{f(i,j) | (i,j) \in \underline{W(x,y)}\})$$

$$\underline{g(x,y)} = T(\{f(i,j) | (i,j) \in W(x,y)\})$$

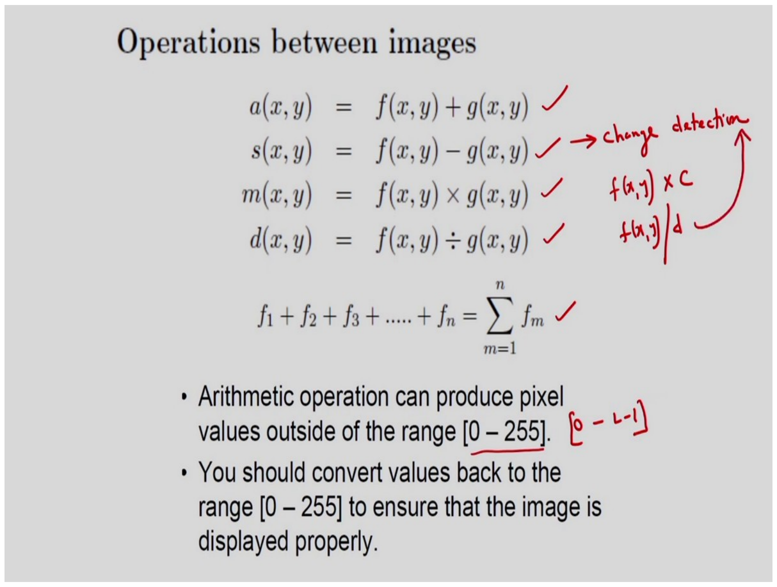
$$g(x,y) = \sum_{\underline{(i,j) \in W(x,y)}} \underline{f(i,j)}/n \ , \quad n : \text{number of neighbourhood pixels.}$$

And in this case I am showing some neighborhood operations that is the spatial or neighborhood operations. So, for this I am considering one window the window is given by W x y, that is the window. And I have the output image the output image is g x y.

Then in this case I am considering the neighborhood pixels for the processing so if you see this window, this is the window, and corresponding to this central pixel I am considering the neighborhood pixel. And based on these neighborhood pixels I can do the processing. That is called the neighborhood operations.

I can give one example of a neighborhood operation here the input image is f i j, this is the input image, divided by n that is the number of neighborhood pixels, and I am considering one window that window is W x y. That window I am considering. And I am getting the output image, the output is g x y. So, this is one example of neighborhood operations.

(Refer Slide Time: 05:14)



Operations between images

$$a(x,y) = f(x,y) + g(x,y) \checkmark$$
$$s(x,y) = f(x,y) - g(x,y) \checkmark \rightarrow \text{change detection}$$
$$m(x,y) = f(x,y) \times g(x,y) \checkmark \quad f(x,y) \times C$$
$$d(x,y) = f(x,y) \div g(x,y) \checkmark \quad f(x,y)/d$$

$$f_1 + f_2 + f_3 + \ldots + f_n = \sum_{m=1}^{n} f_m \checkmark$$

- Arithmetic operation can produce pixel values outside of the range $\underline{[0 - 255]}$. $[0 - L-1]$
- You should convert values back to the range [0 – 255] to ensure that the image is displayed properly.

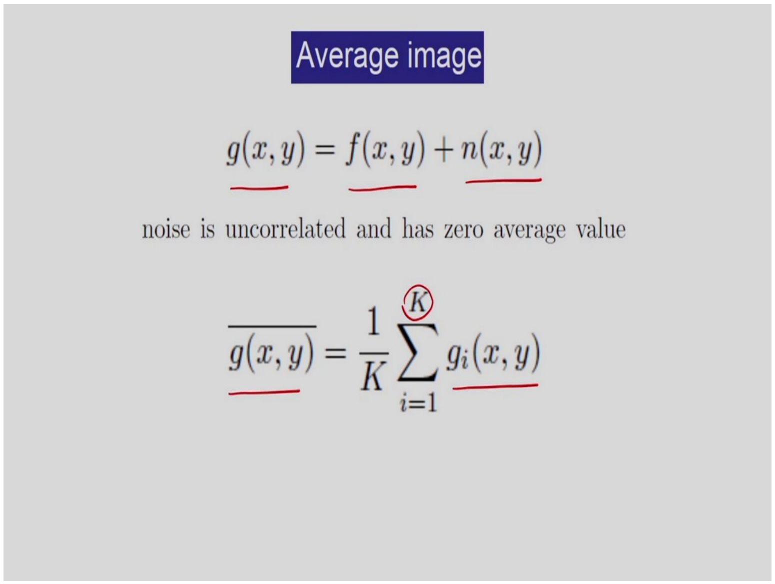
And I can do some operations between images. So, first you can see the addition of two images, the subtraction of two images that is the g x y is subtracted from f x y, so this is the subtraction operation. Multiplication of two images and you can see the division operation. So, in case of the subtraction, that is nothing but the change detection, already I have explained in my second class. So, it is nothing but the change detection.

If I want to detect the moving regions, then in this case I have to apply the subtraction operation. That is called the change detection. And in this case you can see that I am doing the addition of number of images. Now, because of these operations, the arithmetic operation can produce pixel values outside the range, the range is already we are considering from 0 to 255. That is mainly from 0 to L minus 1 number of levels I am considering.

So, arithmetic operation can produce pixel values outside the allowable range. Then in this case what I have to do I have to convert the values back to the range, the range is from 0 to 255. So, I have to do this. And in this case if I consider the multiplication of an image, suppose the multiplication of f x y by constant, suppose the constant is c, that is nothing but the scaling, I can do the scaling. Then in this case, by this operation, I can change the brightness of an image. The brightness I can adjust.

If the scaling factor is greater than 1, that means in this case the brightness of the image will be more and a factor less than 1 darkens the image. So, I can do the scaling of the image. And similarly, in the division also if I consider f x y divided by some constant, suppose d, by a factor of d, that is very similar to change detection. That is very similar to change detection.

(Refer Slide Time: 07:44)



Average image

$$\underline{g(x,y)} = \underline{f(x,y)} + \underline{n(x,y)}$$

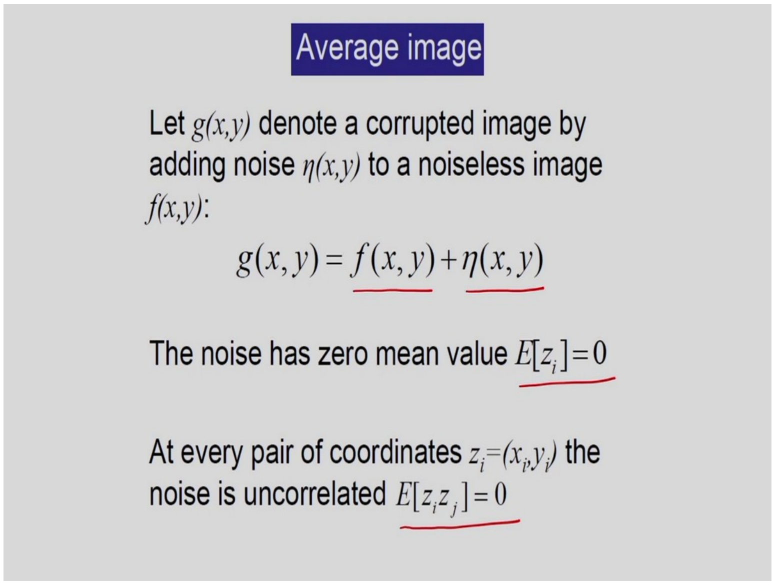
noise is uncorrelated and has zero average value

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

Now, in this case I am going considering how to remove noises based on averaging. So, I am considering one image that is f x y is the noiseless image and I am considering the noise is n x y. And suppose the g x y is equal to f x y plus n x y. And one consideration I am considering, that is noise is uncorrelated and has 0 average value.

Then in this case if I consider the K number of images, I am taking the average of K number of images. The images you can see, the images is g x y and I am determining the mean value I am determining, that is the average value I am determining. So, by this process I can remove noise.

(Refer Slide Time: 08:35)



Average image

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

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

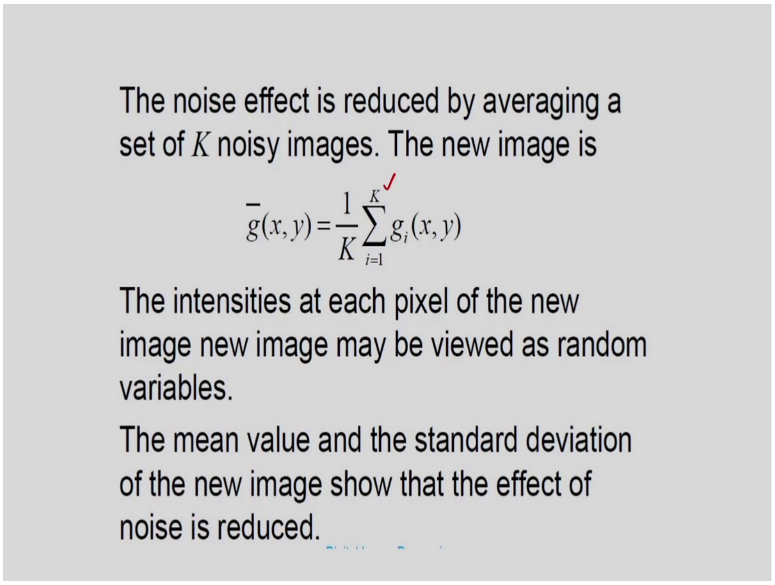
The noise has zero mean value $\underline{E[z_i]=0}$

At every pair of coordinates $z_i=(x_i,y_i)$ the noise is uncorrelated $\underline{E[z_i z_j]=0}$

So, I am explaining it again. So, you can see the g x y is the corrupted image, and in this case I am adding noise, that noise is eta x y. And in this case I am considering the noiseless image the noiseless image is f x y, that is the noiseless image. And we have considered the noise here has 0 mean value. So, that is why the expected value of zi is equal to 0.

And also, I am considering at every pair of coordinates the noise is uncorrelated. So, that means the expected value of zi zj is equal to 0.

(Refer Slide Time: 09:13)



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

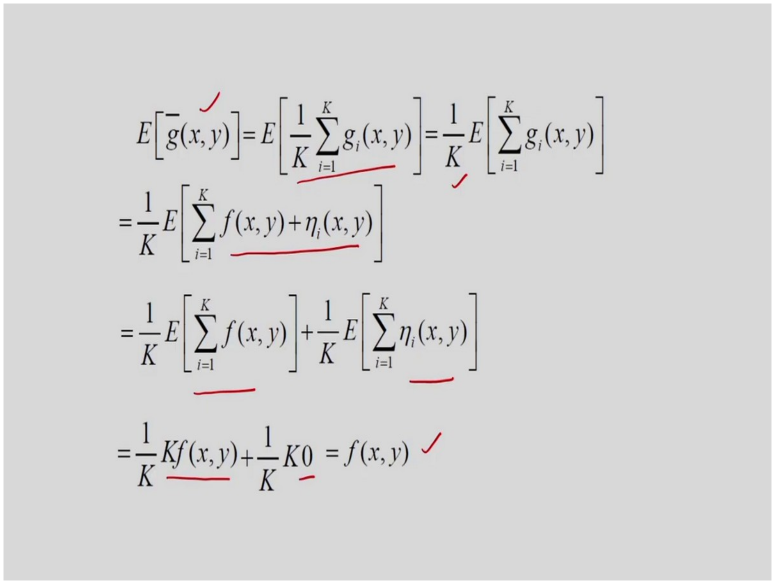
$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

The intensities at each pixel of the new image new image may be viewed as random variables.

The mean value and the standard deviation of the new image show that the effect of noise is reduced.

And this the noise effect is reduced by averaging a set of K noisy images. So, I am considering K number of images and I am determining the mean value. So, K number of images I am considering from i is equal to K and I am taking the the average of this. Now in this case the mean value and the standard deviation of the new image show that the effect of noise is reduced.

(Refer Slide Time: 09:39)



$$E\left[\bar{g}(x,y)\right] = E\left[\underline{\frac{1}{K}\sum_{i=1}^{K} g_i(x,y)}\right] = \frac{1}{K}E\left[\sum_{i=1}^{K} g_i(x,y)\right]$$

$$= \frac{1}{K}E\left[\sum_{i=1}^{K} \underline{f(x,y)+\eta_i(x,y)}\right]$$

$$= \frac{1}{K}E\left[\sum_{i=1}^{K} f(x,y)\right] + \frac{1}{K}E\left[\sum_{i=1}^{K} \eta_i(x,y)\right]$$

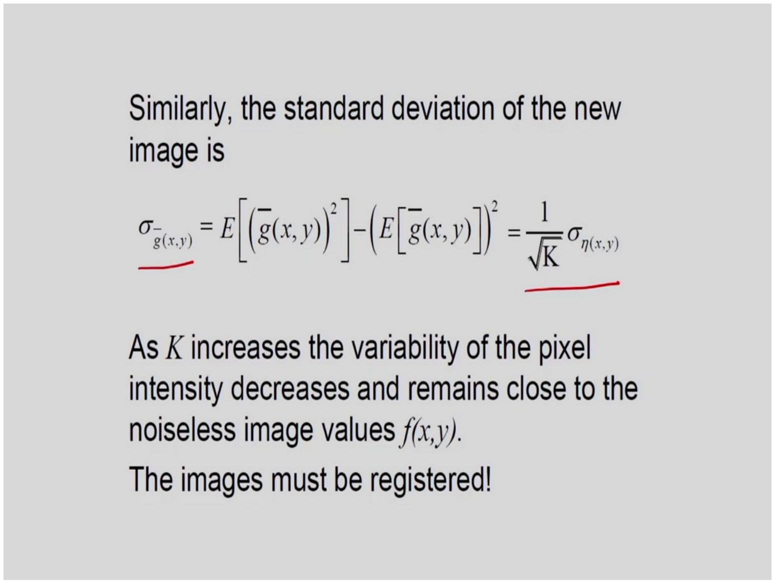
$$= \frac{1}{K}\underline{Kf(x,y)} + \frac{1}{K}K\underline{0} = f(x,y)$$

So, first I am showing the new images, there is a mean g, mean x y. And I am determining the mean value that the expected value I want to see. So if you see it is expected value of 1 by K this expression, in place of g mean I am putting this expression. And since 1 by K is constant, so I am taking out from the summation.

And after this, in place of gi x y I am putting f x y plus noise I am considering. And it is separated, the f x y is separated from the noise. And already I have considered that the noise has 0 mean, so that is why this will be 0 and this value will be K f x y. So, ultimately, I am getting f x y. So, if I take the average of number of images then I am getting f x y.

What is f x y? f x y is the noiseless image.

(Refer Slide Time: 10:43)



Similarly, the standard deviation of the new image is

$$\underline{\sigma_{\bar{g}(x,y)}} = E\left[\left(\bar{g}(x,y)\right)^2\right] - \left(E\left[\bar{g}(x,y)\right]\right)^2 = \underline{\frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}}$$

As K increases the variability of the pixel intensity decreases and remains close to the noiseless image values $f(x,y)$.

The images must be registered!

And also I can determine the standard deviation of the new image. So this is the standard deviation of the new image. And ultimately if I do this calculation, you can see it is 1 by root K sigma eta x y. So, if I increase K what will happen? If I increase K, K means the number of images if I increase the variability of the pixel intensity decreases and remains close to the noiseless image value f x y.

So, registered case. If I increase K then the variability of the pixel intensity decreases and remains close to the noiseless image value f x y. So, you can see if I consider number of images and if I take the average of this, then the noise can be reduced.

(Refer Slide Time: 11:36)

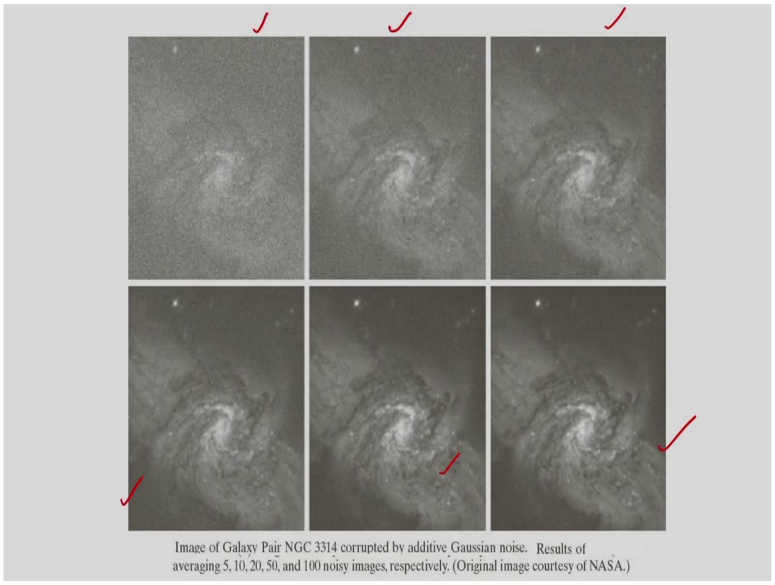


Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

And in this example I have shown the noisy image. The first one is the noisy image. The second one is the averaging of five images I am considering, the second visual is the average of five images. Next one is the averaging of 10 images, next one is the averaging of 20 images. And if you see this image, averaging of 50 noisy images. And this is a averaging of 100 noisy images. So, you can see that the noise is removed because of the averaging operation.

(Refer Slide Time: 12:10)

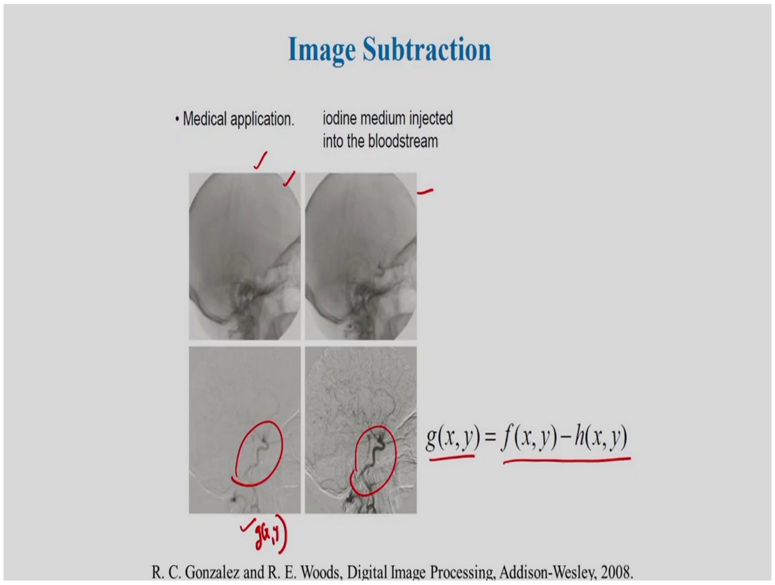


Image Subtraction

- Medical application. iodine medium injected into the bloodstream

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

R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

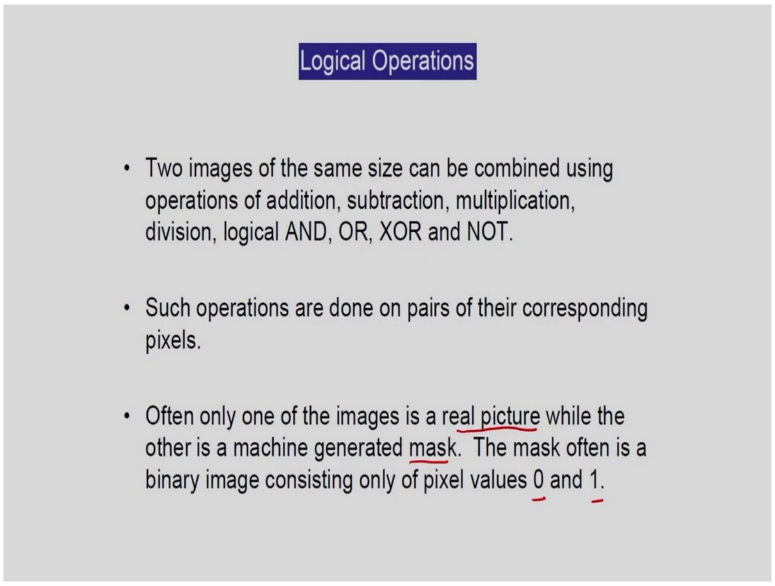
The next one is the image subtraction. So, already I have explained, because of the image subtraction I can determine the moving regions. That is called the change detection. So, for medical application that can be used.

Here I am giving one example. So suppose this is my one input image and suppose I want to determine the locations of the arteries and the veins, so for this I am injecting iodine into the blood stream. So, iodine medium injected into the blood stream I am doing. And in this case, if I do the subtraction between these two images, so this is the first image and this is my second image.

If I do the subtraction, that is f x y minus h x y, then I will be getting a g x y, the subtracted image I am getting. So, you can see the subtracted image. That is the subtracted image is the g x y I am getting. That means I can see the locations of the arteries and the veins. And in the next image you can see depth portion, depth portion is enhanced. Depth portion is enhanced by image enhancement techniques.

So, you can see the application of image subtraction.

(Refer Slide Time: 13:36)



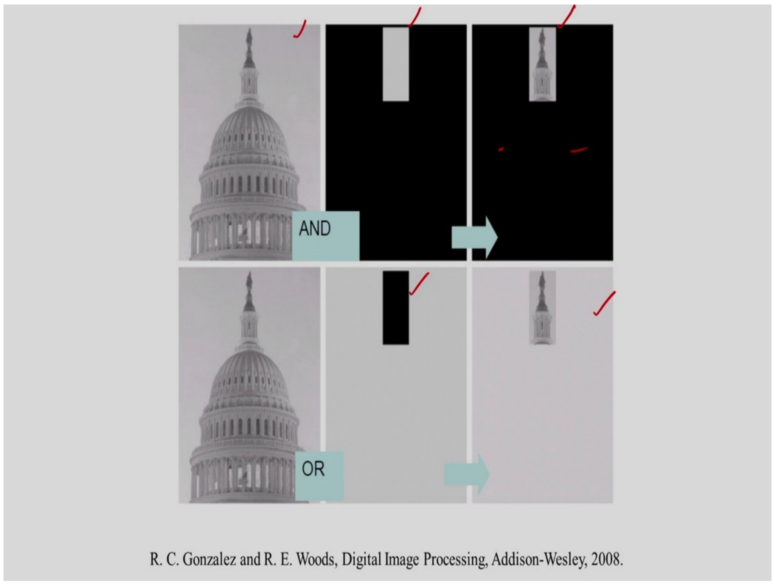
Logical Operations

- Two images of the same size can be combined using operations of addition, subtraction, multiplication, division, logical AND, OR, XOR and NOT.
- Such operations are done on pairs of their corresponding pixels.
- Often only one of the images is a real picture while the other is a machine generated mask. The mask often is a binary image consisting only of pixel values 0 and 1.

Also, I can do some logical operations. Like I can do AND logical operation, OR operation, XOR operation, NOT operations. Then these operation I can do for image processing. Than in this case such operations are done on pairs of their corresponding pixels. And in this case, I have to consider one real picture and another one is I have to consider is mask for this operation.

So, first I have to consider the real picture, real picture means the image I have to consider. And also I have to consider a machine generated mask I have to consider. So, in case of the mask it is nothing but the binary image consisting only of pixel values 0 and 1. So, this logical operation is performed between the mask and the image. So, I can show some examples of this logical operations.

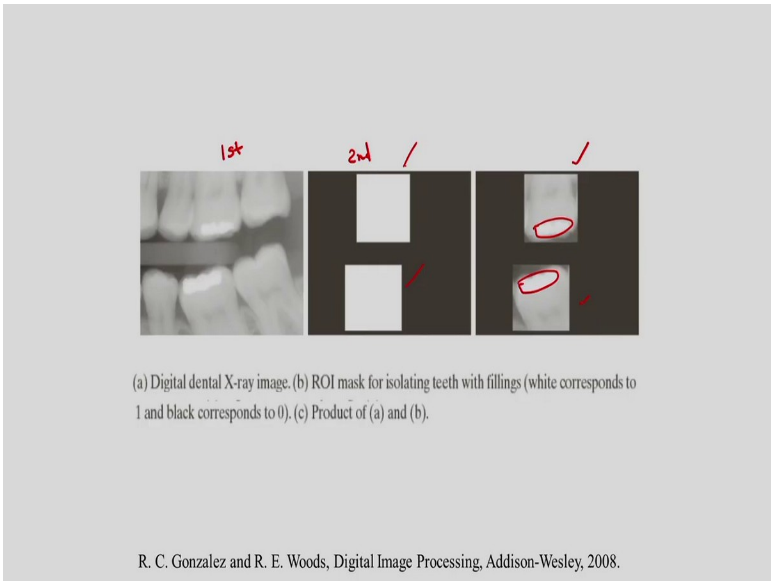
(Refer Slide Time: 14:34)


AND
OR
R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

So, first one is the AND operation. So, I am considering the image first one is the image and I am considering the mask term mask is like this. So if I do the AND operation, then in this case the output will be like this. If you see this portion will be black, this remaining portion will be black.

Similarly, if I consider the mask, suppose this mask, and if I do the OR operation then in this case I will be getting the output image something like this. So, I can apply these operations, the logical operations, AND operation, OR operation, all these operation I can apply.

(Refer Slide Time: 15:14)



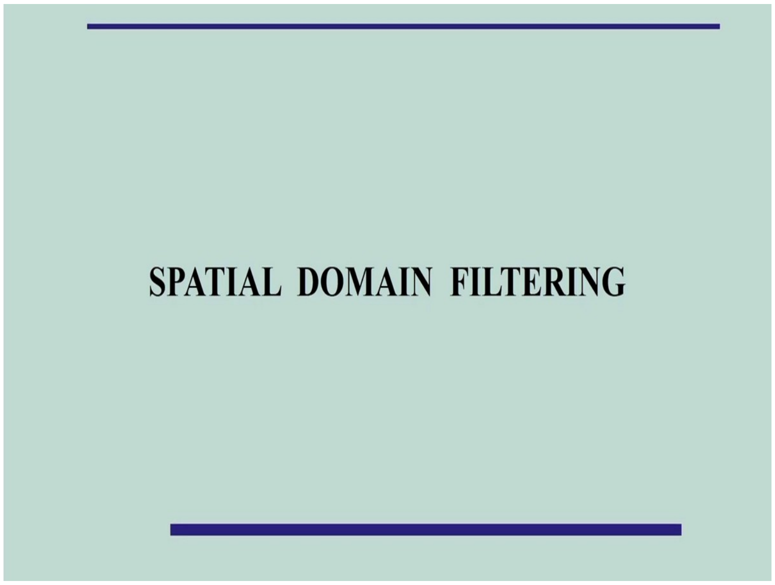
(a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

And here I am showing one another example. That is how to enhance region of interest. So, it is a dental X-ray image. And I am considering the mask, two masks I am considering. And in this case what I am doing, the multiplication of two images. This is the first image and this is the second image. I am doing the multiplication between these 2 images. Then in this case, you can see the result is this.

So, what I am getting? I am getting the region of interest mask for isolating teeth with fillings. So, that means, in this case I am getting the teeth with fillings, so mainly I am doing the multiplications between the image, the first image and the second image. In the second image I am considering mask. The mask means the region of interest mask for isolating teeth with fillings. So, these operations I can apply.

(Refer Slide Time: 16:18)



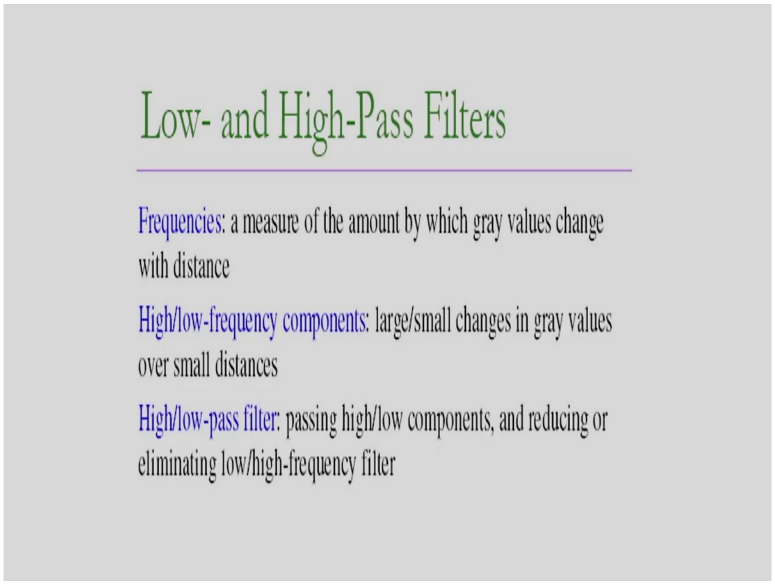
SPATIAL DOMAIN FILTERING

Now, let us consider the spatial domain filtering operation. So, what is the spatial domain filtering operation I want to explain. So, in spatial domain filtering operation I have to consider the neighborhood operations so for this I have to consider one mask.

Suppose, if I want to modify a pixel value, then in this case I have to consider the neighborhood pixels and mainly I have to do some convolution, the convolution operation I have to do. And based on these operations I can consider low pass filter, high pass filter, high boost filter, I can consider these type of filters. Also another important filter is the median filter I can consider.

So, first I will explain the concept of the spatial filtering, and in this case I will explain the mask. How to do the masking operation. And after this I will explain the concept of the low pass filter, high pass filter, a median filter I will explain.

(Refer Slide Time: 17:22)



Low- and High-Pass Filters

Frequencies: a measure of the amount by which gray values change with distance

High/low-frequency components: large/small changes in gray values over small distances

High/low-pass filter: passing high/low components, and reducing or eliminating low/high-frequency filter

So, in case of the digital image the frequency means the measure of the amount by which grey value change with distance. So, already you know about the frequency, this is the definition of the frequency and I may have the low frequency component or the high frequency component in an image. And for this I have the high pass filter or the low pass filters.

If I want to select the low frequency component, then in this case I have to apply the low pass filter. If I want to select the high frequency component, the high frequency component in an image is nothing but the edges and the boundaries. And if I consider a low frequency information that corresponds to the constant intensity or the homogeneous portion of the image.

(Refer Slide Time: 18:07)

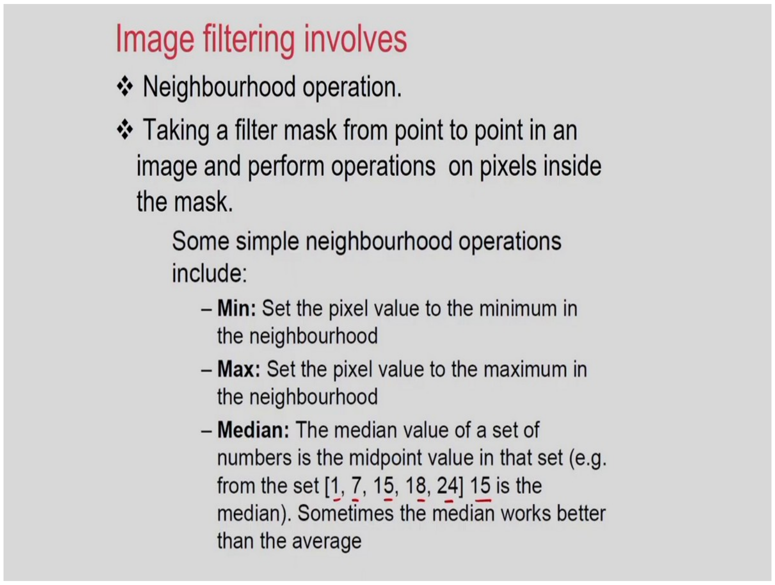


Image filtering involves

- ❖ Neighbourhood operation.
- ❖ Taking a filter mask from point to point in an image and perform operations on pixels inside the mask.

  Some simple neighbourhood operations include:

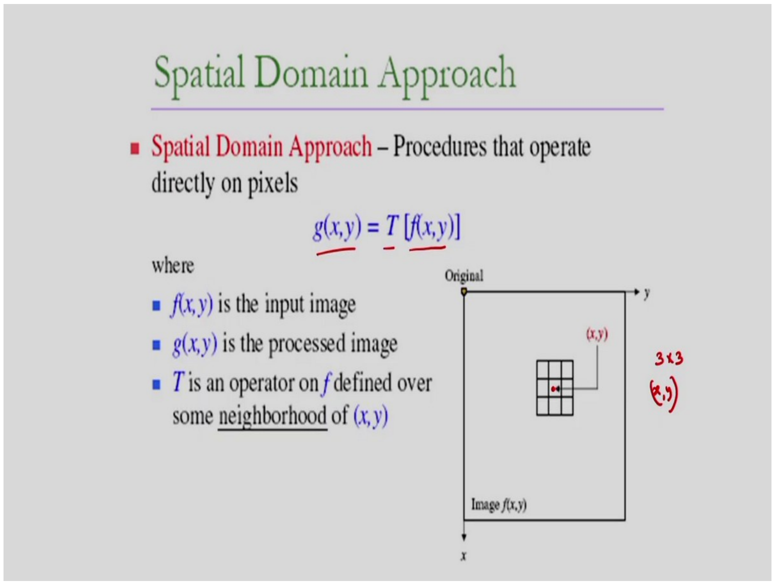
  - **Min:** Set the pixel value to the minimum in the neighbourhood
  - **Max:** Set the pixel value to the maximum in the neighbourhood
  - **Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

So, for the neighborhood operations I have to consider a filter mask and I have to take the filter mask from point to point in an image and perform operation on the pixel inside the mask. And based on this I may have this type of operations. One is the min operation, so what is the minimum operation, the min filter? Set the pixel value to the minimum in the neighborhood. That operation I can do.

Another operation is the max filter. Set pixel value to the maximum in the neighborhood, that operation also I can do. And I can do the median operation. So, in this case suppose if I consider neighborhood pixel value is 1, 7, 15, 18, 24, so for this what is the median value? Median value is 15, so that means the pixel under consideration, that value is replaced by 15.

In this case, I am considering the neighborhood operation. So, that is I am considering a mask, and in this mask I am applying this these operations, the min filters, max filter and the median filter.

(Refer Slide Time: 19:23)



Spatial Domain Approach

- Spatial Domain Approach – Procedures that operate directly on pixels

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

where

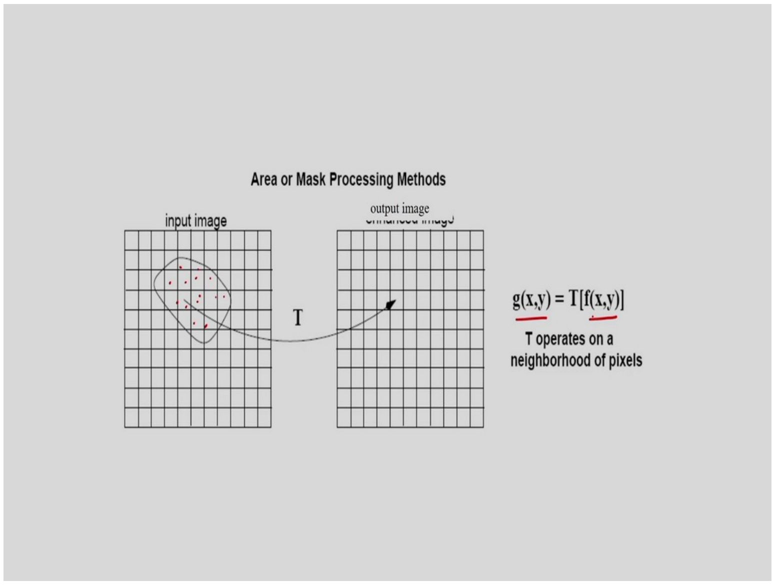
- $f(x,y)$ is the input image
- $g(x,y)$ is the processed image
- T is an operator on f defined over some neighborhood of (x,y)

Original
y
(x,y)
3x3
(x,y)
Image f(x,y)
x

In this case I am showing the spatial domain approach. And here you can see f x y is the input image and T is the transformation, I am doing some transformation. And I am getting the output image, the output image is g x y. T is an operator on the input image, the input image is f x y. And in this case I am considering the neighborhood of x y. So for this I am considering a mask, in this example I am considering the 3 by 3 mask, and on the pixel the pixel is x comma y. That is the central pixel a central pixel is x comma y.

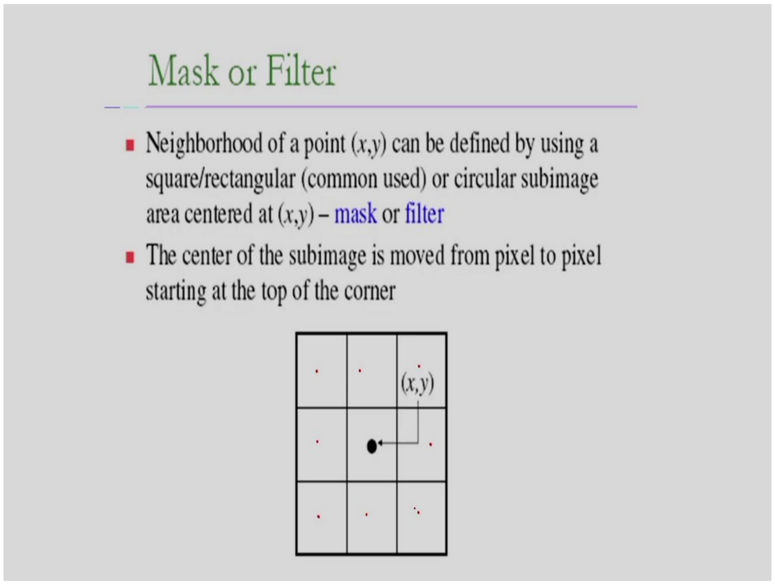
And let us see how to do this operation.

(Refer Slide Time: 20:06)


Area or Mask Processing Methods
input image
output image
T
g(x,y) = T[f(x,y)]
T operates on a neighborhood of pixels

So, again I am showing this one I have the input image and I have the output image and I am considering the neighborhood pixels. So, these are the pixels, the neighborhood pixels. And I am considering the mask and I am getting the output image. The output image is g x y f x y is the input image and T is the operation that is the neighborhood operation that I am doing.

(Refer Slide Time: 20:30)

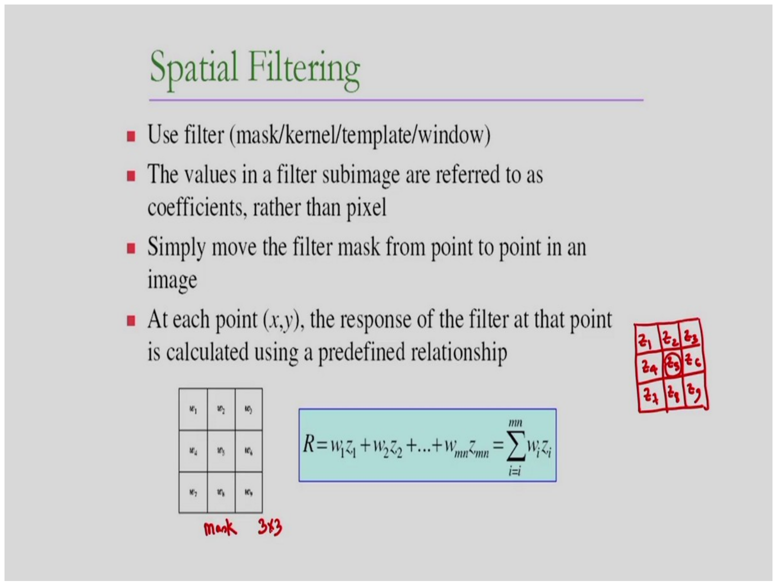


Mask or Filter

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

So, mask is similar to this. So it is a symmetric mask. And what is the central pixel of mask? The central pixel of the mask is x comma y. And that I can consider as a filter. So, I have the filter values, the coefficient values, or I can consider as widths of the masks. So, this is one example of the mask or the filter.

(Refer Slide Time: 20:58)



Spatial Filtering

- Use filter (mask/kernel/template/window)
- The values in a filter subimage are referred to as coefficients, rather than pixel
- Simply move the filter mask from point to point in an image
- At each point (x,y), the response of the filter at that point is calculated using a predefined relationship

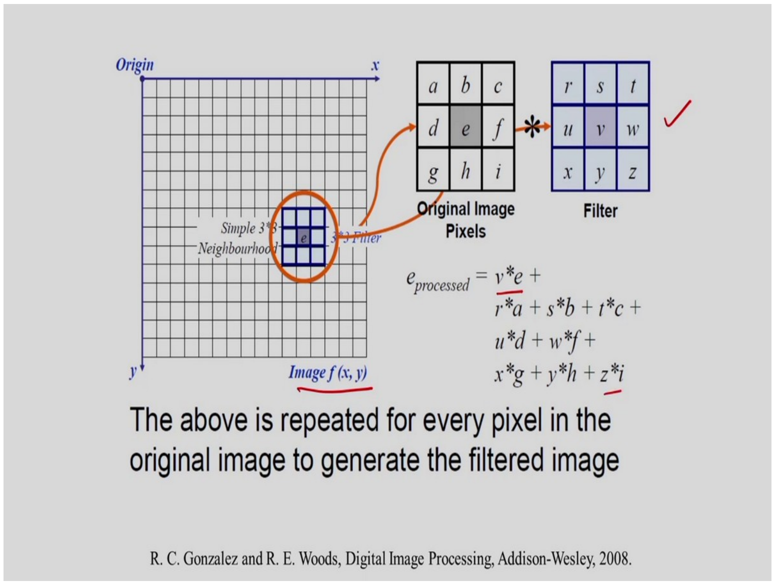
$$R = w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn} = \sum_{i=i}^{mn} w_i z_i$$

w_1 w_2 w_3
w_4 w_5 w_6
w_7 w_8 w_9
mask 3×3
z_1 z_2 z_3
z_4 z_5 z_6
z_7 z_8 z_9

And in spatial domain filtering what I have to do, I have to put the mask over the image, suppose I am considering one image Z1, the pixel value is Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8, Z9. And I have the mask, here this is the mask. So, what are the widths of the mask? The widths are w1, w2 like this, these are the width of the mask. And what I have to do? I have to put the mask over the image. So, corresponding to this pixel the central pixel I have to put the mask over the image, that means I am doing the overlapping of the mask with the image.

After this, corresponding to the pixel Z5 I am determining the response of the mask. The response of the mask is given by w1 into z1, plus w2 into z2, like this I have to determine the response of the mask for the image pixels. So, pixels are Z1, Z2, Z3 up to Z9. And in this case, I am considering the 3 by 3 mask, the 3 by 3 mask is considered. That is the symmetric mask.

(Refer Slide Time: 22:26)



$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image

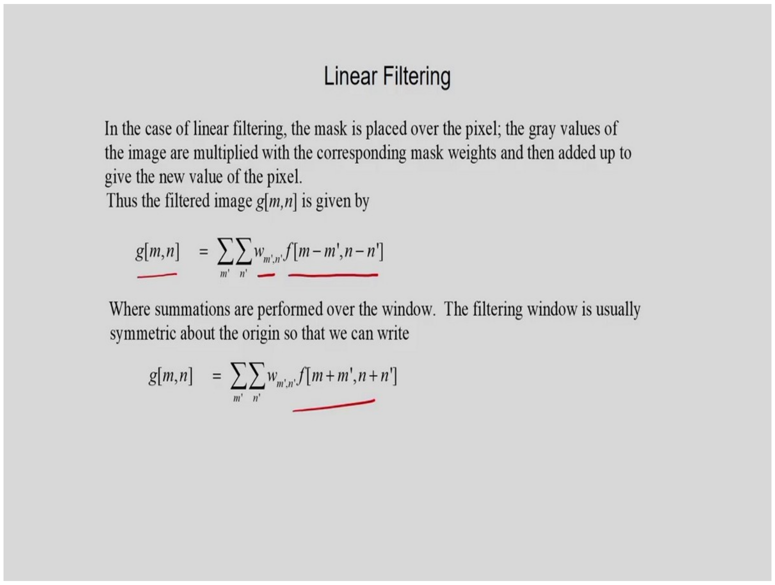
R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

This operation again I am showing here. I am considering a 3 by 3 filter, my input image is f x y, I am considering the mask, 3 by 3 mask I am considering. So, this is the filter and if you see the widths of the filter, the filter widths or the coefficients are r, s, t, u, v, w, x, y, z. And if I considered the image pixel value a, b, c, d these are the a, b, c d upto i. These are the original image pixels I am considering.

And already I have explained, so how to do the masking. I have to put the mask over the image and I have to determine the response of the mask, that is nothing but v is multiplied with e, plus r is multiplied. This is the multiplication, like this I have to do the multiplication and addition. This process is repeated for all the pixels of the image.

Then in this case I will be getting the filtered image.

(Refer Slide Time: 23:26)



Linear Filtering

In the case of linear filtering, the mask is placed over the pixel; the gray values of the image are multiplied with the corresponding mask weights and then added up to give the new value of the pixel.
Thus the filtered image $g[m,n]$ is given by

$$\underline{g[m,n]} = \sum_{m'}\sum_{n'} \underline{w_{m',n'}}\, \underline{f[m-m',n-n']}$$

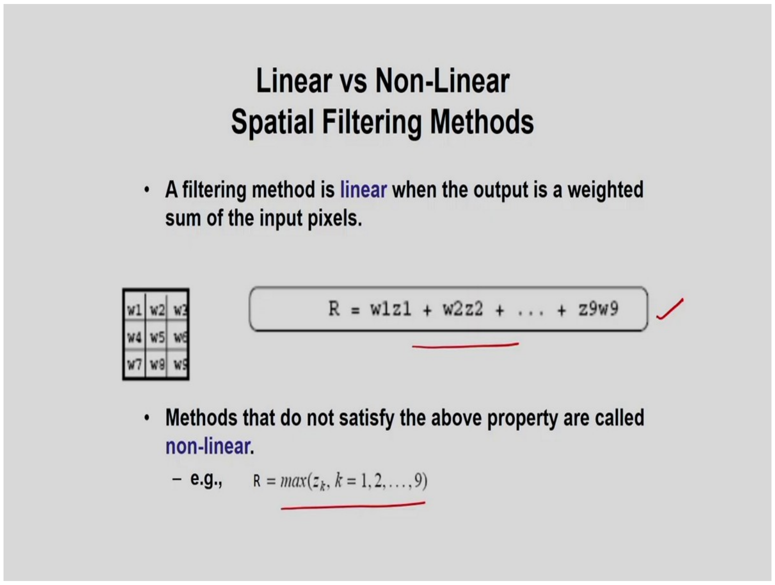
Where summations are performed over the window. The filtering window is usually symmetric about the origin so that we can write

$$g[m,n] = \sum_{m'}\sum_{n'} w_{m',n'}\, \underline{f[m+m',n+n']}$$

So, this concept is called the linear filtering. So, if you see this expression here g m n is the output image and, in this case I am considering the input image is f m n and I am considering the mask that mask is given by w m dash n dash. And in this case summations are performed over the window. And in this case, the filter is the symmetric filter, that is mask is symmetric mask.

Then in this case I can do these operation, that is nothing but you can see it is something like convolution. Because the filter coefficients are multiplied with the pixel values and it is added up. It is nothing but the multiplications and sum up, that is the convolution.

(Refer Slide Time: 24:18)



Linear vs Non-Linear Spatial Filtering Methods

- A filtering method is linear when the output is a weighted sum of the input pixels.

w1	w2	w3
w4	w5	w6
w7	w8	w9

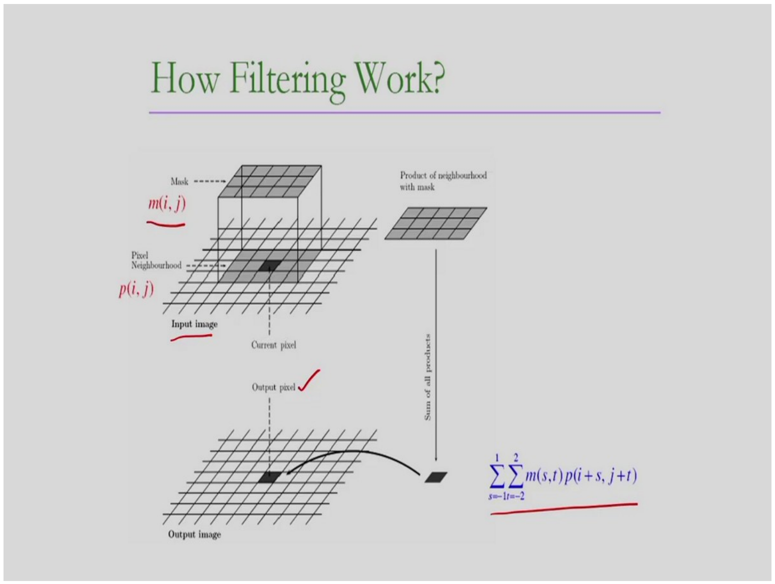
$$\boxed{R = w1z1 + w2z2 + \ldots + z9w9}$$ ✓

- Methods that do not satisfy the above property are called non-linear.
  - e.g., $R = max(z_k, k = 1, 2, \ldots, 9)$

And in this case I can show the difference between the linear filtering and non-linear filtering. I am considering the spatial filtering methods. And in this case if I consider this operation, that means I am determining the response of the mask. A filtering method is linear when the output is output is a weighted sum of the input pixels.

So, in this case this is the linear filtering. But if I consider this operation max, the max operation, I am considering 9 pixels because k is equal to 1 to 9. And out of 9 pixels I am determining the maximum pixel, that is the maximum valued pixel. Then in this case this is not the linear filtering. This is the example of the non-linear filtering. Because I am just determining the maximum value out of 9 pixels.

(Refer Slide Time: 25:15)



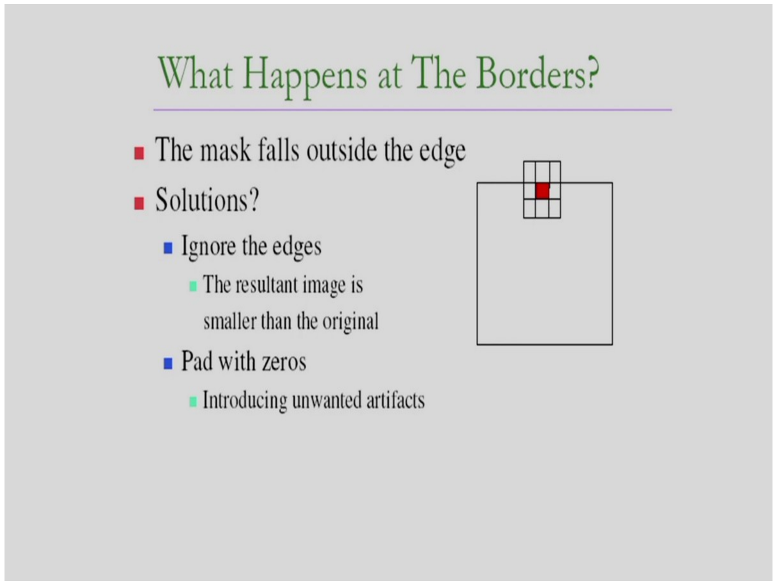
How Filtering Work?

$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(s,t)\,p(i+s,\,j+t)$$

And already I have explained how to do the spatial filtering. You can see here, I have the input image and this is the mask. And in the mask mainly I have to consider the neighborhood pixels. And in this case how to determine the output pixel? So, I have to place the mask over the image and after this I have to do the spatial convolution.

So, here you can see this expression here is nothing but the spatial convolution. Multiplication and the sum up I have to do. Like this I have to determine the filtered image or maybe output image you can determine.

(Refer Slide Time: 25:57)



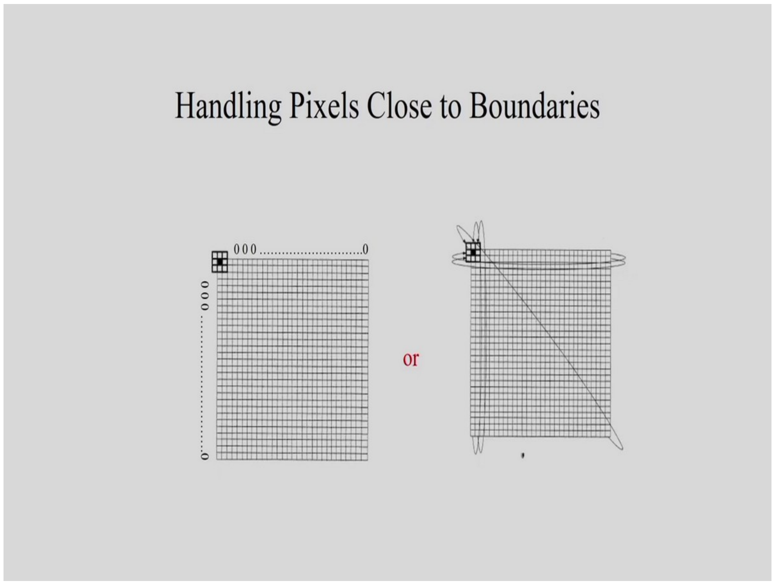
What Happens at The Borders?

- The mask falls outside the edge
- Solutions?
  - Ignore the edges
    - The resultant image is smaller than the original
  - Pad with zeros
    - Introducing unwanted artifacts

And one thing is that the mask falls outside the edge. Then in this case you can either ignore the edges the boundary pixels you can ignore or otherwise what we can do I can do the 0 padding in the boundary pixels. So, that I can put the mask in the boundary pixels.

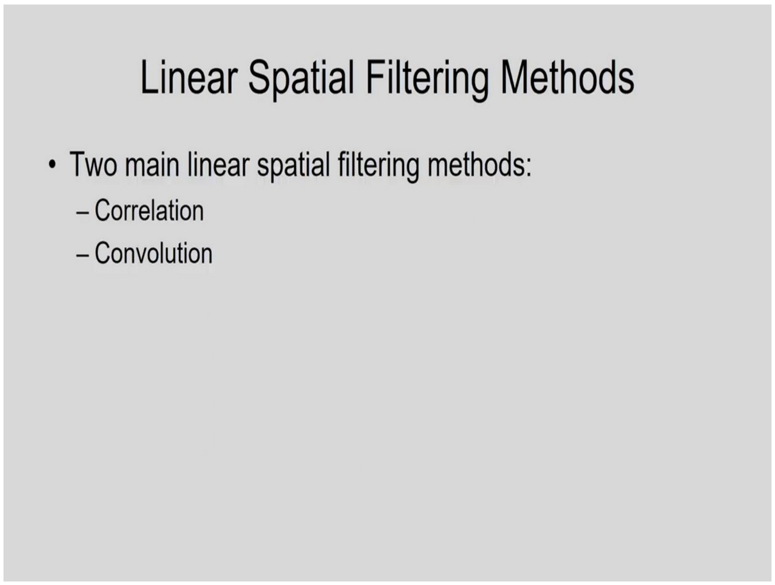
These two solutions, one is, I can neglect the boundary pixels that is the ignored edges I can consider. And if I ignore the edges, the resultant image will be smaller than the original. And if I consider the 0 padding, then in this case I will be getting unwanted artifacts.

(Refer Slide Time: 26:39)


Handling Pixels Close to Boundaries

Here I have shown these two cases, one is the 0 padding and another one is the I can neglect the boundary pixels I can neglect.

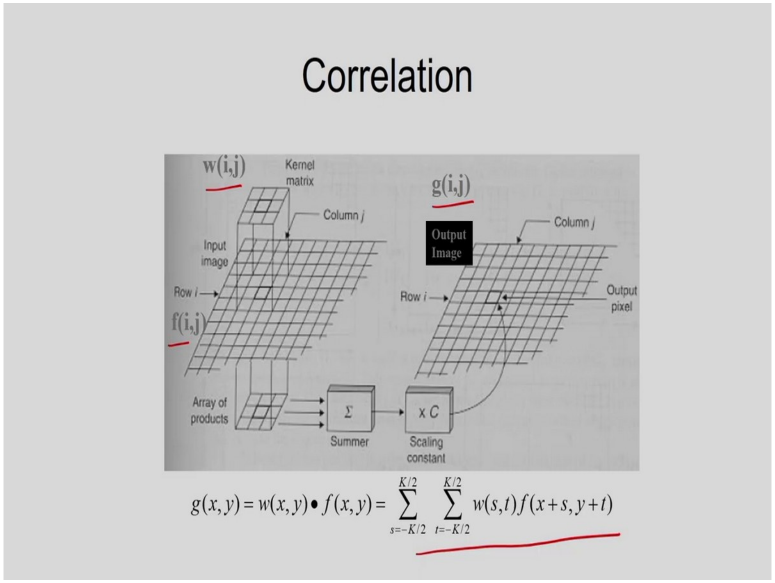
(Refer Slide Time: 26:47)


Linear Spatial Filtering Methods

- Two main linear spatial filtering methods:
  - Correlation
  - Convolution

Now in case of the linear filtering methods we have two operations, one is the correlation operation another one is the convolution operation. So, what is the correlation operation and how the correlation is related to convolution?

(Refer Slide Time: 27:01)



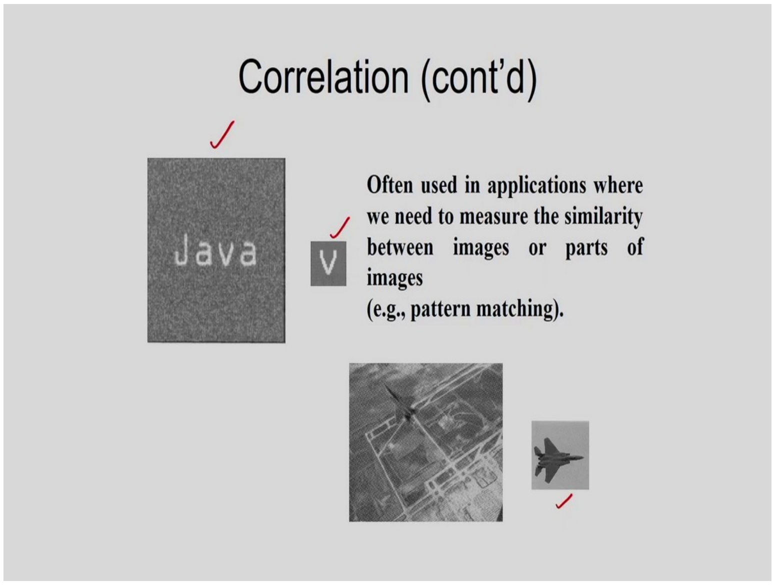
Correlation

$$g(x,y) = w(x,y) \bullet f(x,y) = \underline{\sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(x+s, y+t)}$$

In this case I have shown the same thing, the masking operation. I am considering the mask and I am considering the image. The image is f i j. And I am considering the mask that is the kernel matrix, the kernel matrix is w i j. And corresponding to this I am getting the output image the output image is g i j I am getting.

So, in this case what I have to do I have to put the mask over the image and I have to do the multiplication between the pixel value and the widths value of the mask. And after this I have to add all these things. So, that means it is nothing but the multiplication and the sum up. So, like this by using this expression you can see, for all the pixels of the image I have to do this operation. For this I have to shift the mask to the next pixel like this so that I can consider all the pixels of the image. So, w s t is nothing but that is the mask.

(Refer Slide Time: 28:10)



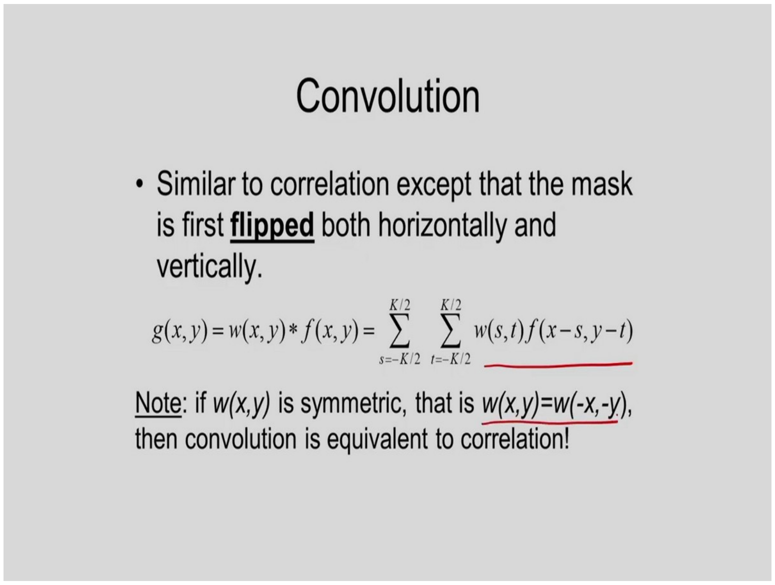
Correlation (cont'd)

Often used in applications where we need to measure the similarity between images or parts of images (e.g., pattern matching).

This is the correlation operation. And one application of the correlation operation in pattern matching you can see if I consider suppose this pattern, the pattern is this and I want to detect whether this pattern is available in the input image, my input image is this. So, if I find the correlation between the pattern and the image, then in this case based on the correlation I can detect whether particular pattern is available in the image or not.

In the second example also I can consider the pattern which pattern I am considering, and by using the correlation I can detect whether this pattern is available in the image or not, that I can detect. That is called pattern matching.

(Refer Slide Time: 28:56)



Convolution

- Similar to correlation except that the mask is first **flipped** both horizontally and vertically.

$$g(x,y) = w(x,y) * f(x,y) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} \underline{w(s,t) f(x-s, y-t)}$$

Note: if *w(x,y)* is symmetric, that is $\underline{w(x,y)=w(-x,-y)}$, then convolution is equivalent to correlation!

And what is the convolution? In the convolution what I have to, the mask is first flipped both horizontally and the vertically. That operation I have to do for the convolution. So, mask is first flipped both horizontally and the vertically, and you can see here. And after this the operation is very similar to the correlation, only difference is mask is first flipped both horizontally and vertically, and the rest of the operation is very similar to correlation.

But if I consider symmetric mask, the symmetric mask is something like this. Then in this case convolution is equivalent to correlation. So, for a symmetric mask the convolution is equivalent to correlation.

(Refer Slide Time: 29:48)

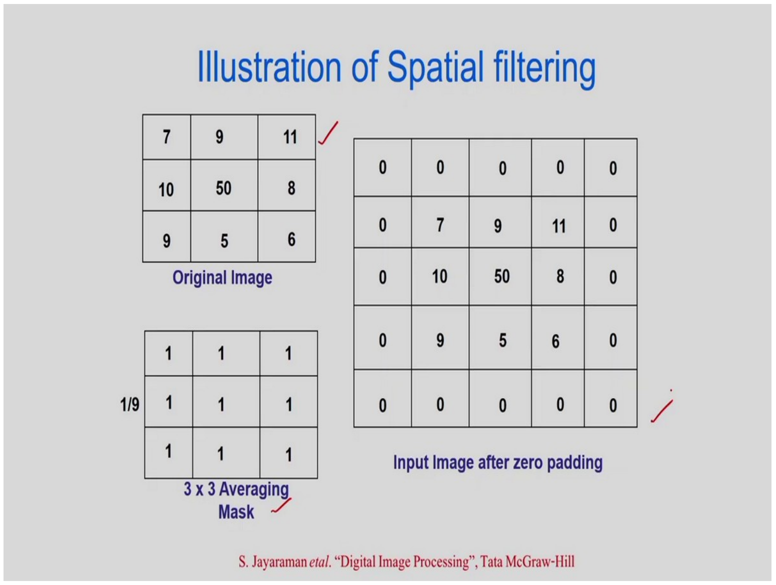


Illustration of Spatial filtering

7	9	11
10	50	8
9	5	6

Original Image

0	0	0	0	0
0	7	9	11	0
0	10	50	8	0
0	9	5	6	0
0	0	0	0	0

Input Image after zero padding

	1	1	1
1/9	1	1	1
	1	1	1

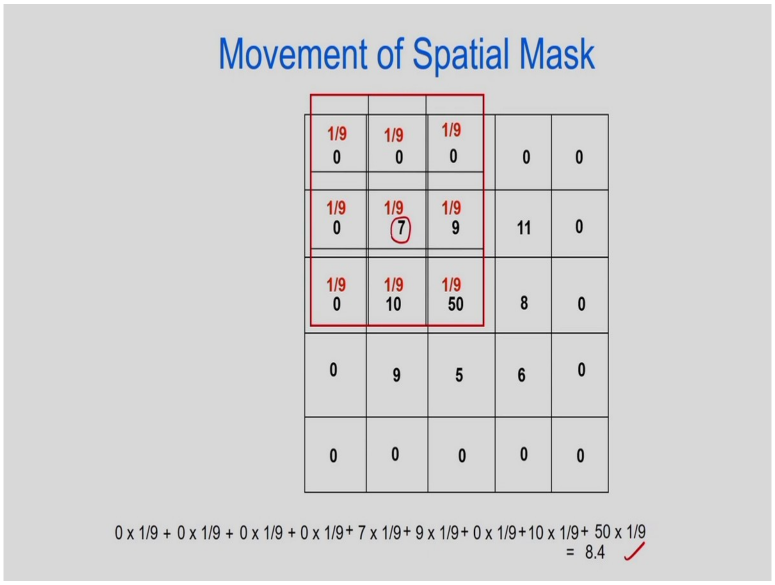
3 x 3 Averaging Mask

S. Jayaraman *etal.* "Digital Image Processing", Tata McGraw-Hill

And in this I want to give one illustration of the spatial filtering. And in this case I am considering the low pass filter. Here you can see I am considering the original image, and I am considering the mask, that is the 3 by 3 mask for the averaging operation. So, if you see the value, it is 1, 1, 1 and mainly it is divided by 9.

Because I am considering averaging over 9 pixels. So, this is the mask corresponding to the averaging filter. That is the low pass filter. And for considering the boundary pixels I am doing the 0 padding, so you can see the 0 padding I am doing. After this what I have to do? I have to put the mask over the image and I have to do the convolution.

(Refer Slide Time: 30:39)



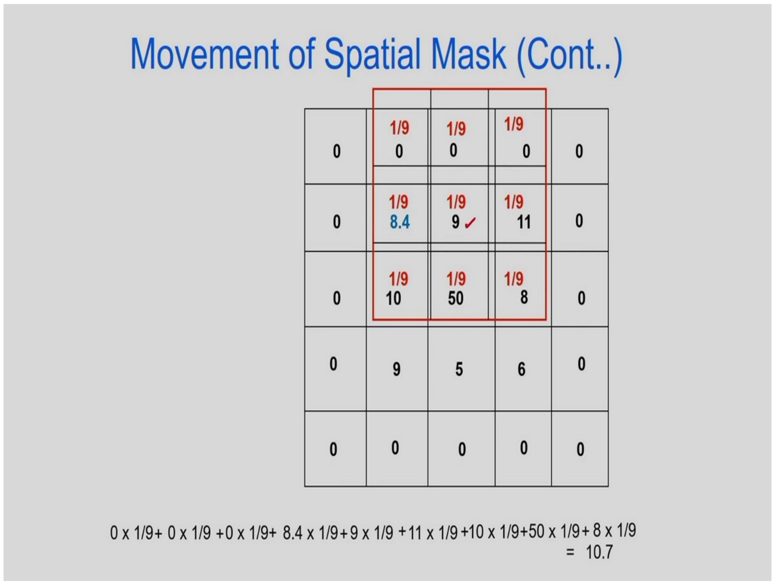
Movement of Spatial Mask

1/9 0	1/9 0	1/9 0	0	0
1/9 0	1/9 $\textcircled{7}$	1/9 9	11	0
1/9 0	1/9 10	1/9 50	8	0
0	9	5	6	0
0	0	0	0	0

$$0 \times 1/9 + 0 \times 1/9 + 0 \times 1/9 + 0 \times 1/9 + 7 \times 1/9 + 9 \times 1/9 + 0 \times 1/9 + 10 \times 1/9 + 50 \times 1/9 = 8.4 \checkmark$$

So, you can see in the next slide, here I have to put the mask over the image. And in this case I have to determine the response of the mask corresponding to the center pixel. The center pixel is 7 here. Corresponding of the center pixel, I am determining the response of the mask, the response of the mask is 8.4.

(Refer Slide Time: 31:01)



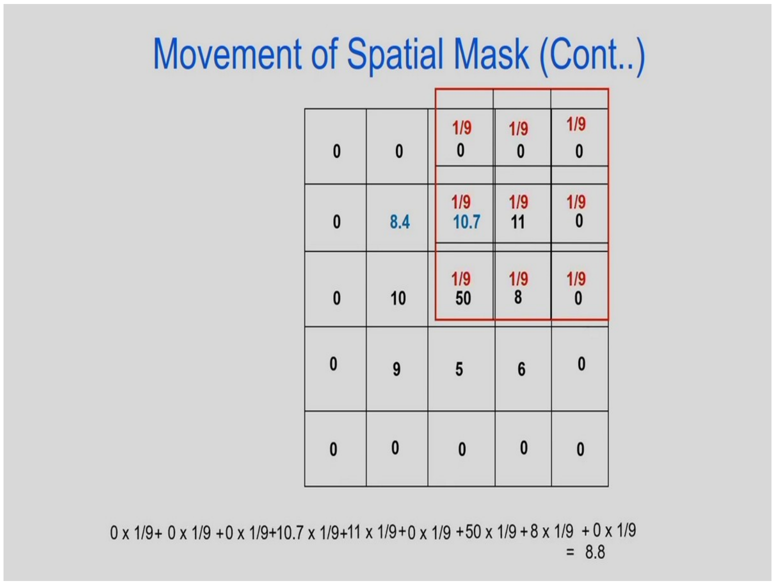
Movement of Spatial Mask (Cont..)

0	1/9 0	1/9 0	1/9 0	0
0	1/9 8.4	1/9 9 ✓	1/9 11	0
0	1/9 10	1/9 50	1/9 8	0
0	9	5	6	0
0	0	0	0	0

$$0 \times 1/9 + 0 \times 1/9 + 0 \times 1/9 + 8.4 \times 1/9 + 9 \times 1/9 + 11 \times 1/9 + 10 \times 1/9 + 50 \times 1/9 + 8 \times 1/9 = 10.7$$

After this what I have to do? I have to move the mask to the next pixel. And corresponding to the center pixel the center pixel is 9 in this case, corresponding to the center pixel I have to determine the response of the mask. So, the response of the mask I can determine so you can see. This is the response of the mask corresponding to the center pixel, the center pixel is 9. So, like this I have to do this operation for all the pixels of the image.

(Refer Slide Time: 31:34)



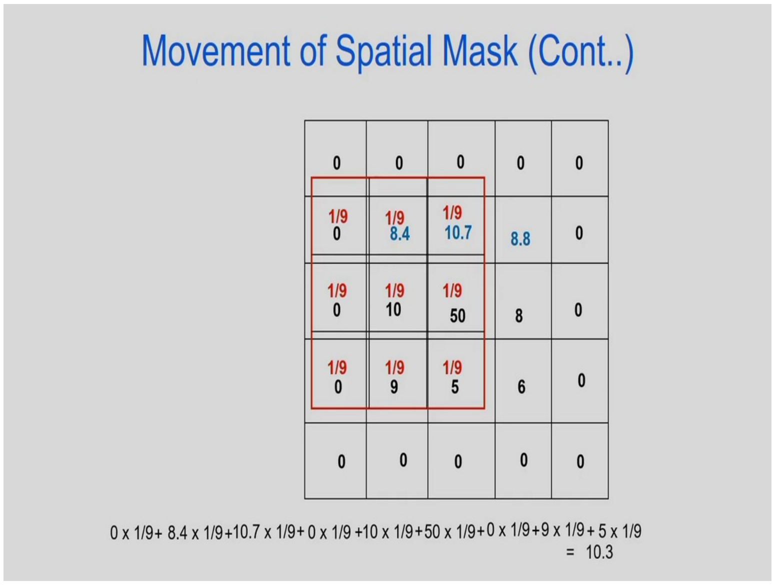
Movement of Spatial Mask (Cont..)

0	0	1/9 0	1/9 0	1/9 0
0	8.4	1/9 10.7	1/9 11	1/9 0
0	10	1/9 50	1/9 8	1/9 0
0	9	5	6	0
0	0	0	0	0

0 x 1/9+ 0 x 1/9 +0 x 1/9+10.7 x 1/9+11 x 1/9+0 x 1/9 +50 x 1/9 +8 x 1/9 +0 x 1/9
= 8.8

So, like this you can see, again the mask is shifted and I have to determine the response of the mask.

(Refer Slide Time: 31:48)



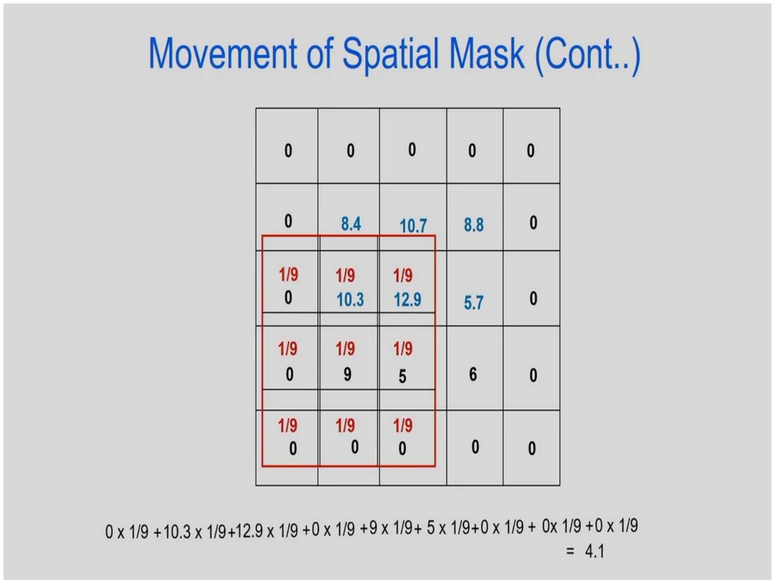
Movement of Spatial Mask (Cont..)

0	0	0	0	0
1/9 0	1/9 8.4	1/9 10.7	8.8	0
1/9 0	1/9 10	1/9 50	8	0
1/9 0	1/9 9	1/9 5	6	0
0	0	0	0	0

0 x 1/9+ 8.4 x 1/9+10.7 x 1/9+ 0 x 1/9 +10 x 1/9+50 x 1/9+0 x 1/9+9 x 1/9+ 5 x 1/9
= 10.3

And again you can see, I am moving the mask to the next pixel. And I am determining the response of the mask.

(Refer Slide Time: 32:04)



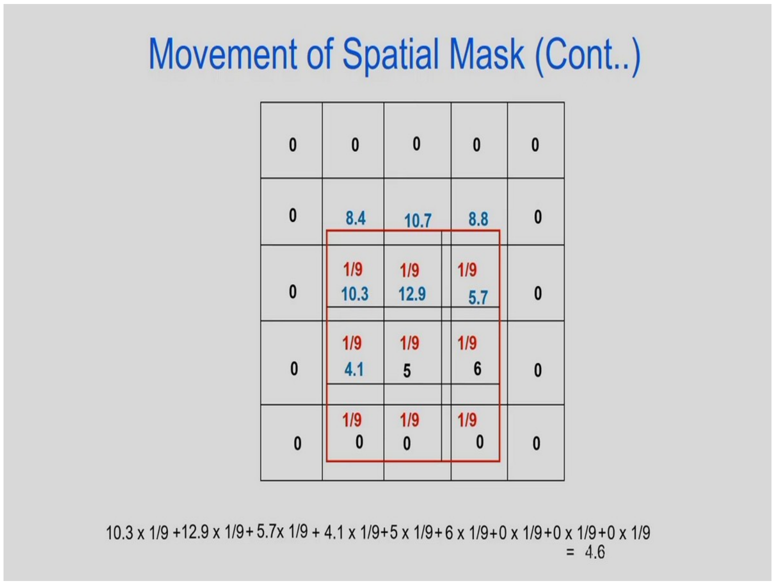
Movement of Spatial Mask (Cont..)

0	0	0	0	0
0	8.4	10.7	8.8	0
1/9 0	1/9 10.3	1/9 12.9	5.7	0
1/9 0	1/9 9	1/9 5	6	0
1/9 0	1/9 0	1/9 0	0	0

0 x 1/9 +10.3 x 1/9+12.9 x 1/9 +0 x 1/9 +9 x 1/9+ 5 x 1/9+0 x 1/9 + 0x 1/9 +0 x 1/9
= 4.1

And like this for all the pixels of the image I have to do this operation. So, in this demonstration you can see how to do the spatial filtering. And in this example I am only considering the low pass filter.

(Refer Slide Time: 32:20)



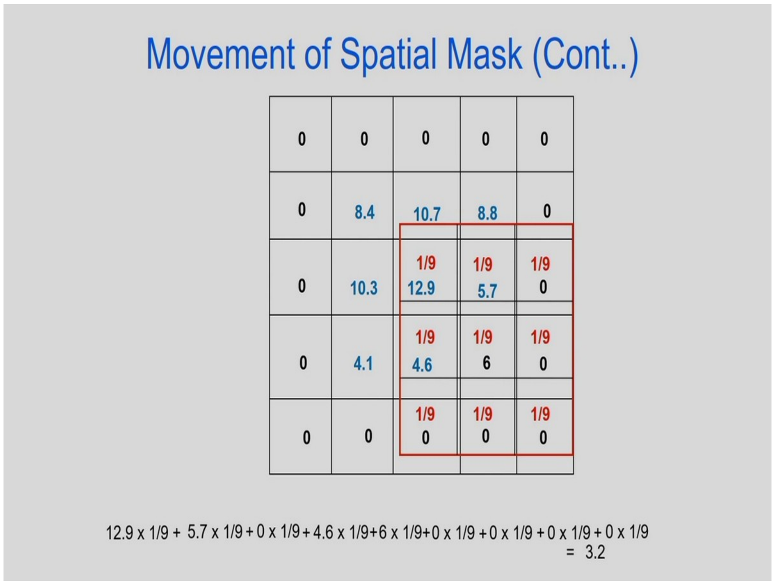
Movement of Spatial Mask (Cont..)

0	0	0	0	0
0	8.4	10.7	8.8	0
0	1/9 10.3	1/9 12.9	1/9 5.7	0
0	1/9 4.1	1/9 5	1/9 6	0
0	1/9 0	1/9 0	1/9 0	0

10.3 x 1/9 +12.9 x 1/9+ 5.7x 1/9 + 4.1 x 1/9+5 x 1/9+6 x 1/9+0 x 1/9+0 x 1/9+0 x 1/9
= 4.6

So, this operation is very similar to the spatial convolution.

(Refer Slide Time: 32:33)



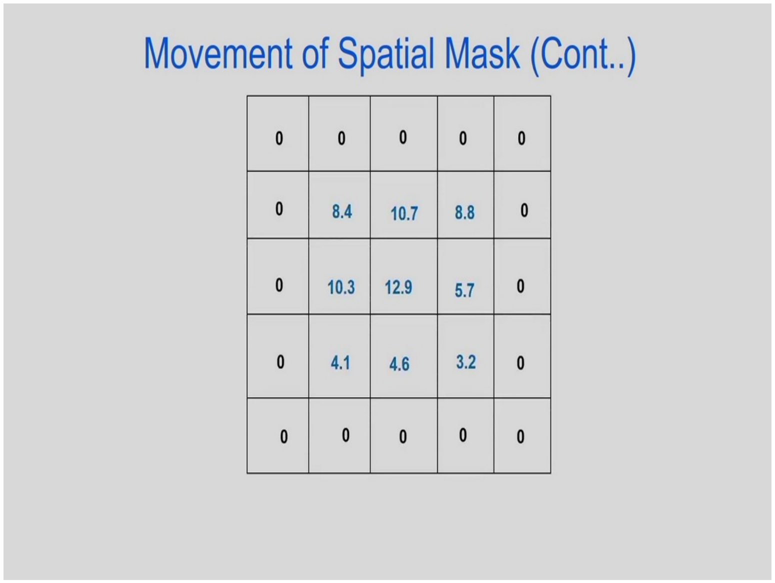
Movement of Spatial Mask (Cont..)

0	0	0	0	0
0	8.4	10.7	8.8	0
0	10.3	1/9 12.9	1/9 5.7	1/9 0
0	4.1	1/9 4.6	1/9 6	1/9 0
0	0	1/9 0	1/9 0	1/9 0

12.9 x 1/9 + 5.7 x 1/9 + 0 x 1/9 + 4.6 x 1/9+6 x 1/9+0 x 1/9 + 0 x 1/9 + 0 x 1/9 + 0 x 1/9 = 3.2

So, for all the pixels of the image I have to determine the response of the mask.

(Refer Slide Time: 32:45)

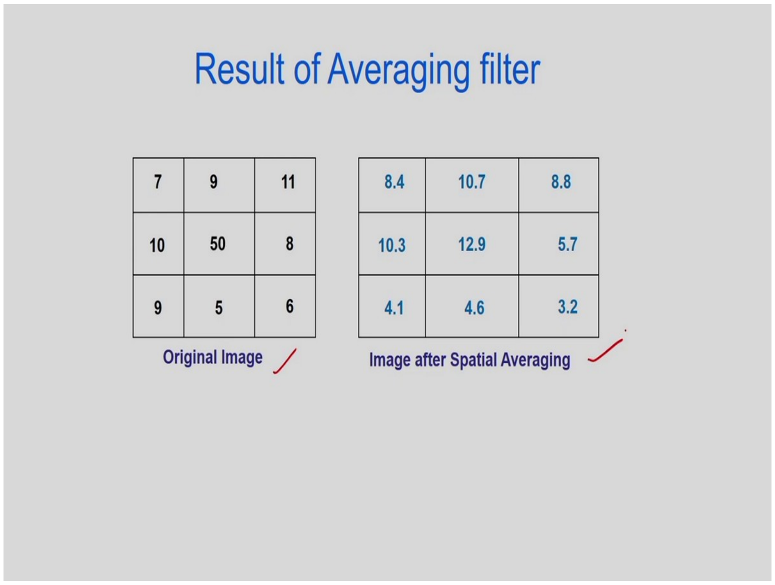


Movement of Spatial Mask (Cont..)

0	0	0	0	0
0	8.4	10.7	8.8	0
0	10.3	12.9	5.7	0
0	4.1	4.6	3.2	0
0	0	0	0	0

And finally I am getting this image.

(Refer Slide Time: 32:51)



Result of Averaging filter

7	9	11
10	50	8
9	5	6

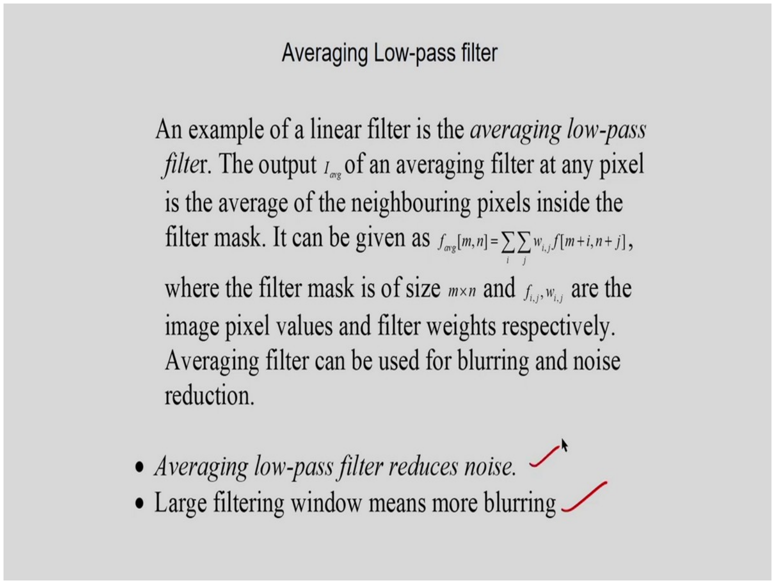
Original Image

8.4	10.7	8.8
10.3	12.9	5.7
4.1	4.6	3.2

Image after Spatial Averaging

So, you can see I have the original image and this is the image after spatial averaging that I am getting. So, you can see the result of the averaging filter.

(Refer Slide Time: 33:05)



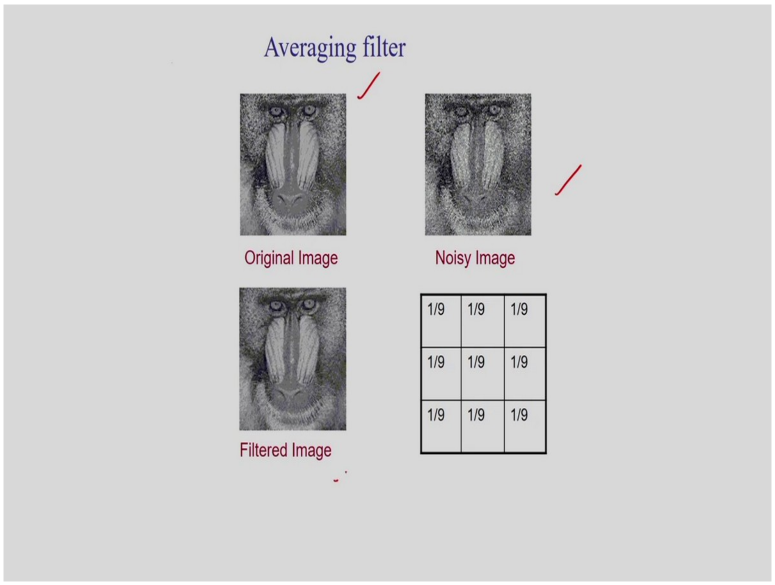
Averaging Low-pass filter

An example of a linear filter is the *averaging low-pass filter*. The output I_{avg} of an averaging filter at any pixel is the average of the neighbouring pixels inside the filter mask. It can be given as $f_{avg}[m,n]=\sum_i\sum_j w_{i,j} f[m+i,n+j]$, where the filter mask is of size $m\times n$ and $f_{i,j}, w_{i,j}$ are the image pixel values and filter weights respectively. Averaging filter can be used for blurring and noise reduction.

- *Averaging low-pass filter reduces noise.*
- Large filtering window means more blurring

So, if I do the averaging the averaging low pass filter reduces the noise. And in this case if I considered the large filtering window, that means the blurring will be more. In my previous example I have considered only the 3 by 3 mask, and if I considered the large filtering window then the blurring will be more. So, these are the case, the blurring will be more and the averaging low pass filter reduces noise.

(Refer Slide Time: 33:34)

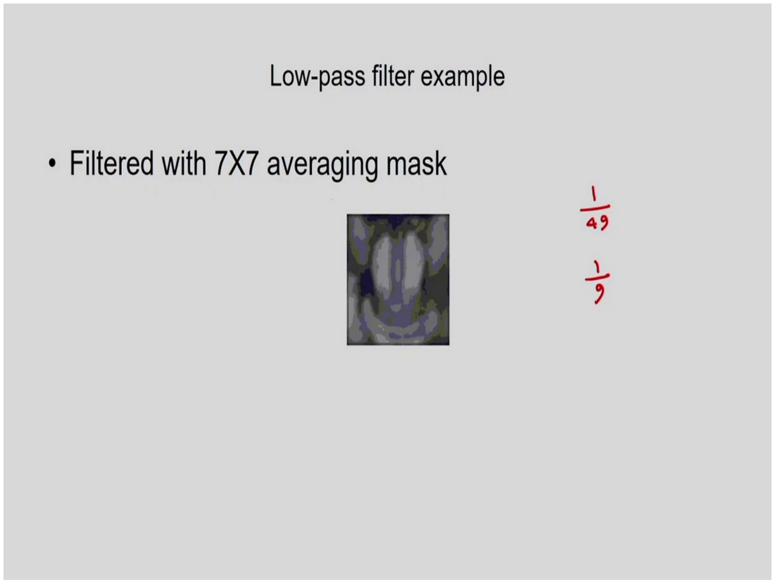


Averaging filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

And in this case I have shown one example of the averaging filter, original image is there and I am considering the noisy image, and I am having the filtered image. And this is the mask corresponding to the low pass filter, that is the averaging filter.

(Refer Slide Time: 33:51)



Low-pass filter example

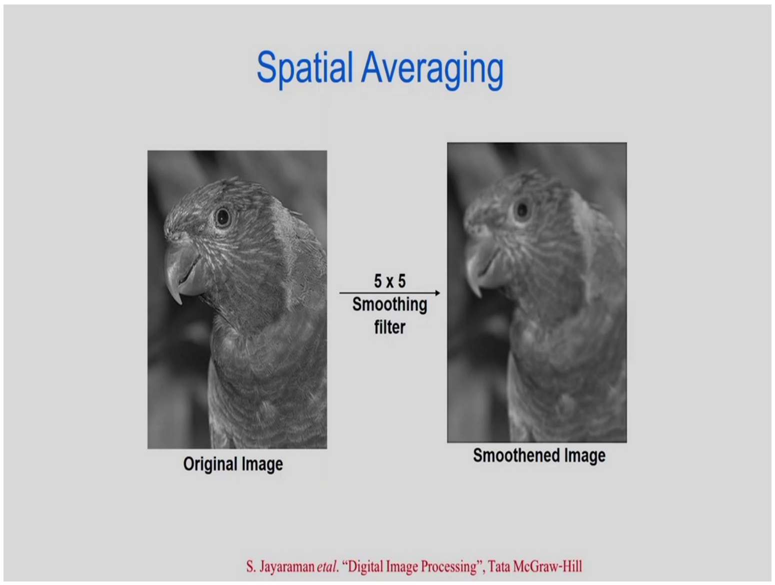
- Filtered with 7X7 averaging mask

$\frac{1}{49}$

$\frac{1}{9}$

And if I considered a 7 by 7 mask, that is the blurring will be more. Because if I consider 7 by 7 mask, this is nothing but I have to divide by 49. And if I consider 3 by 3, mask that means I am dividing by 9. So, that is why corresponding to the 7 by 7 averaging mask, the blurring will be more.

(Refer Slide Time: 34:14)



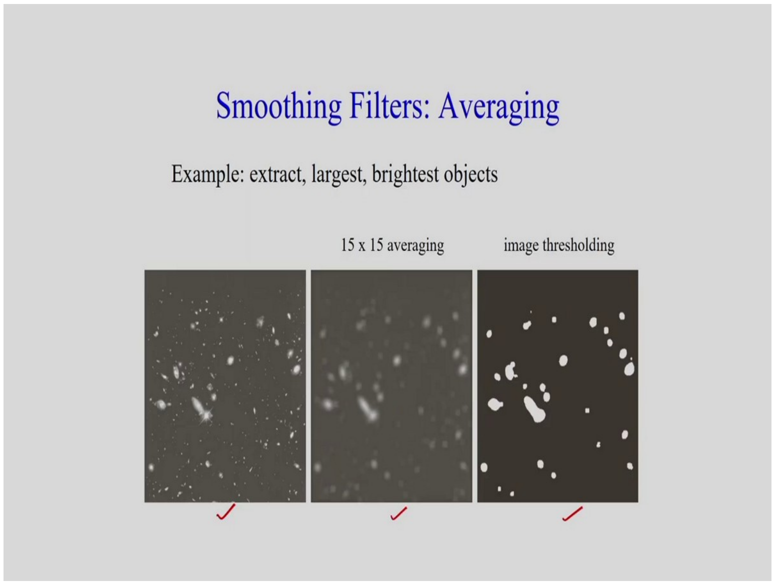
Spatial Averaging

S. Jayaraman *etal.* "Digital Image Processing", Tata McGraw-Hill

Here I am giving another example. So, original image is this and I am applying 3 by 3 mask, that is the averaging filter I am applying. And you can see the output image, that is the smoothened image I am having.

Similarly, I am considering another example, I have the original image and I am considering the 5 by 5 mask and I am having the smoothened image. So, in this case you can see that blurring is more as compared to the previous example.

(Refer Slide Time: 34:45)



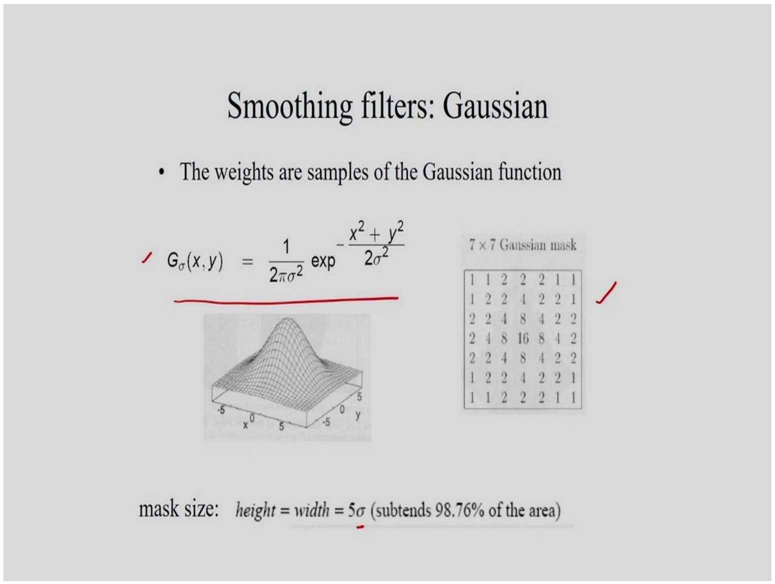
Smoothing Filters: Averaging

Example: extract, largest, brightest objects

15 x 15 averaging
image thresholding

And in this case I am considering the extraction of the brightest object in an image. So, my input image is this, and after this I am applying 15 by 15 averaging filter. And after this I am just applying the image thresholding technique. So, by this technique you can see I can extract the largest and the brightest object in the image. This is one application of the averaging filter.

(Refer Slide Time: 35:17)



Smoothing filters: Gaussian

- The weights are samples of the Gaussian function

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

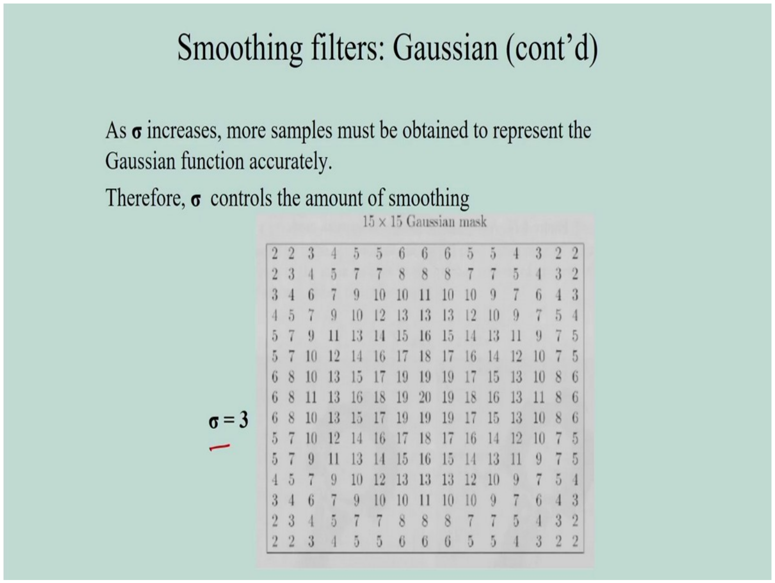
mask size: *height* = *width* = 5σ (subtends 98.76% of the area)

Next filter I am considering that is the smoothing filter, the Gaussian filter. In case of a Gaussian filter if you see, the weights are samples of the Gaussian function. So, I am considering the

Gaussian function here, this is the Gaussian function. Corresponding to this Gaussian function I have the Gaussian mask, here I am considering the 7 by 7 Gaussian mask.

And what about the weight of the mask? Weights are nothing but the weights are the samples of the Gaussian function. So I have the Gaussian function, only I have to put the value of x and y so that I will get the weights of the mask. And in this case if I considered the mask size, the mask size depends on the sigma. Because if I consider height is equal to width 5 sigma, then in this case area will be 98.76 percent of the area it will cover. So, the mask size depends on sigma.

(Refer Slide Time: 36:15)



Smoothing filters: Gaussian (cont'd)

As **σ** increases, more samples must be obtained to represent the Gaussian function accurately.

Therefore, **σ** controls the amount of smoothing

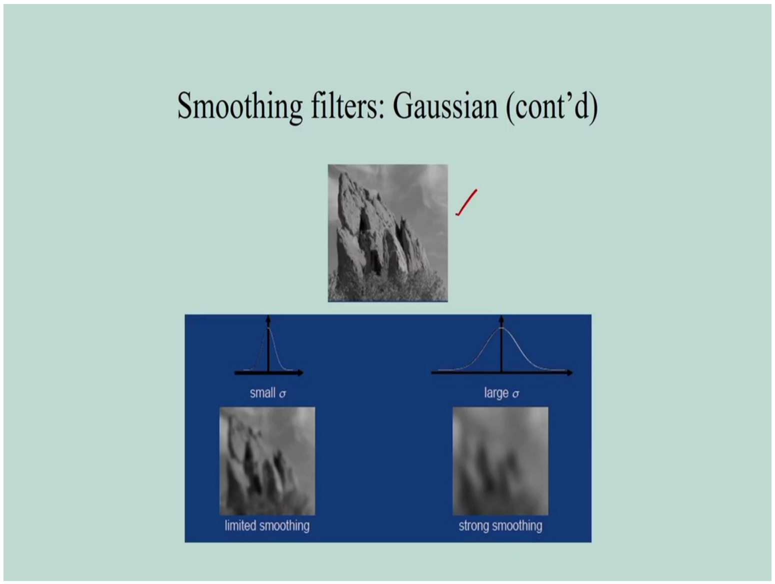
$\sigma = 3$

15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

And corresponding to sigma is equal to 3, you can see I am considering 15 by 15 Gaussian mask. So, in this case the parameter sigma controls the amount of smoothing. That is, it controls the amount of blurring. So, this is one example of the 15 by 15 Gaussian mask.

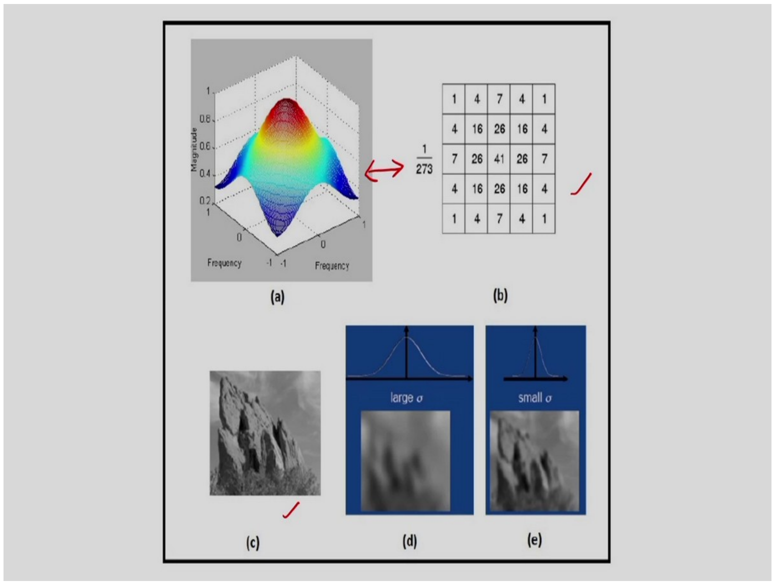
(Refer Slide Time: 36:36)



Smoothing filters: Gaussian (cont'd)

And corresponding to the Gaussian mask, you can see I have the input image, I am considering small sigma, then in this case it is limited smoothing. And if I consider large sigma, you can see more blurring, the strong smoothing. So the blurring depends on sigma. This is called the Gaussian blurring.

(Refer Slide Time: 37:01)



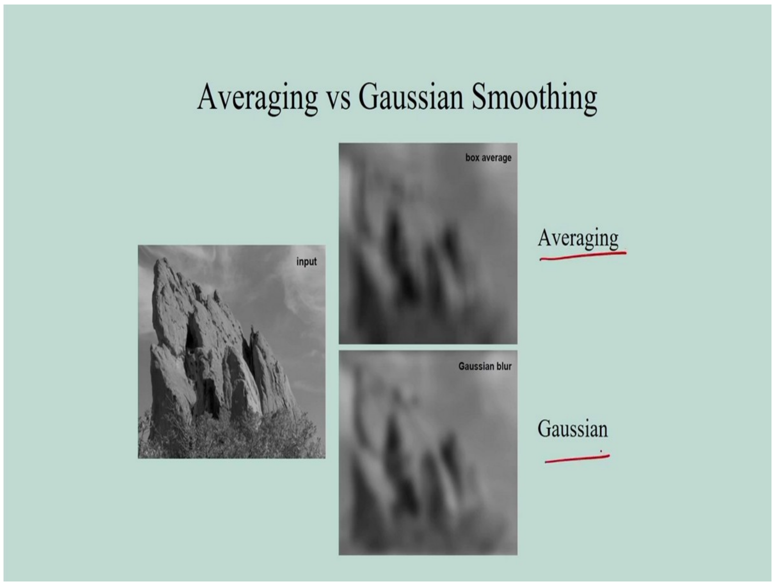
$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Again, I am showing another example. I am considering the Gaussian mask corresponding to this Gaussian function, corresponding to this Gaussian function I am considering the Gaussian mask.

And my input image is this. And I am considering two cases one is the large sigma another one is the small sigma. So, for the large sigma blurring is more as compared to small sigma.

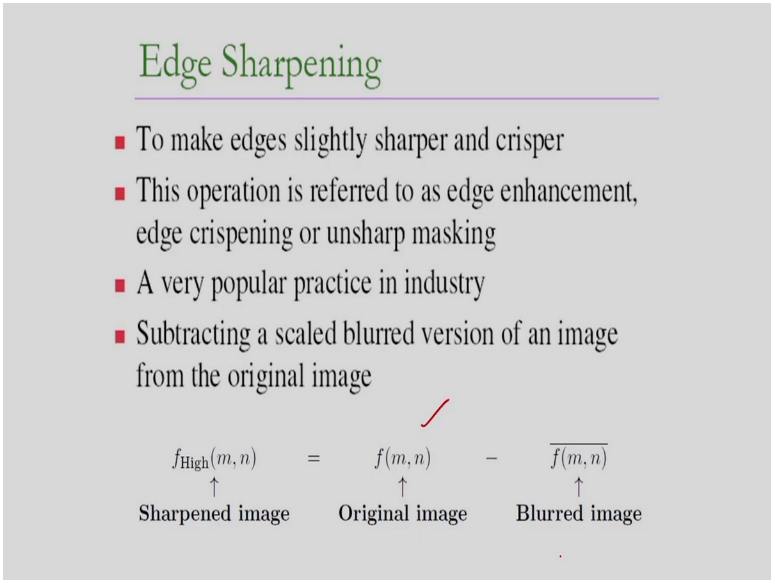
(Refer Slide Time: 37:27)



Averaging vs Gaussian Smoothing

And you can see the difference between the Gaussian smoothing and the averaging smoothing. One is the averaging I am considering; another one is the Gaussian smoothing I can consider. That called the Gaussian blur.

(Refer Slide Time: 37:40)



Edge Sharpening

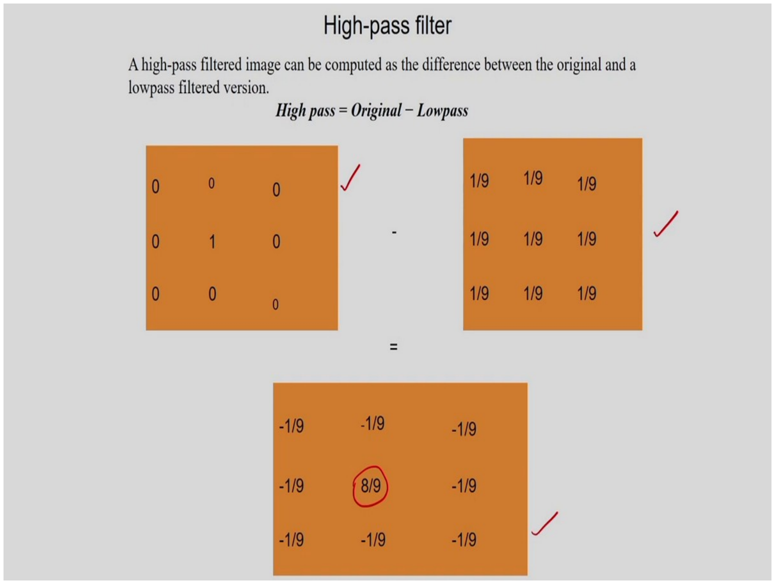
- To make edges slightly sharper and crisper
- This operation is referred to as edge enhancement, edge crispening or unsharp masking
- A very popular practice in industry
- Subtracting a scaled blurred version of an image from the original image

$$\underset{\text{Sharpened image}}{\underset{\uparrow}{f_{\text{High}}(m,n)}} = \underset{\text{Original image}}{\underset{\uparrow}{f(m,n)}} - \underset{\text{Blurred image}}{\underset{\uparrow}{\overline{f(m,n)}}}$$

The next filter I want to consider the high pass filter, that is used for sharpening edges. Then in this case what I have to do? I have to develop a high pass filter, I have the original image f m and if I subtract the blurred image, that is the low pass filtered image from the original image, then in this case I will be getting the sharpened image, that is nothing but I will be getting high frequency components.

So, a sharpened image is represented by f High m n, that is nothing but the high frequency components I will be getting corresponding to the edges and the boundaries. So, this is the definition of the high pass filter. So what I have to do, I have to subtract the blurred image, that is the average version of the image from the original image, then in this case I will be getting the certain image. This operation is important for edge enhancement so I can highlight the edges.

(Refer Slide Time: 38:42)



High-pass filter

A high-pass filtered image can be computed as the difference between the original and a lowpass filtered version.

High pass = Original − Lowpass

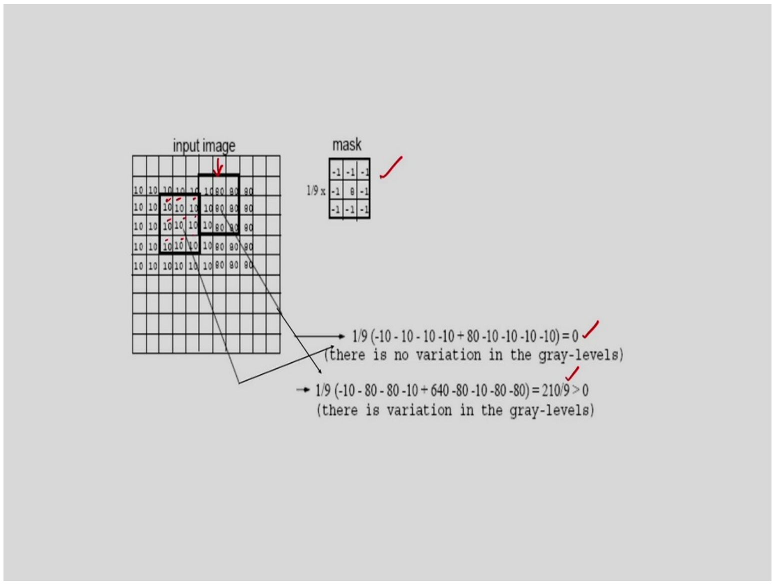
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \checkmark - \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \checkmark$$

$$= \begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix} \checkmark$$

So, in this case I have shown the mask corresponding to the high pass filter. So, first mask if you see this one, if I apply this mask nothing will happen, so that means I will be getting the same image. That is why it is called the identity mask. The original image I will be getting.

Original image minus the average portion of the image that is the mask corresponding to the low pass filtering 1 by 9, 1 by 9, 1 by 9 like this, so this is the mask corresponding to the high pass filter. So, central pixel is 8 by 9 and remaining you can see, it is minus 1 by 9, minus 1 by 9, minus 1 by 9 like this. So, you can see how to get the high pass filter mask.

(Refer Slide Time: 39:26)

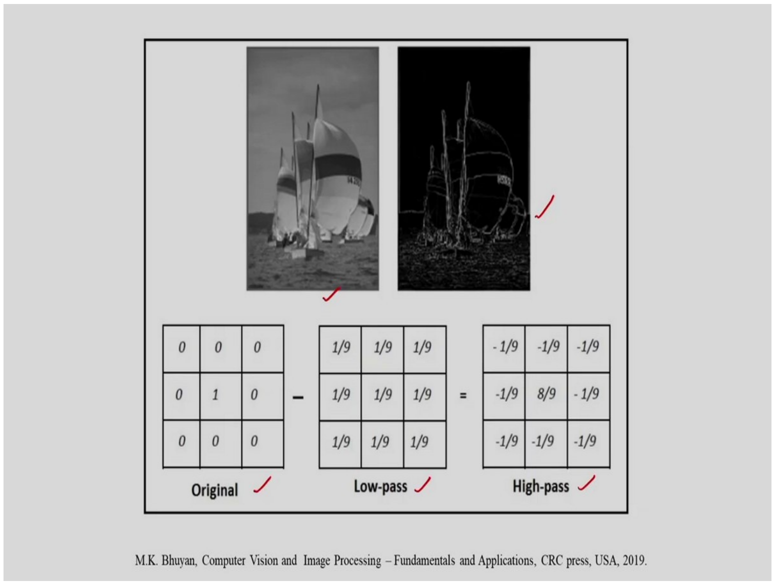


And if I apply this mask, the high pass filter mask, this is the mask corresponding to the high pass filter. You can see here, I am considering one input image. And in the input image I am considering the constant pixel value, if you see the 10, 10, 10 like this, this value.

Corresponding to this portion of the image what will be the response of the mask? If you apply the convolution operation and if you determine the response of the mask you can see corresponding to that portion the response is 0. But if I see here that this portion the pixel value is suddenly changing from 10 to 80, that means that corresponds to edges or the boundaries.

So, corresponding to that location if I apply the mask then you can see the response of the mask is this. So, that means corresponding to the high frequency I will be getting the response corresponding to the high pass filtered mask. So, this is the mask corresponding to the high pass filter.

(Refer Slide Time: 40:32)

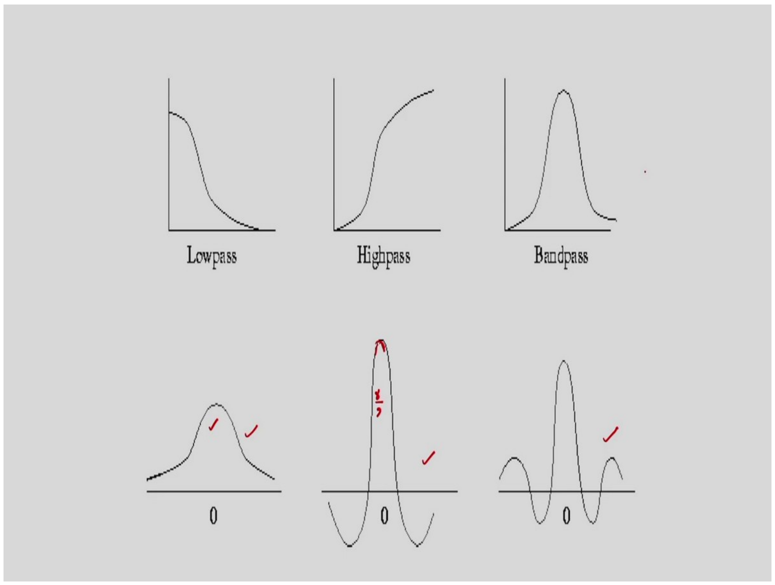


$$\underset{\text{Original}}{\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}} - \underset{\text{Low-pass}}{\begin{array}{|c|c|c|}\hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline\end{array}} = \underset{\text{High-pass}}{\begin{array}{|c|c|c|}\hline -1/9 & -1/9 & -1/9 \\ \hline -1/9 & 8/9 & -1/9 \\ \hline -1/9 & -1/9 & -1/9 \\ \hline\end{array}}$$

M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

And here I am showing one example, the original image I am considering, that is the original mask I am considering that is nothing but the identity mask. And this is a low pass filter mask. And if I subtract the low pass filter mask from the original, I will be getting the mask corresponding to the high pass filter. And if I apply this filter in this image, the image is this, the input image, then in this case I will be getting the edges. So, that means edge sharpening I can do by using the high pass filter. Because I am considering only the high frequency components.

(Refer Slide Time: 41:08)

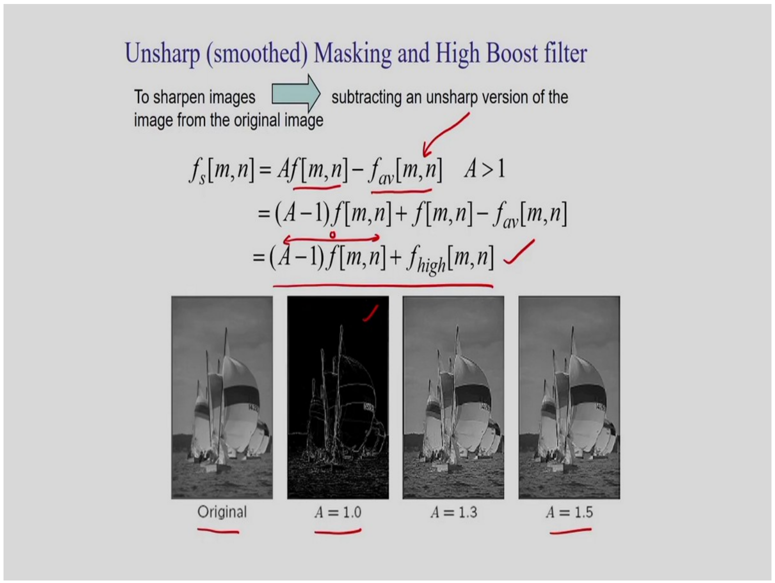


And in this case you can see, in frequency domain I have shown the first row is the frequency domain. Corresponding to the low pass filter, high pass filter and the band pass filter. In the second row I am showing the response in the spatial domain.

In the spatial domain this is the response for the low pass filter, this is the response for the high pass filter and this is the response for the band pass filter. So, in this case the central pixel already you know, the central pixel is 8 by 9, so that means in this case it is a maximum value here you are getting.

And in this case it is the averaging operation I am considering. So, you can see the concept of the low pass filter, high pass filter and the band pass filter in frequency domain and in the spatial domain. First row is the frequency domain and second row are the spatial domain.

(Refer Slide Time: 42:00)



Unsharp (smoothed) Masking and High Boost filter

To sharpen images ⇨ subtracting an unsharp version of the image from the original image

$$f_s[m,n] = A\underline{f[m,n]} - \underline{f_{av}[m,n]} \quad A > 1$$
$$= (A-1)f[m,n] + f[m,n] - f_{av}[m,n]$$
$$= \underline{(A-1)f[m,n] + f_{high}[m,n]} \checkmark$$

Original
$A = 1.0$
$A = 1.3$
$A = 1.5$

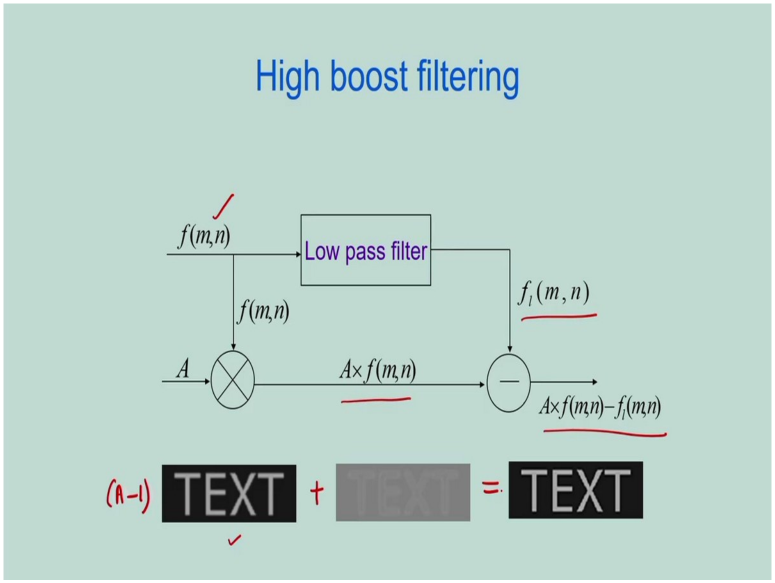
The next filter is the high boost filter. So, in this case what I have to consider, subtracting an unsharp version of the image from the original image. So, that means what I am considering, if you see I am considering the average version of the image, that is nothing but the low pass filtered image, and I am considering the original image, original image is f m n. And I am considering a factor, the factors A. The A is greater than 1.

So, what I am doing the subtracting the unsharp version of the image from the original image. The unsharp version of the image means this is the unsharp version of the image. That is the average image. And after this you can see this operation, so I can write like this, A minus 1 f m n plus f m n minus f average I can do. And finally, I will be getting this expression, A minus 1 f m n, that is the original image; and after this, f High m n, that is the high frequency image. So, if I put A is equal to 1, then in this case this component will be 0, if I considering this A is equal to 1 so this will be 0. Then in this case I will be getting the high pass filtered output.

So, this is my original image if I consider A is equal to 1 then only I will be getting the high frequency components. If A is equal to 1.3, then in this case you can see if you put this value in this expression A is equal to 1.3, then in this case corresponding to 1.3 I have the high frequency information as well as the low frequency information. So, in this output I do not have the low frequency information but in the second case you can see, because of A is equal to 1.3, that means I am considering low frequency information as well as the high frequency information.

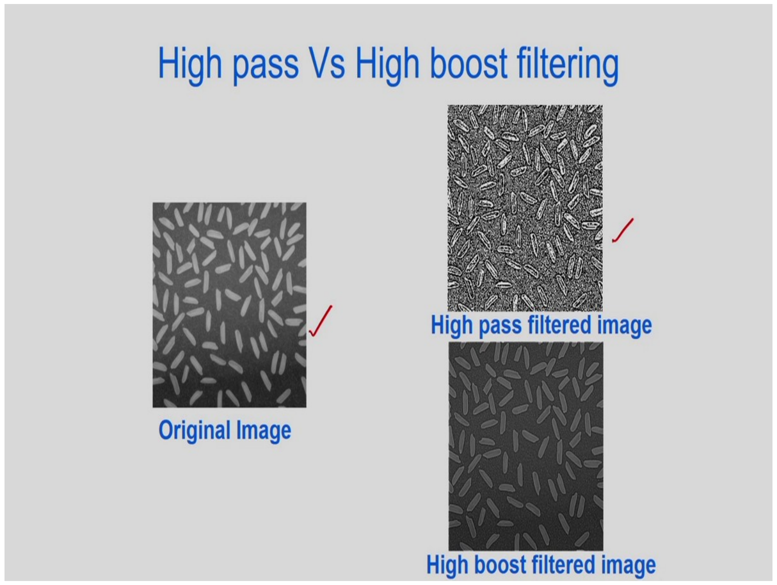
If I consider the A is equal to 1.5, that means I am considering more low frequency information and as well as high frequency information I am considering. So, by applying the high boost filter I can consider both low frequency information and the high frequency information I can consider. So, I am getting the edges, that is sharpening the edges as well as I am having the low frequency components.

(Refer Slide Time: 44:31)



And in this case I am showing the same thing, the high boost filtering, f m n is the input image and I am considering A factor, A cross f m n. And in this case I am considering the average version of the image, and this is nothing but the high boost filter I am considering. So, you can see this is my original image and I am considering suppose it is A minus 1. Plus, I am considering the blurred version of the image and I am getting the output, the output is this I am getting. That is the high boost filtering.

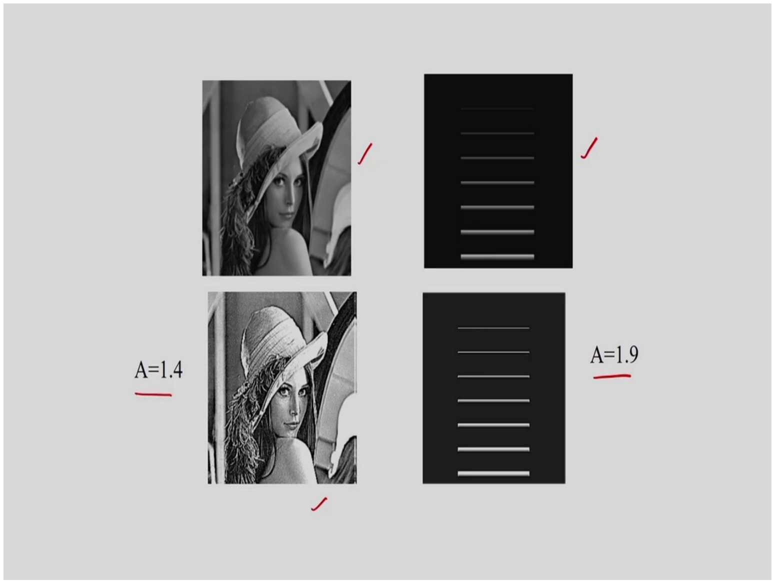
(Refer Slide Time: 45:17)



High pass Vs High boost filtering

You can see the difference between the high pass filter and the high boost filter. So, I am considering the original image is this, the first output is the high pass filter image. I am having the high pass filtered image. And second one is the high boost filter. In the high boost filter, you can see I have the background information along with the edge information. So, in the high boost filter I have the high frequency information as well as the low frequency information. But in case of the high pass filtered image I have only the high frequency information, that means I have the edges.

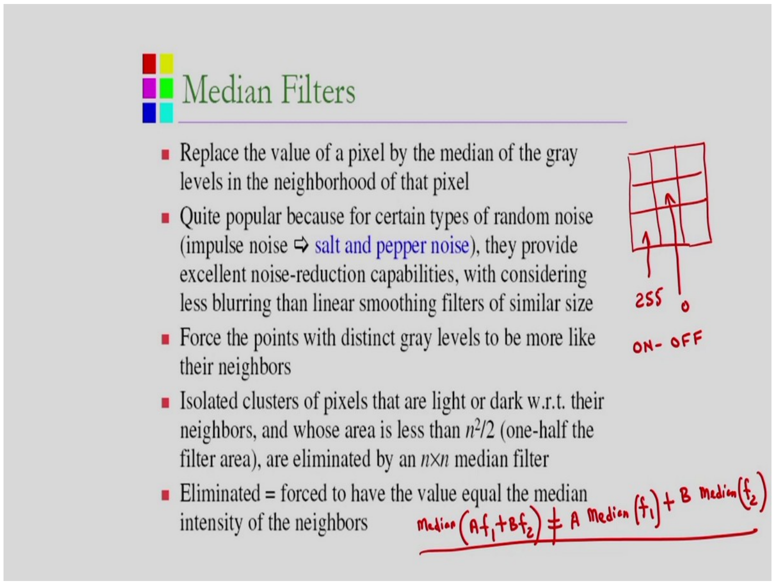
(Refer Slide Time: 45:51)46.21



And in this case I can show these two examples. Corresponding to this image I may applying the high boost filter. And in this case I am considering A is equal to 1.4, and corresponding to this I am getting the output image, this. That means I am getting the background information as well as I am sharpening the edges.

And corresponding to the second example, I am considering this is the input image and I am considering A is equal to 1.9. Corresponding to A is equal to 1.9 I have the output image, this is the output image. So, that means I am sharpening the edges and also I have the low frequency information.

(Refer Slide Time: 46:30)48.21



Median Filters

- Replace the value of a pixel by the median of the gray levels in the neighborhood of that pixel
- Quite popular because for certain types of random noise (impulse noise ⇨ salt and pepper noise), they provide excellent noise-reduction capabilities, with considering less blurring than linear smoothing filters of similar size
- Force the points with distinct gray levels to be more like their neighbors
- Isolated clusters of pixels that are light or dark w.r.t. their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an $n \times n$ median filter
- Eliminated = forced to have the value equal the median intensity of the neighbors

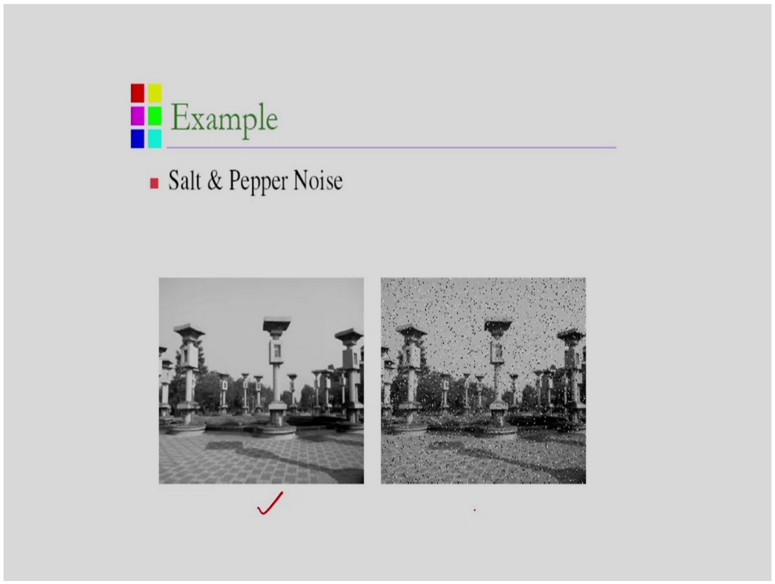
$\text{Median}(Af_1+Bf_2) \neq A\,\text{Median}(f_1) + B\,\text{Median}(f_2)$

The next filter is the median filter. So, median filter is applicable for to remove the salt and pepper noise. The salt and pepper noise is mainly the impulse noises. I have the means so these pixel values will be affected by the impulse noises. And in this case the high value of this, the high value of this maybe is 255, but the low value of this noise may be 0. This is something like ON and OFF noise. That is the impulse noise I am considering.

And in this the median filter operation what I have to consider, I have to consider a mask, and within this mask I have to consider all the pixels, the neighborhood pixels I have to consider. And this central pixel value is replaced by the median value, so that concept I am going to explain. And one thing it is important that median is a non-linear filter.

Why it is a non-linear filter? So, I can show you, suppose if I take the median A f1 plus B f2 that is not equal to A median f1 plus B median f2. So, if you see this one the median is a non-linear filter. The median A f1 plus B f2 is not equal to A median f1 plus B median f2. And how to apply the median filter? I can explain you in the next slide.

(Refer Slide Time: 48:24)

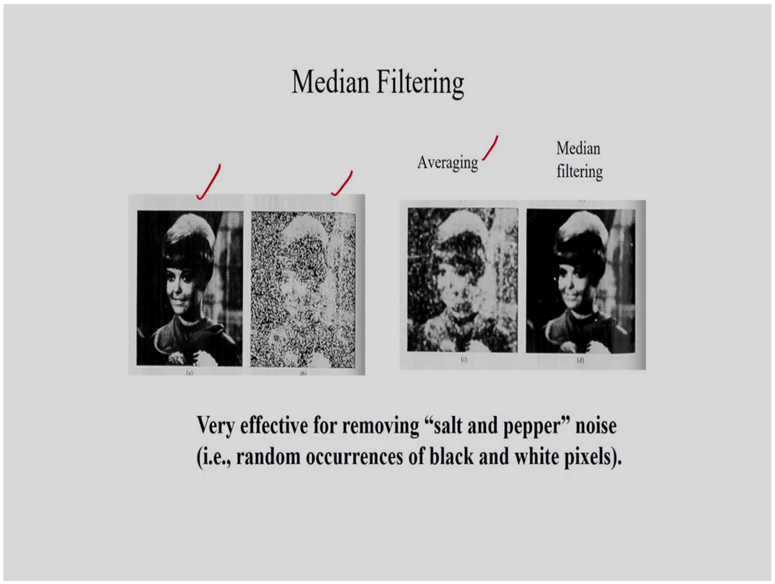


Example

- Salt & Pepper Noise

So, this is one example of the salt and pepper noise. The salt and pepper noise is nothing but the impulse noises. And it is not affecting all the pixels of the image. Like that in Gaussian noise the Gaussian noise affects all the pixels of the image. But in case of the salt and pepper noise, it affects only a few pixels of the image and that is nothing but ON and OFF noise. So, my input image is this and I am considering the salt and pepper noise.

(Refer Slide Time: 48:56)



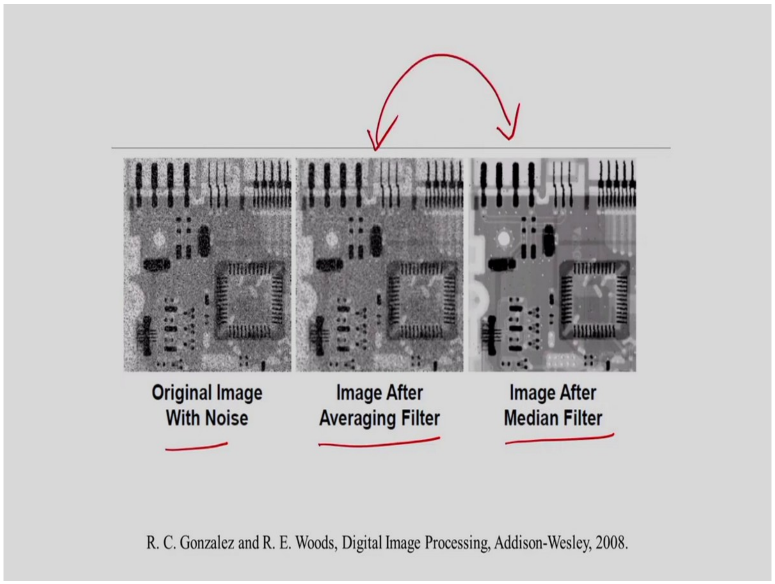
Median Filtering

Averaging
Median filtering

Very effective for removing "salt and pepper" noise (i.e., random occurrences of black and white pixels).

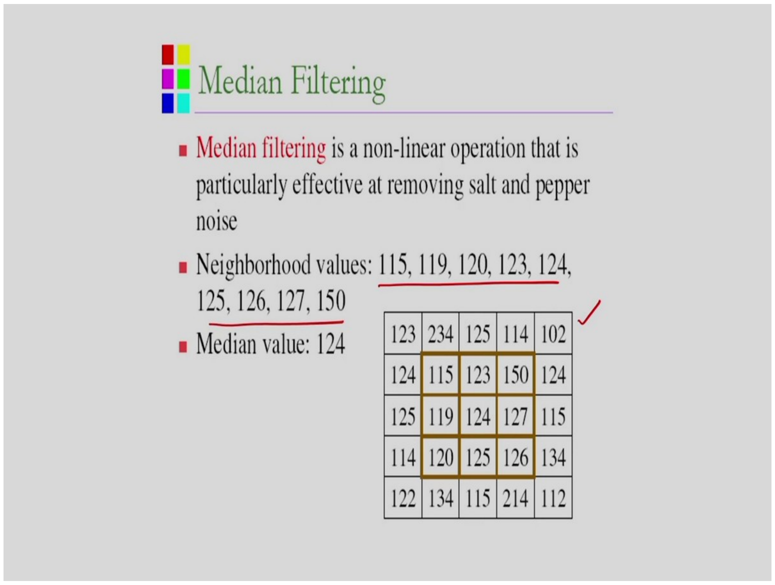
Again I am considering here you can see my input image is this and I am considering the salt and pepper noise. And in the first case I am applying the averaging operation and in the second case I am applying the median filter operation. You can see the difference between the averaging operation and the median filtering operation. So, this median filter is very effective to remove salt and pepper noise.

(Refer Slide Time: 49:23)


Original Image With Noise
Image After Averaging Filter
Image After Median Filter
R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

And here also I have shown one example. The original image with noise and in first case I am applying the averaging filter that is the output of the averaging filter, in the second case I am considering the median filter. You can see the distinction between these two images, one is the averaging filter another one is the median filter. So, I think the median filter gives better result as compared to averaging filter.

(Refer Slide Time: 49:49)



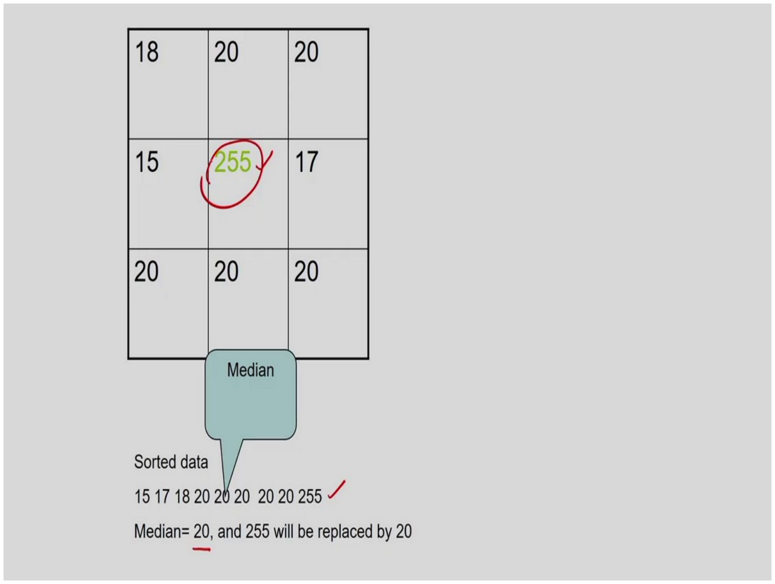
Median Filtering

- Median filtering is a non-linear operation that is particularly effective at removing salt and pepper noise
- Neighborhood values: 115, 119, 120, 123, 124, 125, 126, 127, 150
- Median value: 124

123	234	125	114	102
124	115	123	150	124
125	119	124	127	115
114	120	125	126	134
122	134	115	214	112

And how to apply the median filter? So, in this example I am considering one image my image is this the pixels value I have shown. And I am considering a 3 by 3 mask and I have to consider a neighborhood values. The neighborhood value is 115, 119, 120, these are the neighborhood values. And for these values I have to determine the median value. So, for this I have to arrange in the ascending order or the descending order, and after this I have to find the median value. So, in this case the median value will be 124. So, like this I have to determine the median value.

(Refer Slide Time: 50:27)



18	20	20
15	255	17
20	20	20

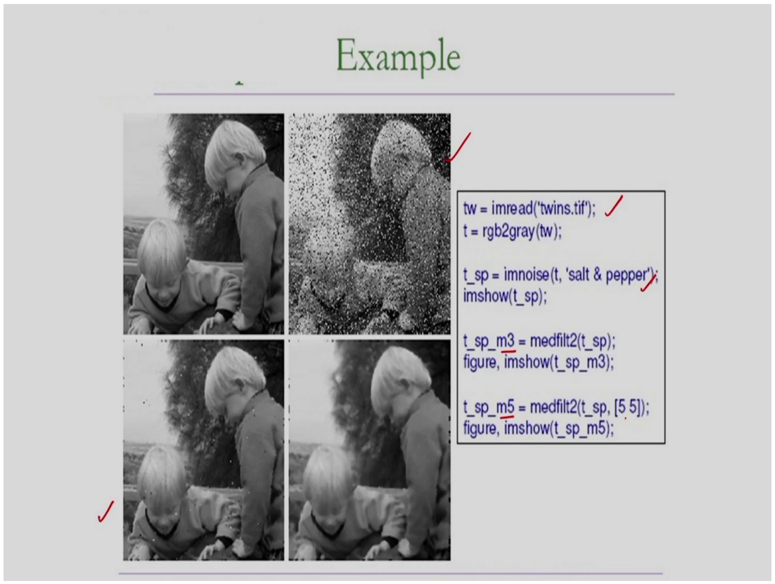
Sorted data

15 17 18 20 20 20 20 20 255

Median= 20, and 255 will be replaced by 20

And in this case, in this example suppose if I consider the 3 by 3 mask, and in this case this pixel is affected by the salt and pepper noise, because the value is 255. Then in this case, corresponding to this 3 by 3 window I have the pixel values, the pixel values are 15, 17, 18, 20, 20, 20, 20, 20, 255. That means I have to arrange in the ascending order, and after this I have to find the median value. So, median value will be 20 then this 255 pixel value that will be replaced by 20. So, this is the concept of the median filter.

(Refer Slide Time: 51:07)

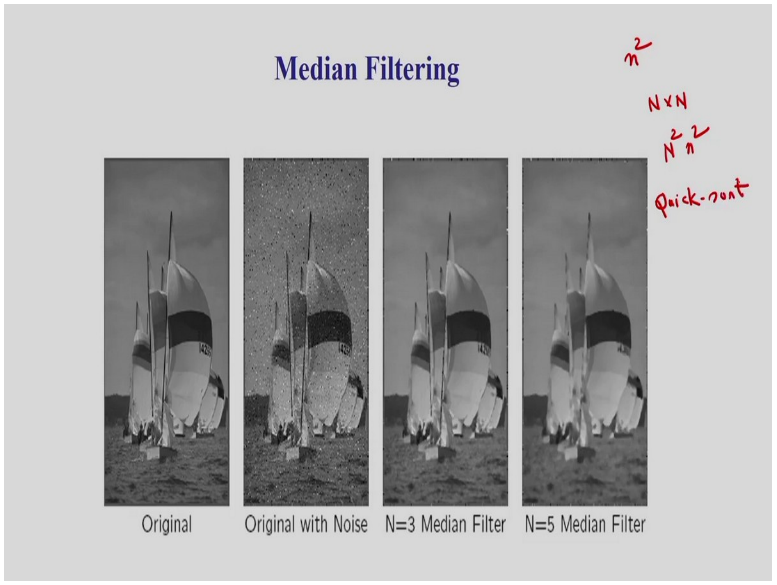


Example

```
tw = imread('twins.tif');
t = rgb2gray(tw);

t_sp = imnoise(t, 'salt & pepper');
imshow(t_sp);

t_sp_m3 = medfilt2(t_sp);
figure, imshow(t_sp_m3);

t_sp_m5 = medfilt2(t_sp, [5 5]);
figure, imshow(t_sp_m5);
```

And in this case I am showing one example. The first one is the input image and after this I am considering the salt and pepper noise. And after this what I am considering? If you see first one is the input image I am considering after this I am applying the salt and pepper noise. And after this I am applying the 3 by 3 median filter I am applying. So, corresponding to the 3 by 3 filter I am getting this output. The next one is, I am applying the 5 by 5 median filter, the 5 by 5 median filter I am applying. And corresponding to this I have the output this one.

(Refer Slide Time: 51:43)



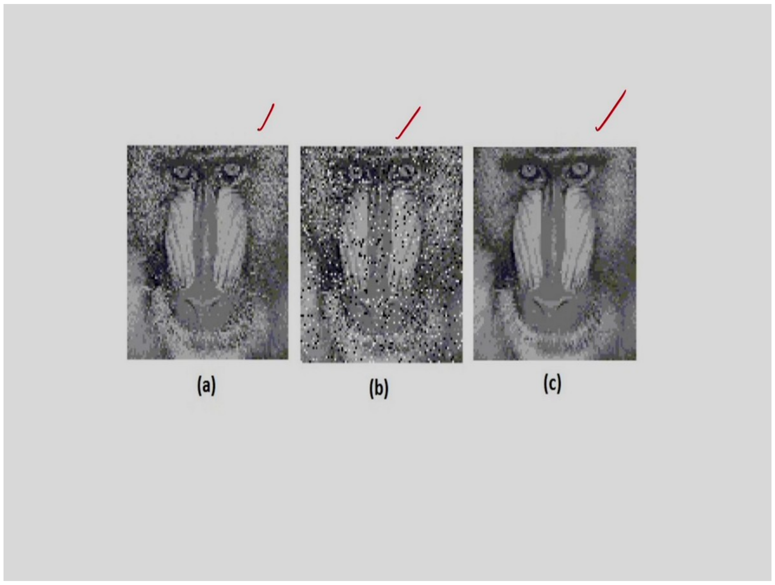
Median Filtering

So, here also I have shown one example. I have the original image, original image with noise and I am considering N is equal to 3 median filter, N is equal to 5 median filter I am considering. So, you can see the outputs. Then one problem of the median filter is the computational complexity. The computational complexity is in the order of n square. So, n means the small n means the number of pixels in the window.

So, the computational complexity is very high, because I have to find the median value. And in this case the computational complexity is in the order of n square, n is the number of pixels in the window. So, that means I have to do n square comparisons to determine the median value for the window.

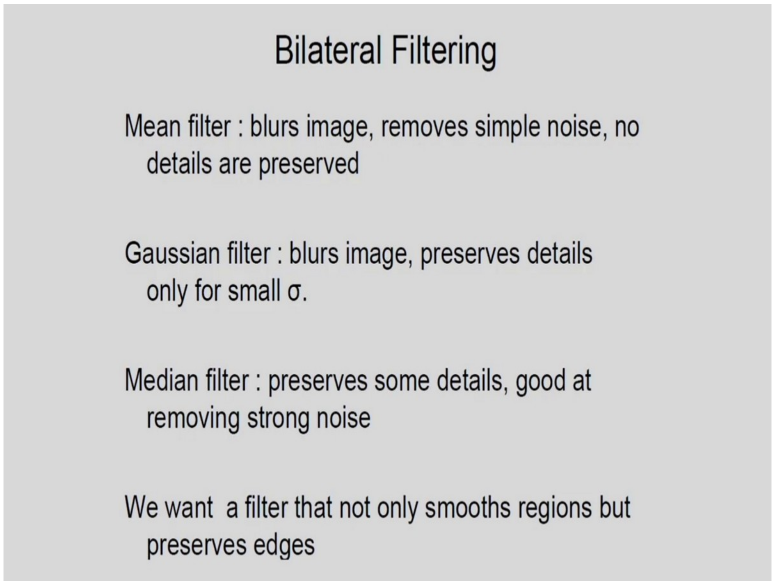
And suppose if I consider N by N image, then in this case, for the entire image if I want to determine the median value, then my computational complexity N square into n square. That is the computational complexity. Then in this case some algorithms like the quick sort algorithm I can apply, because I have to do the sorting so in this case I can apply some algorithm like quick sort algorithm I can apply to reduce the computational complexity.

(Refer Slide Time: 53:09)



And again, here I have shown the example of the median filter the first one is the input image, second one I am considering the salt and pepper noise. So all the pixels are now affected by the salt and pepper noise, and after this I am applying the median filter. So, this is the output of the median filter image.

(Refer Slide Time: 53:28)



Bilateral Filtering

Mean filter : blurs image, removes simple noise, no details are preserved

Gaussian filter : blurs image, preserves details only for small σ.

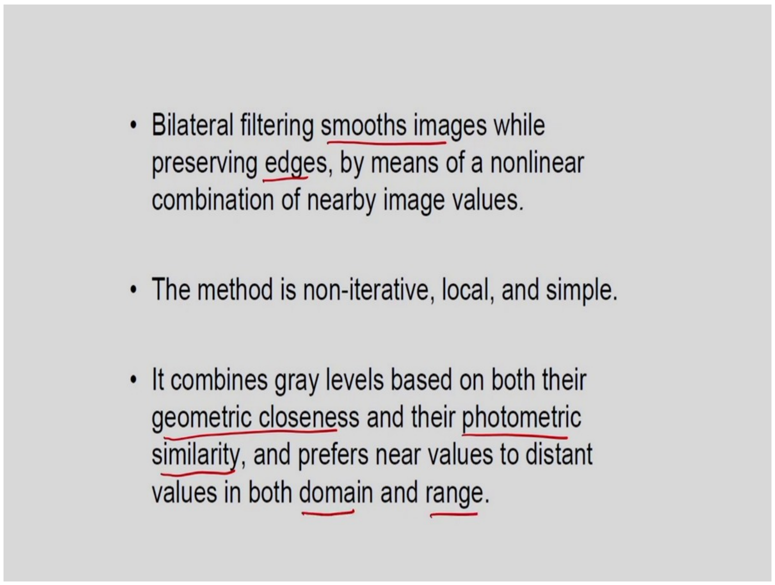
Median filter : preserves some details, good at removing strong noise

We want a filter that not only smooths regions but preserves edges

Finally, I want to discuss another filter that is the bilateral filter. So, what is the bilateral filter? So, before discussing the bilateral filter I want to discuss again the mean filter. What is the mean filter? Mean filters blurs the image, removes simple noise, and no details are preserved in case of the mean filter. In case of the Gaussian filter, Gaussian filter blurs the image, preserves details only for small value of sigma. Already I have explained about the Gaussian filter.

And after this I discussed about the median filter. Median filter preserves some details and it is very good for removing strong noises. Now in this case we want a filter that not only smooths the region but it can preserve edges. That means I need the edge information also I have to do the smoothing. So, for this I have to apply the bilateral filtering.

(Refer Slide Time: 54:33)

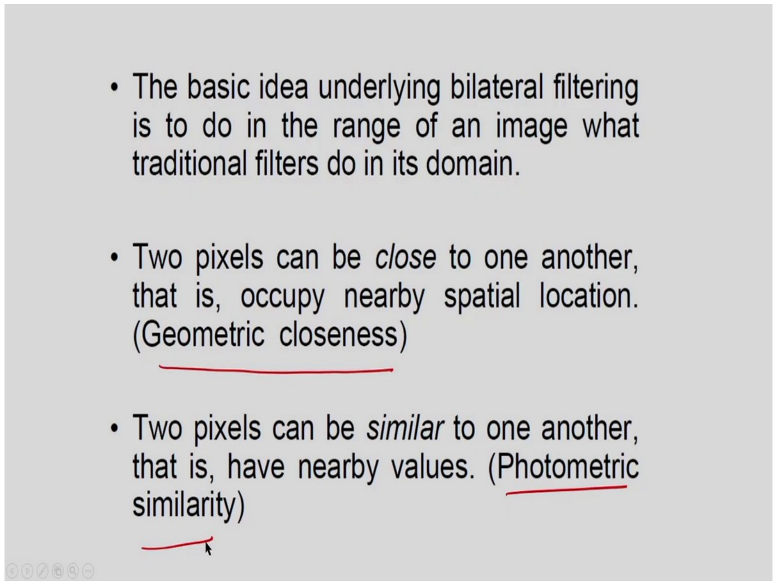


- Bilateral filtering smooths images while preserving edges, by means of a nonlinear combination of nearby image values.
- The method is non-iterative, local, and simple.
- It combines gray levels based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range.

So, bilateral filtering smooths images while preserving edges. That means I have to preserve the edges and also I have to smooth the images. So, these two considerations, one is I have to do the smoothing and also I have to preserve the edges. And in this case I have consider two cases, one is the geometric closeness I have to consider, that means the domain I have to see.

And also I have to consider a photometric similarity between the pixels, the neighborhood pixels. So, these two considerations I have to take into account. One is the geometric closeness between the neighborhood pixels and also I have to consider the photometric similarity between the neighborhood pixels. That means I am doing operations for the domain as well as the range. That means I am looking at the domain and the range of the image.

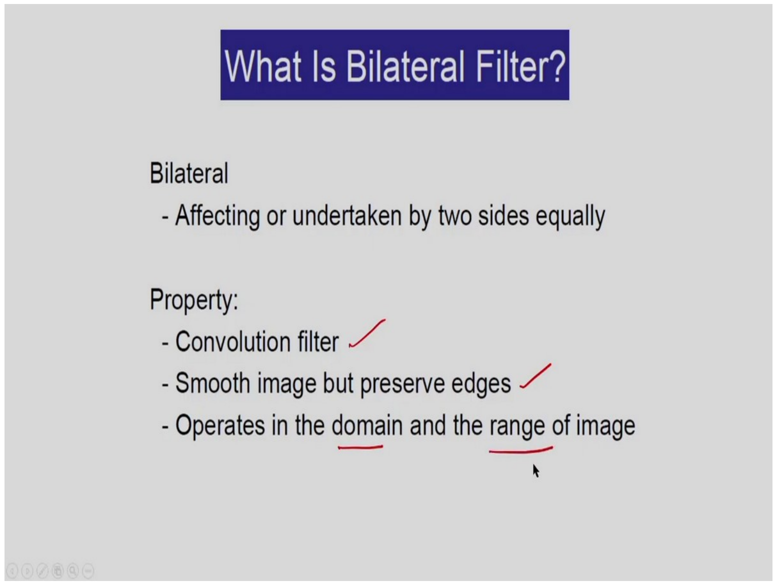
(Refer Slide Time: 55:33)



- The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain.
- Two pixels can be *close* to one another, that is, occupy nearby spatial location. (Geometric closeness)
- Two pixels can be *similar* to one another, that is, have nearby values. (Photometric similarity)

So, already I have explained. So what I have to do, for this I have to consider a geometric closeness I have to consider. The two pixels can be close to one another, that is occupy nearby spatial location. So that means I have to consider the domain and also I have to consider the photometric similarity. Two pixels can be similar to one another, that is they have the nearby values. So, that means I have to consider the range, the pixel range I have to consider.

(Refer Slide Time: 56:03)



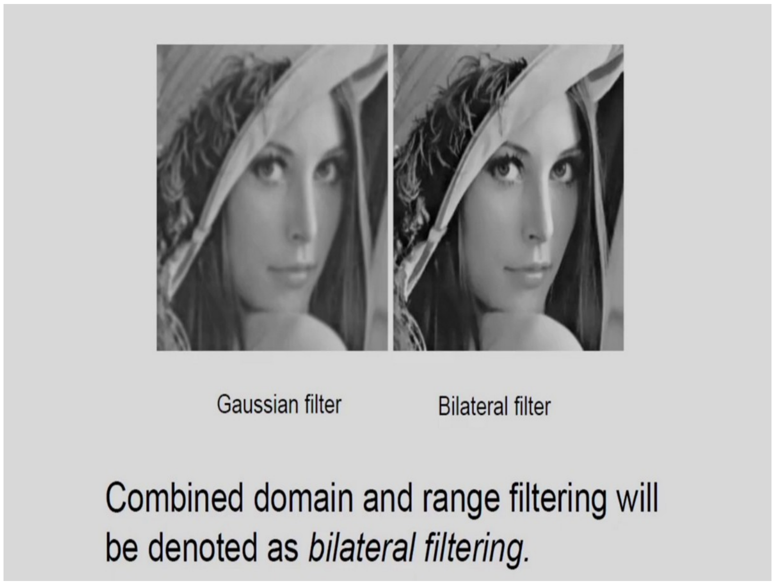
What Is Bilateral Filter?

Bilateral
- Affecting or undertaken by two sides equally

Property:
- Convolution filter ✓
- Smooth image but preserve edges ✓
- Operates in the domain and the range of image

And in this case what I have to consider? Property is nothing but the convolution filter I have to do, smooth images but preserve edges. So that means I am considering the edges and also I am doing the smoothing of the image. And in this case, I have to operate in the domain and the range. For range I have to consider the photometric similarity and for the domain I have to consider geometric similarity.

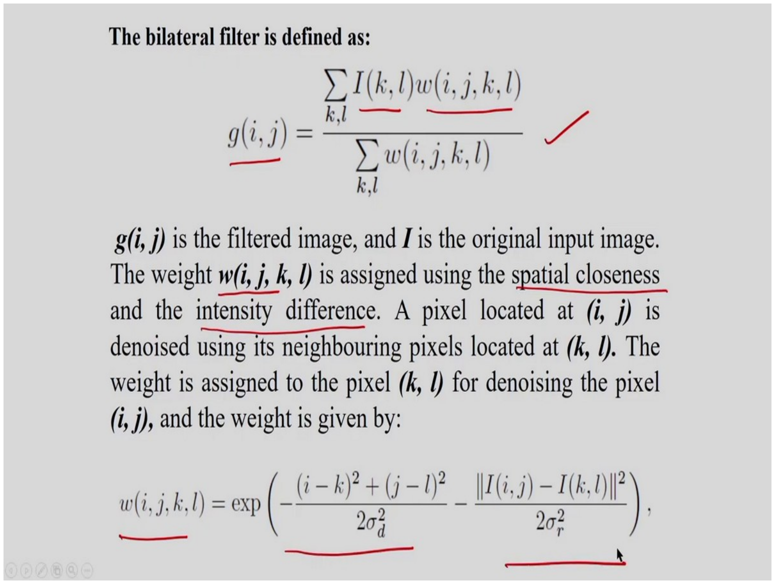
(Refer Slide Time: 56:32)



Combined domain and range filtering will be denoted as *bilateral filtering.*

So, you can see the output, one is the Gaussian filter output another one is the bilateral filter. In case of the bilateral filter you can see the edges, I am having the edges and also the image is also smoothed. So, combined domain and range filtering I have to do in the bilateral filtering.

(Refer Slide Time: 56:53)



The bilateral filter is defined as:

$$\underline{g(i,j)} = \frac{\sum_{k,l} \underline{I(k,l)}\,\underline{w(i,j,k,l)}}{\sum_{k,l} w(i,j,k,l)} \quad ✓$$

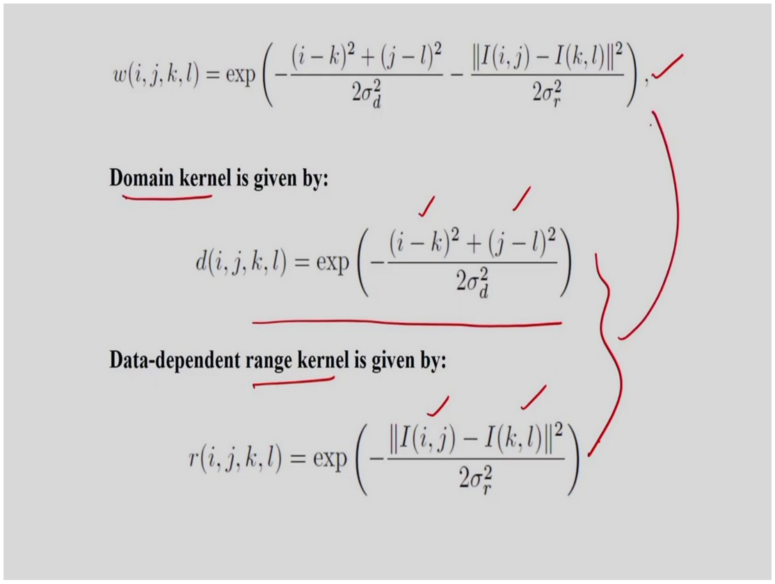
g(i, j) is the filtered image, and ***I*** is the original input image. The weight ***w(i, j, k, l)*** is assigned using the spatial closeness and the intensity difference. A pixel located at ***(i, j)*** is denoised using its neighbouring pixels located at ***(k, l).*** The weight is assigned to the pixel ***(k, l)*** for denoising the pixel ***(i, j),*** and the weight is given by:

$$\underline{w(i,j,k,l)} = \exp\left(-\underline{\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}} - \underline{\frac{\|I(i,j)-I(k,l)\|^2}{2\sigma_r^2}}\right),$$

And in this case the bilateral filtering is defined like this, g i j. And in this case I am considering the input image, the original image is I, and also I am considering the weight. The weight is w i, j, k, l. And g, i, j is the filtered image. The weight w, i, j, k, l, that is the weight I am considering is assigned using the spatial closeness that is the geometric closeness. And the intensity difference that is the photometric similarity. One is the geometric closeness this is the spatial closeness, another one is the intensity difference that is the photometric similarity I have to consider.

A pixel located at i, j is denoised using its neighboring pixels located at k, l. And in this case the weight is defined like this. You can see we have two parts, the first part is this, the second part is this. So, what is meaning of the first part and what is the meaning of the second part? I can show you in the next slide.

(Refer Slide Time: 57:57)



$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|I(i,j)-I(k,l)\|^2}{2\sigma_r^2}\right),$$

Domain kernel is given by:

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}\right)$$

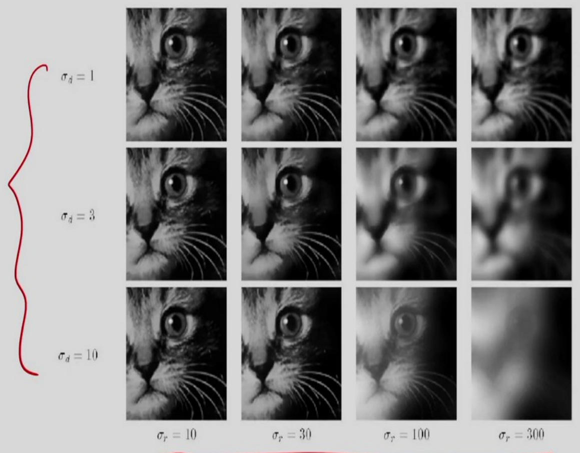
Data-dependent range kernel is given by:

$$r(i,j,k,l) = \exp\left(-\frac{\|I(i,j)-I(k,l)\|^2}{2\sigma_r^2}\right)$$

So, this is the weight. The first part is the domain kernel, you can see because I have to consider geometric closeness I have to consider. The second part is the range kernel I have to consider the range kernel. The first one is the domain kernel. For the range kernel I have to consider the photometric similarity.

So here you can see I am finding the photometric similarity between the pixels and also I am considering the geometric similarity between the pixels. So, you can see the geometric similarity between the pixels. So, that means in bilateral filtering I am considering the domain kernel and also the range kernel I am considering. And that is combined and if I combine these two I will be getting the weight.

(Refer Slide Time: 58:45)



Bilateral Filters with various range and domain parameter values

R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 2008.

And in this case you can see the bilateral filters with various range of domain parameters value. So, I am considering the sigma d, that is the domain parameters value and also I am considering sigma r, that is the range parameters value. For defining parameters you can see the output images. That means I am considering both range and the domain filtering in the bilateral filters.

So, in this class today I have discussed about the spatial filtering. First I discussed about the concept of the masking. After this I discussed about the concept of the low pass filter and the high pass filter, and the high boost filter. After this one non-linear filter I discussed, that is the median filter I have discussed. And finally, I discussed the concept of the bilateral filtering.

So, in the bilateral filtering we considered the domain filtering and the range filtering. So, this is about to spatial filtering. In my next class I will discuss about the concept of frequency domain filtering. In a frequency domain filtering I have to modify the Fourier Transform of the image. So, that concept I am going to discuss in the next class. So, let me stop here today. Thank you.