Digital Image Processing

Digital Image Fundamentals – II

12th June, 2017

Image Enhancement

- Process an image to make the result more suitable than the original image for a specific application
- The reasons for doing this include:
  - ✓ Highlighting interesting details in the image
  - ✓ Removing noise from images
  - ✓ Making images visually more appealing
- *Image enhancement is subjective (problem /application oriented)*

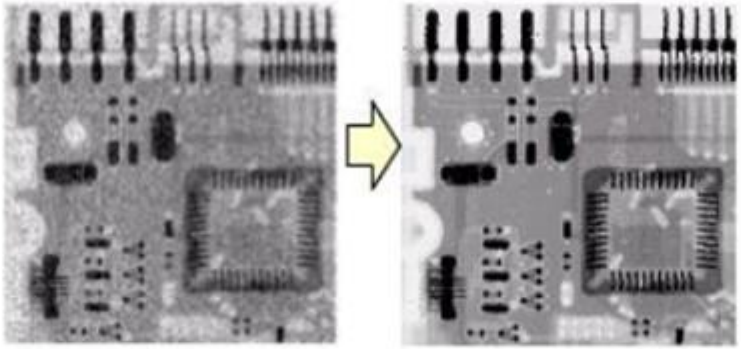
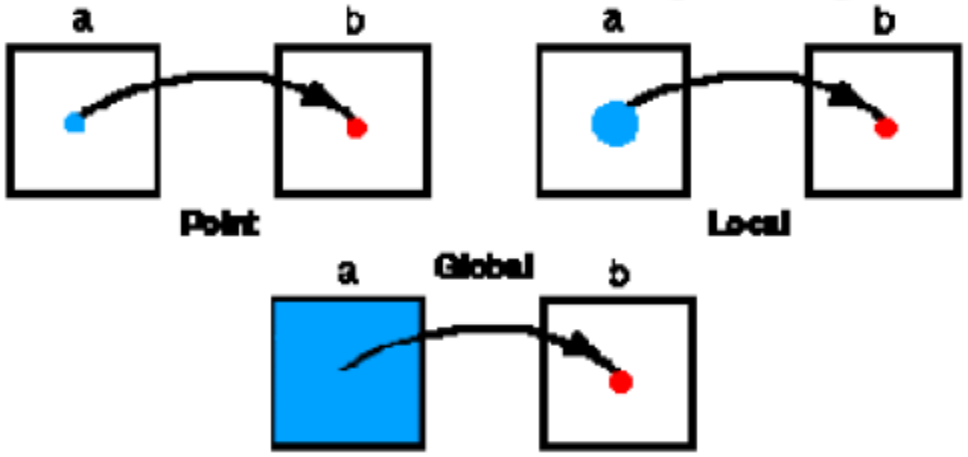


Image Enhancement

- There are two broad categories of Image enhancement techniques:
  - ✓ Spatial domain: Direct manipulation of pixel in an image (on the image plane)
  - ✓ Frequency domain: Processing the image based on modifying the Fourier transform of an image
- Many techniques are based on various combinations of methods from these two categories

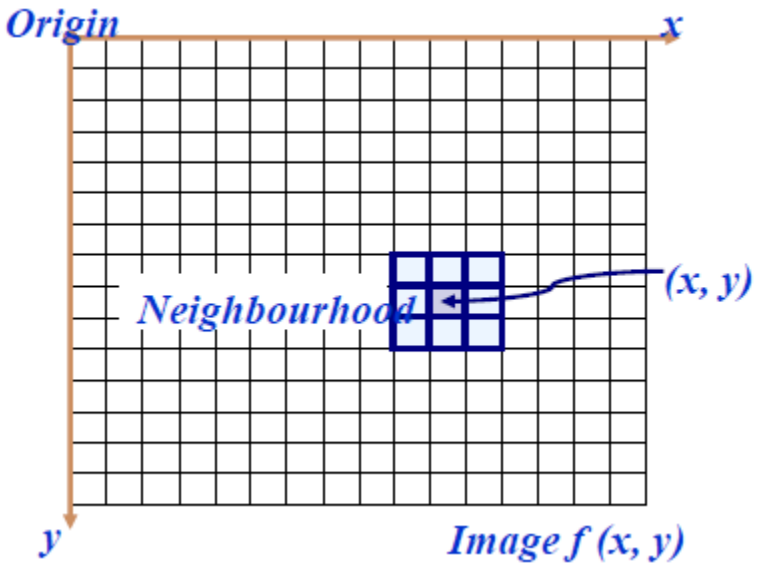
Types of Image Enhancement Operations

Point/pixel operations	Output value at specific coordinates (x,y) is dependent only on the input value at (x,y)
Local operations	The output value at (x,y) is dependent on the input values in the *neighborhood* of (x,y)
Global operations	The output value at (x,y) is dependent on all the values in the input image



Neighborhood Operations on Images

- Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations
- Neighbourhoods are mostly selected a square regions around a central pixel
- Any size and any shape filter are possible



Spatial Filtering

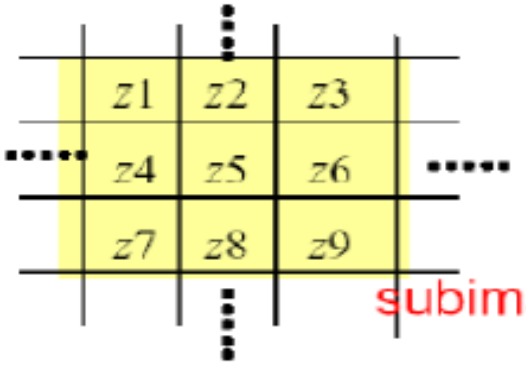
- Filtering is a *neighborhood operation,* in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel.

Local Operations through Spatial Filtering

- The output intensity value at (x,y) depends not only on the input intensity value at (x,y) but also on the specified number of neighboring intensity values around (x,y)
- Spatial masks (also called window, filter, kernel, template) are used and convolved over the entire image for local enhancement (spatial filtering)
- The size of the mask determines the number of neighboring pixels which influence the output value at (x,y)
- The values (coefficients) of the mask determine the nature and properties of enhancing technique

Basics of Spatial Filtering

- Given the 3×3 mask with coefficients: $w_1, w_2, \ldots, w_9$
- The mask covers the pixels with gray levels: $z_1, z_2, \ldots, z_9$



subimage

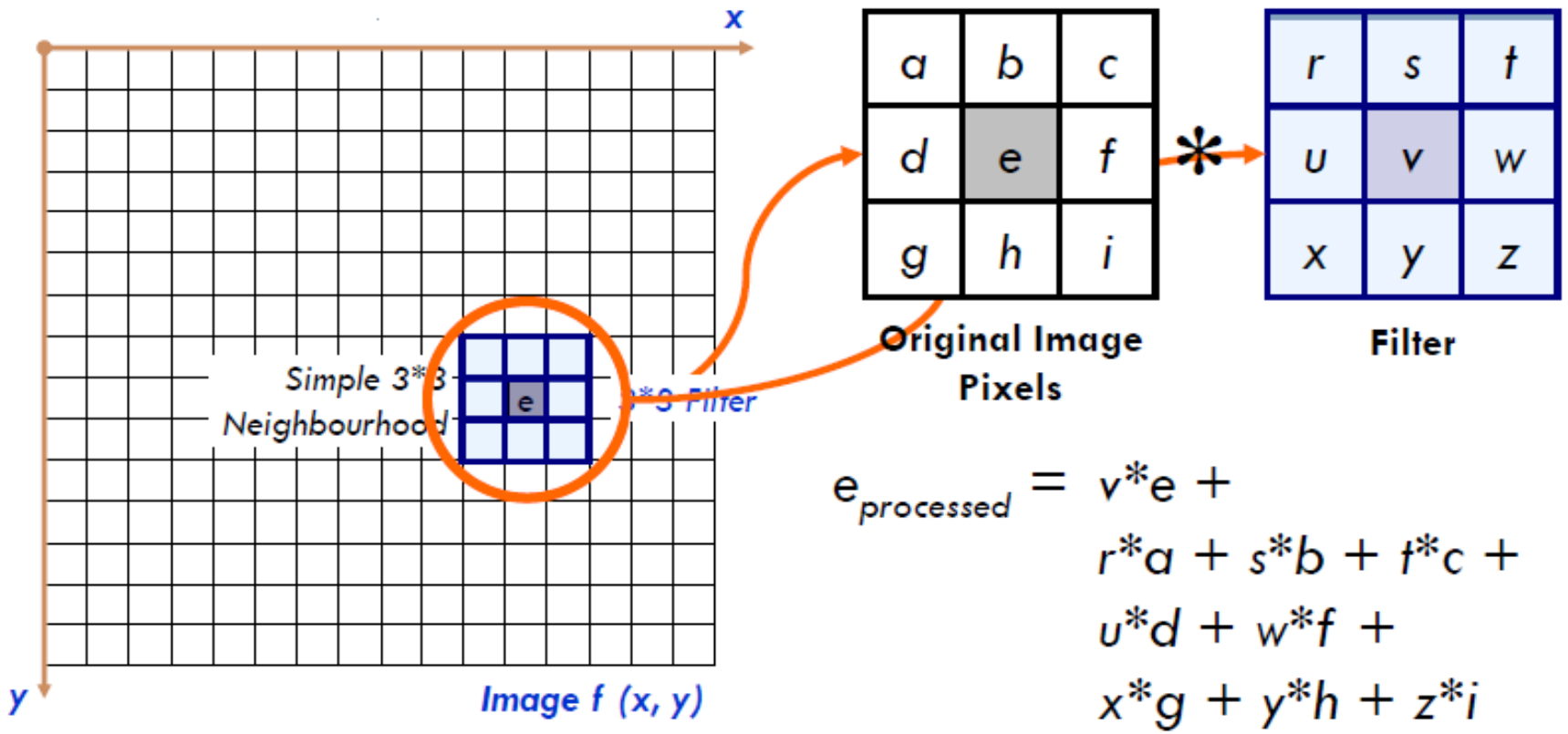
w1	w2	w3
w4	w5	w6
w7	w8	w9

Mask coefficients

$$z_5 \leftarrow z = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

- z gives the output intensity value for the processed image (to be stored in a new array) at the location of z_5 in the input image

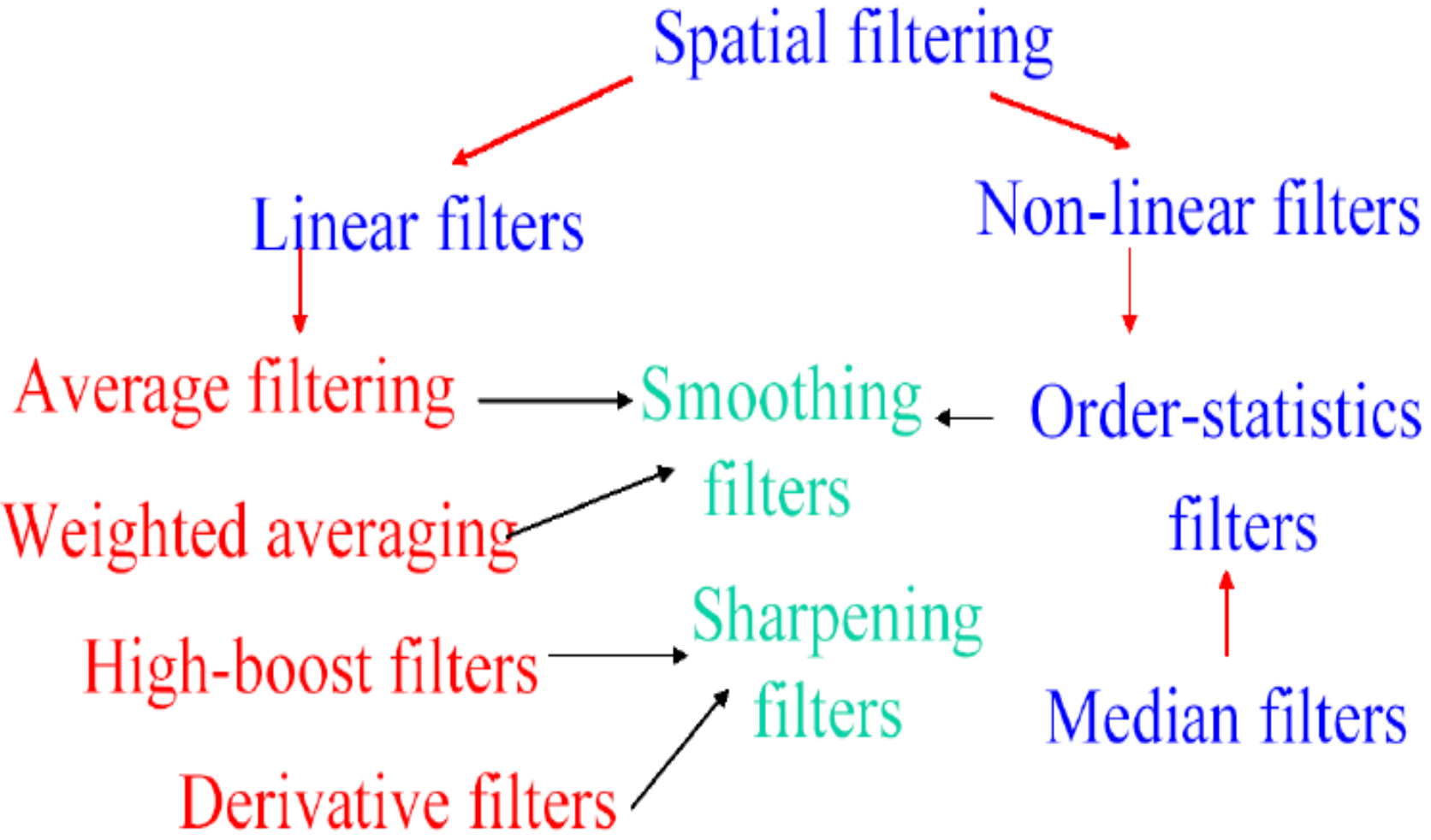
Local Operations through Spatial Filtering



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

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

Types of Spatial Filtering



Spatial Filters for Smoothing

- Used for: Noise Reduction
- Side Effects: Edge Blurring

Linear Filtering

- *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

- In mathematics, a **linear combination** is an expression constructed from a set of terms by multiplying each term by a constant and adding the results (e.g. a **linear combination** of x and y would be any expression of the form ax + by, where a and b are constants).

Convolution

- Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

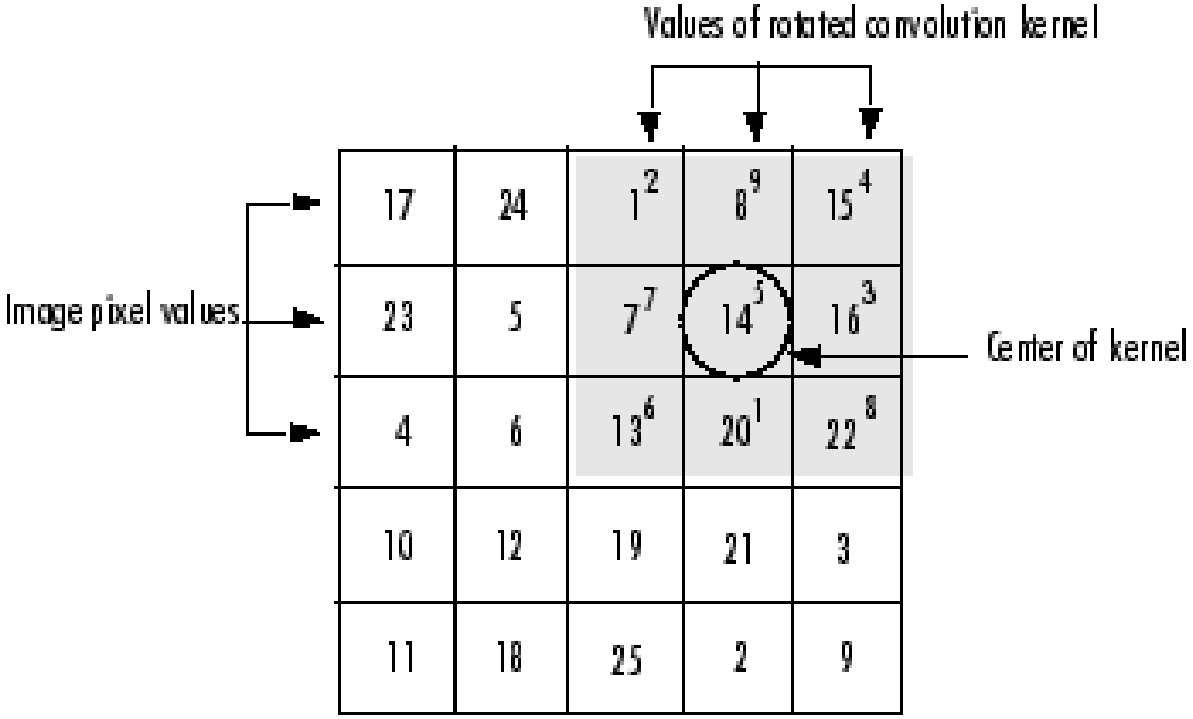
- For example, suppose the image is
- A = [17 24 1 8 15; 23 5 7 14 16; 4 6 13 20 22; 10 12 19 21 3; 11 18 25 2 9]
- and the correlation kernel is
- h = [8 1 6; 3 5 7; 4 9 2]

Convolution

- Use the following steps to compute the output pixel at position (2,4):
  - Rotate the correlation kernel 180 degrees about its center element to create a convolution kernel.
  - Slide the center element of the convolution kernel so that it lies on top of the (2,4) element of A.
  - Multiply each weight in the rotated convolution kernel by the pixel of A underneath.
  - Sum the individual products from step 3.

Convolution

- **Computing the (2,4) Output of Convolution**
- The (2,4) output pixel from the convolution is
- 1.2 + 8.9 + 15.4 + 7.7 + 14.5 + 16.3 + 13.6 + 20.1 + 22.8 = 575



Correlation

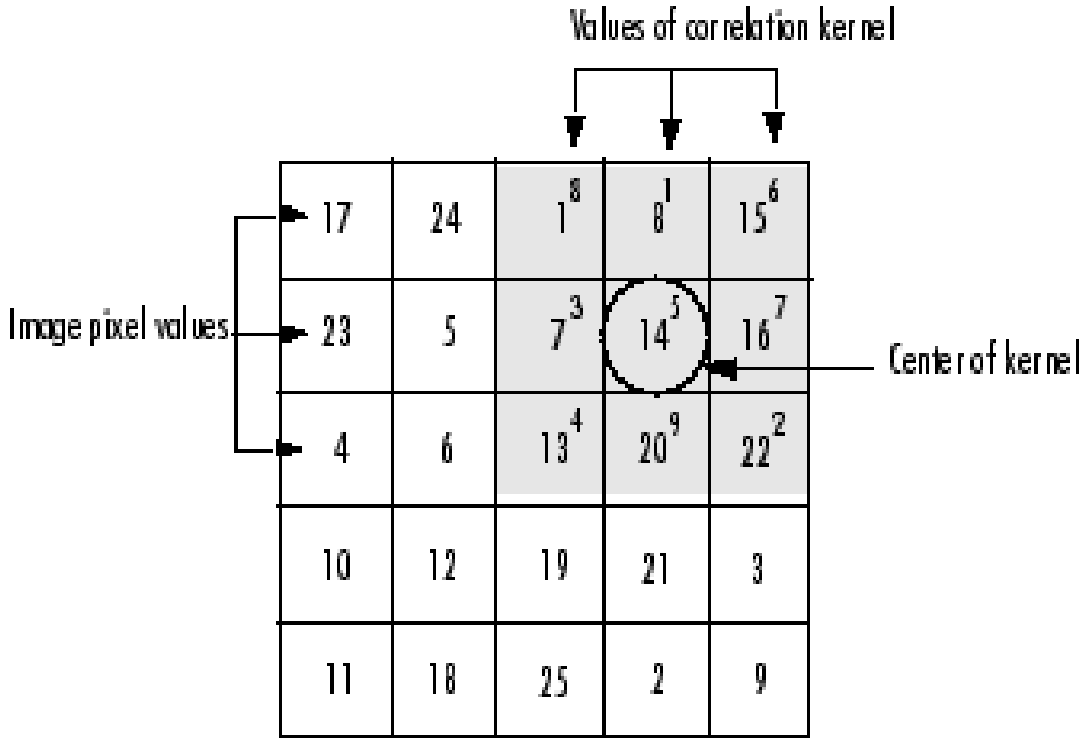
- The operation called *correlation* is closely related to convolution. In correlation, the value of an output pixel is also computed as a weighted sum of neighboring pixels. The difference is that the matrix of weights, in this case called the *correlation kernel*, is not rotated during the computation.

- To compute the (2,4) output pixel of the correlation of A, assuming h is a correlation kernel instead of a convolution kernel, using these steps:

  - Slide the center element of the correlation kernel so that lies on top of the (2,4) element of A.
  - Multiply each weight in the correlation kernel by the pixel of A underneath.
  - Sum the individual products.

Correlation

- **Computing the (2,4) Output of Correlation**
- The (2,4) output pixel from the correlation is
- 1.8 + 8.1 + 15.6 + 7.3 + 14.5 + 16.7 + 13.4 + 13.4 + 20.9 + 22.2 = 585



Smoothing Spatial Filters

- One of the simplest spatial filtering operations we can perform is a smoothing operation
- Simply average all of the pixels in a neighbourhood around a central value
- Especially useful in removing noise from images

Examples:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

3 x 3 Averaging Mask

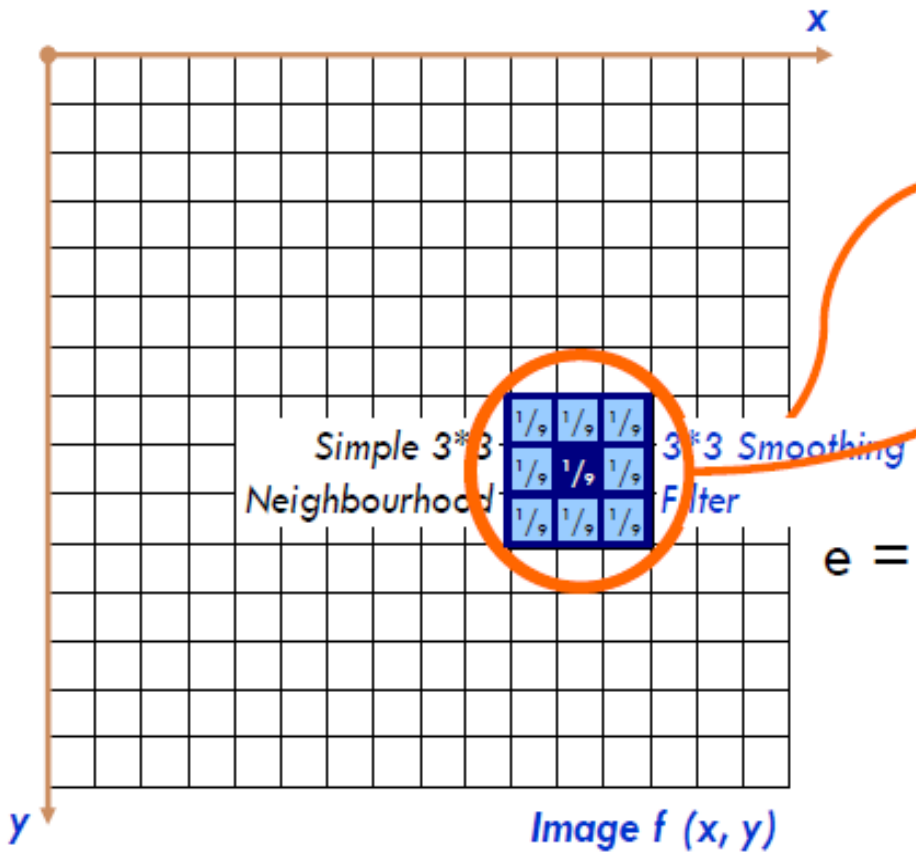
$$\frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

5 x 5 Averaging Mask

$$\frac{1}{81}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

9 x 9 Averaging Mask

Smoothing Spatial Filters



104	100	108
99	106	98
95	90	85

*

Original Image Pixels

$$e = \tfrac{1}{9}*106 + \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85$$

$$= 98.3333$$

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

Smoothing Spatial Filters

- Let's study the example in detail

$$e = {}^{1}/_{9}*106 + {}^{1}/_{9}*104 + {}^{1}/_{9}*100 + {}^{1}/_{9}*108 + {}^{1}/_{9}*99 + {}^{1}/_{9}*98 + {}^{1}/_{9}*95 + {}^{1}/_{9}*90 + {}^{1}/_{9}*85$$

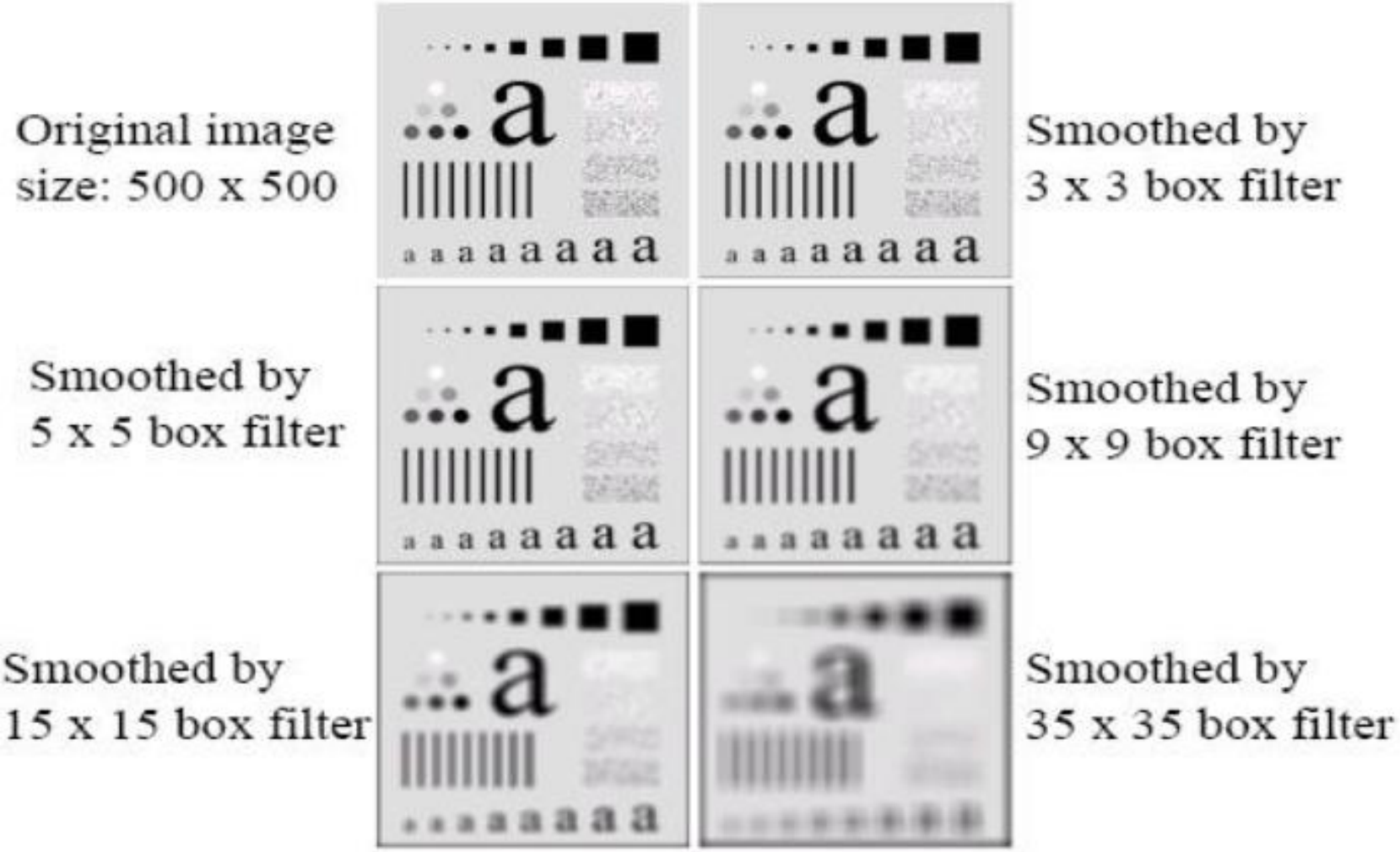
e = 1/9 * (106 + 104 + 100 + 108 + 99 + 98 + 95 + 90 + 85)
e = 1/9 * 885
e = 885/9
e = 98.3333

Isn't it exactly the same thing as averaging?
You are essentially adding up all the pixels in the neighborhood and dividing them by the total number of pixels
Note: If we get a decimal number, we round to be sure that we have an integer number

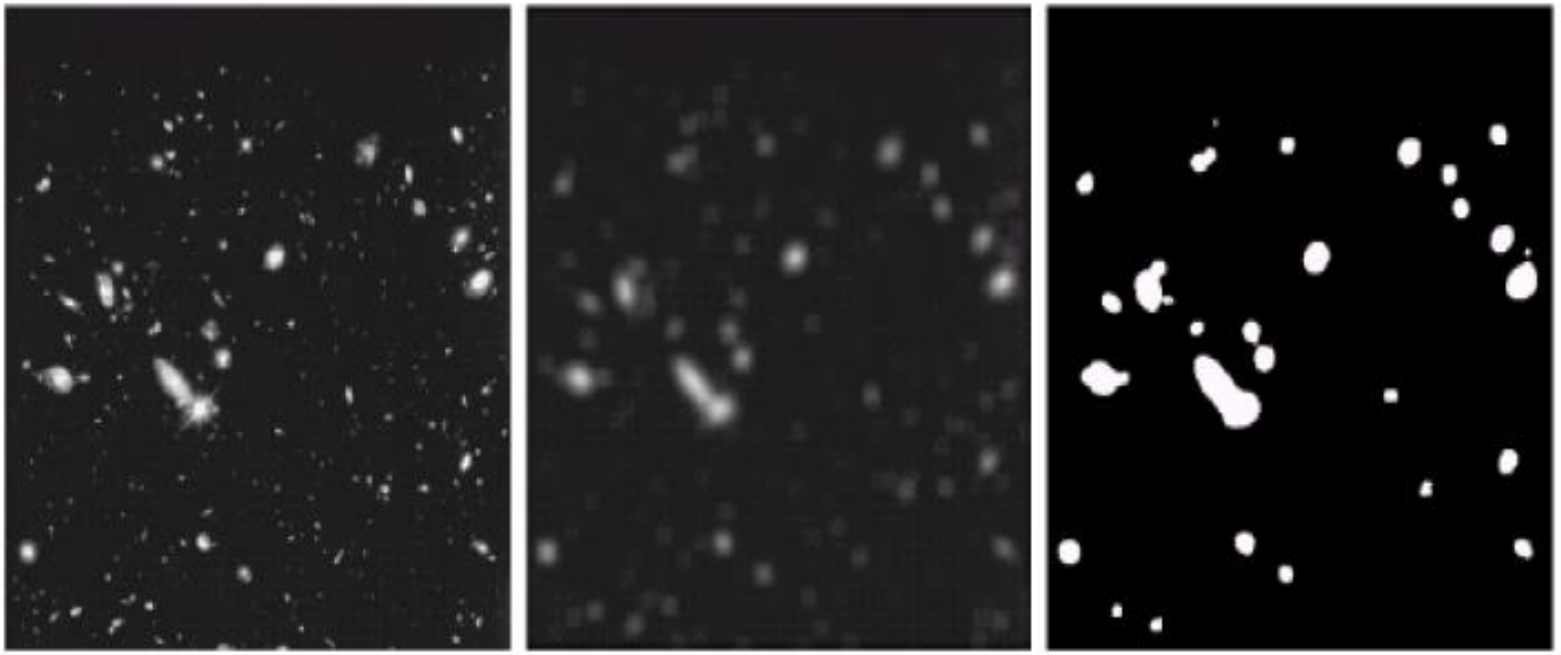
Smoothing Spatial Filters – Matlab Commands

- Processing (Reducing Noise) from Images in Matlab
- First, we need to create the mask
- We do this by calling `fspecial ()`
- `mask = fspecial ( 'average', N);`
- mask contains the averaging mask to use
- First parameter specifies we want an averaging mask
- N specifies the size of the mask, bigger the N the more is blurriness effect
- Next call a command `imfilter ()`, this command will perform multiplication and addition for each pixel in the image
- `out = imfilter (im, mask)` where, out is the output image, im is the input image which we want to blur, mask in this case is averaging
- `imfilter ()` works on both grayscale and color images.

Spatial Filtering for Smoothing: Example



Spatial Filtering for Smoothing: Example



a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Weighted Smoothing Spatial Filters

- More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function
  - ✓ Pixels closer to the central pixel are more important
  - ✓ Often referred to as *a weighted averaging*

$1/16$	$2/16$	$1/16$
$2/16$	$4/16$	$2/16$
$1/16$	$2/16$	$1/16$

Weighted Averaging Filter

Non-Linear Spatial Filters

Nonlinear spatial filters

Output is based on order of gray levels in the masked area (sub-image)

Examples: Median filtering, Max & Min filtering

- ✓ **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**

Median Filter

- Median filtering

  Assigns the mid value of all the gray levels in the mask to the center of mask;

  Particularly effective when

  - ✓ *the noise pattern consists of strong, spiky components (salt-and-pepper)*
  - ✓ *edges are to be preserved*

10	20	20
20	15	20
20	25	100

Output = ? **20**

Median Filter

- For example if you have following 1D image array:
- 3,9,4,52,3,8,6,2,2,9
  - Median (3,4,9) = 4
- Keep the window/kernel/filter size 3

Median Filter

- For each pixel *(r,c)* in the image, extract an M x N subset of pixels centered at *(r,c)*

- Sort these pixels in ascending order, and grab the *median* value The output image at *(r,c)* is this value

- How do we perform median filtering in MATLAB?

  out = medfilt2(im, [M N]);

References

- DIP by Gonzalez
- For Convolution and Correlation matrix problems follow: https://www.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html