
**National University of Sciences and Technology (NUST)**
**SEECS**

# Digital Image Processing



Khawar Khurshid

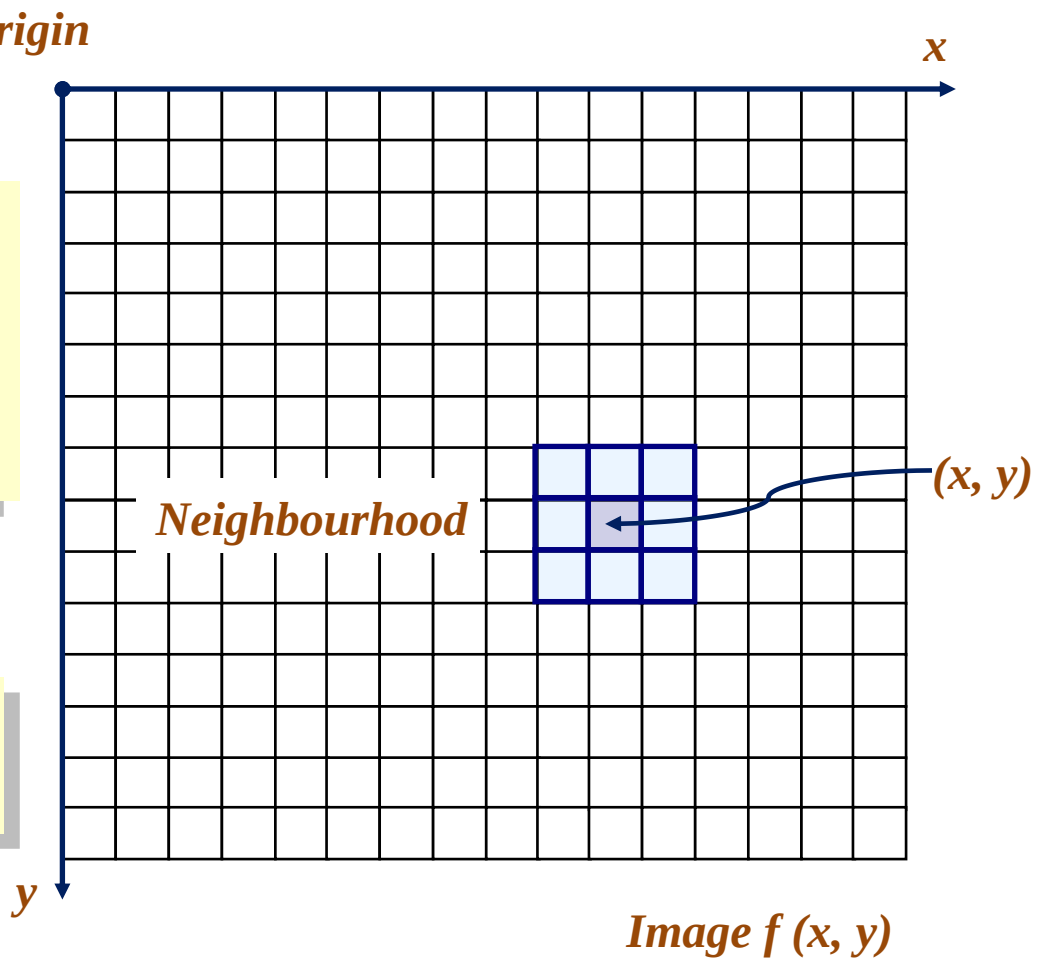
# Spatial Filtering

# Spatial Filtering - Basics

## Neighbourhood operations:

Operate on a larger neighbourhood of pixels than point operations

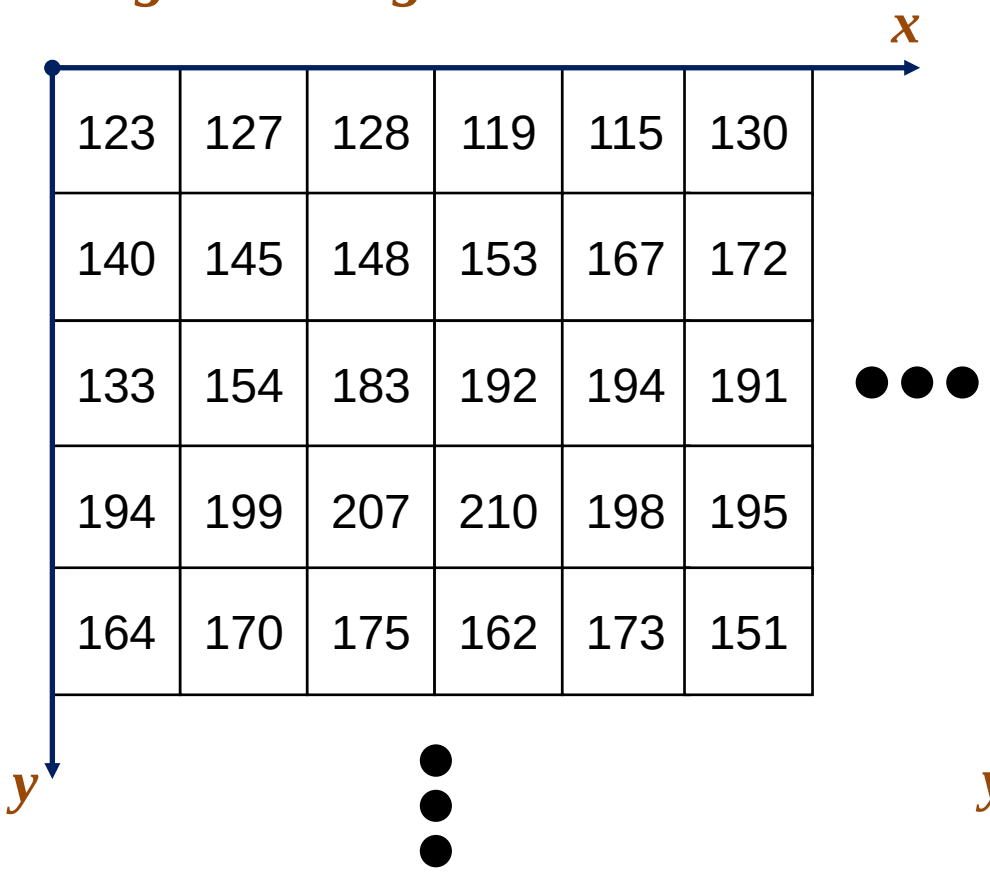
Neighbourhoods are mostly a rectangle around a central pixel



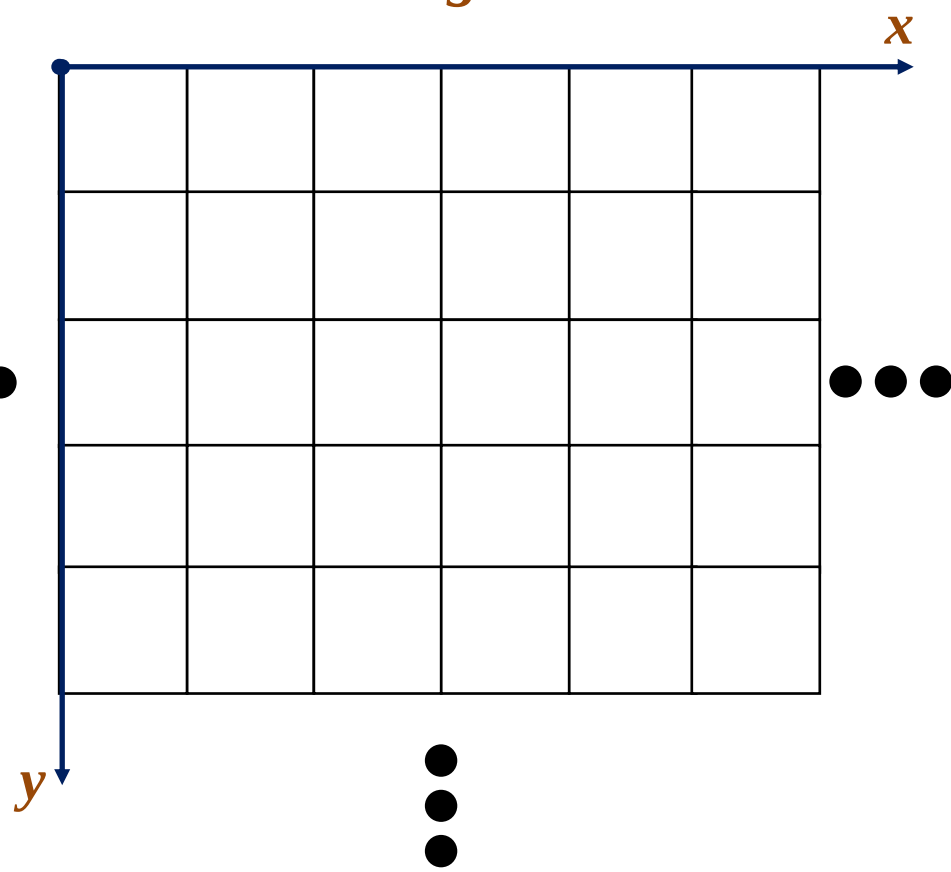
# Spatial Filtering - Basics

NUST

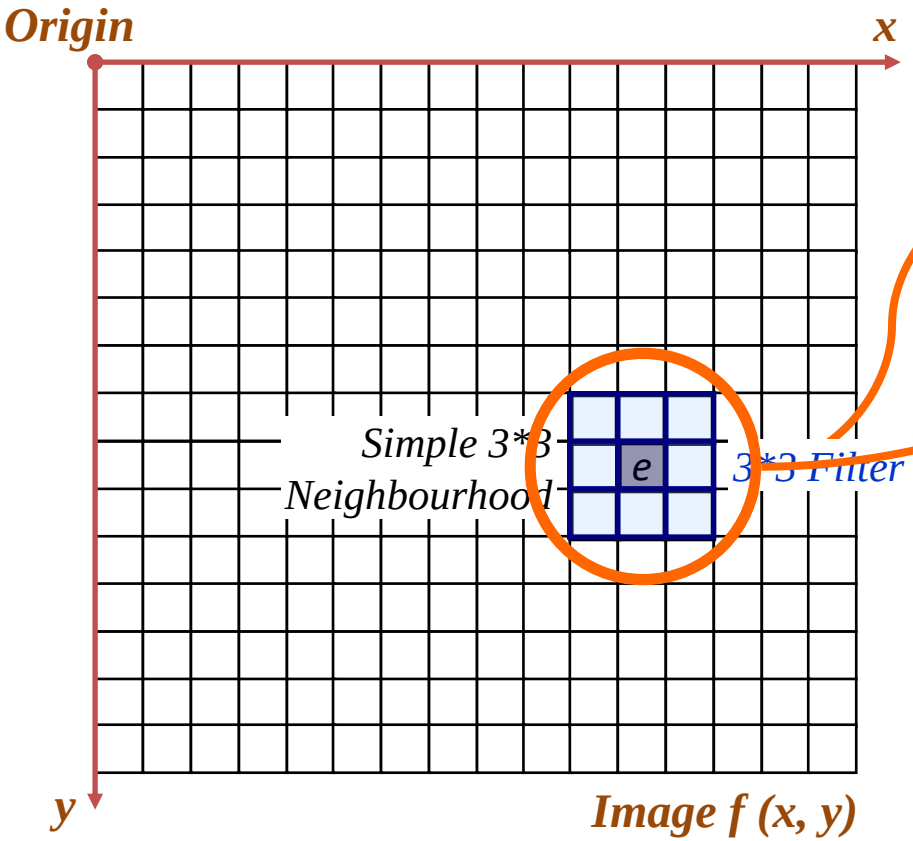
## *Original Image*




## *Enhanced Image*



# Spatial Filtering - Basics




| | | |
|---|---|---|
| a | b | c |
| d | e | f |
| g | h | i |

Original Image Pixels

*

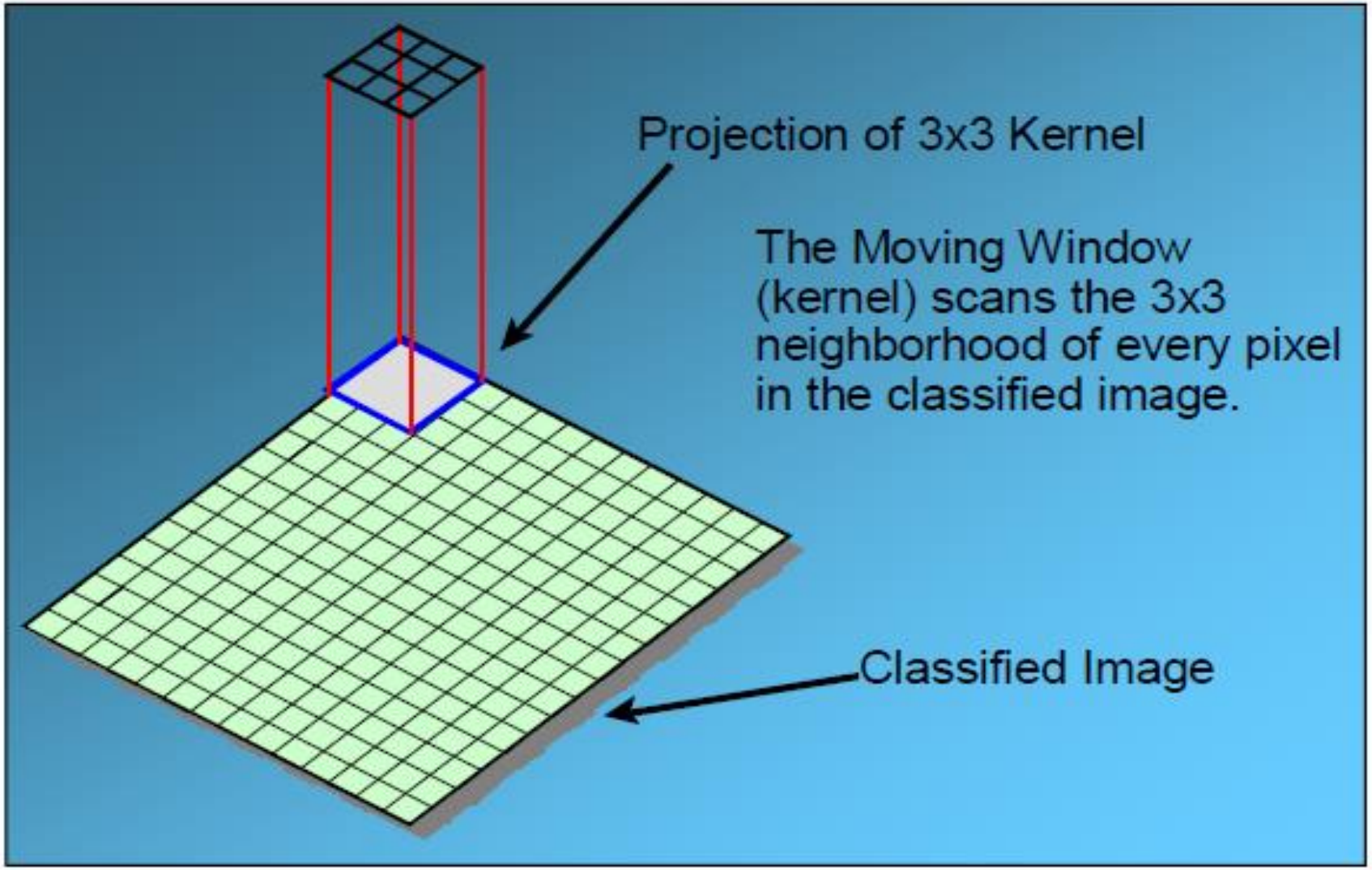
| | | |
|---|---|---|
| r | s | t |
| u | v | w |
| x | y | z |

Filter

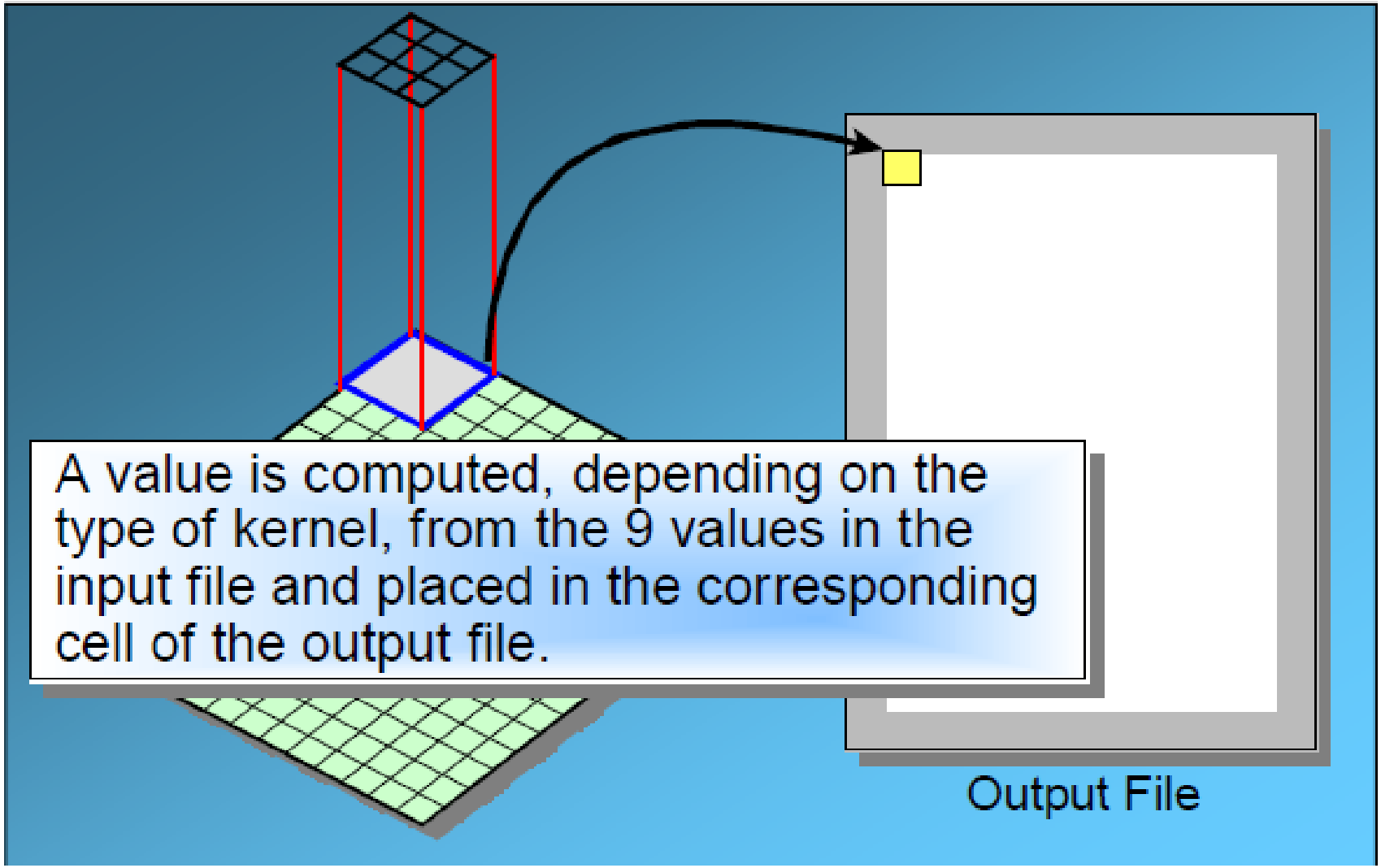
$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

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

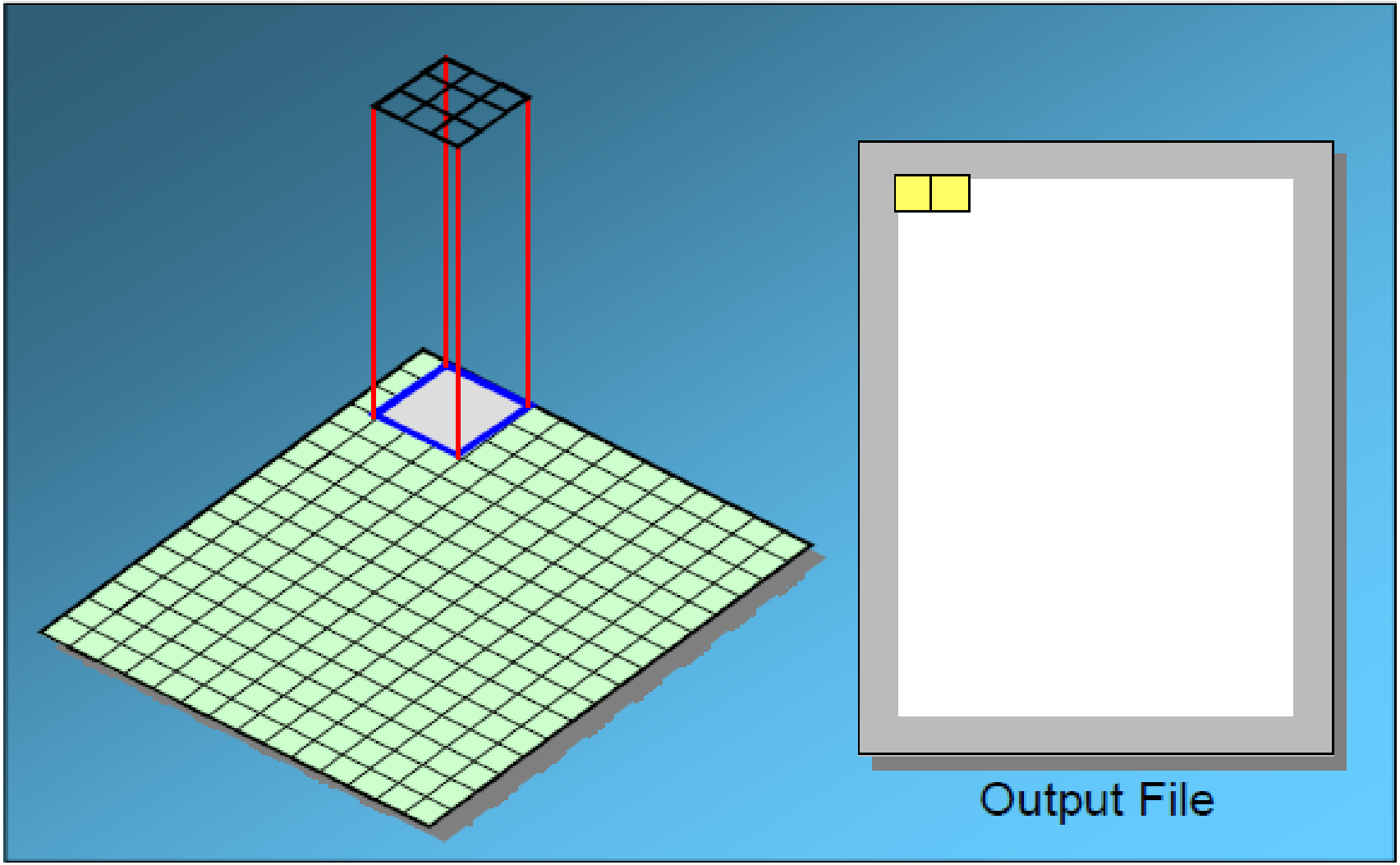
# Spatial Filtering - Basics



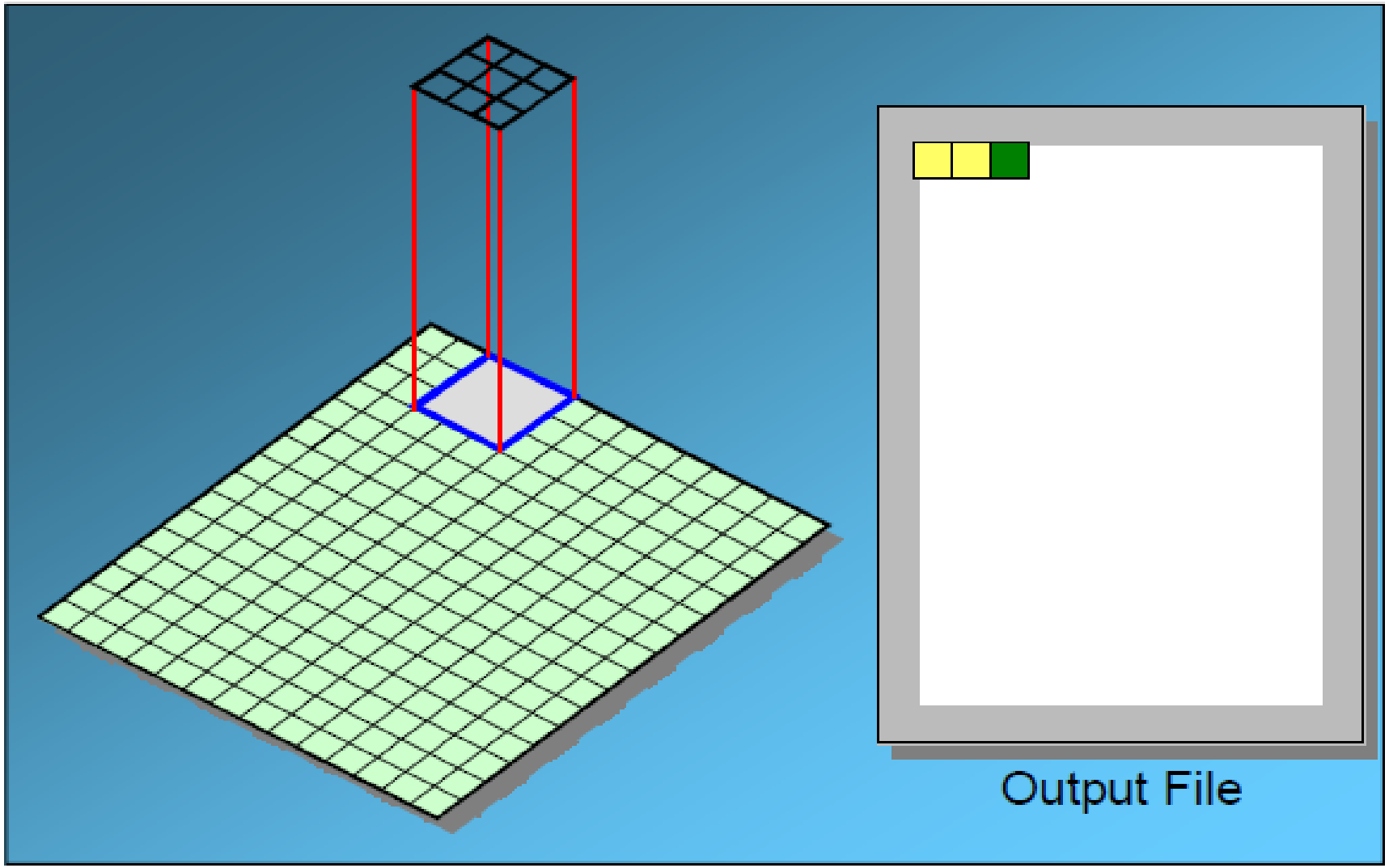
# Spatial Filtering - Basics



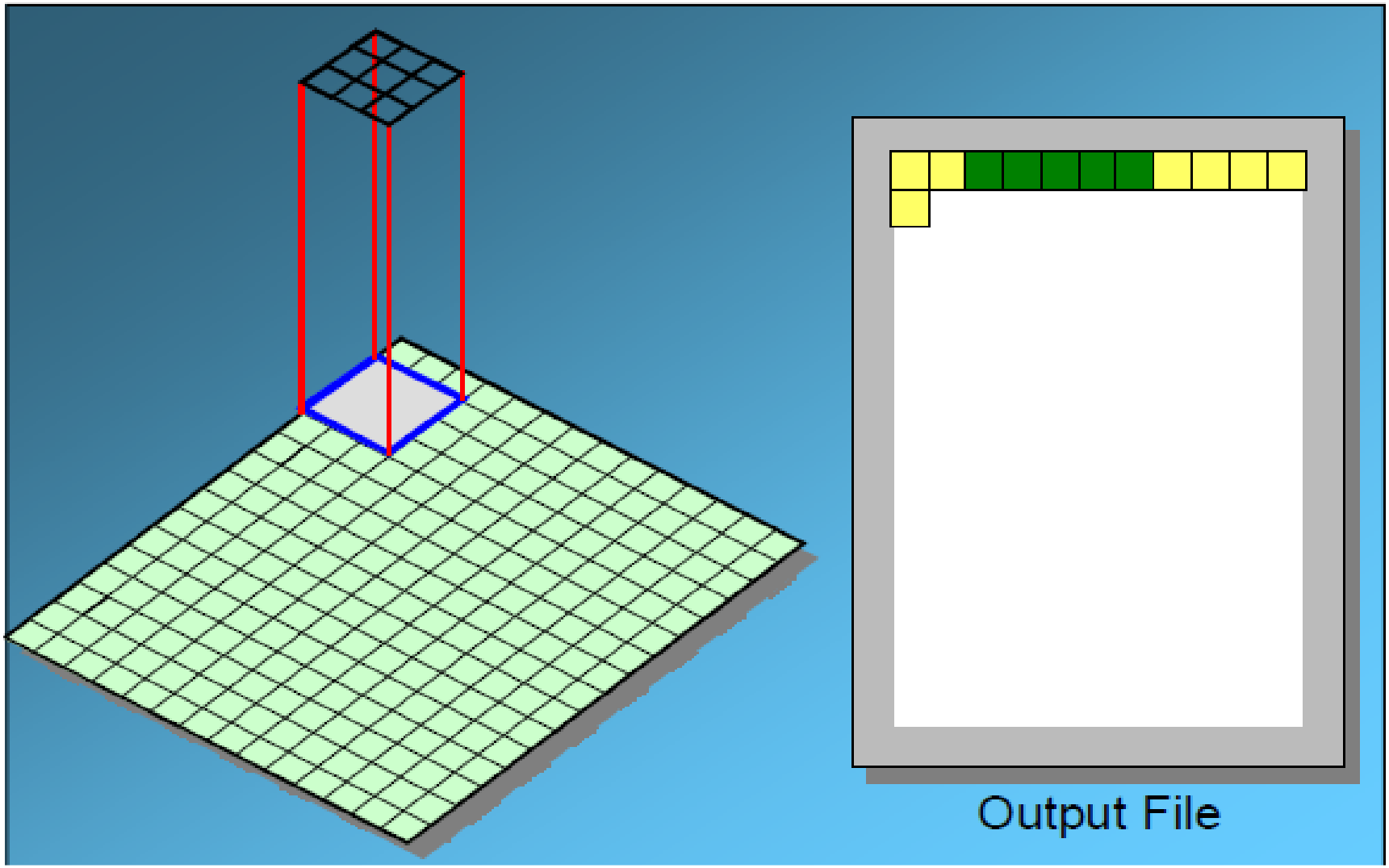
# Spatial Filtering - Basics



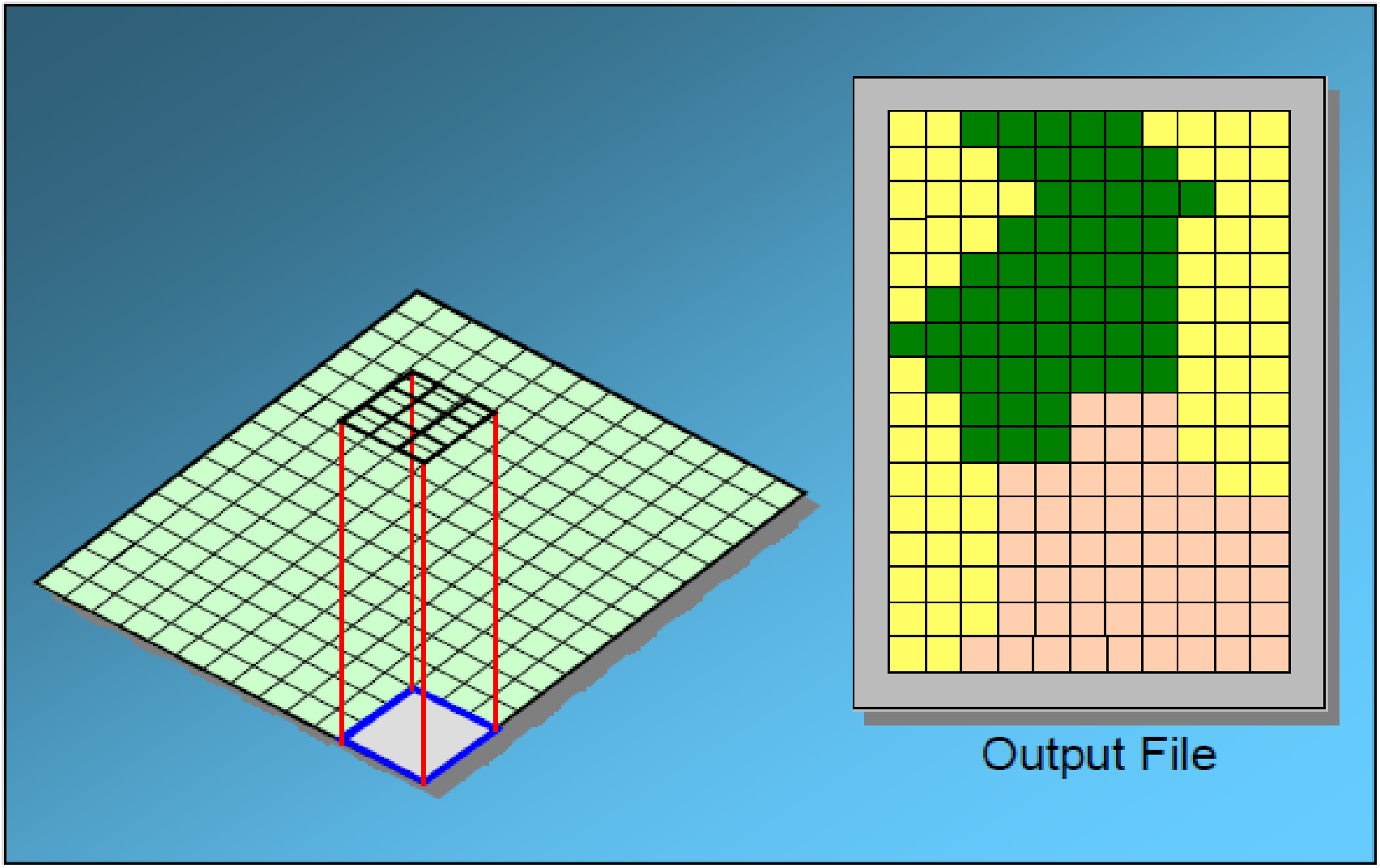
# Spatial Filtering - Basics



# Spatial Filtering - Basics



# Spatial Filtering - Basics



# Spatial Filtering - Basics

NUST

## Mask operation near the image border

Problem arises when part of the mask is located outside the image plane

Discard the problem pixels (e.g. 512x512 input 510x510 output if mask size is 3x3)

Zero padding: Expand the input image by padding zeros (512x512 original image, 514x514 padded image, 512x512 output)

*Zero padding is not recommended as it creates artificial lines or edges on the border*

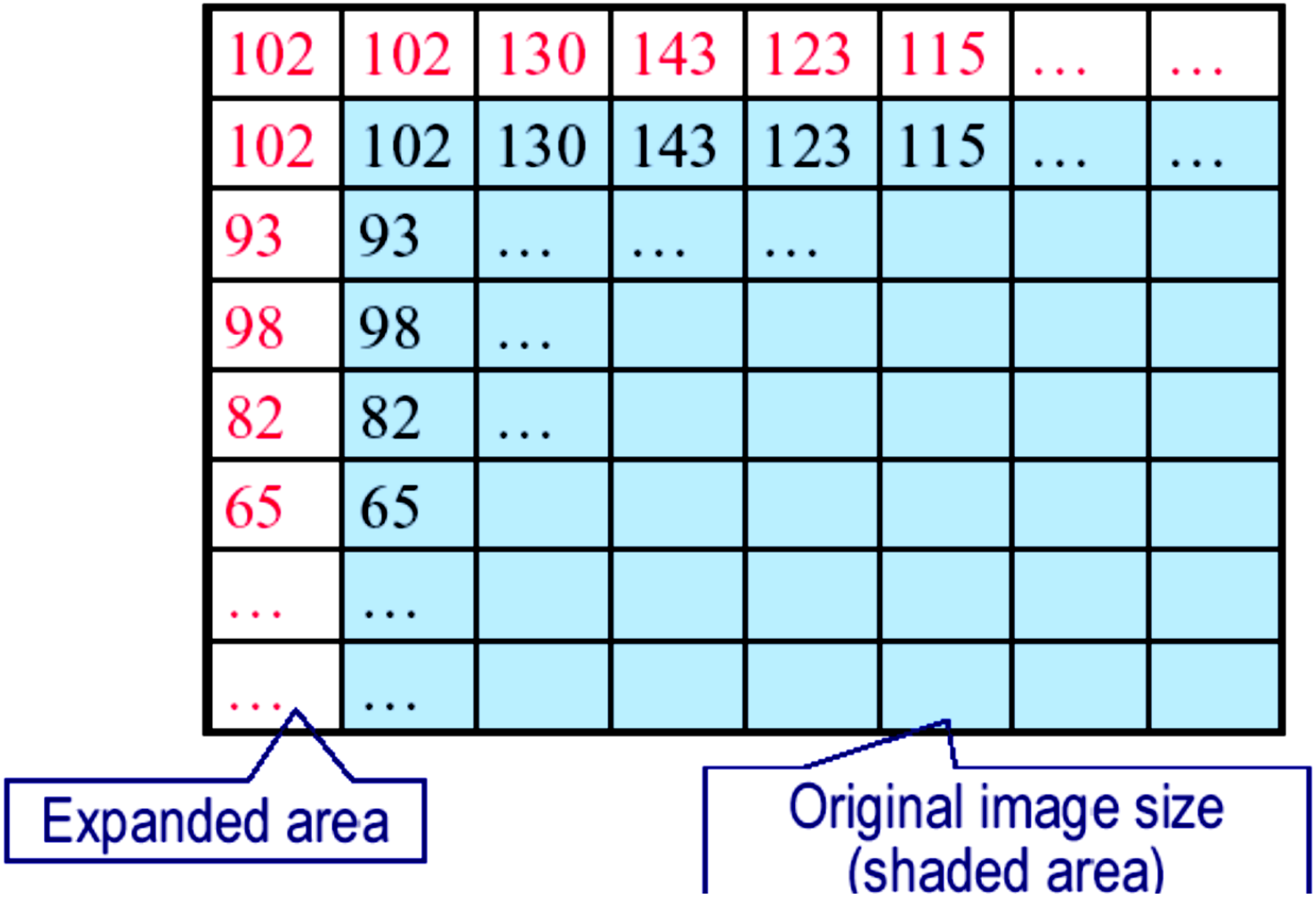
Pixel replication:

We normally use the gray levels of border pixels to fill up the expanded region (for 3x3 mask). For larger masks a border region equal to half of the mask size is mirrored on the expanded region.

Customized Processing:

Using only those coefficients of the mask which overlap with the image pixels on the borders and ignoring the others.

# Spatial Filtering - Basics

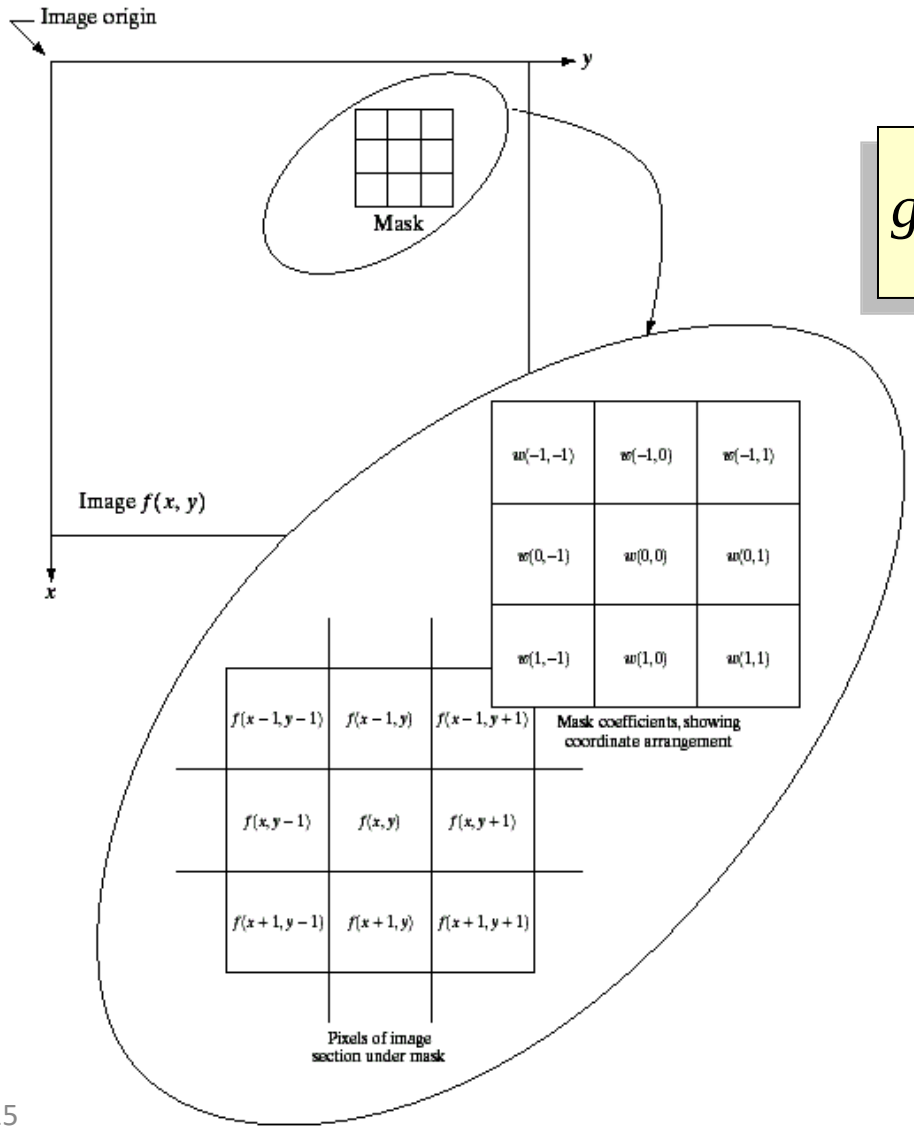


# Spatial Filtering - Basics

NUST

- 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 enhancement (spatial filtering)
- The size of the masks 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

# Spatial Filtering - Basics




$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

where a = $\frac{m-1}{2}$, b = $\frac{n-1}{2}$

$x = 0,1,2,....,M-1, y = 0,1,2,...,N-1$

Filtering can be given in equation form as shown above

# Spatial Filtering - Basics

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

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

$$z \longleftarrow z_1 w_1 + z_2 w_2 + z_3 w_3 + \cdots + z_9 w_9 = \sum_{i=1}^{9} z_i w_i$$

- $z$ 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.

# Spatial Filtering - Basics

NUST

- For blurring/noise reduction
- Blurring is usually used in preprocessing steps, e.g., to remove small details from an image prior to object extraction, or to bridge small gaps in lines or curves
- Equivalent to Low-pass spatial filtering in frequency domain because smaller (high frequency) details are removed based on neighborhood averaging (averaging filters)

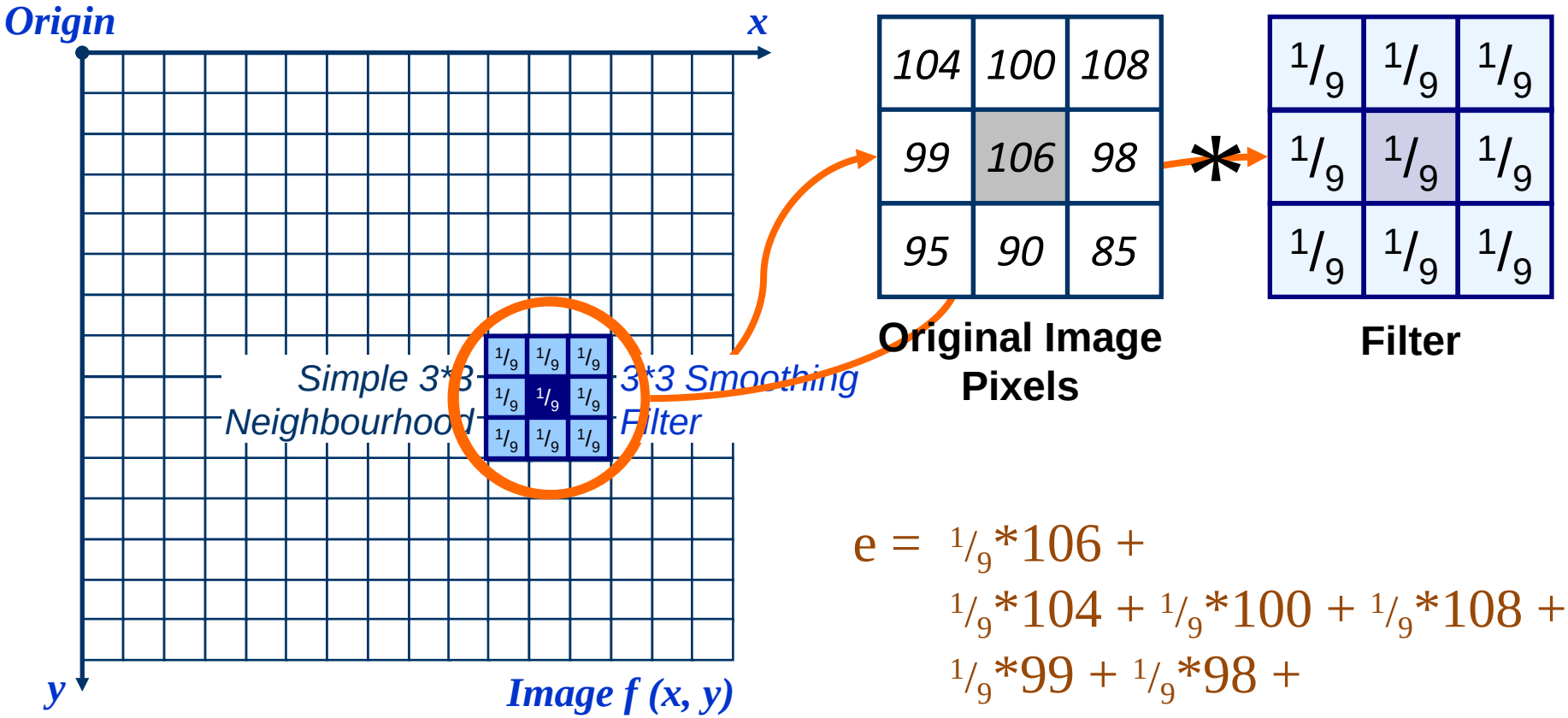
# Spatial Filtering - Basics

- Simply average all of the pixels in a neighborhood around a central value

| | | |
|---|---|---|
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |

Simple averaging filter

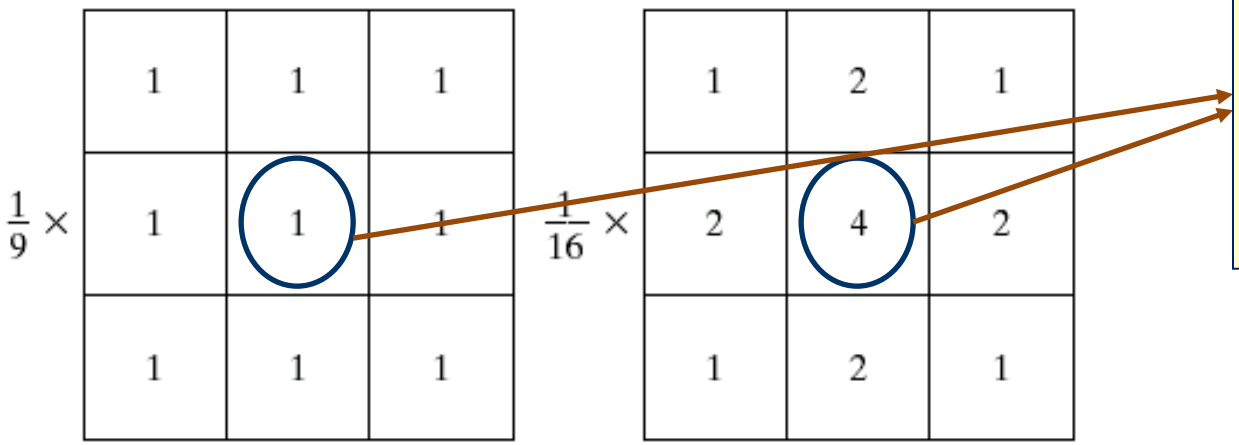
# Spatial Filtering - Basics




$$e = \tfrac{1}{9}*106 + \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85$$
$$= 98.3333$$

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

# Smoothing Filter




Consider the output pixel is positioned at the center

Box Filter all coefficients are equal

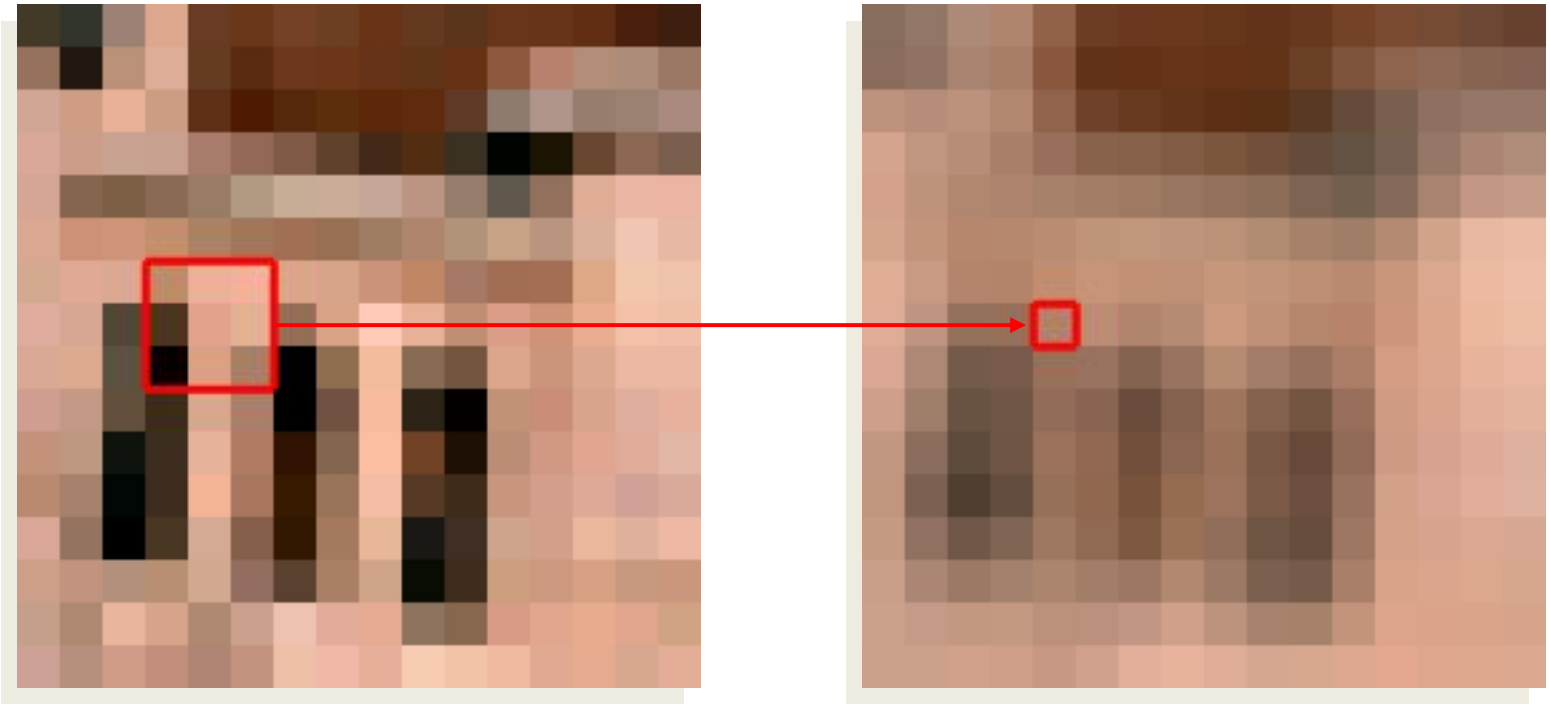
Weighted Average give more (less) weight to near (away from) the output location

# Smoothing Filter - Example

original

3x3 average

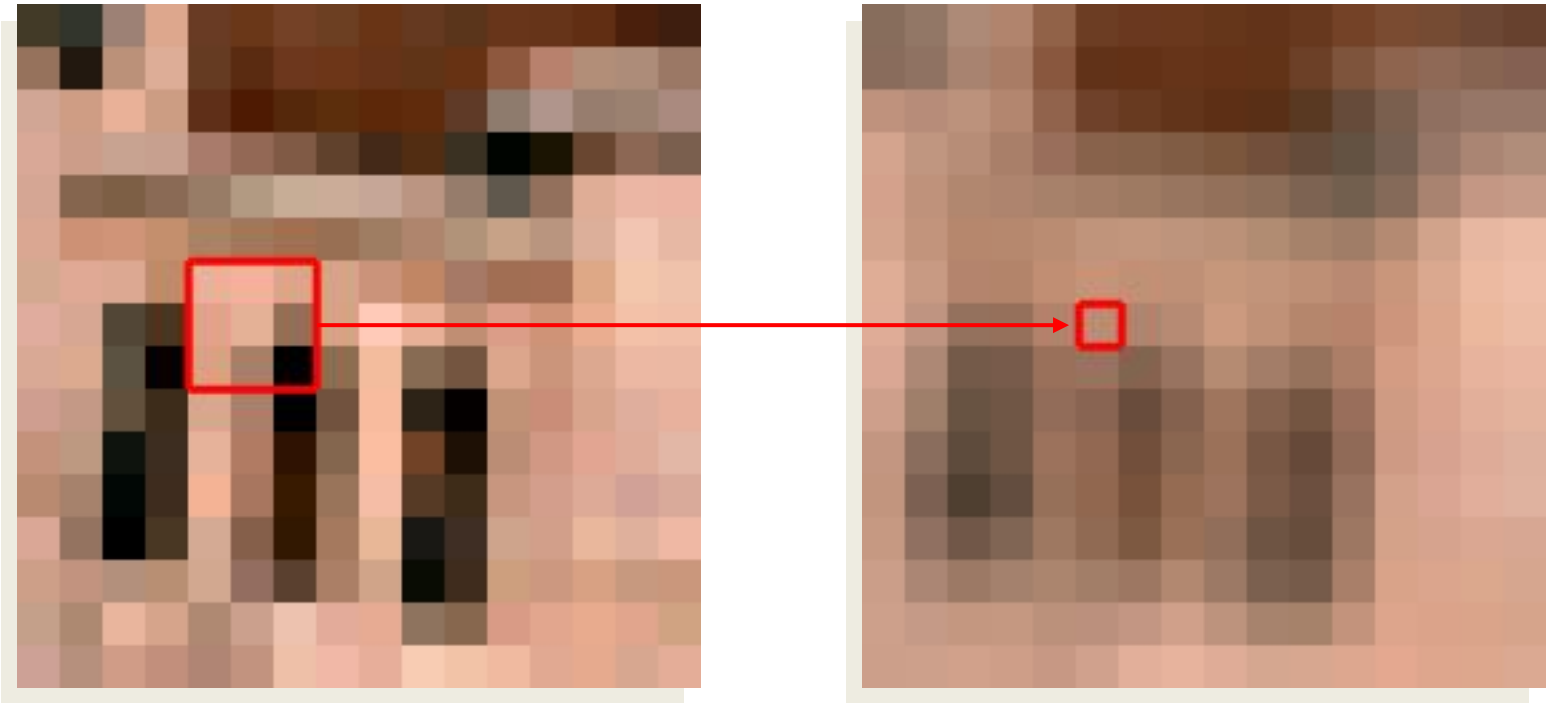
# Smoothing Filter - Example



original

3x3 average

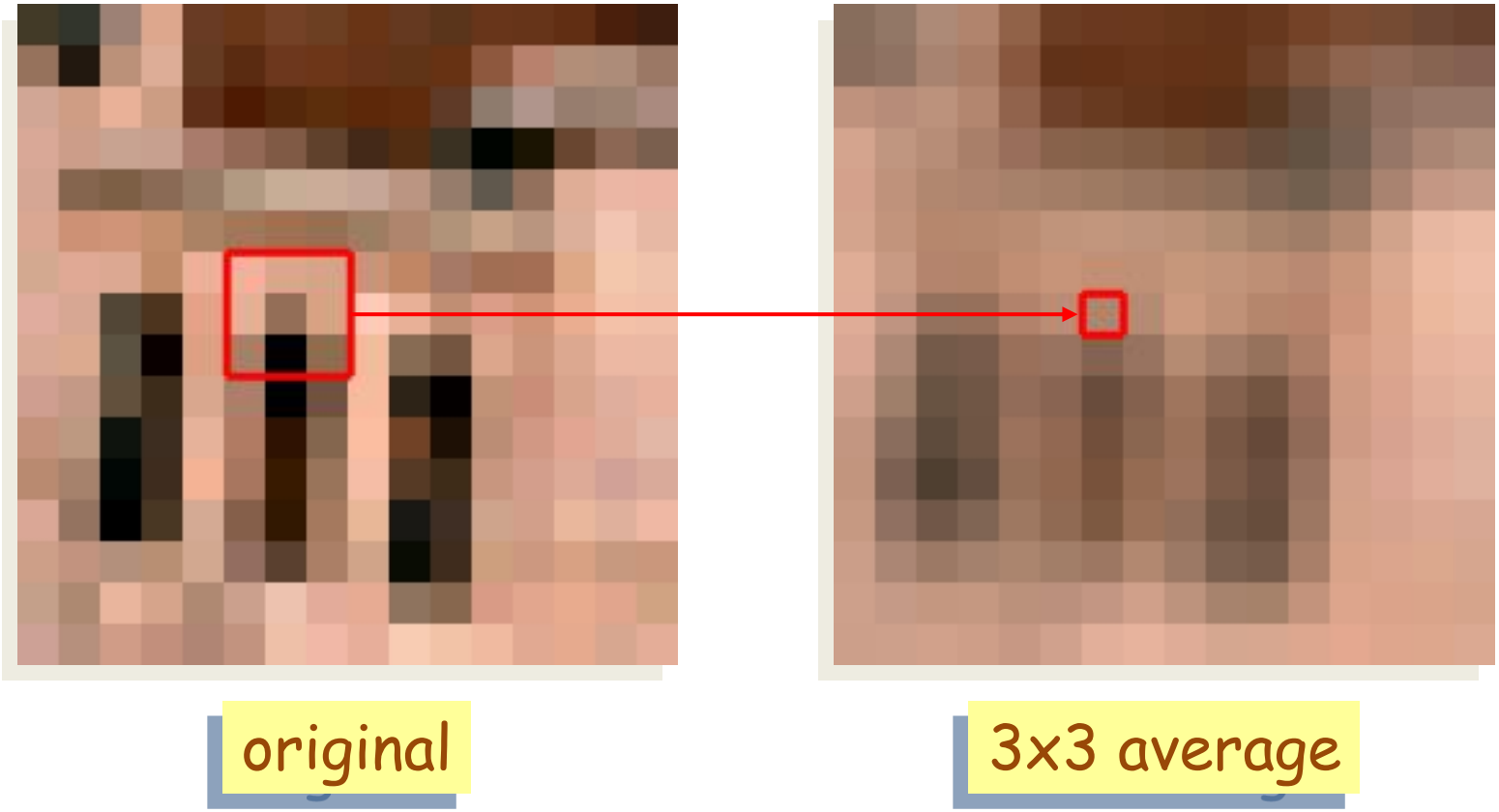
# Smoothing Filter - Example



original

3x3 average

# Smoothing Filter - Example



original

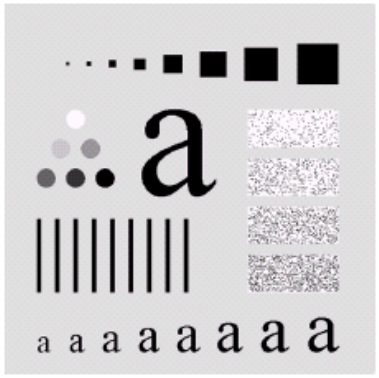
3x3 average

# Smoothing Filter - Example

NUST

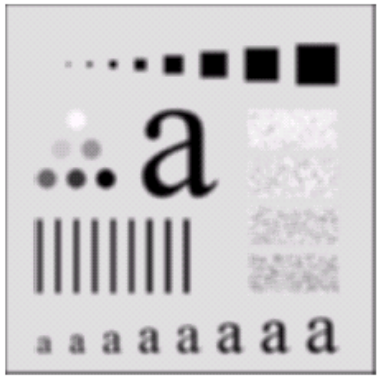
Original image

Size: 500x500



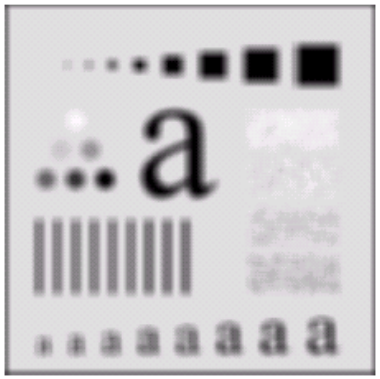
Smooth by 3x3 box filter

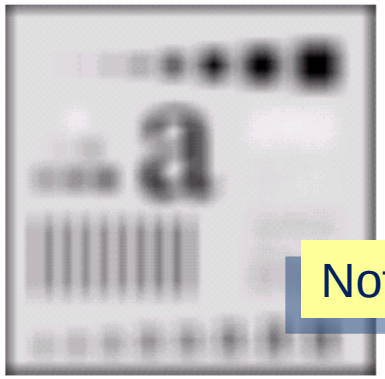
Smooth by 5x5 box filter



Smooth by 9x9 box filter

Smooth by 15x15 box filter





Smooth by 35x35 box filter

Notice how detail begins to disappear

# MATLAB - Exercise

- Smoothing filters
  - Implement a box filter
  - Add noise yourself

```
N = imnoise(I,'salt & pepper',0.02);
```

# MATLAB - Exercise

```
clear all;
II=imread('rain.jpg');
I=rgb2gray(II);
[x,y] = size(I)

I=double(I)/255;
I = imnoise(I,'salt & pepper',0.02);

w=(ones(3,3));
 O=ones(x,y);

for i=2:x-2
   for j=2:y-2
    O(i,j)=1/9*(w(1,1)*I(i-1,j-1)+w(1,2)*I(i-1,j)+w(1,3)*I(i-1,j+1)+w(2,1)*I(i,j-1) +
w(2,2)*I(i,j)+w(2,3)*I(i,j+1)+w(3,1)*I(i+1,j-1)+w(3,2)*I(i+1,j) +w(3,3)*I(i+1,j+1));
   end
end
```

Or use the built in conv2 function

# Sharpening Filters

Previously we have looked at smoothing filters which remove fine detail.

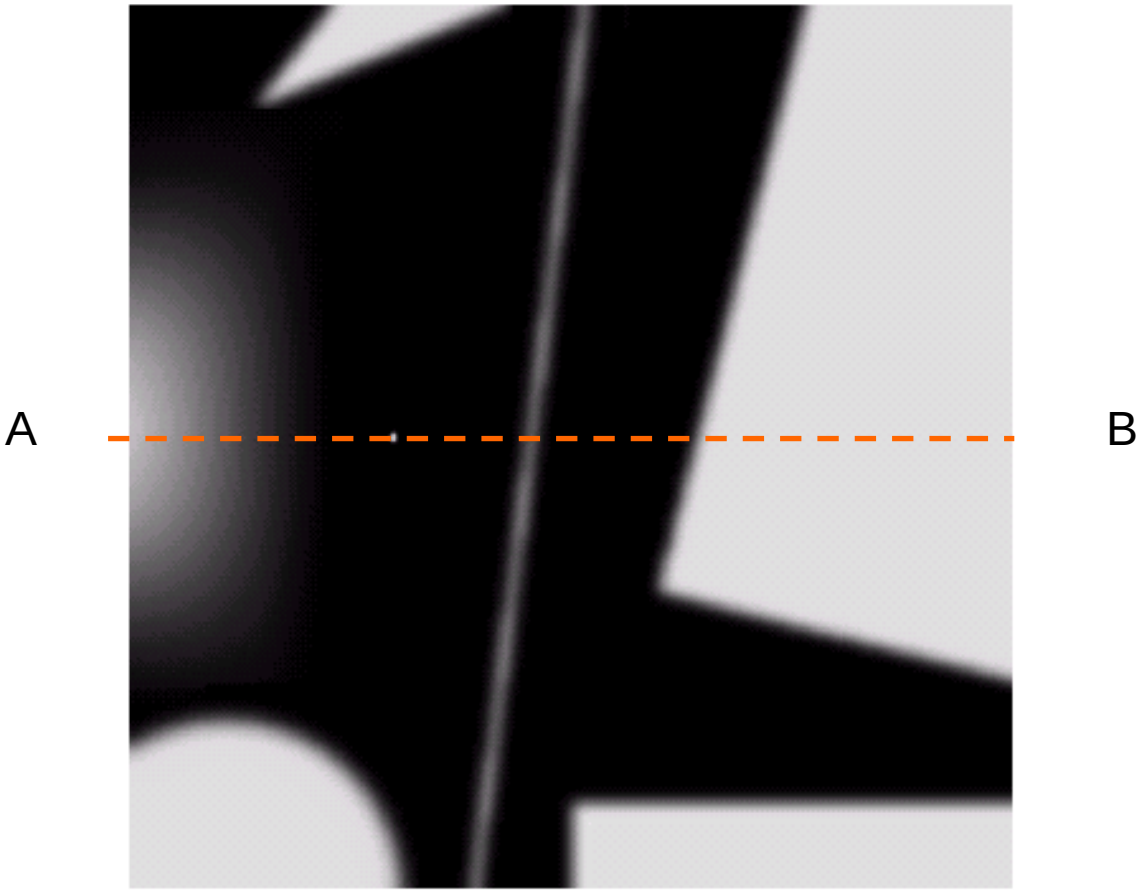
Sharpening spatial filters seek to highlight fine detail

- Remove blurring from images
- Highlight edges

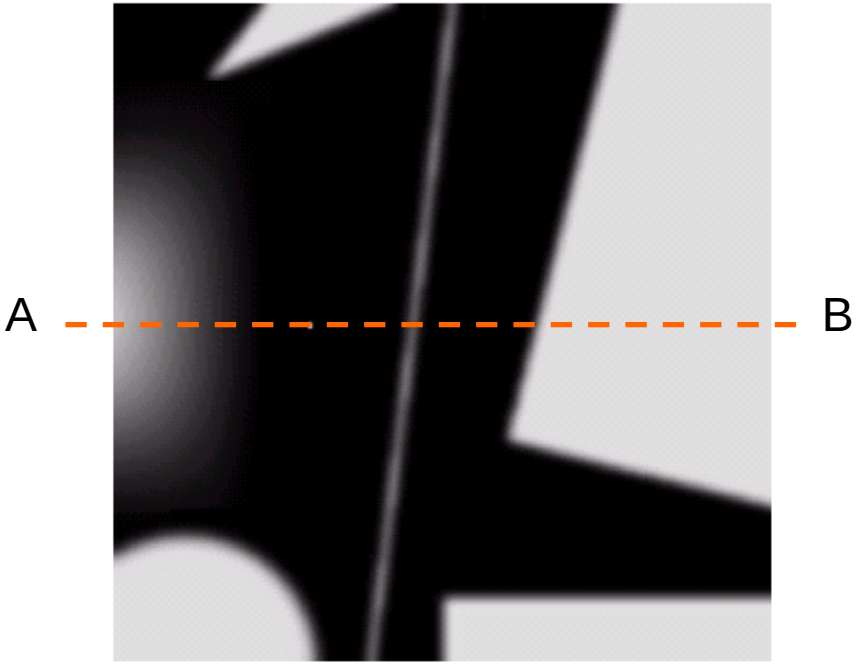
Sharpening filters are based on spatial differentiation

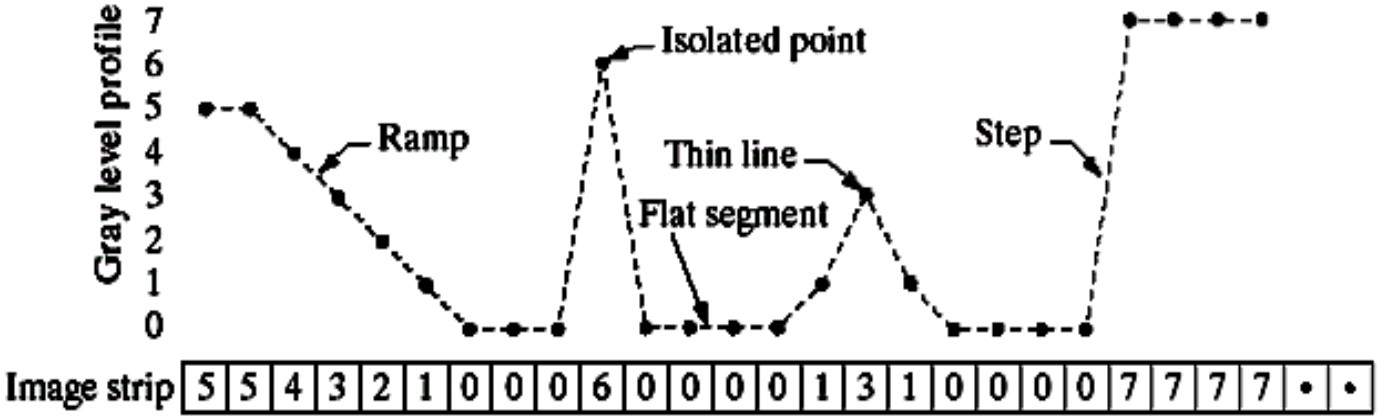
# Spatial Differentiation

- Let's consider a simple 1 dimensional example



# Spatial Differentiation



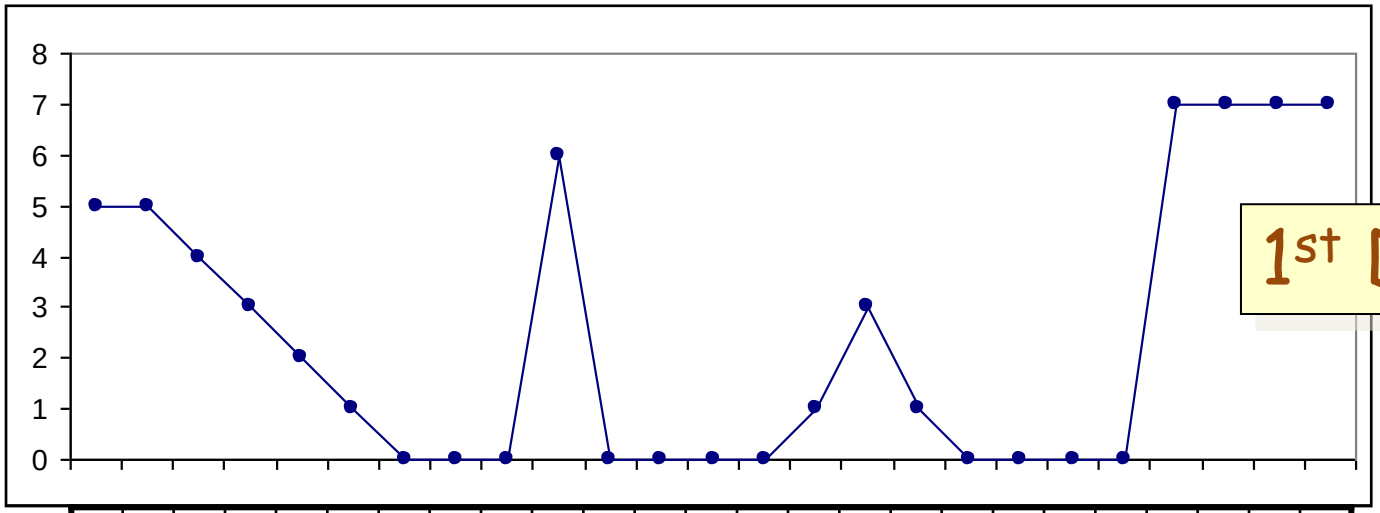
# Spatial Differentiation

The 1st derivative of a function is given by:

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

Its just the difference between subsequent values and measures the rate of change of the function.

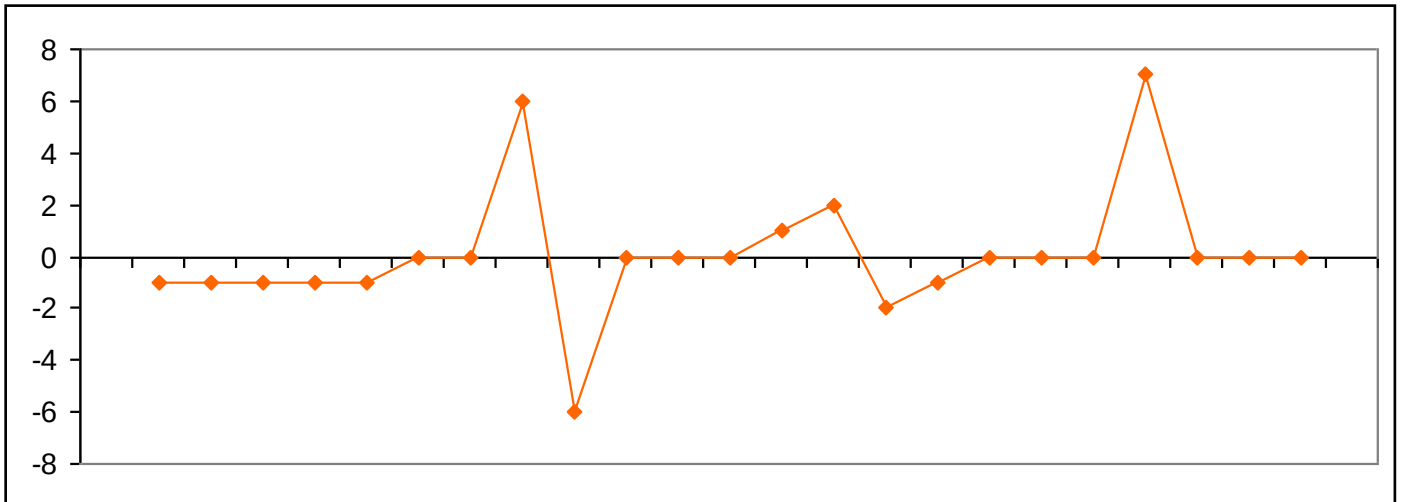
# Spatial Differentiation




| 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

1st Derivative

| | -1 | -1 | -1 | -1 | -1 | 0 | 0 | 6 | -6 | 0 | 0 | 0 | 1 | 2 | -2 | -1 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



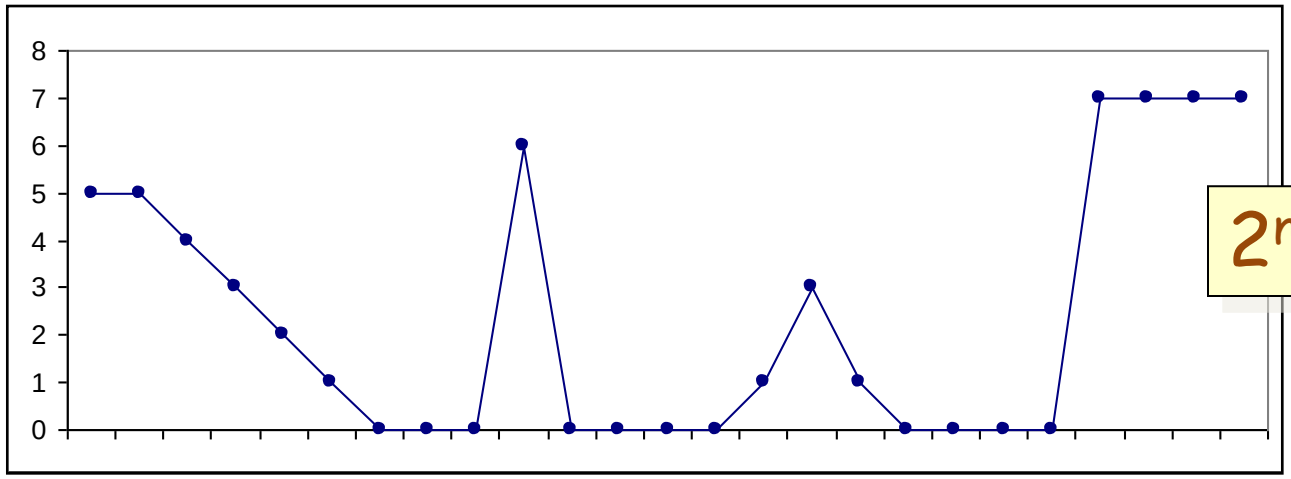
# Spatial Differentiation

The 2nd derivative of a function is given by:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

Simply takes into account the values both before and after the current value

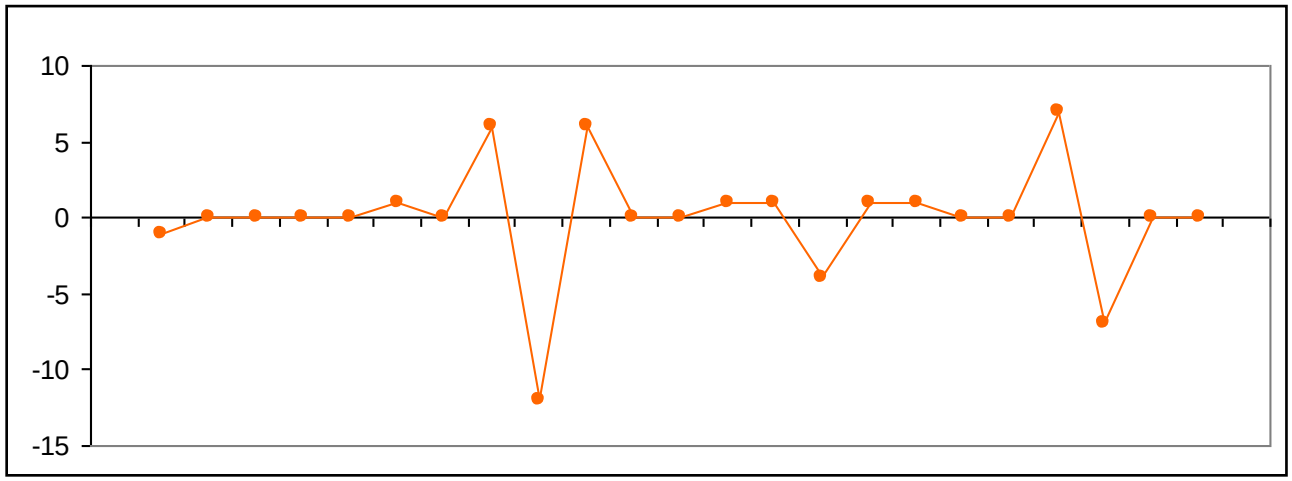
# Spatial Differentiation




2nd Derivative

| 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | -1 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | -12 | 6 | 0 | 0 | 1 | 1 | -4 | 1 | 1 | 0 | 0 | 7 | -7 | 0 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



# 2nd Derivative – Image Enhancement

The 2nd derivative is more useful for image enhancement than the 1st derivative - *Stronger response to fine detail*

We will come back to the 1st order derivative later on

The first sharpening filter we will look at is the *Laplacian*

# Laplacian Filter

The Laplacian is defined as follows:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

# Laplacian Filter

So, the Laplacian can be given as follows:

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

Can we implement it using a mask?

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

# Laplacian Filter

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

a b
c d

**FIGURE 3.39**
(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

# Laplacian Filter – Image Enhancement

The result of a Laplacian filtering is not an enhanced image

To generate the final enhanced image

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f, & w_5 < 0 \\ f(x,y) + \nabla^2 f, & w_5 > 0 \end{cases}$$

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

# Laplacian Filter – Image Enhancement

The entire enhancement can be combined into a single filtering operation.

$$g(x,y) = f(x,y) - \nabla^2 f$$

$$= f(x,y) - [f(x+1,y) + f(x-1,y)$$

$$+ f(x,y+1) + f(x,y-1)$$

$$- 4f(x,y)]$$

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

# Laplacian Filter – Image Enhancement

The entire enhancement can be combined into a single filtering operation

$$g(x,y) = f(x,y) - \nabla^2 f$$

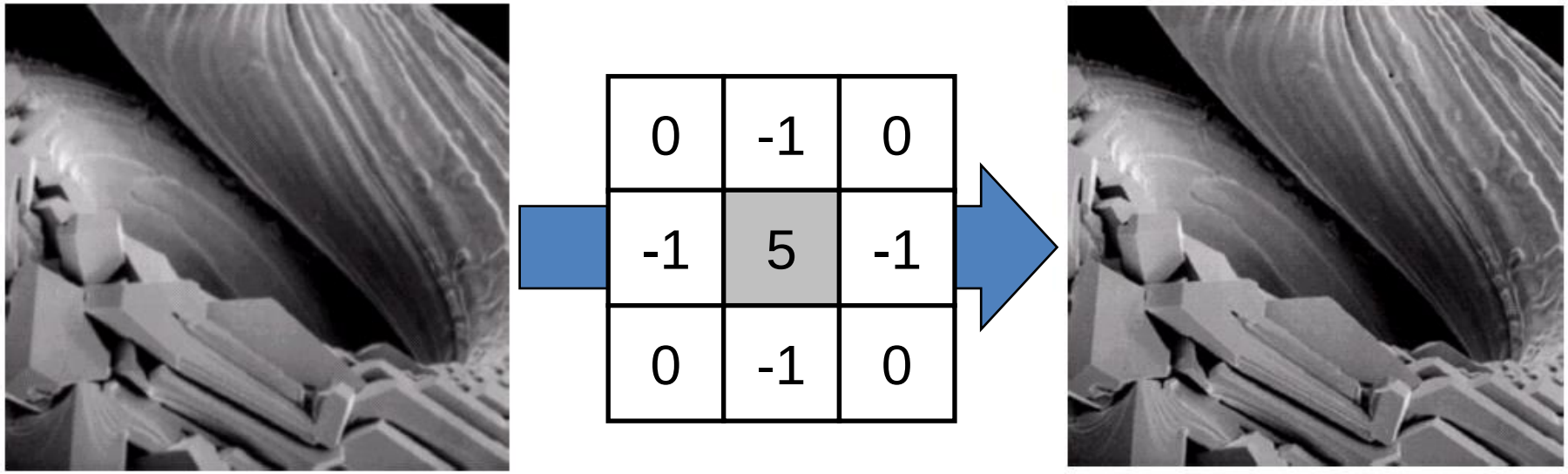
$$= 5f(x,y) - f(x+1,y) - f(x-1,y)$$

$$- f(x,y+1) - f(x,y-1)$$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

# Laplacian Filter – Image Enhancement

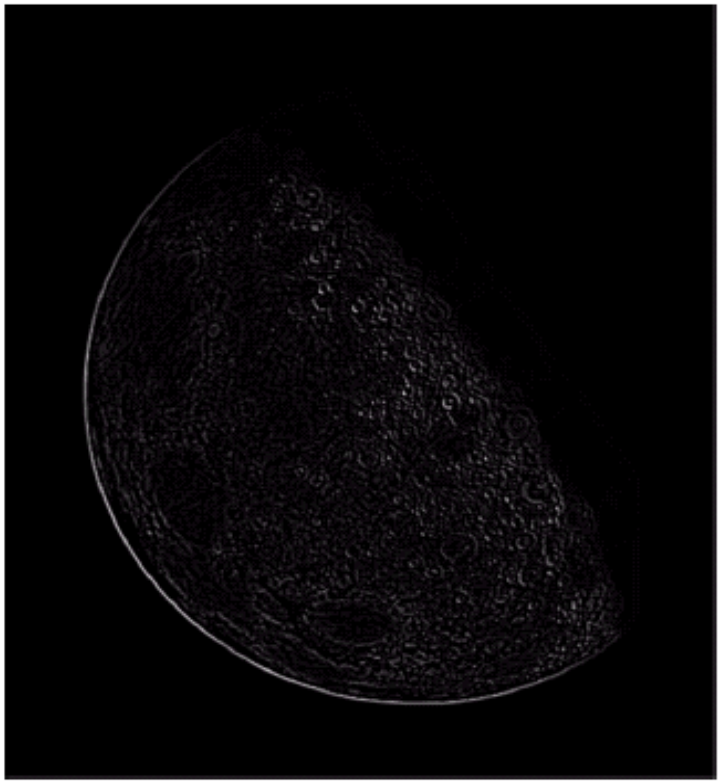
This gives us a new filter which does the whole job for us in one step.



# Laplacian Filter – Image Enhancement

Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities

Original Image



Laplacian Filtered Image

# Laplacian Filter – Image Enhancement

Original Image

+

Laplacian Filtered Image

=

Sharpened Image

In the final sharpened image edges and fine detail are much more obvious

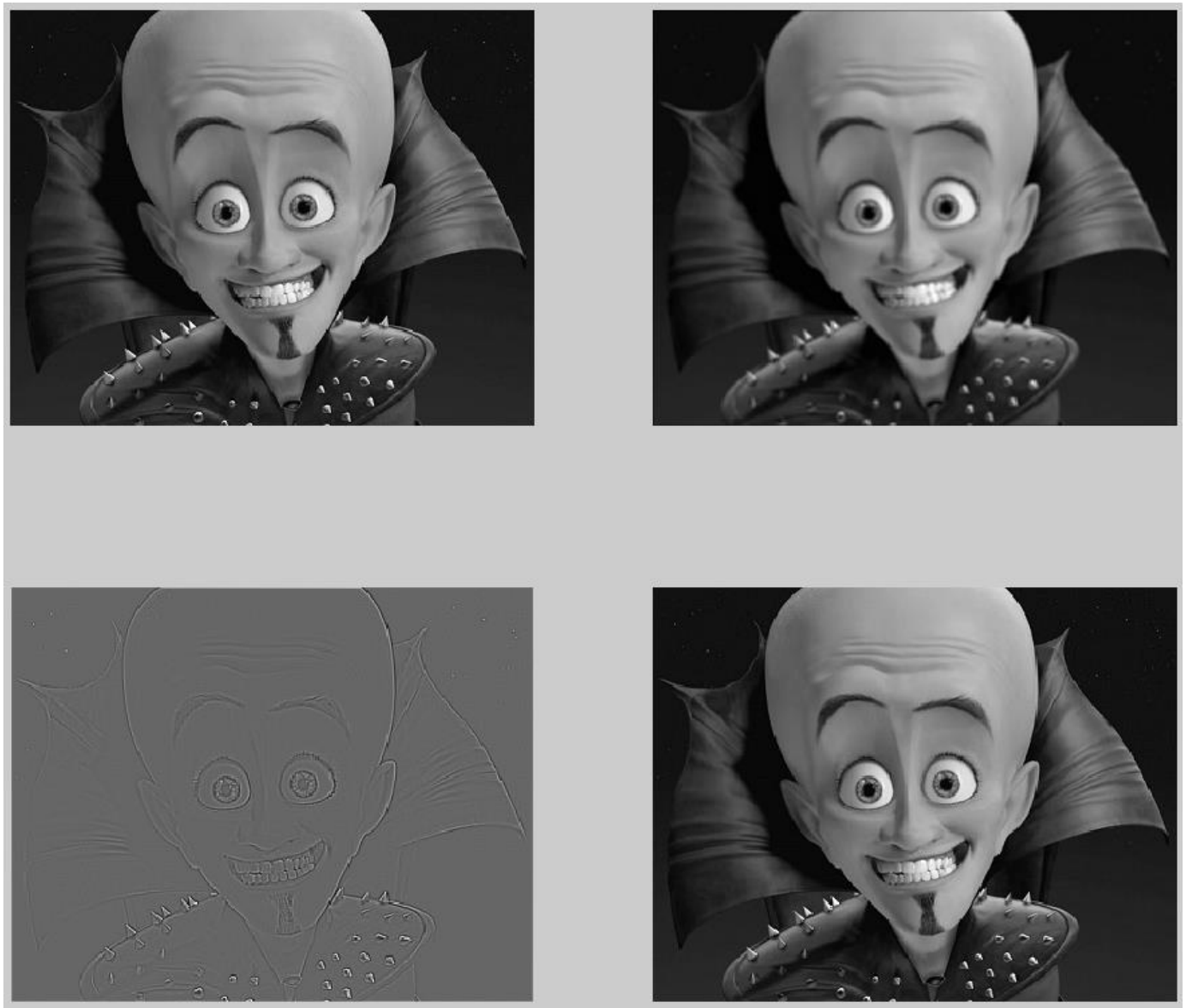
# Laplacian Filter – Image Enhancement

# Laplacian Filter – Image Enhancement

NUST

```
x_rgb = imread('MM.jpg');
x = double(rgb2gray(x_rgb));

figure;
subplot(2,2,1); imshow(x,[]);

mask1 = [1 1 1; 1 1 1; 1 1 1];
x_lowpass = conv2(x,mask1,'same');
x_lowpass = (x_lowpass - min(min(x_lowpass)) ) / (max(max(x_lowpass))-min(min(x_lowpass)) );
subplot(2,2,2); imshow(x_lowpass,[]);

mask2 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
x_highpass = conv2(x,mask2,'same');
x_highpass = (x_highpass - min(min(x_highpass)) ) / (max(max(x_highpass))-min(min(x_highpass)) );
subplot(2,2,3); imshow(x_highpass,[]);


x_enhanced = x_lowpass+x_highpass;
subplot(2,2,4); imshow(x_enhanced,[]);
```

# Laplacian Filter – Image Enhancement



# Gradient Operators

## Simplest operator

$$\frac{\partial f}{\partial x} = (z_8 - z_5), \frac{\partial f}{\partial y} = (z_6 - z_5)$$

$$\nabla f = \sqrt{(z_8 - z_5)^2 + (z_6 - z_5)^2}$$

$$\nabla f \approx |(z_8 - z_5)| + |(z_6 - z_5)|$$

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

# Gradient Operators

## Prewitt operator

$$\nabla f \approx \left|(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)\right| + \left|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)\right|$$

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

$$\frac{\partial f}{\partial y} =$$

| | | |
|---|---|---|
| **-1** | **-1** | **-1** |
| **0** | **0** | **0** |
| **1** | **1** | **1** |

Extract horizontal edges

$$\frac{\partial f}{\partial x} =$$

| | | |
|---|---|---|
| **-1** | **0** | **1** |
| **-1** | **0** | **1** |
| **-1** | **0** | **1** |

Extract vertical edges

# Gradient Operators

## Sobel operator

$$\frac{\partial f}{\partial y} =$$

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

Extract horizontal edges

$$\frac{\partial f}{\partial x} =$$

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

Extract vertical edges

Emphasize more the current point (x direction)

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

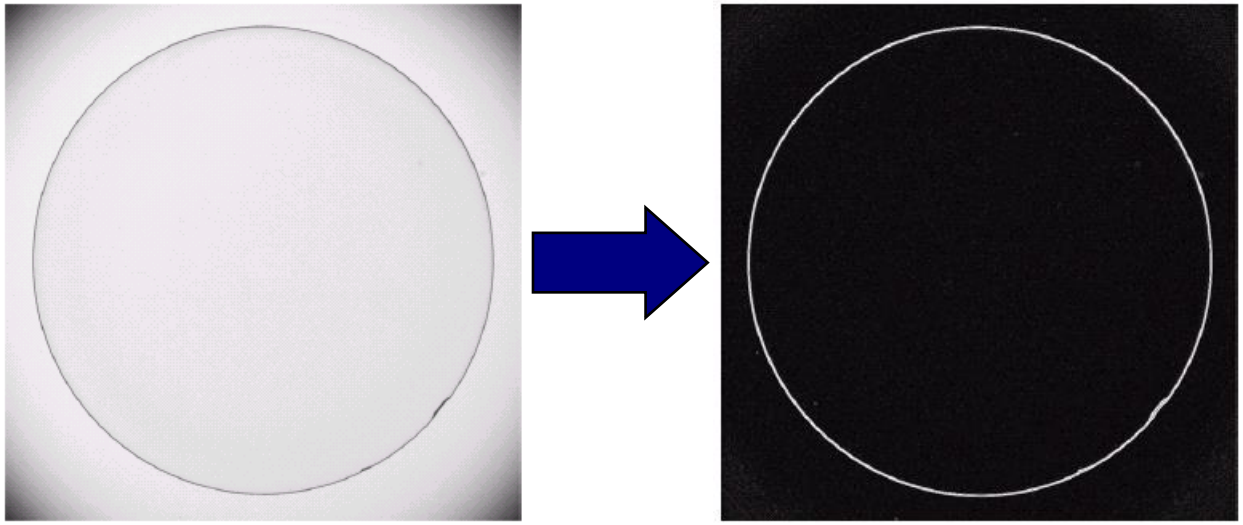
Emphasize more the current point (y direction)

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

Pixel Arrangement

# Gradient Operators

Sobel Operator along with the threshold.



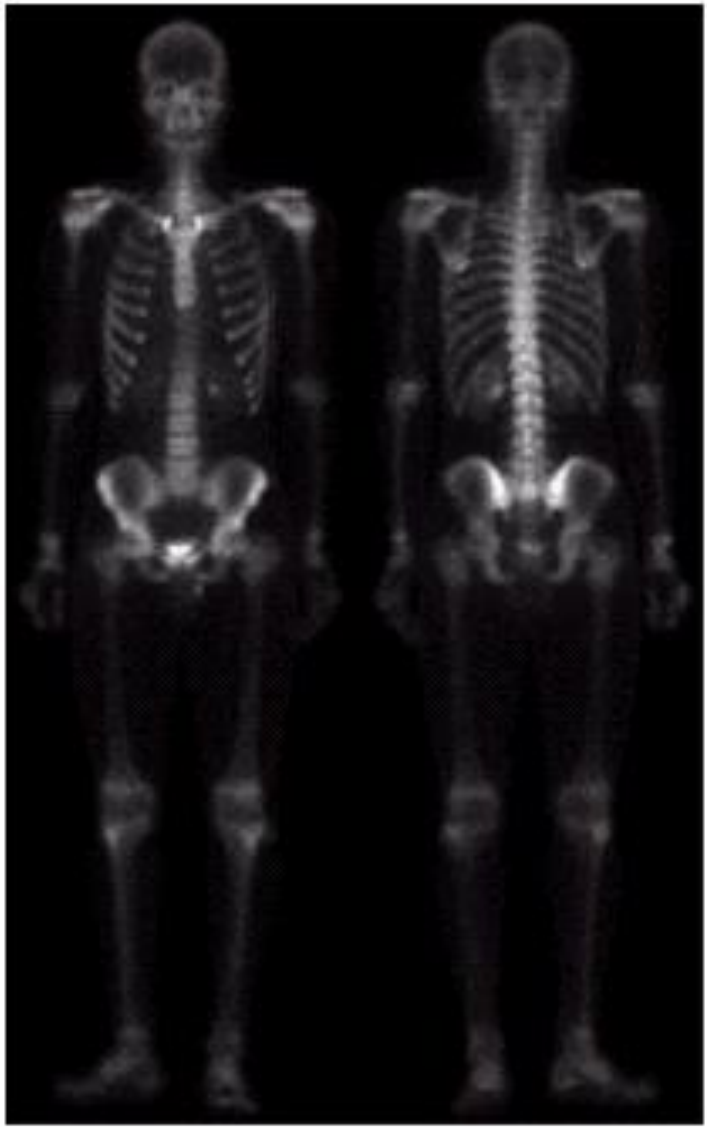
An image of a contact lens which is enhanced in order to make defects more obvious

# Combining Spatial Enhancement Methods

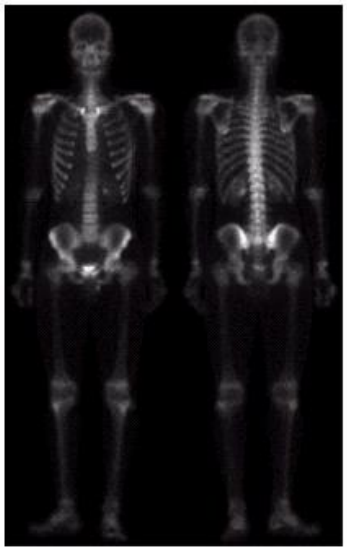
Successful image enhancement is typically not achieved using a single operation

Rather we combine a range of techniques in order to achieve a final result

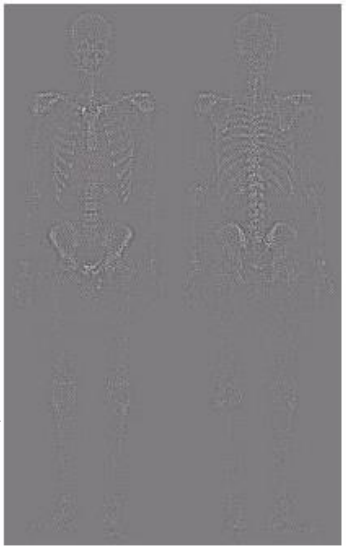
This example will focus on enhancing the bone scan



# Combining Spatial Enhancement Methods



(a)

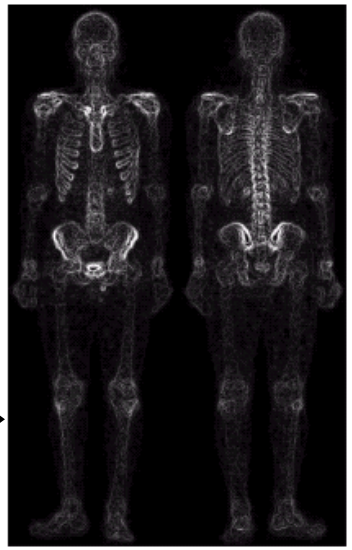


Laplacian filter of bone scan (a)

(b)

Sharpened version of bone scan achieved by subtracting (a) and (b)

(c)



Sobel filter of bone scan (a)

(d)

# Combining Spatial Enhancement Methods

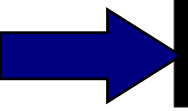
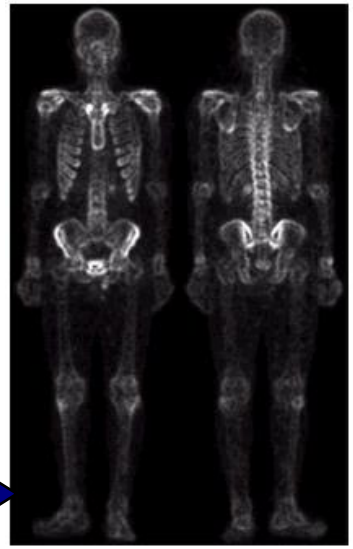


Image (d) smoothed with a 5*5 averaging filter

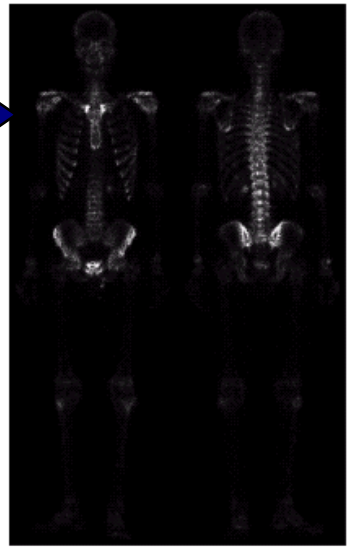
The product of (c) and (e) which will be used as a mask

**(e)**



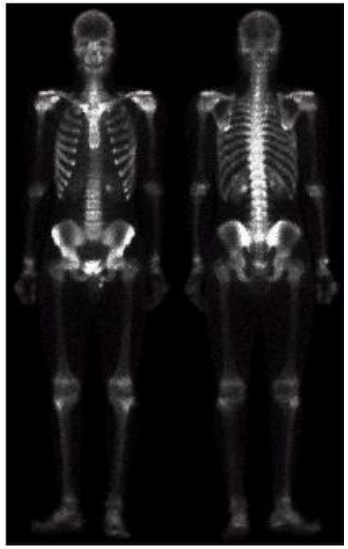
Sharpened image which is sum of (a) and (f)

**(f)**



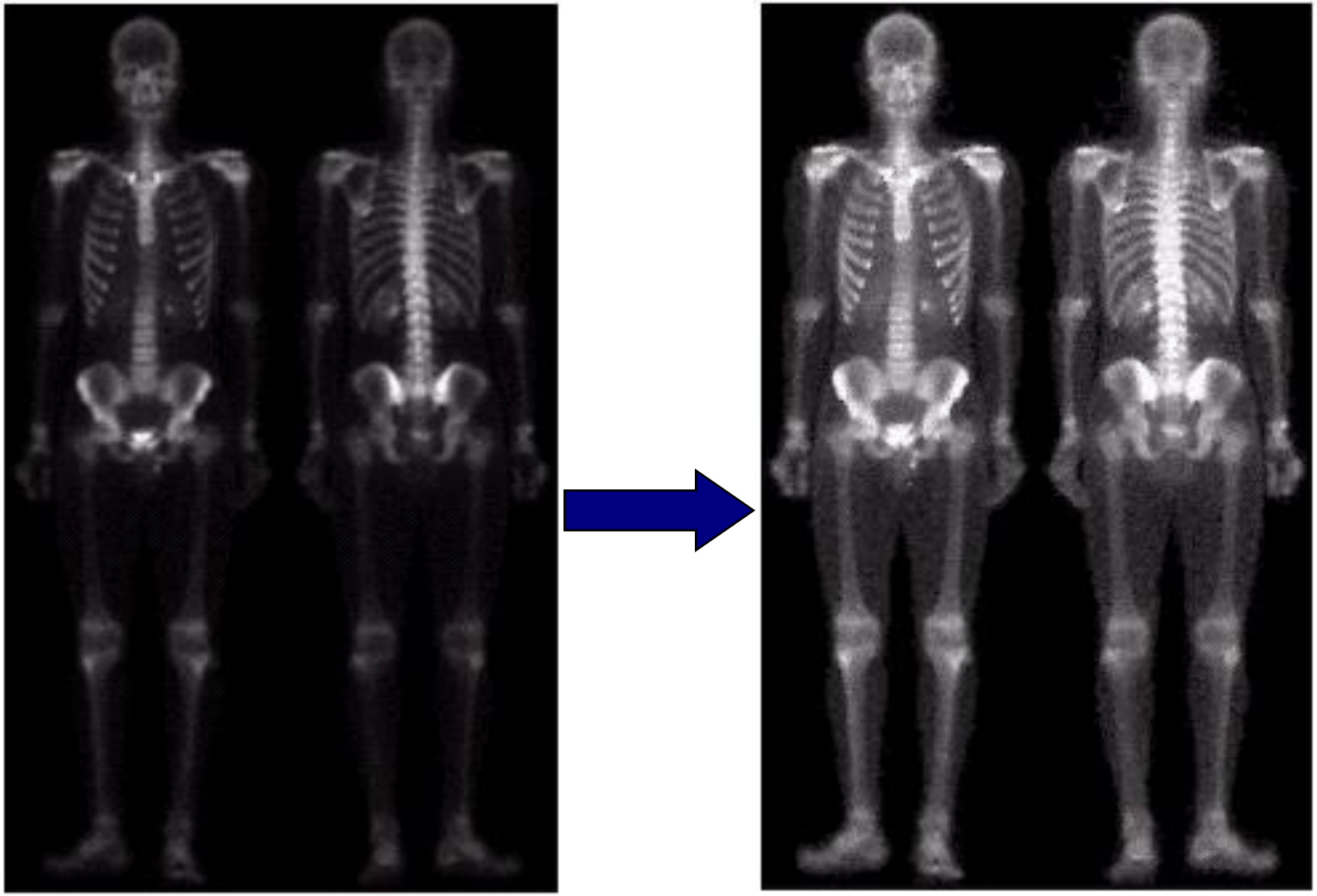
Result of applying a power-law trans. to (g)

**(g)**



**(h)**

# Combining Spatial Enhancement Methods



Original

Enhanced

# Noise Models

Most types of noise are modeled as known probability density functions.

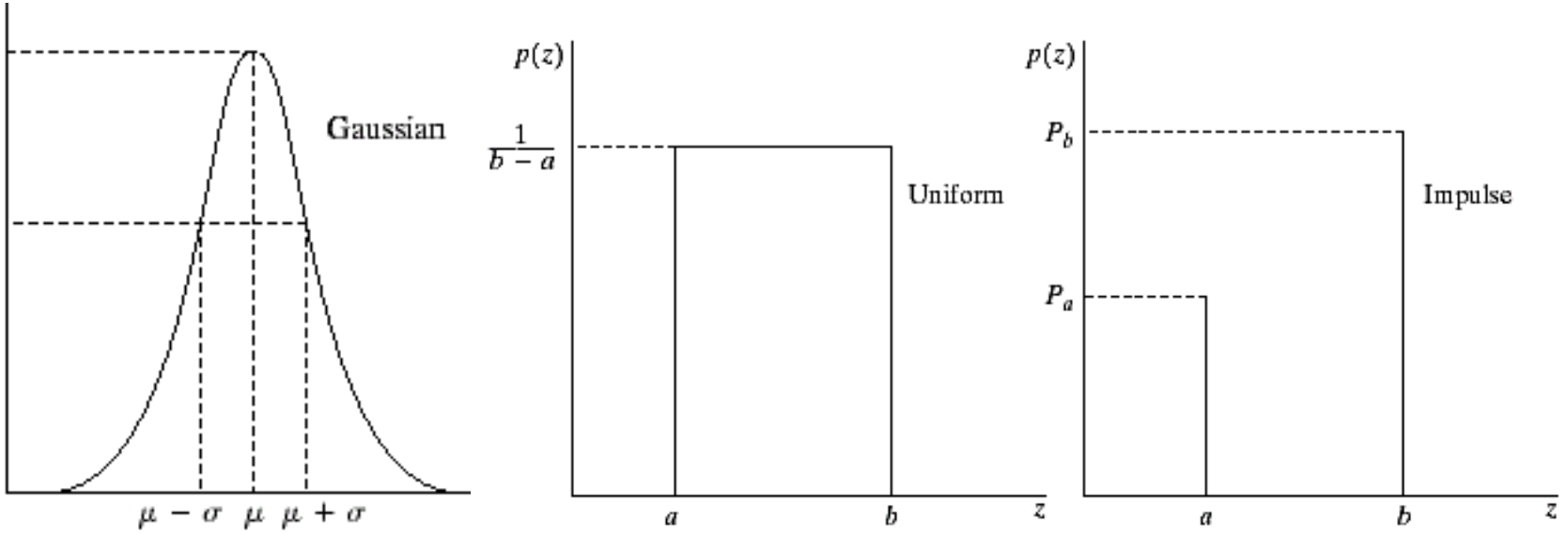
Noise model is decided based on understanding of the physics of the sources of noise.

Gaussian: Poor illumination

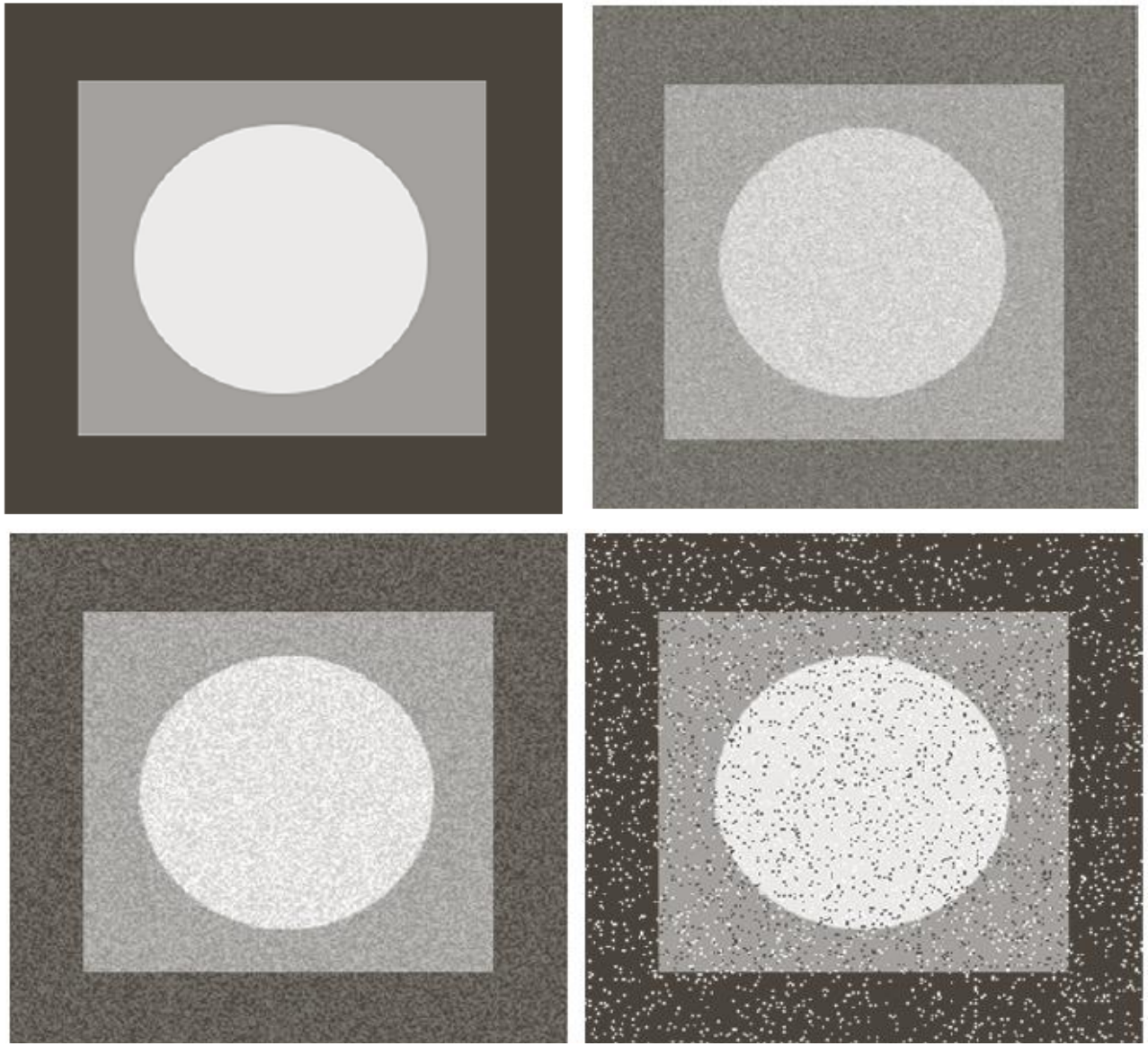
Impulse: Faulty Pixels

Uniform: Quantization

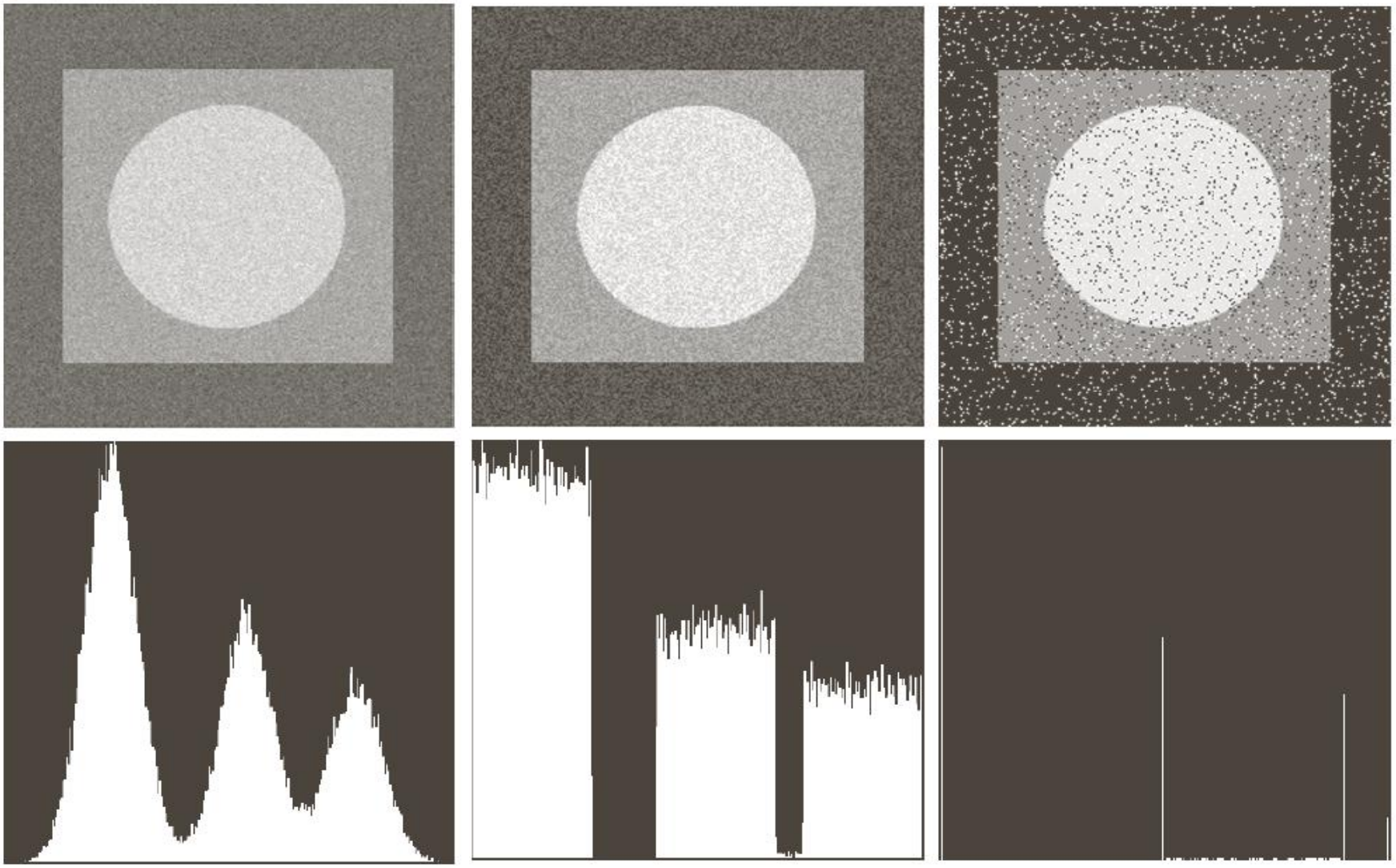
Parameters can be estimated based on histogram on small flat area of an image.



# Noise Models



# Noise Models



# Noise Filters

NUST

## Spatial Domain for Noise Removal without caring for Degradation

- Mean Filters
- Order Statistics Filter
- Adaptive Filter (based on local statistics)

## Frequency Domain

- Band Reject/Notch Filtering

# Noise Filters

Output is based on order of gray levels in the masked area.

## Order filters

- Minimum
- Maximum
- Median
- Mid-Point
- Alpha-Trimmed

## Mean filters

- Arithmetic Mean
- Geometric Mean
- Contra-Harmonic Mean

# Order Filters

- Given an *N*x*N* window, the pixel values can be ordered from smallest to largest as follows:
  - $I_1 \leq I_2 \leq I_3 \leq \ldots\ldots \leq I_{N^2}$
  - Where $\{I_1, I_2, I_3, \ldots\ldots, I_{N^2}\}$ *are* the gray-level values of the subset of pixels in the image, that are in the *N*x*N* window.
- Different types of order filters select different values from the ordered pixel list.

# Order Filters

- Median filter
  - Select the middle pixel value from the ordered set.
  - Used to remove salt-and-pepper noise.
- Maximum filter
  - Select the highest pixel value from the ordered set.
  - Remove pepper-type noise.
- Minimum filter
  - Select the lowest pixel value from the ordered set.
  - Remove salt-type noise.
- As the size of the window gets bigger, the more information loss occurs.
  - With windows larger than about 5x5, the image acquires an artificial, “painted”, effect.

# Order Filters

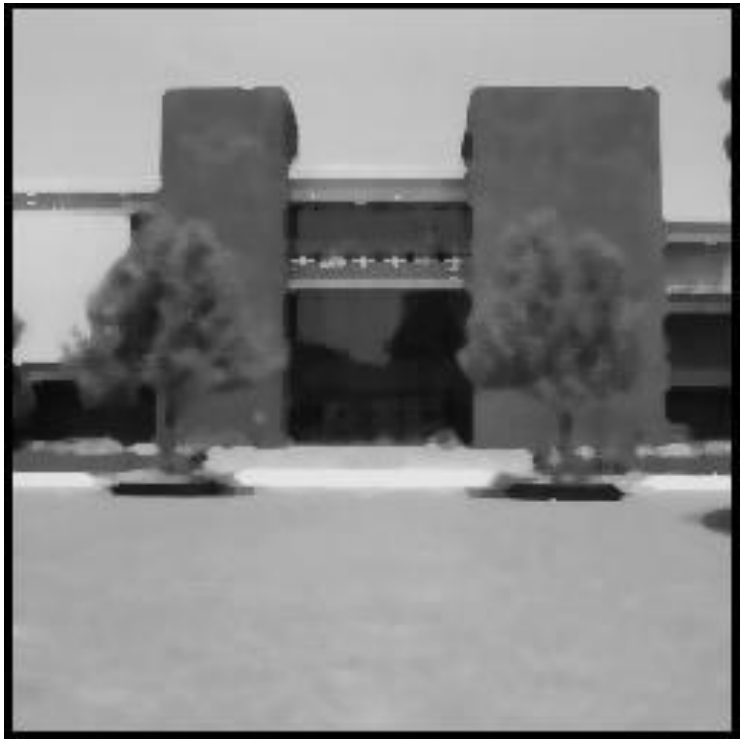
NUST

## Minimum Filter

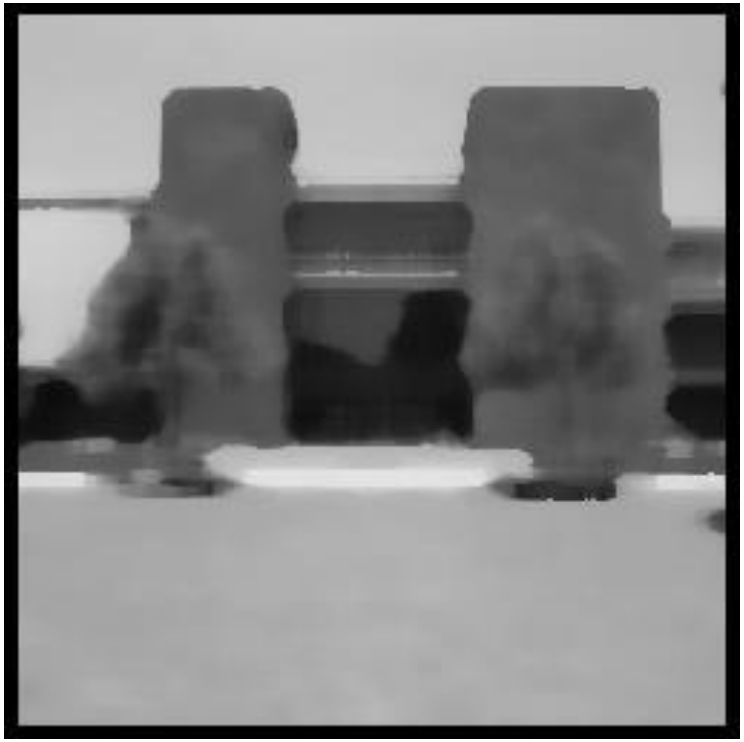
Image with salt noise
Probability = .04

Result of minimum filtering
Mask 3 x 3

# Order Filters



Minimum filtering
Mask 5 x 5



Minimum filtering
Mask 9 x 9

# Order Filters

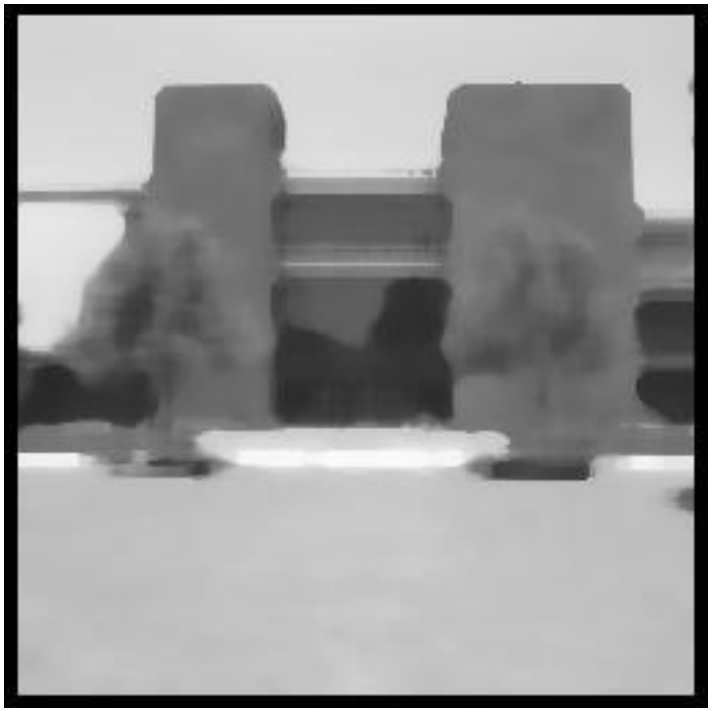
## Maximum Filter

Image with pepper noise
Probability = .04

Maximum filtering
Mask 3 x 3

# Order Filters

Maximum filtering
Mask 5 x 5



Maximum filtering
Mask 9 x 9

# Order Filters

- Order filters can also be defined to select a specific pixel rank within the ordered set.
  - For example, we may find the second highest value is the better choice than the maximum value for certain pepper noise.
  - This type of ordered selection is application specific.
- Minimum filter tend to darken the image and maximum filter tend to brighten the image.

# Median Filtering

| | | |
|---|---|---|
| 10 | 20 | 20 |
| 20 | 15 | 20 |
| 20 | 25 | 100 |

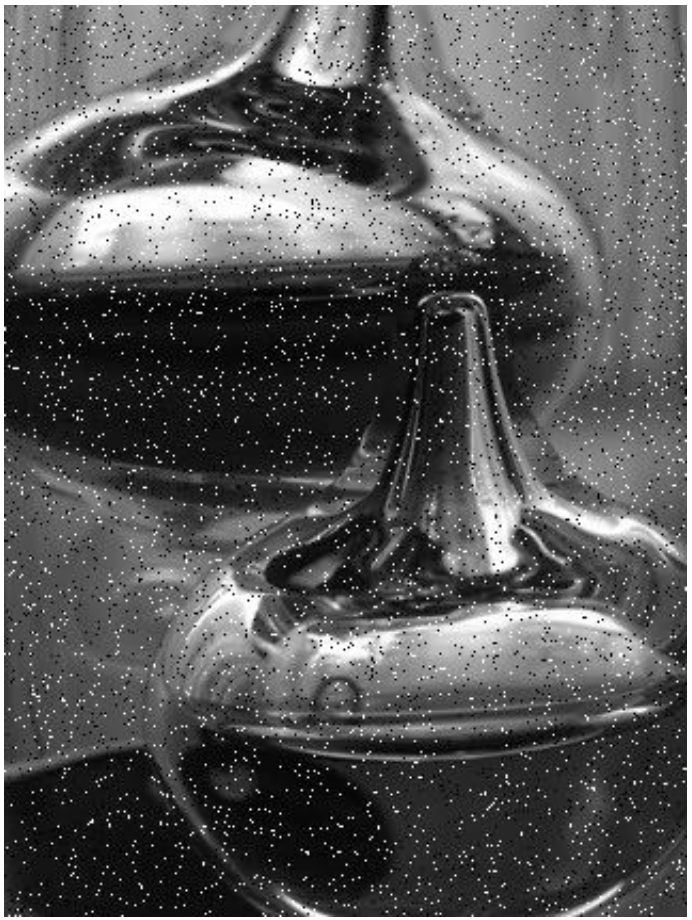
Median = ? 20

Sort the values
Determine the median

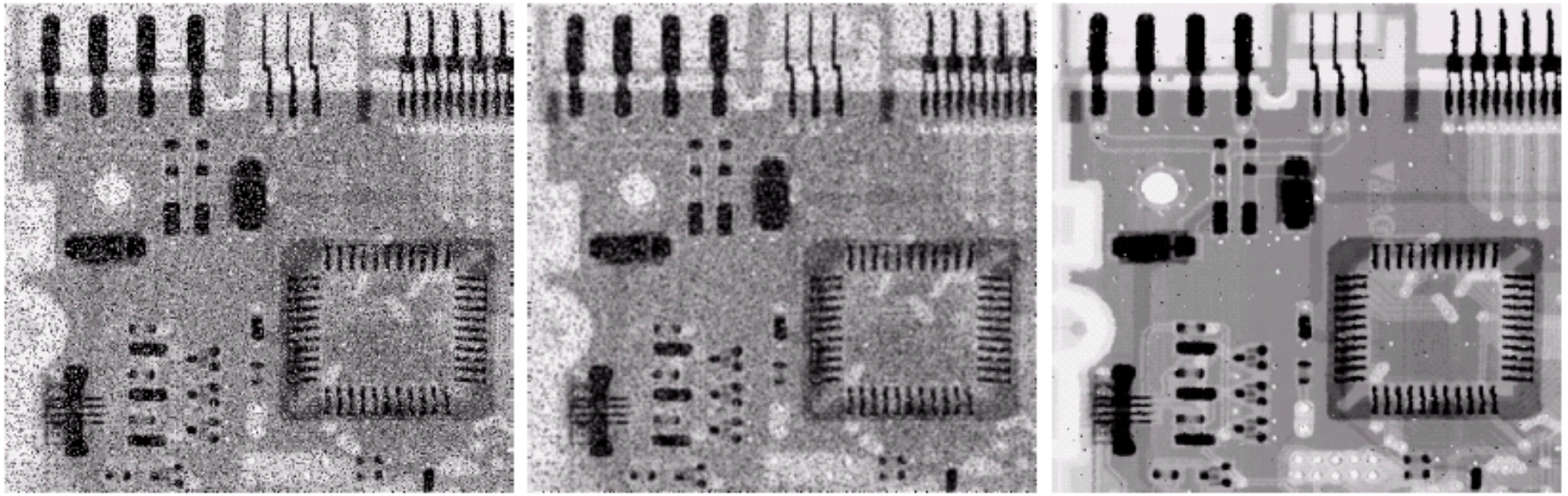
- Particularly effective when
  - The noise pattern consists of strong impulse noise ( salt-and-pepper)

# Median Filtering - Application

Removal of Salt and Pepper Noise.



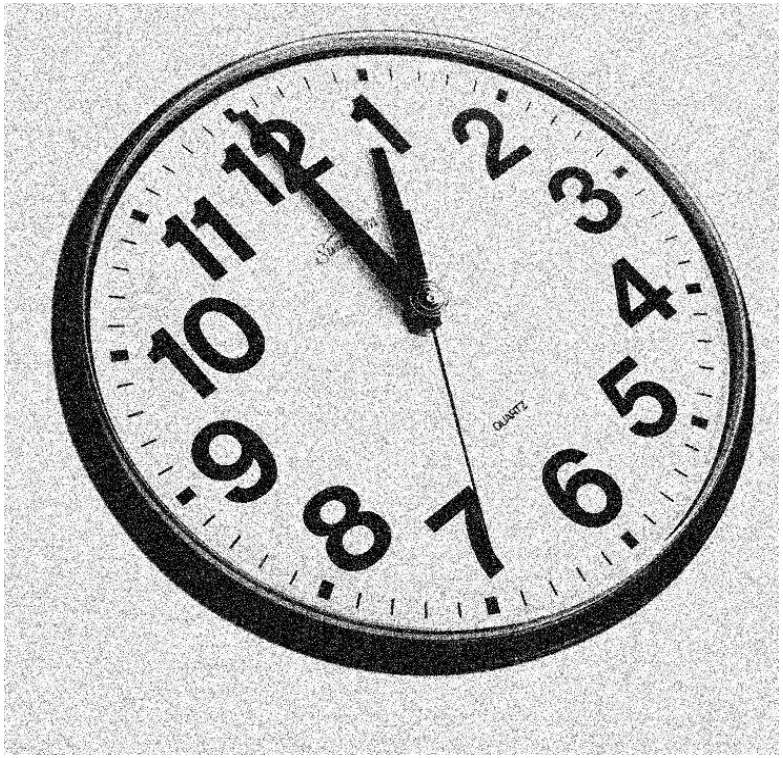
# Mean-Median Filter - comparison



a b c

**FIGURE 3.37** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a $3 \times 3$ averaging mask. (c) Noise reduction with a $3 \times 3$ median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

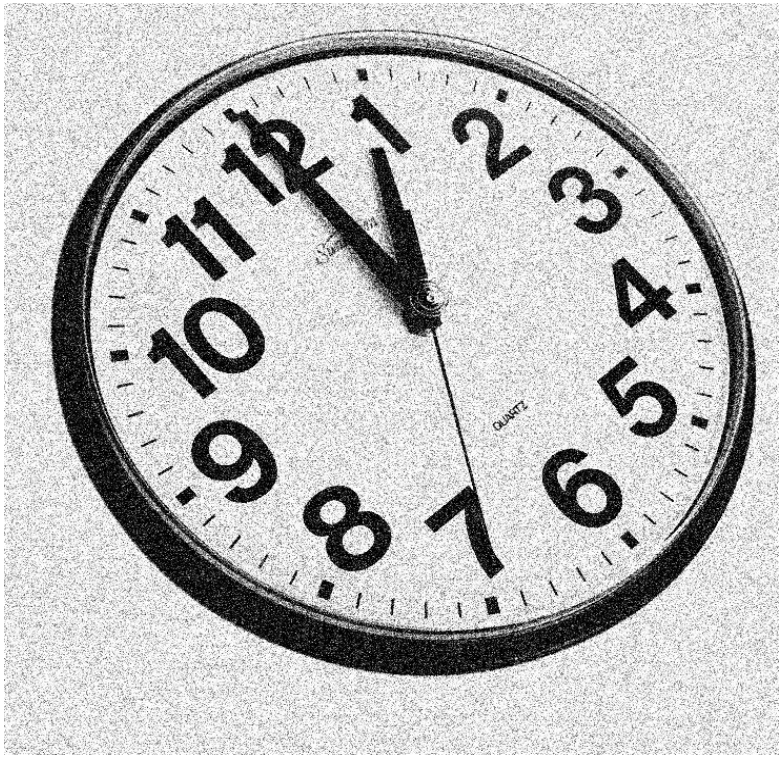
# Mean-Median Filter - comparison



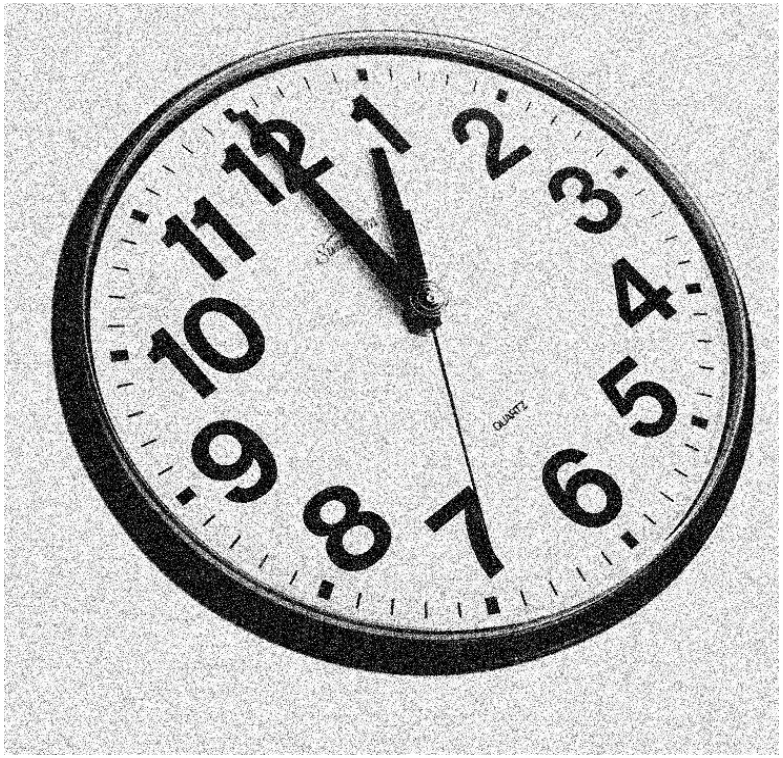
Noisy

Original

# Mean-Median Filter - comparison

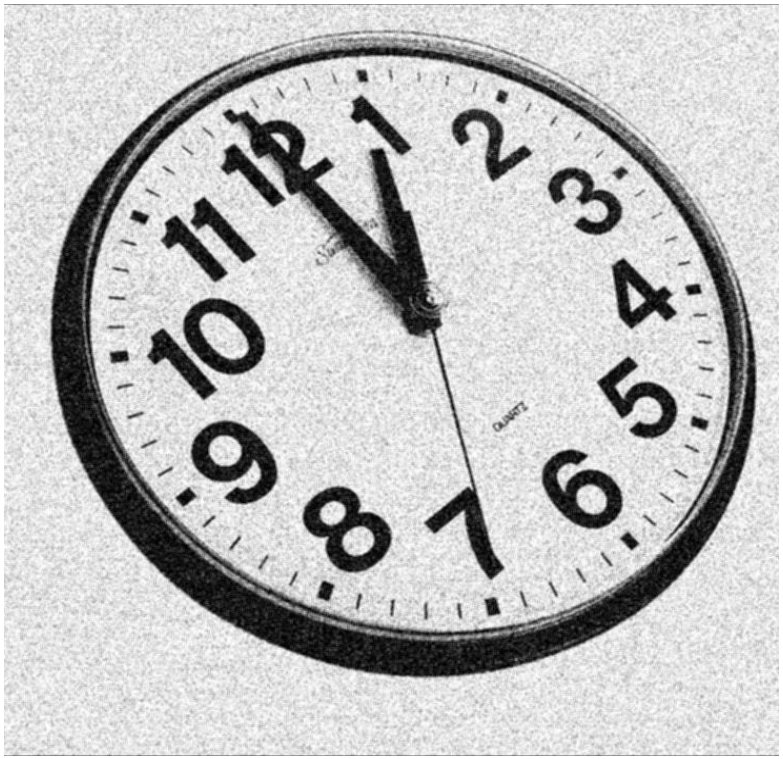


Noisy

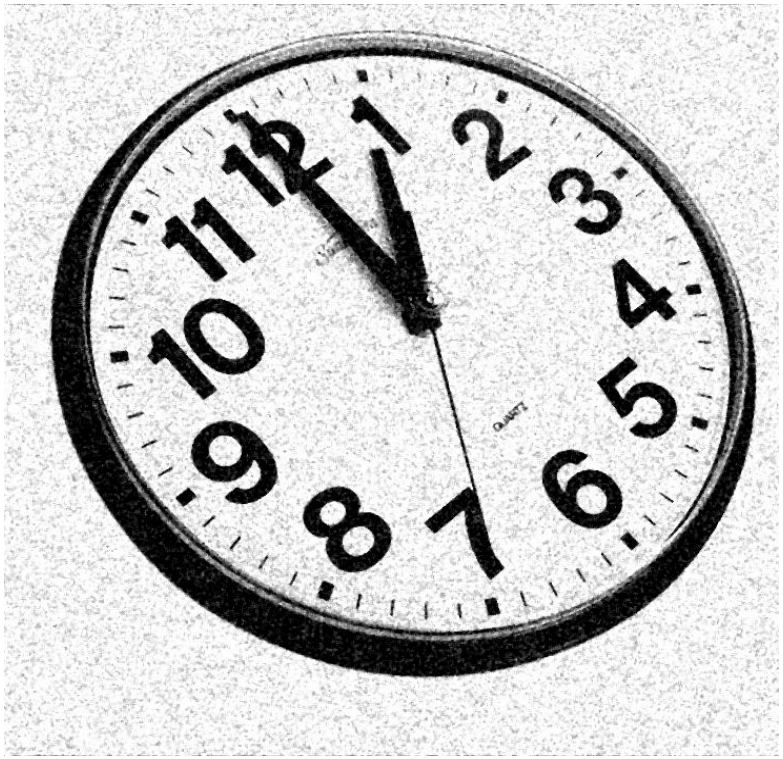


Noisy

# Mean-Median Filter - comparison

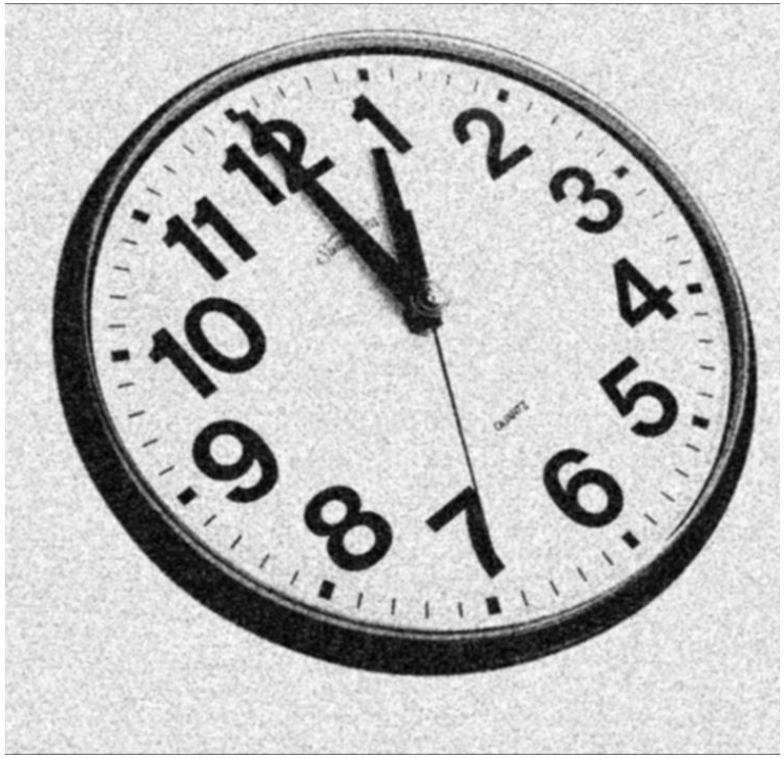


3x3-blur x 1

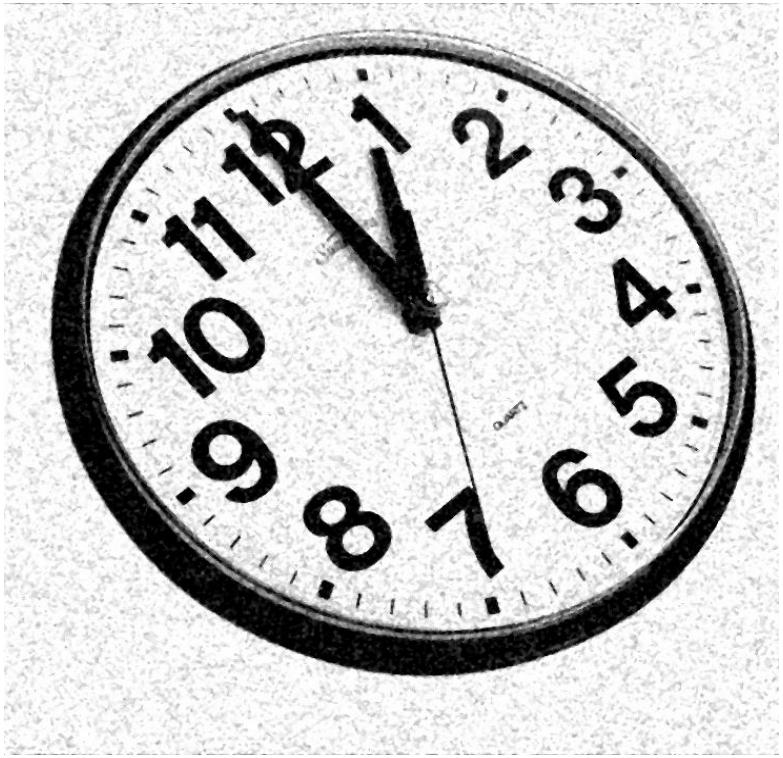


3x3-median x 1

# Mean-Median Filter - comparison

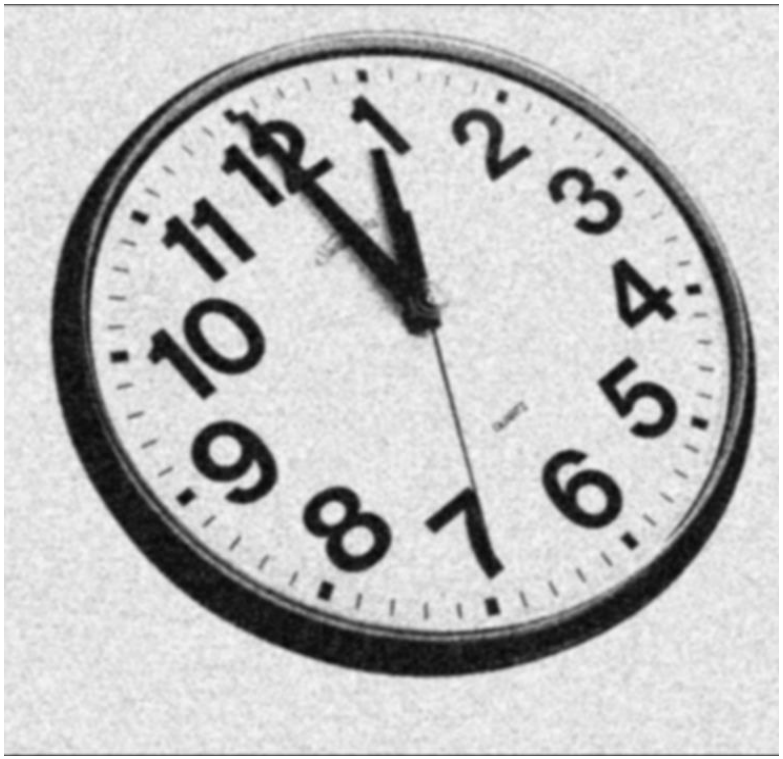


3x3-blur x 2

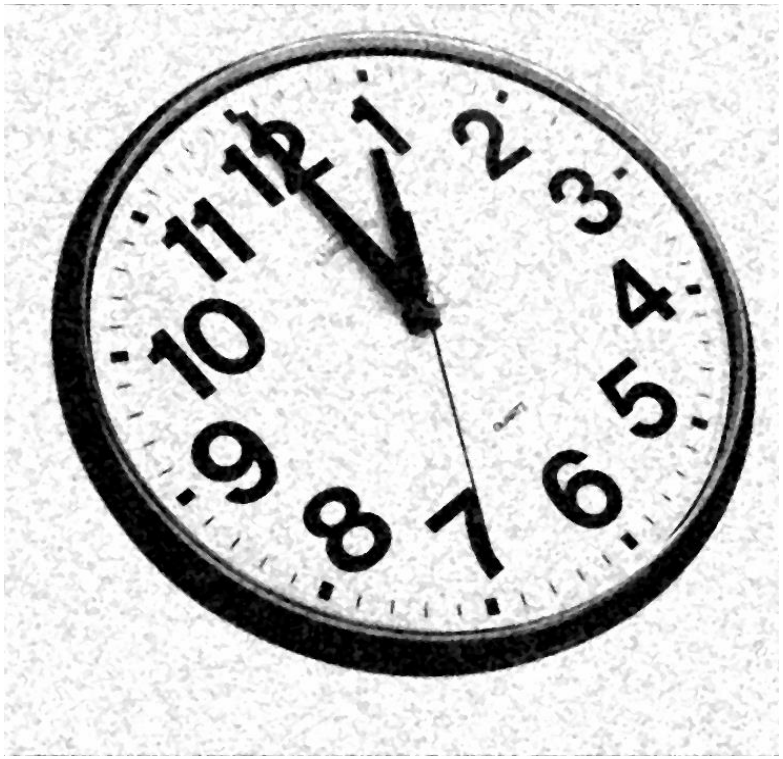


3x3-median x 2

# Mean-Median Filter - comparison



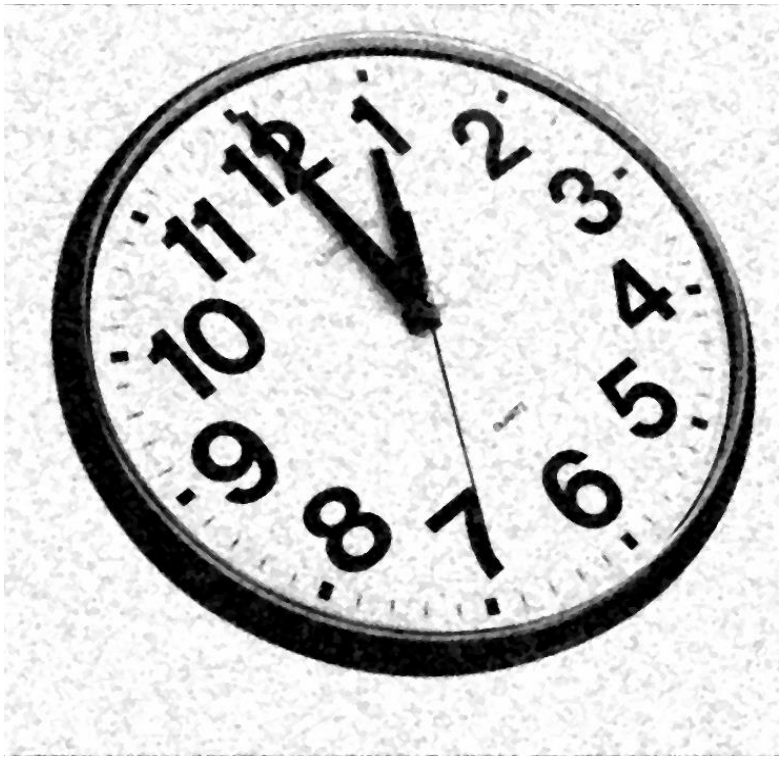
3x3-blur x 5



3x3-median x 5

# Mean-Median Filter - comparison

3x3-blur x 10



3x3-median x 10

# Order Filters

NUST

## Midpoint filter

– Average of the maximum and minimum within the window.

$$\text{Midpoint} = \frac{I_1 + I_{N^2}}{2}$$

# Order Filters

Image with Gaussian noise

Result of midpoint filter
Mask size = 3

# Order Filters

Image with uniform noise

Result of midpoint filter
Mask size = 3

# Order Filters

- Alpha-trimmed mean filter
  - The average of the pixel values within the window, but with some endpoint-ranked values excluded.

**Alpha-trimmed mean =** $$\frac{1}{N^2 - 2T} \sum_{i=T+1}^{N^2-T} I_i$$

  - T is the number of pixels excluded at each end of the ordered set
- The alpha-trimmed mean filter ranges from a mean to median filter, depending on the value selected for the T parameter.

If T = 0, → mean filter.
If T = $(N^2 - 1) / 2$, → median filter.

# Order Filters

Image with Gaussian and salt pepper noise.

Result of alpha-trimmed mean filter
Mask size = 3
Trim size = 0

# Order Filters

Result of alpha-trimmed mean filter
Mask size = 3
Trim size = 1

Result of alpha-trimmed mean filter
Mask size = 3
Trim size = 4

# Mean Filters

- The mean filters function by finding some form of an average within the *N*x*N* window.

- The most basic of these filters is the arithmetic mean filter.

  - This filter mitigates the noise effect, but at the same time tend to blur the image.
  - The blurring effect is not desirable, and therefore other mean filters are designed to minimize this loss of detail information.

# Mean Filters

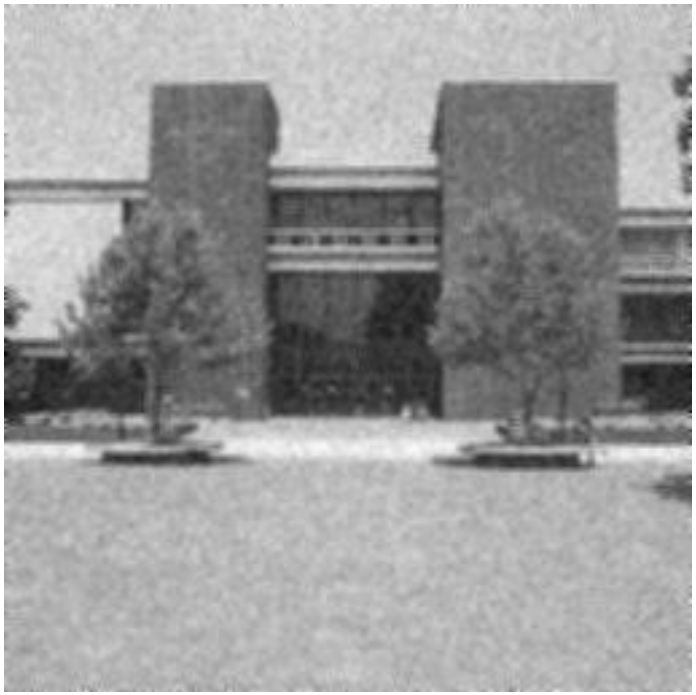
- Arithmetic mean filter
  - Find the arithmetic average of the pixel values in the window.

**Arithmetic Mean =** $$\frac{1}{N^2}\sum_{(r,c)\in w} d(r,c)$$

  - Smooth out local variations in an image.
  - Tend to blur the image.

# Mean Filters

Image with Gaussian noise



Result of arithmetic mean filter
Mask size = 3

# Mean Filters

Result of arithmetic mean filter
Mask size = 5

Result of arithmetic mean filter
Mask size = 9

# Mean Filters

- Geometric mean filter

**Geometric Mean =** $$\prod_{(r,c)\in w}[g(r,c)]^{\frac{1}{N^2}}$$

  - Retains detail better than arithmetic mean filter.
  - Drawback?

  - Ineffective in the presence of pepper noise (if very low values present in the window, the equation will return a very small number).

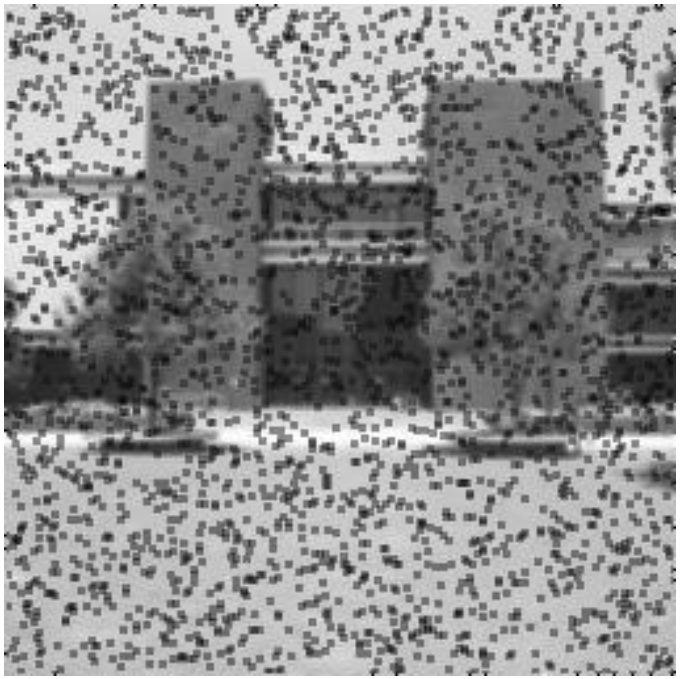
# Mean Filters

Image with salt noise
Probability=.04

Result of geometric filter
Mask size = 3

# Mean Filters

Image with pepper noise
Probability = .04



Result of geometric filter
Mask size = 3

# Mean Filters

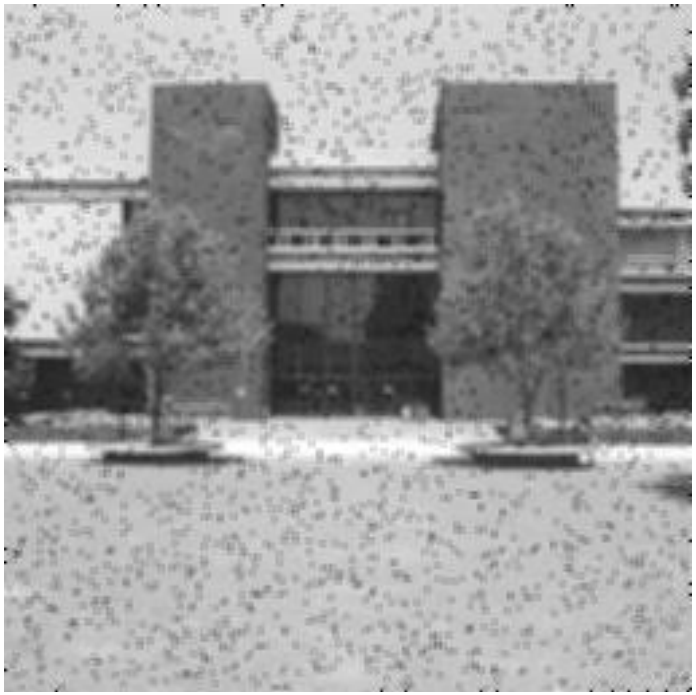
- Contra-harmonic mean filter

$$\text{Contra-Harmonic Mean} = \frac{\sum_{(r,c)\in w} g(r,c)^{R+1}}{\sum_{(r,c)\in w} g(r,c)^{R}}$$

  - Works for salt OR pepper noise, depending on the filter order R.
  - Positive R → Eliminate pepper-type noise.
  - Negative R → Eliminate salt-type noise.

# Mean Filters

Image with pepper noise
Probability = .04



Result of contra-harmonic filter
Mask size = 3; order = 0

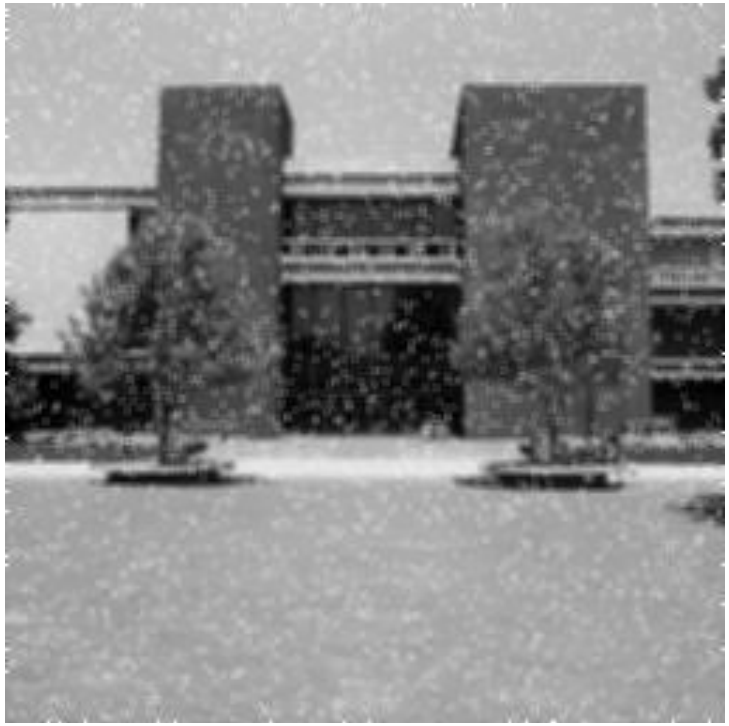
# Mean Filters

Result of contra harmonic filter
Mask size = 3; order = +1

Result of contra harmonic filter
Mask size = 3; order = +5

# Mean Filters

Image with salt noise
Probability = .04



Result of contra-harmonic filter
Mask size = 3; order = 0

# Mean Filters

Result of contra-harmonic filter
Mask size = 3; order = -1

Result of contra-harmonic filter
Mask size = 3; order = -5

# Mean Filters

- Harmonic mean filter

$$\textbf{Harmonic Mean} = \frac{N^2}{\sum_{(r,c)\in w} \frac{1}{g(r,c)}}$$

  - By using the Value of **R = -1** in the contra-harmonic filter.

# Mean Filters

Image with pepper noise
Probability = .04

Result of harmonic filter
Mask size = 3

# Mean Filters

Image with salt noise
Probability=.04

Result of harmonic filter
Mask size = 3

# End
# Spatial Filtering