Digital Image Processing (DIP)
Lec-(4-5)

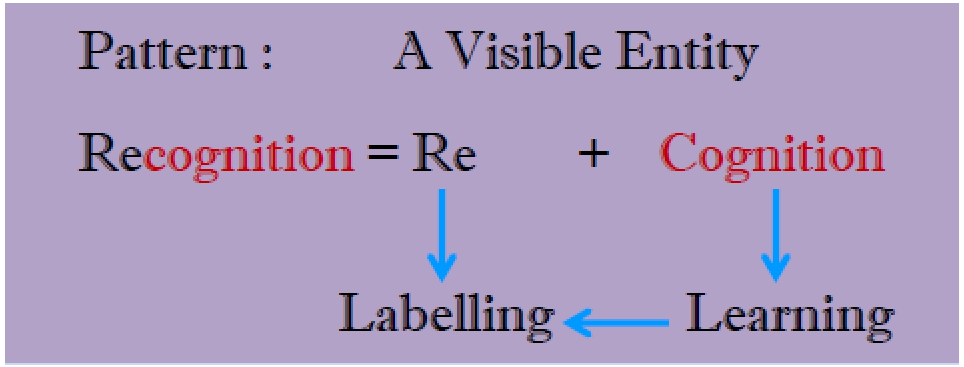
By

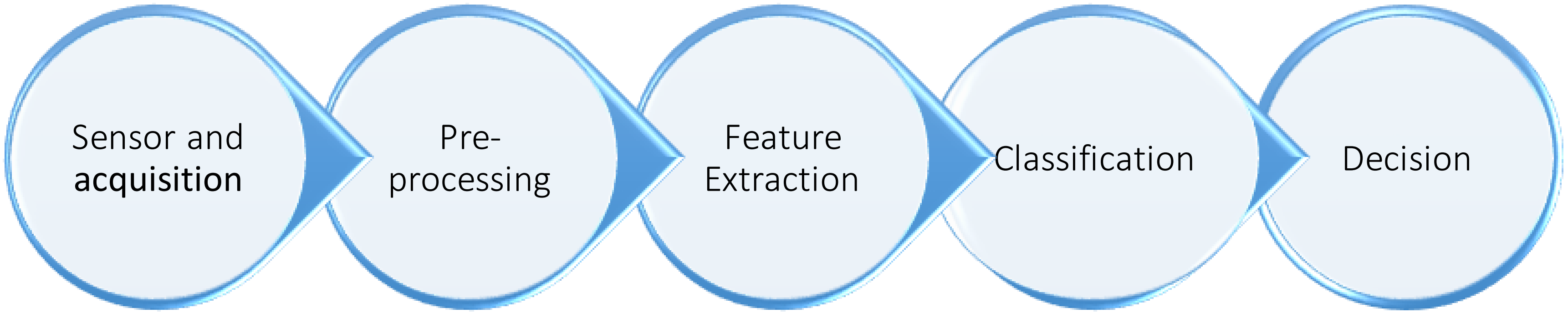
Dr. Akram Alsubari

e-mail:- akram.alsubari87@gmail.com

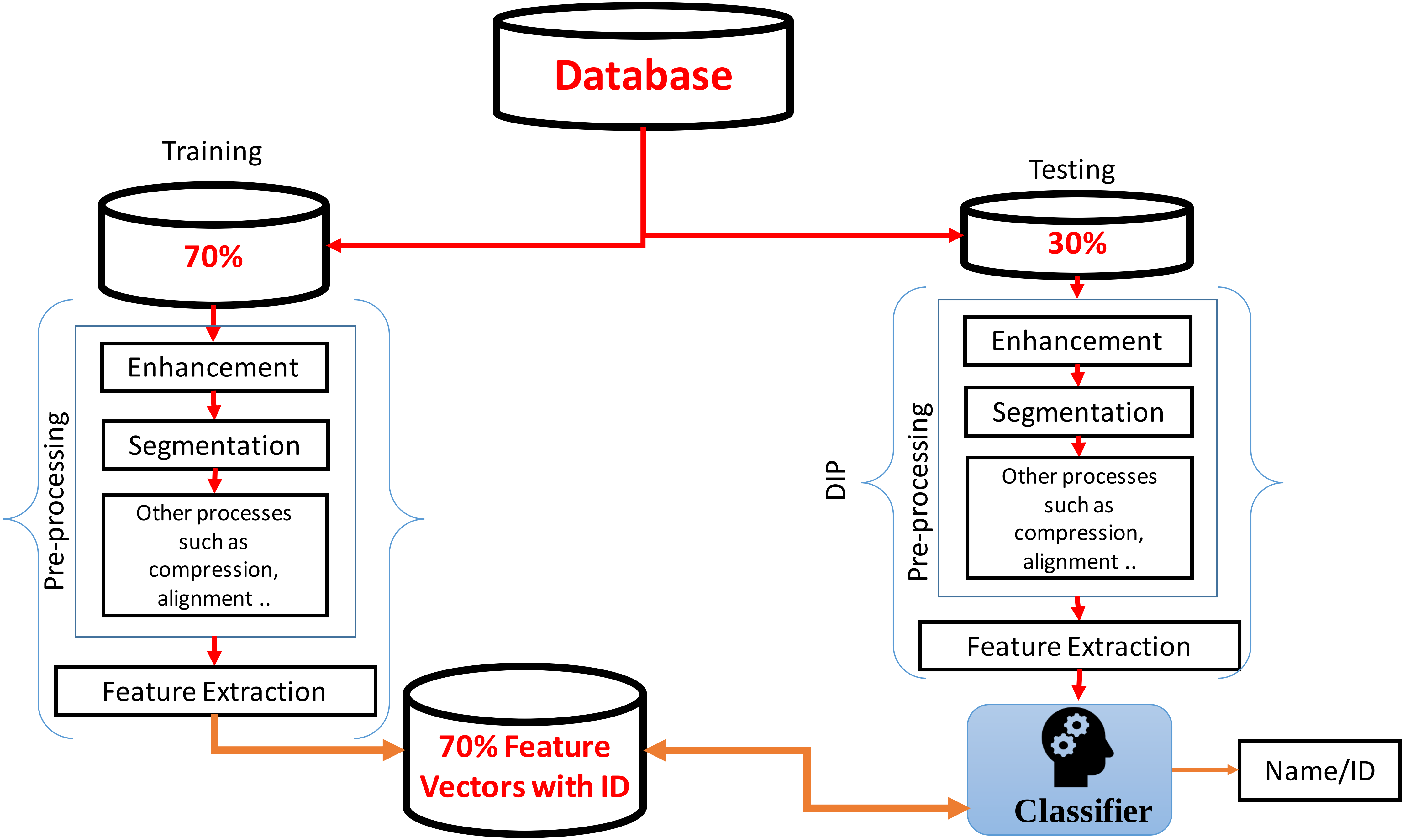
Pervious Lecture

- Pattern Recognition
- Pattern
  - **Spatial patterns** (patterns are located in space)
  - **Temporal patterns** (Distributed in time)
  - **Abstract patterns** (patterns are distributed neither in space nor time)





نفرض انها 100 صوره لعشره اشخاص وكل شخص له عشره صور



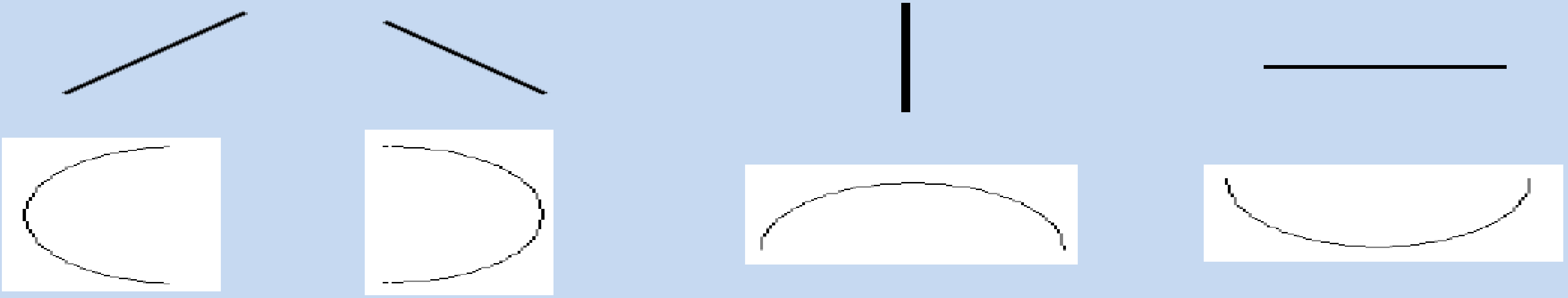
Example : Feature Extraction

Objects :

A B C D E F

Feature?...

Line and Curve Segments



Object	0	45	90	145	Top semi circle	Bottom Semi circle	Left Semi circle	Right Semi circle
A	1	1	0	1	0	0	0	0
B	0	0	1	0	0	0	0	2
C	0	0	0	0	0	0	1	0
D	0	0	1	0	0	0	0	1
E	3	0	1	0	0	0	0	0
F	2	0	1	0	0	0	0	0

Feature Vector

Label/class

Face Recognition in MATLAB

```
clear
imgcoll=imageSet('24_ORL','recursive');
[training, test]=partition(imgcoll,[0.5 0.5]);% split the database into traning sets and testing sets
K=1;
for i=1:size(training,2 )
    for j=1:training(i).Count
        trainingFeatures(K,:)= extractHOGFeatures(read(training(i),j));
        traininglabel{K}=training(i).Description;
        K=K+1;
    end
end
 faceclssifier= fitcknn(trainingFeatures,traininglabel);
 cn=0;
 for i=1:size(test,2)
     for j=1:test(i).Count
         InserFace=read(test(i),j);
         FaceFeatures=extractHOGFeatures(InserFace);
         personlabel=predict(faceclssifier,FaceFeatures);
         if strcmp(personlabel,test(i).Description)
             cn=cn+1;
         end
      end
 end
  Recognition_rate=(cn/(i*j))*100
```

Image Digitization

Image Sensing and Acquisition

- An image captured by **sensor** is expressed as continuous function $f(x, y)$ of two co-ordinates in the plane.
- **Image Digitization**
  - **Image sampling** is to convert/sample continuous function $f(x, y)$ into matrix with M rows and N columns
  - **Image quantization** is to assign to each continuous sample an integer value

Sampling **Quantization**

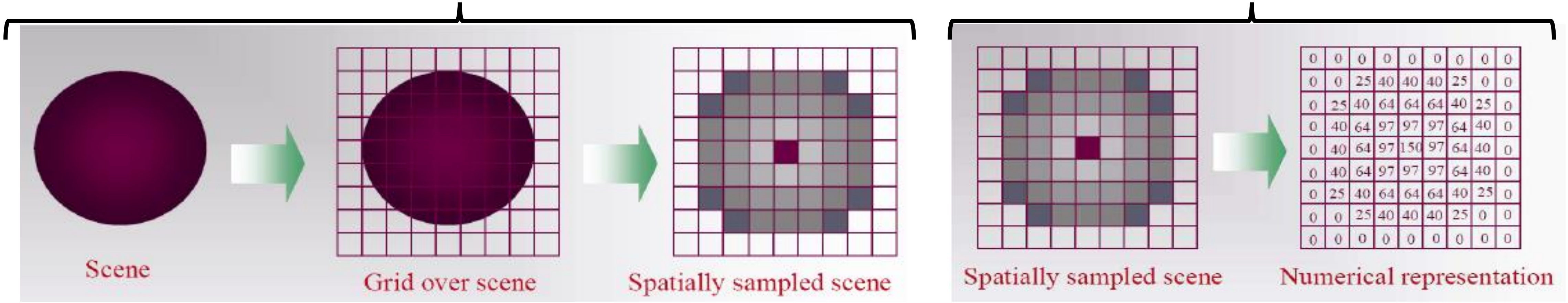
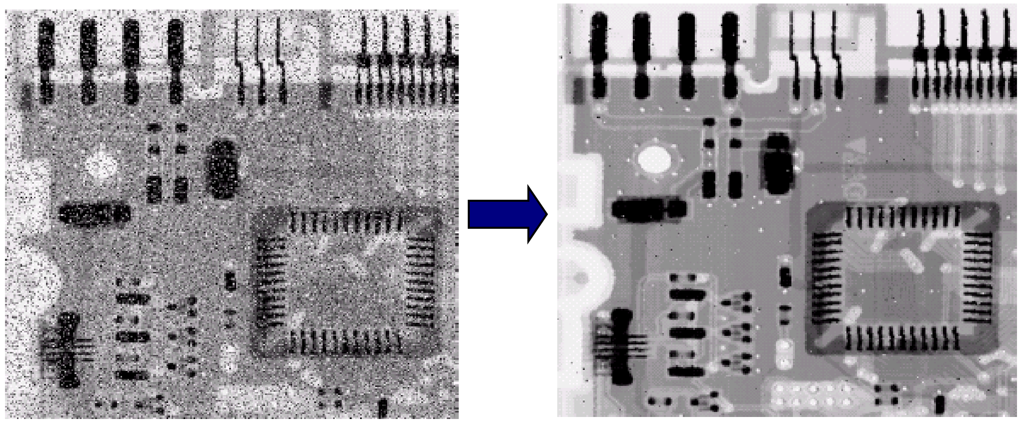


Image Enhancement

Enhancement

Image enhancement is the process of making images more useful

- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing



Enhancement

There are two broad categories of image enhancement techniques

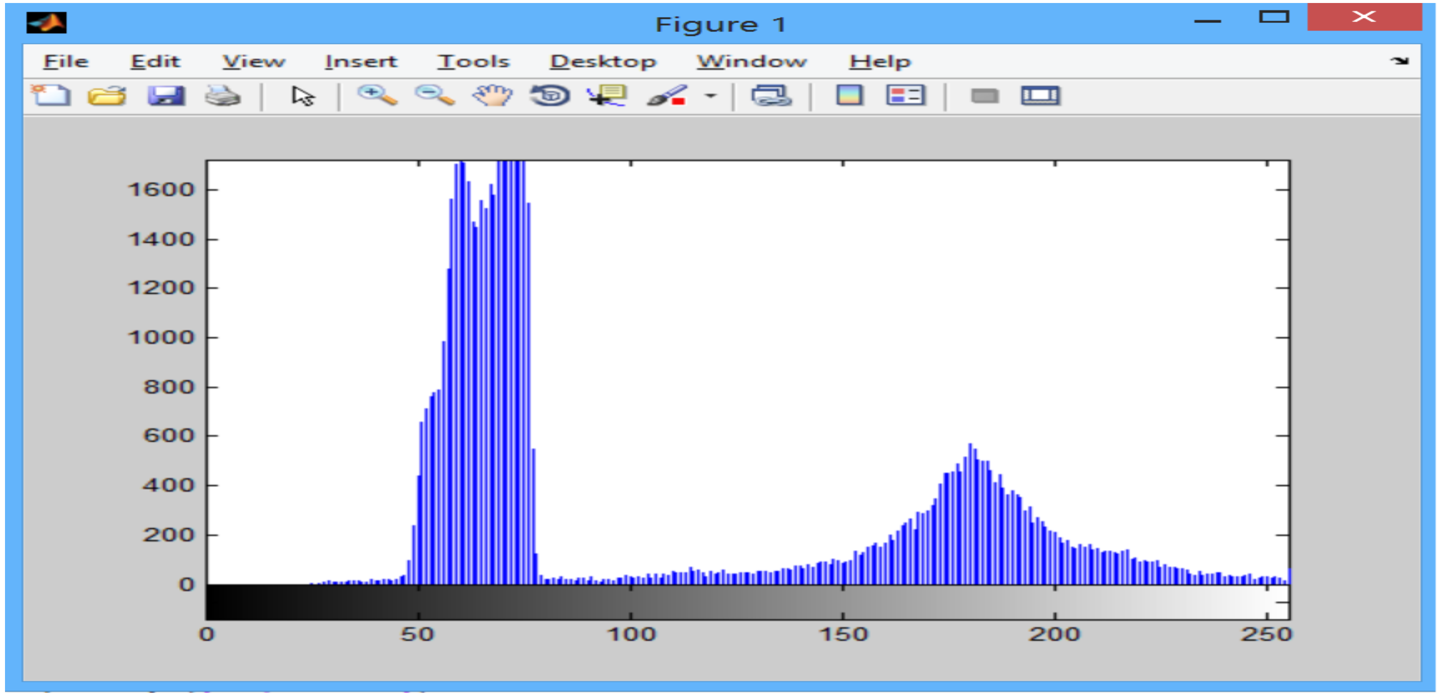
- **Spatial domain** techniques
  - Direct manipulation of image pixels
- **Frequency domain** techniques
  - Manipulation of Fourier transform or wavelet transform of an image

Image Histogram

- The histogram of an image shows us the distribution of grey levels in the image

f = imread ('coins.png');

figure, imhist (f);

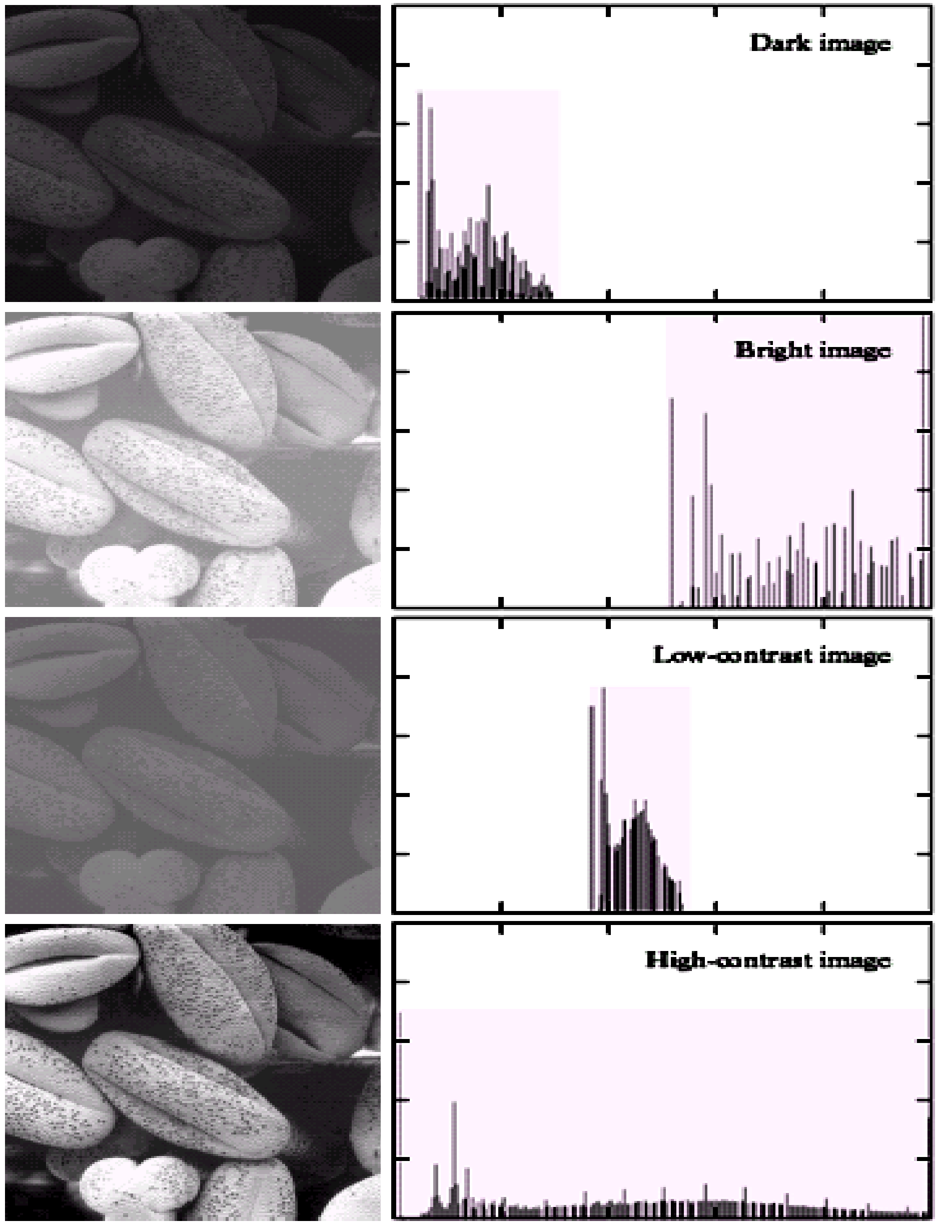


Histogram Examples

A selection of images and their histograms

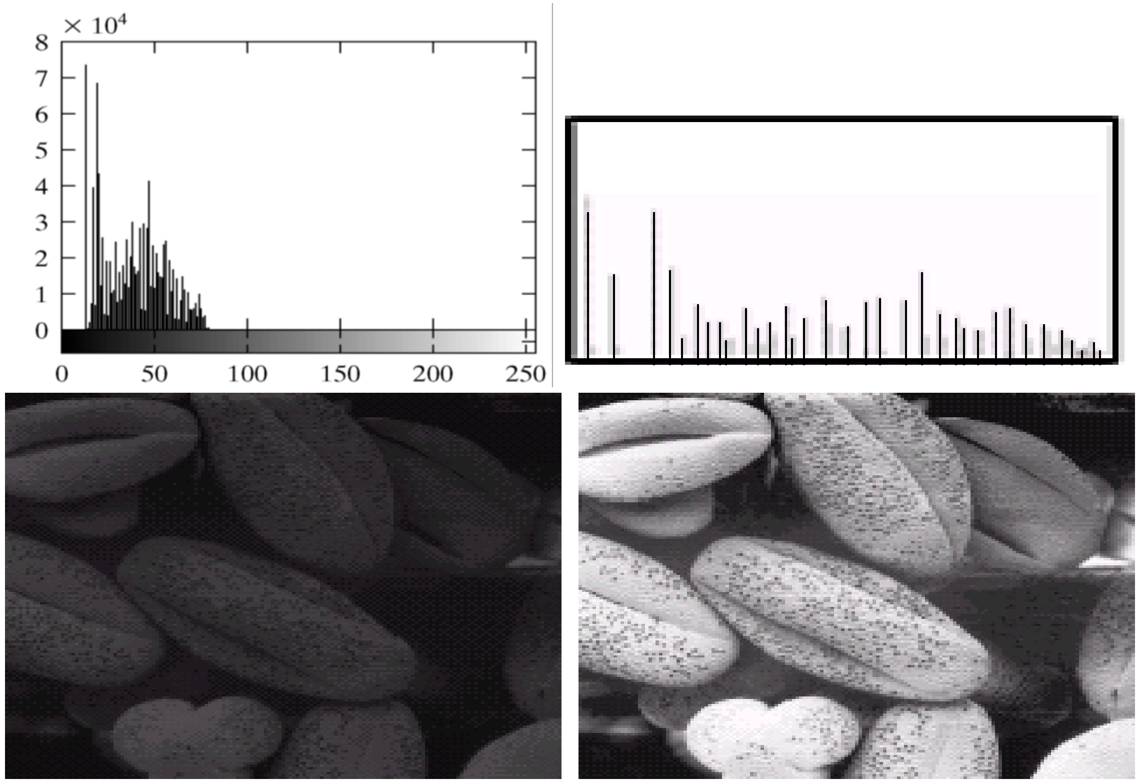
Notice the relationships between the images and their histograms

Note that the high contrast image has the most evenly spaced histogram



Histogram Equalisation

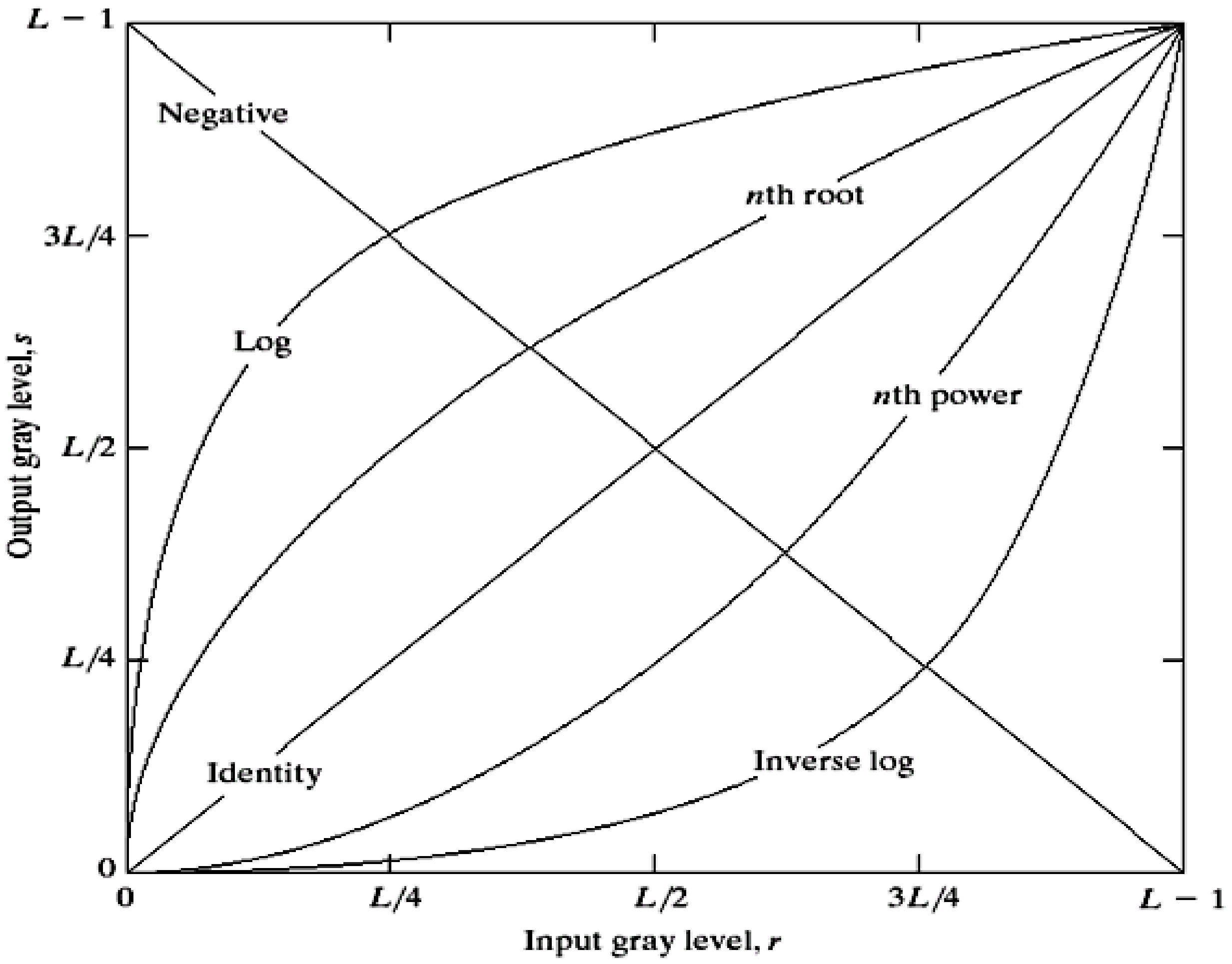
- Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images



Basic Grey Level Transformation

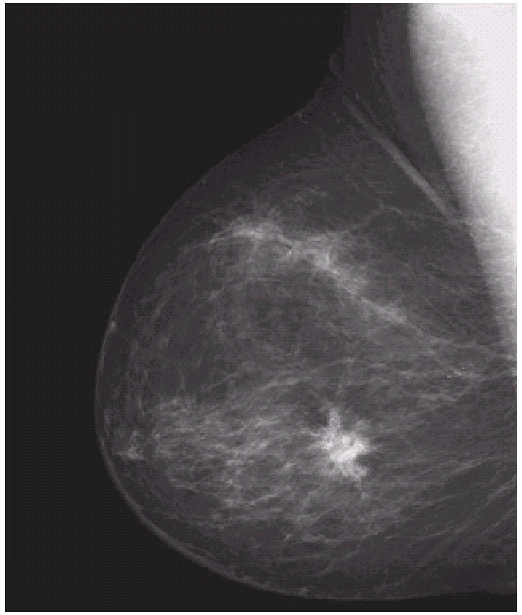
- Basic Grey Level Transformation

  1) Linear (negative and identity)
  2) Logarithmic
  3) Power – law



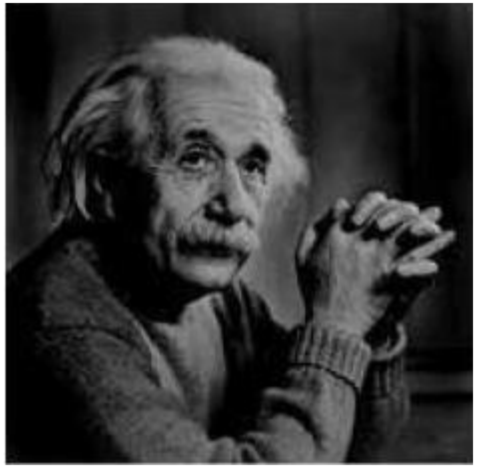
Negative Transformation

- $s = intensity_{max} - r$



Logarithmic Transformation

$$s = c \log(r + 1)$$

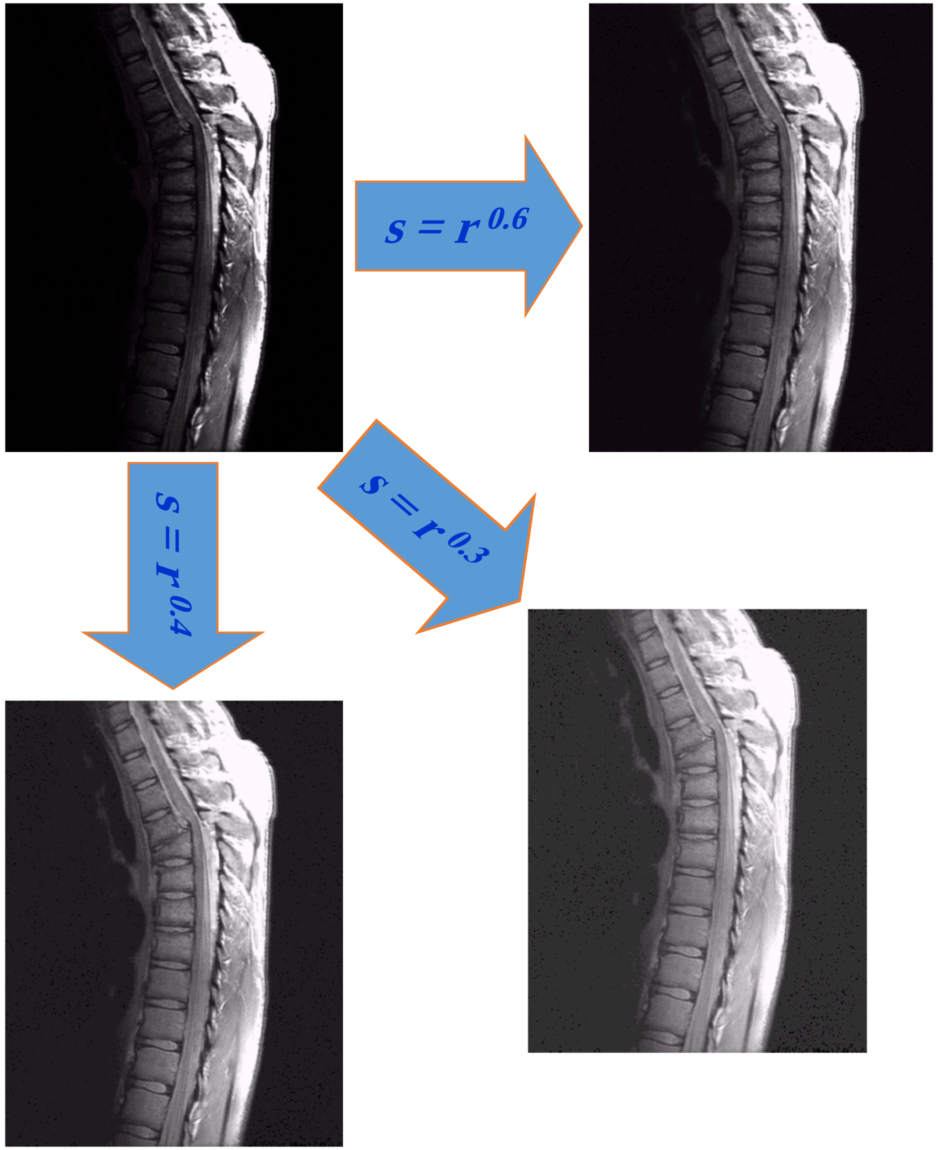


Log functions are particularly useful when the input grey level values may have an extremely large range of values

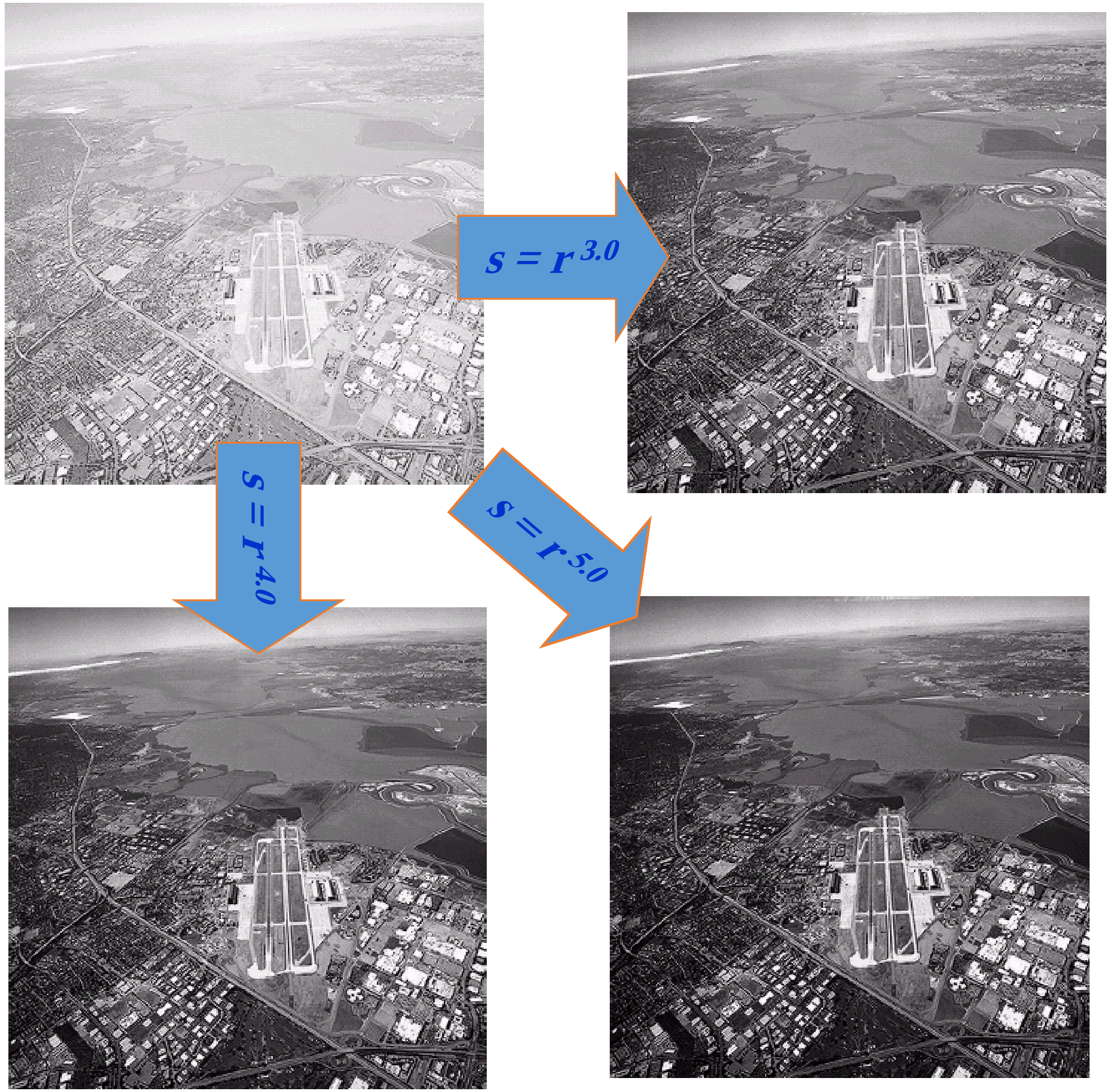
Power – law Transformation

s=cr^γ

Map a narrow range of dark input values into a wider range of output values or vice versa



Cont..



Brightness and Contrast

- **Brightness:**

intensity of light emit by a particular light source.

- **Contrast:**

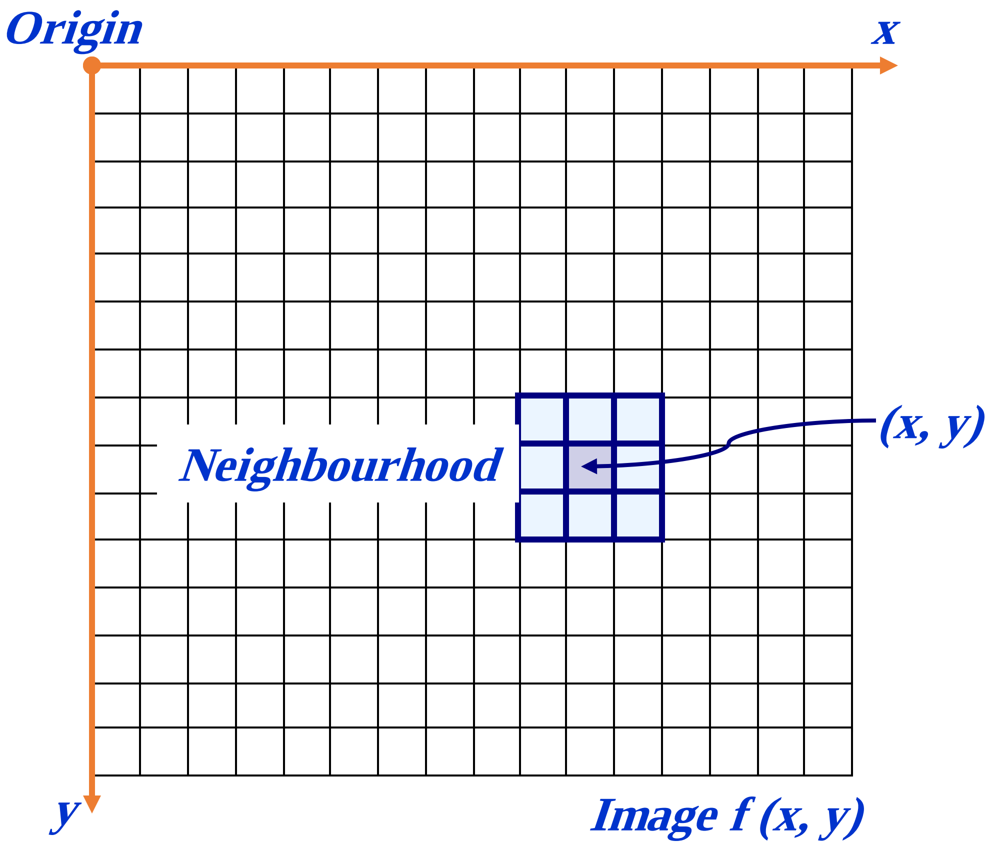
the difference between maximum and minimum pixel intensity in an image.

CODES

```
%% Read an 8 bit image and then apply different image enhancement techniques
clear; clc;
close all
Img= imread('pout.tif');
BI= imadjust(Img);        % Adjust image intensity values
BR= Img./2;               % Brightness Reduction by dividing each pixel by 2
BW= im2bw(Img);           % Convert Gray Image into Binary Image Based on Thresholding
BW2= Img<110;             % another way to converet gray image into binary image, where the Thresholding is 110
NI = imcomplement(Img);   % Negative of the Image
LT= log(double(Img));     % Log Transformation
DImg=double(Img);
a=1;gamma=2;
PLT= a*(DImg).^gamma; % Power Lower Transform
subplot(3,3,1);
imshow(Img); title ('Orignal Image')
subplot (3,3,2);
imshow(BI); title ('image Improvement')
subplot(3,3,3);
imshow(BR); title ('Brightness Reduction')
subplot(3,3,4);
imshow(BW); title ('Binary Image')
subplot(3,3,5);
imshow(BW2); title ('Thresholding=110')
subplot(3,3,6);
imshow(NI); title ('Negative of the Iamge')
subplot(3,3,7);
imshow (LT, []); title ('Log Transformation ')
subplot(3,3,8);
imshow(PLT, []); title ('Power Low Transform')
```

Neighbourhood Operations

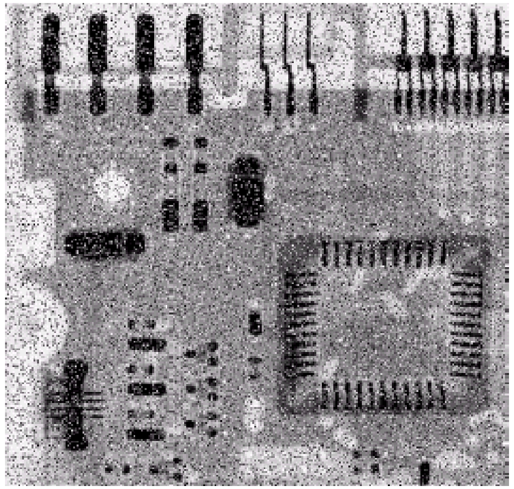
- Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations
- Neighbourhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible

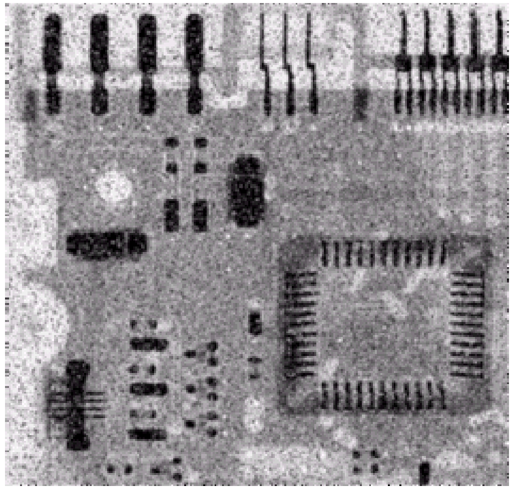


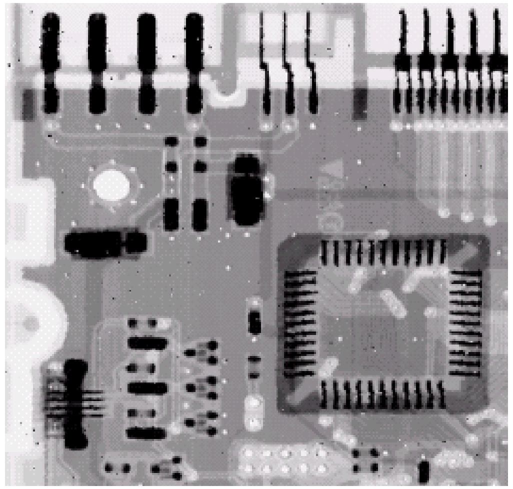
Simple Neighbourhood Operations

- 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

Averaging Filter Vs. Median Filter Example



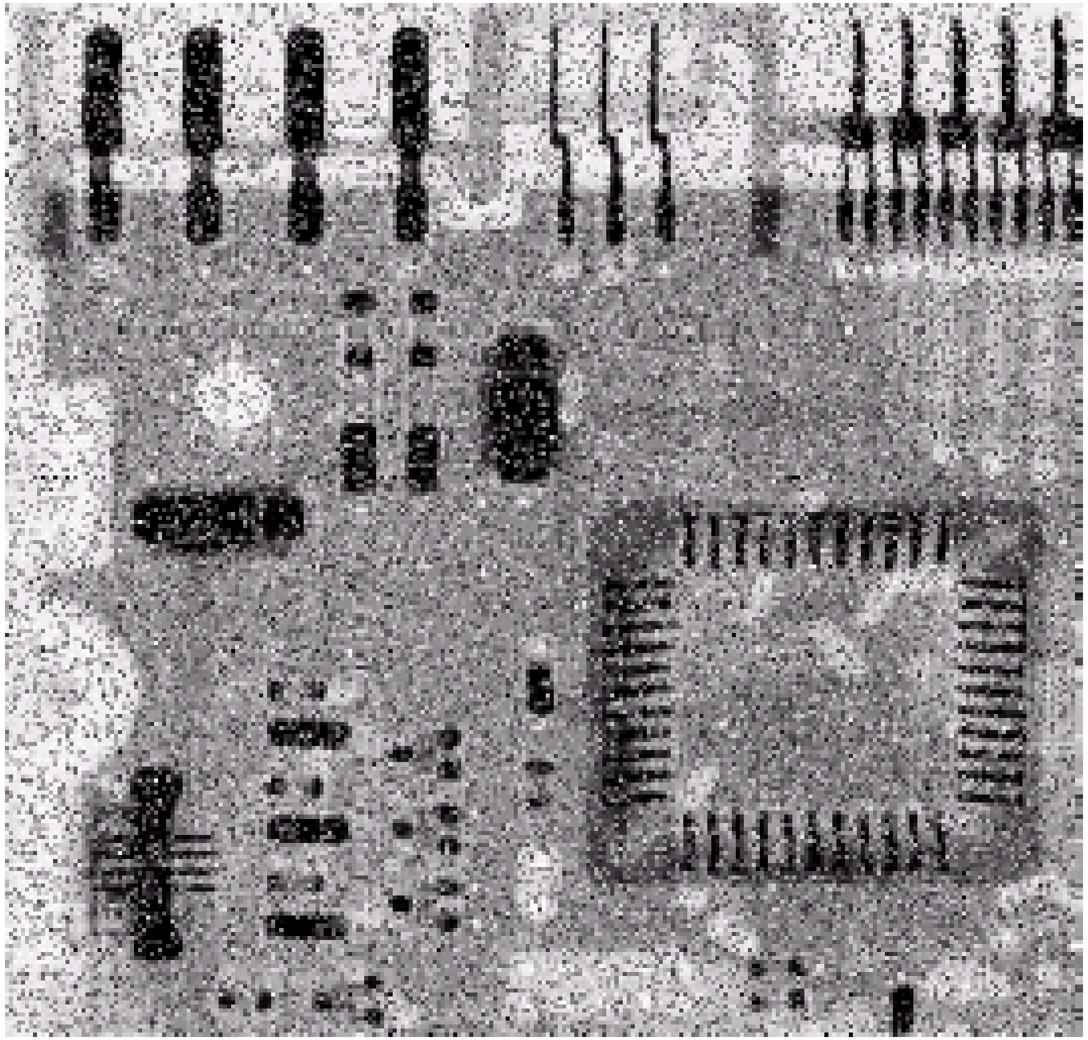




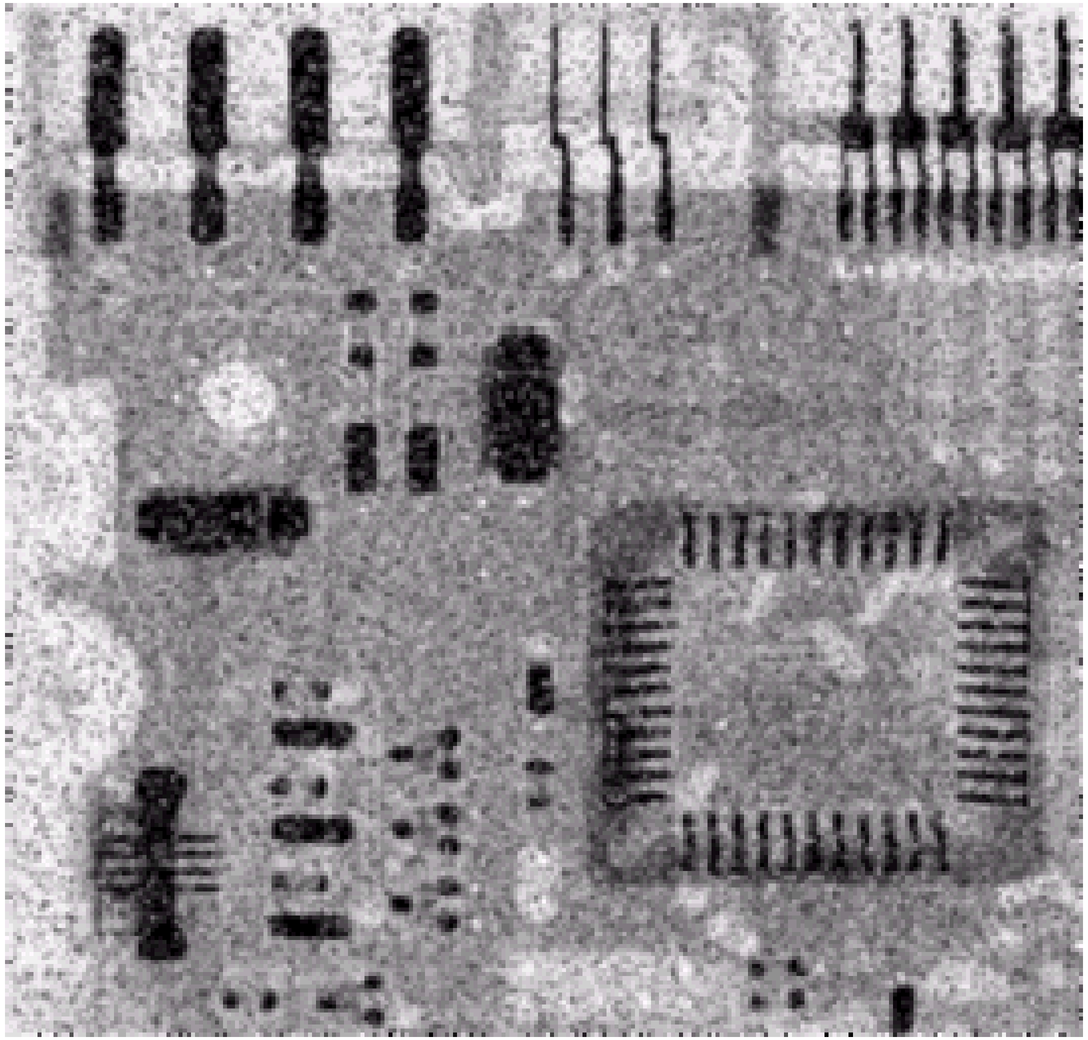
Original Image With Noise	Image After Averaging Filter	Image After Median Filter

- Filtering is often used to remove noise from images
- Sometimes a median filter works better than an averaging filter

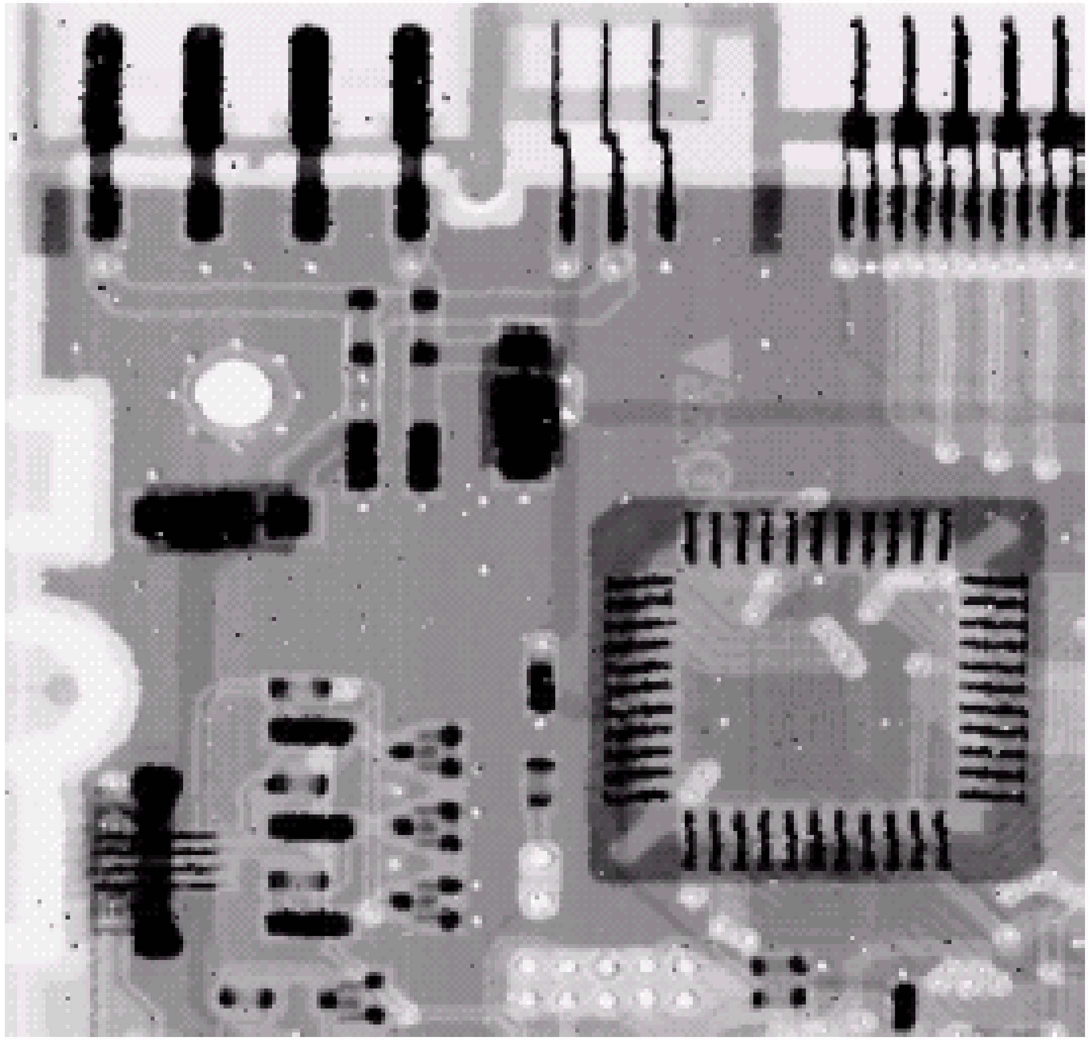
Averaging Filter Vs. Median Filter Example



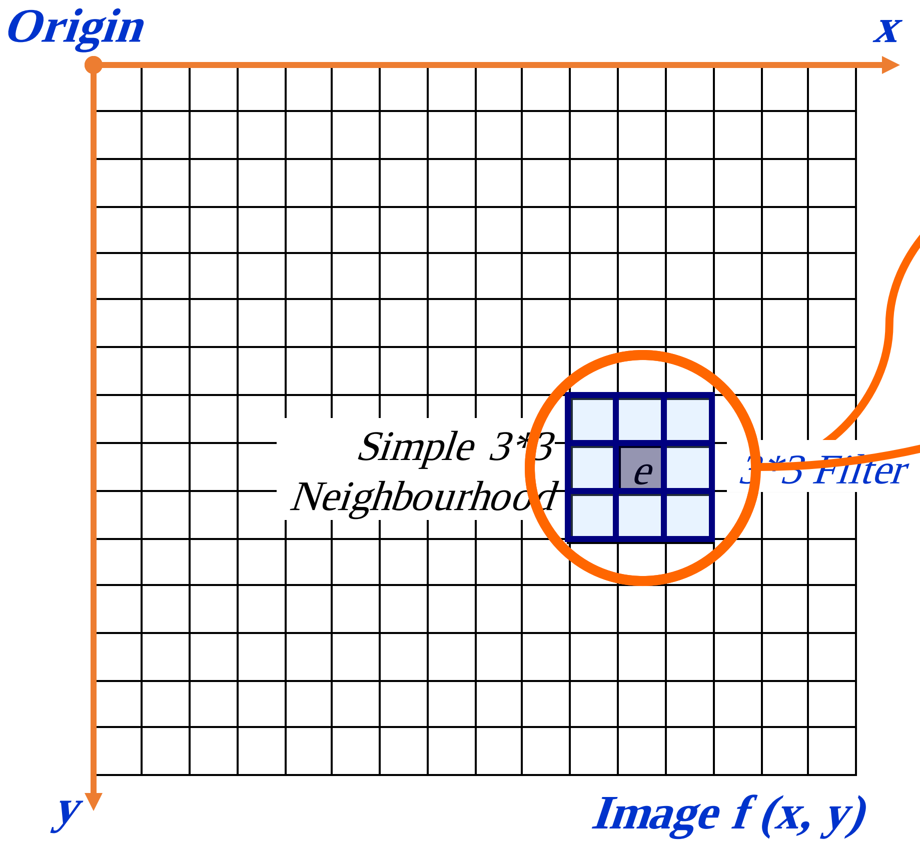
Averaging Filter Vs. Median Filter Example

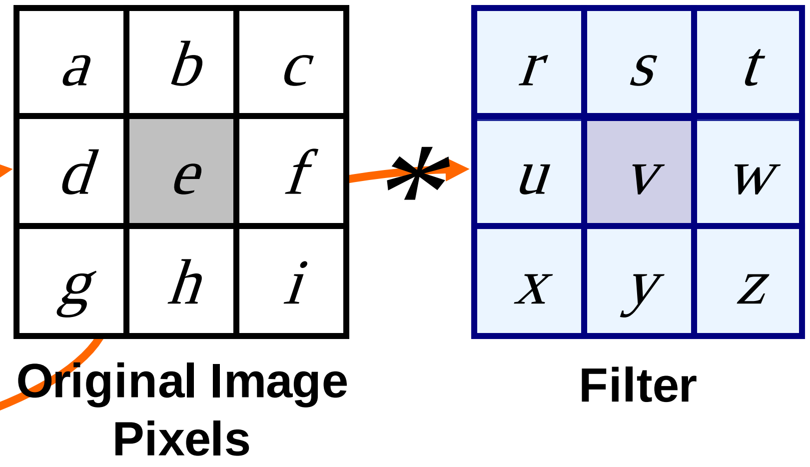


Averaging Filter Vs. Median Filter Example



The Spatial Filtering Process

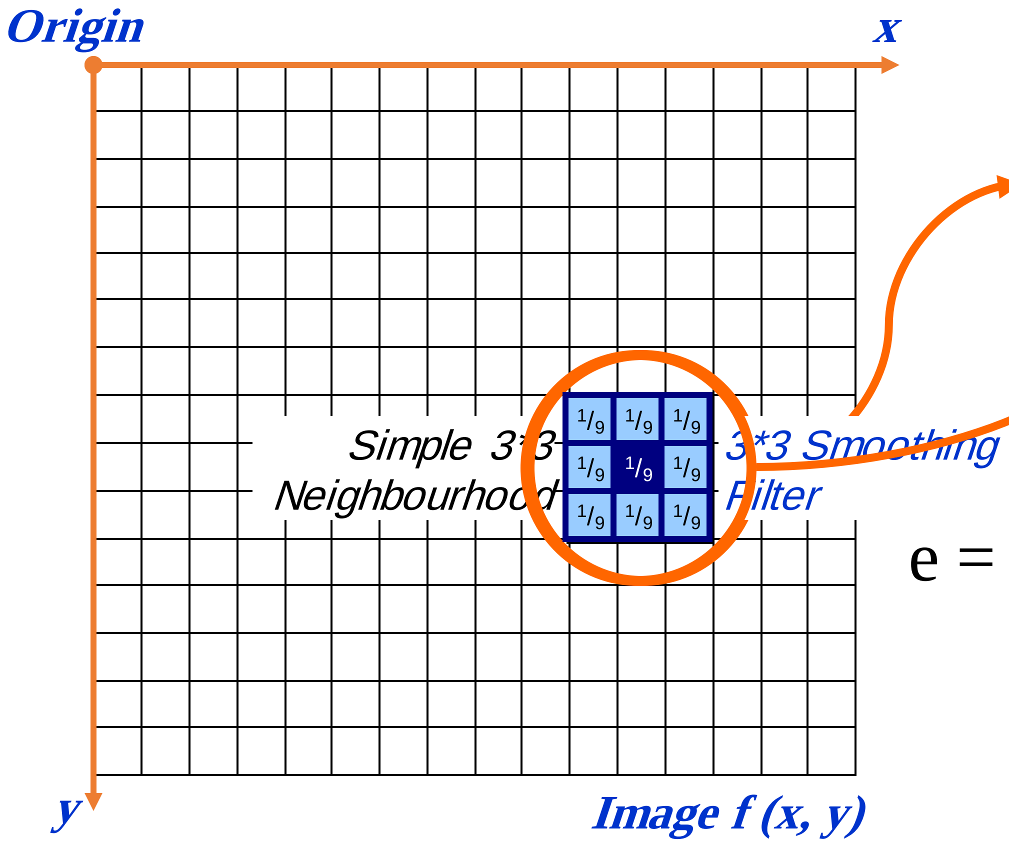




$$
\begin{aligned}
e_{processed} = \; & v*e + \\
& r*a + s*b + t*c + \\
& u*d + w*f + \\
& x*g + y*h + z*i
\end{aligned}
$$

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

Smoothing Spatial Filtering



104	100	108
99	106	98
95	90	85

Original Image Pixels

$*$

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Filter

$$
\begin{aligned}
e = {} & {}^1/_9*106 + \\
& {}^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\
& {}^1/_9*99 + {}^1/_9*98 + \\
& {}^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\
= {} & 98.3333
\end{aligned}
$$

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

Image Smoothing Example

- The image at the top left is an original image of size 500*500 pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes
    - 3, 5, 9, 15 and 35
- Notice how detail begins to disappear

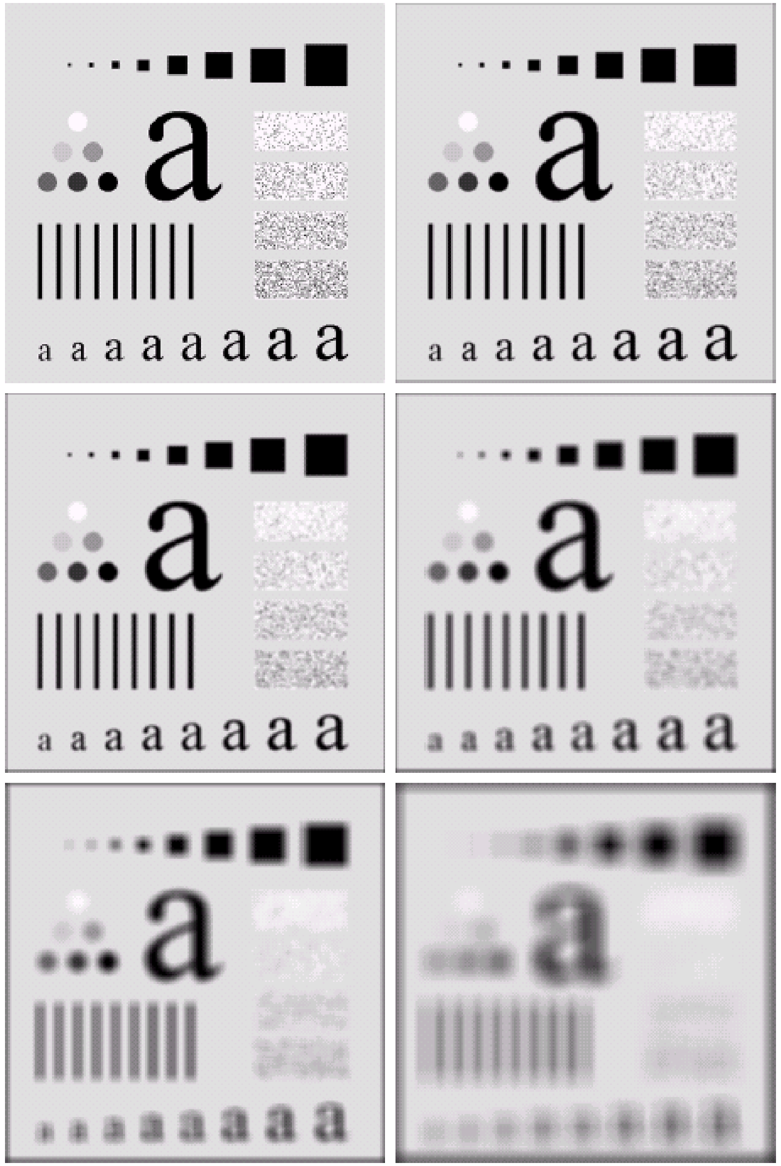


Image Smoothing Example

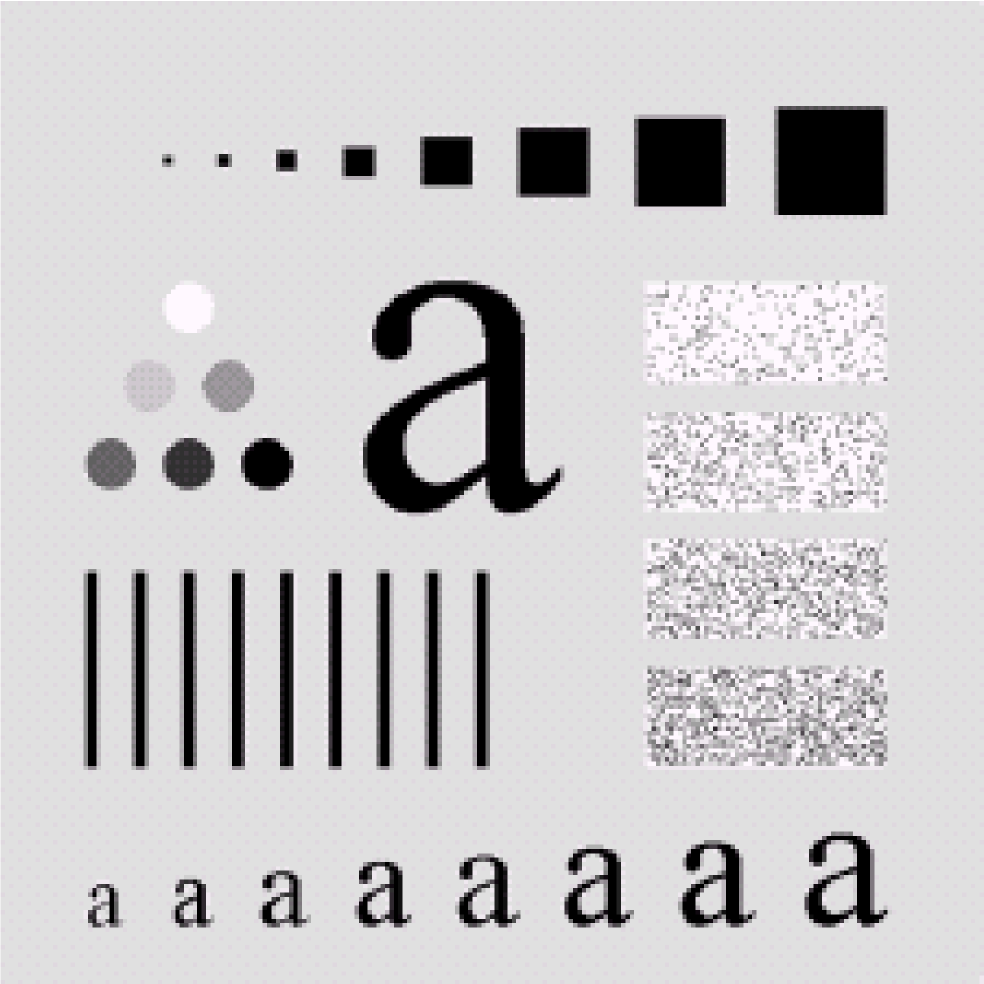


Image Smoothing Example

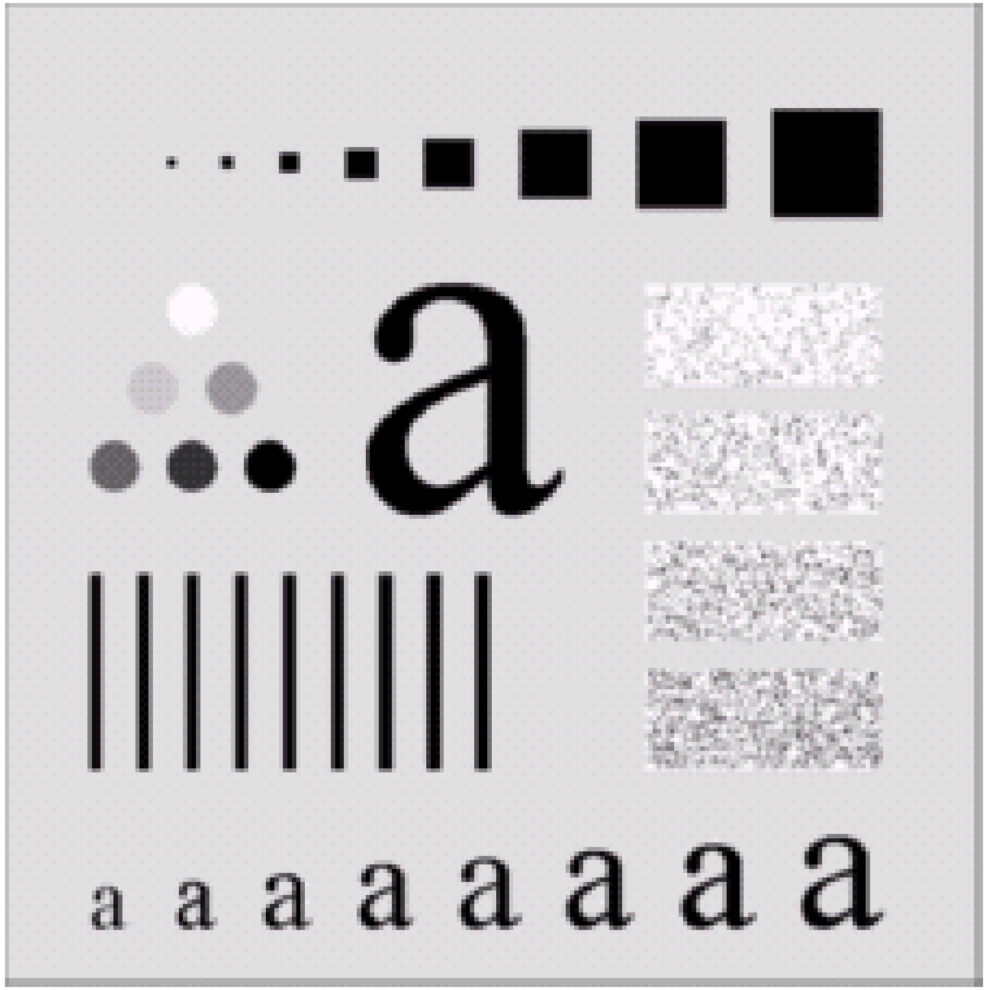


Image Smoothing Example

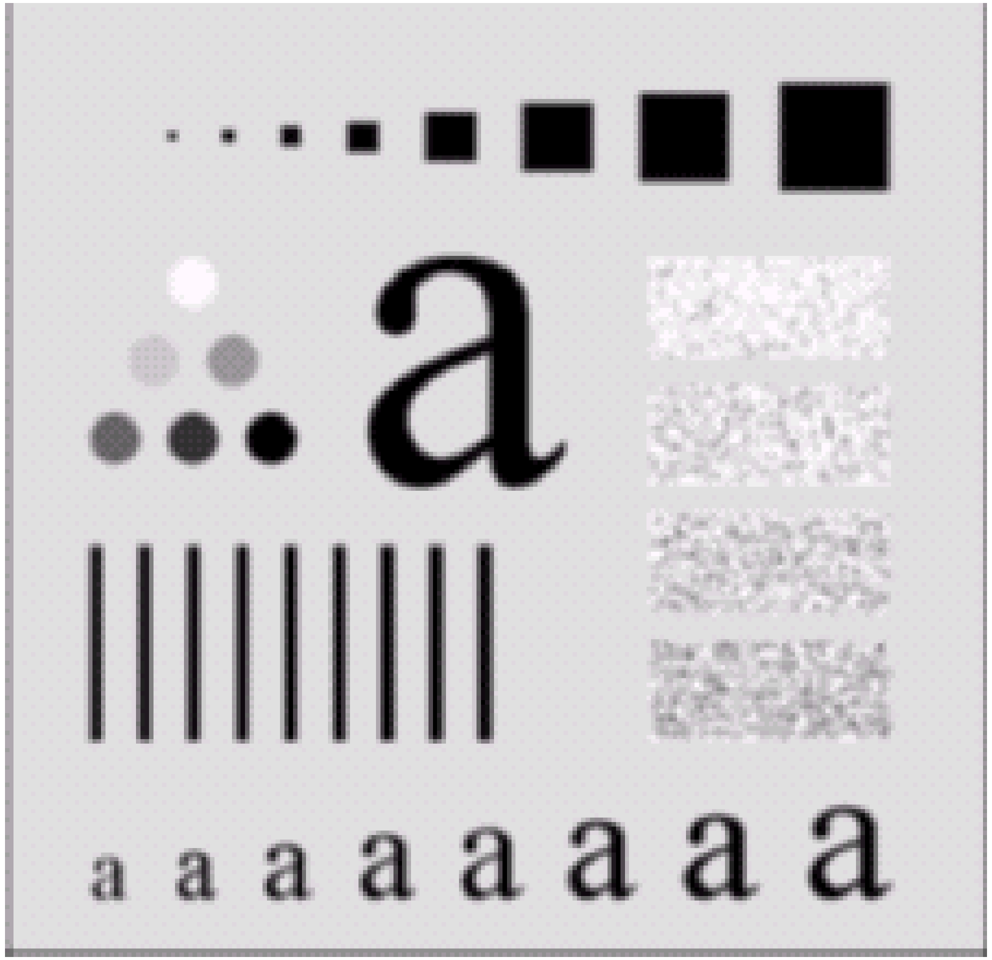


Image Smoothing Example

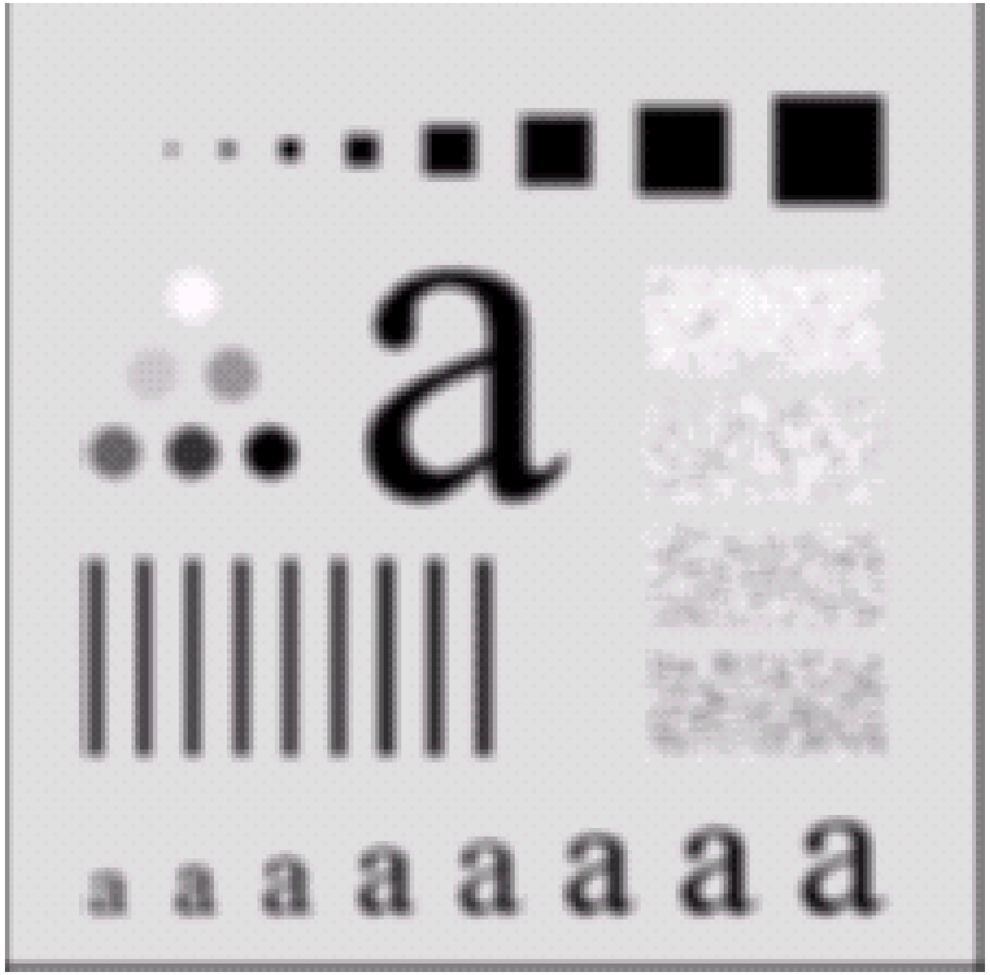


Image Smoothing Example

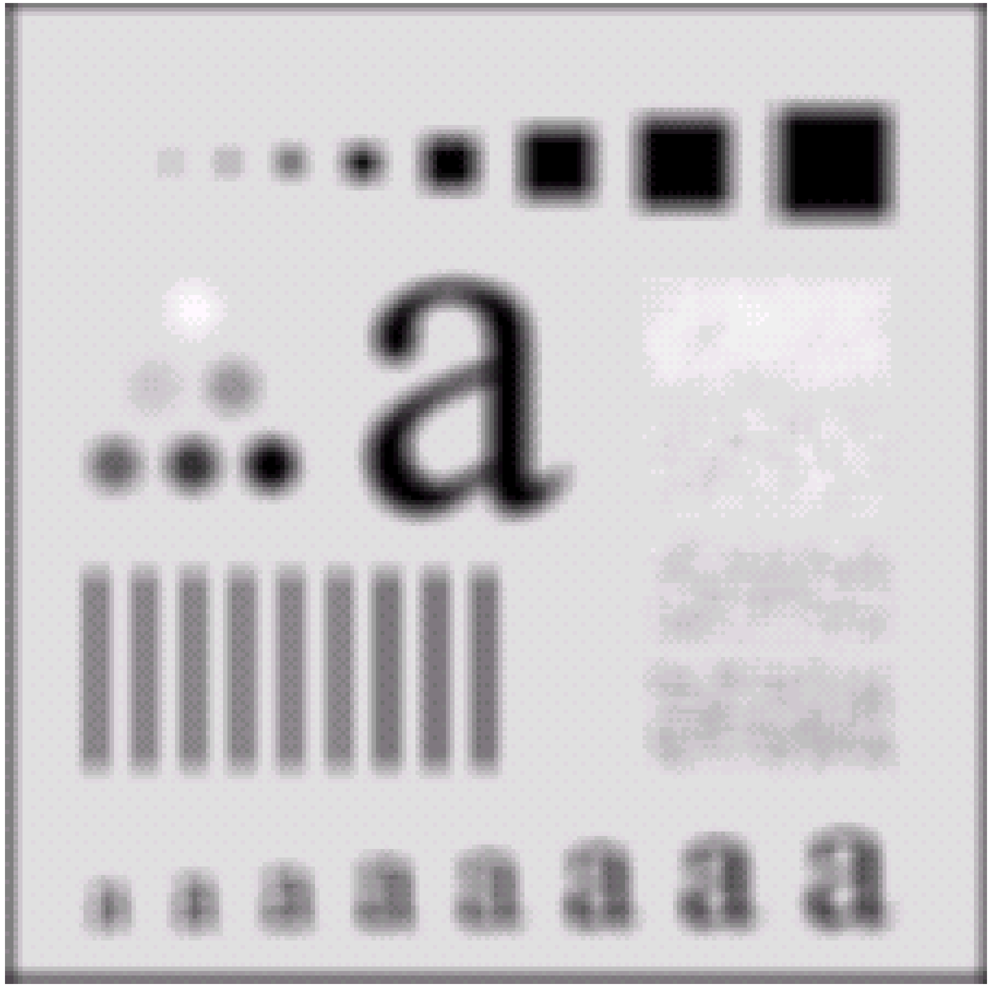
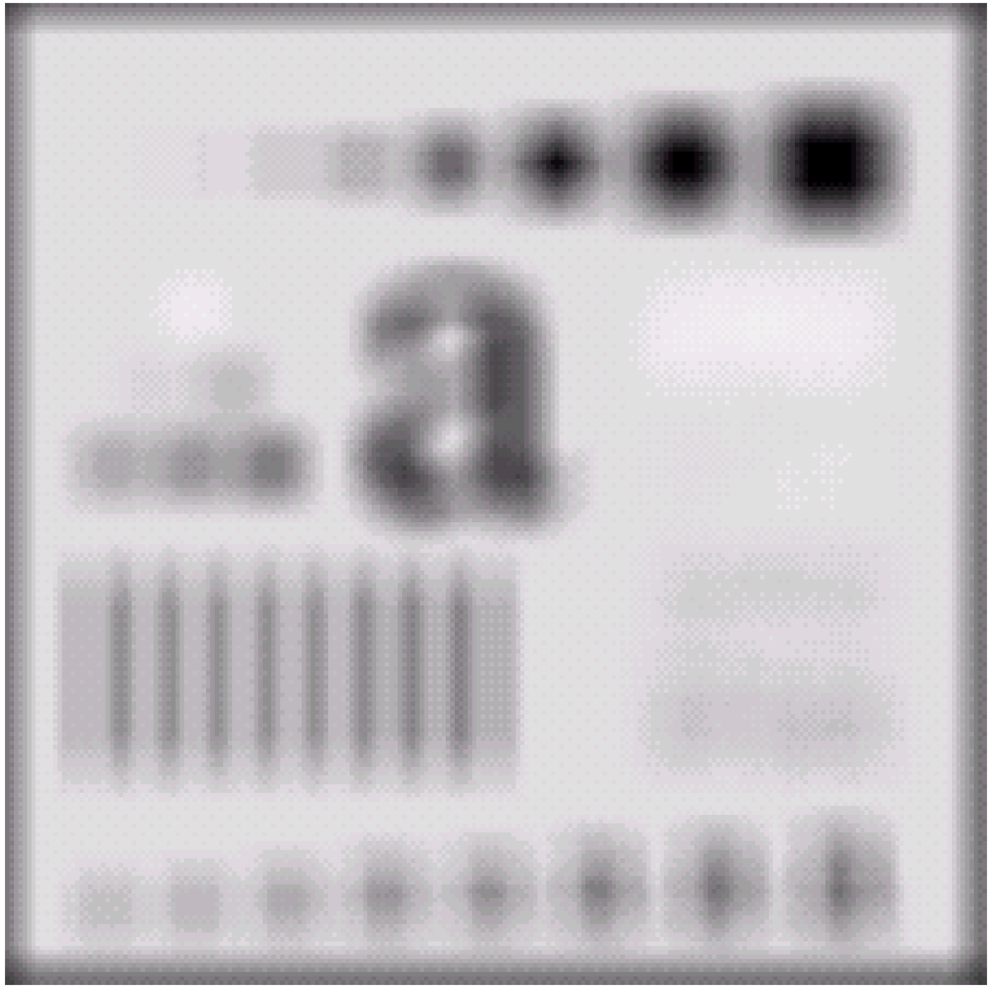


Image Smoothing Example



Weighted Smoothing Filters

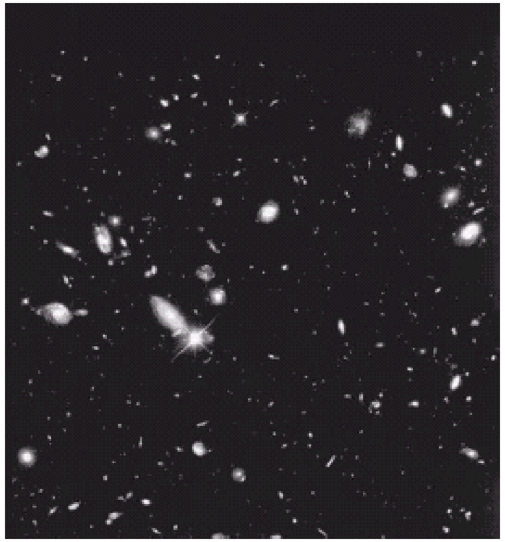
- More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function
  - Pixels closer to the central pixel are more important
  - Often referred to as a *weighted averaging*

$^1/_{16}$	$^2/_{16}$	$^1/_{16}$
$^2/_{16}$	$^4/_{16}$	$^2/_{16}$
$^1/_{16}$	$^2/_{16}$	$^1/_{16}$

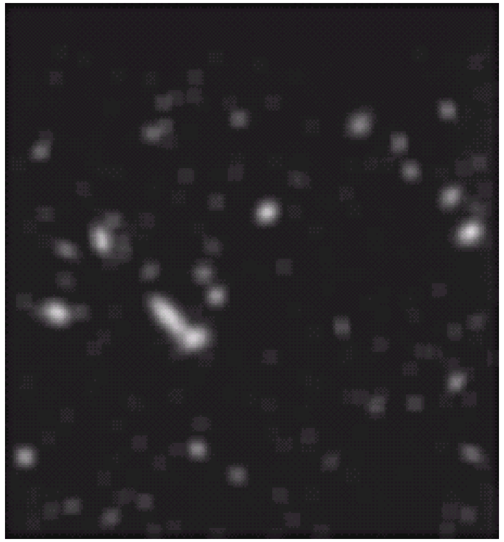
Weighted averaging filter

Another Smoothing Example

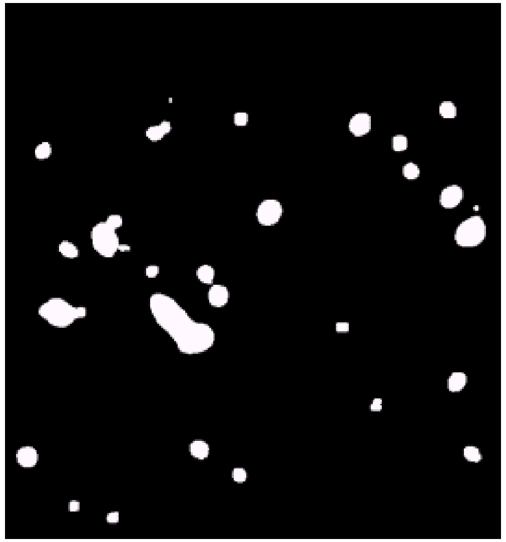
- By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image



Smoothed Image



Thresholded Image

Sharpening Spatial 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*

```
clc
clear all
close all
a=imread('horse.jpg');
%Addition of salt and pepper noise
b=imnoise(a,'salt & pepper',0.1);
%Defining the box and median filters
h1=1/9*ones(3,3);
h2=1/25*ones(5,5);
c1=conv2(b,h1,'same');
c2=conv2(b,h2,'same');
c3=medfilt2(b,[3 3]);
c4=medfilt2(b,[5 5]);
subplot(3,2,1),imshow(a),title('Original image')
subplot(3,2,2),imshow(b),title('Salt & pepper noise')
subplot(3,2,3),imshow(uint8(c1)),title('3 x 3 smoothing')
subplot(3,2,4),imshow(uint8(c2)),title('5 x 5 smoothing')
subplot(3,2,5),imshow(uint8(c3)),title('3x 3 Median filter')
subplot(3,2,6),imshow(uint8(c4)),title('5 x 5 Median filter')
```

Challenge (1)

Thank You