

COMPUTER VISION

Subject Code: 3171614

Prepared By:

Prof. Janki Patel
Department of IT
SPCE, Bakrol

Unit:1

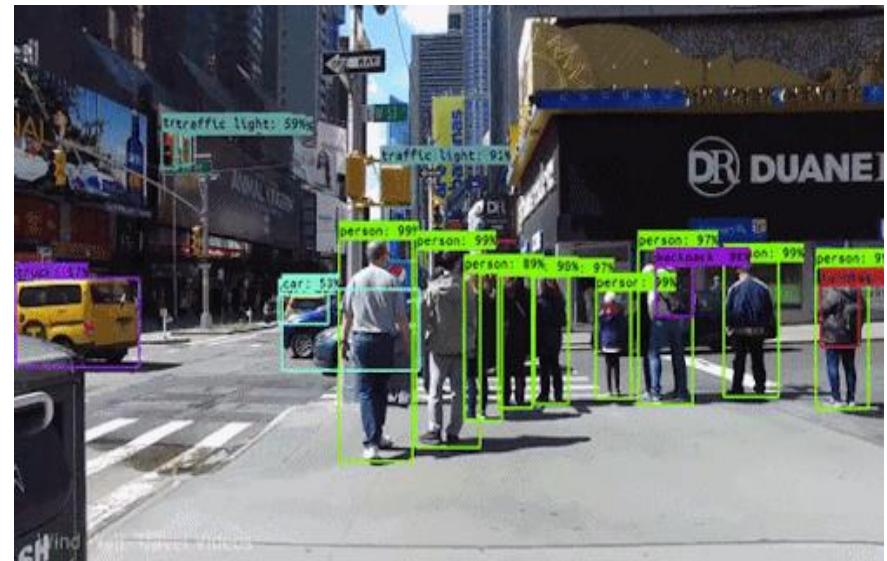
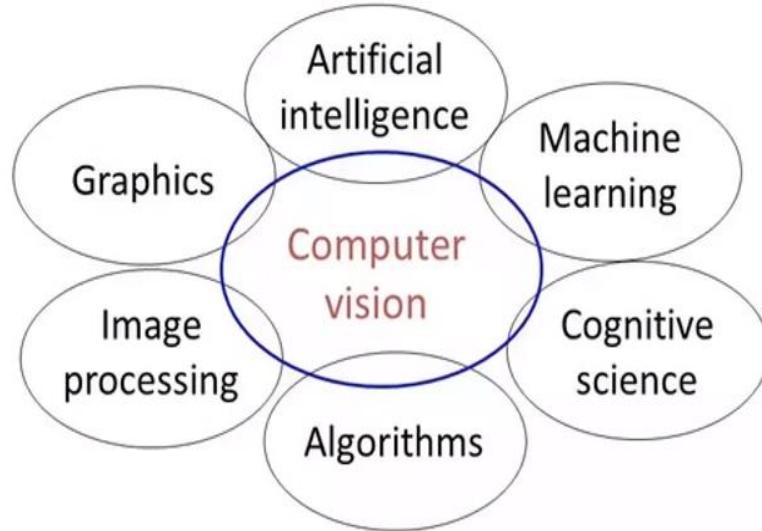
Overview of computer vision and its applications

Content

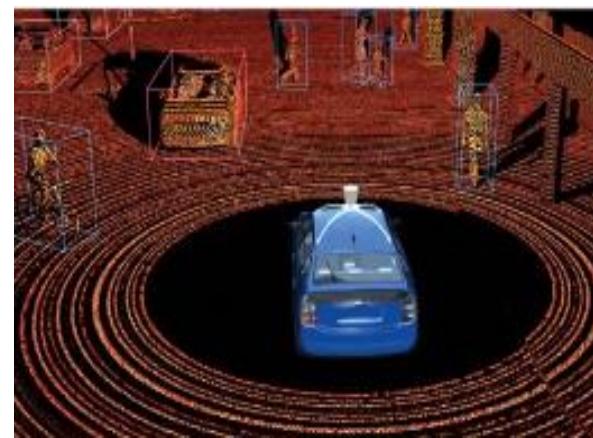
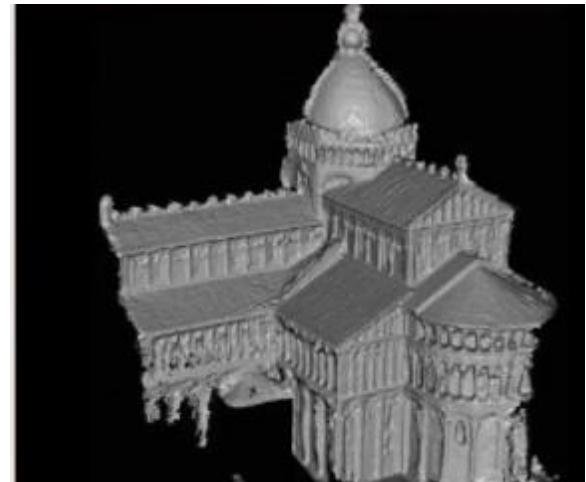
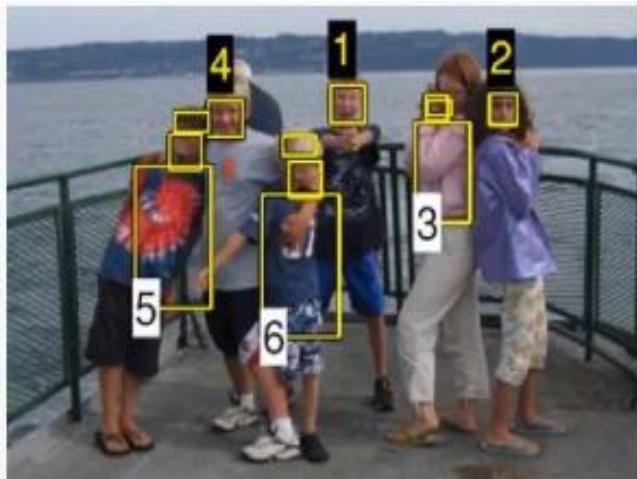
- **Image Formation and Representation:**
- Imaging geometry,
- radiometry,
- digitization,
- cameras and Projections,
- rigid and affine transformation

What is Computer Vision?

- Computer vision is the field of artificial intelligence and computer science that aims at giving computer a visual understanding of the world , and is the heart of Hayo's powerful algorithms.
- It is one of the main components of machine understanding:



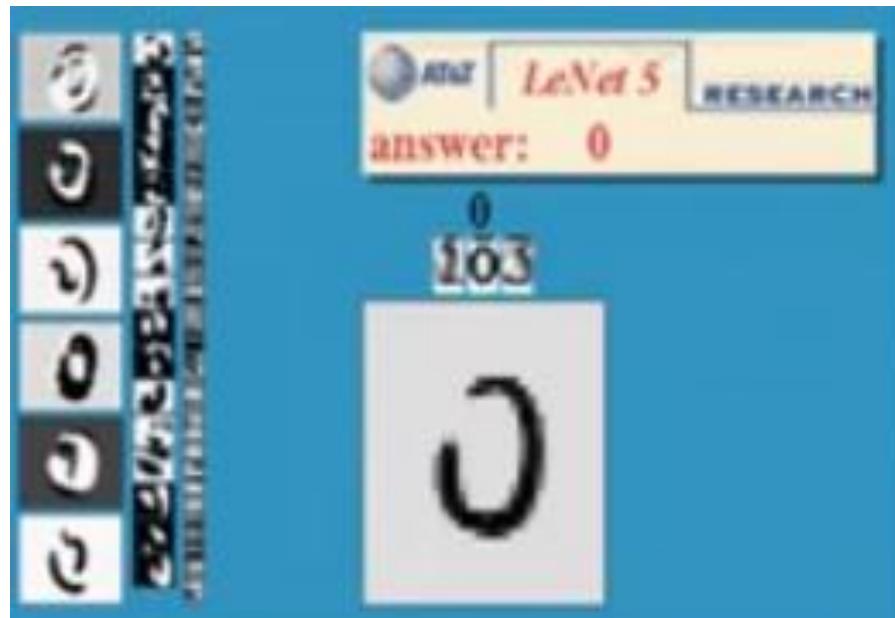
Example of computer vision



Application of computer vision

- The good news is that computer vision is being used today in a wide variety of real-world application, which include:
 - Optical Character recognition(OCR)
 - Machine Inspection
 - Retail
 - 3D model building
 - Medical imaging
 - Automation safety
 - Match move

- **Optical Character recognition(OCR):** Reading handwritten code postal codes on letters and automatic number plate recognition[ANPR].



- **Machine Inspection:** Rapid parts inspection for quality assurance using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts of looking for defects in steel casting using X-ray vision.



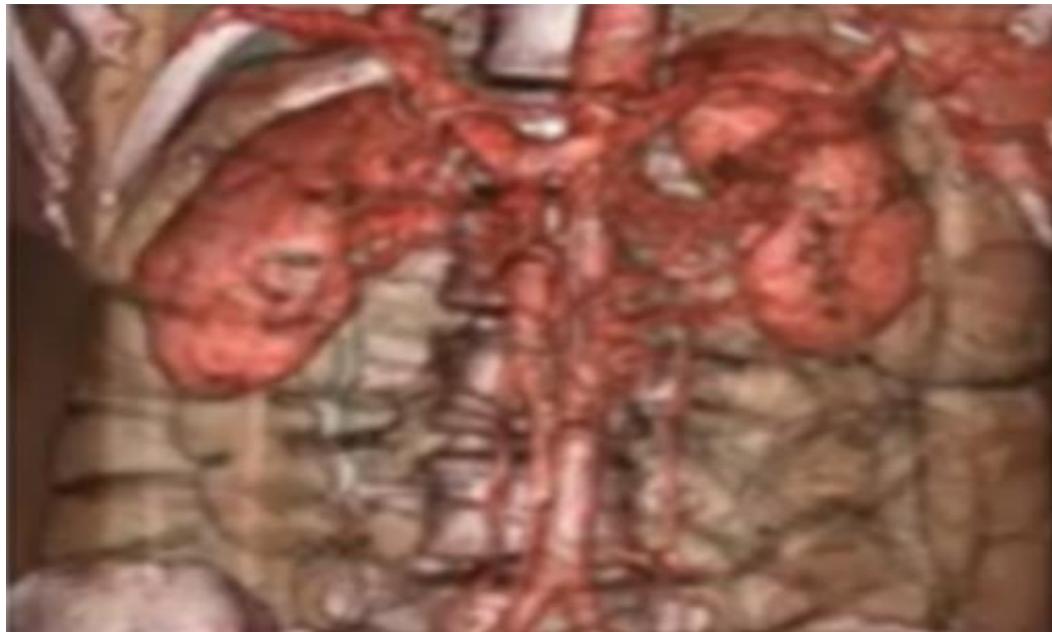
- **Retail:** Object recognition for automated checkout lanes.



- **3D model building:** fully automated construction of 3D models from aerial photographs used in system such Bing Maps.



- **Medical imaging:** Registering pre-operative and intra-operative imagery or performing long-term studies of people's brain morphology as they age.

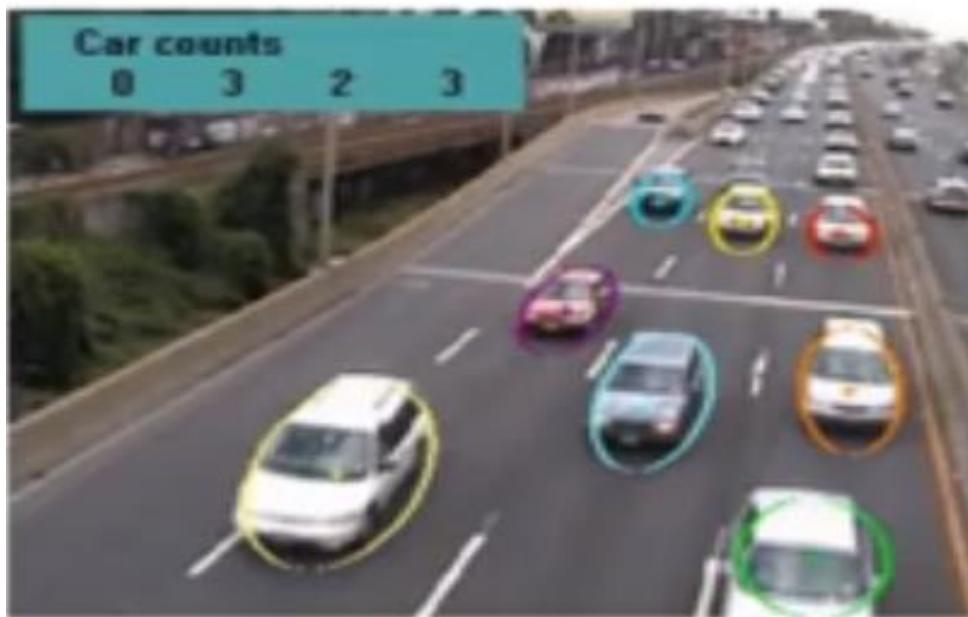


- **Automotive safety:** Detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar or lidar do not work well.



- **Match move:** Merging computer imagery(CGI)with live action footage by tracking feature points in the source video to estimate the 3D camera motion and shape of environment. Such techniques are widely used in Hollywood(e.g, in movies such as Jurassic Park)they also require the use of precise matting to insert new element between foreground and background elements

- **Motion Capture:** Using retro-reflective markers viewed from multiple cameras or other vision-based techniques to capture actors for computer animation
- **Surveillance:** monitoring for intruders, analyzing highway traffic and monitoring pools for drowning victims



- Now, we focus more on broader consumer level applications, such as fun things you can do with your own personal photographs and video. These include:
 - **Stitching:** turning overlapping photos into a single seamlessly stitched panorama
-



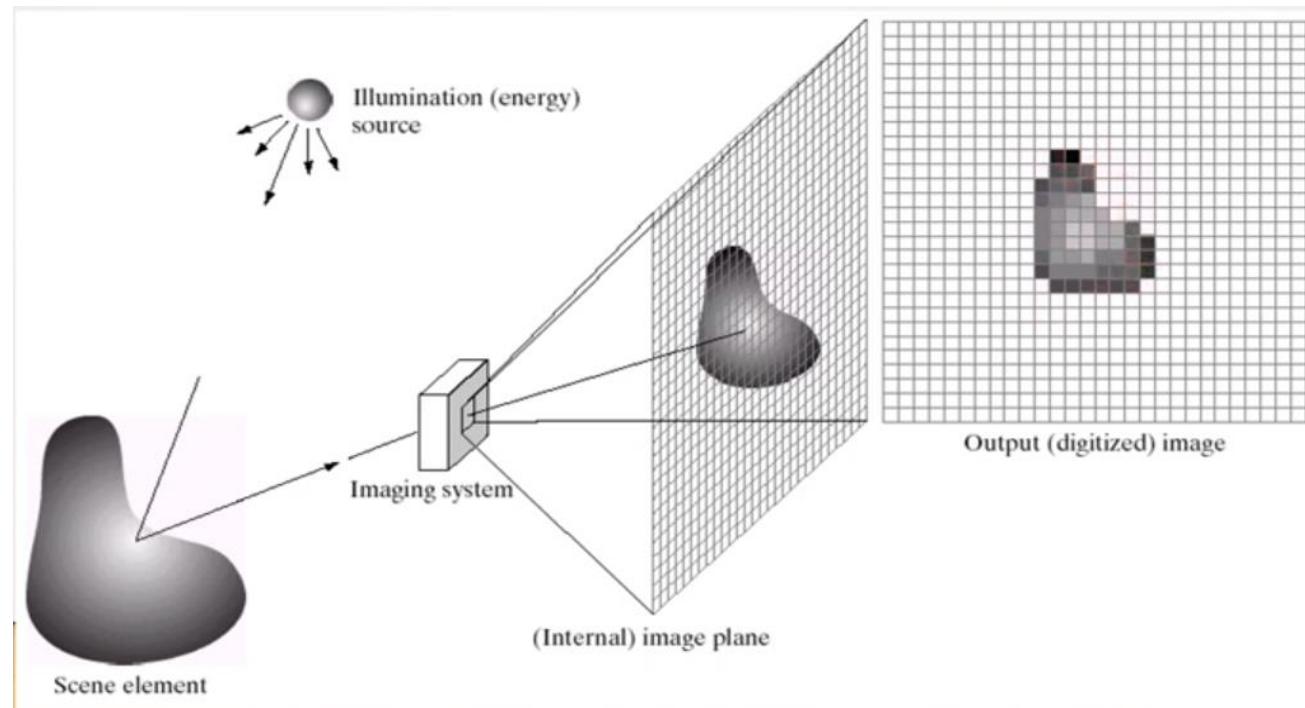
- **Exposure bracketing:** Merging multiple exposures taken under challenging lighting conditions into a single perfectly exposed image.



Image Digitization

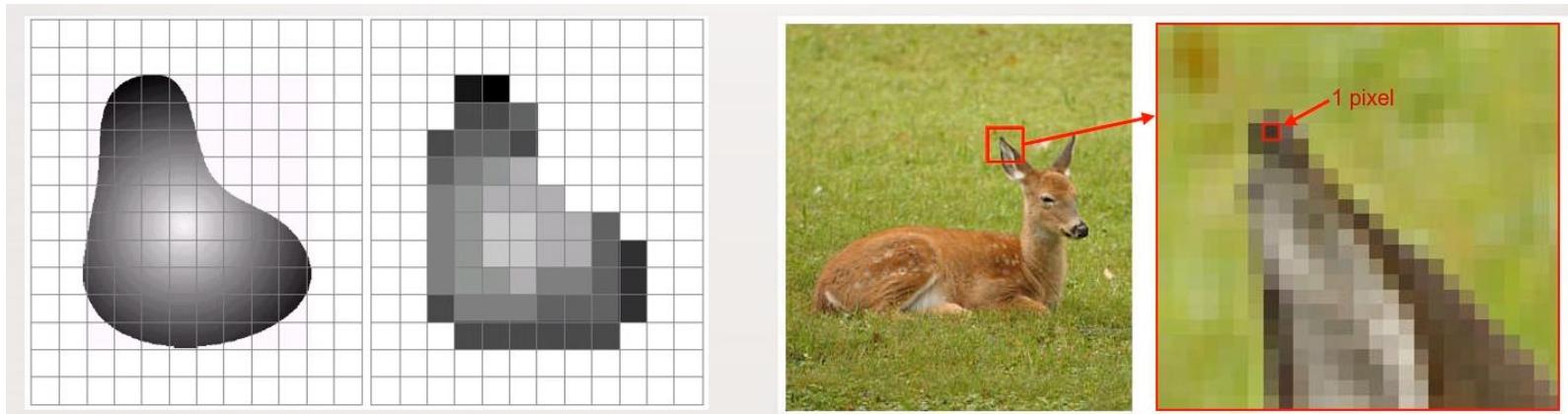
What is a Digital Image?

- A digital image is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels.



Cont...

- Pixel values typically represent gray levels, colors, heights, opacities etc.
- Remember digitization implies that a digital image is an approximation of a real scene.



Cont...

- Common image formats include:
- 1 sample per point(B&W or Grayscale)
- 3sample per point(Red, Green, and Blue)
- sample per point(Red, Green, and Blue, and “Alpha”,a.k.a. Opacity)

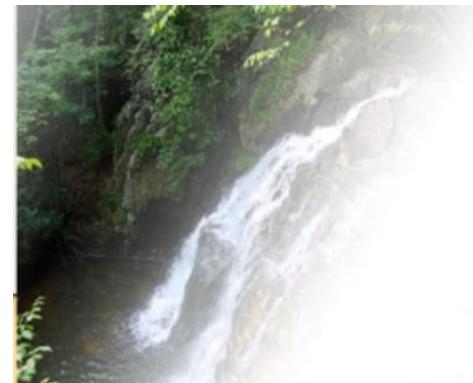


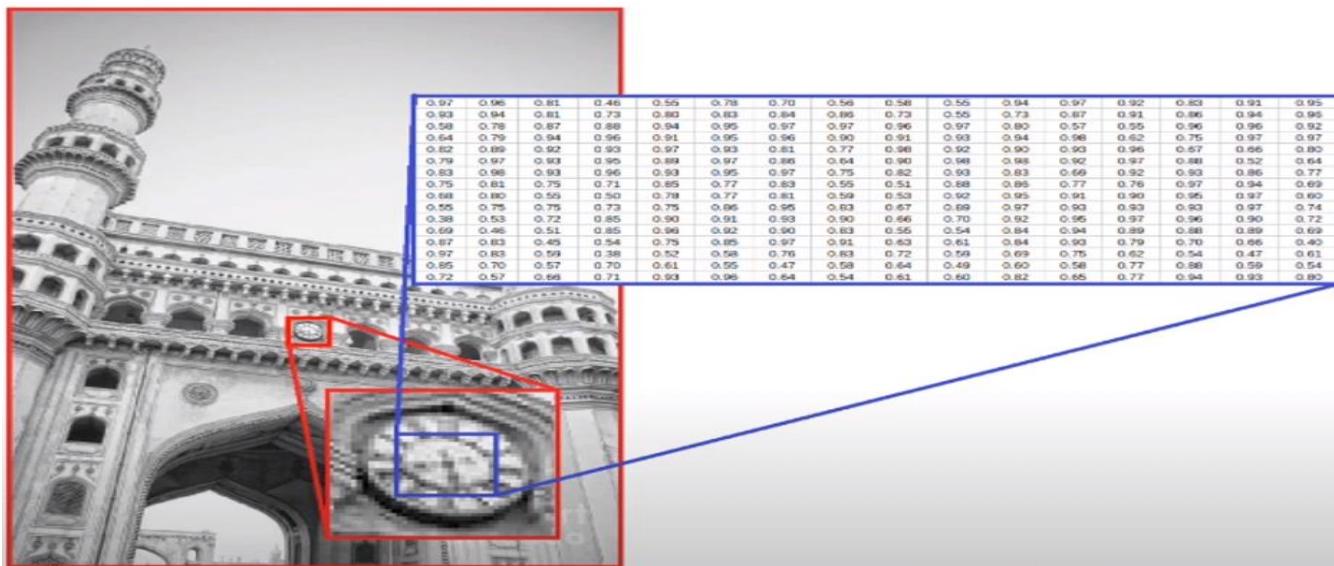
Image Representation

Image Representation

- After getting an image ,it is important to devise ways to represented. Let's look at the most common ways to represent an image.

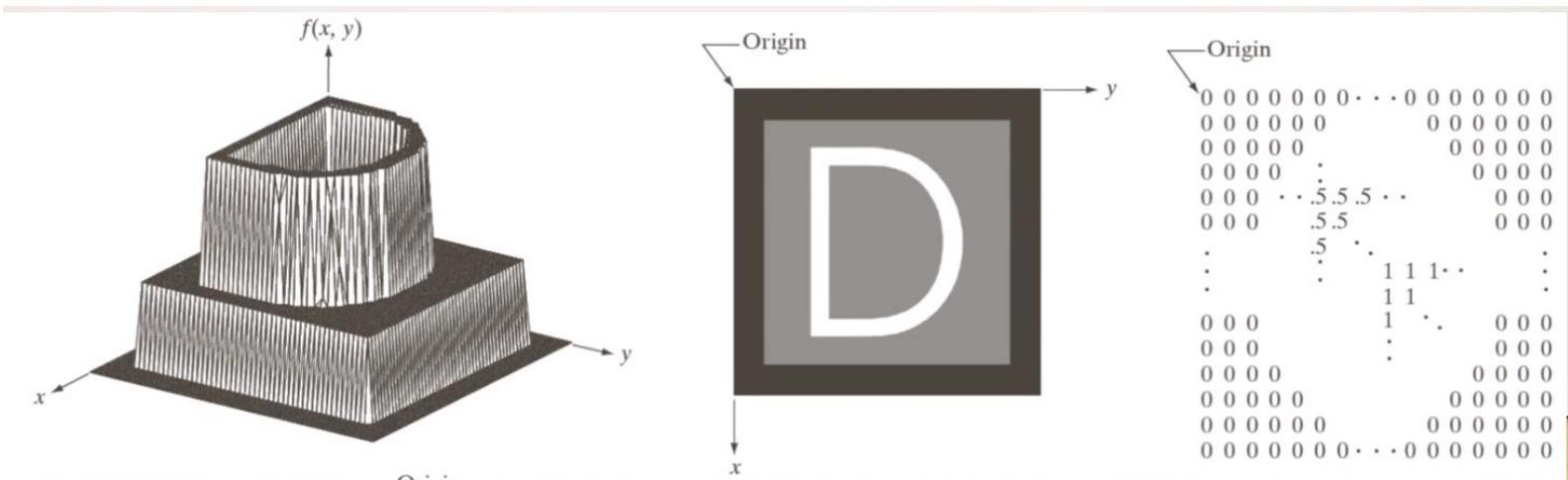
Image Representation

- **Image as matrix**
- The simplest way to represent the image is in the form of a matrix.
- In figure we can see that a part of the image, i.e., the clock, has been represented as a matrix. A similar matrix will represent the rest of the image too.



Cont...

- It is commonly seen that people use up to a byte to represent every pixel of the image. This means that values between 0 to 255 represent the intensity for each pixel in the image where 0 is black and 255 is white. For every color channel in the image, one such matrix is generated.



Cont...

- **Image as a function**
- An image can also be represented as a function. An image(grayscale)can be thought of as a function that takes in a pixel coordinate and gives the intensity at that pixel.
- It can be written as function $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ that outputs the intensity at any input point(x, y).The value of intensity can be between 0 to 255 or 0 to 1 if values are normalized.

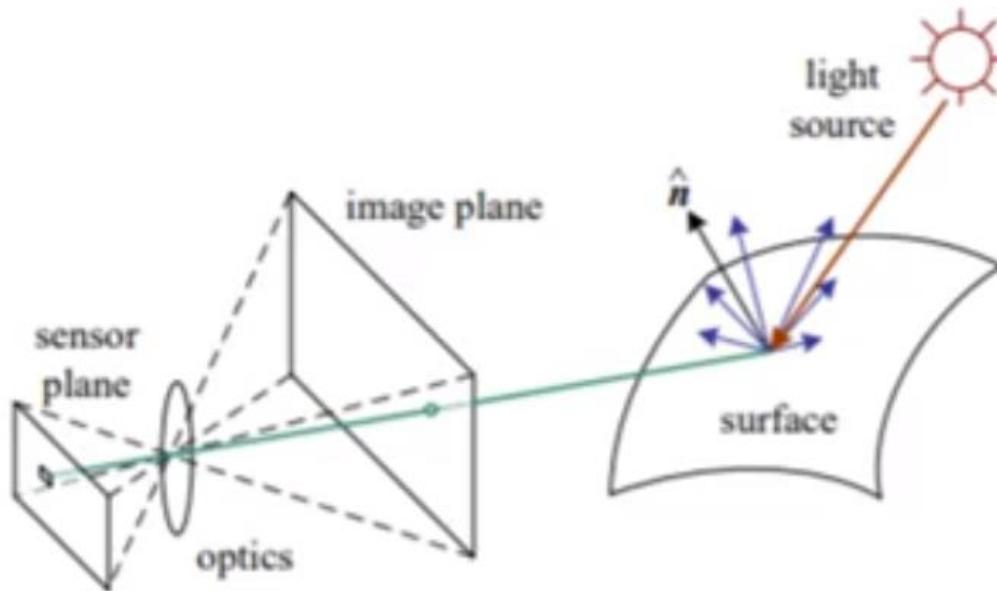
Image Formation

Image Formation

- In modeling any image formation process, geometric primitives and transformations are crucial to project 3-D geometric features. However, apart from geometric features, image formation also depends on discrete color and intensity values.
- It needs to know the lighting of the environment, camera optics, sensor properties, etc. Therefore, while about image formation in computer vision.
 - (200,100,50):RGB
 - (122):Gray scale

Photometric Image Formation

- In figure a simple explanation of image formation. The light from a source is reflected on a particular surface. A part of that reflected light goes through an image plane that reaches a sensor plane via optics.



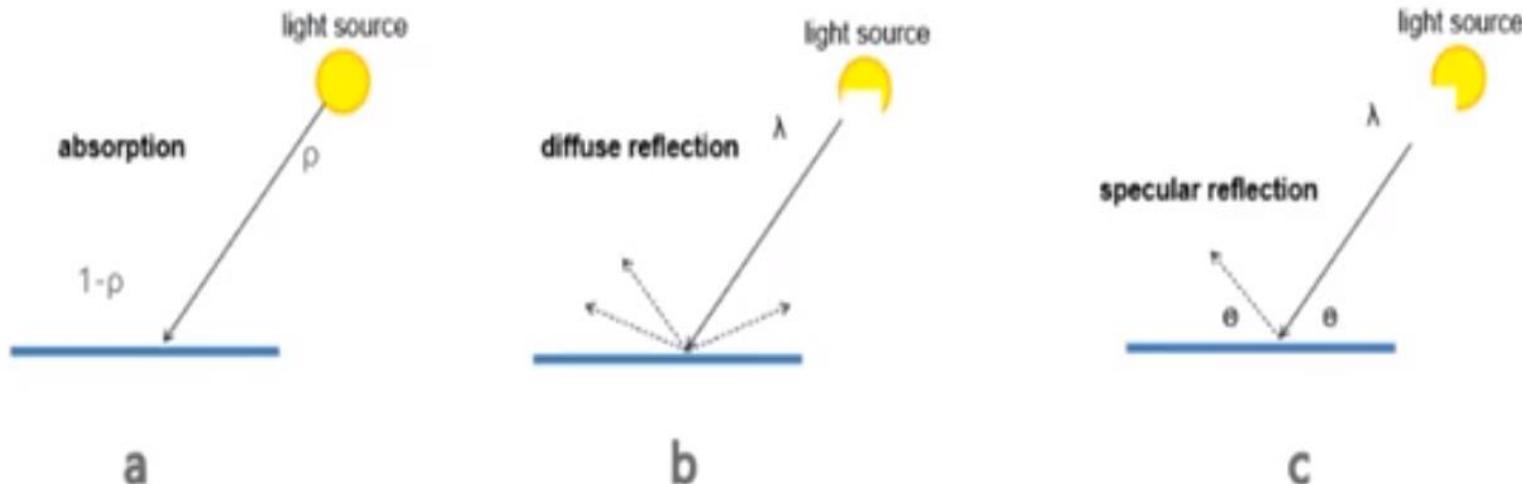
Cont...

- Some factors that affect image formation are:
- The strength and direction of the light emitted from the source.
- The material and surface geometry along with other nearby surfaces.
- Sensor capture properties



Cont...

- **Reflection and Scattering**
- Image cannot exist without light. Light sources can be a point or an area light source. When light hits a surface, three major reactions might occur.



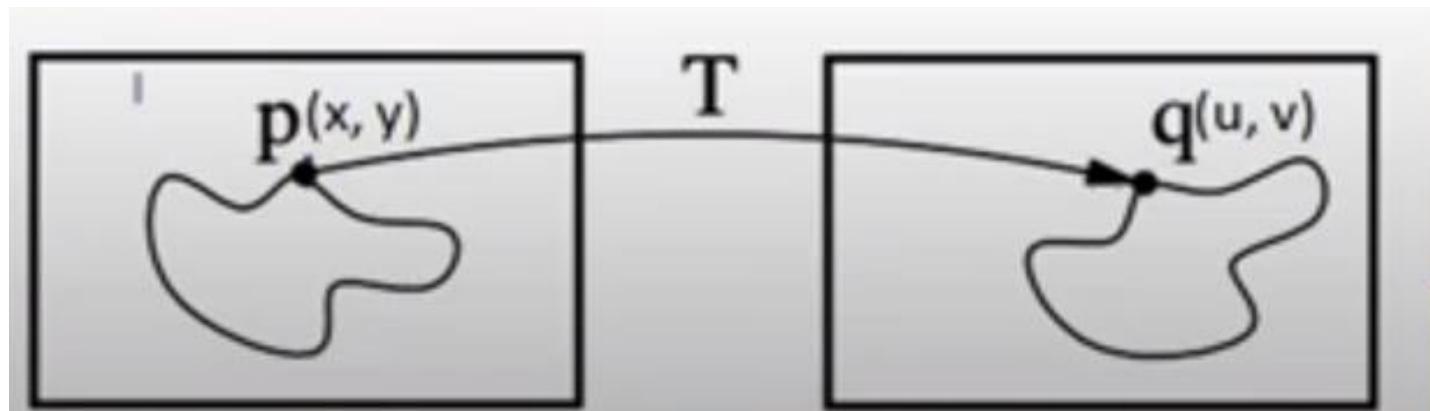
Cont...

- **Color**
- From a viewpoint of color, we know visible light is only a small portion of a large electromagnetic spectrum.
- Two factors are noticed when a colored light arrives at a sensor:
 1. Color of the light
 2. Color of the surface

Image Geometric/Spatial Transformation

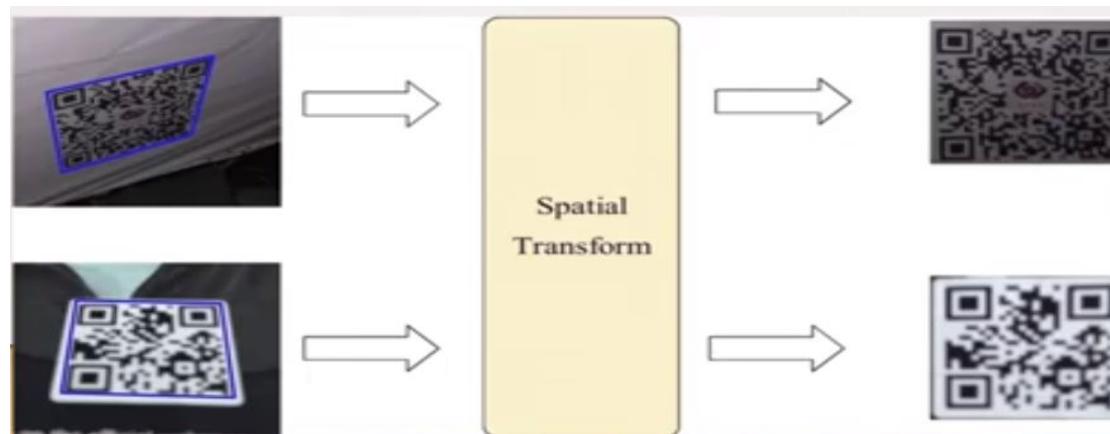
Image Geometric/Spatial Transformation

- Image geometric that means changing the geometry of an image.
- Geometric transforms permit the elimination of distortion that occurs when an image is captured.
- A spatial transformation of an image is a geometric transformation of the image coordinate system.
- In spatial transformation each point (x, y) of image A is mapped to a point (u, v) in a new coordinate system.



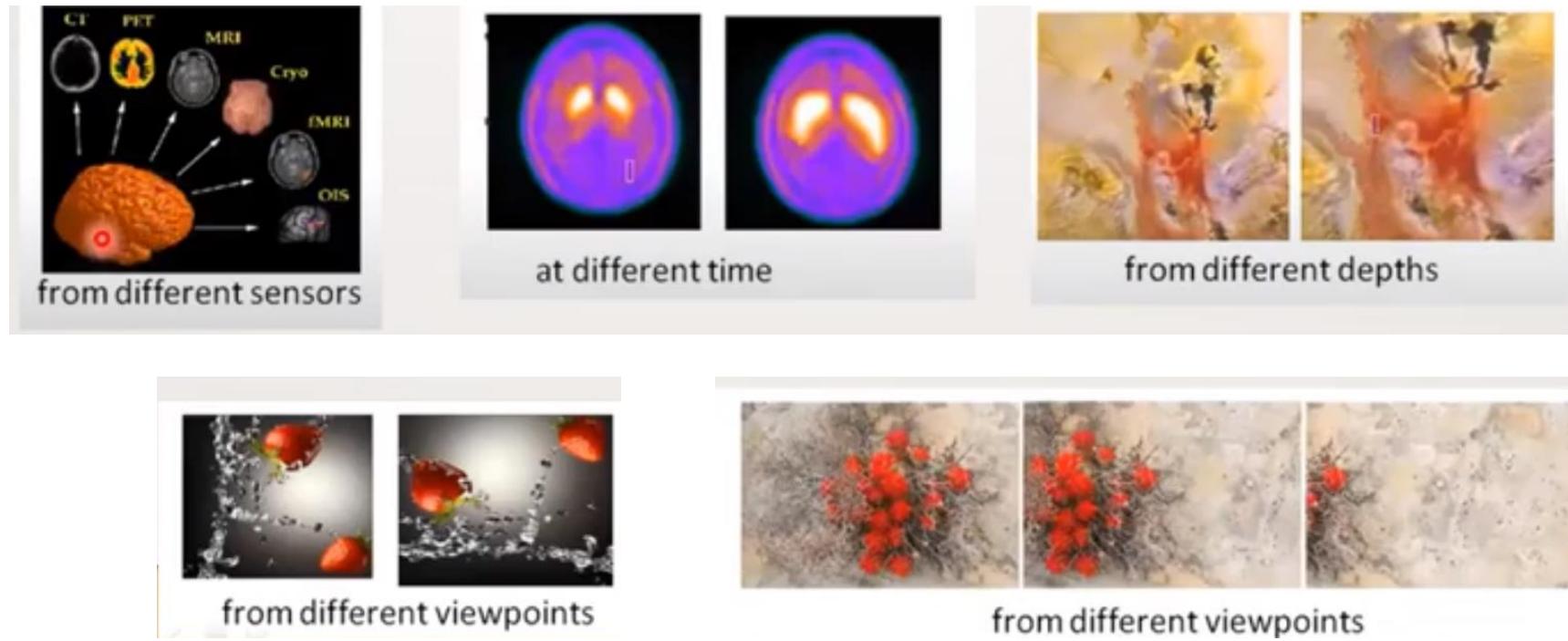
Why it is used?

- Some person clicking the pictures of the same place at different times of the day and year to visualize the changes. Every time he clicks the picture, it's not necessary that he clicks the picture at the exact same angle. So for better visualization, he can align all the images at the same angle using geometric transformation.



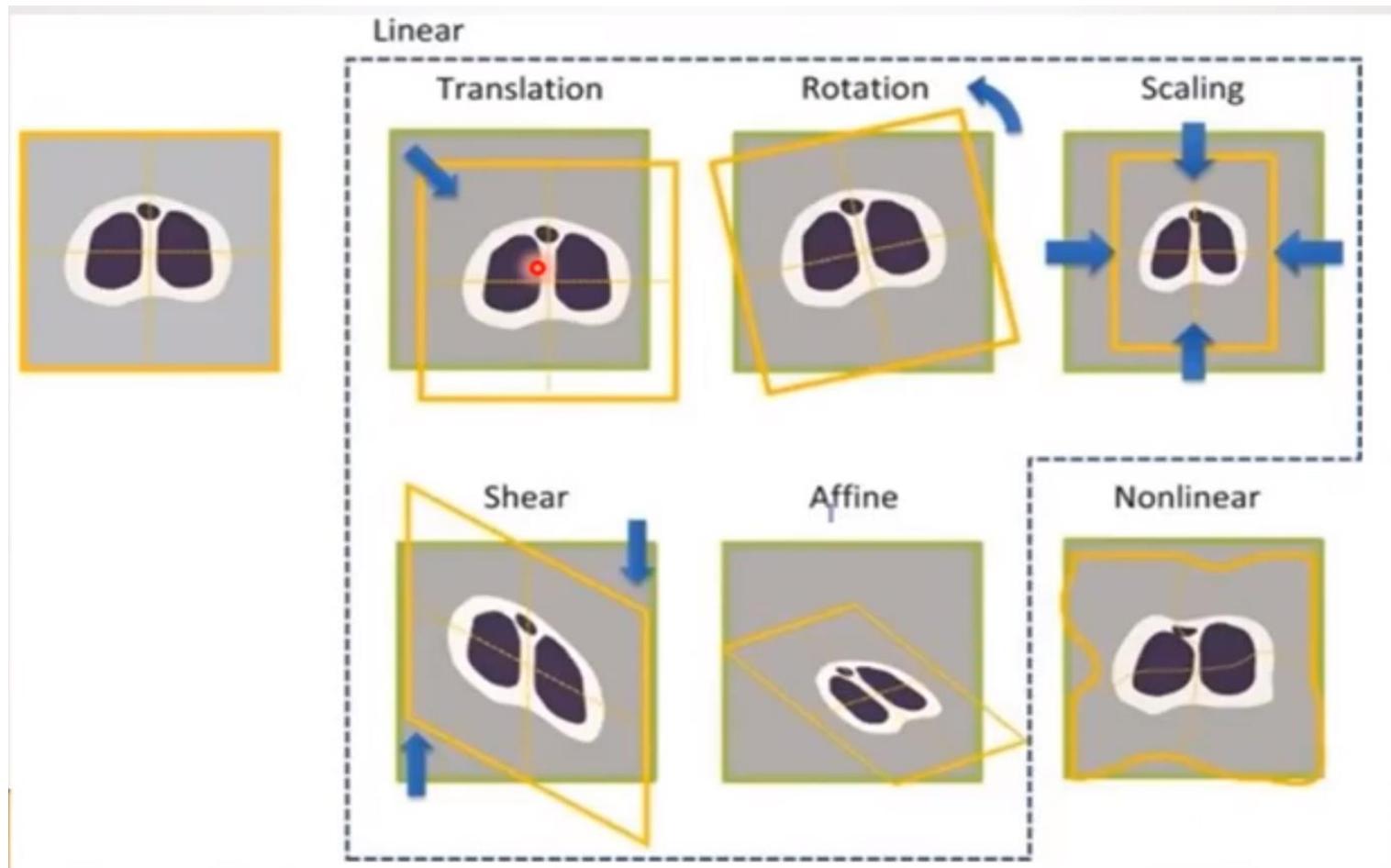
Why geometric transformation is required?

- Image registration is the process of transforming different sets of data into one coordinate system.



Types of geometric Transformation

Types of geometric Transformation



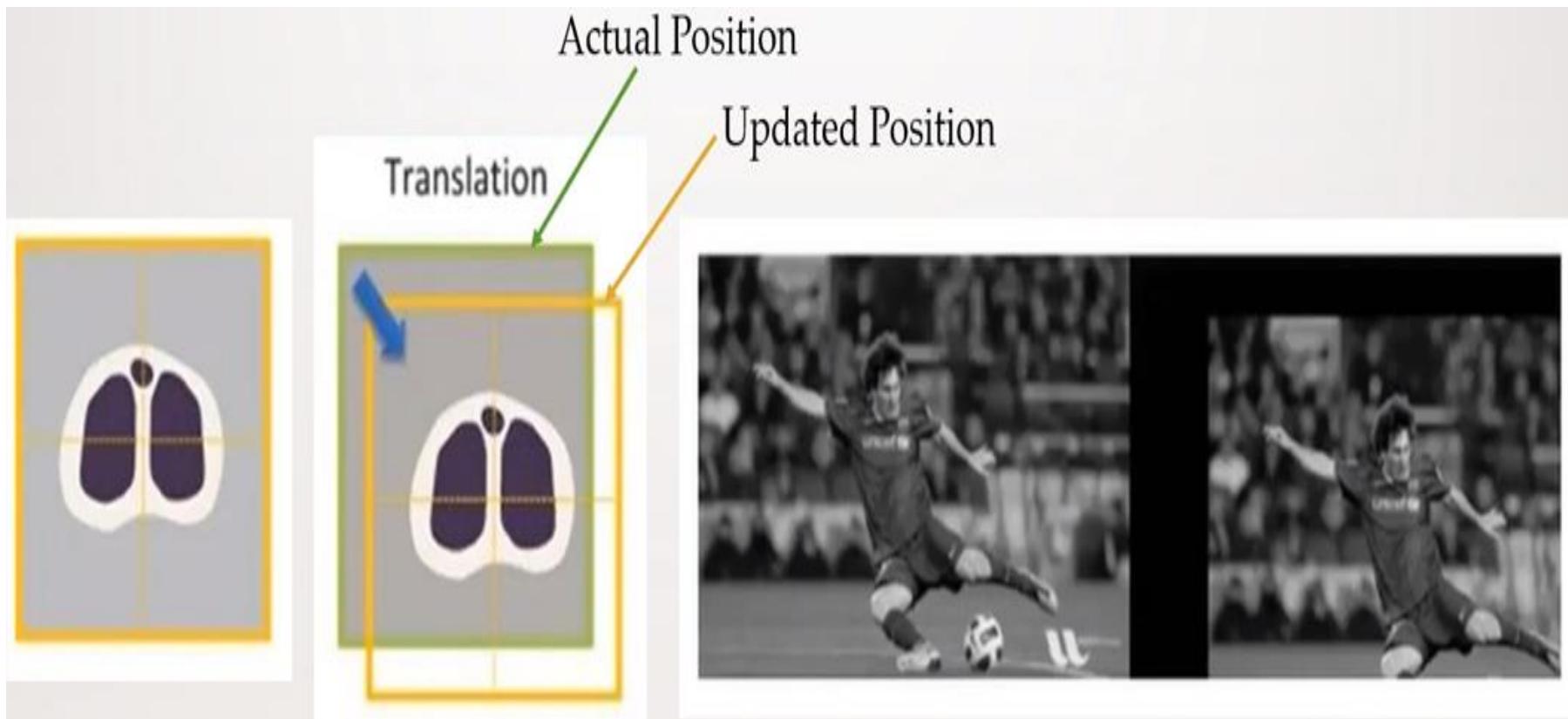
1. Translation

- Translation is the shifting of the object's location. If you know the shift in(x,y) direction, let it be, you can create the transformation matrix as follows:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Cont...



2.Rotation

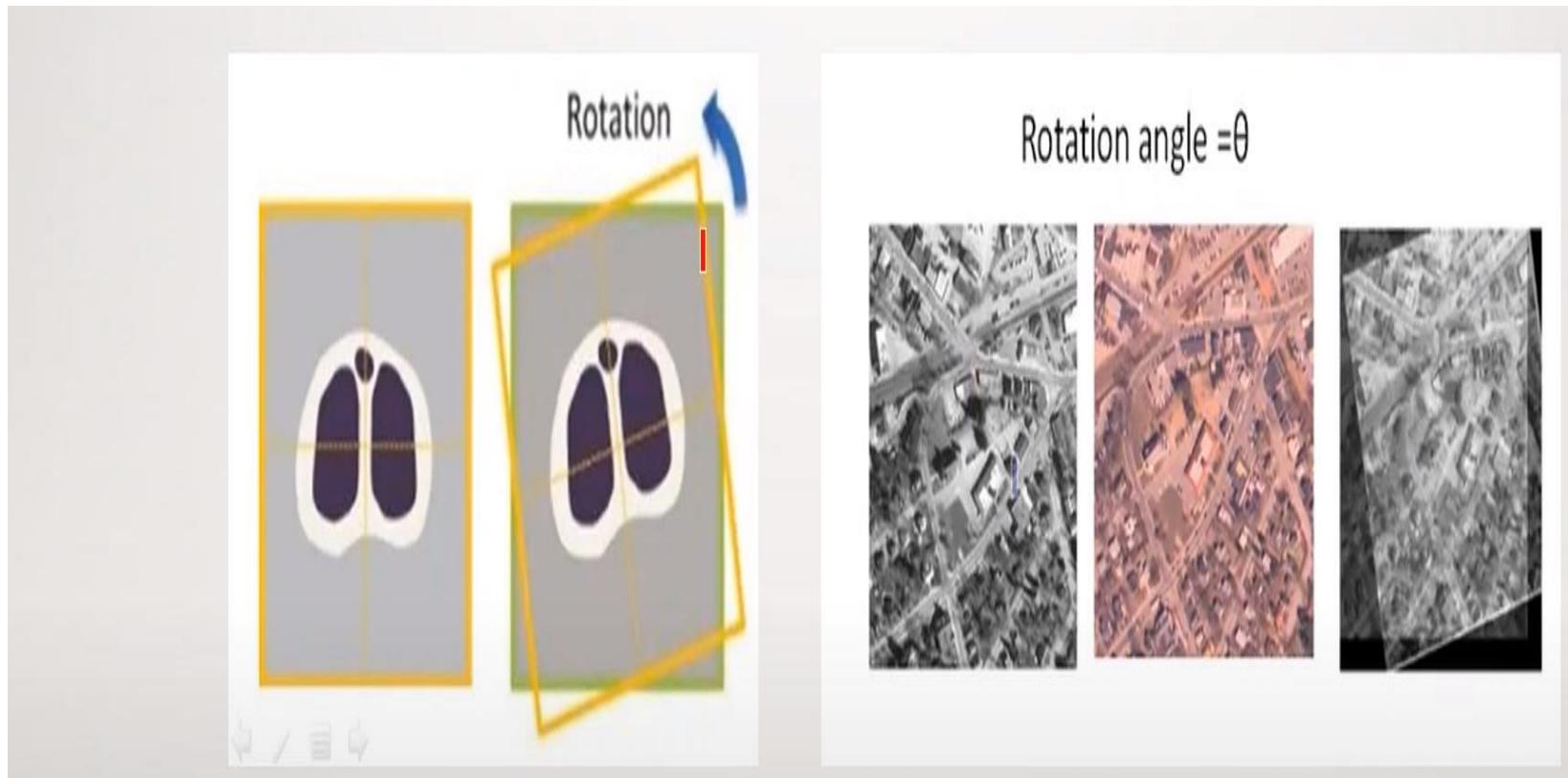
- This technique rotates an image by a specified angle and by the given axis or point.
- The points that lie outside the boundary of an output image are ignored.
- Rotation about the origin by an angle θ is given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$u = x \cos \theta + y \sin \theta$$

$$v = -x \sin \theta + y \cos \theta$$

Cont...



3.Scaling

- Scaling means resizing an image which means an image is made bigger or smaller in x/y direction.
- We can resize an image in terms of scaling factor.

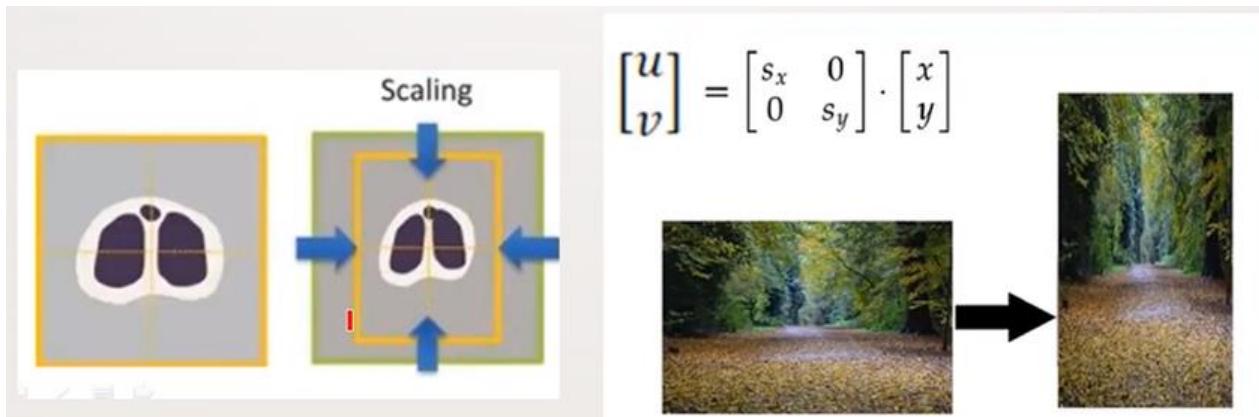
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Cont...

If we have an image of size (300 x 400) and we want to transform it into an image of shape (600 x 200).

The scaling in x- direction will be : $600/300 = 2$. (we denote it as $S_x = 2$)

Similarly $S_y = 200/400 = 1/2$.



$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

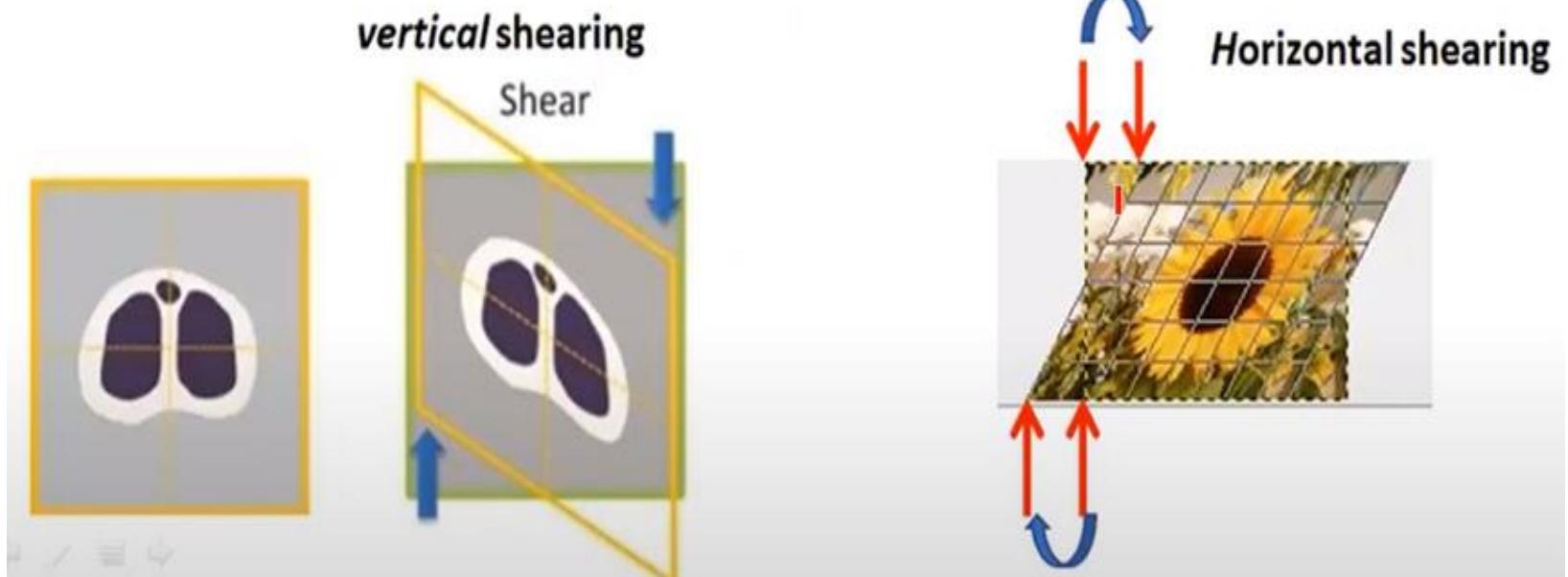


- $(x,y)=(300,400)$
- 300 row and 400 col.
- 300=height 400=width

4.Shearing

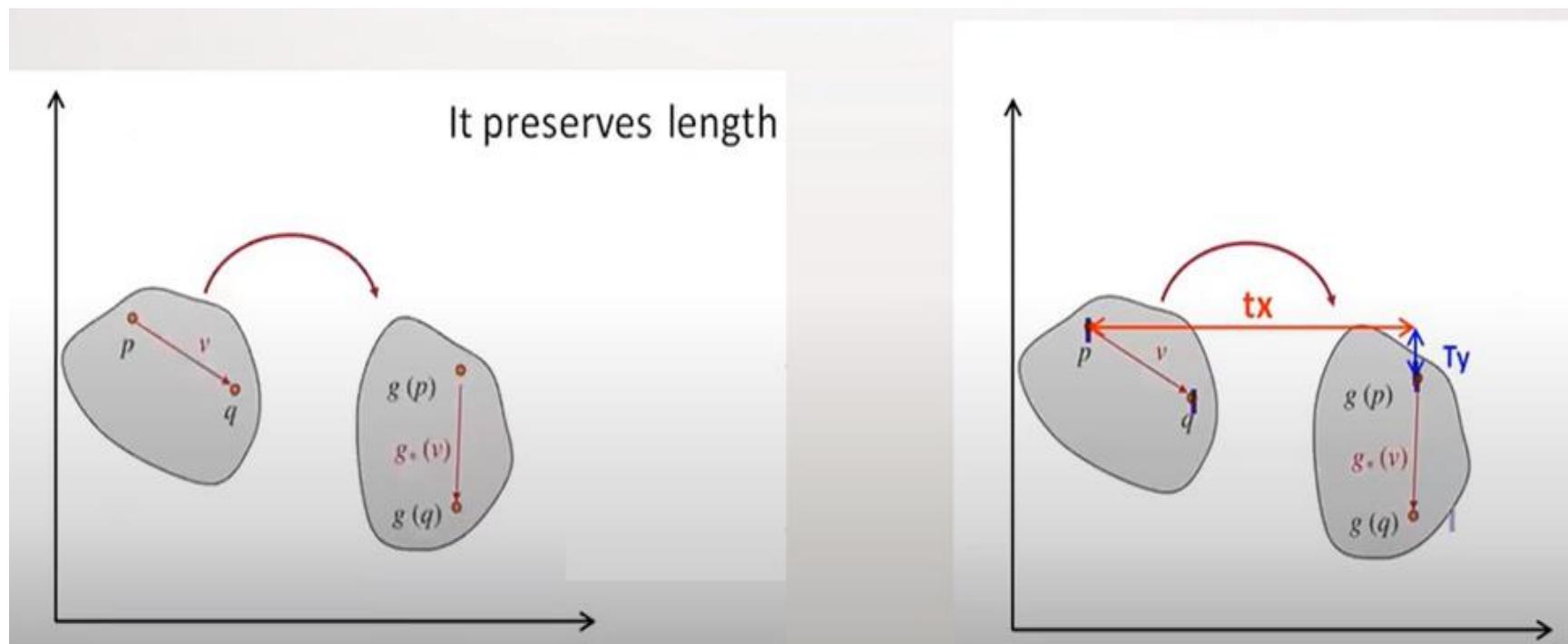
- Shearing an image means shifting the pixel values either horizontally or vertically.
- Basically, this shifts some part of an image to one direction. Horizontal shearing will shift the upper part to the right and lower part to the left.
- Here you can see in gif. That upper part has shifted to the right and the lower part to the left.

Cont...



5.Rigid Transformation

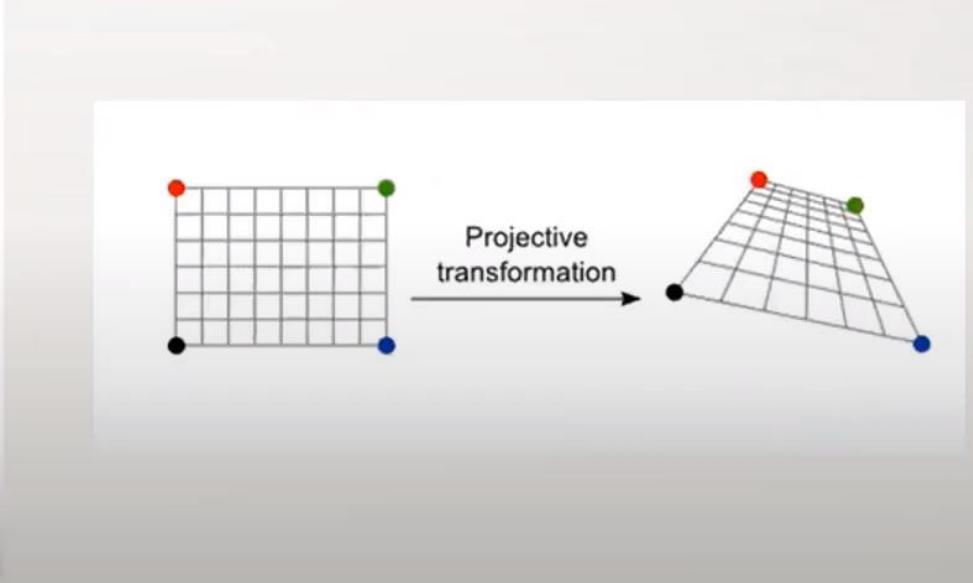
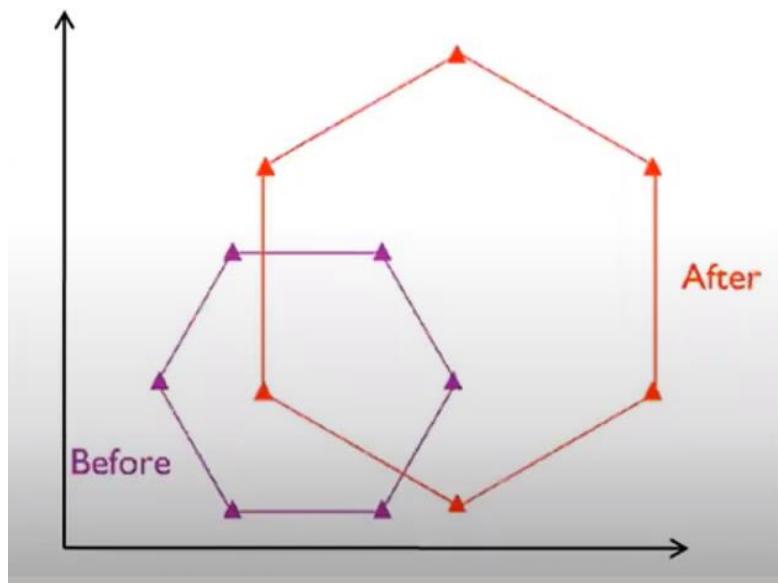
- Rigid = Translations + Rotations



6. Similarity Transformation

- Similarity = Translations + Rotations + Scale

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s \cos\theta & -s \sin\theta & t_x \\ s \sin\theta & s \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{where } s = \text{scaling}$$

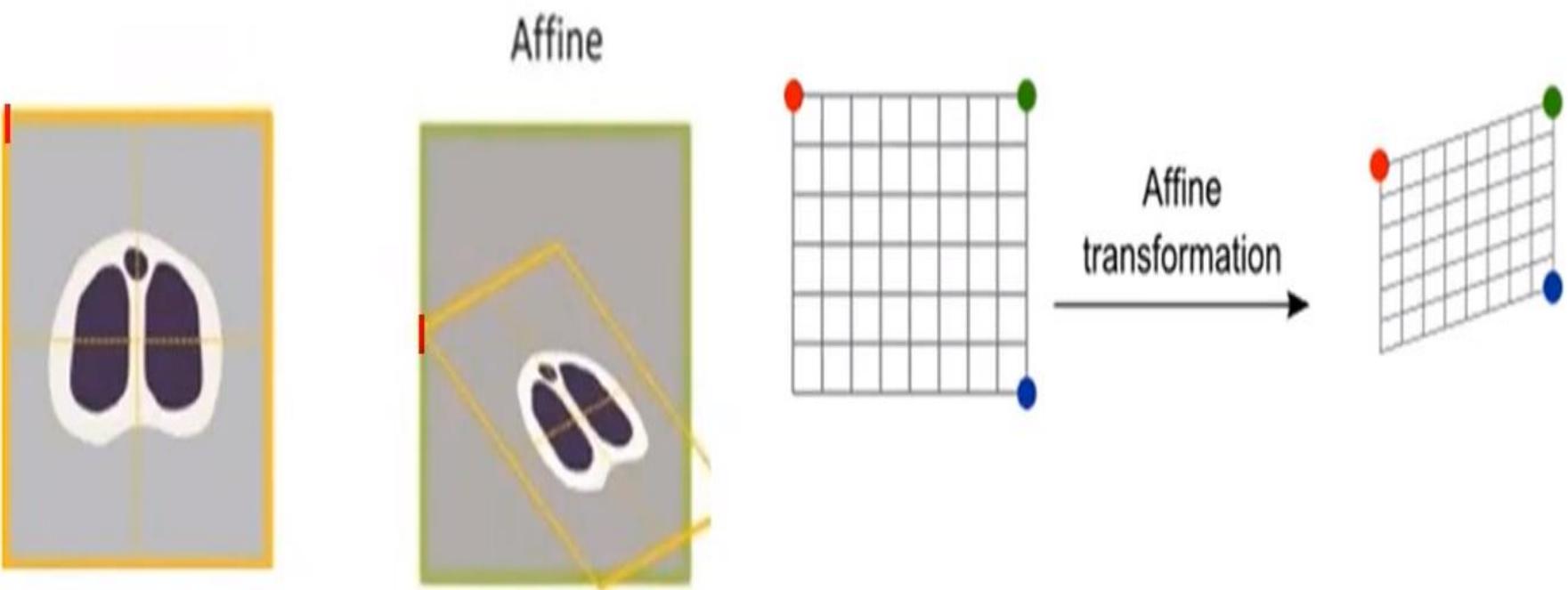


7.Affine Transformation

- Affine = Translations + Rotations + Scale + shear
- An affine transformation is a transformation that preserves co-linearity and the ratio of distances.
- The parallel lines in an original image will be parallel in the output image.

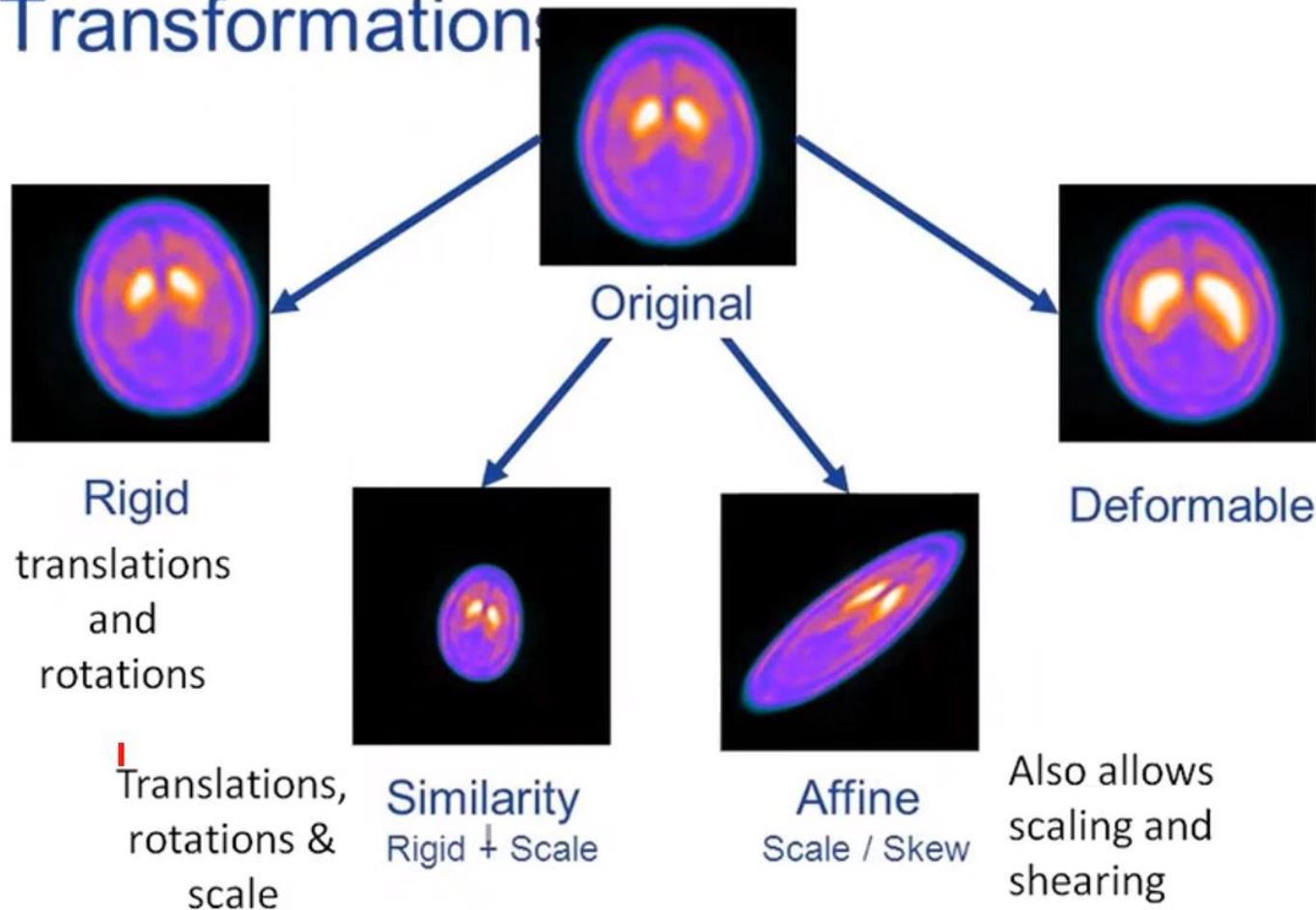
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Cont...

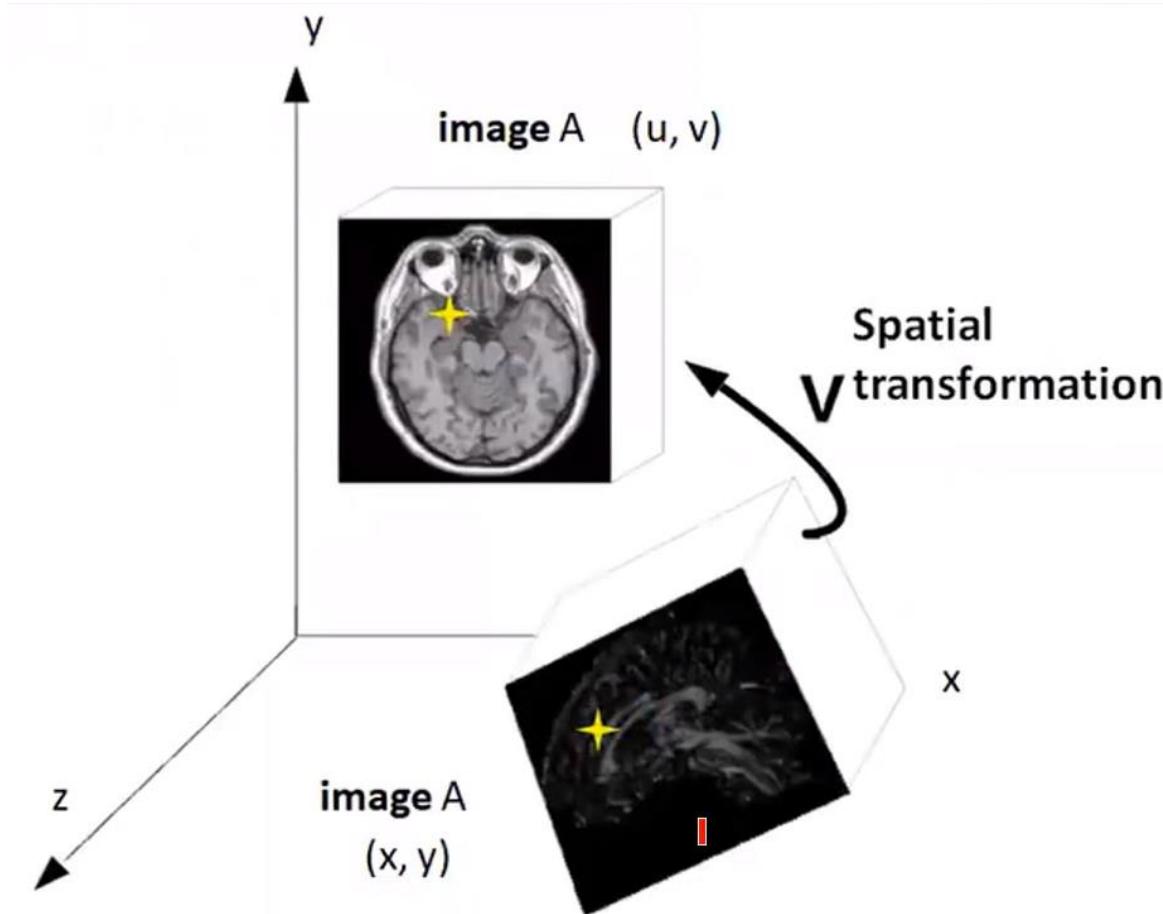


Example Registration Transformations

Example Registration Transformations



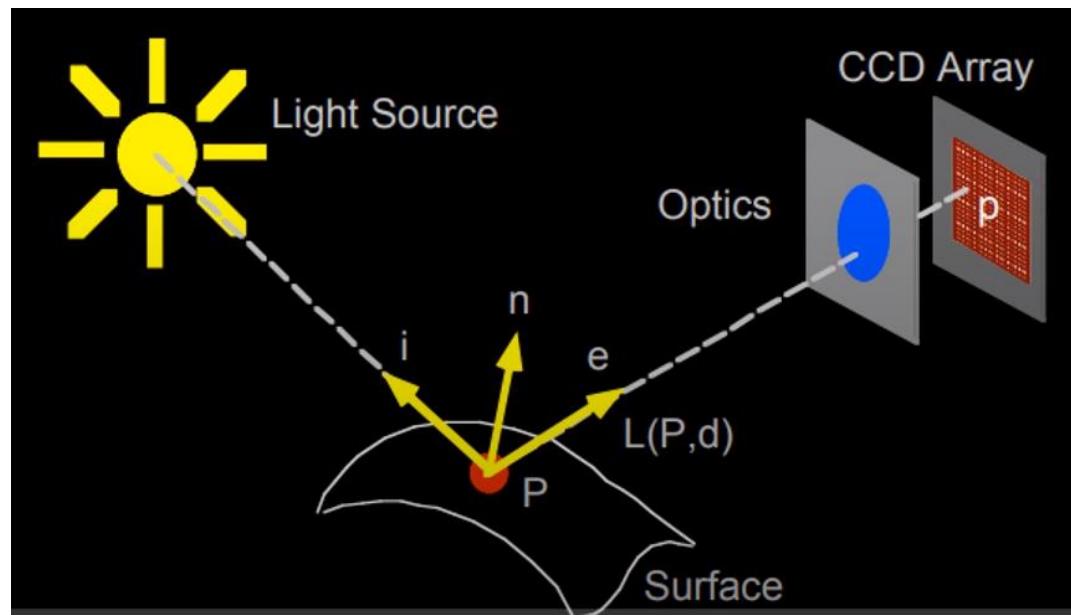
3D spatial Transformation



Radiometry

What is Radiometry?

- Radiometry is the part of image formation concerned with the relation among the amounts of light energy emitted from light sources, reflected from surfaces, and registered by sensors.



Cont...

- Concerned with the relationship between the amount of light radiating from a surface and the amount incident at its image.
- In other words, what the brightness of the point will be.

From 3D to 2D

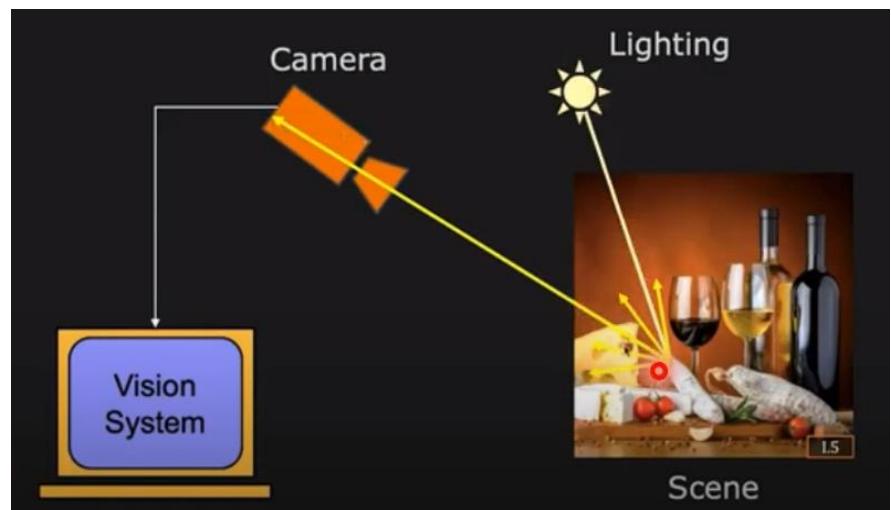
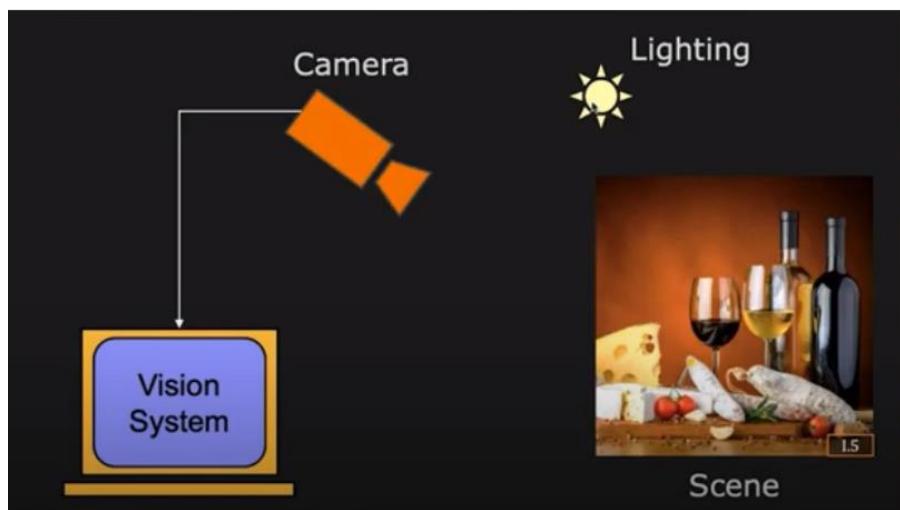


Image Intensity

- Image intensity understanding is under-constrained.

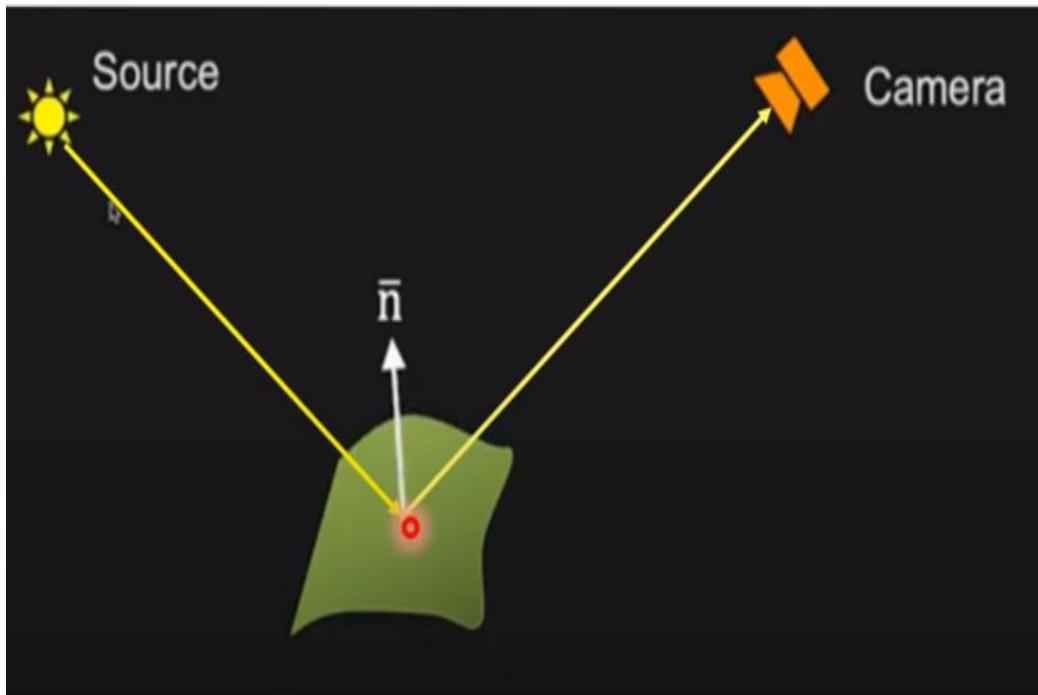


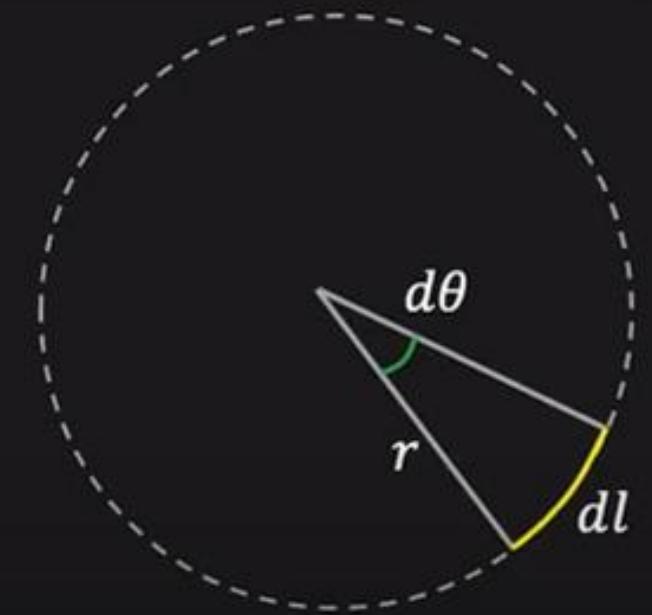
Image Intensity = f (Illumination,
Surface Orientation,
Surface Reflectance)

Concept of Angle (2D): $d\theta$

$$d\theta = \frac{dl}{r}$$

Unit: **radian** (rad)

$d\theta$ is **dimensionless**

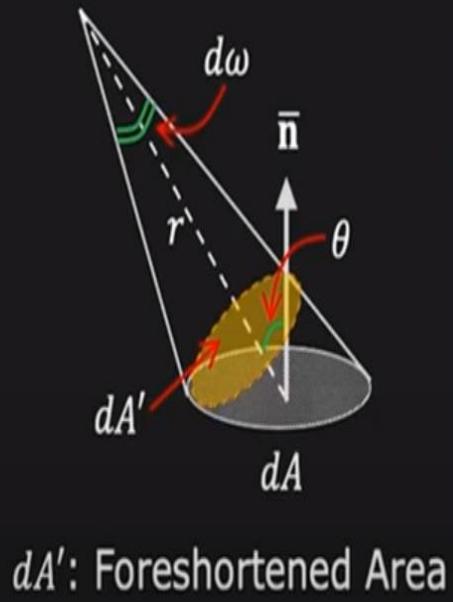


Solid Angle(3D): $d\omega$

$$d\omega = \frac{dA'}{r^2} = \frac{dA \cos \theta}{r^2}$$

Unit: **steradian (sr)**

$d\omega$ is **dimensionless**



dA' : Foreshortened Area

1. Area of sphere $A = 4\pi r^2$
 - Solid angle of sphere = 4π
2. Area of hemisphere $A = 2\pi r^2$
 - Solid angle of sphere = 2π

Light Flux : $d\Phi$



- Luminous flux(in lumens) is a measure of the total amount of light a lamp puts out.

Power emitted within a solid angle

$d\Phi$

Unit: watts (W)

The diagram shows a point source labeled "Source J " emitting yellow arrows representing "Flux". A red arrow points from the source towards a small circular area dA on a plane. A green arc labeled $d\omega$ indicates a differential solid angle subtended by the area dA from the source. A normal vector \bar{n} is shown pointing upwards from the surface.

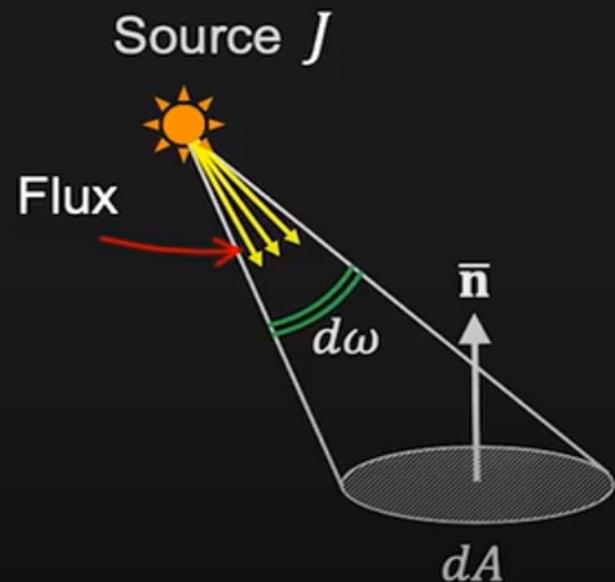
Radiant Intensity: J

- Brightness of source

Light flux emitted per unit solid angle

$$J = \frac{d\Phi}{d\omega}$$

Unit: W sr^{-1}



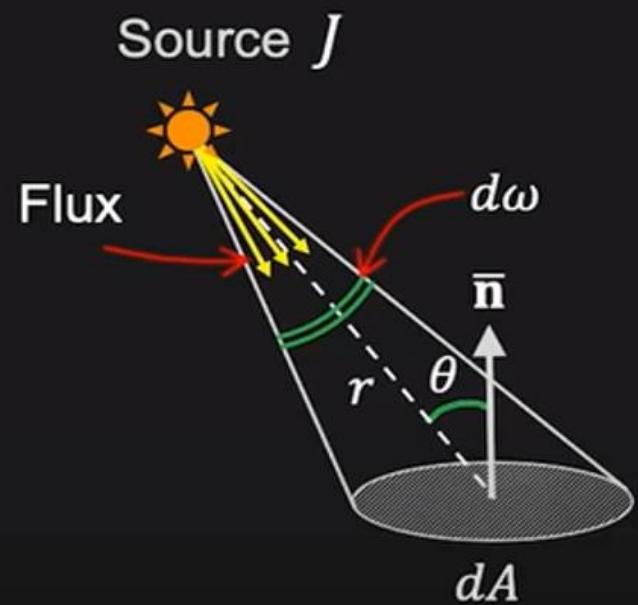
Surface Irradiance: E

- Illumination of surface.

Light flux incident per unit surface area

$$E = \frac{d\Phi}{dA}$$

Unit: W m^{-2}



Cont...

$$E = \frac{d\Phi}{dA}$$

$$d\theta = \frac{dl}{r}$$

$$d\omega = \frac{dA'}{r^2} = \frac{dA \cos \theta}{r^2}$$

$$J = \frac{d\Phi}{d\omega}$$

$$E = \frac{d\Phi}{dA}$$

$$d\phi = J \cdot d\omega$$

$$E = \frac{J \cdot d\omega}{dA}$$

$$E = \frac{J d\omega}{dA} = \frac{J \frac{dA \cos \theta}{r^2}}{dA} = \frac{J \cos \theta}{r^2}$$

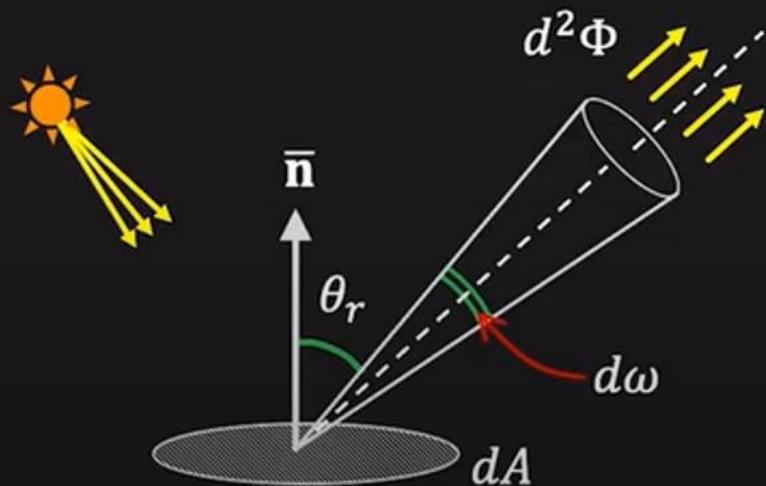
Surface Radiance : L

- Brightness of surface itself.

Light flux emitted per unit foreshortened area per unit solid angle

$$L = \frac{d^2\Phi}{(dA \cos \theta_r) d\omega}$$

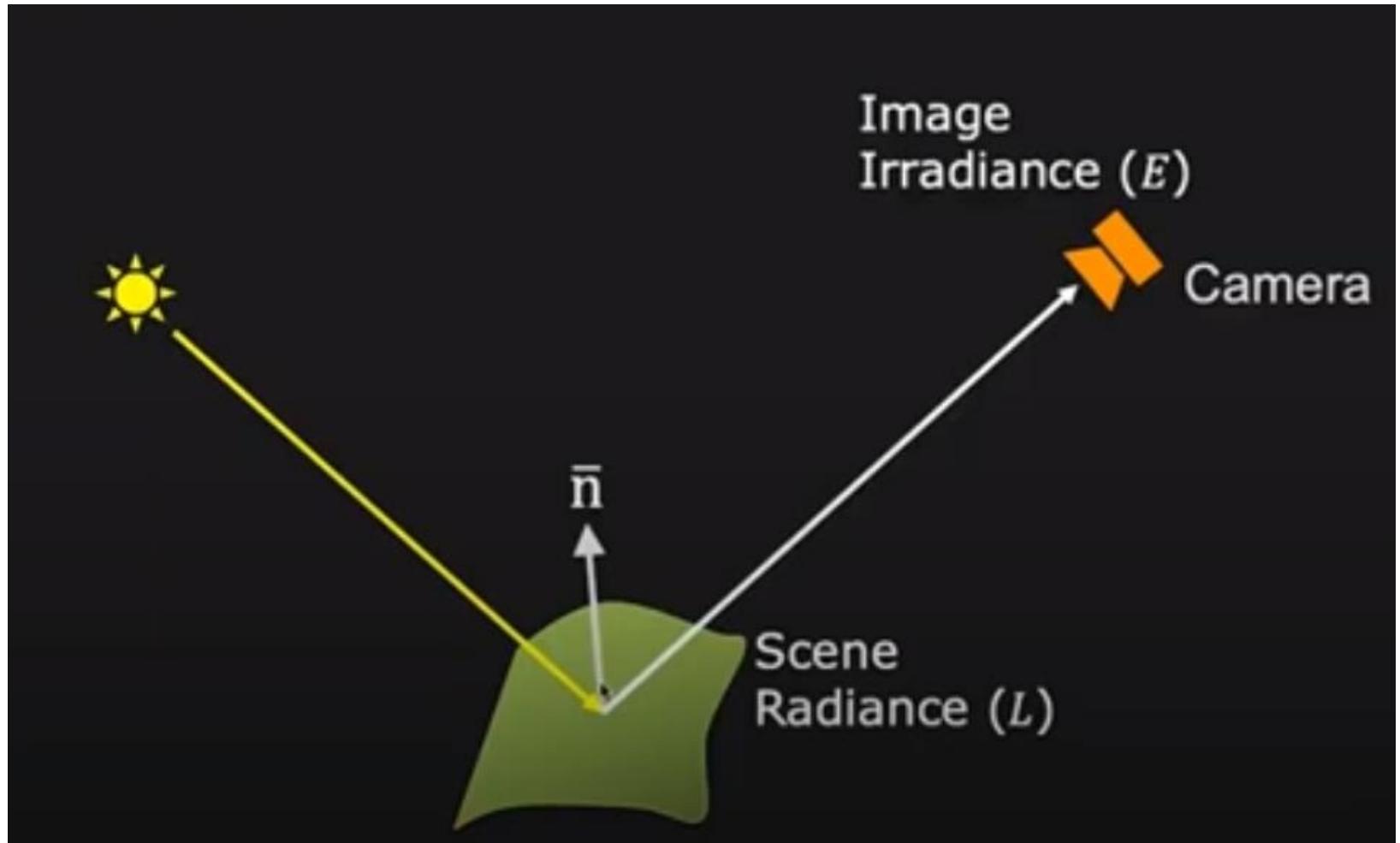
Unit: $\text{W m}^{-2} \text{sr}^{-1}$



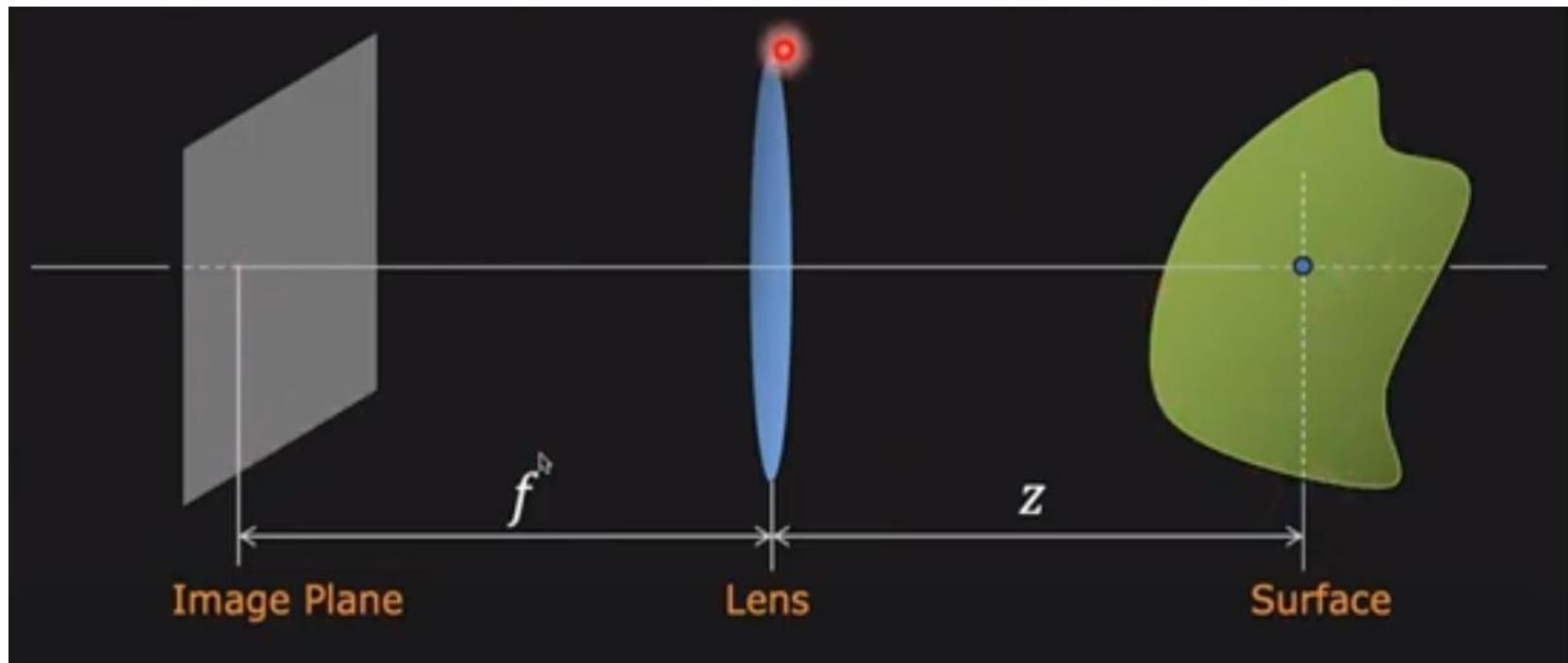
$dA \cos \theta_r$: Foreshortened Area

Radiometry: Scene Radiance and Image Irradiance

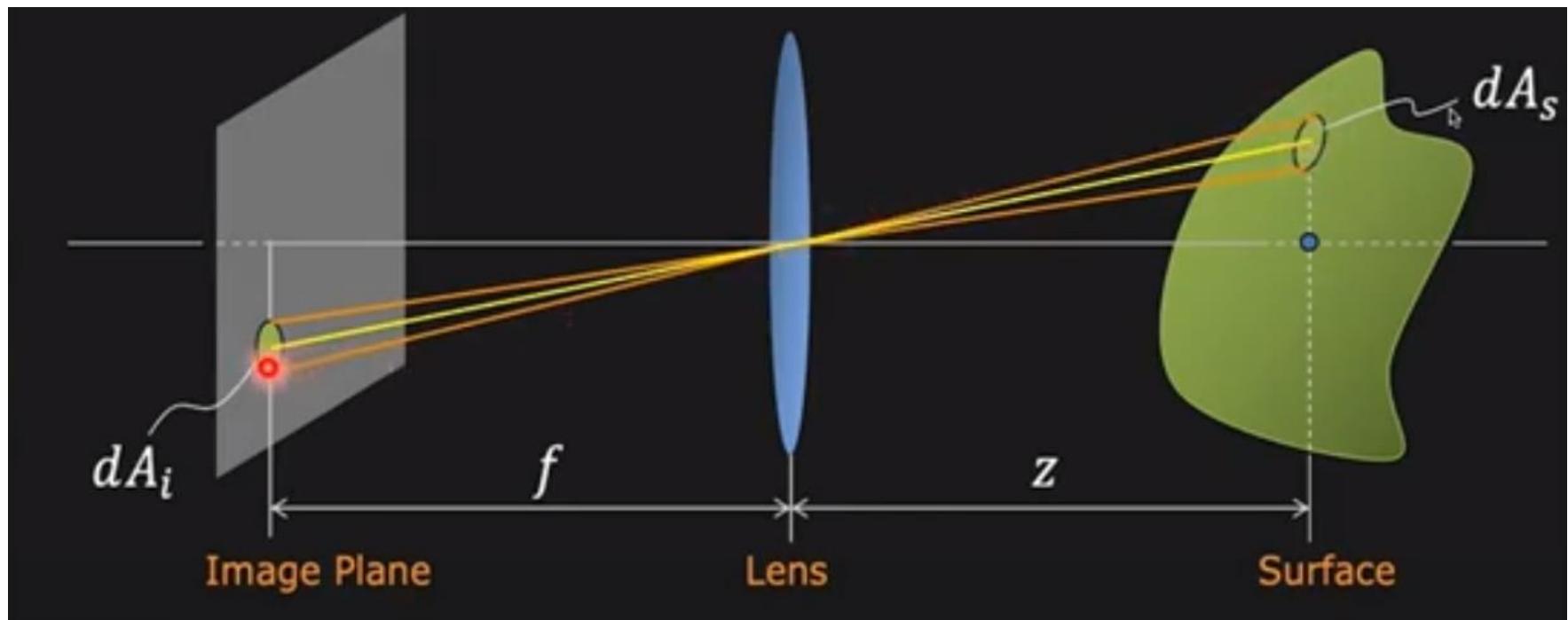
Scene Radiance and Image Irradiance



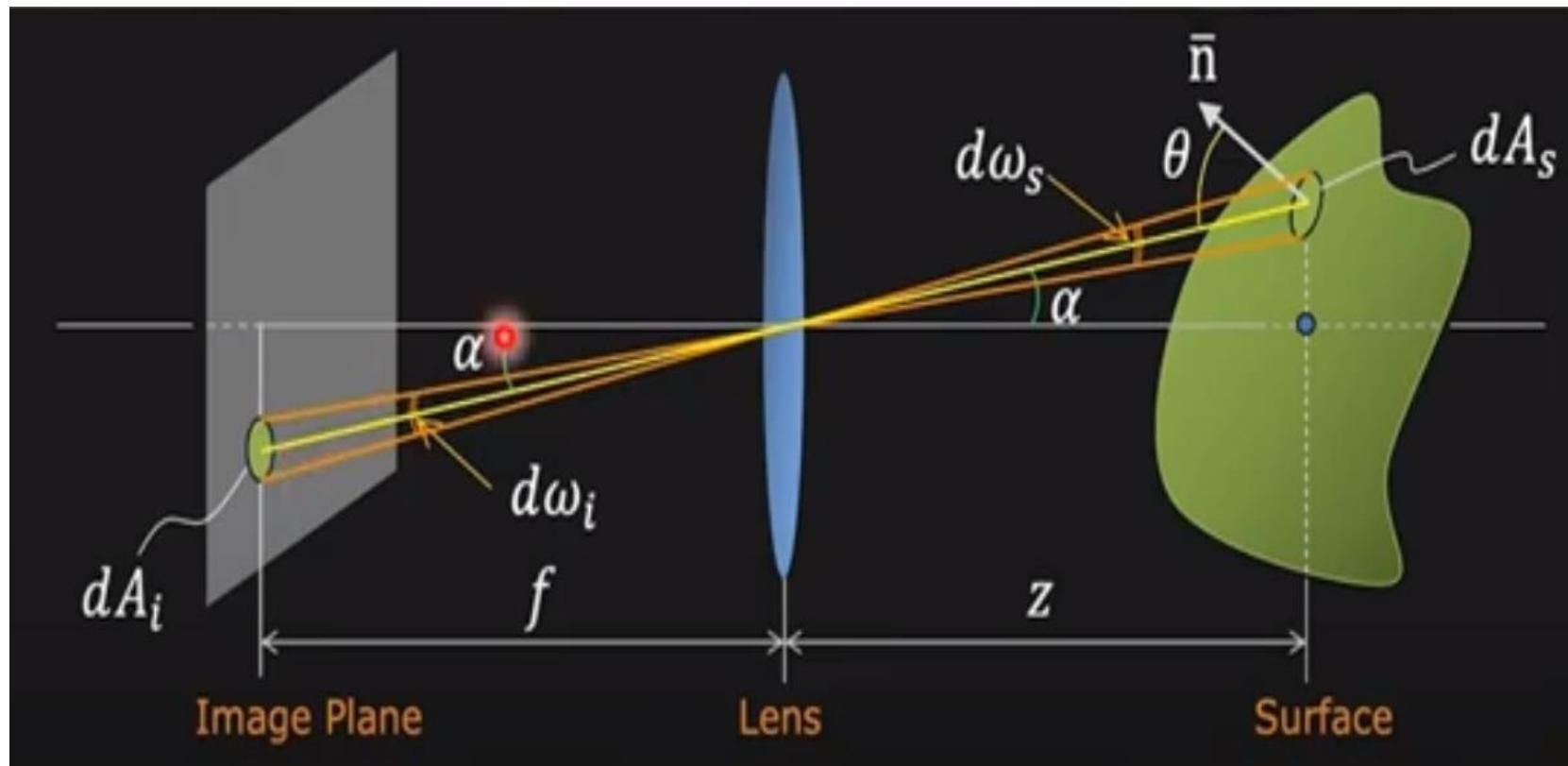
Cont...



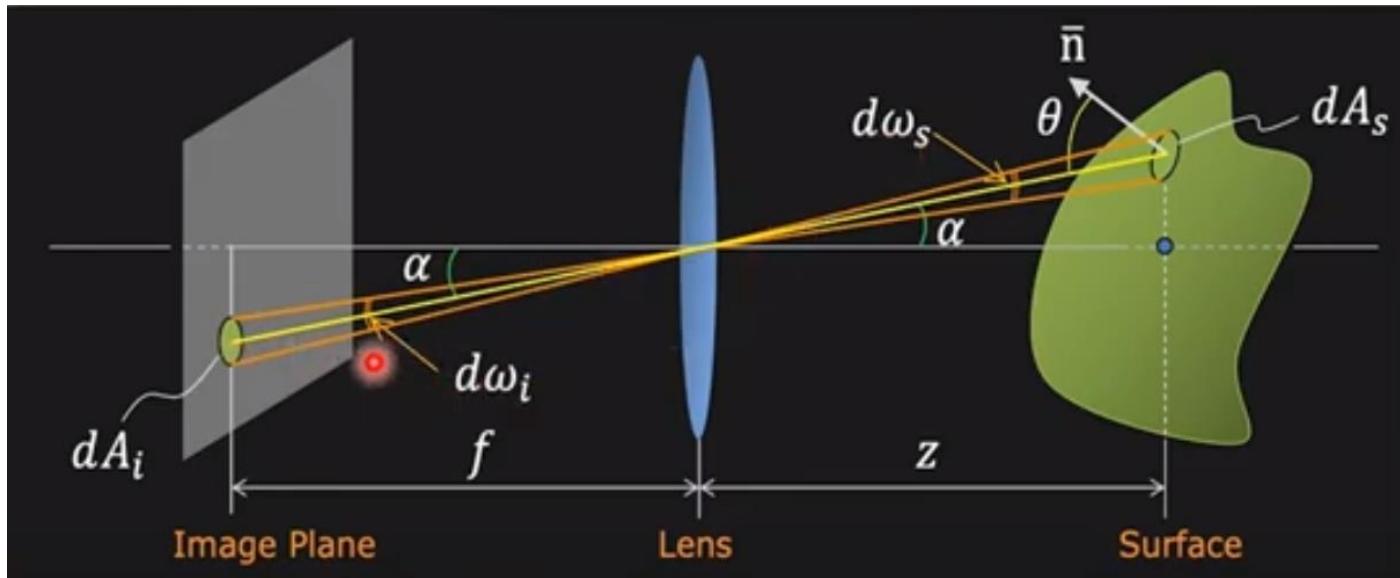
Cont...



Cont...



Cont...

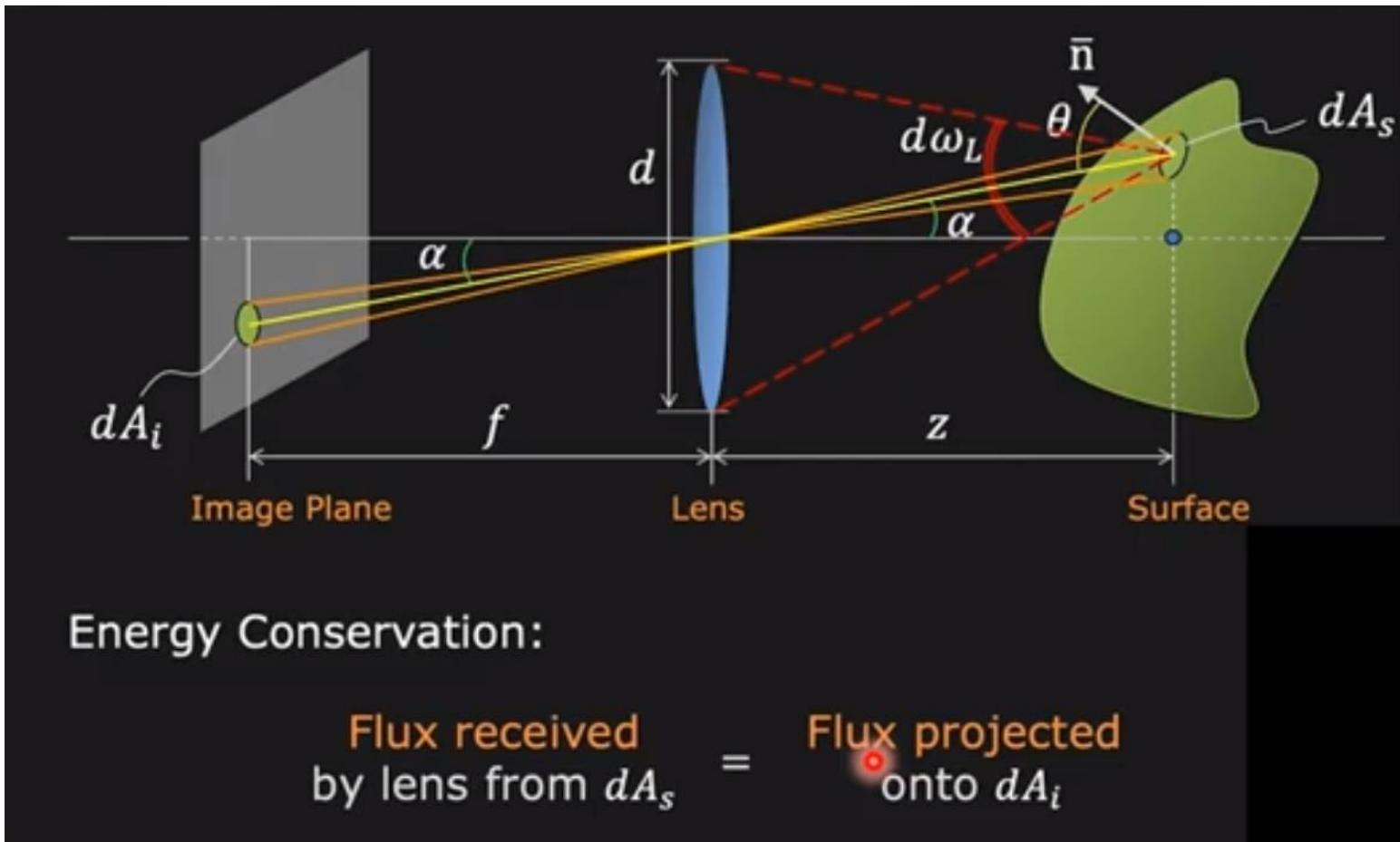


Solid Angles: $d\omega_i = d\omega_s$

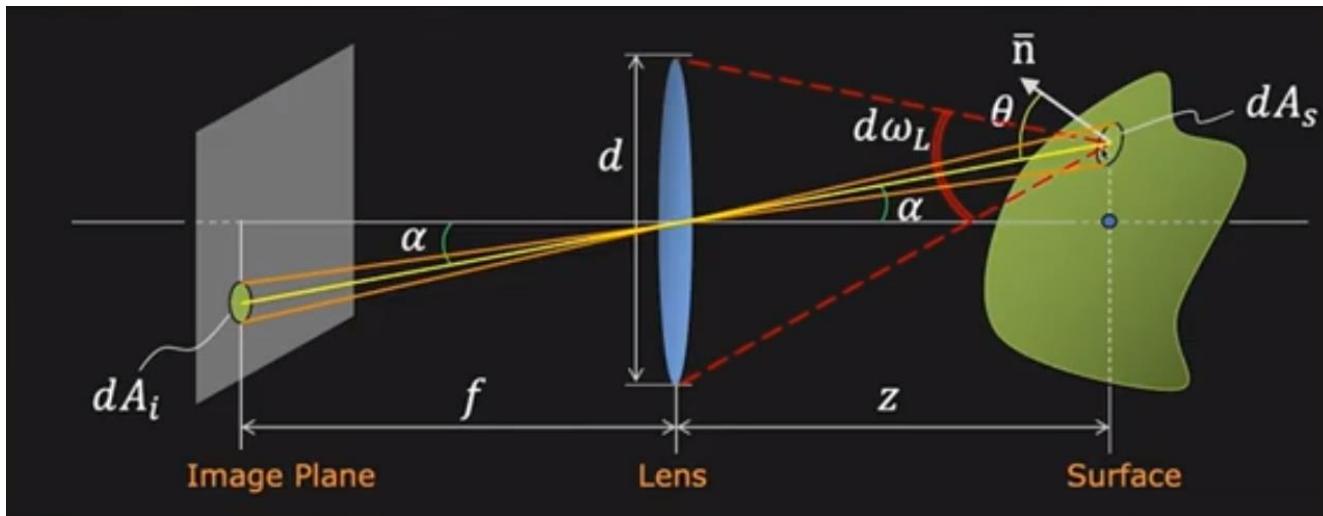
$$\frac{dA_i \cos \alpha}{(f/\cos \alpha)^2} = \frac{dA_s \cos \theta}{(z/\cos \alpha)^2} \quad \rightarrow \quad \boxed{\frac{dA_s}{dA_i} = \frac{\cos \alpha}{\cos \theta} \left(\frac{z}{f}\right)^2}$$

Equation (1)

Cont...



Cont...



Scene Radiance:

$$L = \frac{d^2\Phi}{(dA_s \cos \theta) d\omega_L}$$

Flux received by lens from dA_s

$$d^2\Phi_L = L (dA_s \cos \theta) d\omega_L$$

Equation (3)

Cont...

Equation (1)

$$\frac{dA_s}{dA_i} = \frac{\cos \alpha}{\cos \theta} \left(\frac{z}{f}\right)^2$$

Equation (2)

$$d\omega_L = \frac{\pi d^2}{4} \frac{\cos \alpha}{(z/\cos \alpha)^2}$$

Equation (3)

$$d^2\Phi = L (dA_s \cos \theta) d\omega_L$$

Equation (4)

$$d\Phi = E dA_i$$

$$E = L \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

Cont...

$$E = L \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

Image Irradiance

Scene Radiance

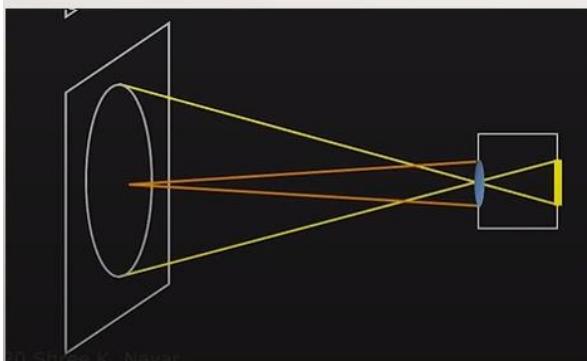
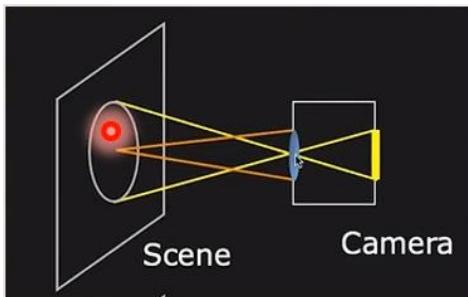
1/Effective F-number
(since f is effective focal length)

- Image Irradiance is proportional to Scene Radiance
- Image brightness falls off from image center as $\cos^4 \alpha$
- For small fields of view, effects of $\cos^4 \alpha$ are small

Cont...

$$E = L \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

- Does image brightness vary with scene depth? Nope



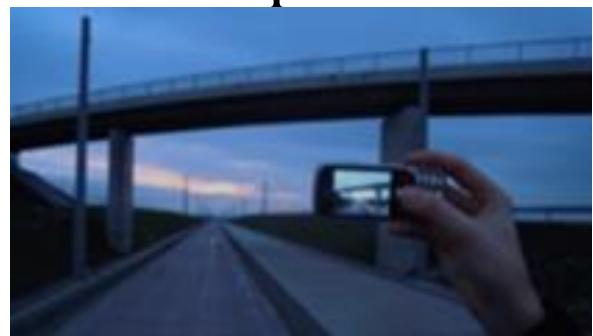
- Larger the scene depth, larger the area of light accumulation.
- Larger the scene depth, smaller the solid angle subtended by each point onto the lens, and hence less light from each point.

Cameras

Cameras

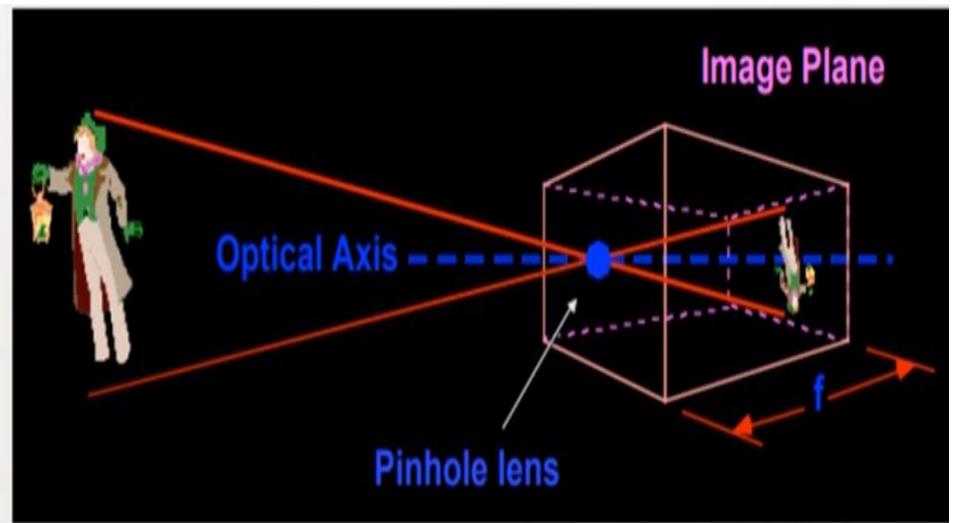
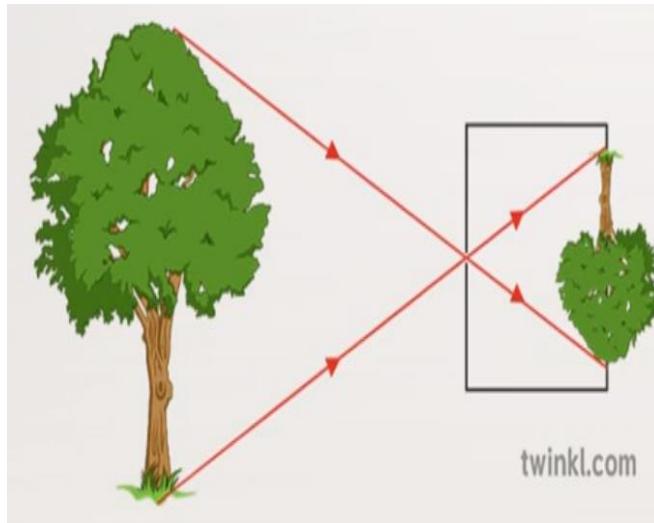


- A camera is an optical instrument used to capture an image. At their most basic, cameras are sealed boxes(the camera body)with a small hole(the aperture)that light in to capture an image on a light-sensitive surface(usually photographic film or a digital sensor).
- Cameras have various mechanisms to control how the light falls onto the light sensitive surface. Lenses focus the light entering the camera, the size of the aperture can be widened or narrowed to let more or less light into the camera, and a shutter mechanism determines the amount of time the photo-sensitive surface is exposed to the light.

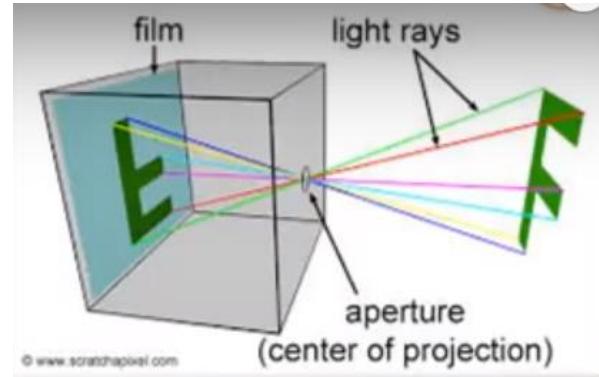


Pinhole Cameras Model

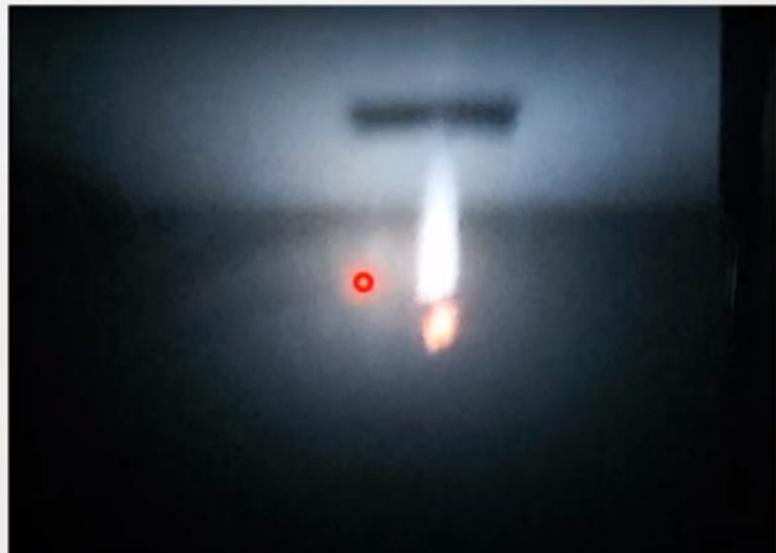
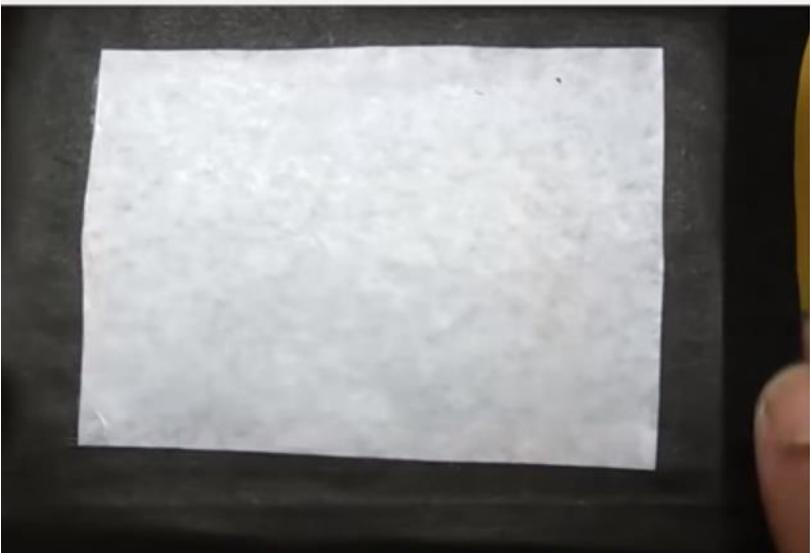
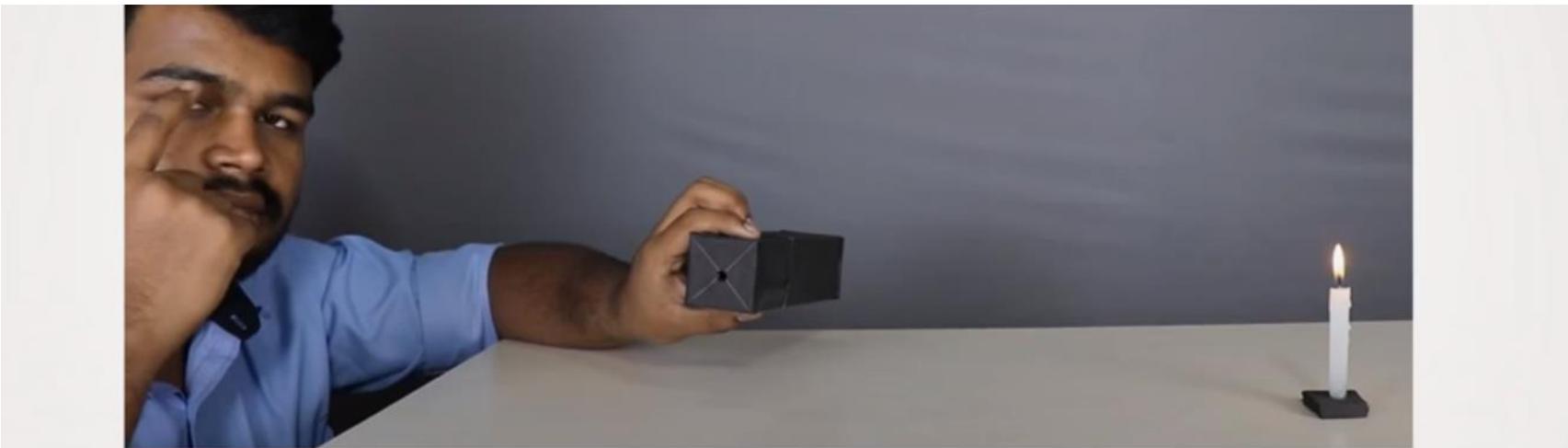
- A pinhole camera is “a simple camera without a lens and with a single small aperture.” Many pinhole cameras are as simple as a box with a hole in the side.



Pinhole Camera Model



- With a pinhole camera, this image is usually upside down and varies in clarity. Some people use a pinhole camera to study the movement of the sun over time(Solargraphy). A type of pinhole camera is often used to view an eclipse. Another type of pinhole camera, the camera obscure, was once used by artists.
- The device allowed the artist to view a scene through a different perspective. The artist would point the lens of the camera at the still life scene they wanted to paint. The camera would frame the image in smaller perspective thus allowing the artist to see the scene as it might appear. The person using a camera obscura might even trace the image on a piece of paper and achieve a very accurate copy of the scene.



Projections



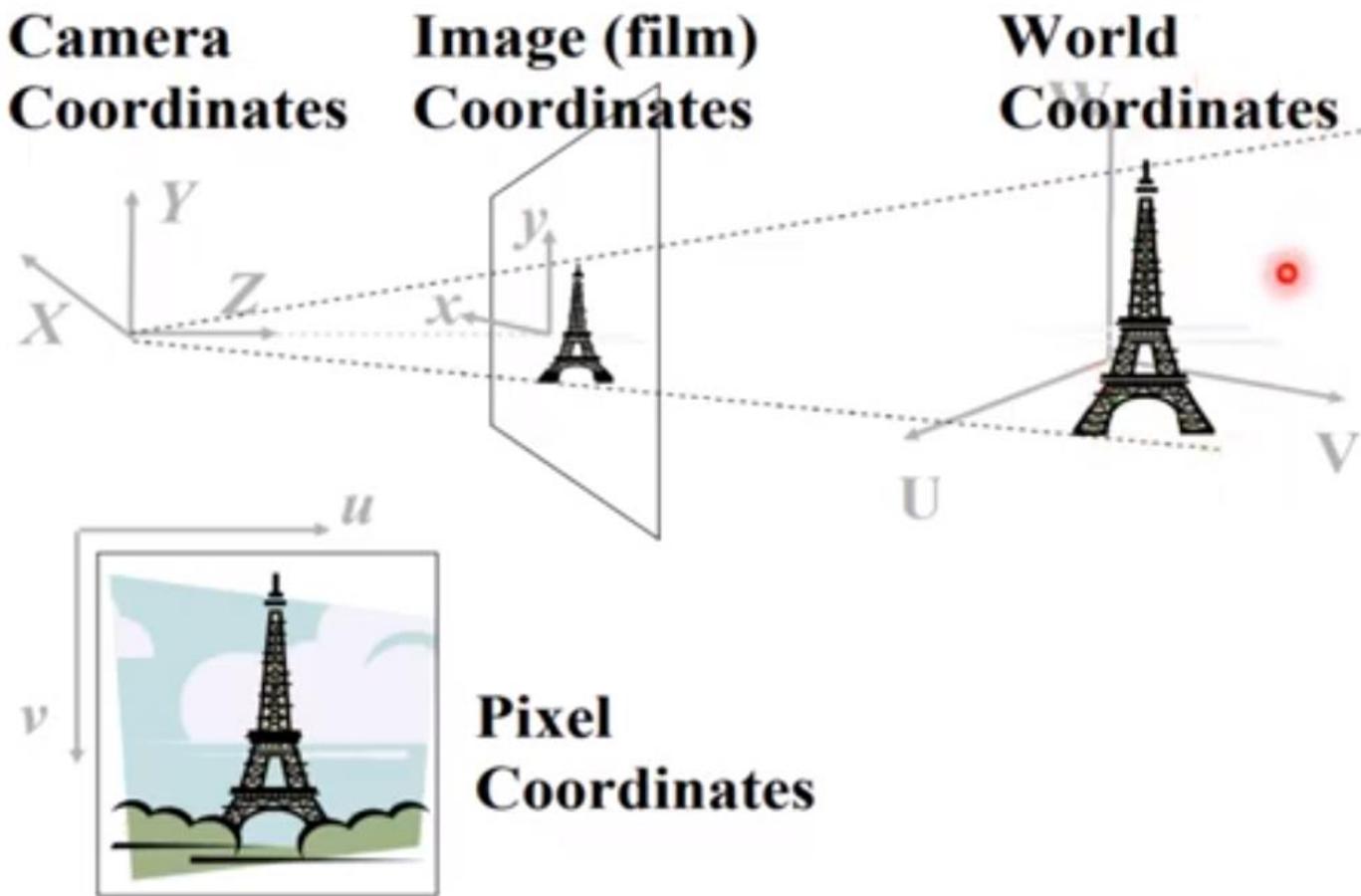
Projection of Image

Actual Image



CoolOpticalIllusions.com

Projections

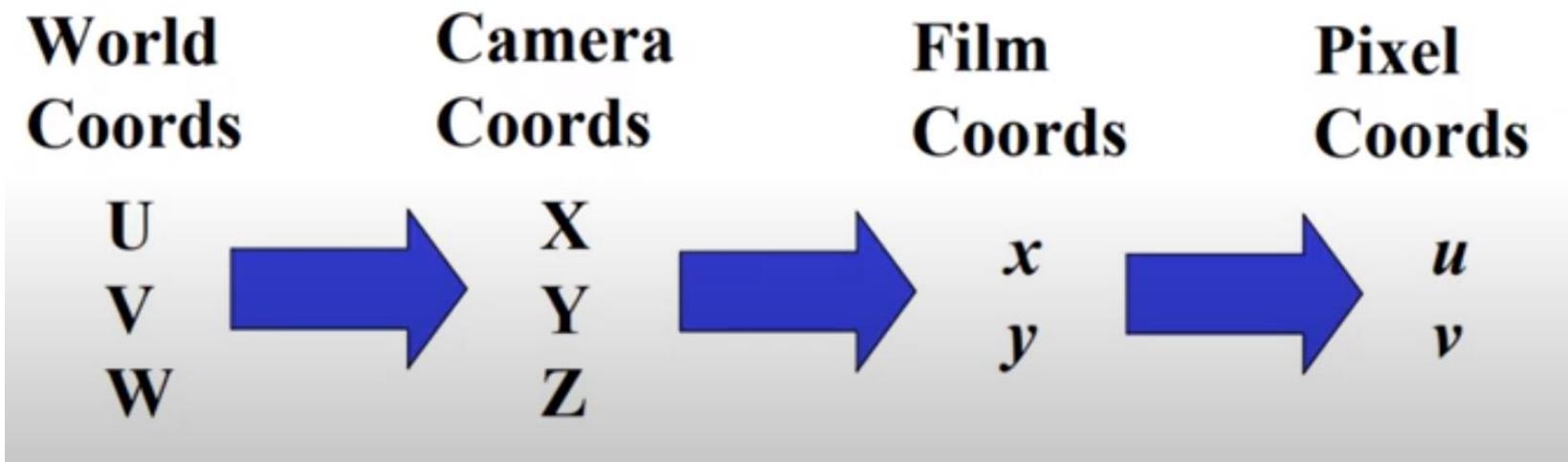


Cont...

1. Forward Projection
2. Backward Projection

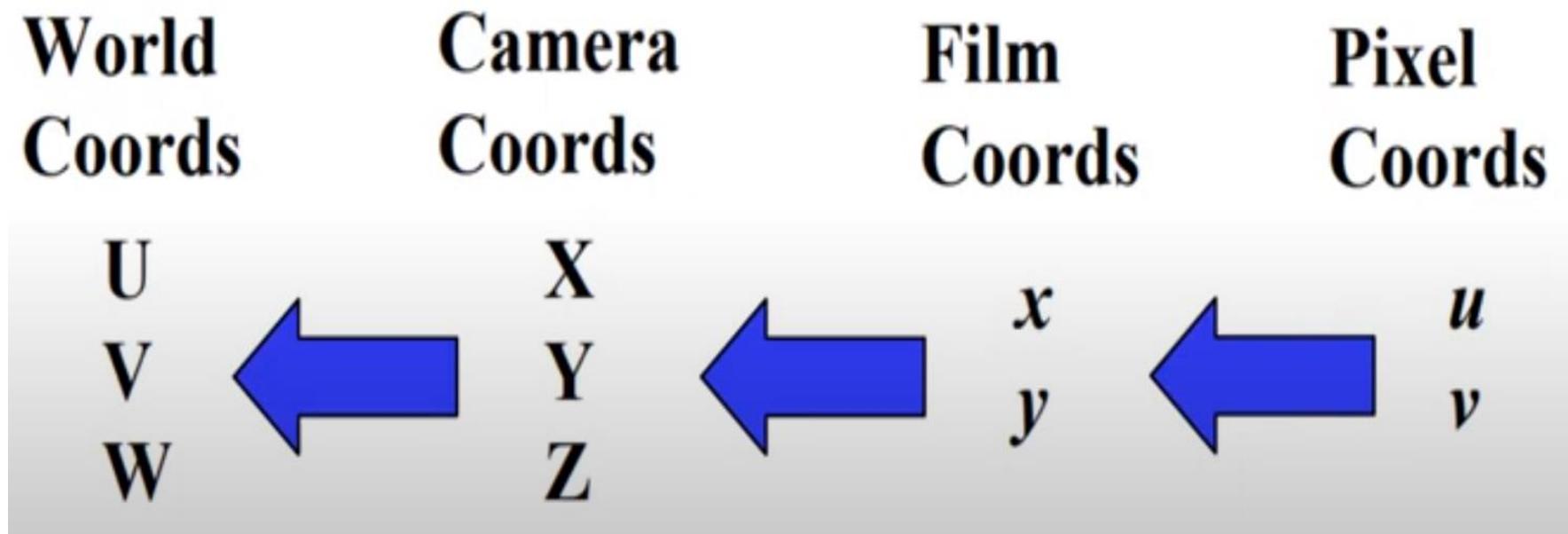
1. Forward Projection

- We want to mathematical model to describe how 3D world points get projected into 2D pixel coordinates.

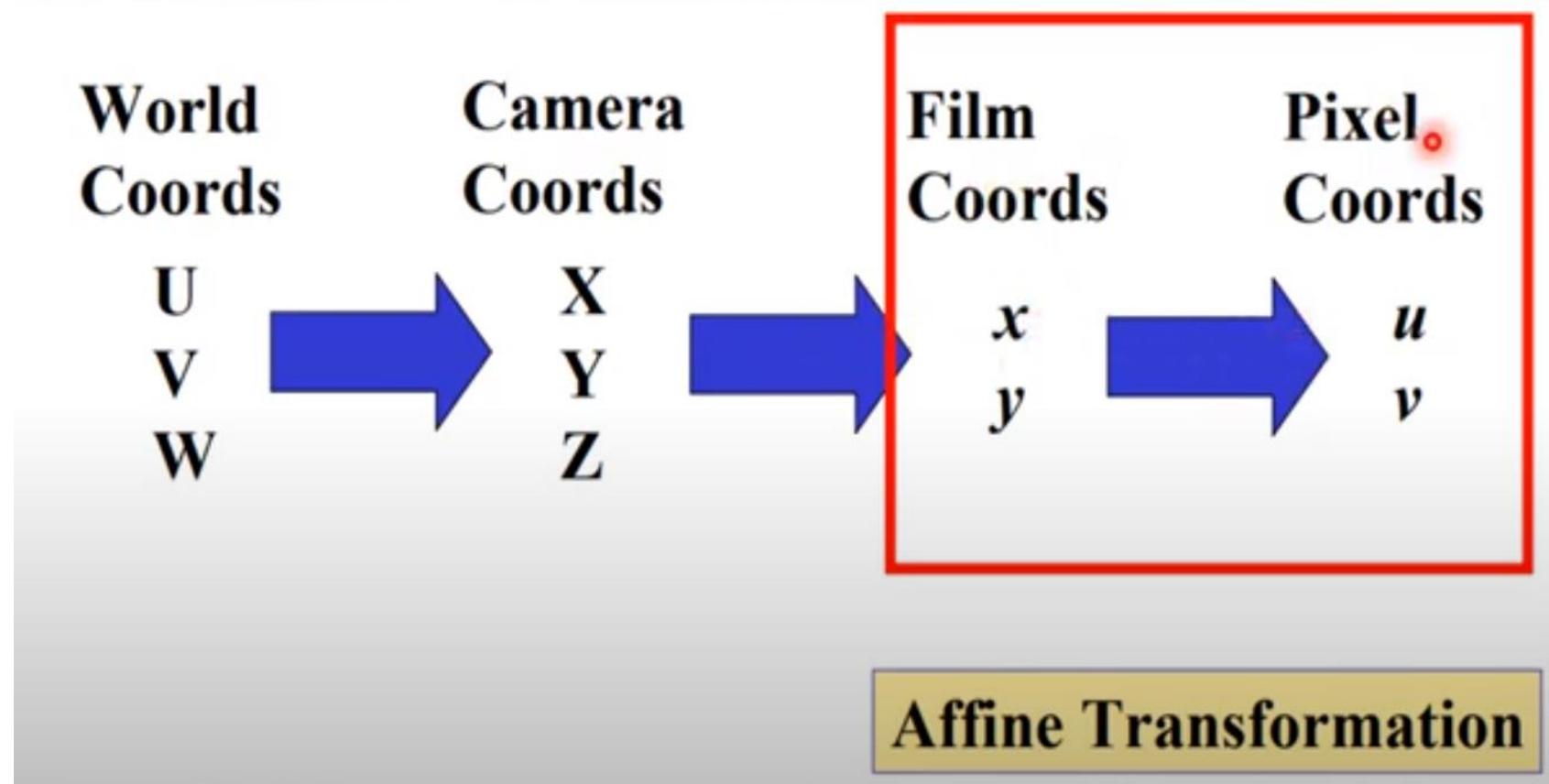


2. Backward Projection

- Recover 3D scene structure from image(via stereo or motion)



Intrinsic Camera Parameters



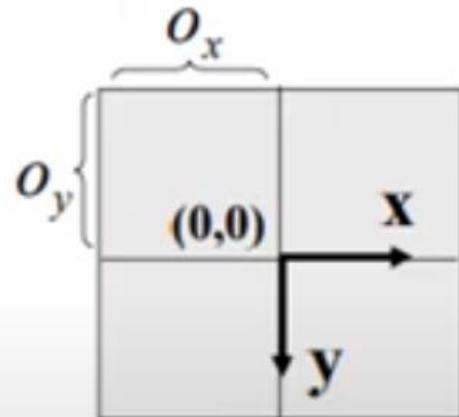
Cont...

- Describes coordinate transformation between film coordinates(projected image) and pixel array.
- Film cameras: scanning/digitization
- CCD cameras: grid of photo sensors

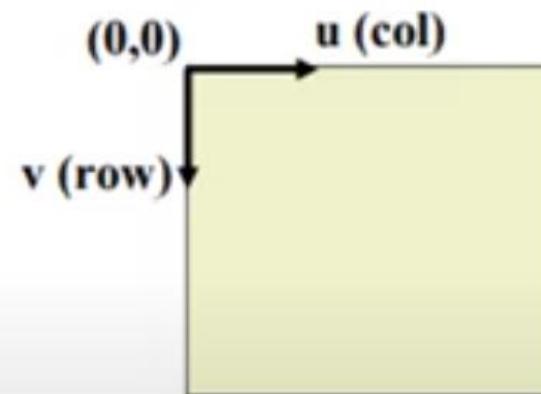
Intrinsic parameters(offsets)

film plane

(projected image)



pixel array

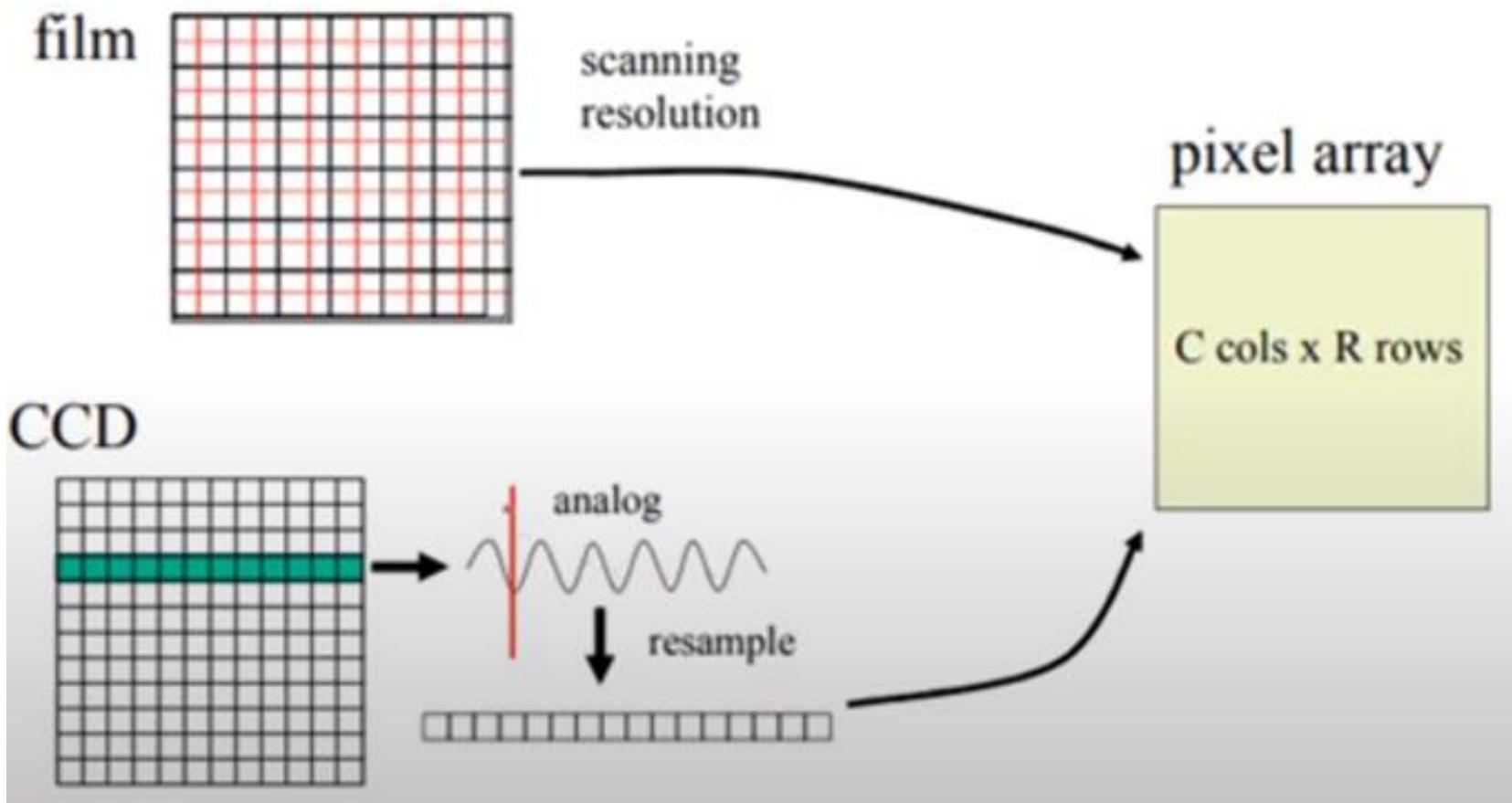


$$u = f \frac{X}{Z} + o_x \quad v = f \frac{Y}{Z} + o_y$$

o_x and o_y called image center or principle point

Intrinsic parameters(Scales)

sampling determines how many rows/cols in the image



Effective Scales: S_x and S_y

$$u = \frac{1}{s_x} f \frac{X}{Z} + o_x \quad v = \frac{1}{s_y} f \frac{Y}{Z} + o_y$$

Note, since we have different scale factors in x and y,
we don't necessarily have square pixels!

Aspect ratio is s_y / s_x

Thank you!!!

Unit:2

Image Processing

Prepared By:
Prof. Janki Patel
Department of IT
SPCE, Bakrol

Content

- Pixel transforms,
- Color transforms,
- Histogram processing and equalization,
- Filtering
- Convolution
- Fourier transformation and its applications in sharpening
- Blurring and noise removal

What is the image processing?

“One picture is worth more than ten thousand words”

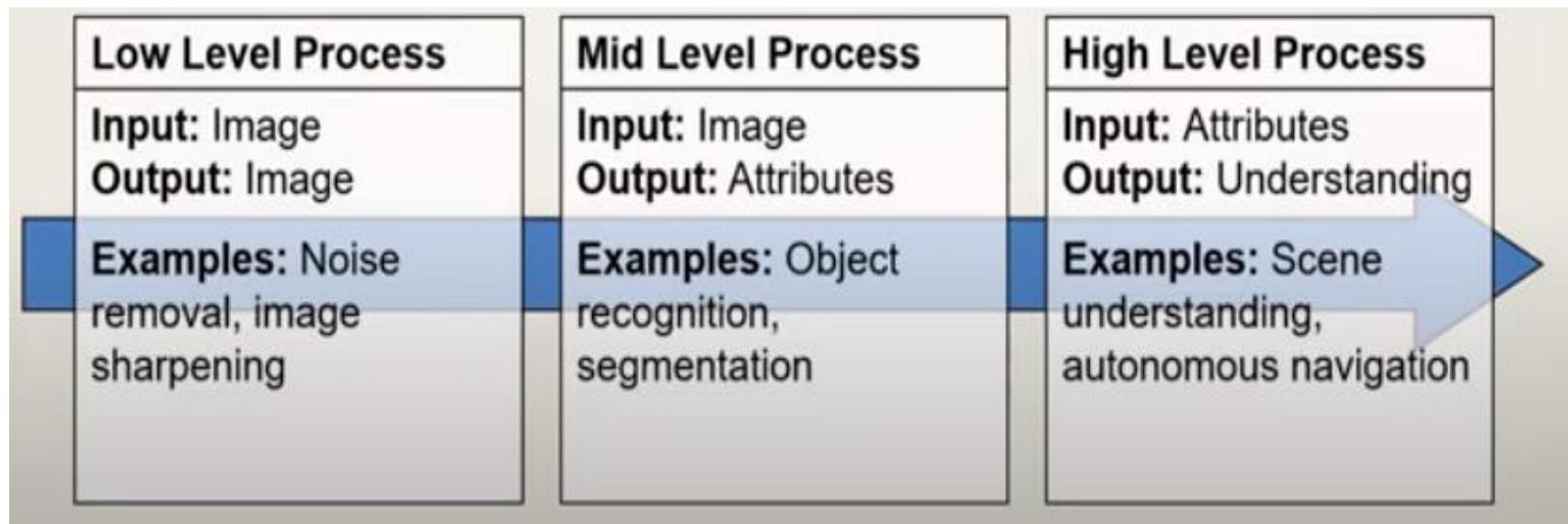
Anonymous

What is Digital Image Processing?

- Digital image processing focuses on two major tasks
 - Improvement of pictorial information for human interpretation
 - Processing of image data for storage, transmission and representation for autonomous machine perception.

Cont...

- The continuum from image processing to computer vision can be broken up into low, mid and high-level processes.



History of Digital Image Processing

- Early 1920s: One the first applications of digital imaging was in the news paper industry.
- The Bart lane cable picture transmission service
- Image were transferred by submarine cable between London and New York.
- Pictures were coded for cable transfer and reconstructed at the receiving end on a telegraph printer



Cont...

- Mid to late 1920s: Improvements to Bartlane system resulted in higher quality images
 - New reproduction processes based on photographic techniques
 - Increased number of tones in reproduced images



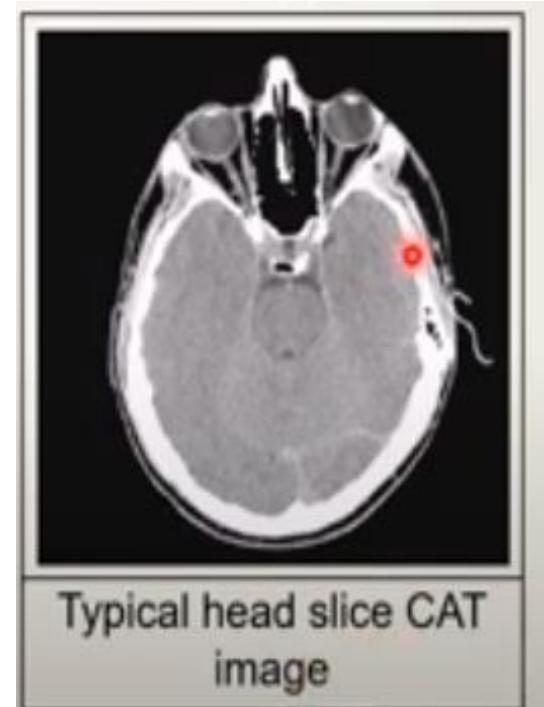
Cont...

- 1960s: Improvements in computing technology and onset of the space race led to a surge of work in digital image processing.
- 1964: Computers used to improve the quality of image of the moon taken by the Ranger 7 probe
- Such techniques were used in other space missions including the Apollo landings



Cont...

- 1970s: Digital image processing begins to be used in medical applications.
 - 1979: Sir Godfrey N. Hounsfield & Prof. Allan M. Cormack share the Nobel Prize in medicine for the invention of tomography, the technology behind Computerized Axial Tomography (CAT)scans

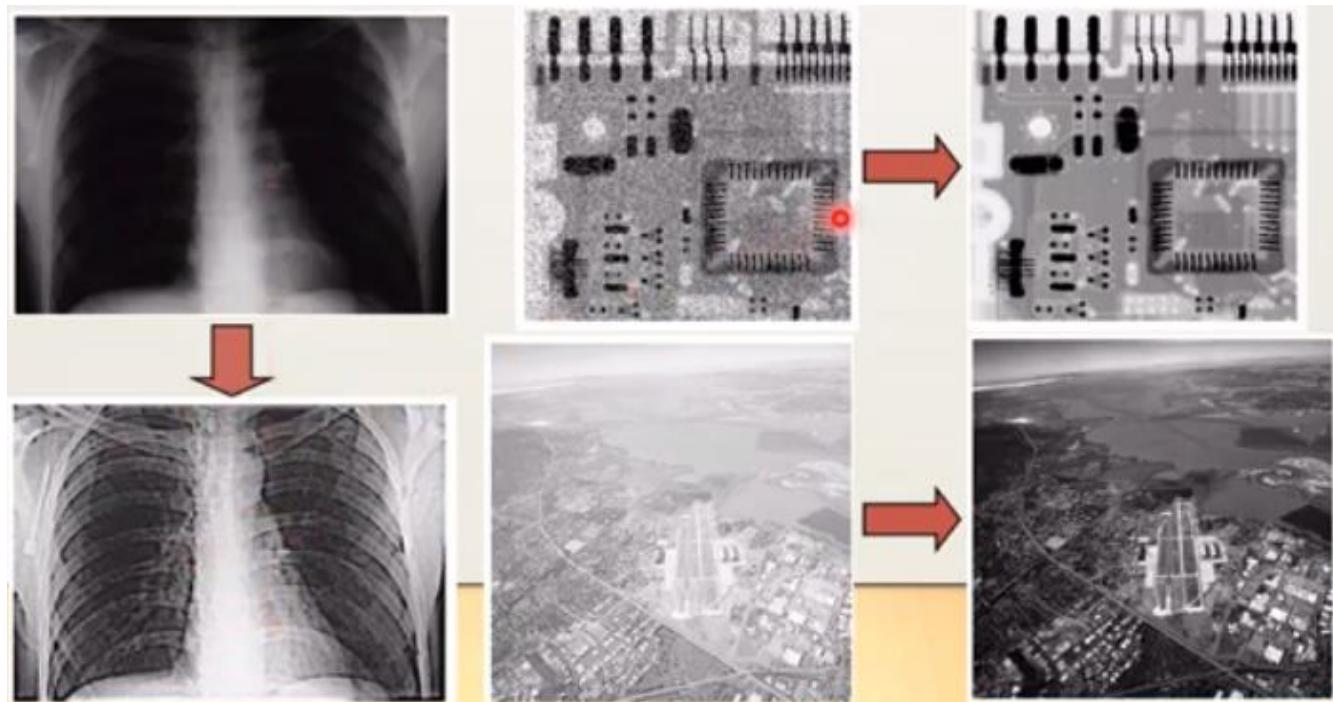


Cont...

- 1980s-Today: The use of digital image processing techniques has exploded and they are now used for all kinds of tasks in all kinds of areas.
 - Image enhancement/restoration
 - Artistic effects
 - Medical visualization
 - Industrial inspection
 - Law enforcement
 - Human computer interfaces

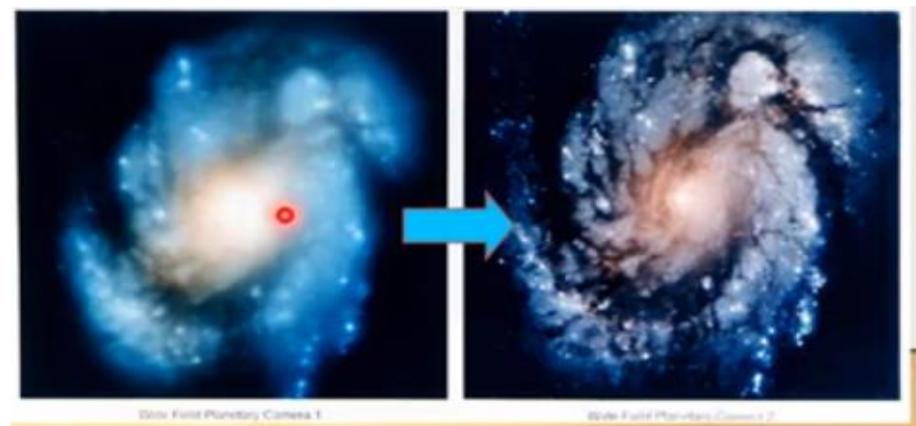
Examples: Image Enhancement

- One of the most common of DIP techniques: improve quality, remove noise etc.



Examples: The Hubble Telescope

- Launched in 1990 the Hubble telescope can take images of very distant objects.
- However, an incorrect mirror made many of Hubble's images useless
- Image processing techniques were used to fix this



Examples: Artistic Effects

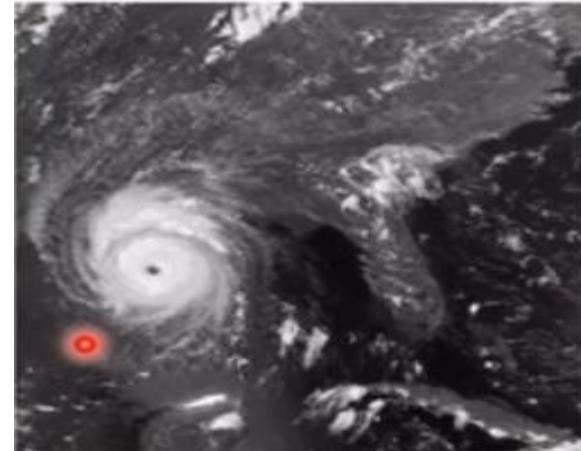
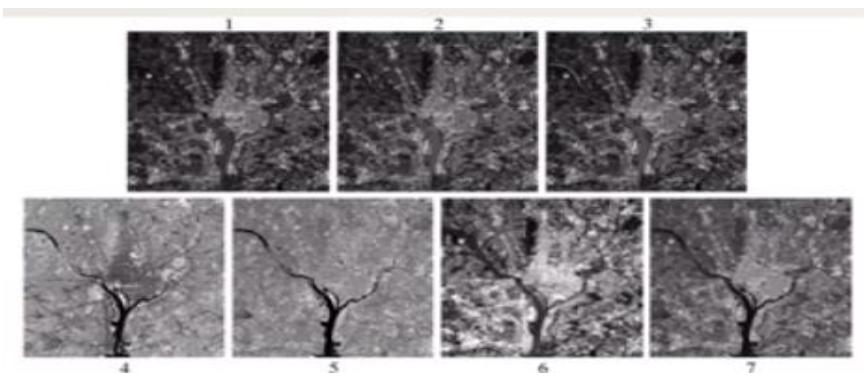
- Artistic effects are used to make images more visually appealing, to add special effects and to make composite images.



Examples: GIS

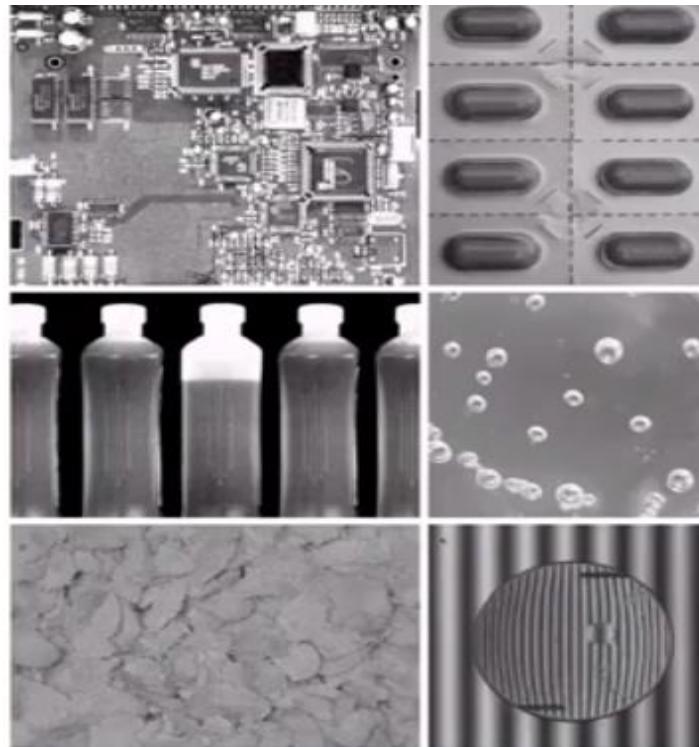
- **Geographic Information System**

- Digital image processing techniques are used extensively to manipulate satellite imagery
- Terrain classification
- Meteorology



Examples: Industrial Inspection

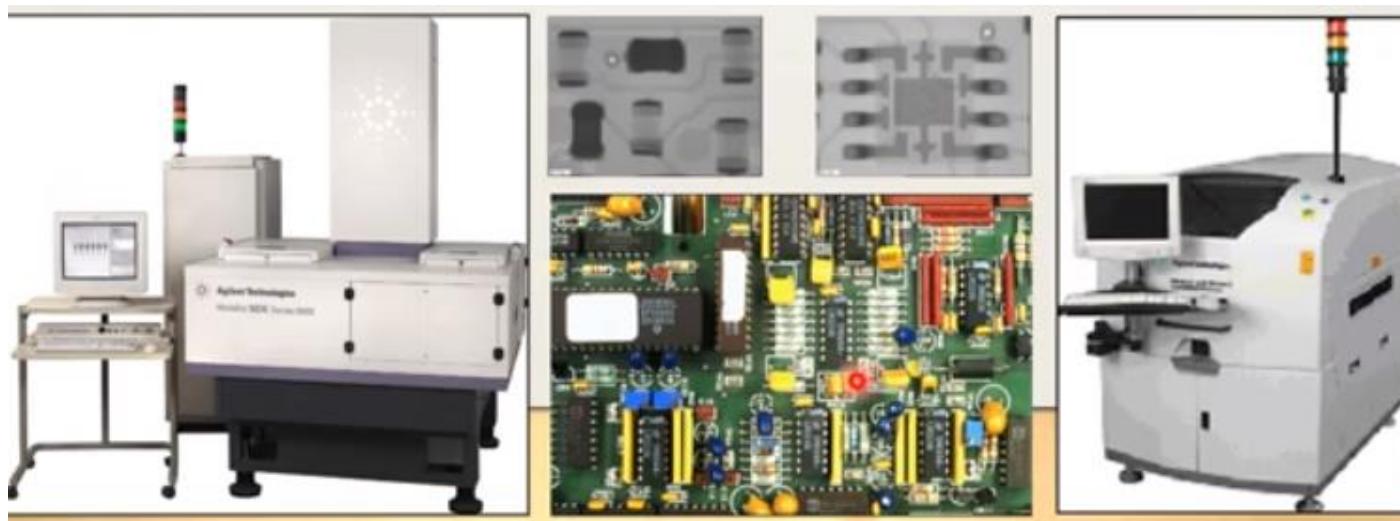
- Human operators are expensive, slow and unreliable
- Make machines do the job instead
- Industrial vision systems are used in all kinds of industries



Examples: PCB Inspection

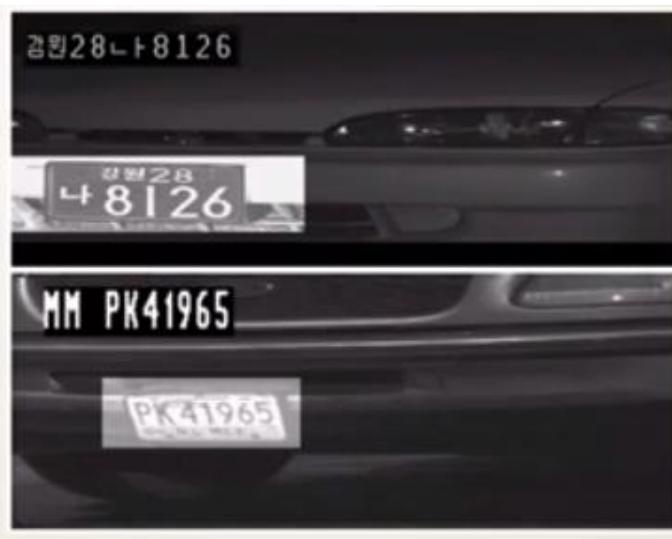
Printed Circuit Board(PCB)inspection

- Machine inspection is used to determine that all components are present and that all solder joints are acceptable
- Both conventional imaging and x-ray imaging are used



Examples: Law Enforcement

- Image processing techniques are used extensively by law enforcers
 - Number plate recognition for speed cameras/automated toll systems
 - Fingerprint recognition
 - Enhancement of CCTV images



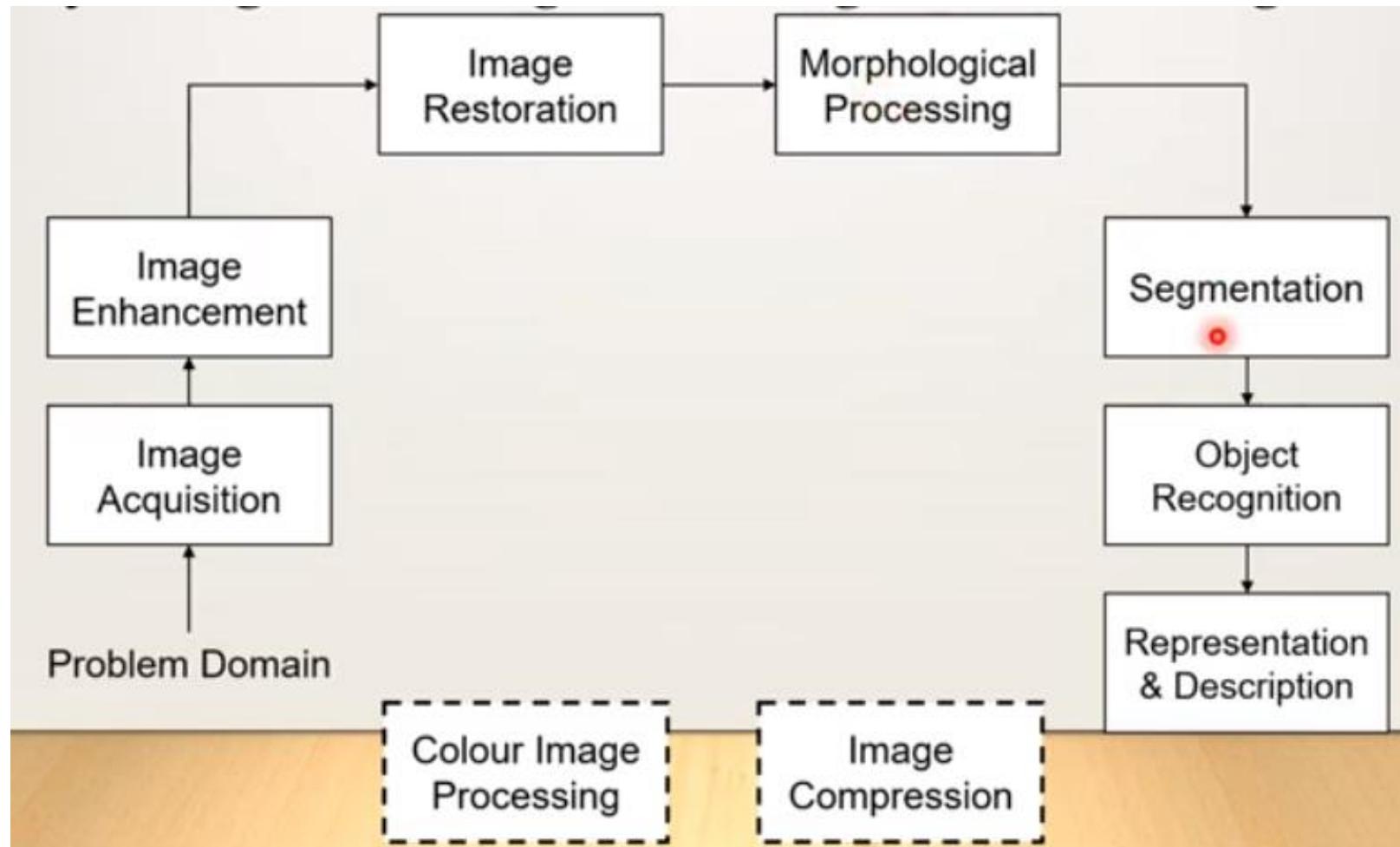
Examples: HCI

- Try to make human computer interfaces more natural
 - Face recognition
 - Gesture recognition

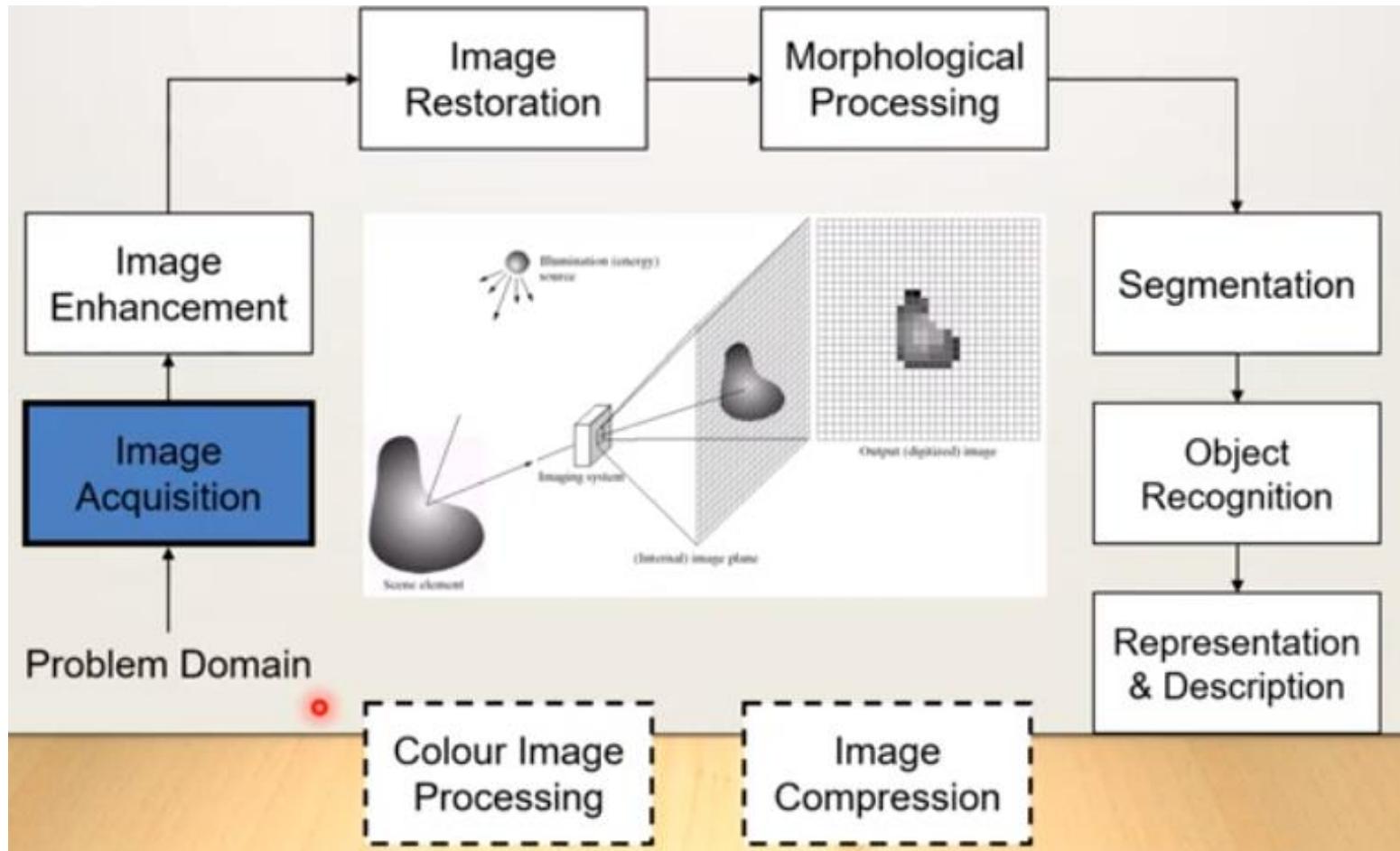


Key Stages in Digital Image Processing

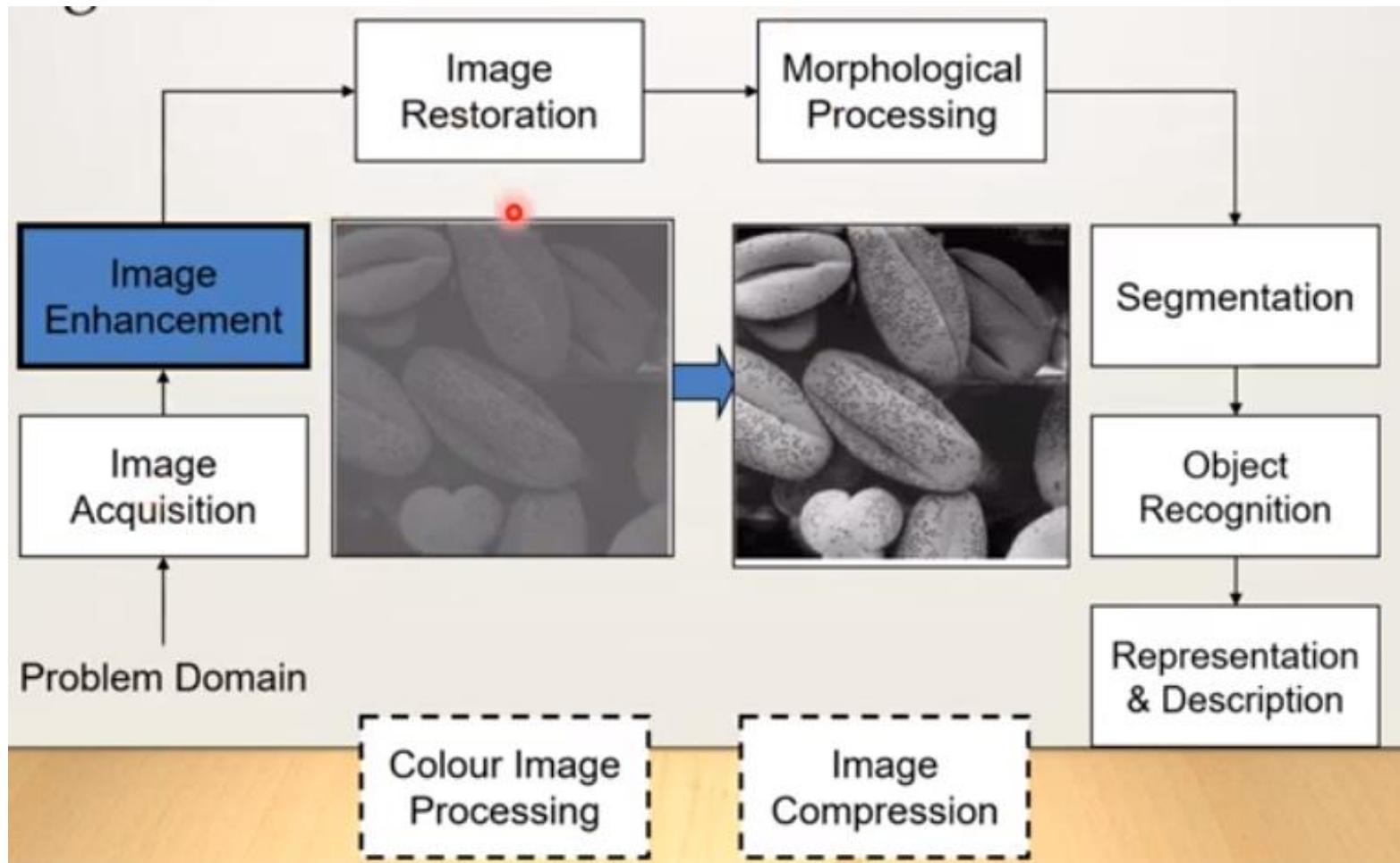
Key Stages in Digital Image Processing



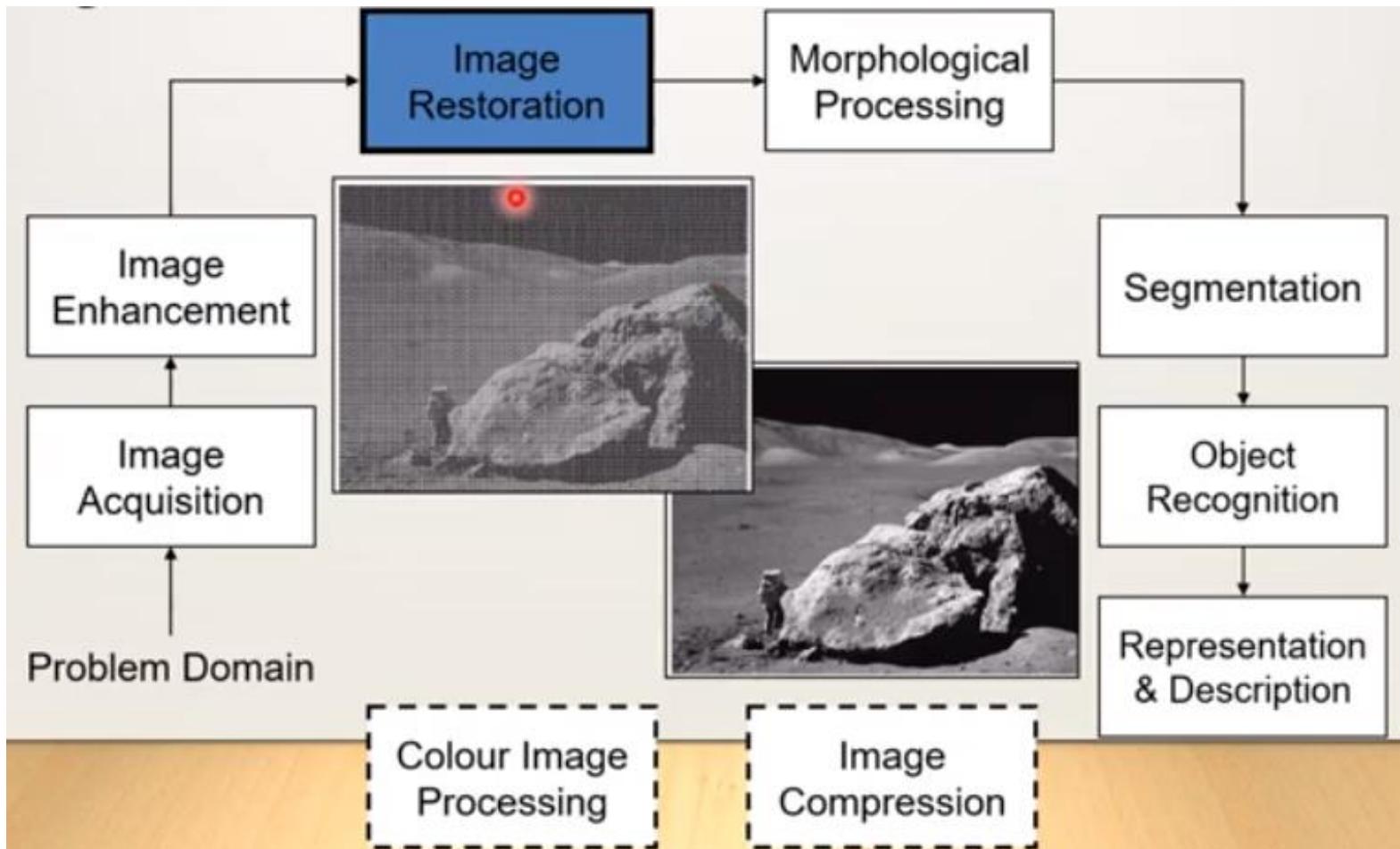
Key Stages in Digital Image Processing: Image Acquisition



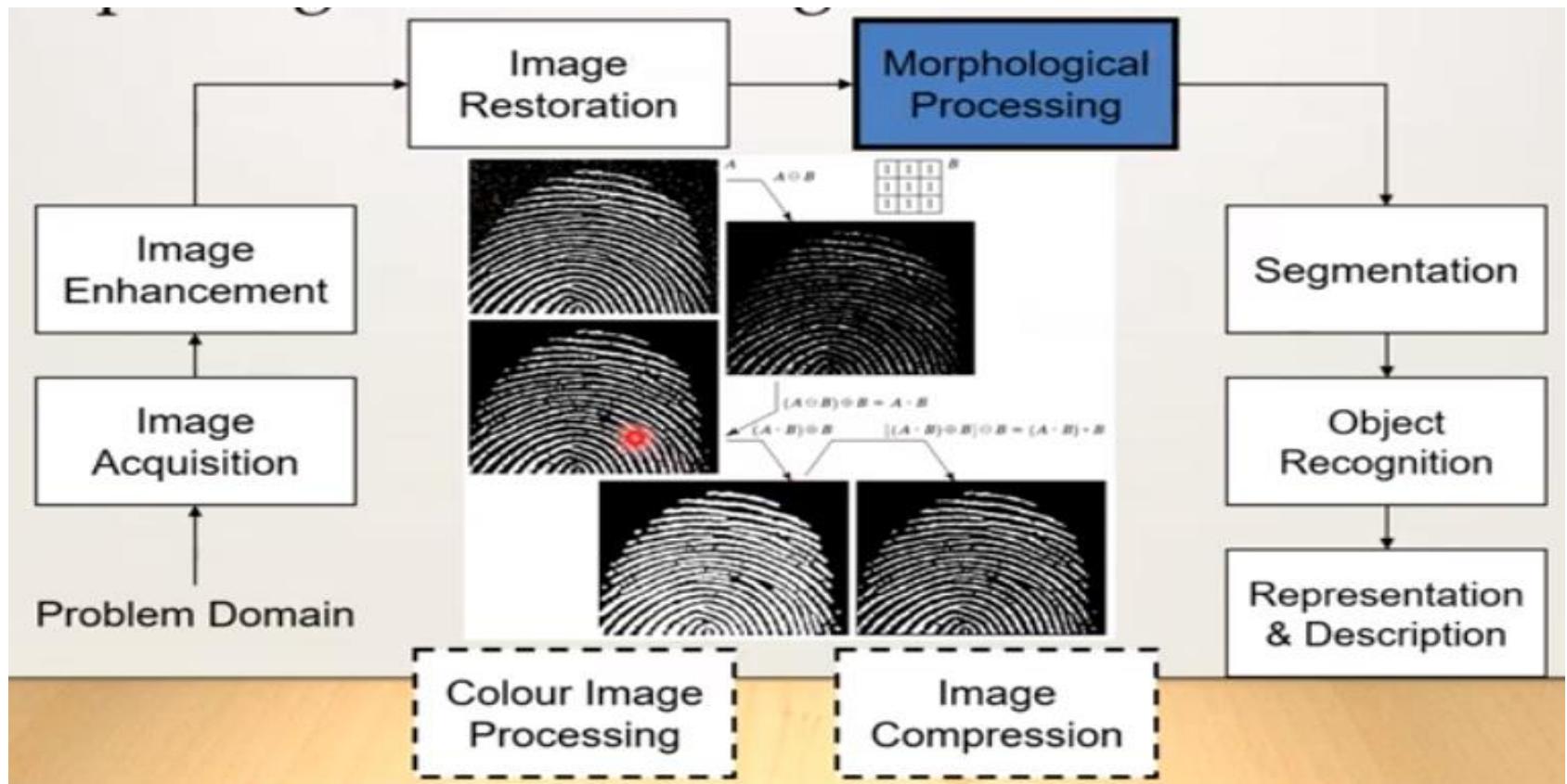
Key Stages in Digital Image Processing: Image Enhancement



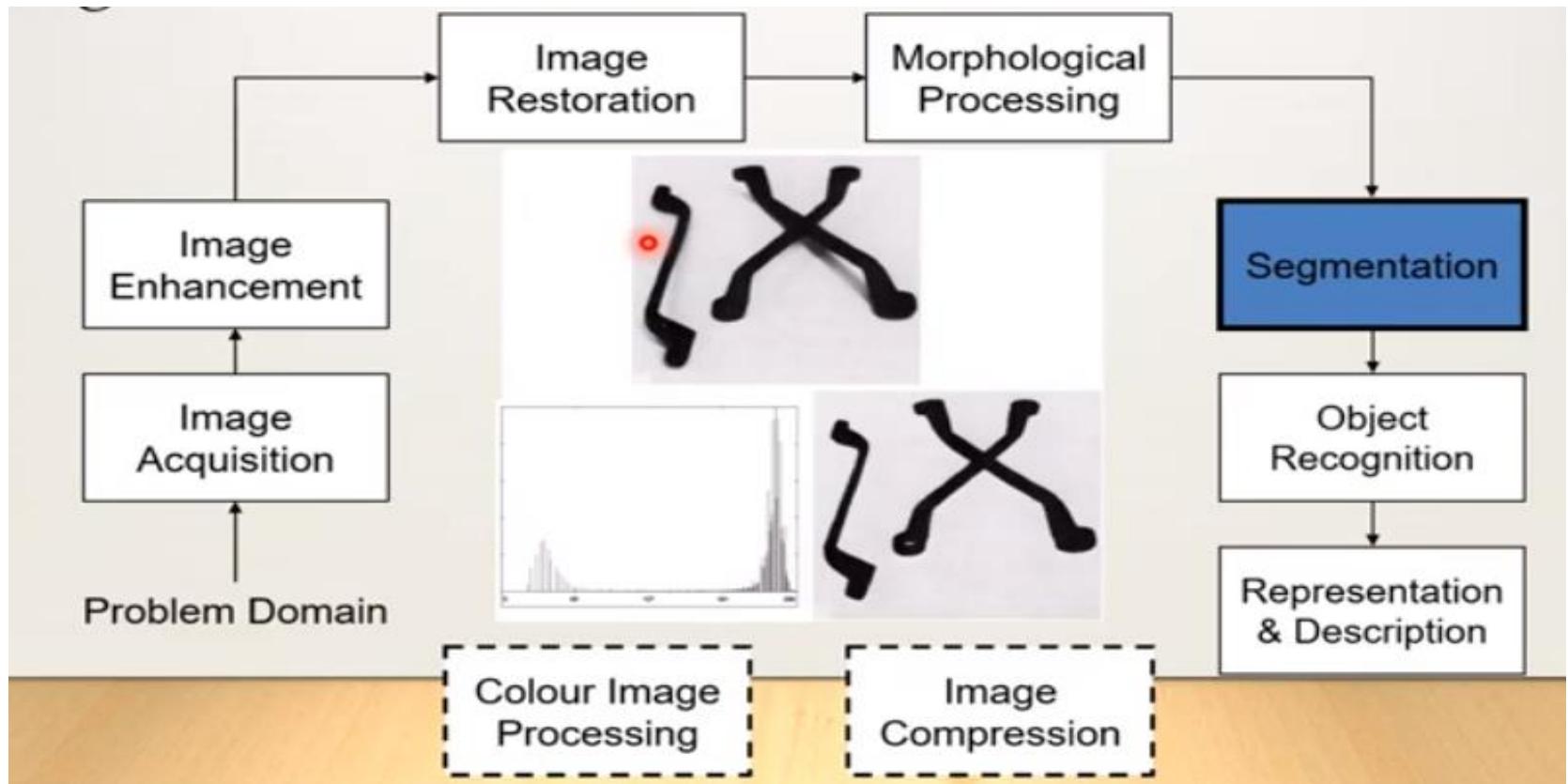
Key Stages in Digital Image Processing: Image Restoration



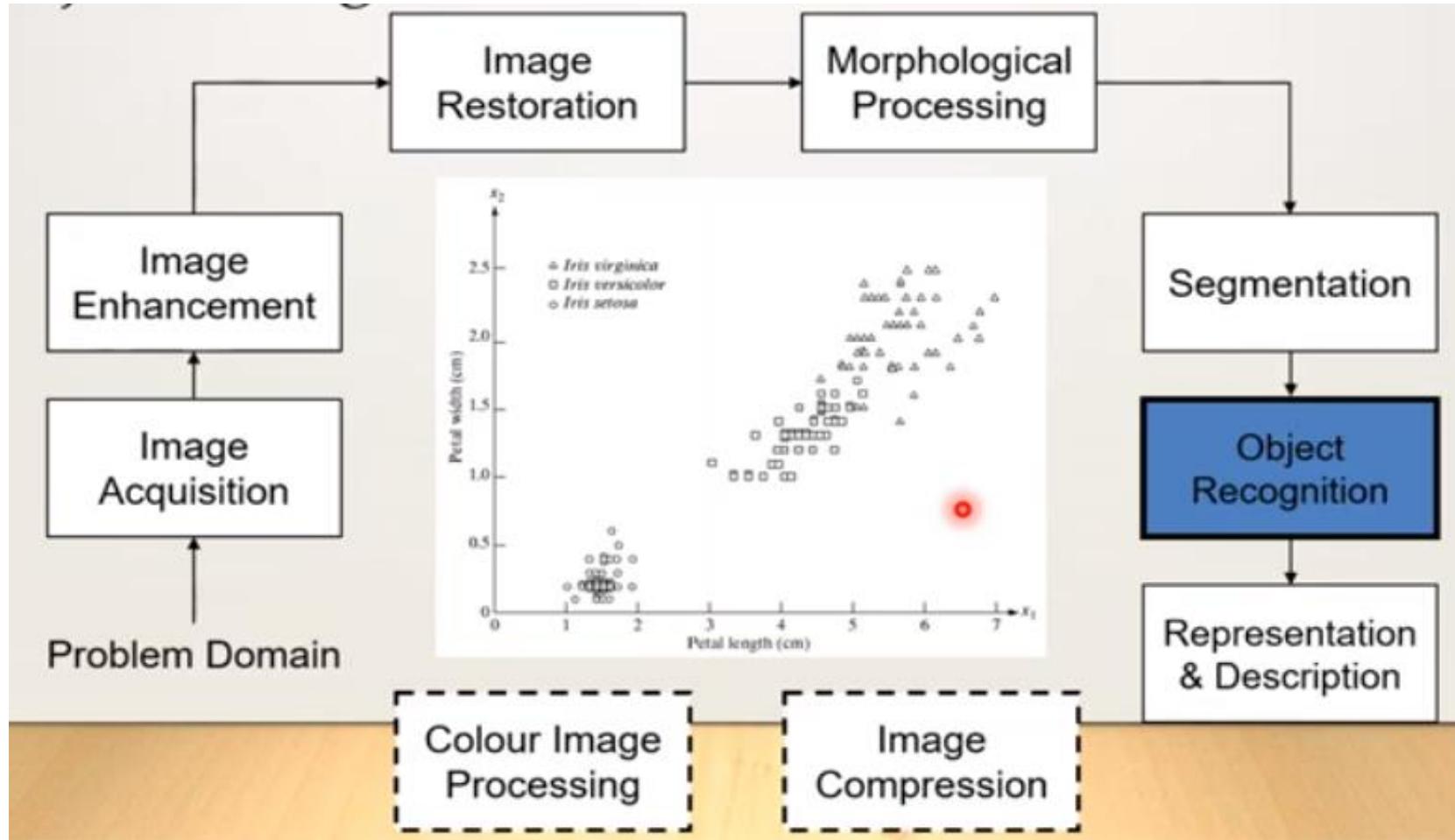
Key Stages in Digital Image Processing: Morphological Processing



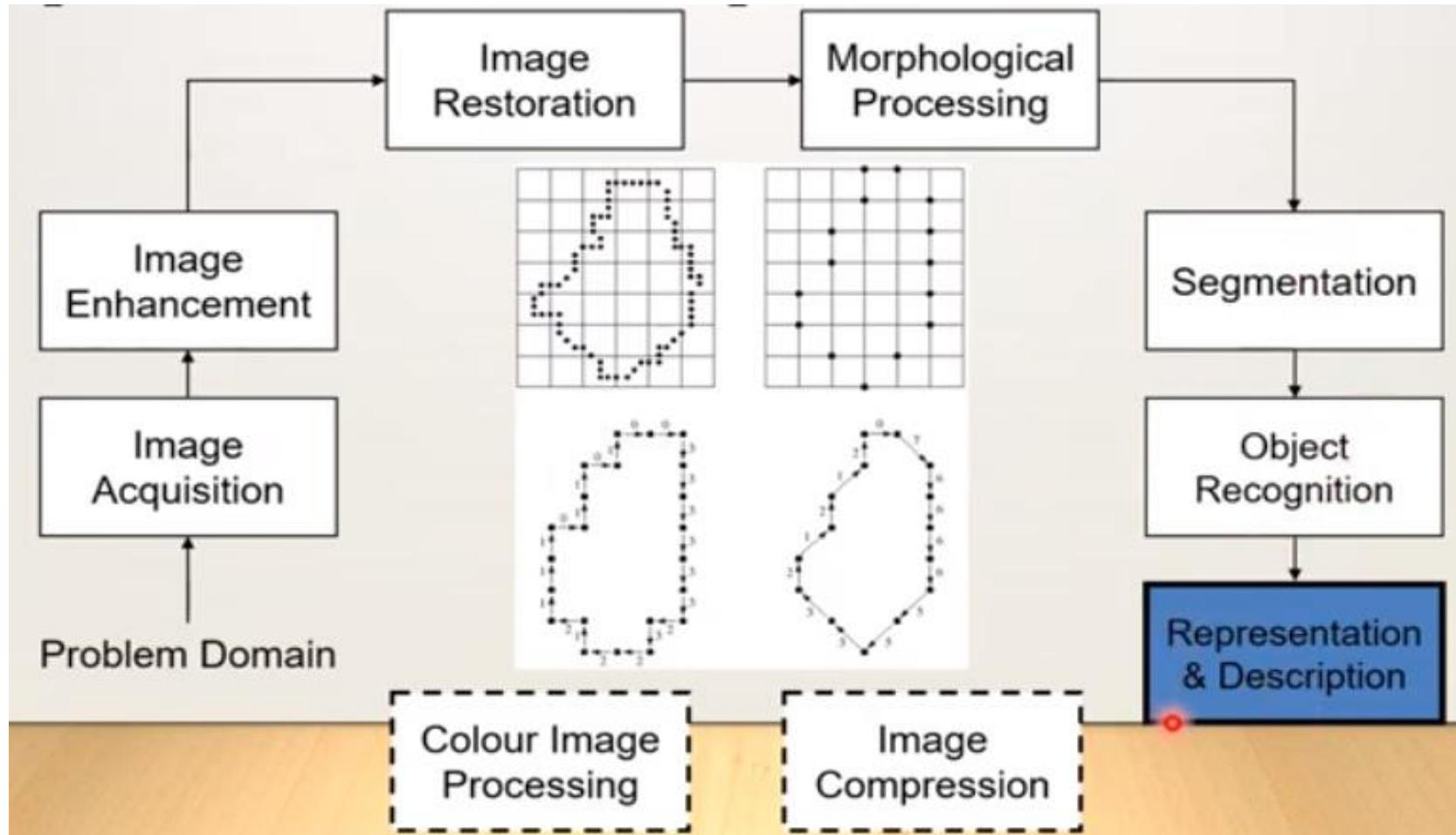
Key Stages in Digital Image Processing: Segmentation



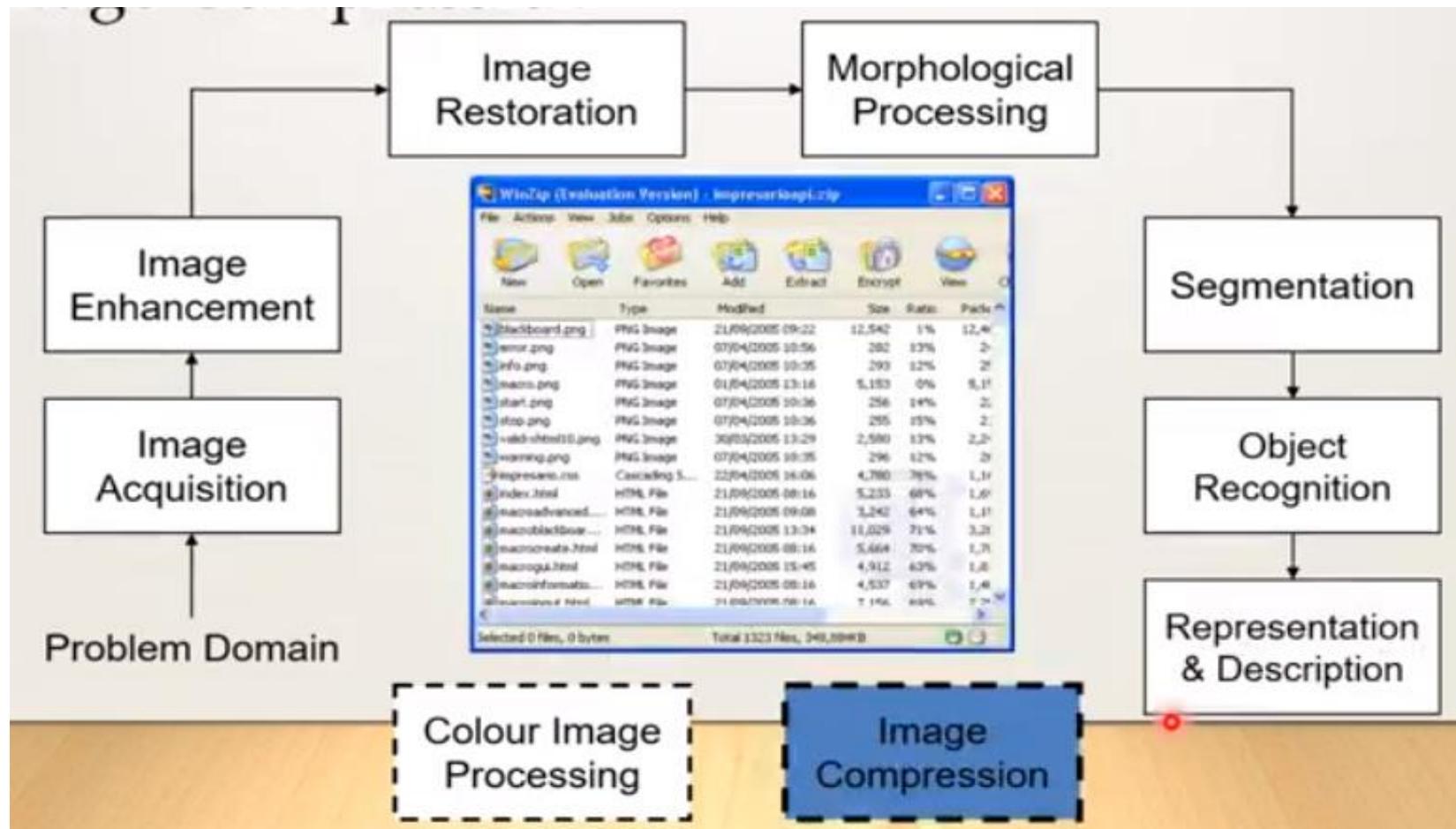
Key Stages in Digital Image Processing: Object Recognition



Key Stages in Digital Image Processing: Representation & Description



Key Stages in Digital Image Processing: Image Compression



Key Stages in Digital Image Processing: Color Image Processing

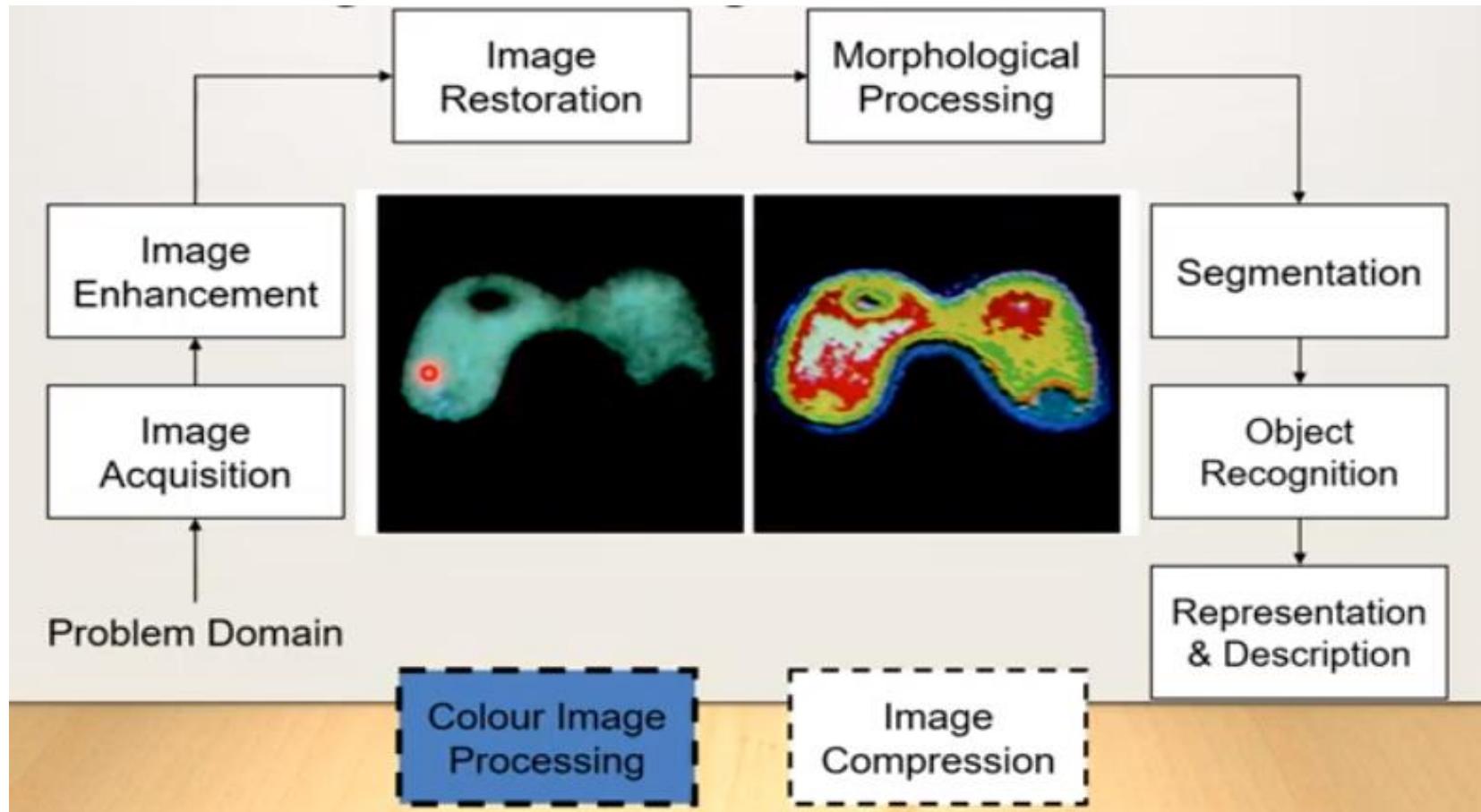


Image Enhancement

What is Image Enhancement?

- Image enhancement is the process of making images more useful
- The reasons for doing this include:
 - Highlighting interesting detail in images
 - Removing noise from images
 - Making image more visually appealing

Image Enhancement Examples

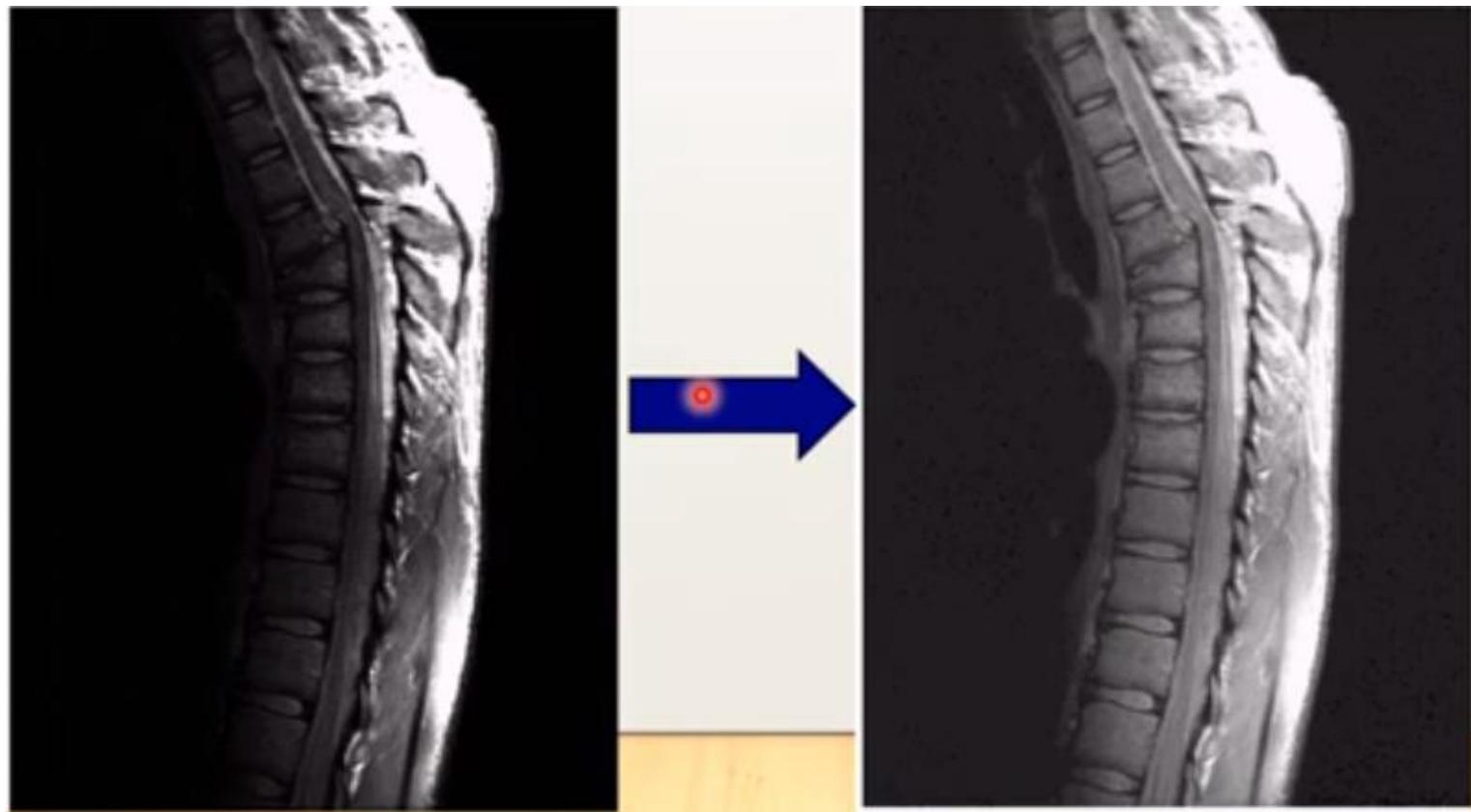


Image Enhancement Examples(Cont...)

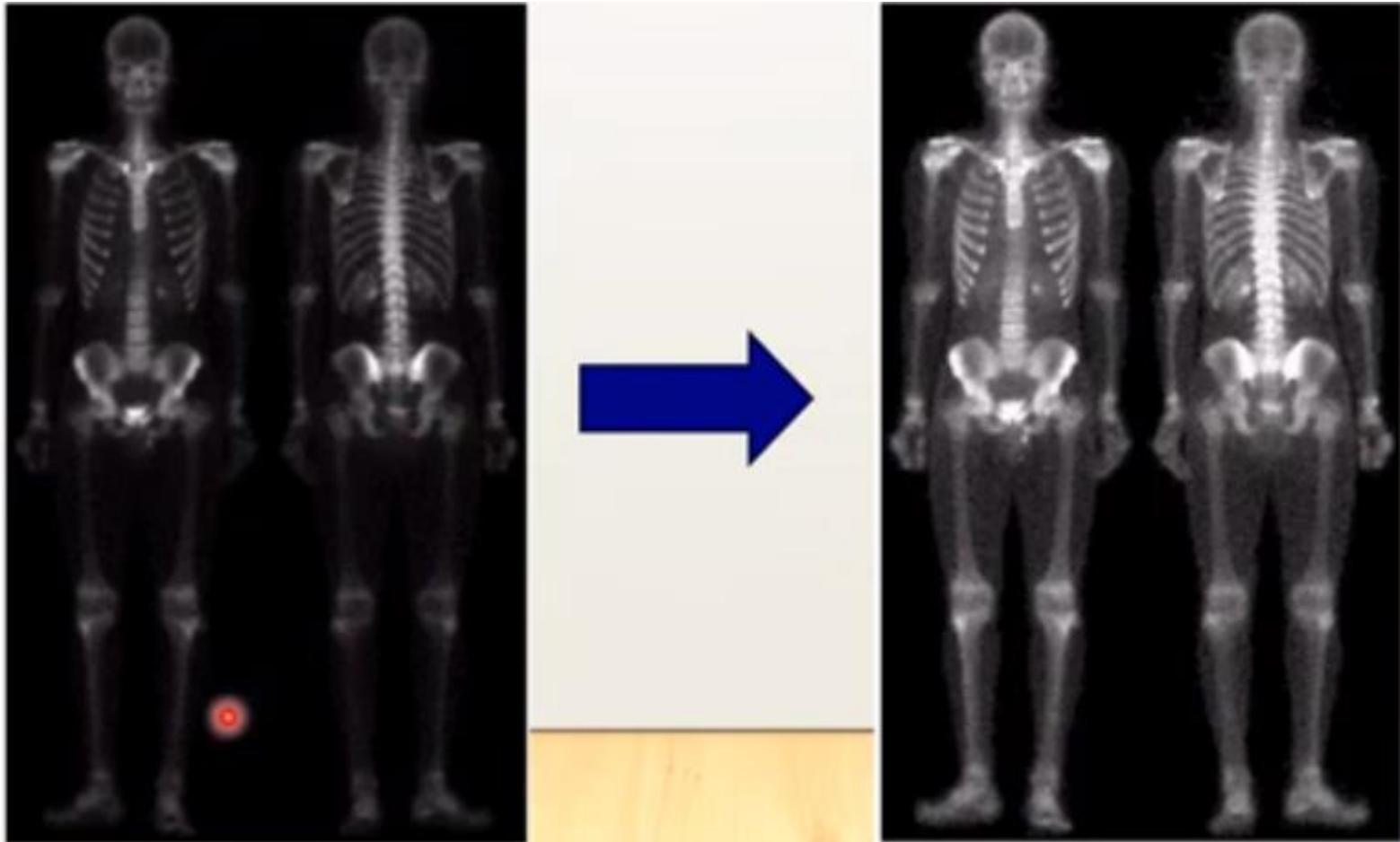


Image Enhancement Examples(Cont...)

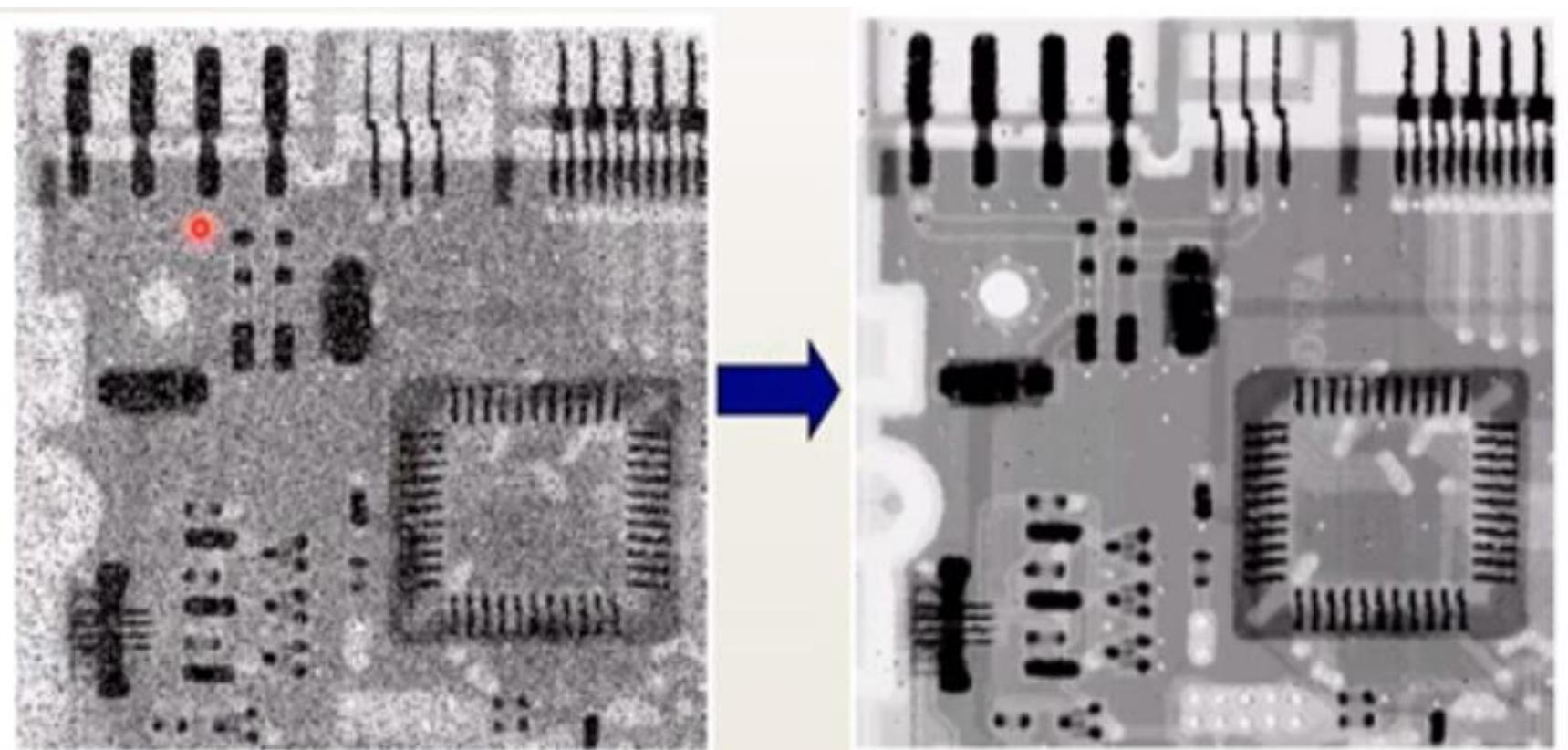


Image Enhancement Examples(Cont...)



Spatial & Frequency Domains

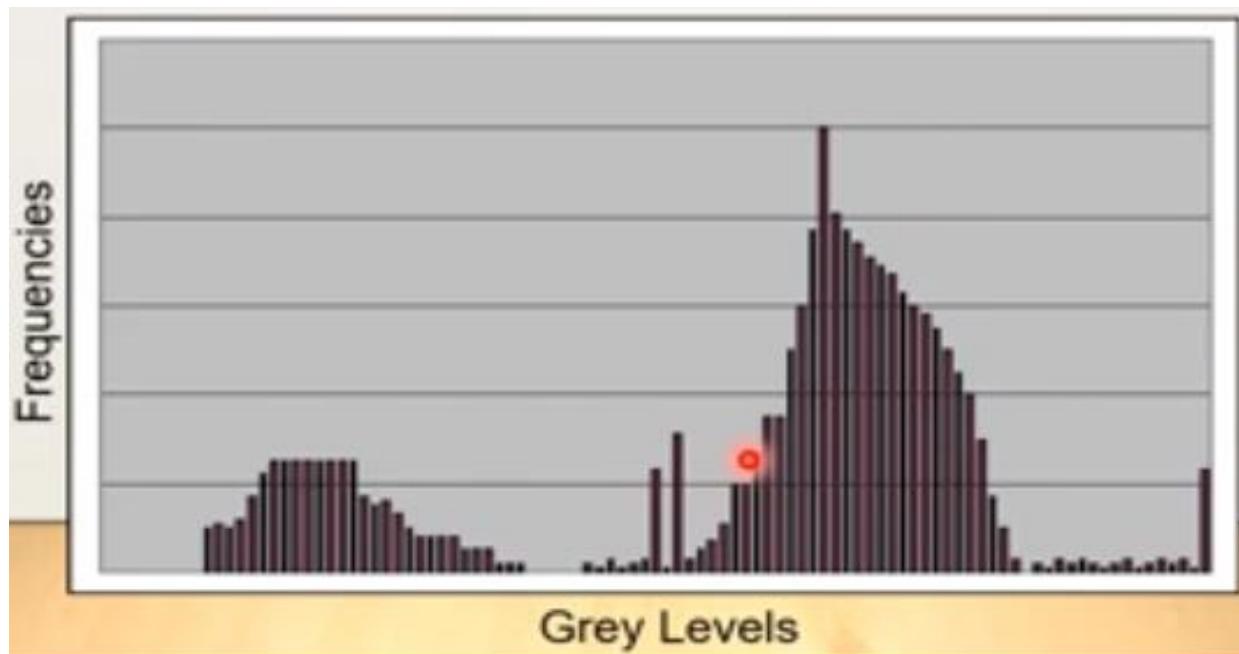
- There are two broad categories of image enhancement techniques
- Spatial domain techniques
 - Direct manipulation of image pixels
- Frequency domain techniques
 - Manipulation of Fourier transform or wavelet transform of image an image

Image Enhancement

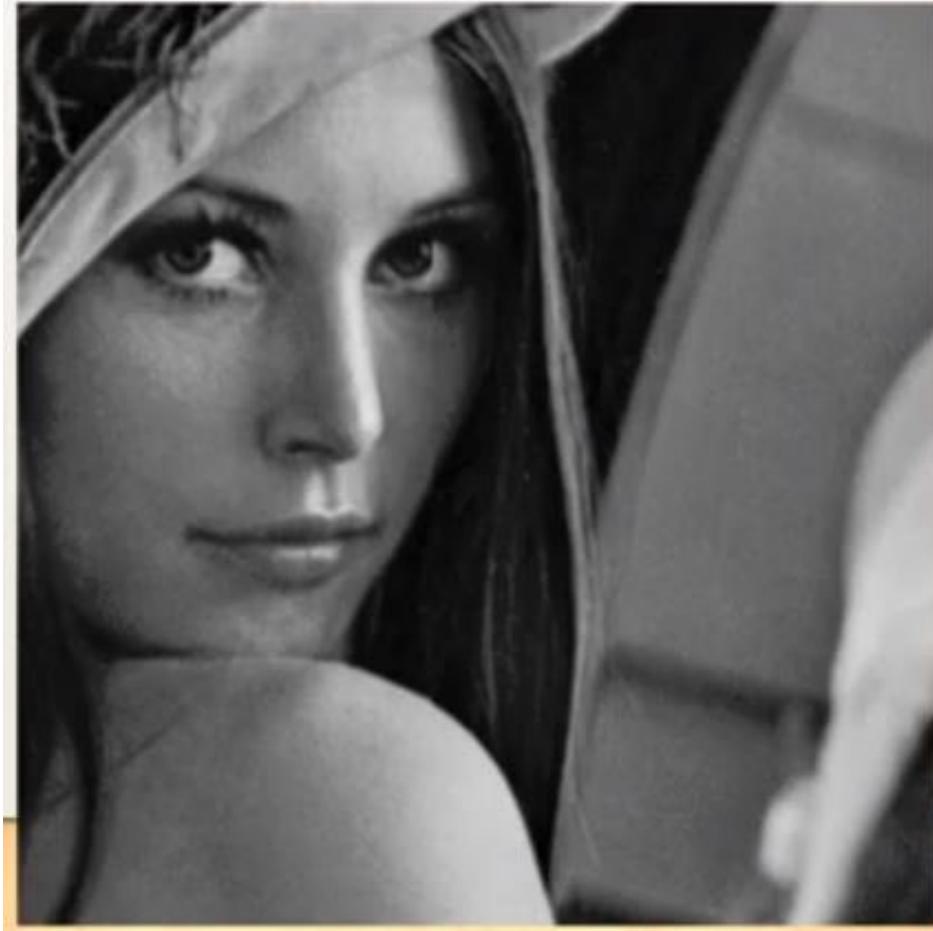
Image Histograms

Image Histograms

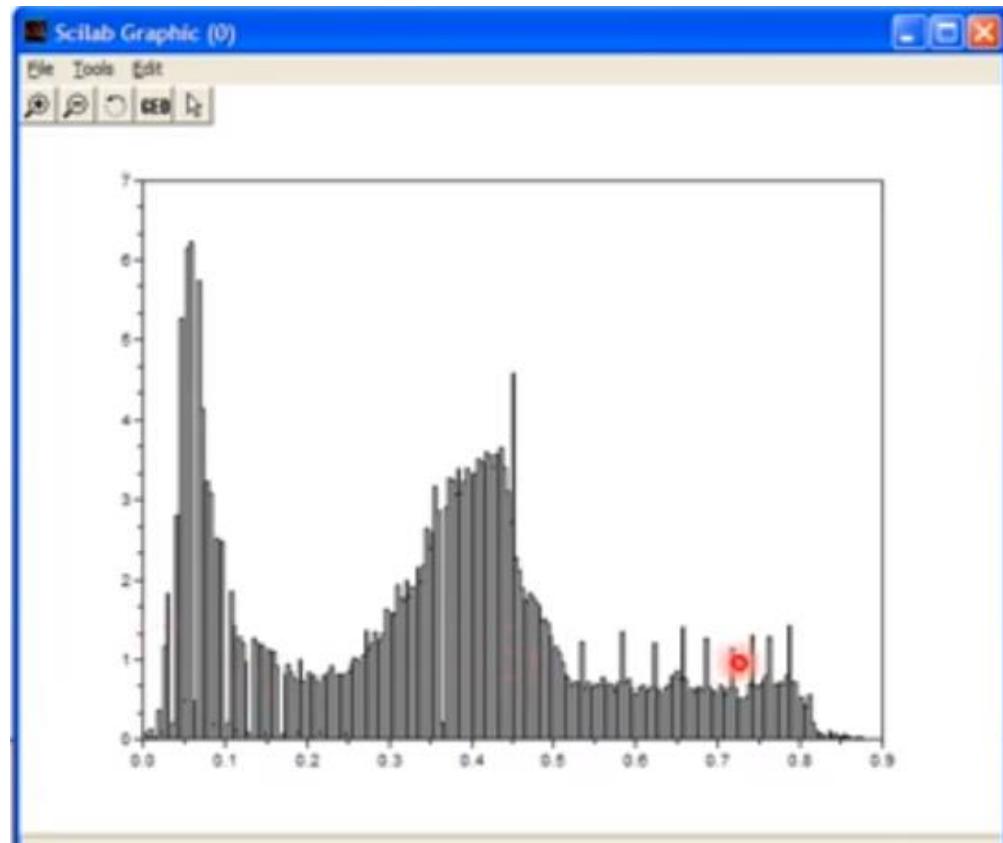
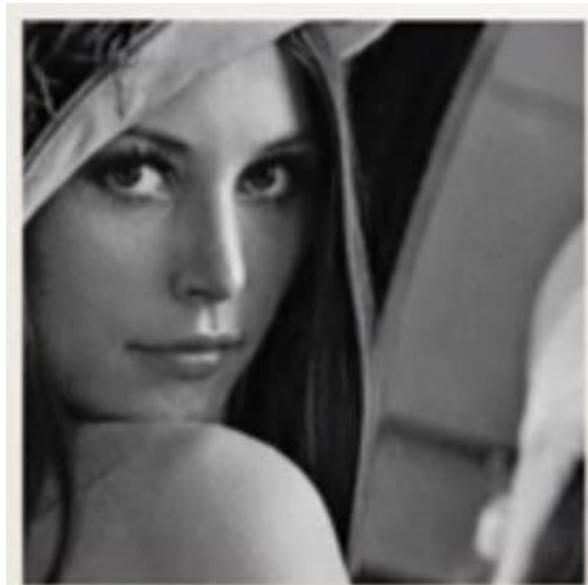
- The histogram of an image shows us the distribution of grey levels in image
- Massively useful in image processing especially in segmentation.



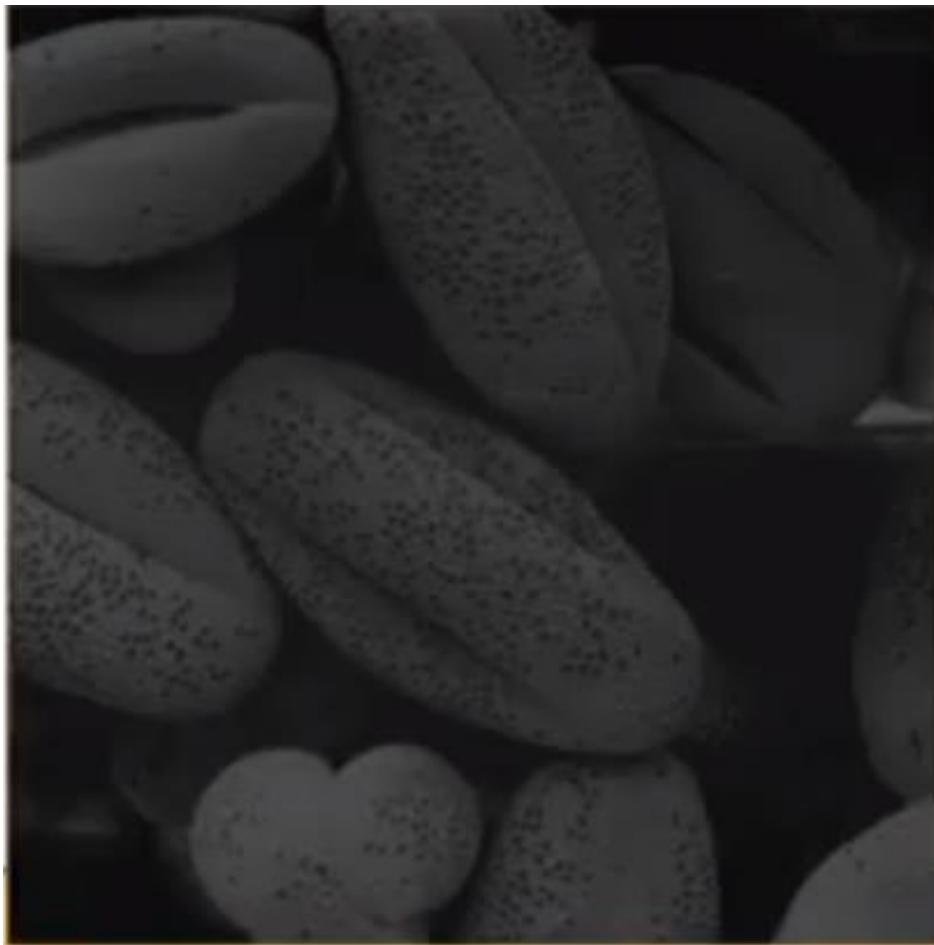
Histogram Examples



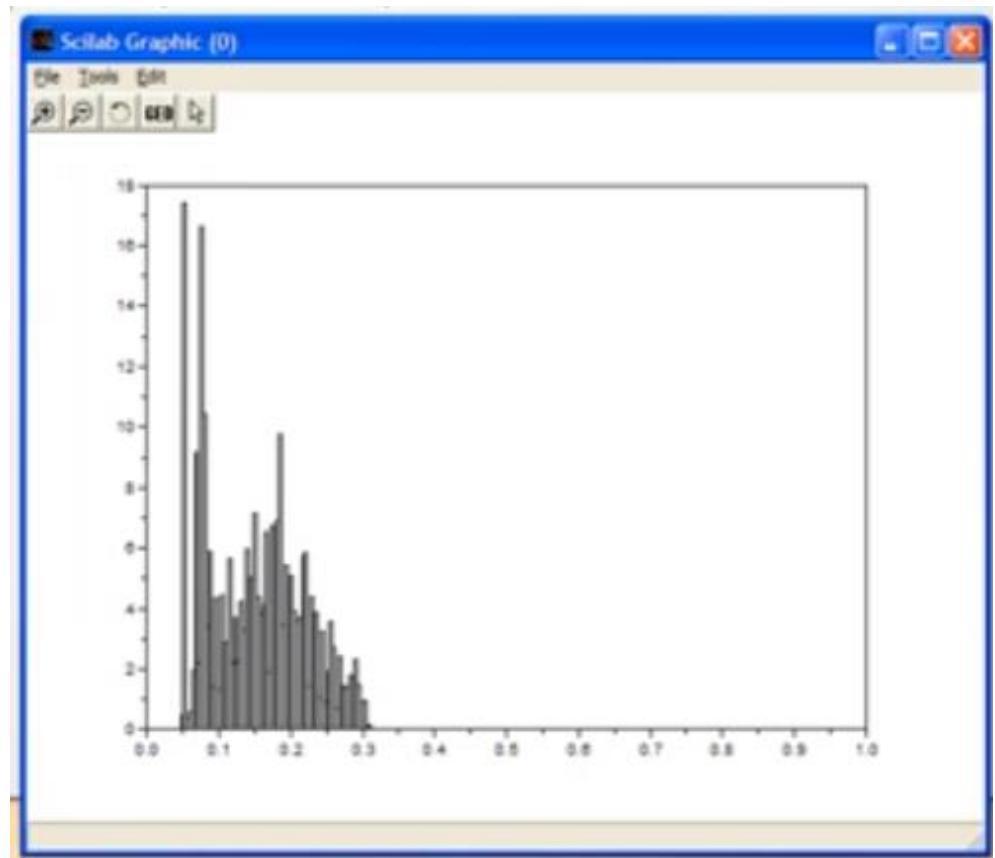
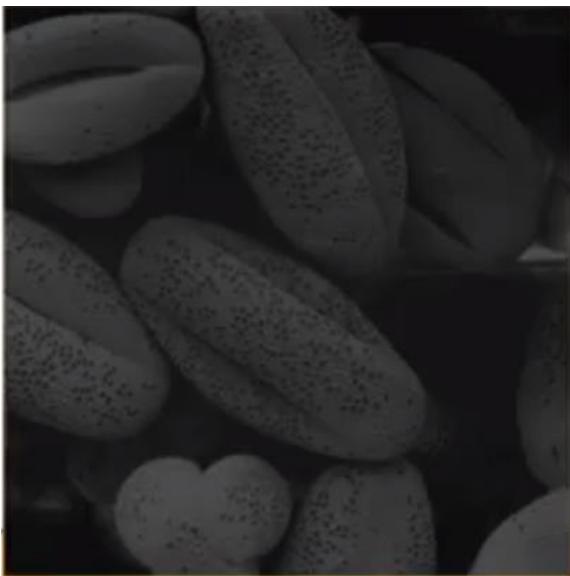
Histogram Examples(Cont...)



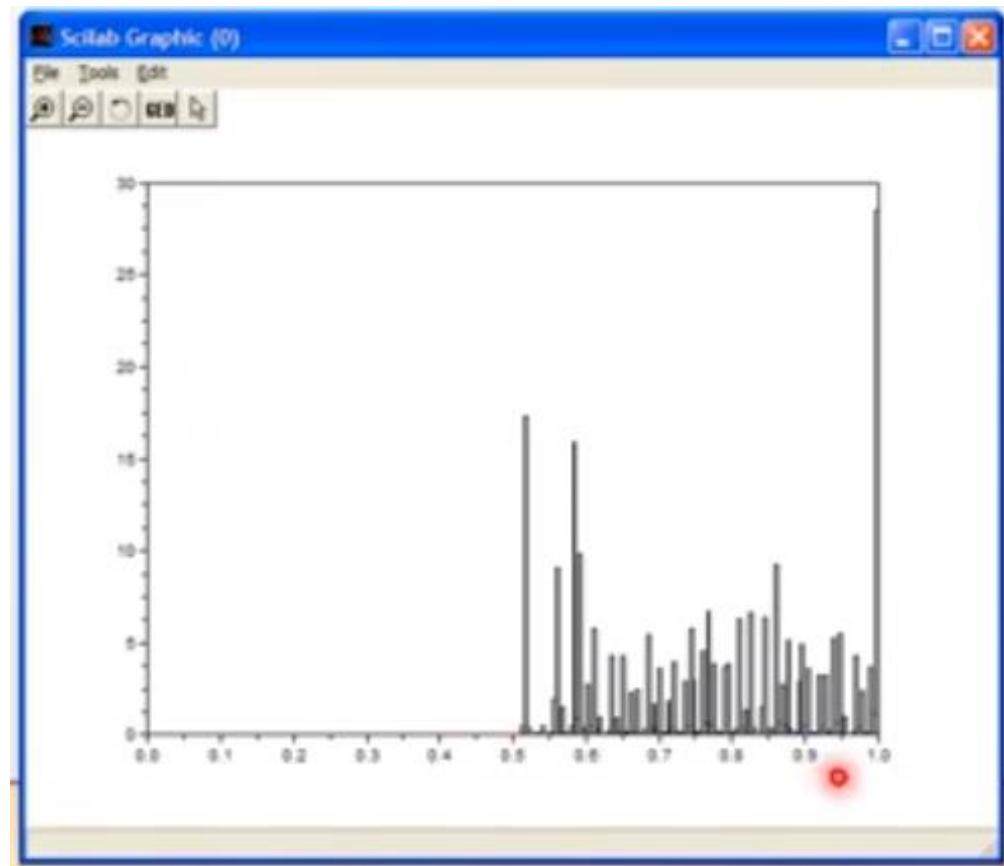
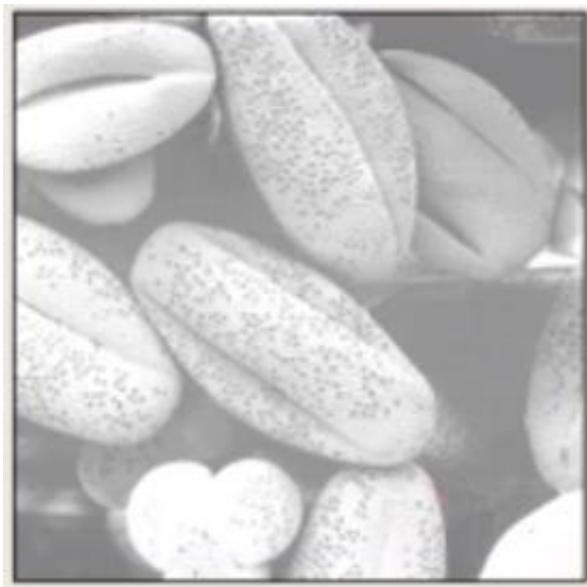
Histogram Examples(Cont...)



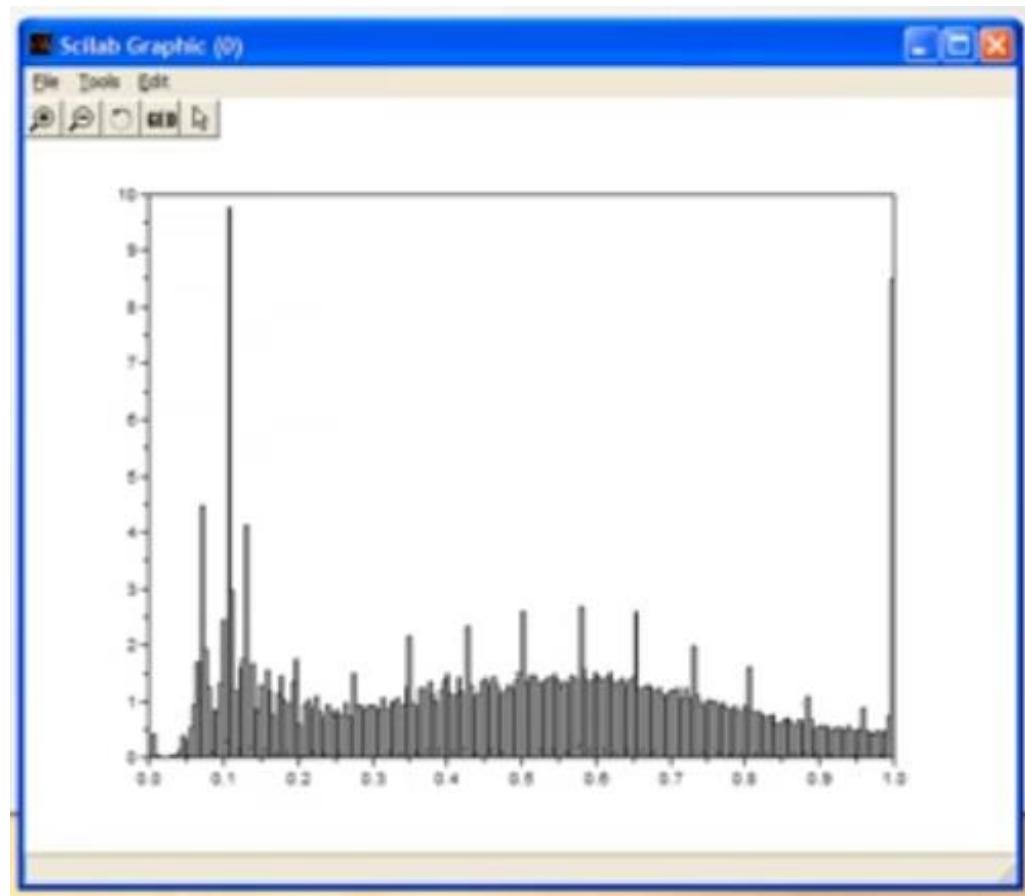
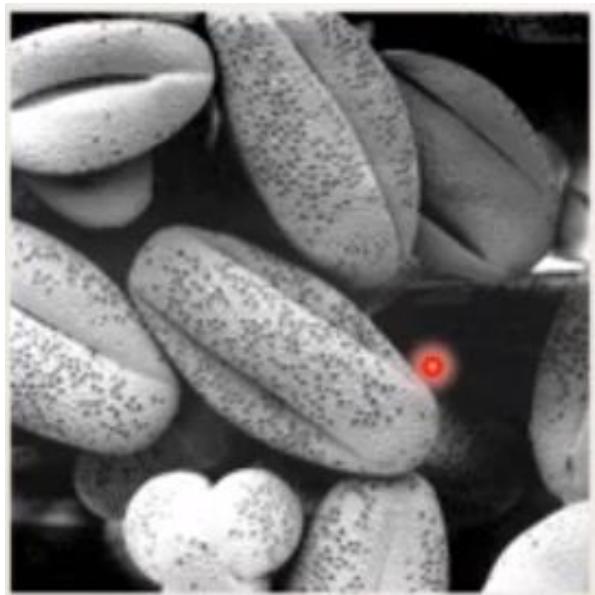
Histogram Examples(Cont...)



Histogram Examples(Cont...)

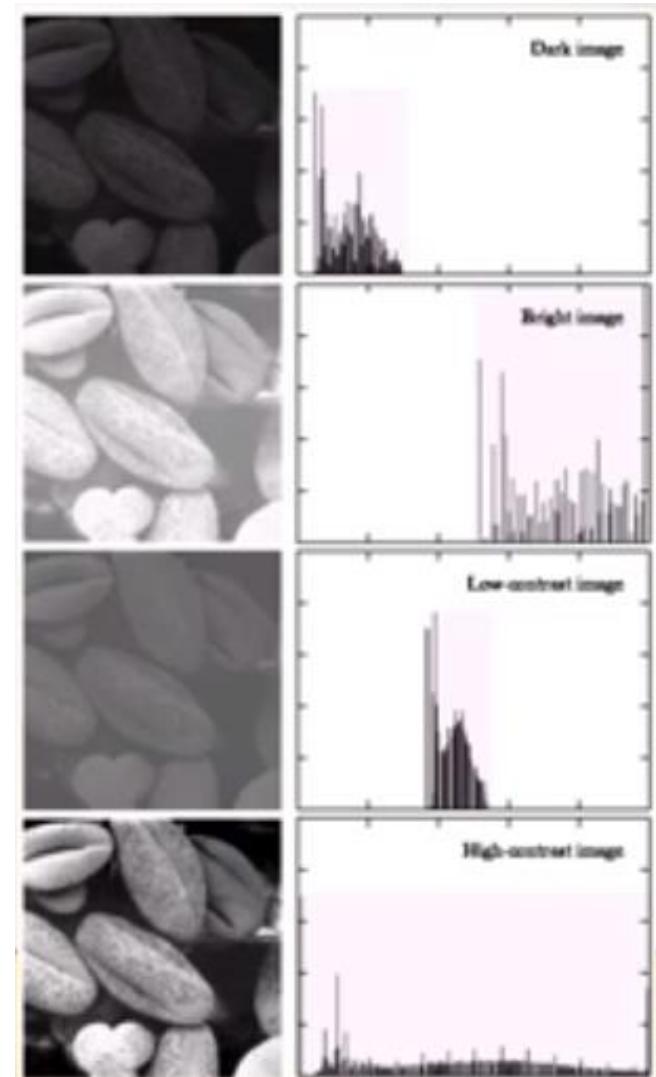


Histogram Examples(Cont...)

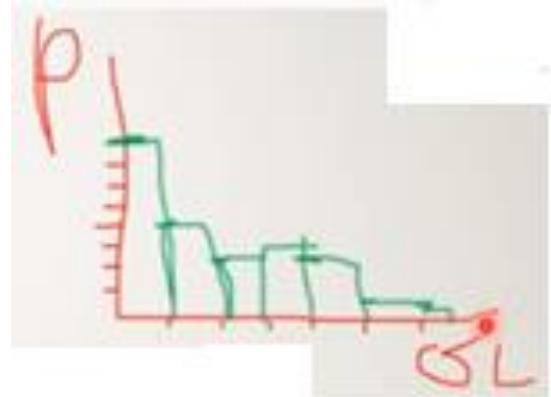


Histogram Examples(Cont...)

- A selection of images and their histogram
- Notice the relationships between the images and their histograms
- Note that the high contrast image has the most evenly spaced histogram



Histogram Examples



- Plot the histogram for following data.

Grey Level	Number of Pixels
0	40
1	20
2	10
3	15
4	10
5	3
6	2

Histogram Examples

- Calculate the probability of occurrence of the grey level(normalized histogram)

Grey Level	Number of Pixels(nk)	$P(r_k) = nk/n$
0	40	0.4
1	20	0.2
2	10	0.1
3	15	0.15
4	10	0.1
5	3	0.03
6	2	0.02
	n=100	

Image Enhancement Histograms Stretching

Histogram Examples

- Calculate the probability of occurrence of the grey level(Probability Density Functions).

Grey Level	Number of Pixels(nk)	$P(r_k) = n_k/n$
0	40	
1	20	
2	10	
3	15	
4	10	
5	3	
6	2	
	n=100	

Linear Stretching

- One way to increase the dynamic range is by using a technique known as histogram stretching.
- In this method, we do not alter the basic shape of the histogram, but we spread it so as to cover the entire dynamic range.

$$s = T(r) = \frac{S_{\max} - S_{\min}}{R_{\max} - R_{\min}} (r - R_{\min}) + S_{\min}$$

- S_{\max} -> Maximum grey level of output image
- S_{\min} -> Minimum grey level of output image
- r_{\max} -> Maximum grey level of input image
- r_{\min} -> Minimum grey level of input image

Histogram Stretching : Examples

- Perform histogram stretching so that new image has a dynamic range of [0,7].

Grey Level	Number of Pixels
0	0
1	0
2	50
3	60
4	50
5	20
6	10
7	0

Histogram Stretching : Examples

$$R_{\min} = 2$$

$$S_{\min} = 0 \quad r_i = [2, 6]$$

$$R_{\max} = 6$$

$$S_{\max} = 87$$

$$S = \frac{S_{\max} - S_{\min}}{r_{\max} - r_{\min}} (r - r_{\min}) + S_{\min}$$

Histogram Stretching : Examples

Put the value of ,

$$S = \frac{7-0}{6-2} (n-2) + 0$$

$$S = \frac{7}{4} (n-2) \Rightarrow n=2 : S = \frac{7}{4} (2-2) = 0$$

$$\Rightarrow n=3 : S = \frac{7}{4} (3-2) = \frac{7}{4} = 1.75 \approx 2$$

$$\Rightarrow n=4 : S = \frac{7}{4} (4-2) = \frac{7}{4} \times 2 = 3.5 \approx 4$$

$$\Rightarrow n=5 : S = \frac{7}{4} (5-2) = \frac{7}{4} \times 3 = 5.25 \approx 5$$

$$\Rightarrow n=6 : S = \frac{7}{4} (6-2) = \frac{7}{4} \times 4 = 7$$

Here we find the all slope value
at different range .

Histogram Stretching : Examples

Cr. L	Old pixel	New pixel
0	0	50
1	0	0
2	50	60
3	60	0
4	50	50
5	20	20
6	10	0
7	0	10

$$S = \{0, 2, 4, 5, 7\}$$

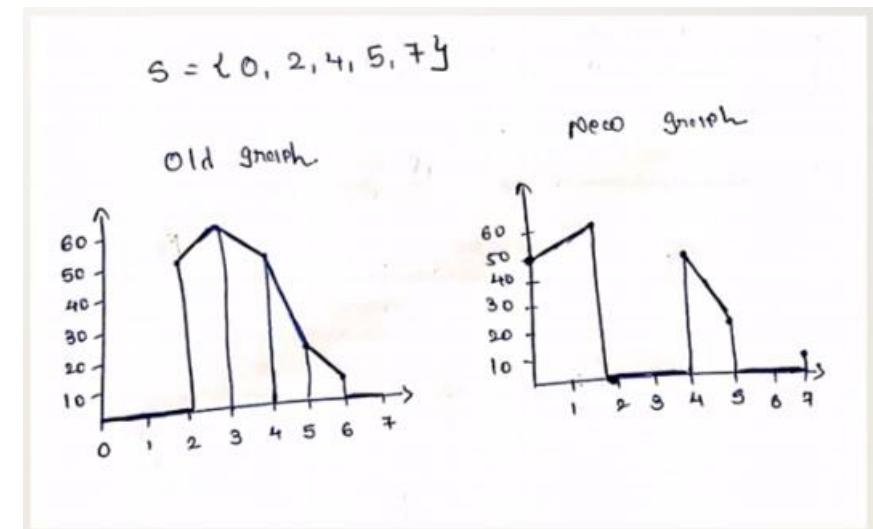


Image Enhancement

Histogram Equalization

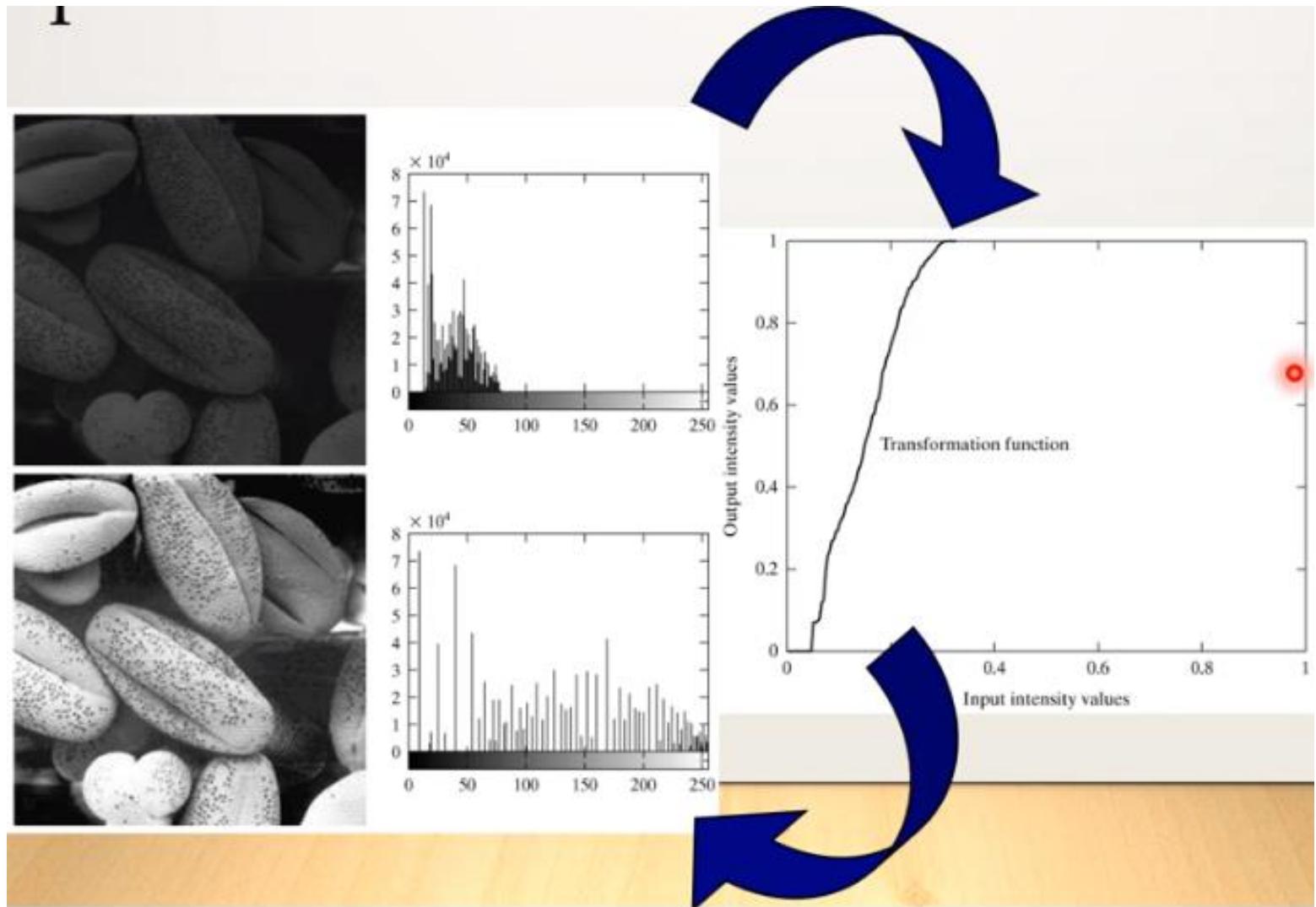
Histogram Equalisation

- Spreading out the frequencies in an image(or equalizing the image) is a simple way to improve dark or washed out images
- The formula for histogram Equalisation is given where

- r_k : input intensity
- s_k : processed intensity
- k : the intensity range
(e.g 0.0 – 1.0)
- n_j : the frequency of intensity j
- n : the sum of all frequencies

$$\begin{aligned}s_k &= T(r_k) \\ &= \sum_{j=1}^k p_r(r_j) \\ &= \sum_{j=1}^k \frac{n_j}{n}\end{aligned}$$

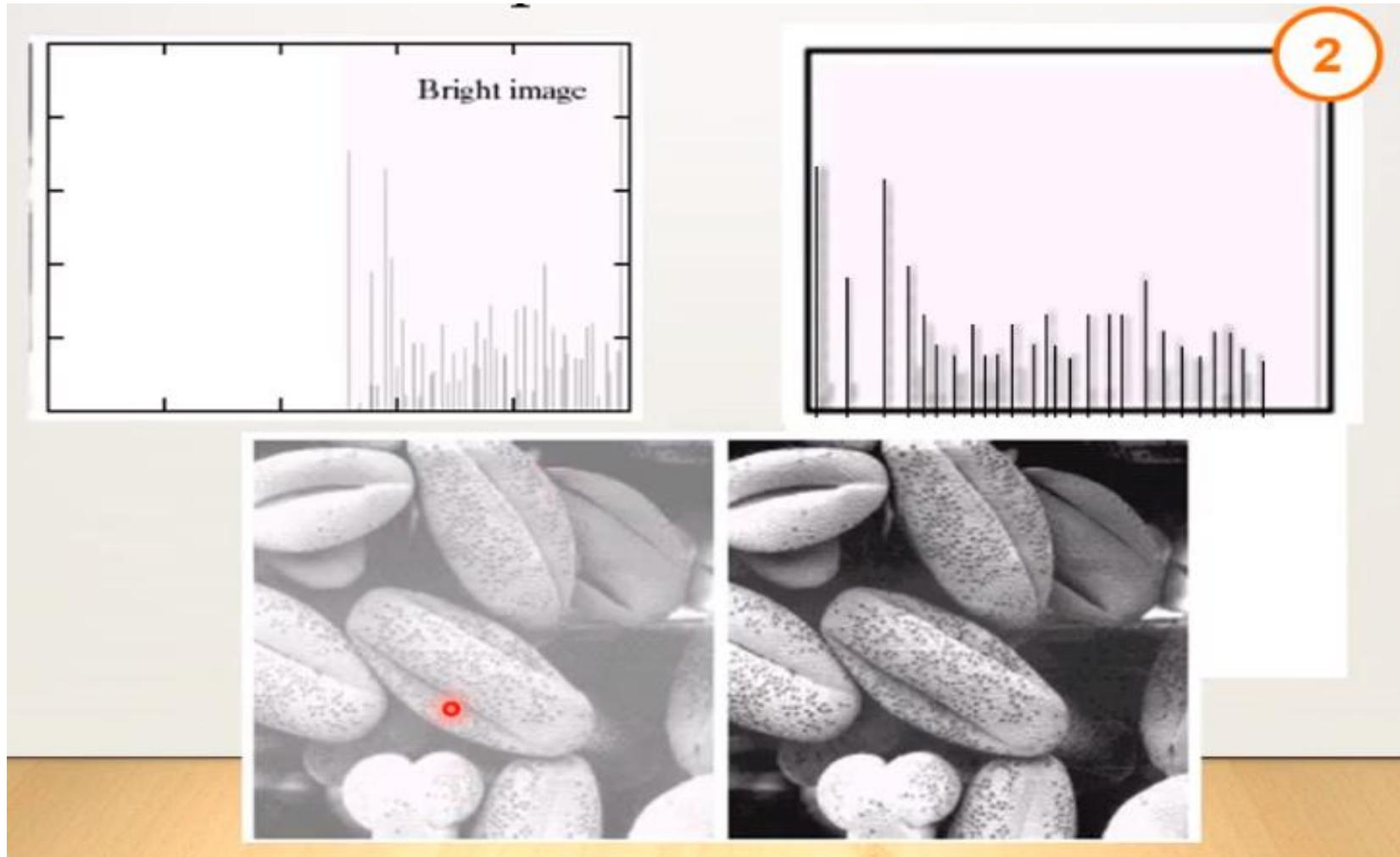
Equalisation Transformation Function



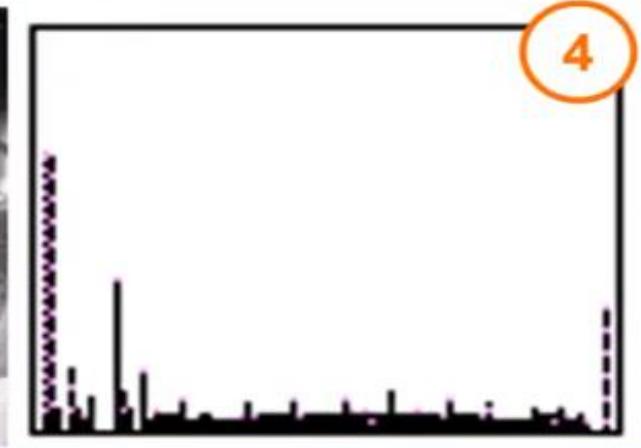
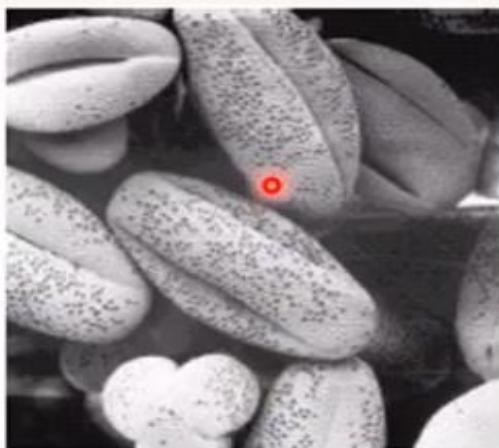
Equalisation Examples



Equalisation Examples



Equalisation Examples



Histogram Equalisation

- Histogram--->PDF ---> CDF ---> Equalized histogram
- CDF – cumulative density function

$$S_k = T(r_k) = \sum_0^r p_r(r)$$

Equalize the given histogram

Grey Level	Number of Pixels(nk)	PDF P(rk) = nk/n	CDF	(L-1)*Sk	Round off
0	790				
1	1023				
2	850				
3	656				
4	329				
5	245				
6	122				
7	81				
4096					

Equalize the given histogram

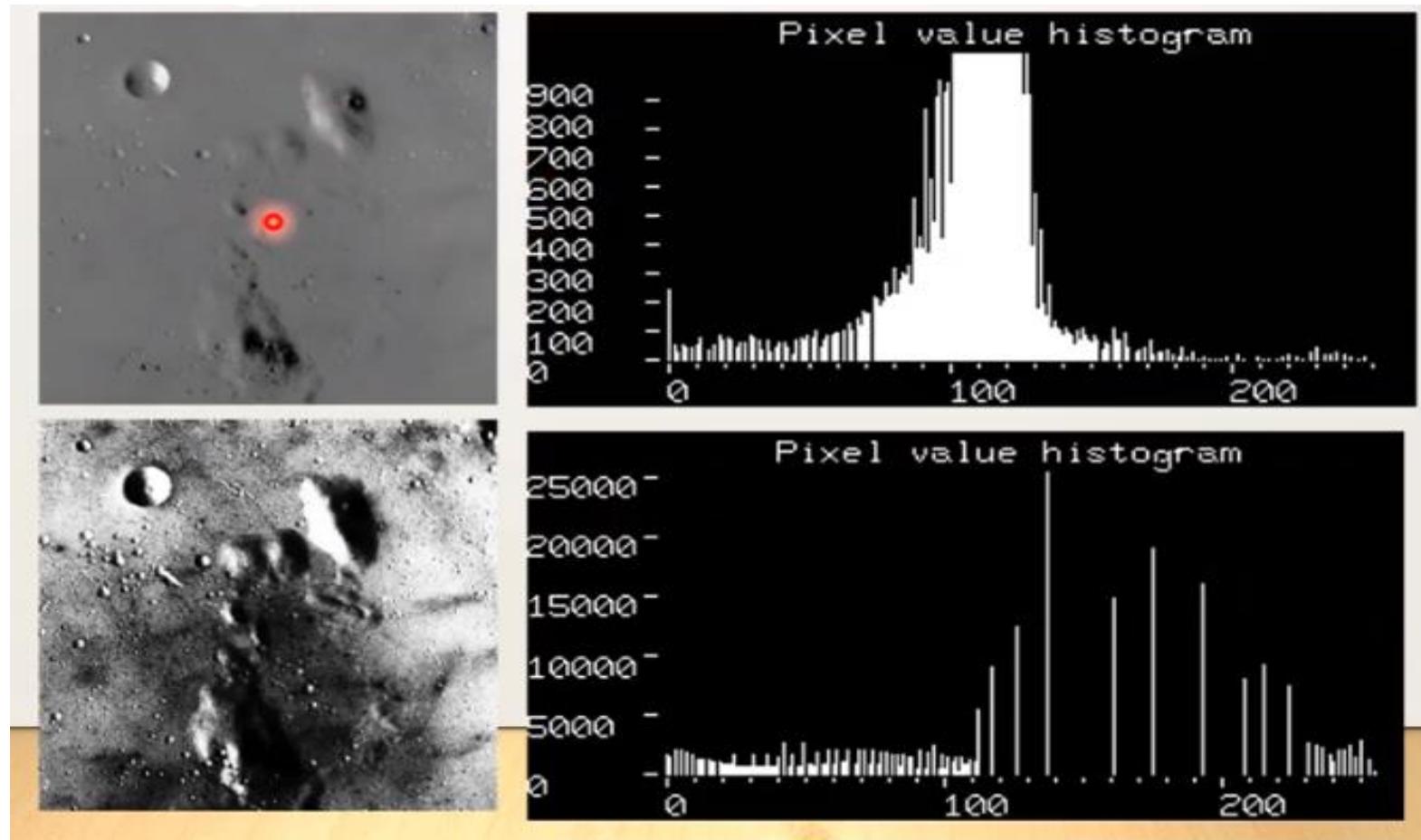
Grey Level	Number of Pixels(nk)	PDF $P(rk) = nk/n$	CDF $Sk = \sum_0^r p_r(r)$	$(L-1)*Sk$	Round off
0	790	0.19	0.19	1.933	1
1	1023	0.25	0.44	3.08	3
2	850	0.21	0.65	4.55	5
3	656	0.16	0.81	5.67	6
4	329	0.08	0.89	6.23	6
5	245	0.06	0.95	6.65	7
6	122	0.03	0.98	6.86	7
7	81	0.02	1	7	7
4096					

Equalized Grey Level

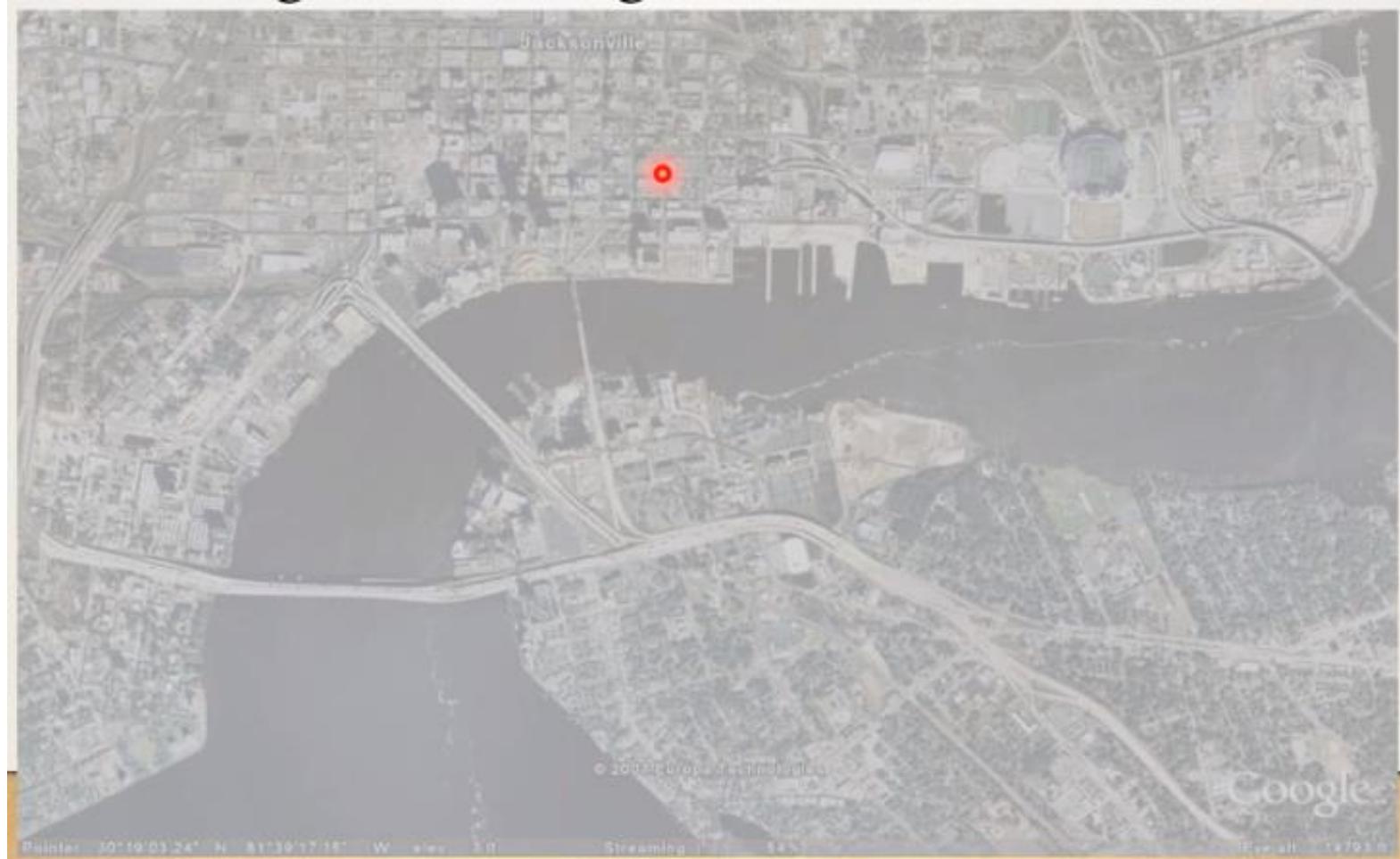
Grey Level	Old Number of Pixels	New Number of Pixels
0	790	0
1	1023	790
2	850	0
3	656	1023
4	329	0
5	245	850
6	122	$656+329 = 985$
7	81	$245+122+81 = 448$

Grey Level	Round off
0	1
1	3
2	5
3	6
4	6
5	7
6	7
7	7

Histogram Equalization



Satellite Original Image



Histogram Equalized Image



Noise Removal and Blurring

What is Image Restoration?

- Image restoration attempts to restore images that have been degraded
 - Identify the degradation process and attempts to reverse it.
 - Similar to image enhancement, but more objective.

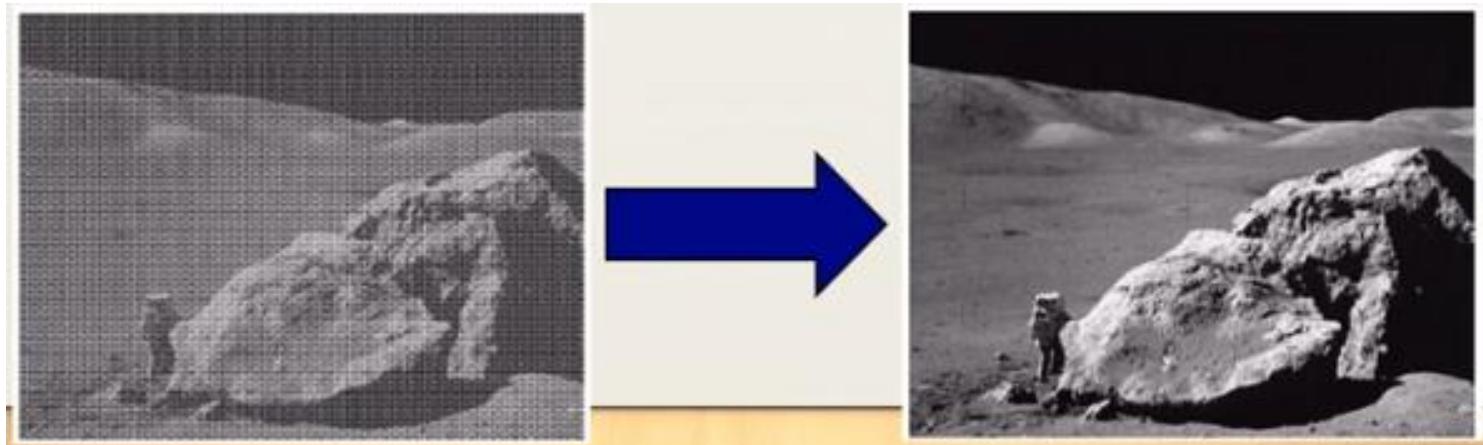


Image restoration vs. Image enhancement

- **Enhancement:**
 - Largely a subjective process
 - Priori knowledge about the degradation is not a must(sometime no degradation is involved)
 - Procedures are heuristic and take advantage of the psychophysical aspects of human visual system
- **Restoration:**
 - More an objective process
 - Images are degraded
 - Tries to recover the image by using the knowledge about the degradation

Noise and Images

- The sources of noise in digital images arise during image acquisition(digitization) and transmission
 - Imaging sensors can be affected by ambient conditions
 - Interference can be added to an image during transmission



Noise Model

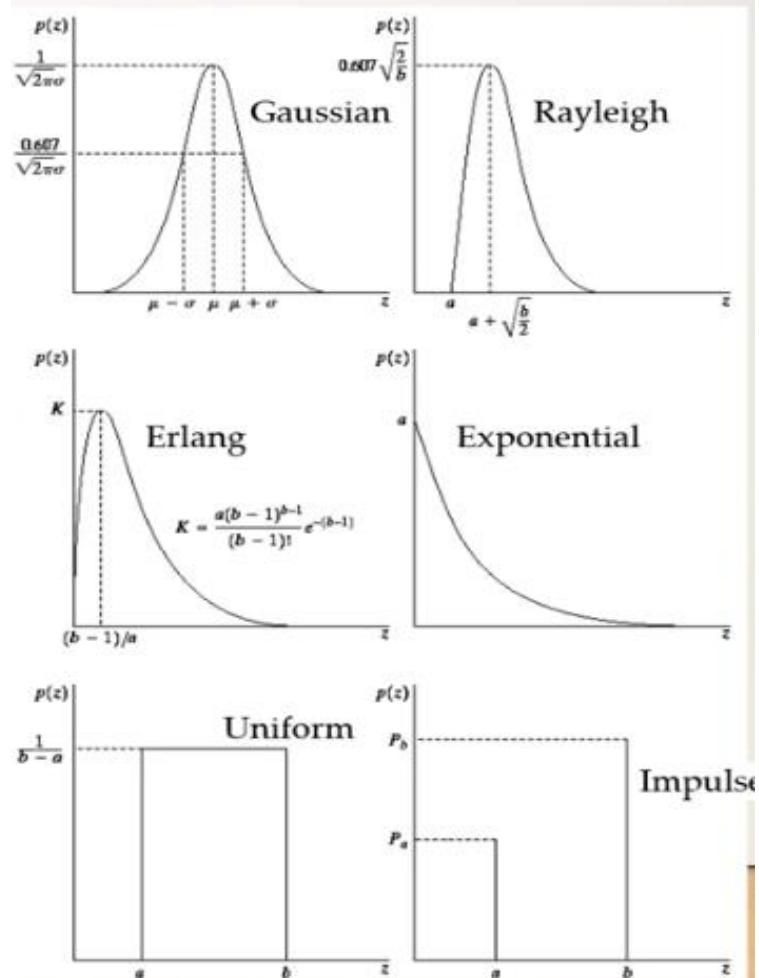
- We can consider a noisy image to be modelled as follows:

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

- where $f(x, y)$ is the original image pixel, $\eta(x, y)$ is the noise term and $g(x, y)$ is the resulting noisy pixel
- If we can estimate the model the noise in an image is based on this will help us to figure out how to restore the image

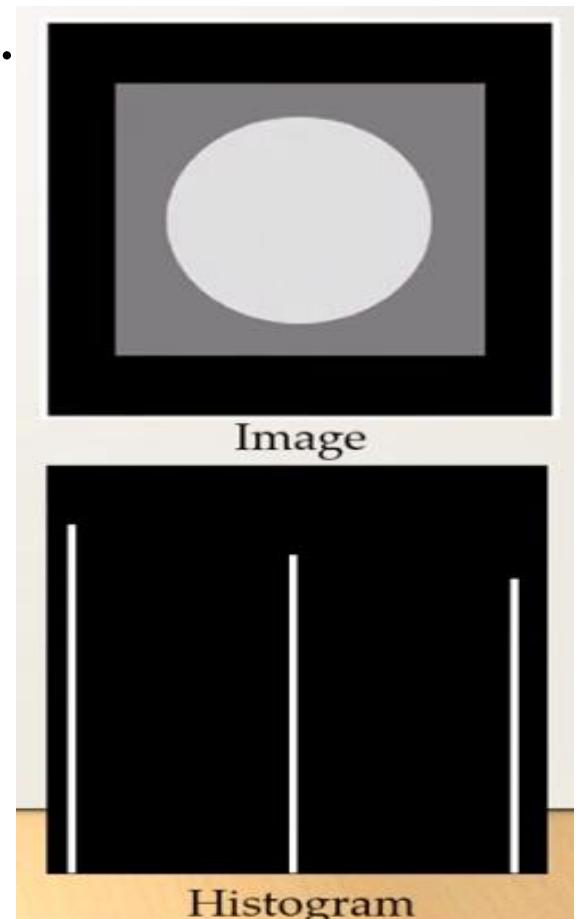
Noise Models

- There are many different models for the image noise term $\eta(x,y)$:
- Gaussian
 - Most common mode
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
 - Salt and pepper noise

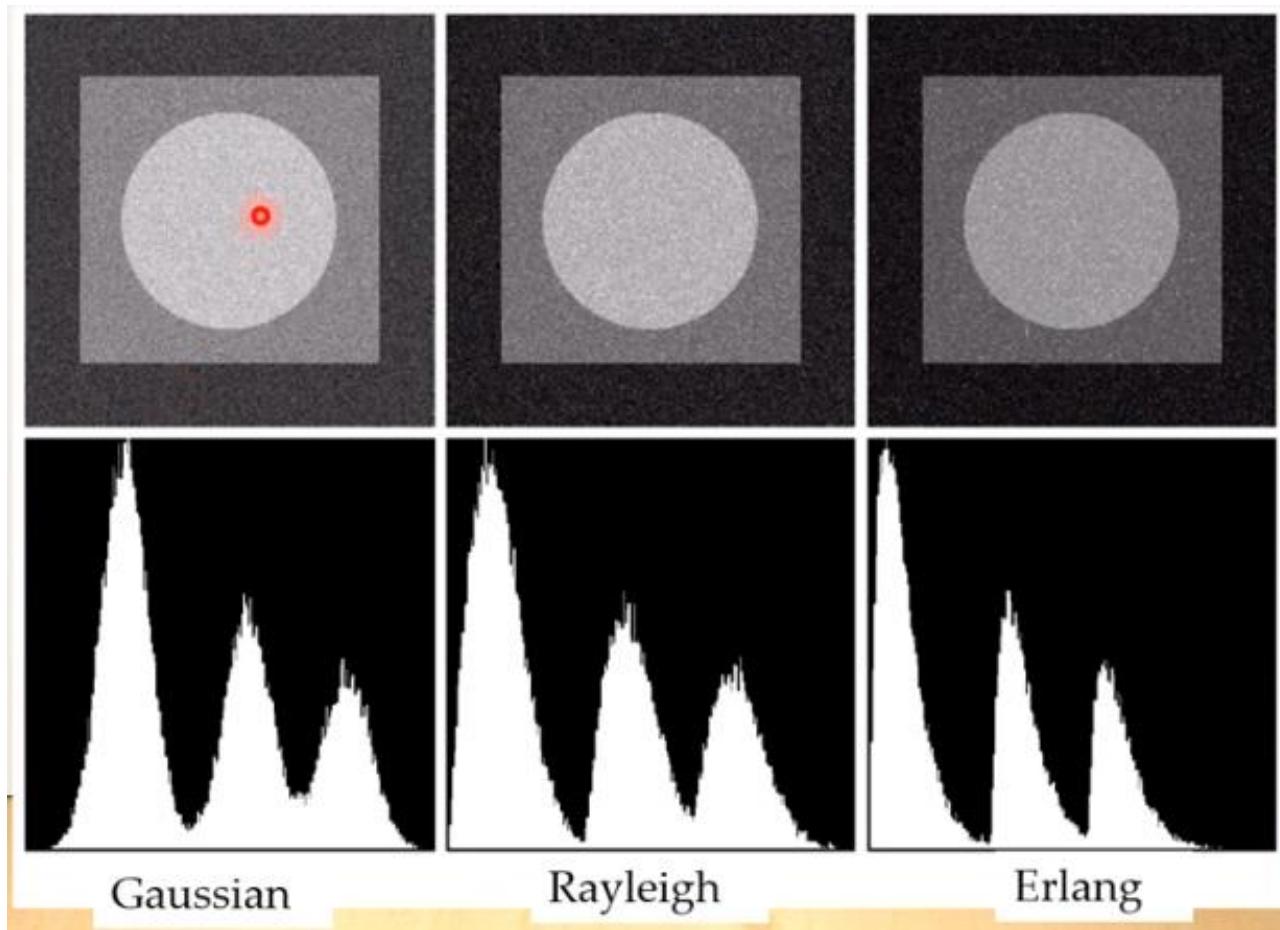


Noise Example

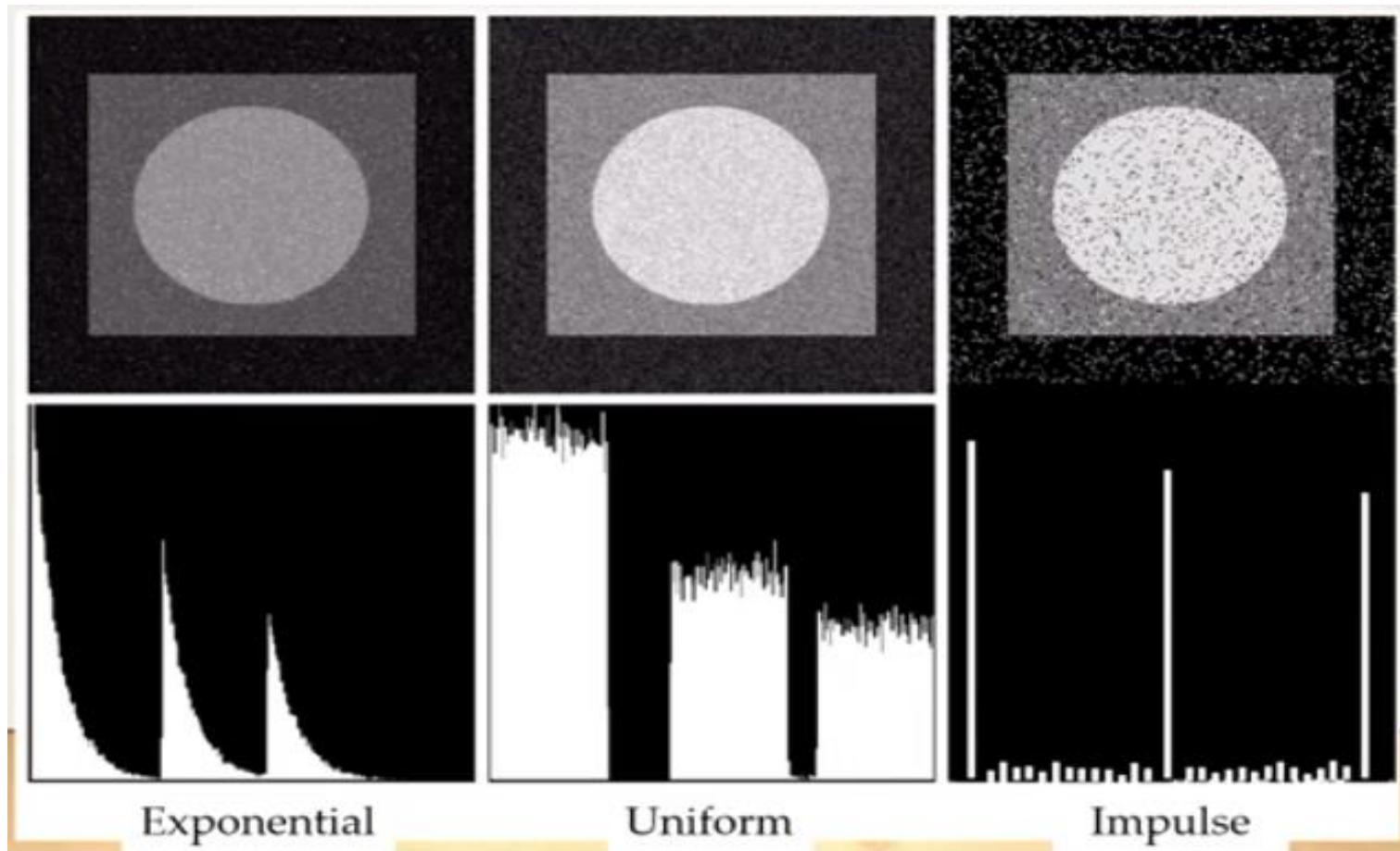
- The test pattern to the right is ideal for demonstrating the addition of noise.
- The following slides will show the result of adding noise based on various models to this image.



Noise Example(Cont...)



Noise Example(Cont...)



Noise Removal

Arithmetic Mean Filter

Filtering to Remove Noise : Arithmetic Mean Filter

- We can use spatial filters of different kinds to remove different kinds of noise
- The *arithmetic mean* filter is a very simple one and is calculated as follows:

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

This is implemented as the
simple smoothing filter

Blurs the image to remove
noise

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s, t)$$

Image Capture Device

:- Image capture device

→ noise model

$$g(x,y) = f(x,y) + n(x,y)$$

resulting original noise

noisy image.

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

Filtering to remove Noise

→ Filtering to remove noise.

- We can use spatial filters of different kinds to remove different kinds of noise.

1. Arithmetic mean filter:-

$$\hat{f}(x, y) = \frac{1}{n} \sum g(x+i, y+j)$$

At x axis -

$$\hat{f}(x) = \frac{1}{m} \sum g(x+i, y)$$

At y axis -

$$\hat{f}(y) = \frac{1}{n} \sum g(x, y+i)$$

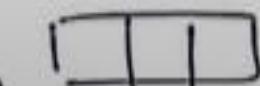
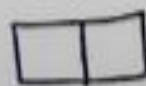
Example: Arithmetic Mean Filter

3, 9, 4, 52, 3, 8, 6, 2, 2, 9.

Step :- 1

1st false windowing size,

:- we can take any size for windowing.



, - - -

Step:2

Step:-2

1. 0 Padding :

Normal: 3, 9, 4, 52, 3, 46, 6, 2, 2, 9

0
padding:

0	3	9
---	---	---

3	9	4
---	---	---

9	4	52
---	---	----

 ...

2	9	0
---	---	---

Replication:

3	3	9
---	---	---

3	9	4
---	---	---

 ...

2	9	9
---	---	---

Trimming:

3	9	4
---	---	---

9	4	52
---	---	----

 ...

2	2	9
---	---	---

Example: Arithmetic Mean Filter

Step 1: 0 Padding

Solve example using arithmetic mean filter.

(ii) 0 Padding:

[0 3 9]

$$= 0 + 3 + 9 = 12$$

[3 9 4]

$$= 16 = 5.33 \approx 5$$

[9 4 5 2]

$$= 21.66 \approx 22$$

[4 5 2 3]

$$= 19.66 \approx 20$$

[5 2 3 8]

$$= 21$$

[1 3 8 6]

$$= 5.66 \approx 6$$

[8 6 2]

$$= 5.33 \approx 5$$

[1 6 2 2]

$$= 3.33 \approx 3$$

[2 2 9]

$$= 4.33 \approx 4$$

[2 9 0]

$$= 3.66 \approx 4$$

[4 5 22 20 21 16 5 3 14 4]

Step 2: Replication

(ii) Replication:

$$\boxed{3 \ 3 \ 9} = \frac{3+3+9}{3} = 5$$

other one same w/ only change boundary

$$\boxed{2 \ 9 \ 9} = 6.66 \approx 7$$

$$\boxed{5 \ 5 \ 22 \ 20 \ 21 \ 16 \ 5 \ 3 \ 4 \ 7}$$

Step 3: Trimming

(iii) trimming:

Does not change boundary pixel (11)

Other pixels are changed,

3	5	22	20	2	6	5	3	4	9
---	---	----	----	---	---	---	---	---	---

Other Means

- There are different kinds mean filters all of which exhibit slightly different behavior:
 - Geometric Mean
 - Harmonic Mean
 - Contrahamonic Mean

Geometric Mean:

- Geometric Mean:

$$\hat{f}(x, y) = \left[\prod_{(s,t) \in S_{xy}} g(s, t) \right]^{\frac{1}{mn}}$$

- Achieves similar smoothing to the arithmetic mean, but tends to lose less image detail

②

Geometric mean :-

$$\hat{f}(xy) = \left(\prod g(s_i) \right)^{1/m}$$

C1+ X class

$$\hat{f}(x) = \left(\prod g(s_i) \right)^{1/m} = \left(\prod x_i \right)^{1/m}$$

C1+ Y class

$$\hat{f}(y) = \left(\prod g(f_i) \right)^{1/n} = \left(\prod y_i \right)^{1/n}$$

Geometric Mean: Example

(i) Replication:-

$$[3, 9, 4, 52, 3, 81, 6, 2, 2, 1, 9]$$

$$\boxed{3 \mid 3 \mid 9} = ((3 \times 3) \times 9)^{1/3} = (81)^{1/3} = 4.32$$

$$\boxed{3 \mid 9 \mid 4} = 4.76 \approx 5$$

$$\boxed{9 \mid 4 \mid 52} = 12.32 \approx 12$$

$$\boxed{4 \mid 52 \mid 3} = 8.54 \approx 9$$

Cont...

$$\boxed{4|52|3} = 8.54 \approx 9$$

$$\boxed{52|3|8} = 10.76 \approx 11$$

$$\boxed{1|3|8|6} = 5.24 \approx 5$$

$$\boxed{8|6|2} = 4.57 \approx 5$$

$$\boxed{1|6|2|2} = 2.88 \approx 3$$

$$\boxed{2|2|9} = 3.30 \approx 3$$

$$\boxed{1|2|9|9} = 5.45 \approx 5$$

$$\boxed{4|5|12|9|11|5|5|3|3|5}$$

Cont...

(iii) Padding:

0|3|9

2|9|0

0|5|12|9|11|5|5|3|3|0

(iv) Trimming:

3|9|4

2|2|9

3|5|12|9|11|5|5|3|3|9|

Harmonic Mean

Harmonic Mean:

$$\hat{f}(x, y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{g(s, t)}}$$

- Works well for salt noise, but fails for pepper noise
- Also does well for other kinds of noise such as Gaussian noise

Cont...

③

Harmonic Mean :-

works for salt

noise

$$\hat{f}(x_i, y_i) = \frac{mn}{\sum \frac{1}{g(s_i)} + \sum \frac{1}{g(t_i)}} = mn \text{ but } f(x_i, y_i) \text{ is pepper noise}$$

at x axis,

$\hat{f}_x(x) = \frac{m}{\sum \frac{1}{g(s_i)}} = m$ crosses

salt

$$\hat{f}_x(x) = \frac{m}{\sum \frac{1}{g(s_i)}} = m$$

at y axis

$$\hat{f}_y(y) = \frac{n}{\sum \frac{1}{g(t_i)}} = n$$

Example: Harmonic Mean Filter

3, 9, 4, 52, 3, 8, 6, 2, 2, 9

ci) Replicating:

$\boxed{3 \ 3 \ 9} = \frac{3}{(1/15)} = \frac{3}{\frac{1}{3} + \frac{1}{3} + \frac{1}{9}} = 3(9) = 3.85$

$\boxed{3 \ 9 \ 4} = \frac{3}{(1/16)} = \frac{3}{\frac{1}{3} + \frac{1}{9} + \frac{1}{4}} = 3(4) = 4.32$

$\boxed{9 \ 4 \ 52} = \frac{3}{(1/65)} = \frac{3}{\frac{1}{9} + \frac{1}{4} + \frac{1}{52}} = 3(52) = 195$

as well as below,

Contrahamonic Mean

- Contraharmonic Mean:
$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} g(s, t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s, t)^Q}$$

- Q is the *order* of the filter and adjusting its value changes the filter's behaviour
- Positive values of Q eliminate pepper noise
- Negative values of Q eliminate salt noise

Contrahamonic Mean Equation

④

Contrahamonic mean :-

$$\mu = \frac{p(s, e)}{q+1} - q \cdot \frac{r(s, e)}{\sum g(s, t)}$$

$$p = \frac{\sum p(s, t)}{\sum g(s, t)} - \frac{\sum r(s, t)}{\sum g(s, t)}^q$$

$$r = \frac{r(s, e)}{\sum g(s, t)} - \frac{r(s, e)}{\sum g(s, t)}$$

q is order of the filter & adjusting its value changes the filter's behavior

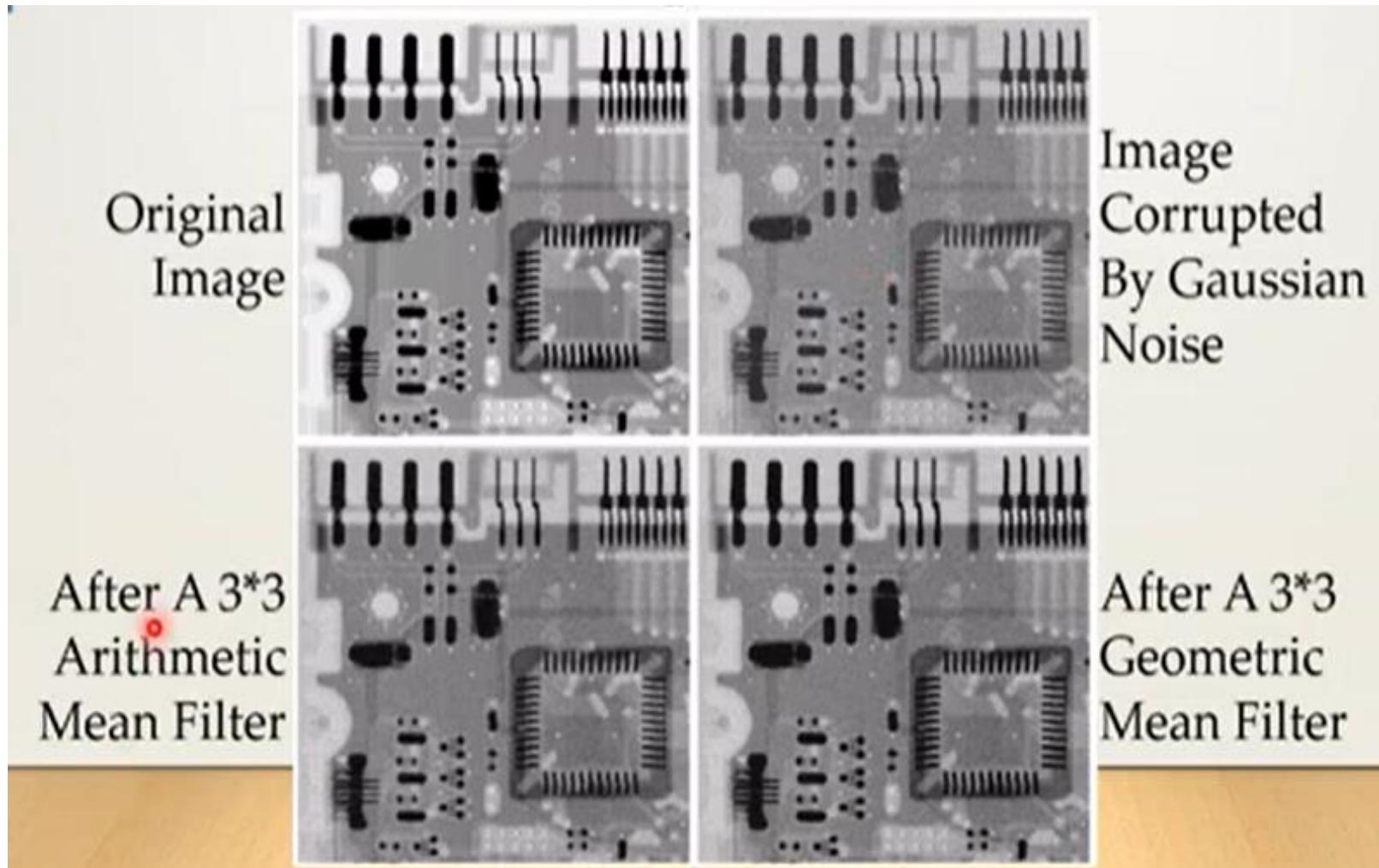
if $q > 0$ \therefore q eliminates Pepper noise

$q < 0$ \therefore " salt "

$q = 0$ \therefore arithmetic mean

$q = -1$ \therefore Hamonic filter

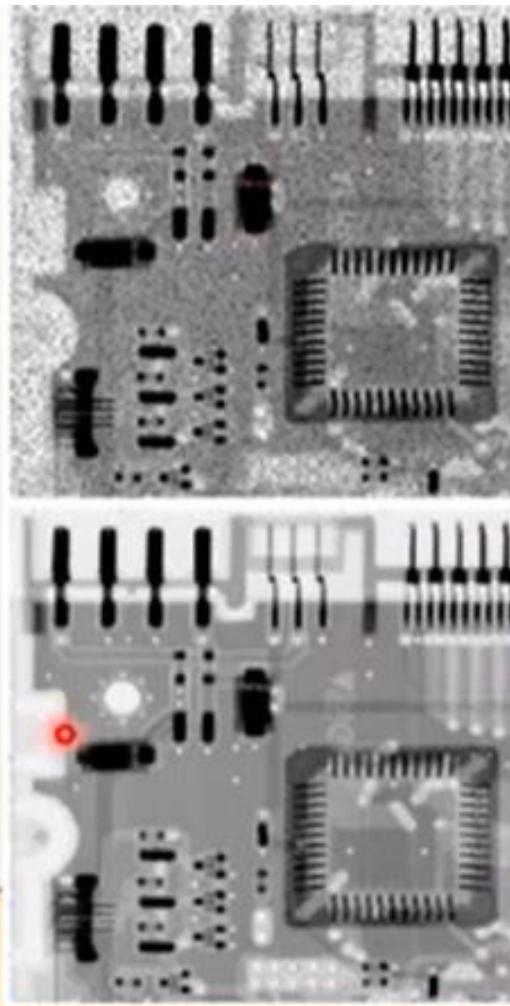
Noise Removal Examples



Noise Removal Examples

Image
Corrupted
By Pepper
Noise

Result of
Filtering Above
With 3×3
Contraharmonic
 $Q=1.5$



Noise Removal Examples

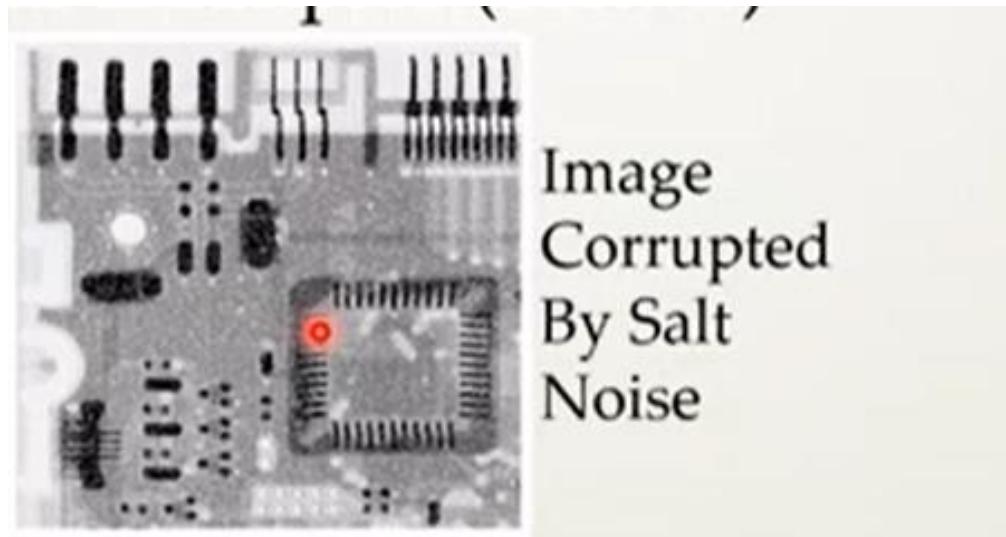
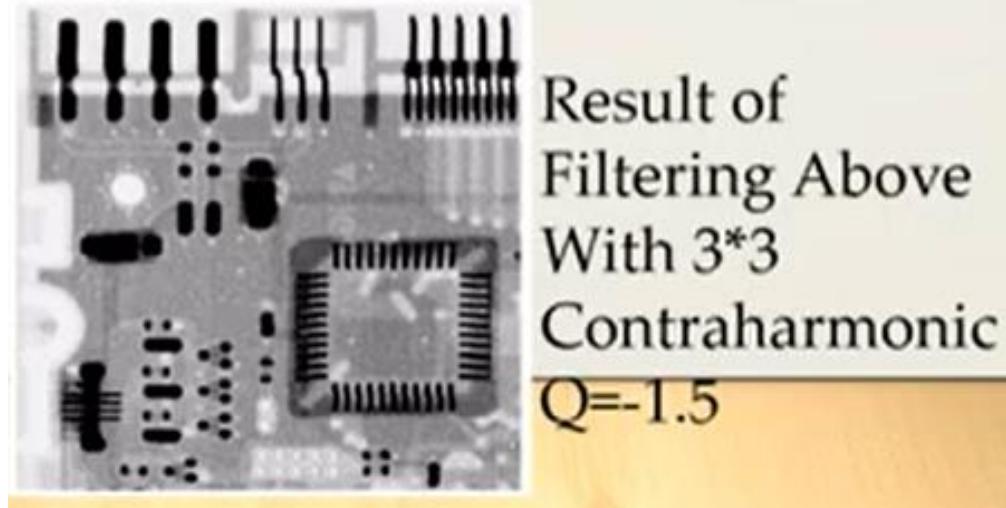


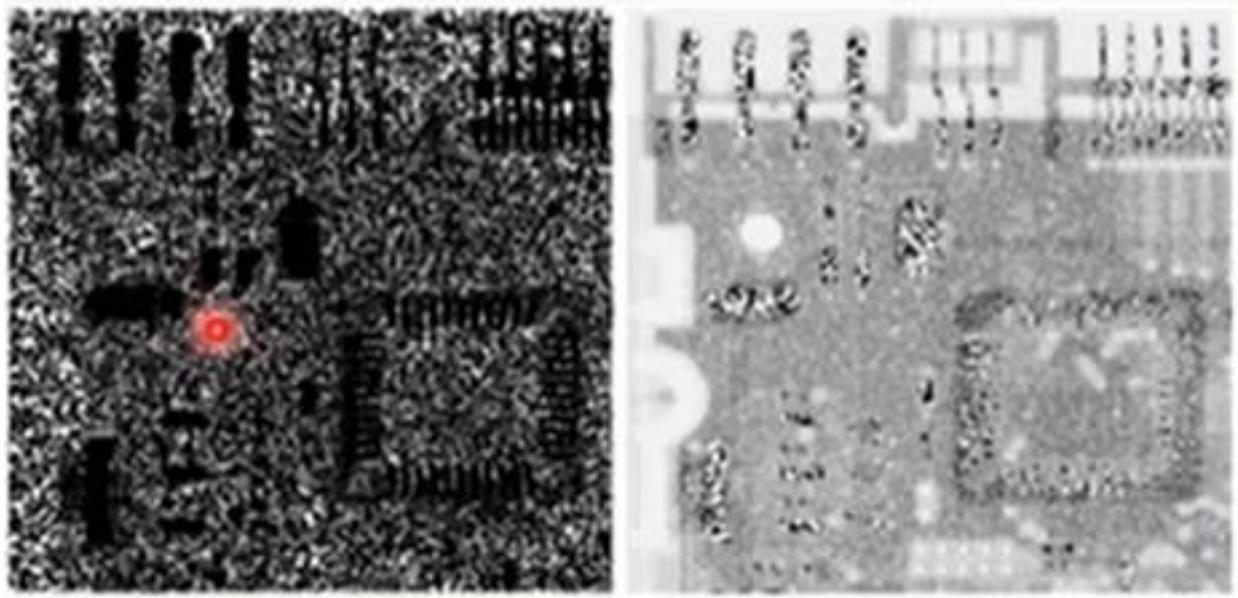
Image
Corrupted
By Salt
Noise



Result of
Filtering Above
With 3*3
Contraharmonic
 $Q=1.5$

Contraharmonic Filter

- Choosing the wrong value for Q when using contra harmonic filter can have drastic results



Noise Removal Order Statistics Filters

Order Statistics Filters

- Spatial filters that are based on ordering the pixel values that make up the neighborhood operated on by the filter
- Useful spatial filters include
 - Median filter
 - Max and Min filter
 - Midpoint filter
 - Alpha trimmed mean filter

Median Filter

- Median Filter:

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\operatorname{median}}\{g(s, t)\}$$

- Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters.
- Particularly good when salt and pepper noise is present.

Median Filter Example

Step 1: Replication

(1) Median filter:

$$f(x, y) = \text{median}(\text{vec}, +)$$

3, 9, 4, 52, 3, 8, 6, 2, 2, 9

(1) Replication:-

$$\boxed{3|9|9} = 3$$

$$\boxed{3|9|4} \rightarrow \boxed{3|4|9} = 4$$

$$\boxed{9|4|52} \rightarrow \boxed{4|9|52} = 9$$

$$\boxed{4|52|3} \rightarrow \boxed{1|3|4|52} = 4$$

$$\boxed{52|3|8} \rightarrow \boxed{1|3|4|52} = 8$$

$$\boxed{3|8|6} \rightarrow \boxed{3|6|8} = 6$$

$$\boxed{8|6|2} \rightarrow \boxed{2|6|8} = 6$$

$$\boxed{6|2|2} \rightarrow \boxed{2|2|6} = 2$$

$$\boxed{2|2|9} \rightarrow \boxed{2|2|9} = 2$$

$$\boxed{2|9|9} = 9$$

3 4 9 1 4 8 6 6 2 2 9

Step 2: Padding

(iii) Padding:

$$\begin{array}{|c|c|c|} \hline 0 & 3 & 9 \\ \hline \end{array} = 3$$

$$\begin{array}{|c|c|c|} \hline 2 & 9 & 0 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|} \hline 0 & 2 & 9 \\ \hline \end{array} = 12$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 3 & 4 & 9 & 4 & 8 & 6 & 1 & 6 & 2 & 2 & 1 & 2 \\ \hline \end{array}$$

Step 3: Trimming

(iii) Trimming:

without effect boundary compiler will

3	4	9	4	8	6	6	6	2	2	10	:	3	7	10
---	---	---	---	---	---	---	---	---	---	----	---	---	---	----

Max and Min Filter

Max Filter:

$$\hat{f}(x, y) = \max_{(s,t) \in S_{xy}} \{g(s, t)\}$$

Min Filter:

o

$$\hat{f}(x, y) = \min_{(s,t) \in S_{xy}} \{g(s, t)\}$$

Max filter is good for pepper noise and min is good for salt noise

Max and Min Filter Example

Step 1: Replication

Input Data: 3, 9, 4, 5, 2, 3, 8, 6, 2, 2, 9			Output Data: 1, 0, 1, 2, 1
Max, Min, Mid			
(i) Replication			
3 9 4	= 9	4	3
3 9 4	= 9	4	3
9 4 5 2	= 5	2	4
4 3 2 3	= 5	2	3

$$\boxed{52} \ 3 \ 18 = 52 \cdot 3 \approx 28$$

$$\boxed{1} \boxed{3} \boxed{8} \boxed{6} = 8 \quad 3 \approx 6$$

$$8 \boxed{6} \boxed{1} \boxed{2} = 8 \quad 2 \approx 5$$

$$[6 \boxed{2} \boxed{2}] \quad \boxed{1} \boxed{2} \boxed{6} \quad \boxed{2} \quad [P] = 6$$

$$2219 \times 2921812 \approx 6$$

$$\left[\begin{array}{|c|c|} \hline 2 & 9 \\ \hline 9 & 4 \\ \hline \end{array} \right] = 9 \cdot 2 - 9 \cdot 2 \approx 6$$

max: 4|9|52|52|52|8|8|6|9|9 | 10,000,000

min: 3 3 4 3 3 3 2 2 2 2 2 2 2 2

Step 2: Trimming

(ii) Trimming:

$$\boxed{3 \ 3 \ 9} \rightarrow (3, 3, 3) \div 3 = 3$$

$$\boxed{2 \ 2 \ 9} \div 9 = 9 \qquad 9 \div 9 = 9$$

max : 3 | 9 | 5 | 2 | 5 | 2 | 5 | 2 | 8 | 8 | 6 | 9 | 3

min : 3 | 3 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 9

Step 3: Padding

iii) Padding:

$$\boxed{0 \ 3 \ 9} \div 9$$

$$0 \div 5$$

$$\boxed{2 \ 9 \ 0} \div 9$$

$$0 \div 5$$

max

9 9 5 2 5 2 5 2 8 8 6 9 3 4

min

1 0 9 4 3 3 3 2 2 2 0

Midpoint Filter

Midpoint Filter:

$$\hat{f}(x,y) = \frac{1}{2} \left[\max_{(s,t) \in S_{xy}} \{g(s,t)\} + \min_{(s,t) \in S_{xy}} \{g(s,t)\} \right]$$

Good for random Gaussian and uniform noise

Midpoint Filter Example

i) Replication:

3|8|9 = max min / 8

midPoint: 5|6|28|28|28|6|5|6|6|6

ii) Trimming:

midPoint: 3|6|28|28|28|6|5|6|6|9

iii) Padding:

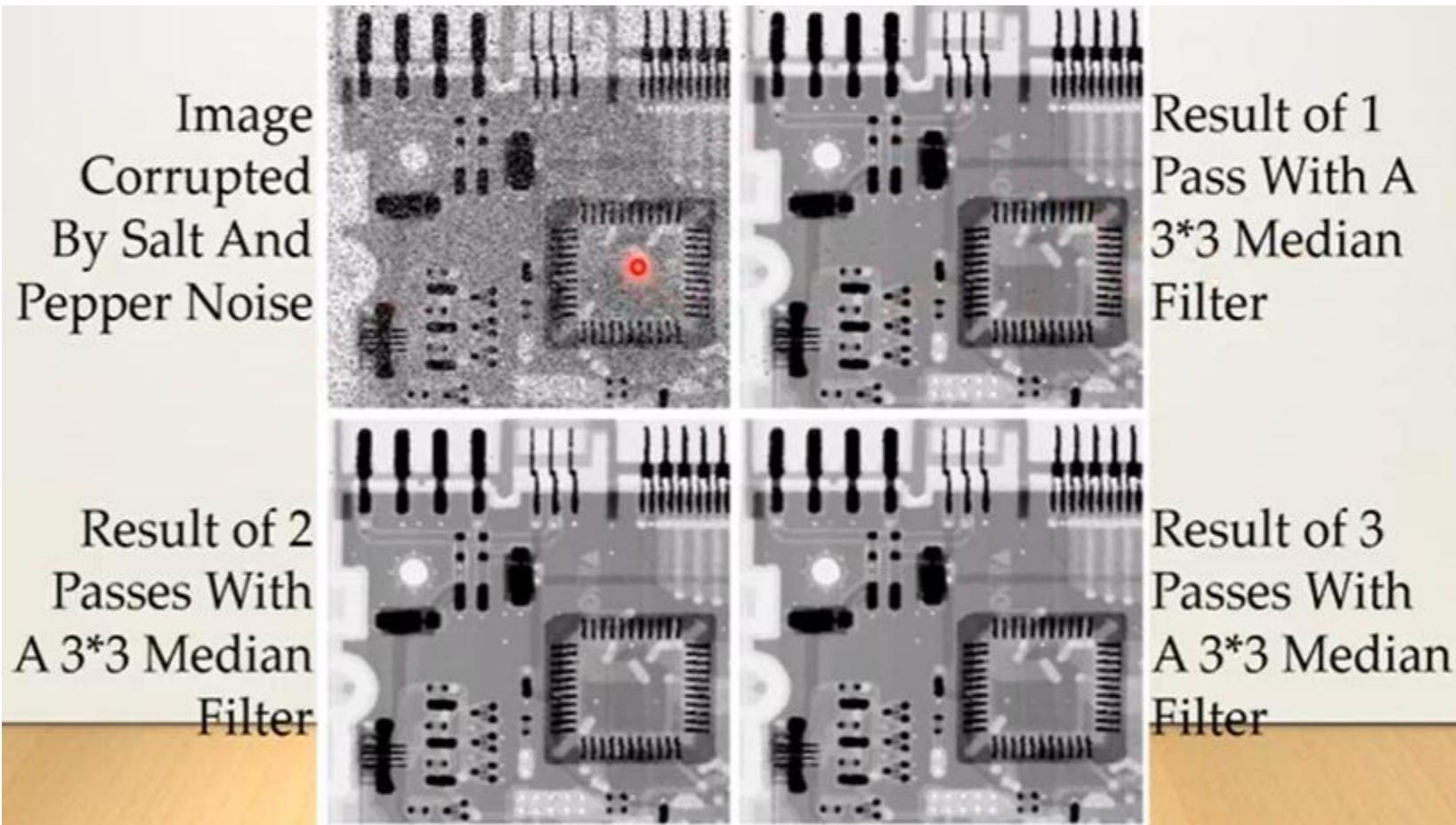
midPoint: 5|6|28|28|28|28|6|5|6|6|5

Alpha-Trimmed Mean Filter

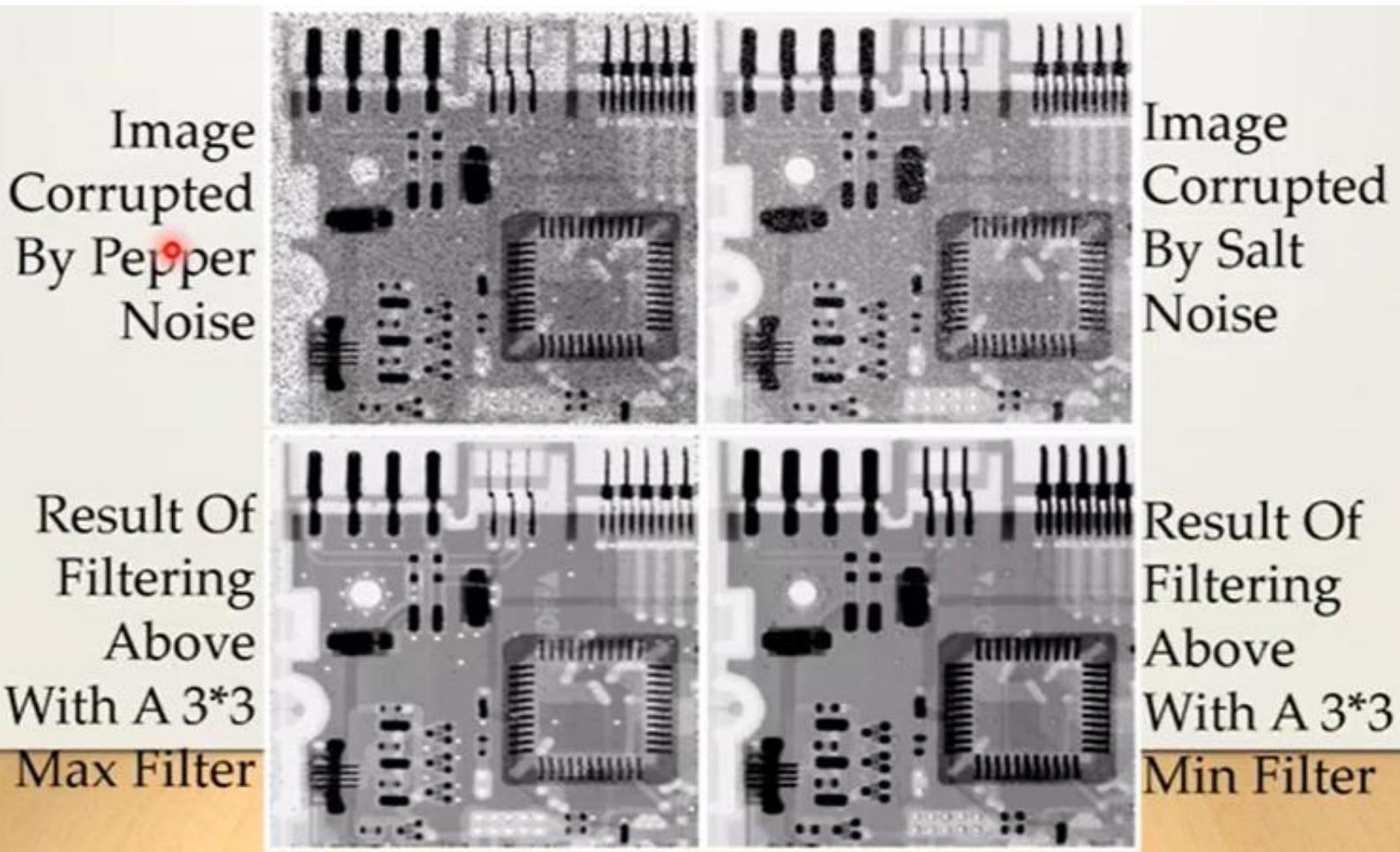
$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s, t) \in S_{xy}} g_r(s, t)$$

- We can delete the $d/2$ lowest and $d/2$ highest grey levels
- So $g_r(s, t)$
represents the remaining $mn - d$ pixels

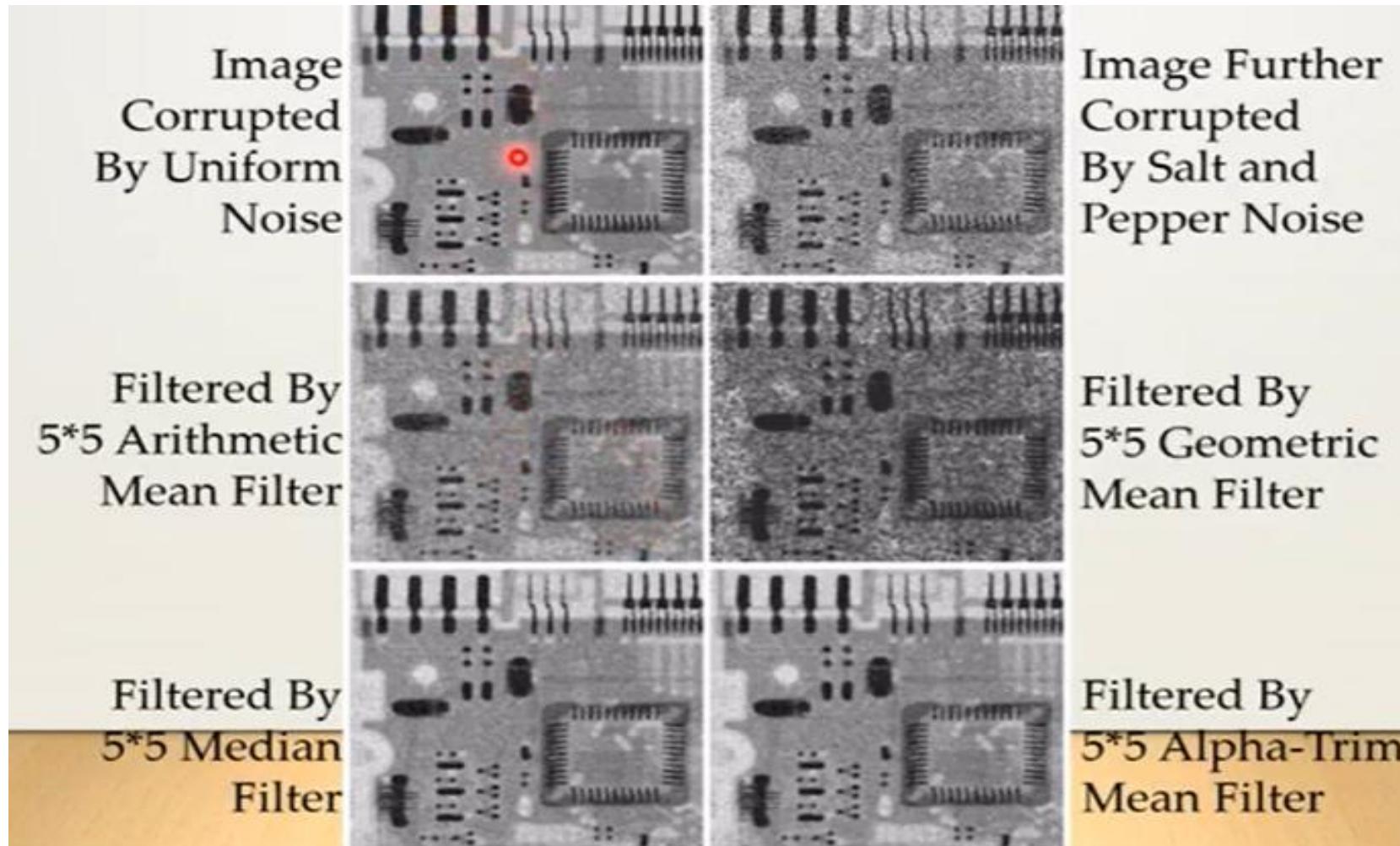
Noise Removal Examples



Noise Removal Examples



Noise Removal Examples



Periodic Noise

- Typically arises due to electrical or electromagnetic interference
- Gives rise to regular noise patterns in image
- Frequency domain techniques in the Fourier domain are most effective at removing periodic noise



Image Enhancement Fourier Transformation

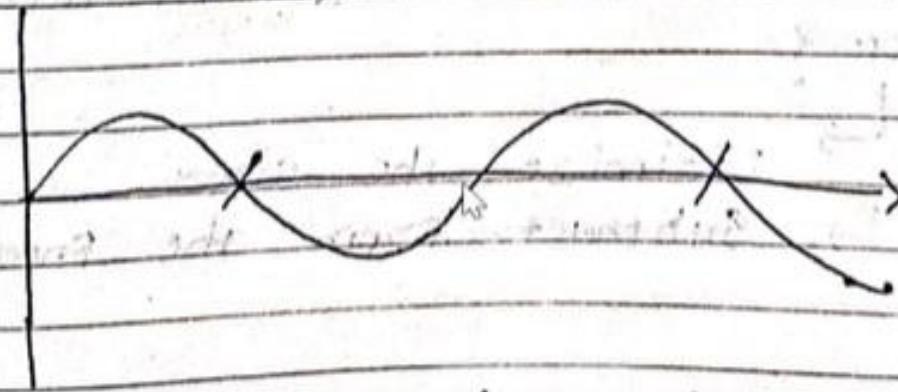
Filter and Frequency

- **Frequency:** The number of times that a periodic function repeats the same sequence of values during a unit variation of the independent variable.
- **Filter:** A device or material for suppressing or minimizing waves or oscillations of certain frequencies

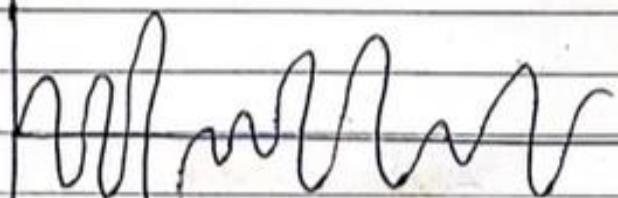
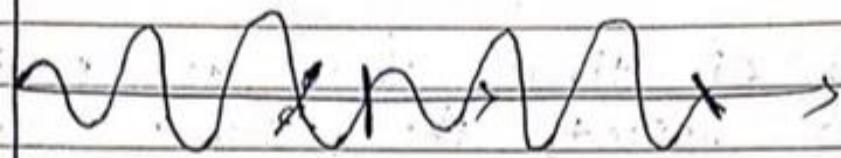
① Frequency :-

- Some set of data how many time repeats.
- The no. of time that a periodic function repeat the same sequence of value during the unit variation of independent variable.

① Frequency :-

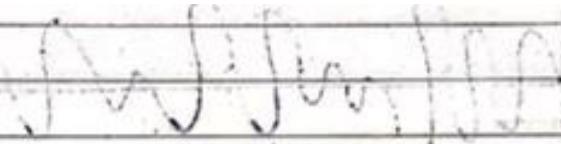


\therefore Deter must be same type.



\therefore Different deter must be repeat at
the same Periods of time T .

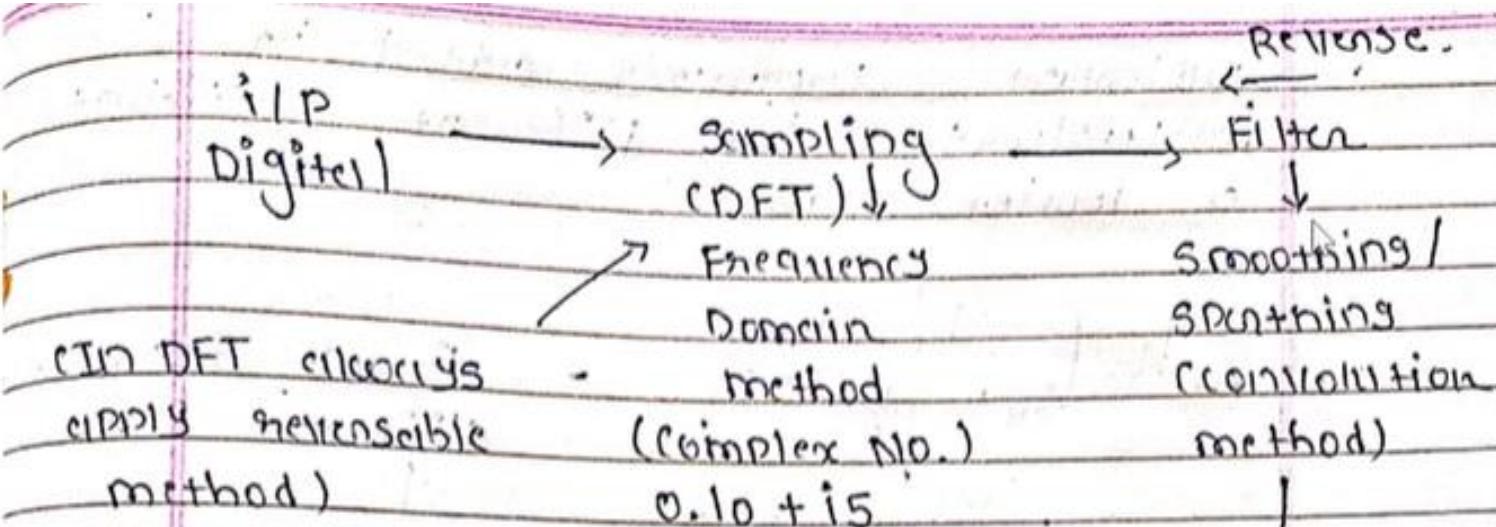
→ ② Filter :-



- :- Minimize the noise from the frequency.
- :- A device / material for suppressing or minimizing waves on oscillations of certain frequency.

:- eliminate noise from frequency
↳ How?

- ↳ Minimize the size
- ↳ Subtract from the frequency.

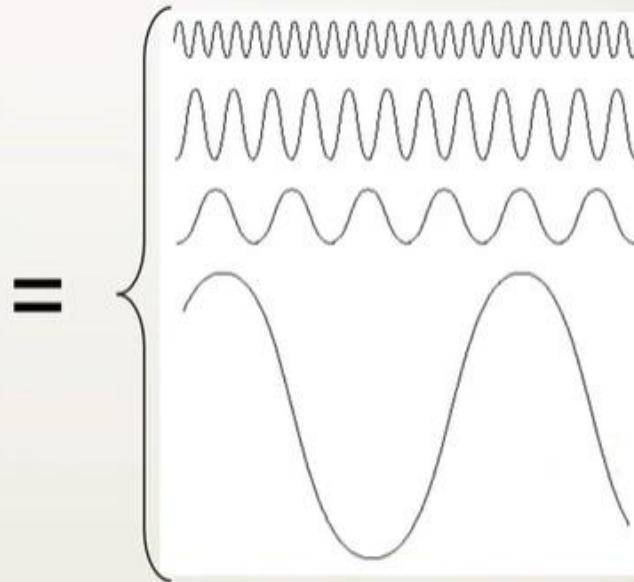


generate
method enhance
will be image

machine can not
display to, the
complex No. then
we neglect to
imagine part.

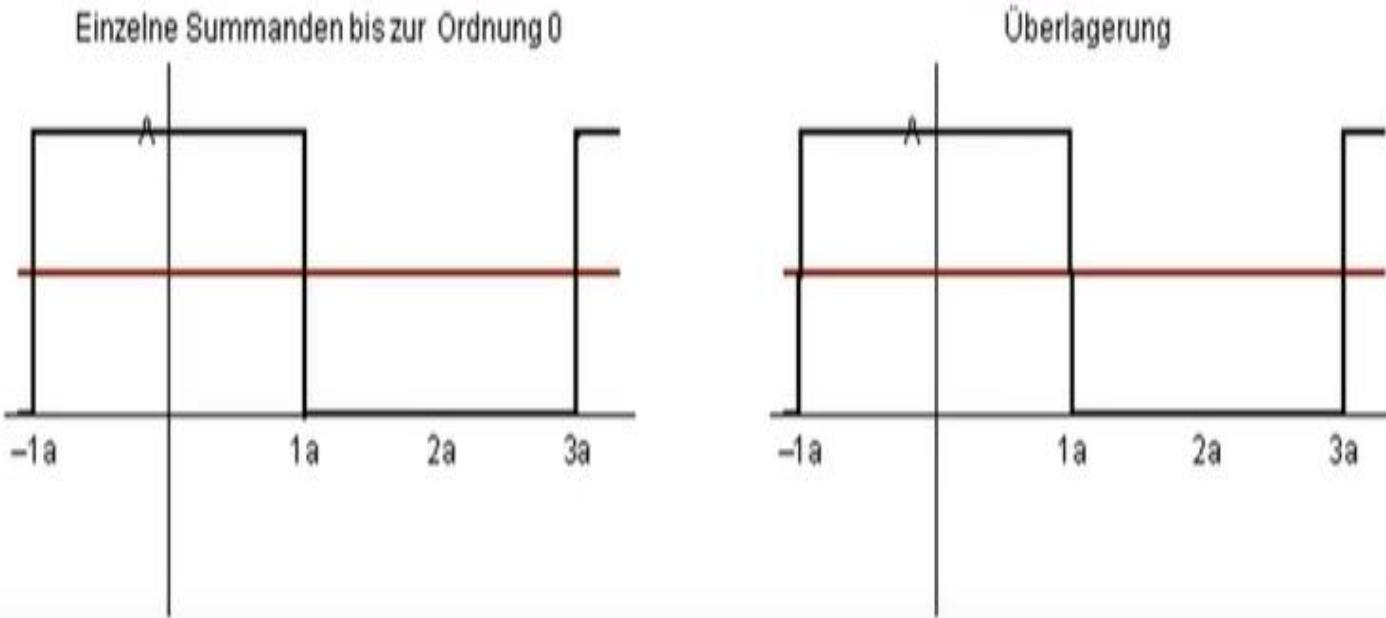
The Big Idea

The Big Idea



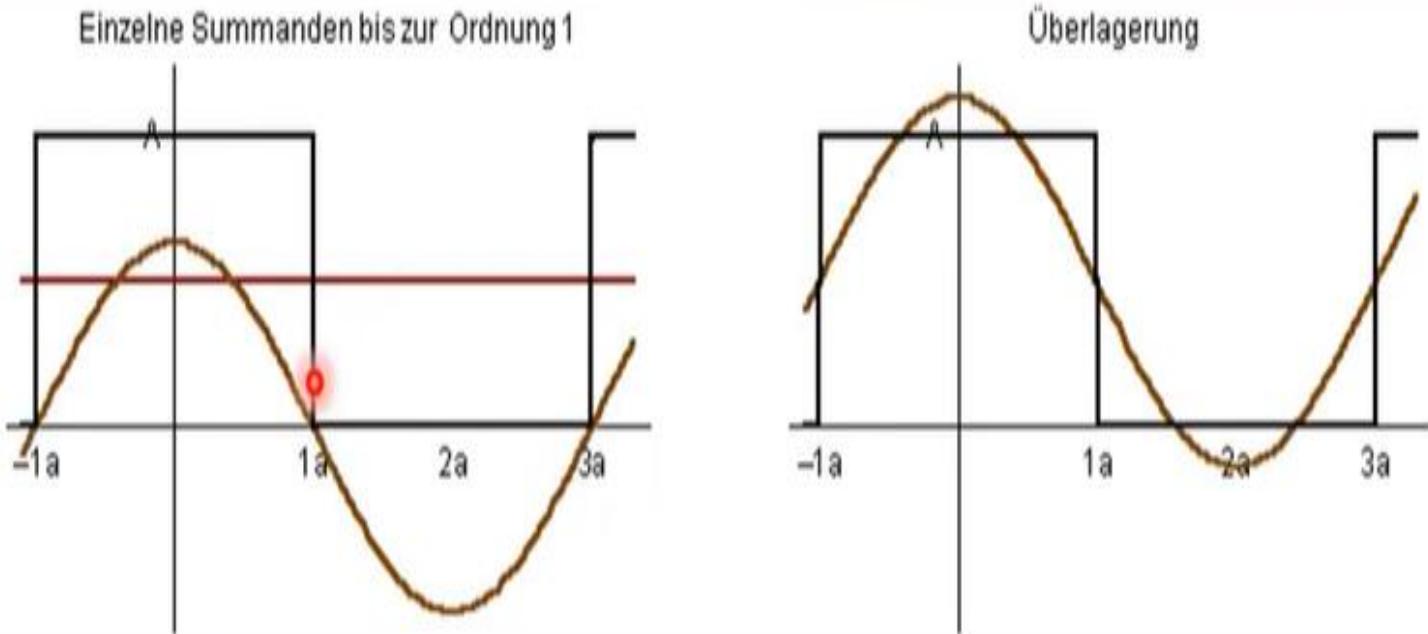
- Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*

The Big Idea(Cont...)



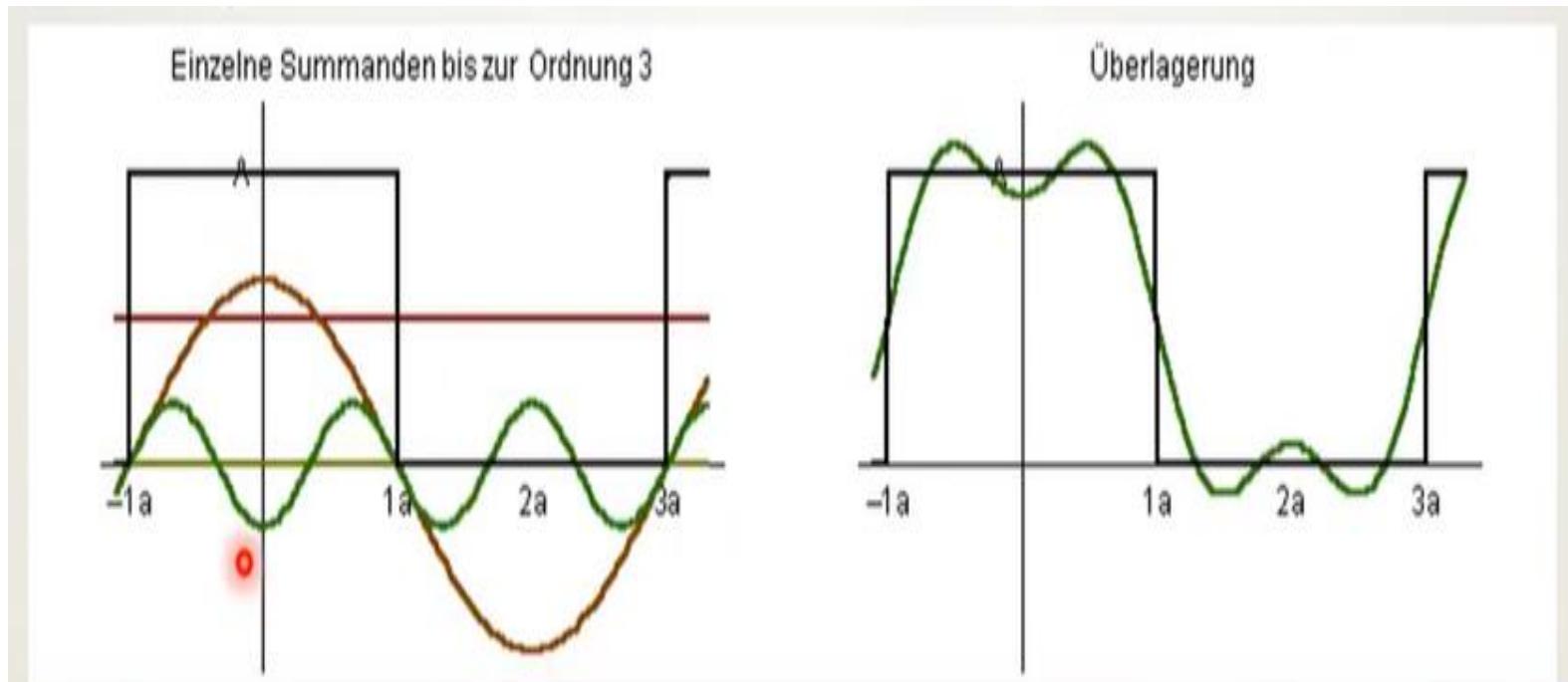
Notice how we get closer and closer to the original function as we add more and more frequencies

The Big Idea(Cont...)



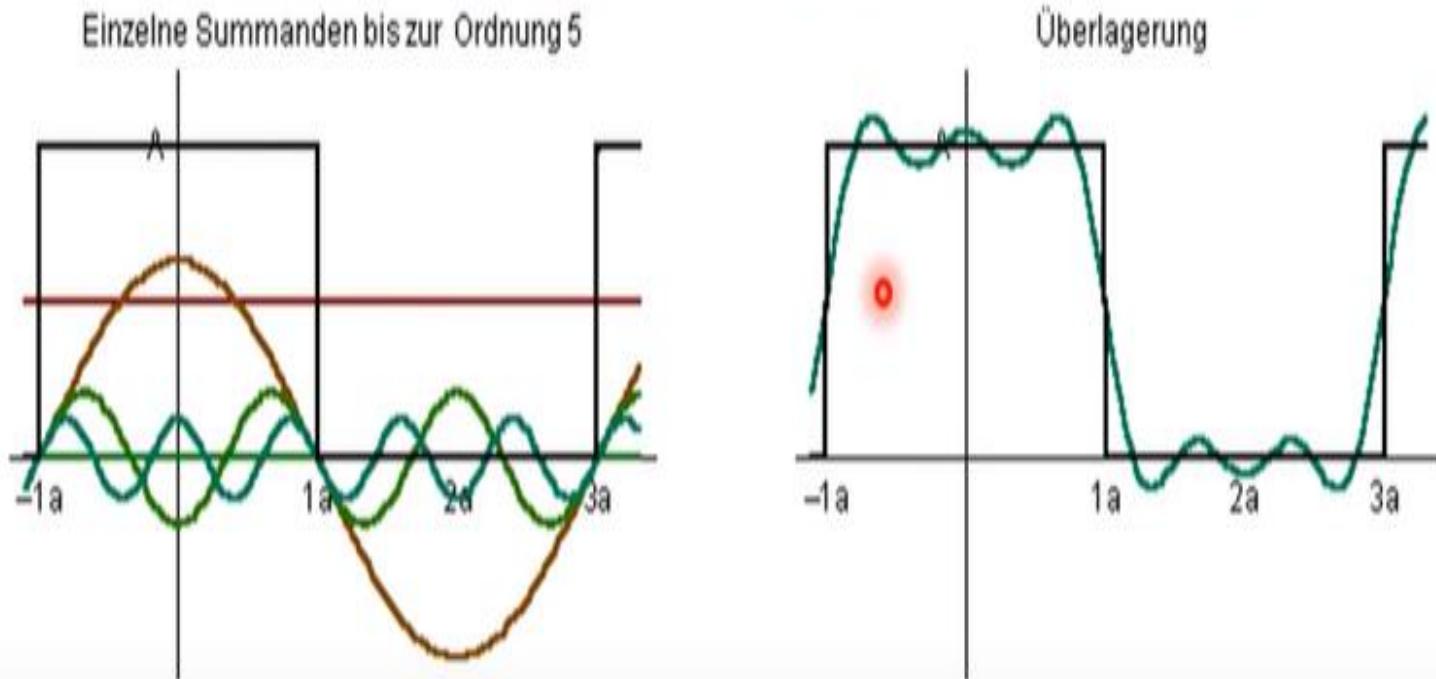
Notice how we get closer and closer to the original function as we add more and more frequencies

The Big Idea(Cont...)



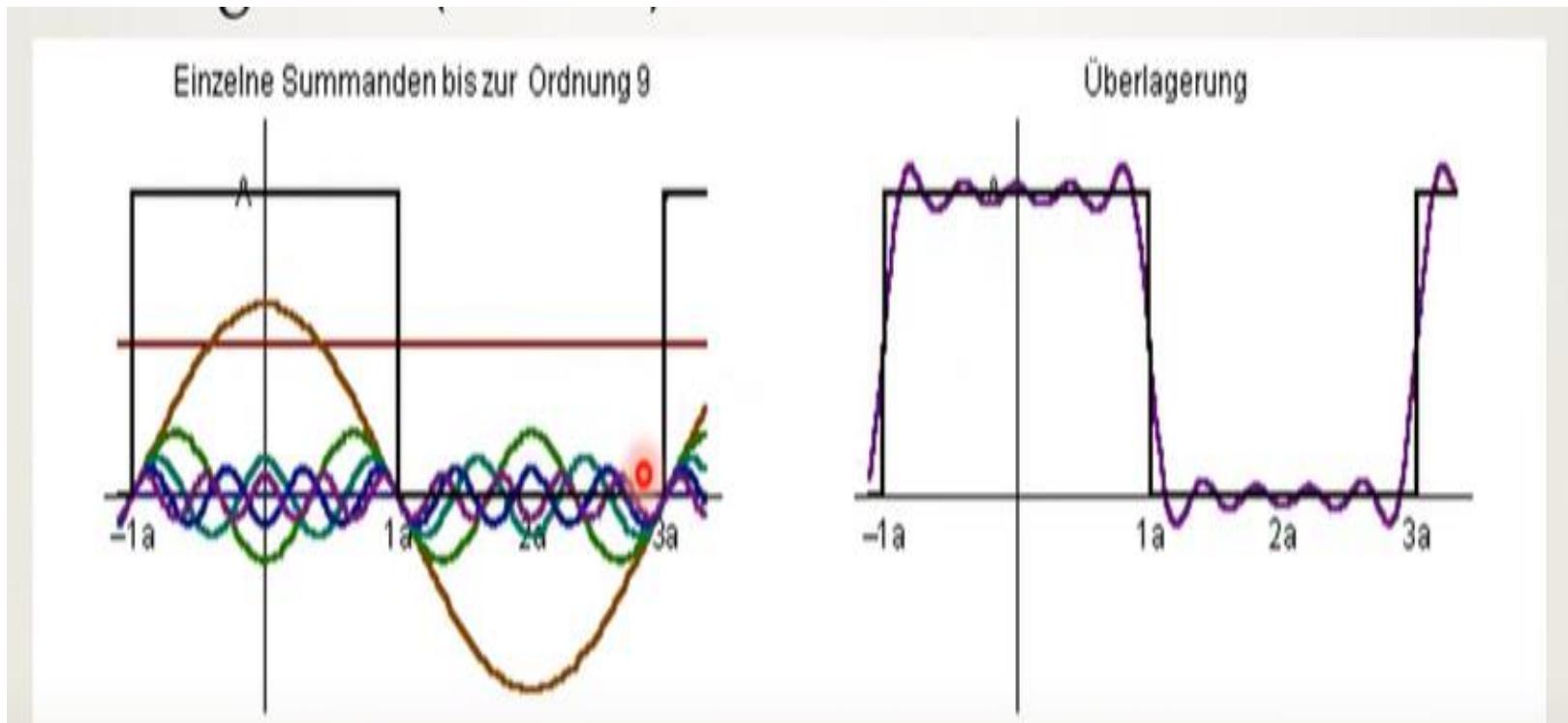
Notice how we get closer and closer to the original function as we add more and more frequencies

The Big Idea(Cont...)

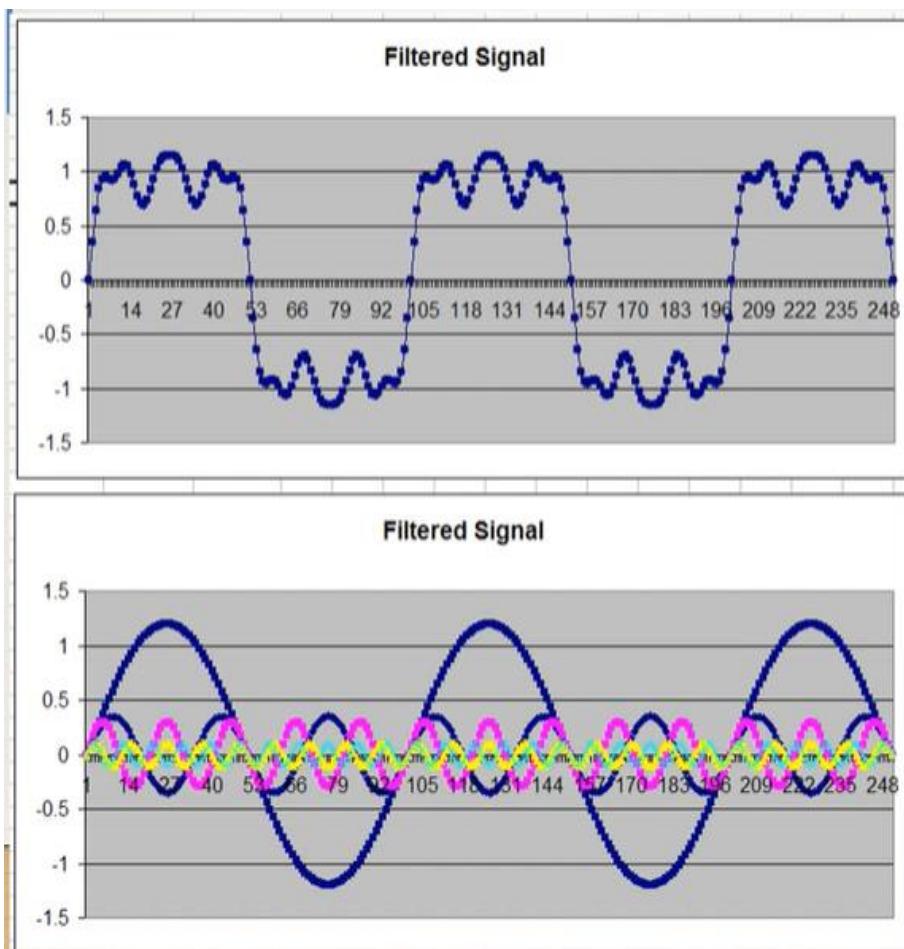


Notice how we get closer and closer to the original function as we add more and more frequencies

The Big Idea(Cont...)



Notice how we get closer and closer to the original function as we add more and more frequencies



Frequency
domain signal
processing
example in Excel

Discrete Fourier Transform(DFT)

- The Discrete Fourier Transform of $f(x,y)$ for $x=0,1,2\dots M-1$ and $y=0,1,2,\dots,N-1$, denoted by $F(u,v)$ is given by the equation

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)}$$

for $u = 0, 1, 2\dots M-1$ and $v = 0, 1, 2\dots N-1$.

Discrete Fourier transform(DFT)



Discrete Fourier transform (DFT)

$$x = 0, 1, 2, \dots, M-1$$
$$y = 0, 1, 2, \dots, N-1$$

2D image:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi j \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cdot e^{-2\pi j \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

$$u = 0, 1, \dots, M-1$$

$$v = 0, 1, \dots, N-1$$

1D

image for x :

$$F(u) = \sum_{x=0}^{M-1} f(x) \cdot e^{-2\pi j \left(\frac{ux}{M} \right)}$$
$$u = 0, 1, 2, \dots, M-1$$

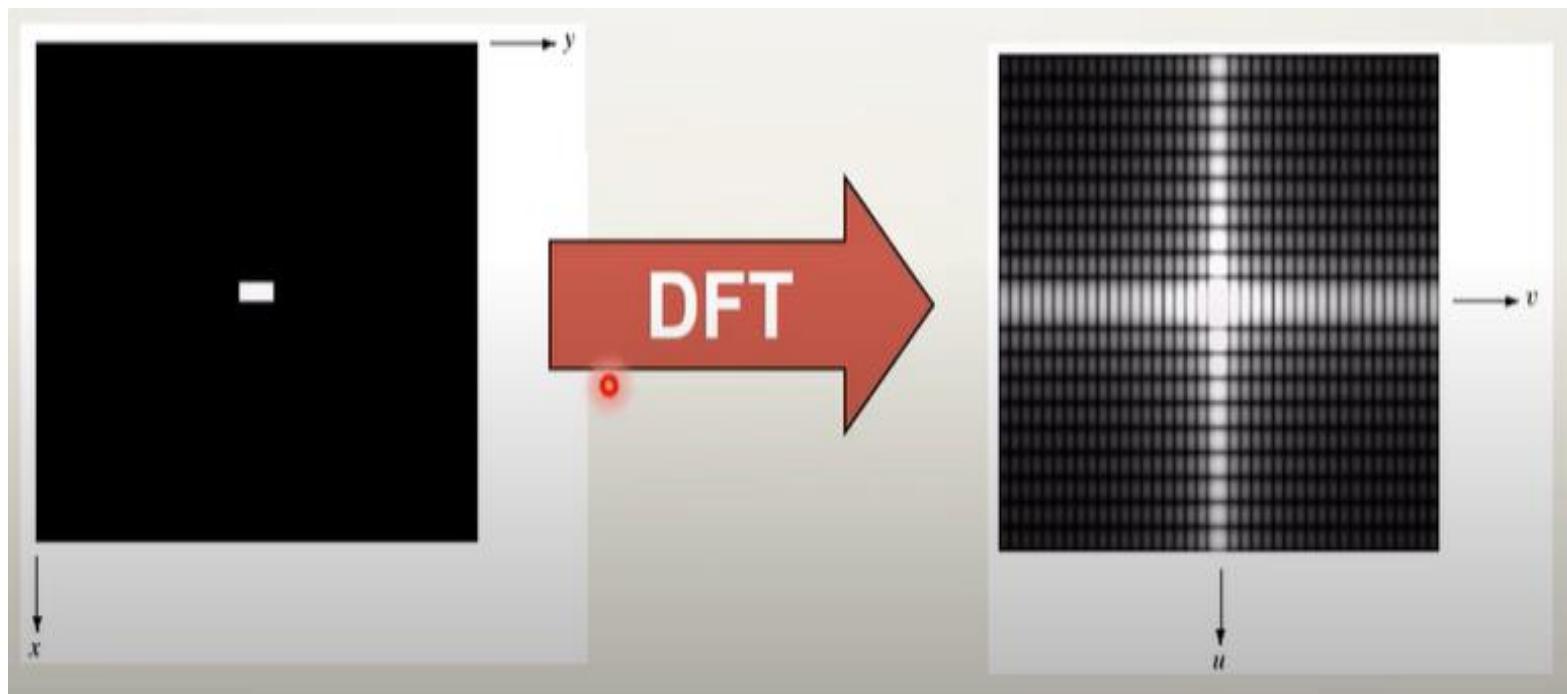
1D

image for y :

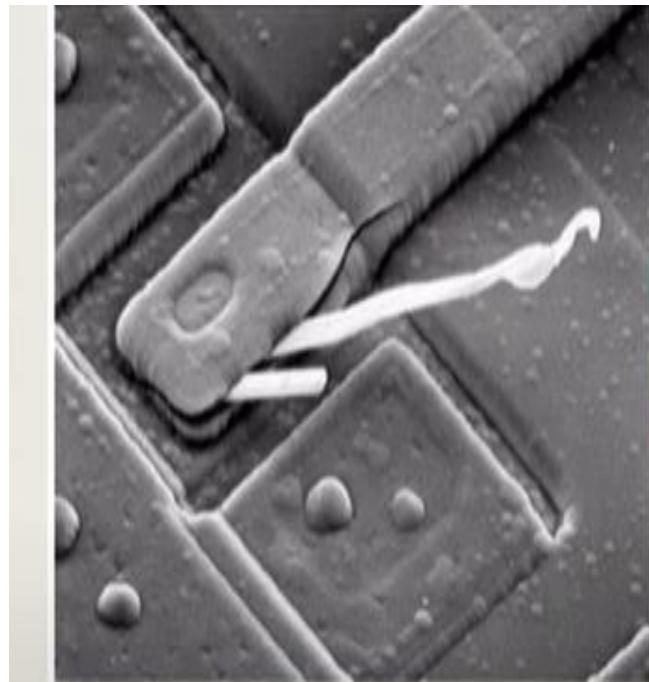
$$F(v) = \sum_{y=0}^{N-1} f(y) \cdot e^{-2\pi j \left(\frac{vy}{N} \right)}$$

DFT & Images

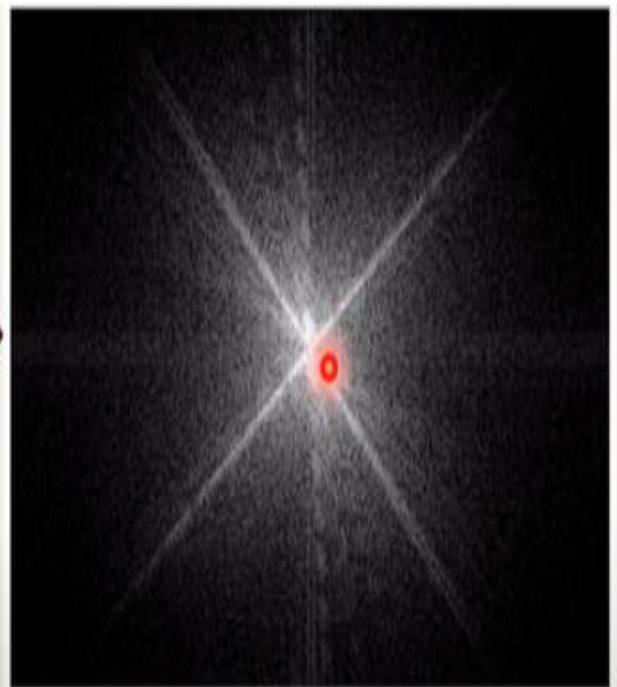
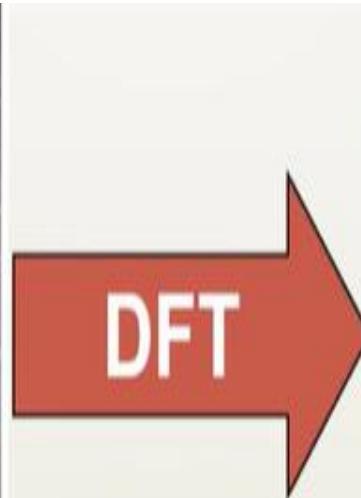
- The DFT of a two dimensional image can be visualised by showing the spectrum of the image component frequencies



DFT & Image(Cont...)



Scanning electron microscope
image of an integrated circuit
magnified ~2500 times



Fourier spectrum of the image

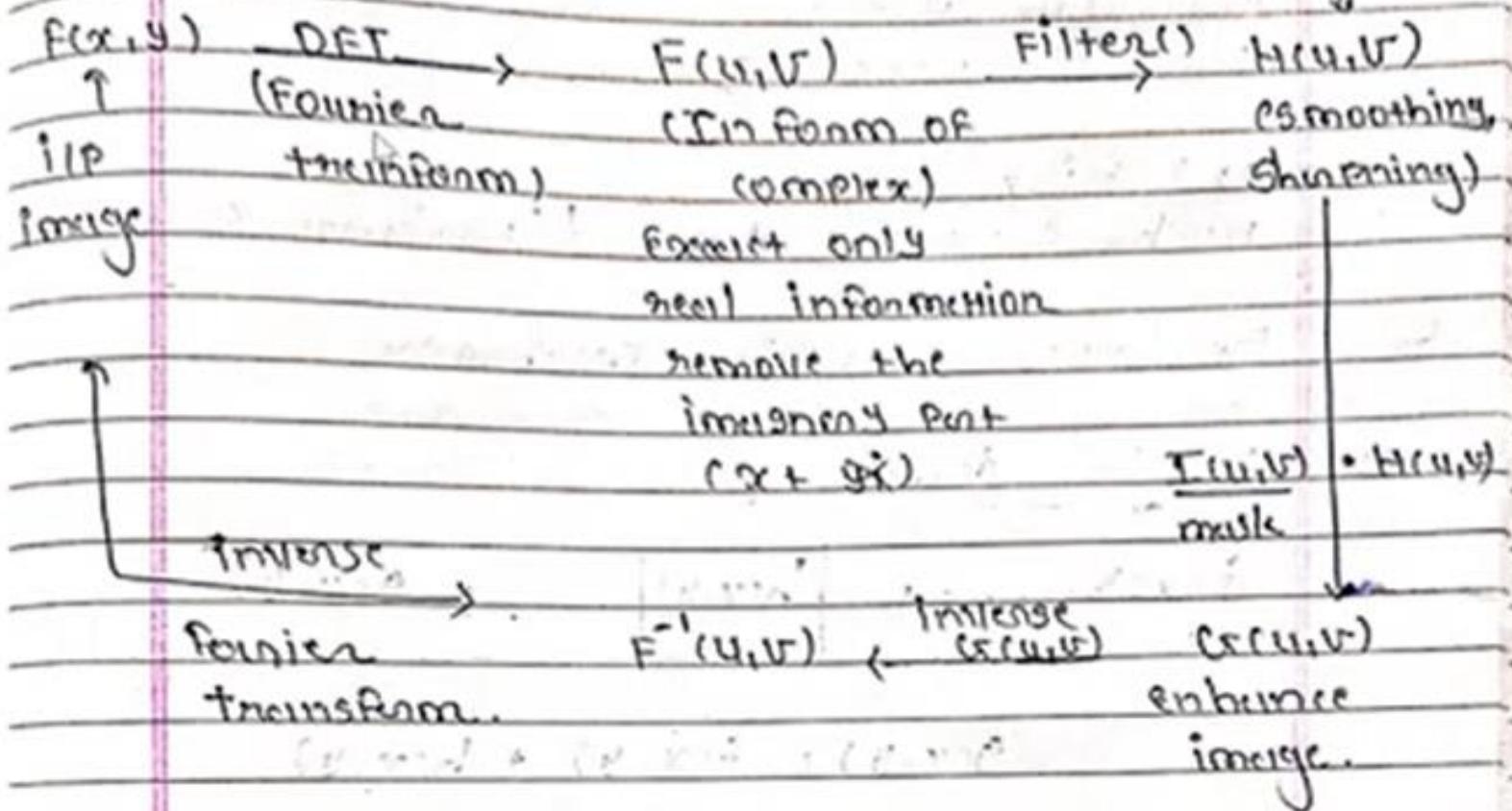
Inverse DFT

- It is really important to note that the Fourier transform is completely reversible
- The inverse DFT is given by:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$$

- For $x=0,1,2\dots M-1$ and $y=0,1,2\dots N-1$

→ The DFT in image processing:



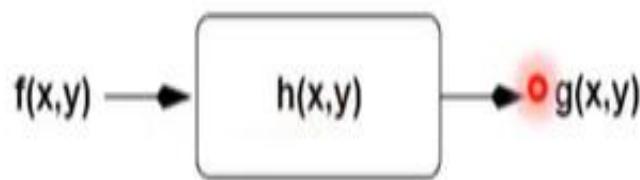
Steps

Steps :-

1. Apply DFT on $f(x,y)$ = (real image)
2. give us $F(u,v)$ = (DFT image)
↳ Remove Imaginary part only - real value.
3. Apply filter on $F(u,v)$
↳ create $H(u,v)$
4. $H(u,v)$ apply mask $T(u,v)$ for the
↳ sharpening & smoothing.
5. give us $G(u,v)$ = (enhanced image)
- $G(u,v)$ apply inverse $F^{-1}(u,v) \rightarrow$ inverse
filter transform & give - real $f(x,y)$
image.

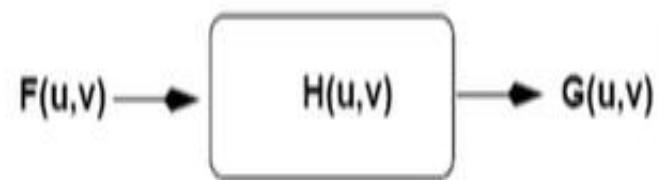
Frequency Domain Methods

Spatial Domain



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

Frequency Domain



$$G(u,v) = F(u,v) H(u,v)$$

$$(g(x,y) = \mathcal{F}^{-1}(F(u,v) H(u,v)))$$

↳ Frequency Domain filters :-

(1) Low pass :-

:- low frequency pass.

(2) High pass :-

:- high frequency pass. / (Attenuate pass)

↳ Frequency domain method :-

(1) spatial domain :-

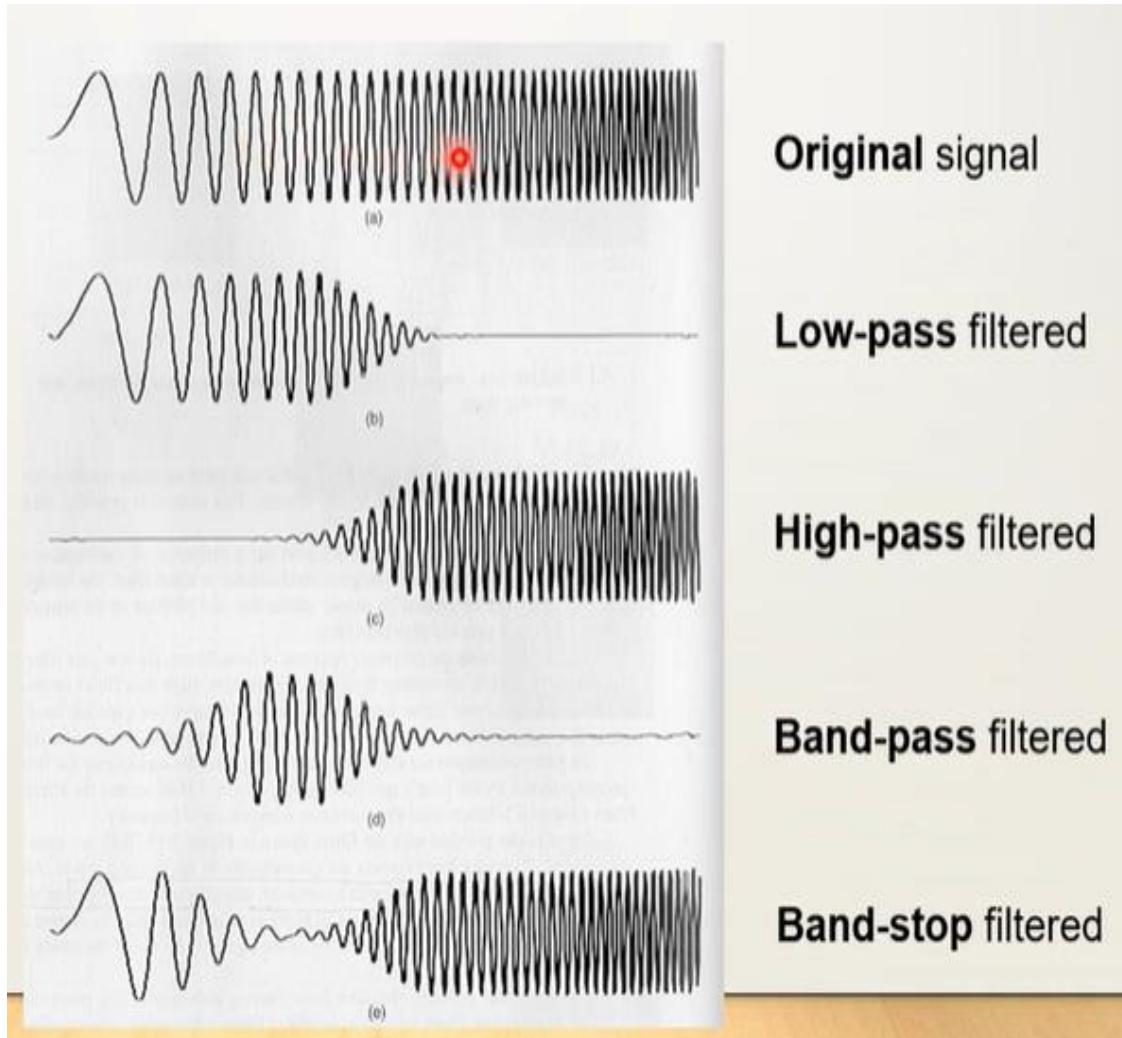
$$f(x,y) \xrightarrow{\text{filter}} [h(x,y)] \xrightarrow{\text{convolution}} g(x,y)$$

$$g(x,y) = f(x,y) * h(x,y).$$

Major Filter Categories

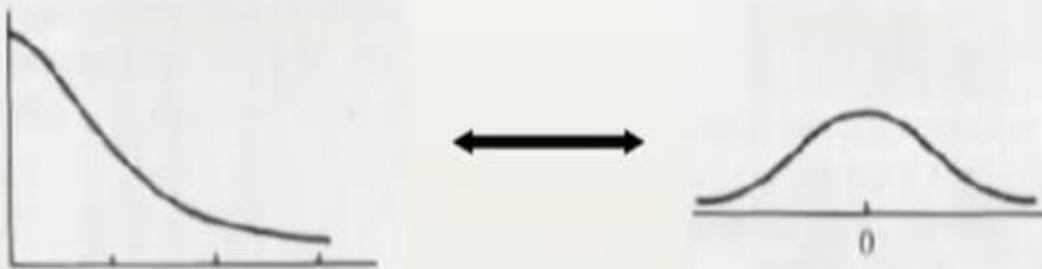
- Typically, filters are classified by examining their properties in the frequency domain:
 - (1)Low-pass
 - (2)High-pass
 - (3)Band-pass
 - (4)Band-stop

Example

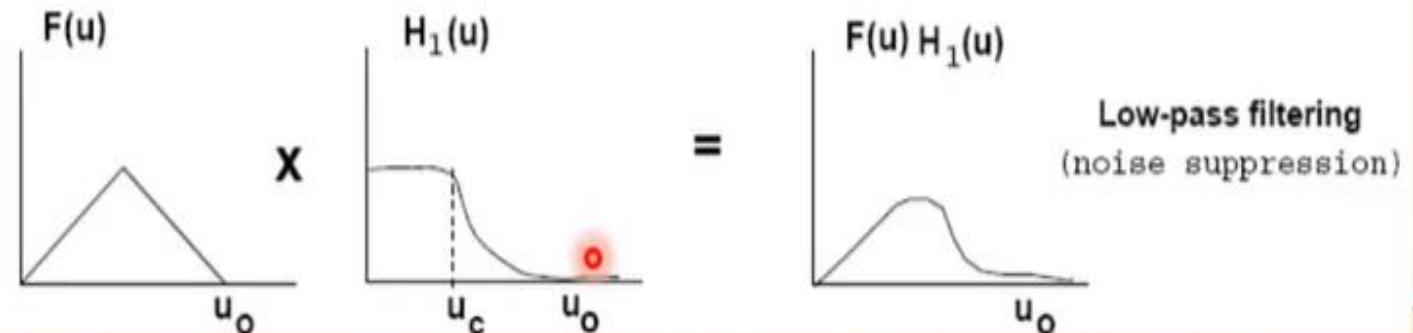


Low-pass filters(i.e., Smoothing filter)

- Preserve low frequencies - useful for noise suppression

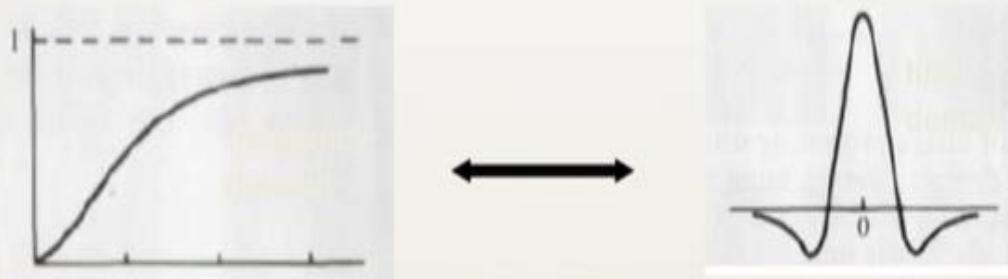


Example:

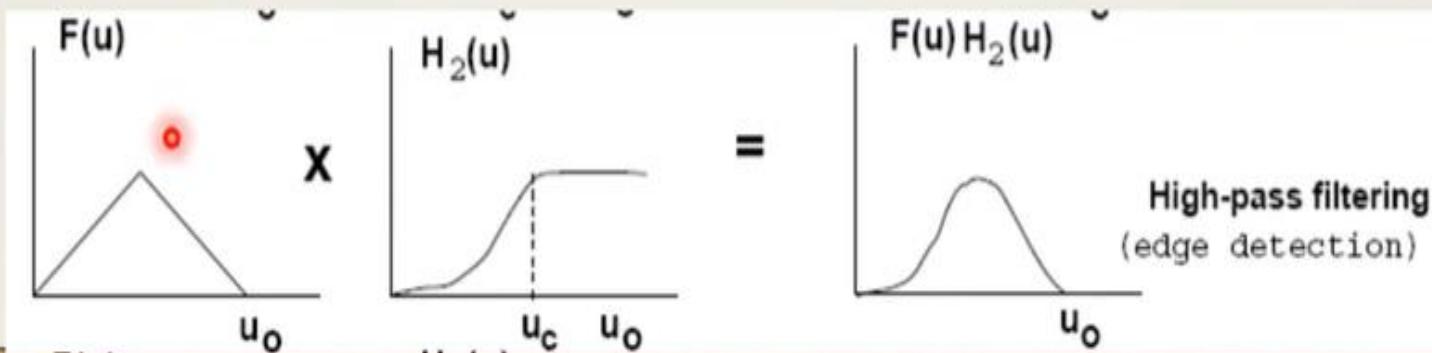


High-pass filters(i.e., Sharpening filters)

- Preserves high frequencies - useful for edge detection

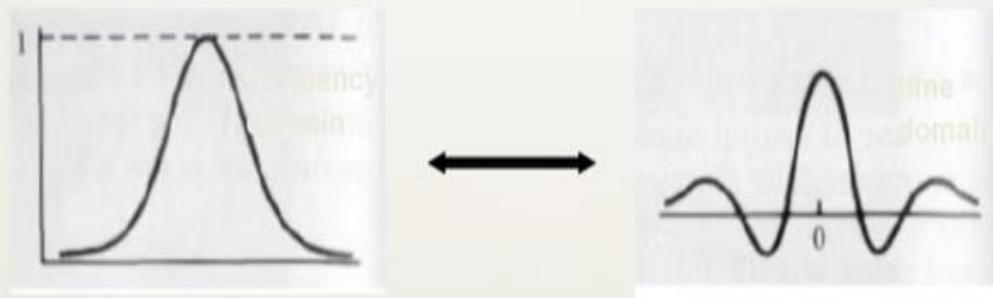


Example:

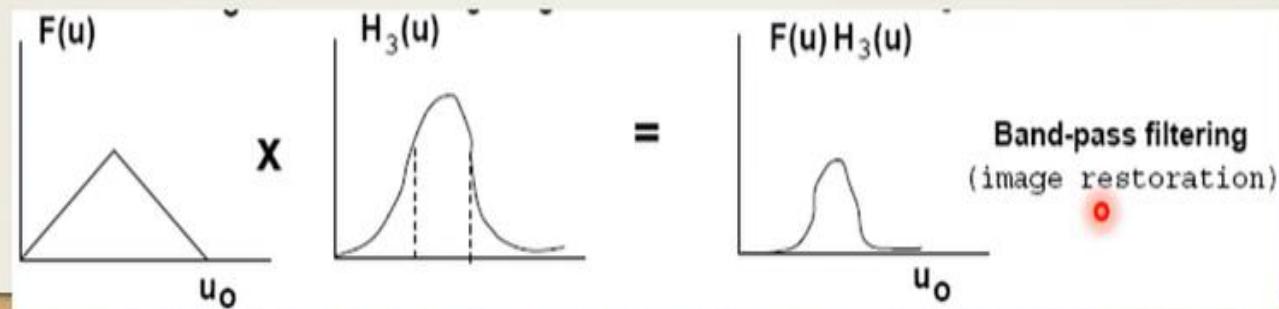


Band-Pass Filters

- Preserves frequencies within a certain band



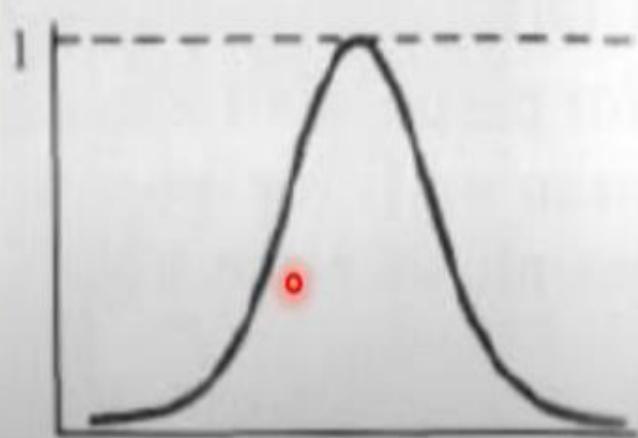
Example:



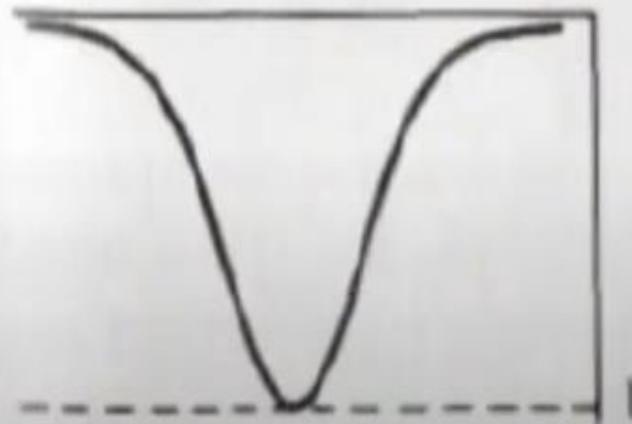
Band-stop Filters

- How do they look like?

Band-pass



Band-stop



Frequency Domain Method

Frequency Domain Met

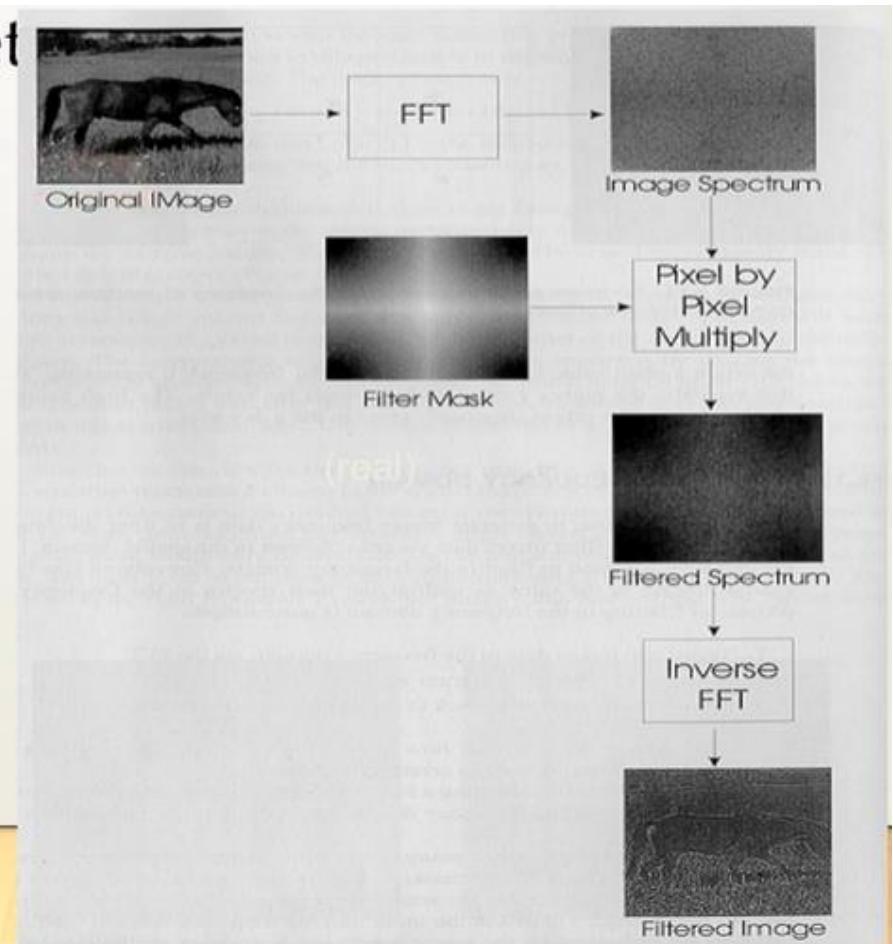
$$f(x, y) * h(x, y) = g(x, y)$$



$$F(u, v) H(u, v) = G(u, v)$$

Case 1: $H(u, v)$ is specified in the frequency domain.

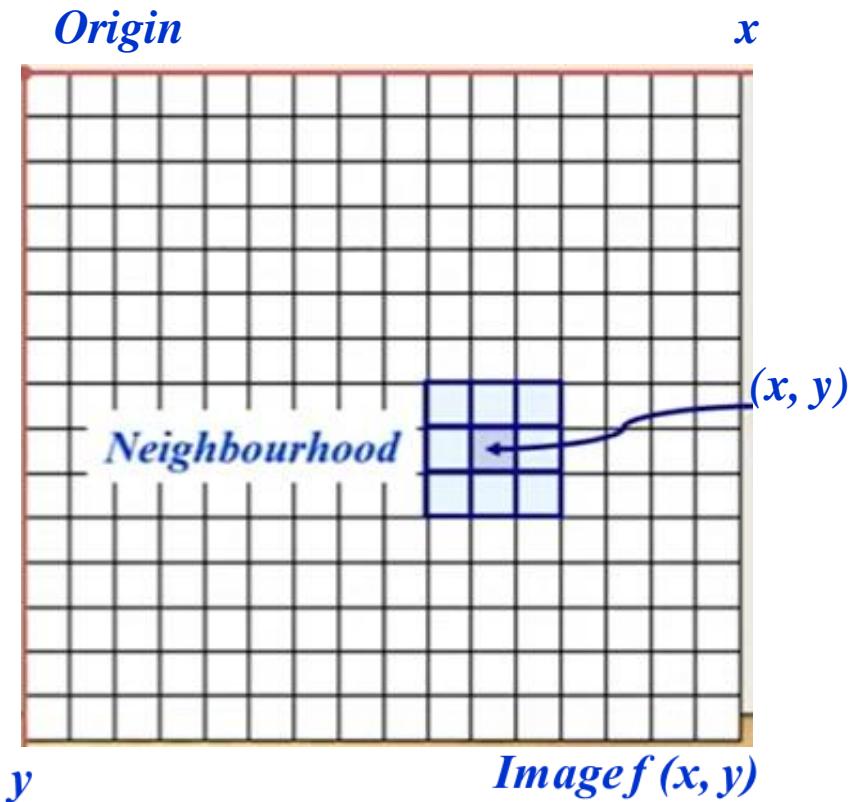
Case 2: $h(x, y)$ is specified in the spatial domain.



Filtering

Neighbourhood Operations

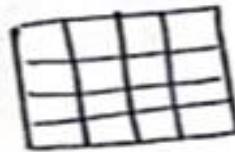
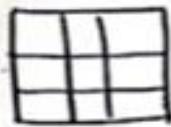
- Neighborhood operations simply operate on a larger neighborhood of pixels than point operations
- Neighborhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible



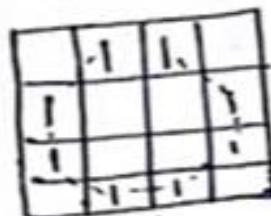
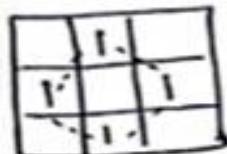
1) Neighbourhood

॥ Neighbourhood :-

; - Mask is always in rectangle / square in any $n \times m$.



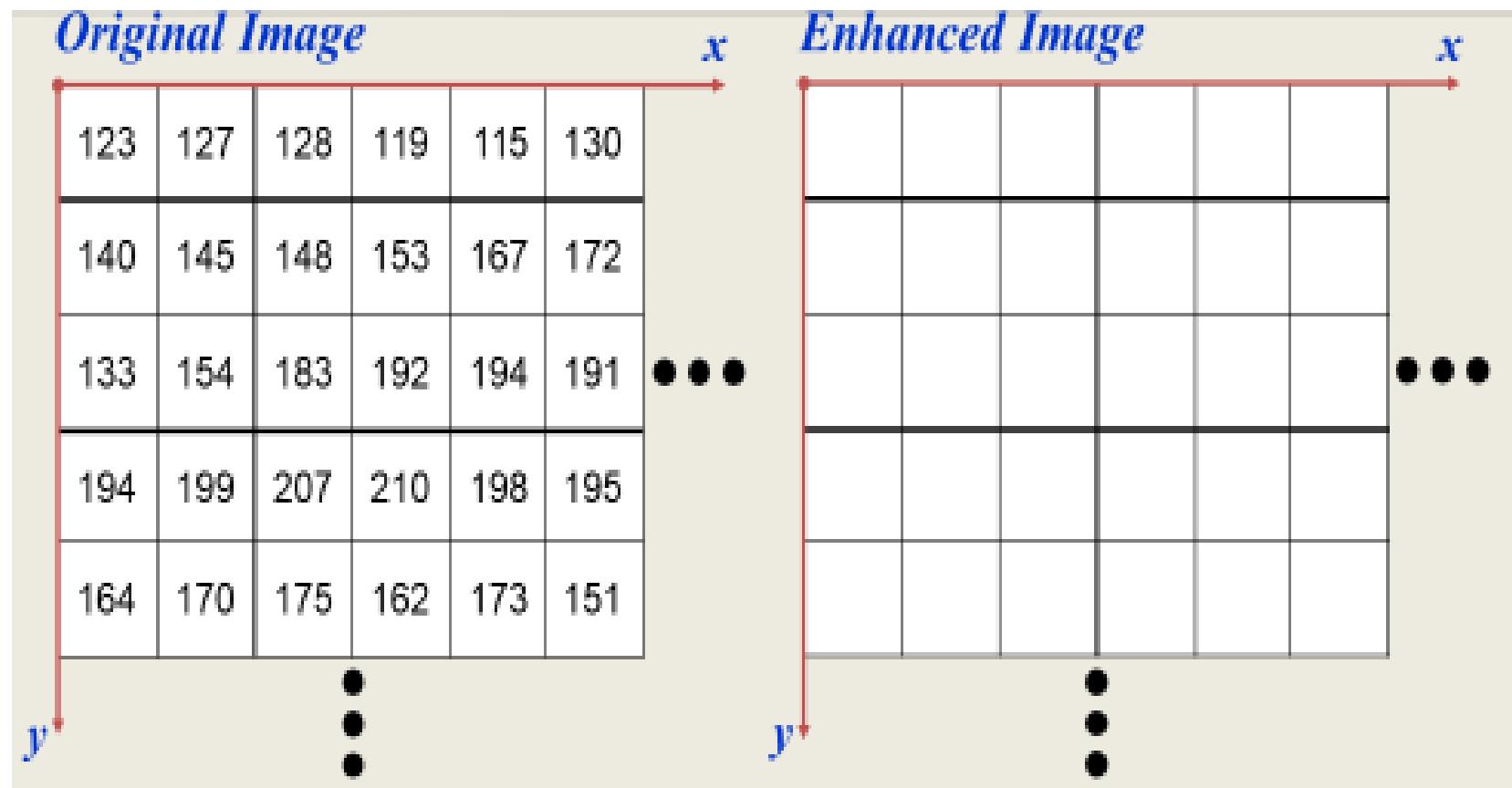
; - Shape



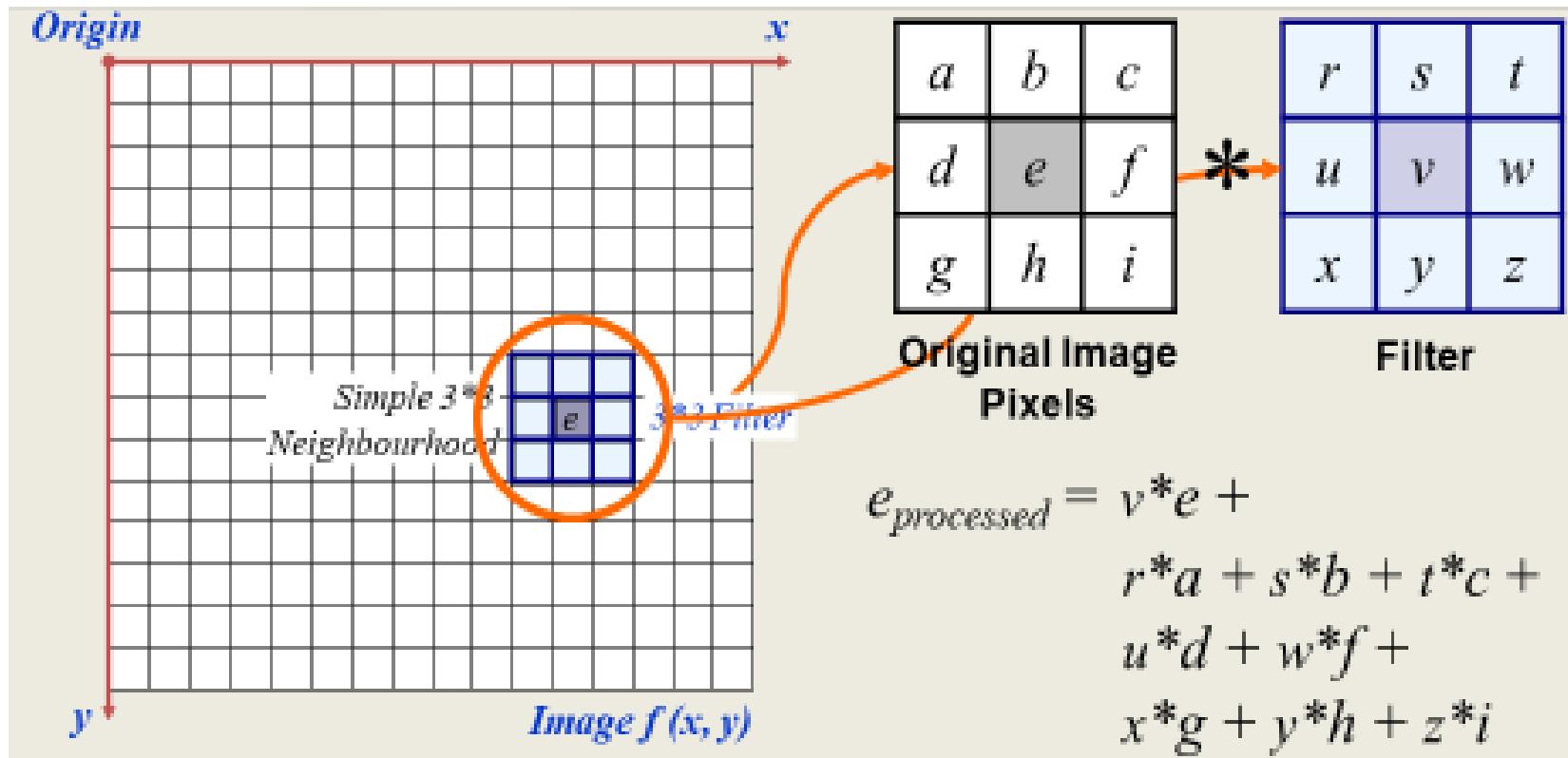
Simple Neighbourhood Operations

- Some simple neighborhood operations include:
 - **Min:** Set the pixel value to the minimum in the neighborhood
 - **Max:** Set the pixel value to the maximum in the neighborhood
 - **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

Simple Neighbourhood Operations Example



The Spatial Filtering Process



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

The Spatial Filtering Process

2) Spatial filtering process :-

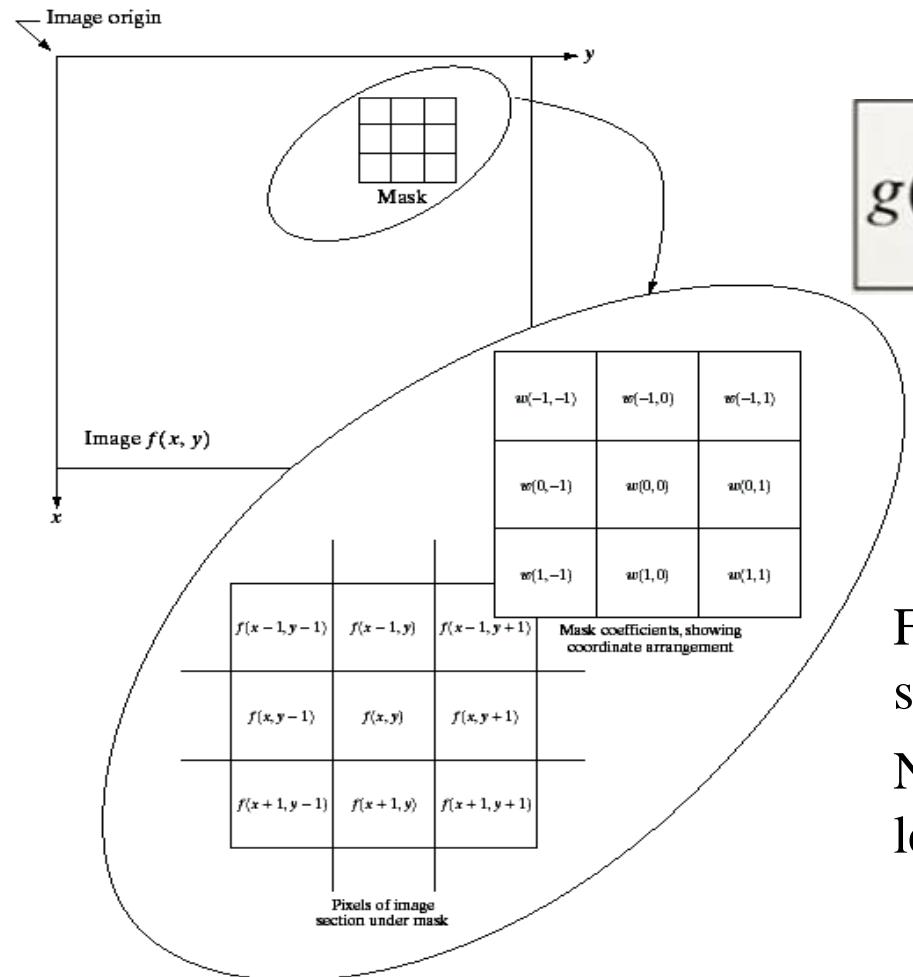
e_{process} = Selected pixel + filter
with neighbour

$$g(x,y) = T \cdot f(x,y)$$

↑ ↑ ↑
filtered mask Original
image image.

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

Spatial Filtering: Equation Form



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

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

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
- Also useful for highlighting gross detail

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Simple averaging filter
(Box filter)

Smoothing Spatial Filters

3) Smoothing w/ spatial filters:-

:- If we want to decrease the pixel value
(divide pixel)

1	1	1
1	1	1
1	1	1

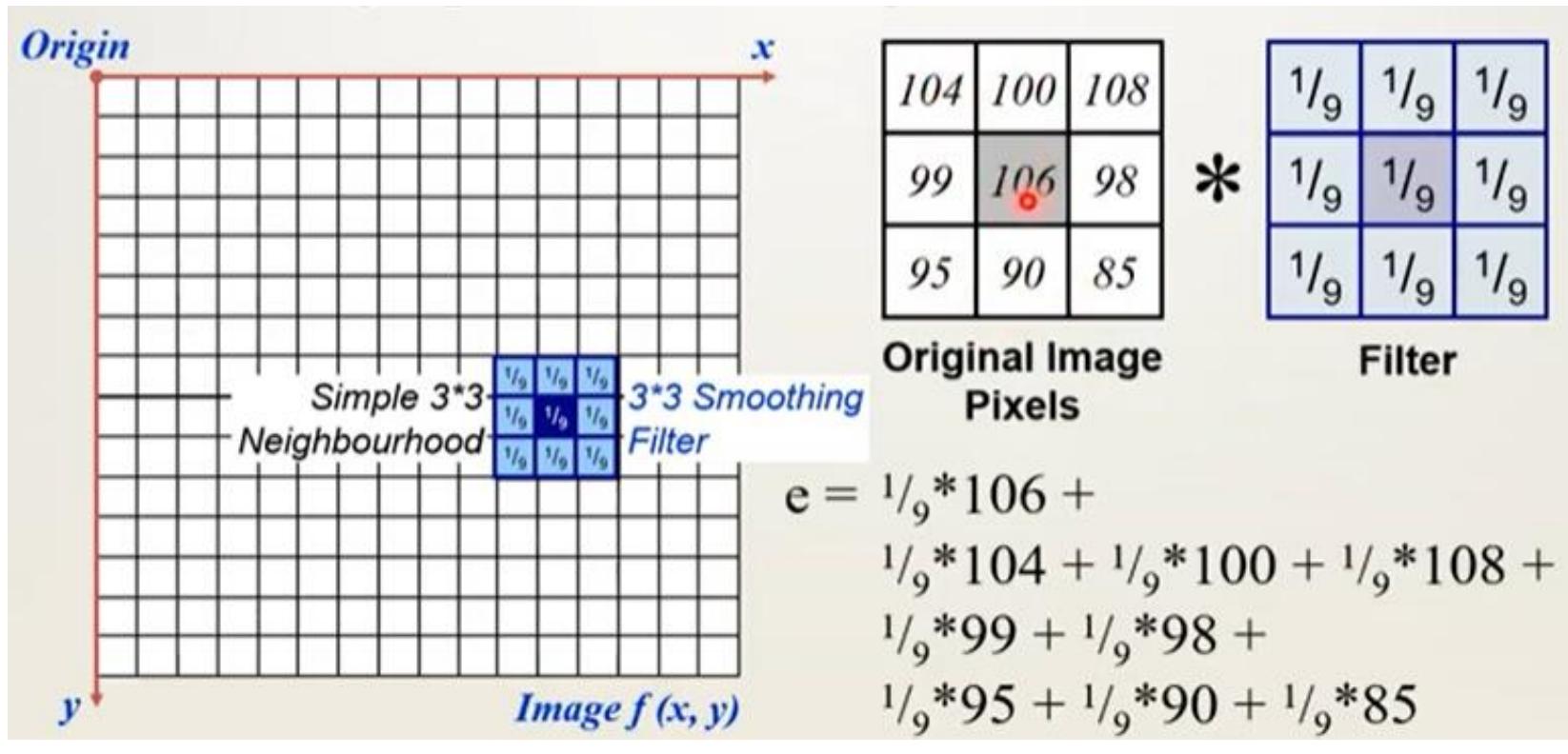
Apply
this filter

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

Original

Image

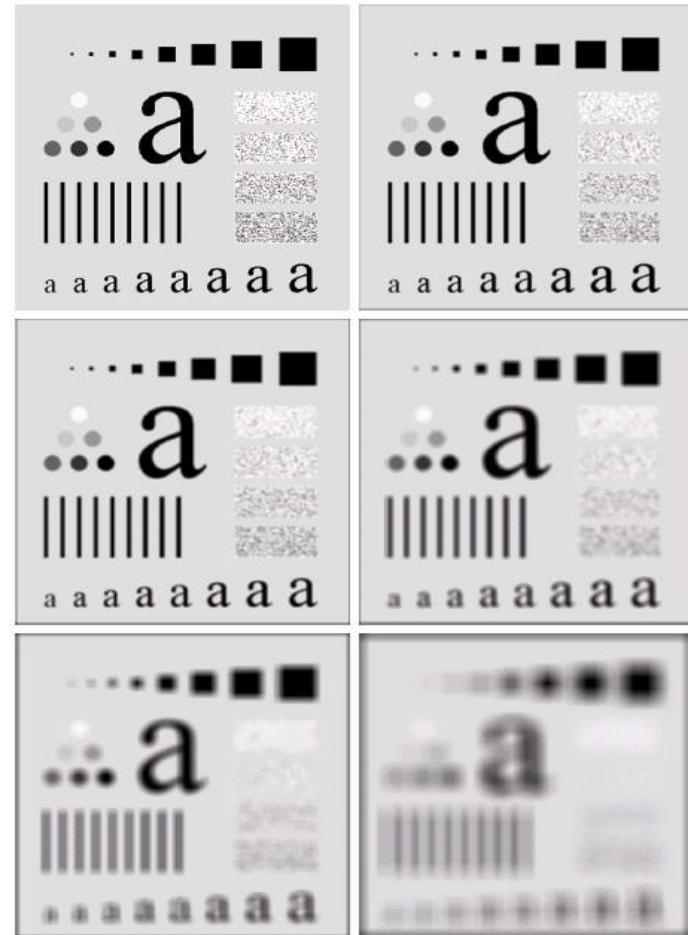
Smoothing Spatial Filtering



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



Filtering Correlation

Correlation

- The equation of correlation is defined as follows :

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

- It performs dot product between weights and function

1	-1	-1
1	2	-1
1	1	1

Correlation

1	-1	-1		
1	4	-2	2	3
1	2	1	3	3
2	2	1	2	
1	3	2	2	

Input Image, f

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

5			

output
Image, g

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Original

1	-1	-1
1	2	-1
1	1	1

Magic.

Step :- 1.

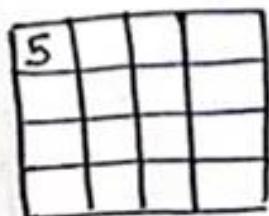
$$\boxed{2} \quad 2 \\ 2 \quad 1$$

$$\boxed{2} \quad -1 \\ 1 \quad 1$$

$$= (2)(2) + 2(-1) + 2(1) + 1(1)$$

$$= 4 - 2 + 2 + 1$$

$$= 5$$



1	-1	-1
1	2	-1
1	1	1

1	-1	-1	
2	4	-2	3
2	1	3	3
2	2	1	2
1	3	2	2

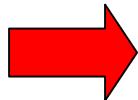
Input Image, f

Correlation

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

5	10		

output
Image, g



Step: - 2

$$\begin{array}{r} 2 \quad \boxed{2} \quad 2 \\ 2 \quad 1 \quad 3 \end{array}$$

$$\begin{array}{r} 1 \quad \boxed{2} - 1 \\ 1 \quad 1 \quad 1 \end{array} \rightarrow$$

$$= 2(1) + 2(2) + 2(-1) + 2(1) + 1(1) + 3(1)$$

$$= 2 + 4 - 2 + 2 + 1 + 3$$

$$= 10$$

5	10	

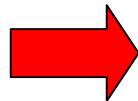
1	-1	-1
1	2	-1
1	1	1

Correlation

1	-1	-1	
2	2	4	-3
2	1	3	3
2	2	1	2
1	3	2	2

Input Image, f

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



5	10	10	

output
Image, g

Step:- 3

$$\begin{array}{rrr} 2 & \boxed{2} & 3 \\ 1 & 3 & 3 \end{array}$$

$$\begin{array}{rrr} 1 & \boxed{2} & -1 \\ 1 & 1 & 1 \end{array}$$

$$= 2(1) + 2(2) + 3(-1) + 1(1) + 3(1) + 3(1)$$

$$= 2 + 4 - 3 + 1 + 3 + 3$$

$$= 10$$

5	10	10	

1	-1	-1
1	2	-1
1	1	1

Correlation

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

		1	-1	-1
2	2	2	6	-1
2	1	3	3	1
2	2	1	2	
1	3	2	2	

Input Image, f



5	10	10	14

output
Image, g

Step: - 4

$$\begin{array}{r} \boxed{2} / 3 \\ 3 \quad 3 \end{array} \quad \begin{array}{r} 2 \boxed{3} \\ 3 \quad 3 \end{array} \quad \begin{array}{r} 1 \boxed{2} \\ 1 \quad 1 \end{array}$$

$$= 2(1) + 3(2) + 3(1) + 3(1)$$

$$= 2 + 6 + 3 + 3$$

$$= 14$$

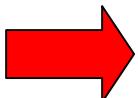
5	10	10	14

1	-1	-1
1	2	-1
1	1	1

Correlation

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	-2	-2	2	3
1	4	-1	3	3
1	2	2	1	2
1	3	2	2	



5	10	10	15
3			

output
Image, g

Input Image, f

Step:- 5

$$\begin{matrix} 2 & 2 \\ \boxed{2} & 1 \\ 2 & 2 \end{matrix}$$

$$\begin{matrix} -1 & -1 \\ \boxed{2} & -1 \\ 1 & 1 \end{matrix}$$

$$= 2(-1) + 2(-1) + 2(2) + 1(-1) + 2(1) + 2(1)$$

$$= -2 - 2 + 4 - 1 + 2 + 2$$

$$= 3$$

Q3

5	10	15	14
3			

1	-1	-1
1	2	-1
1	1	1

Correlation

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	-2	-2	3
2	2	-3	3
2	2	1	2
1	3	2	2



5	10	10	15
3	4		

output
Image, g

Input Image, f

Last step

$$\begin{matrix} 2 & 2 & 2 \\ 2 & \boxed{1} & 3 \\ 2 & 2 & 1 \end{matrix}$$

$$\begin{matrix} 1 & -1 & -1 \\ 1 & \boxed{2} & 1 \\ 1 & 1 & 1 \end{matrix}$$

$$= 2(1) + 2(-1) + 2(-1) + 2(1) + 1(2) + 3(-1) + 2(1) + 2(1) + 1(1)$$

$$= 2 - 2 - 2 + 2 + 2 - 3 + 2 + 2 + 1$$

$$= 4$$

6	10	10	14
3	4		

As per this sequence,
we found final correlation OIP for
given image.

Correlation

5	10	10	15
3	4	6	11
7	11	4	9
-5	4	4	5

Final output Image, g

Filtering Convolution

Correlation & Convolution

- The filtering we have been talking about so far is referred to as *correlation* with the filter itself referred to as the *correlation kernel*
- Convolution* is a similar operation, with just one subtle difference
- For symmetric filters it makes no difference

a	b	c
d	e	e
f	g	h

*

r	s	t
u	v	w
x	y	z

Original Image
Pixels

Filter

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

Convolution

- The equation of correlation is defined as follows :

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

$$g = \omega * f$$

- It performs cross product between weights and function

Convolution

Convolution kernel, ω

1	-1	-1
1	2	-1
1	1	1

Rotate 180°

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Convolution

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	1	1		
-1	4	2	2	3
-1	-2	1	3	3
2	2	1	2	
1	3	2	2	



5			

Output
Image, g

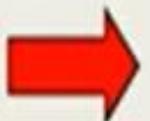
Input Image, f

1	1	1
-1	2	1
-1	-1	1

Convolution

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	1	1	
-2	4	2	3
-2	-1	3	3
2	2	1	2
1	3	2	2



5	4		

Output
Image, g

Input Image, f

Convolution

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2



Input Image, f

5	4	4	-2
9	6		

Output
Image, g

Convolution

5	4	4	-2
9	6	14	5
11	7	6	5
9	12	8	5

Final output Image, g

Thank You!!!

Unit 3

Feature Detection

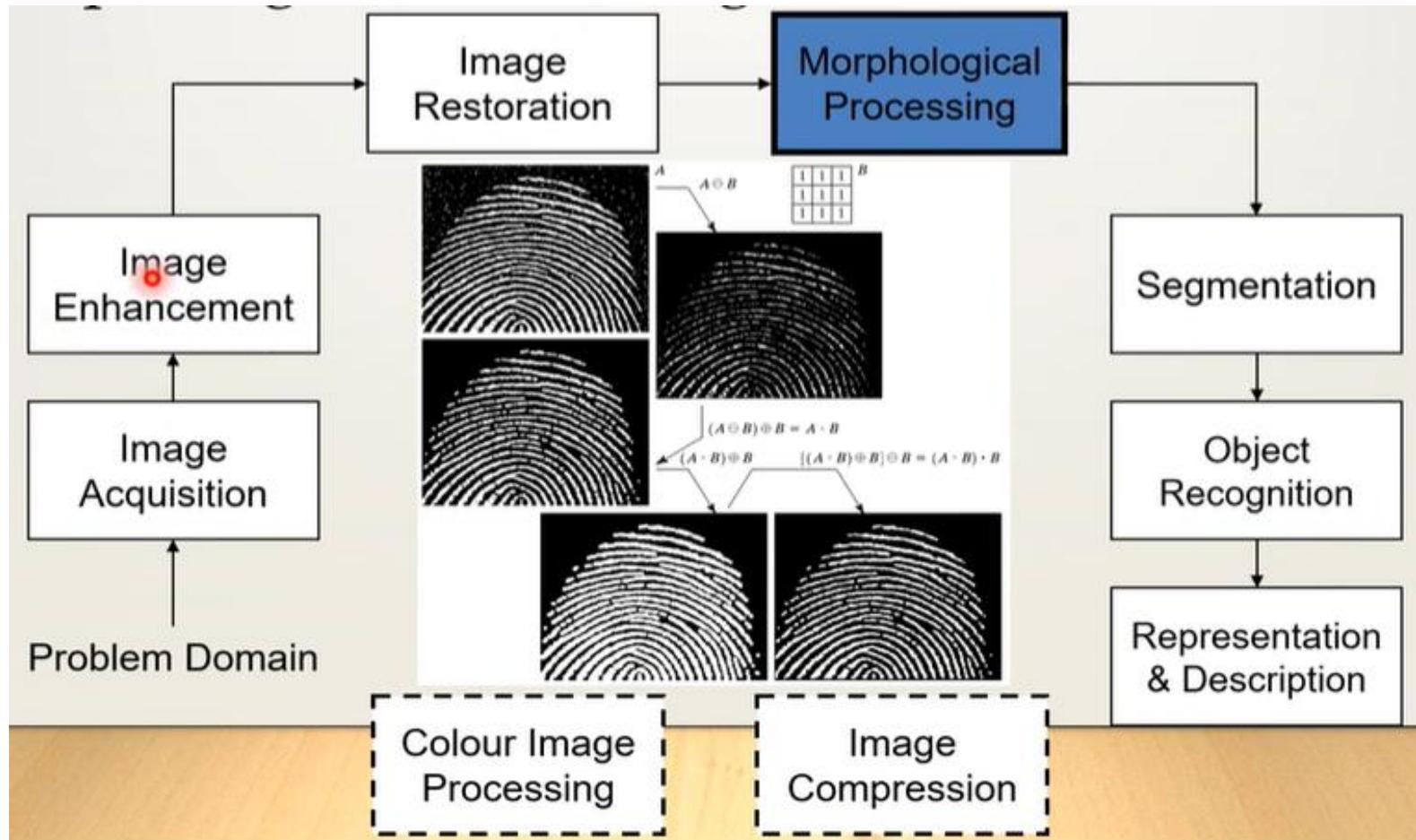
Prof. Janki Patel
Department of IT
SPCE

Content

- Edge Detection
- Corner Detection
- Line and Curve Detection
- Active Contours
- SIFT and HOG Descriptors
- Shape Context Descriptors
- Morphological Operations

Morphological Operation

Key Stages in Digital Image Processing: Morphological Processing



1, 0, Black, White?

- Throughout all the following slides whether 0 and 1 refer to white or black is a little interchangeable.
- All of the discussion that follows assumes segmentation has already taken place and that images are made up to 0s for background pixels and 1s for object pixels
- After this it doesn't matter if 0 is black, white, yellow, green....

What is Morphology?

- Morphological image processing(or morphology) describes a range of image processing techniques that deal with the shape(or morphology) of features in an image
- Morphological operations are typically applied to remove imperfections introduced during segmentation and so typically operate on bi-level images

Quick Example

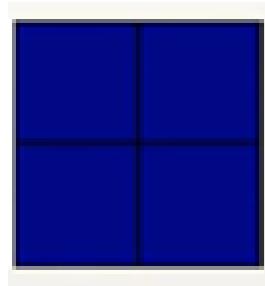
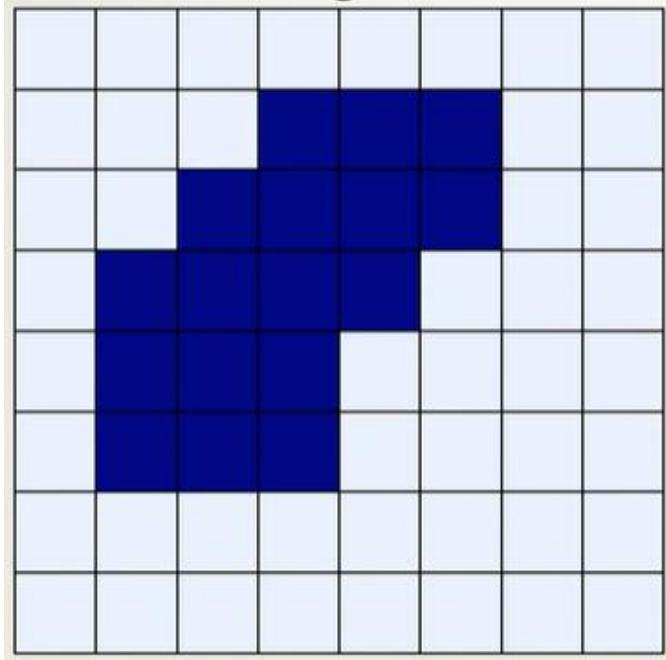


Image after segmentation



Image after segmentation and
morphological processing

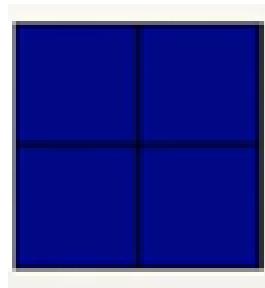
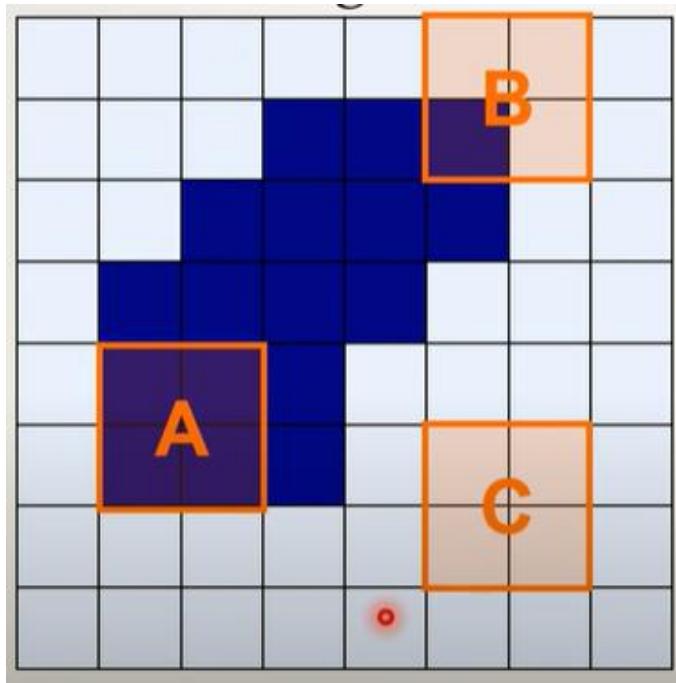
Structuring Elements , Hits & Fits



Structuring Element

- Fit: All on pixels in the structuring element cover on pixels in the image
- Hit: Any on pixel in the structuring element covers an on pixel in the images
- All morphological processing are based on these simple ideas

Structuring Elements , Hits & Fits



Structuring Element

- Fit: All on pixels in the structuring element cover on pixels in the image
- Hit: Any on pixel in the structuring element covers an on pixel in the images
- All morphological processing are based on these simple ideas

Structuring Elements

- Structuring elements can be of any size and make any shape
- However, for simplicity we will use rectangular structuring elements with their origin at the middle pixel

1	1	1
1	1	1
1	1	1

0	1	0
1.	1	1
0	1	0

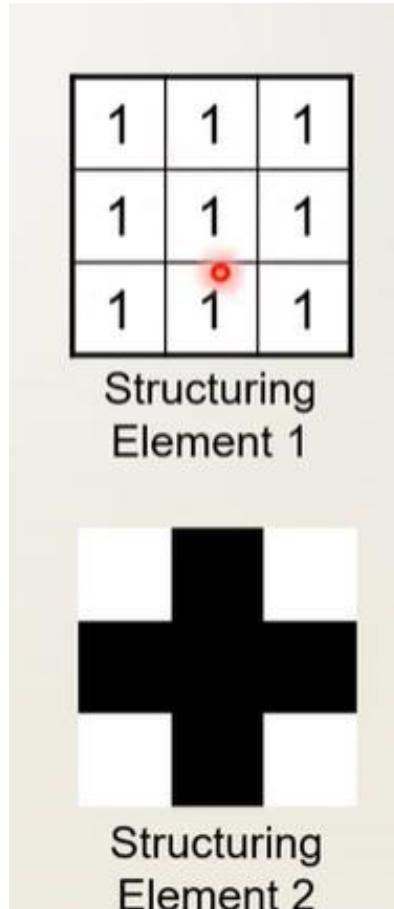
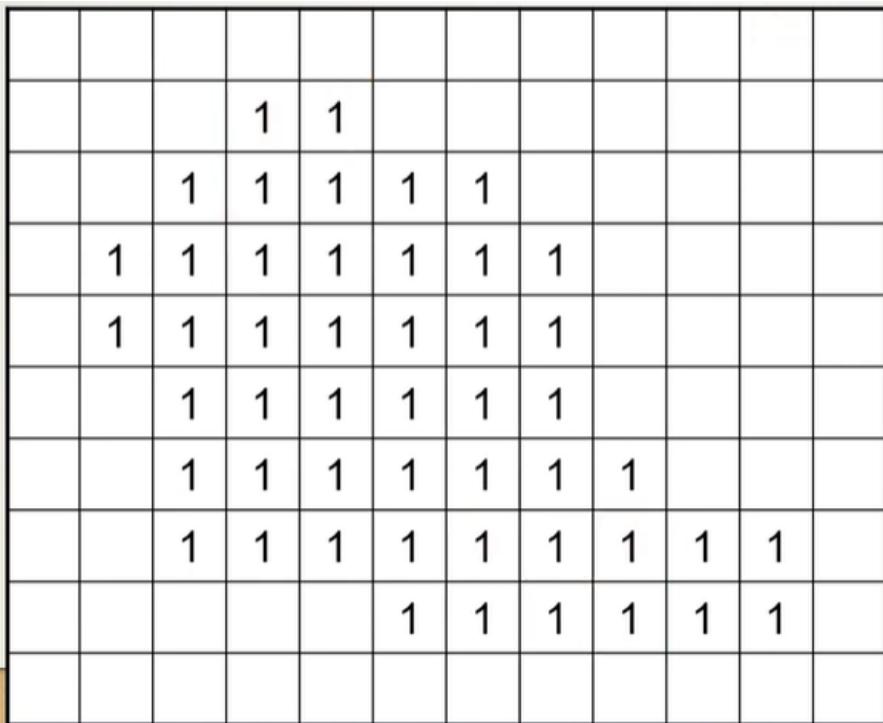
0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

What is use of Hit, Miss and Fit?

- Hit and miss algorithm can be used to thin and skeletonize a shape in a binary image.



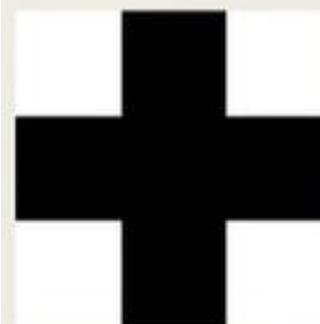
Fitting & Hitting



Fitting & Hitting

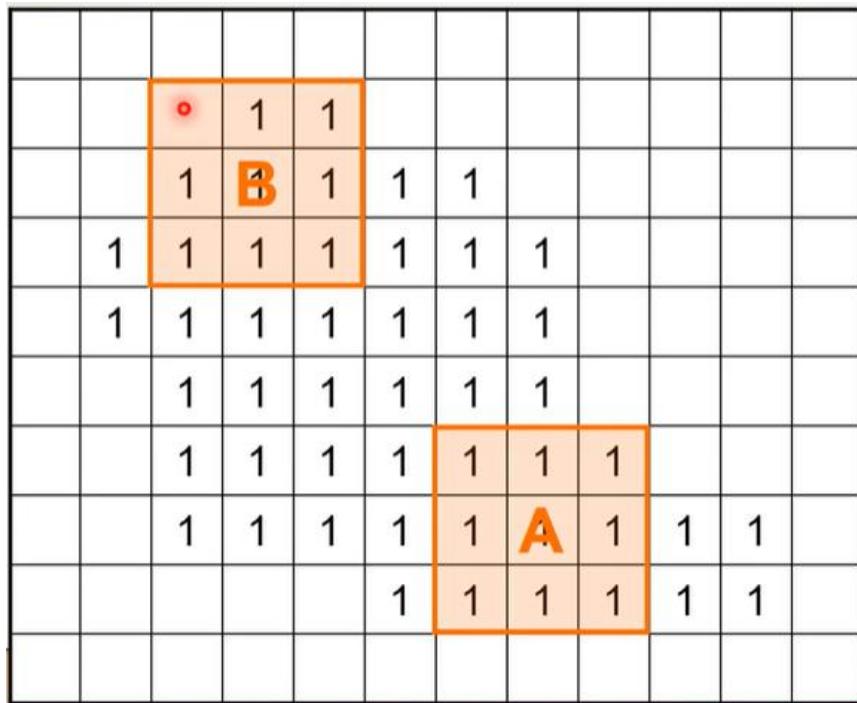
1	1	1
1	1	1
1	1	1

Structuring Element 1



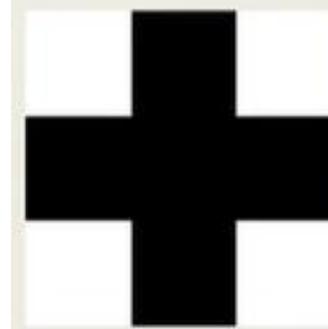
Structuring Element 2

Fitting & Hitting



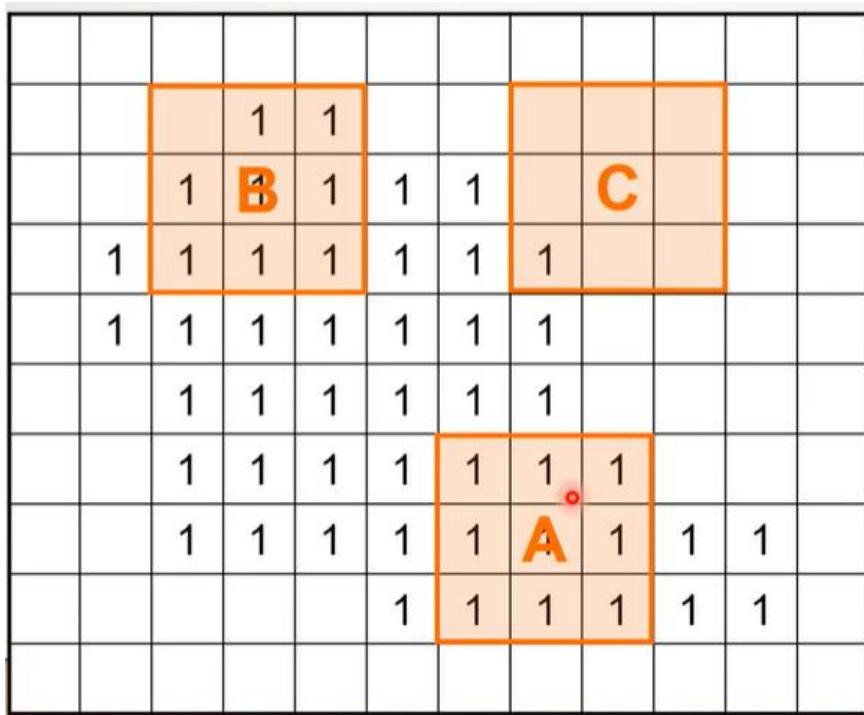
1	1	1
1	1	1
1	1	1

Structuring Element 1



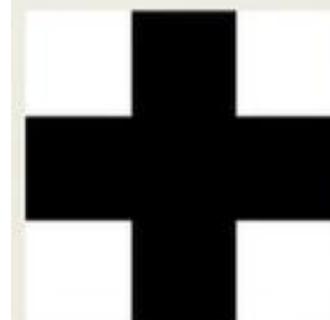
Structuring
Element 2

Fitting & Hitting



1	1	1
1	1	1
1	1	1

Structuring
Element 1



Structuring
Element 2

Fundamental Operations

- Fundamentally morphological image processing is very much like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends in the operation performed
- There are two basic morphological operations: **erosion** and **dilation**

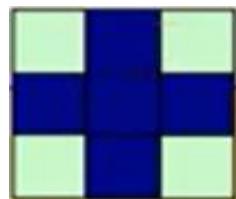
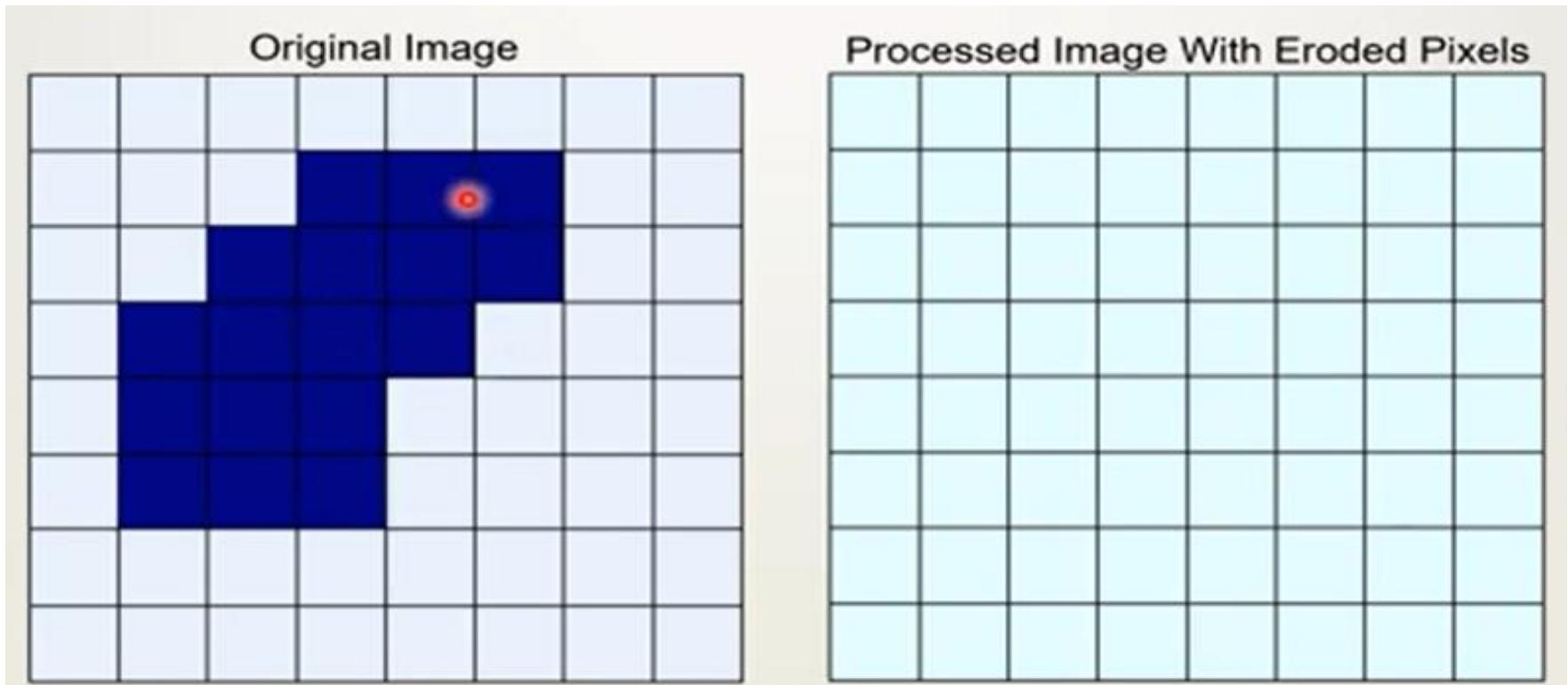
Erosion

Erosion

- Erosion of image f by structuring element s is by $f \Theta s$
- The structuring element s is positioned with its origin at (x,y) and the new pixel value is determined using the rule:

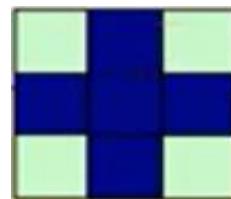
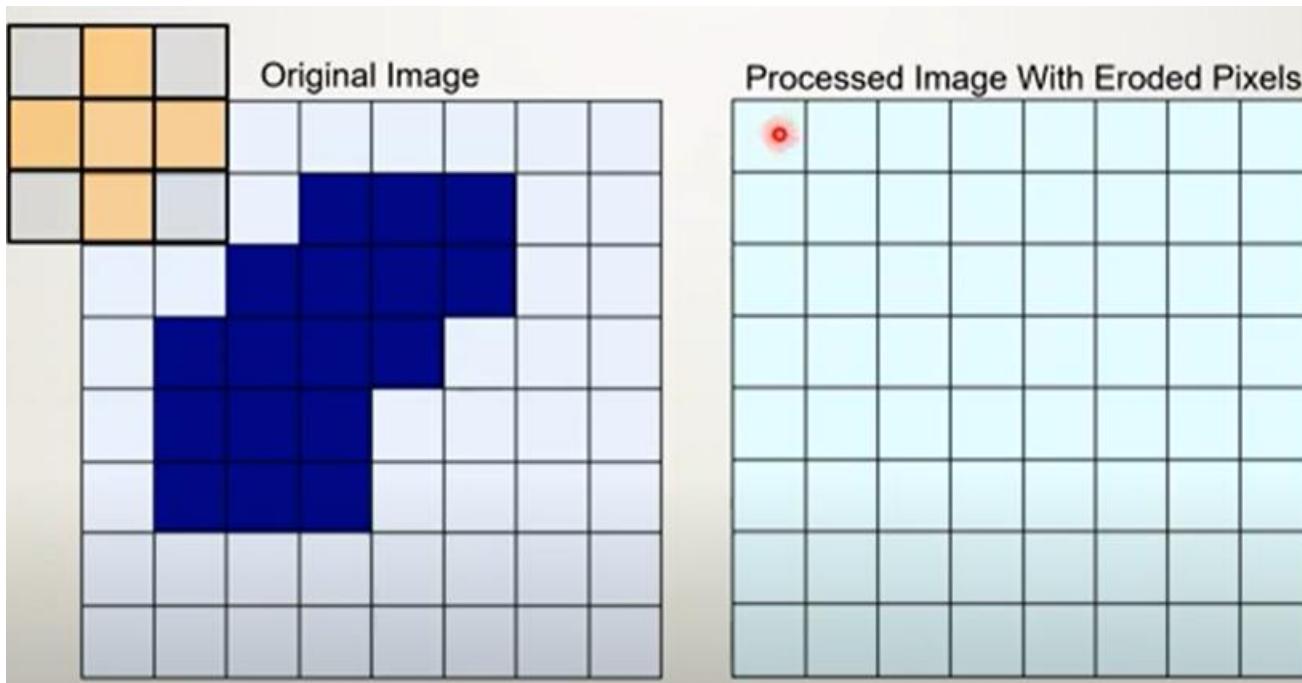
$$g(x, y) = \begin{cases} 1 & \text{if } s \text{ fits } f \\ 0 & \text{otherwise} \end{cases}$$

Erosion Example



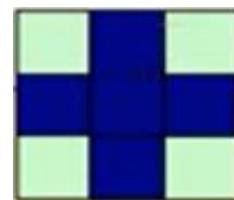
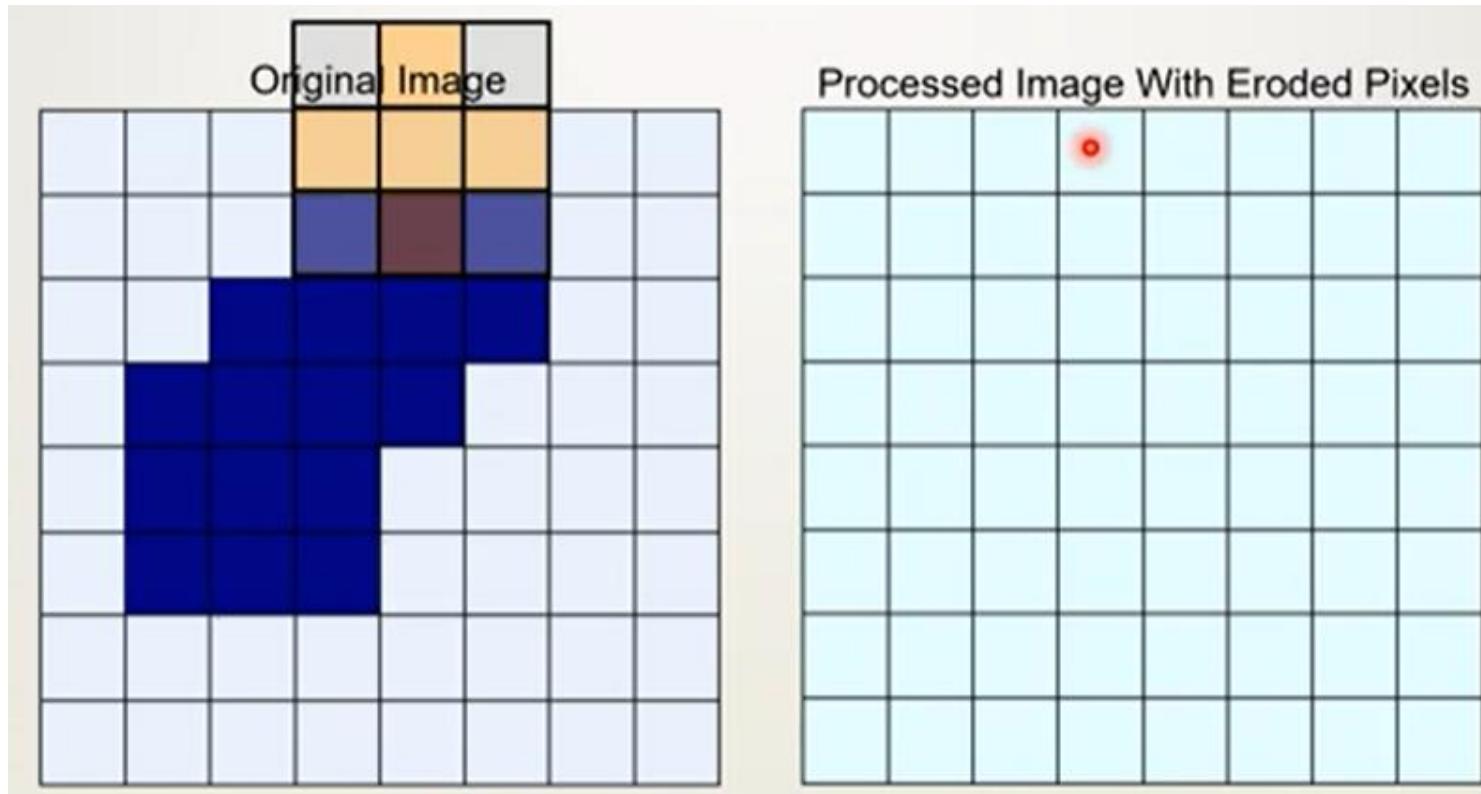
Structuring Element

Erosion Example



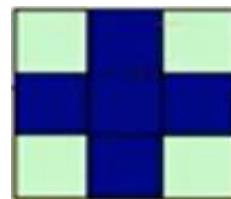
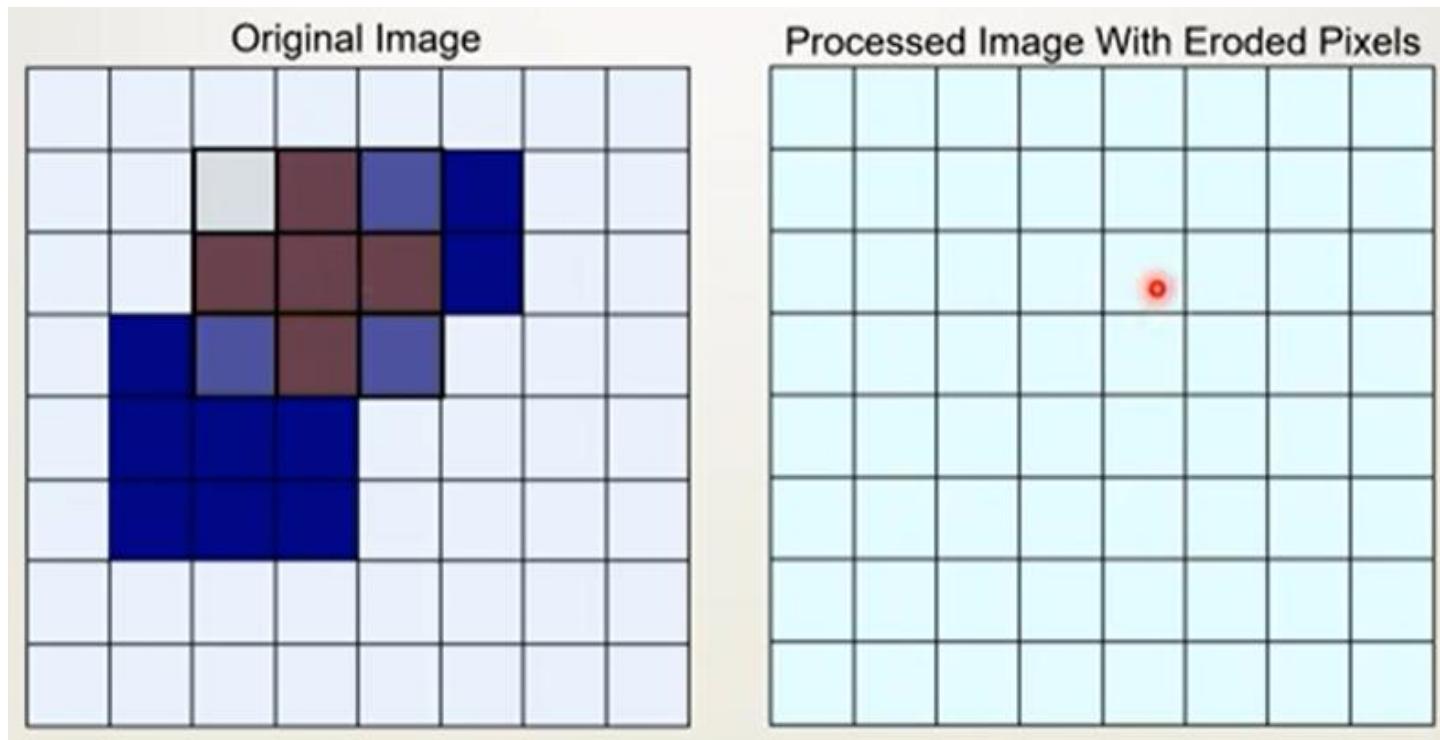
Structuring Element

Erosion Example



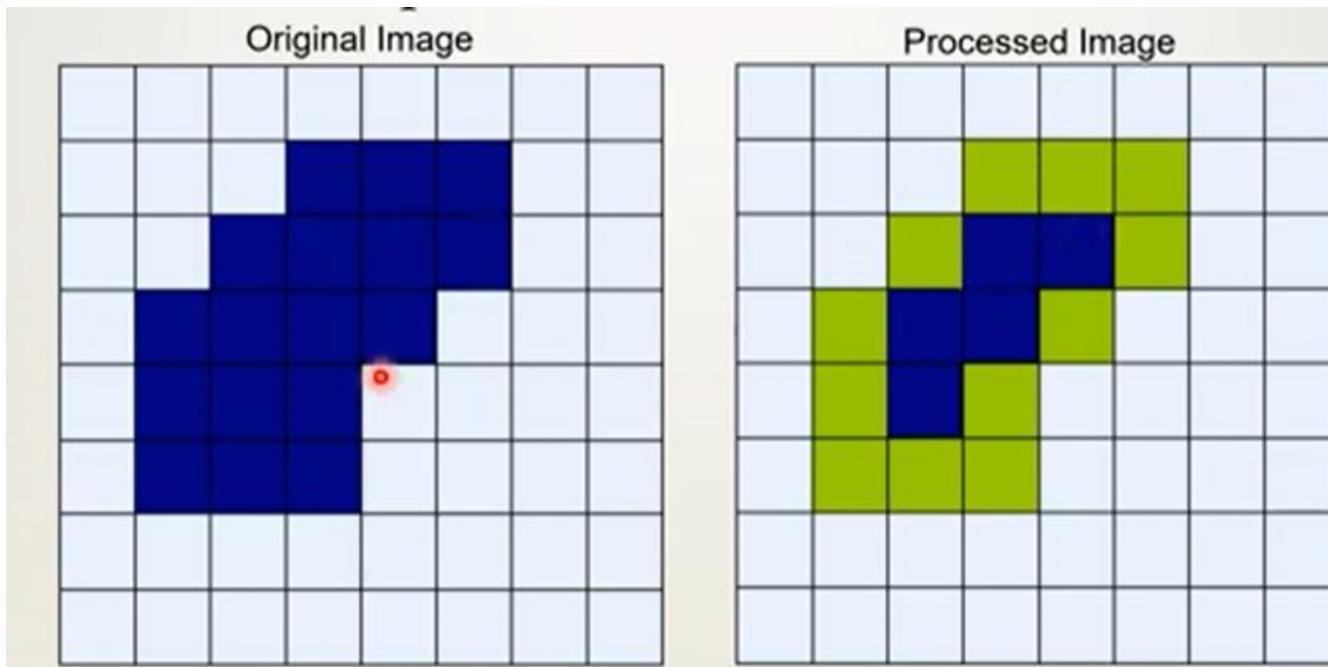
Structuring Element

Erosion Example

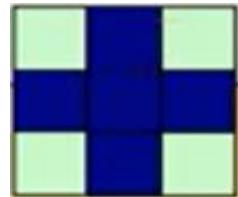


Structuring Element

Erosion Example



0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	1	0	0	0
0	0	0	0	0



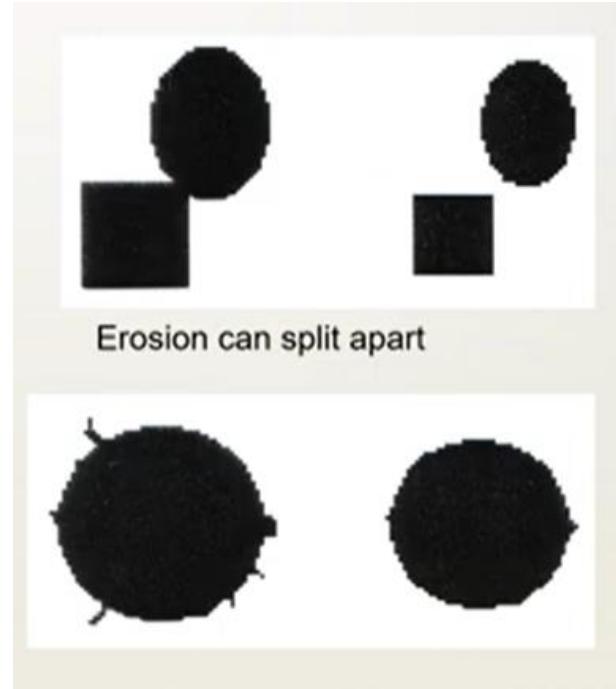
Structuring Element

Erosion Example 1



What is Erosion For?

- Erosion can split apart joined Objects

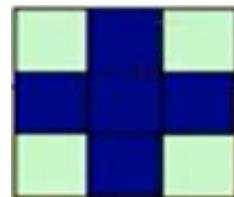


- Erosion can strip away extrusions

- Watch out: Erosion shrinks objects

Apply Erosion on image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



Structuring Element

Eroded Image

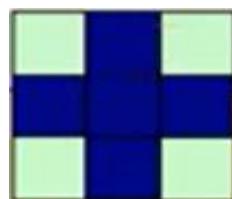
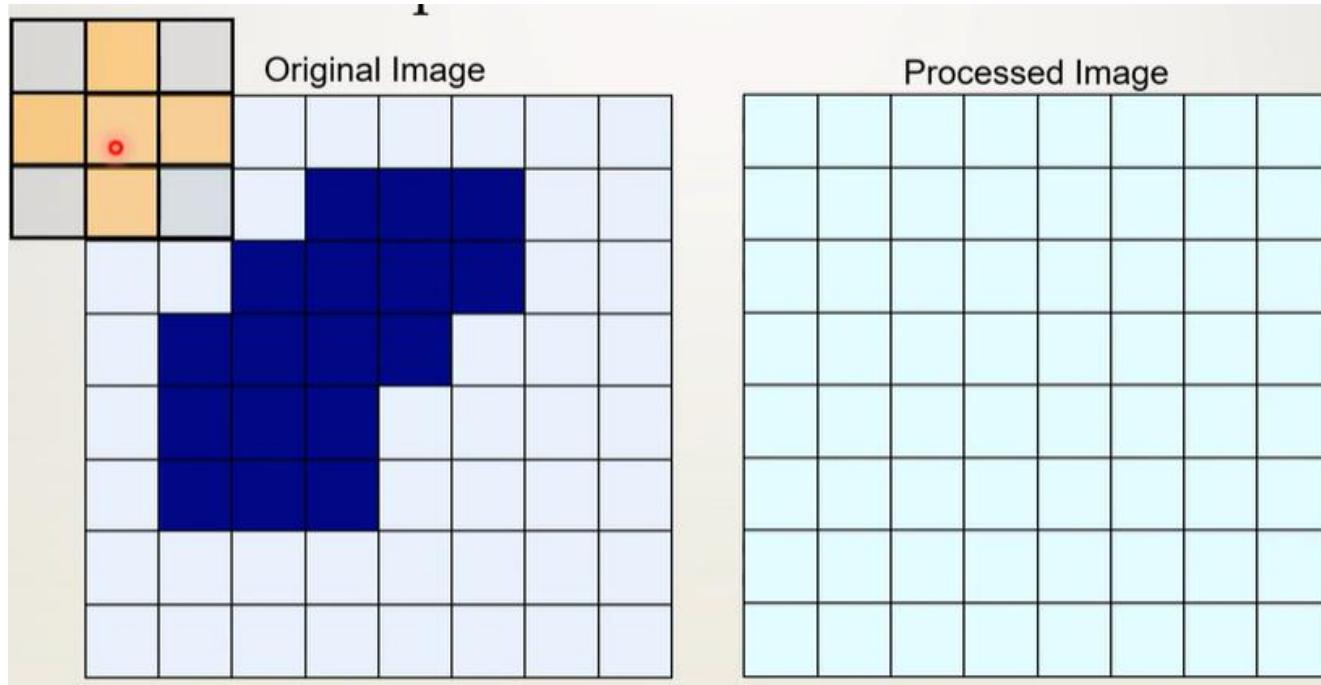
Dilation

Dilation

- Dilation of image f by structuring element s is by $f \oplus s$
- The structuring element s is positioned with its origin at (x,y) and the new pixel value is determined using the rule:

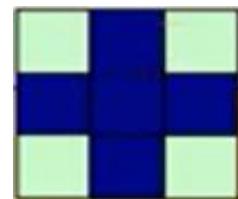
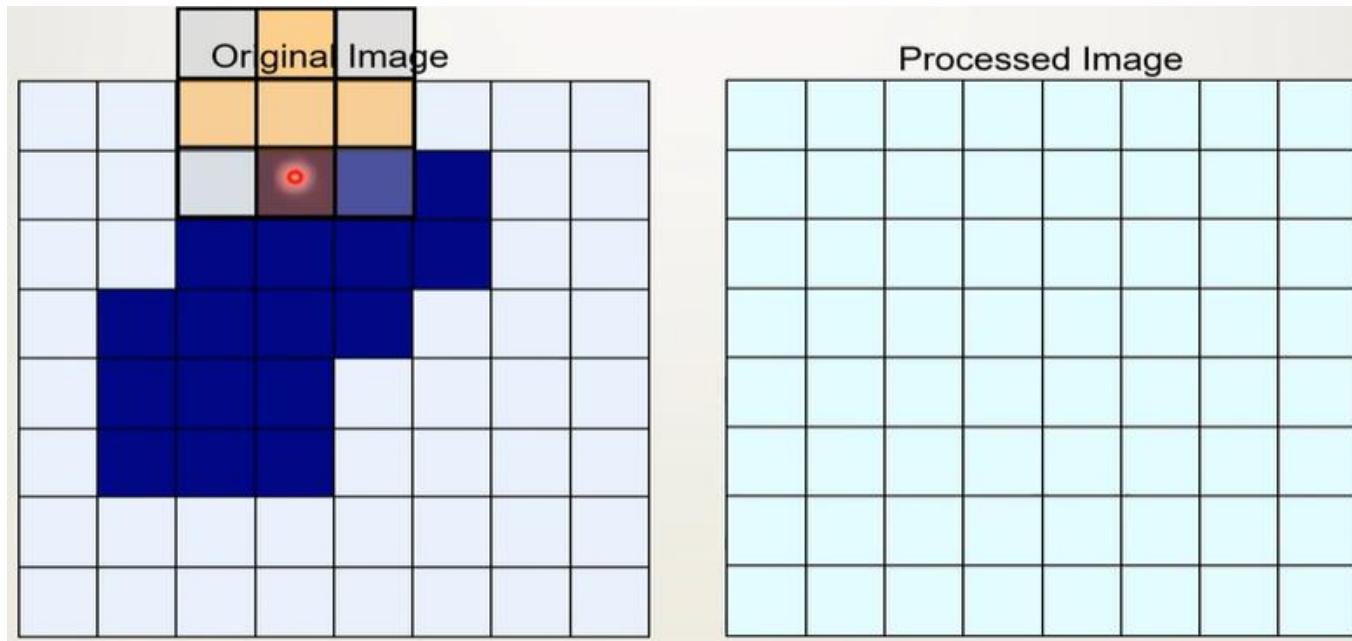
$$g(x, y) = \begin{cases} 1 & \text{if } s \text{ hits } f \\ 0 & \text{otherwise} \end{cases}$$

Dilation Example



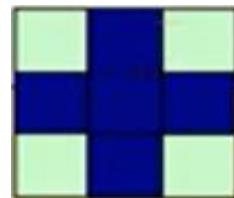
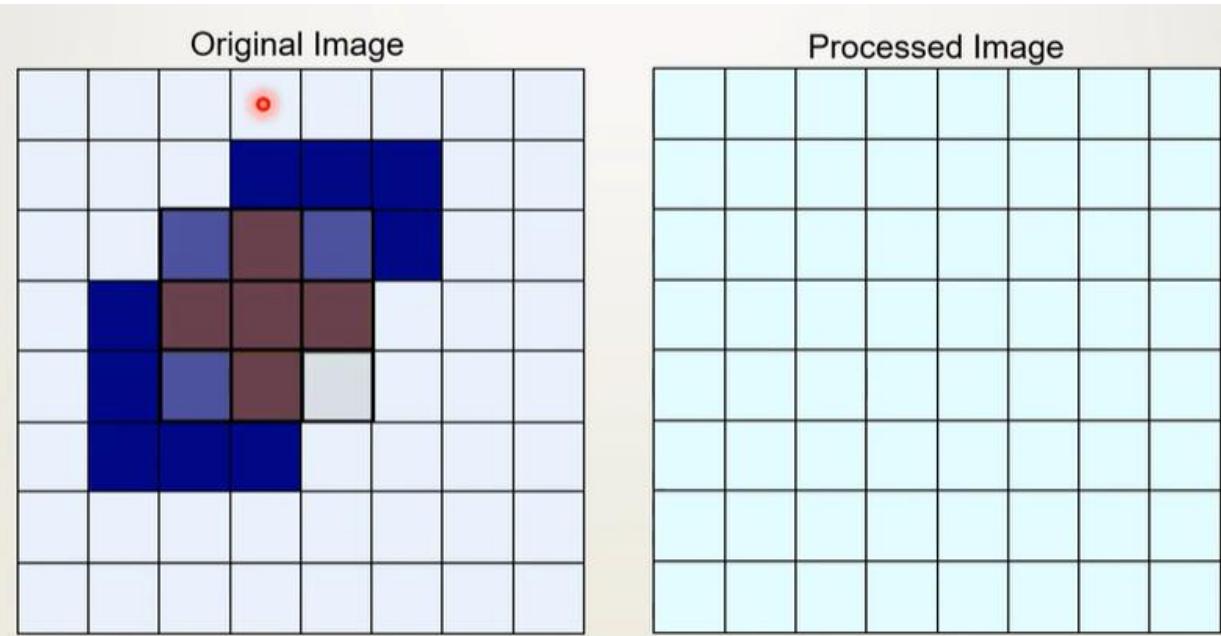
Structuring Element

Dilation Example



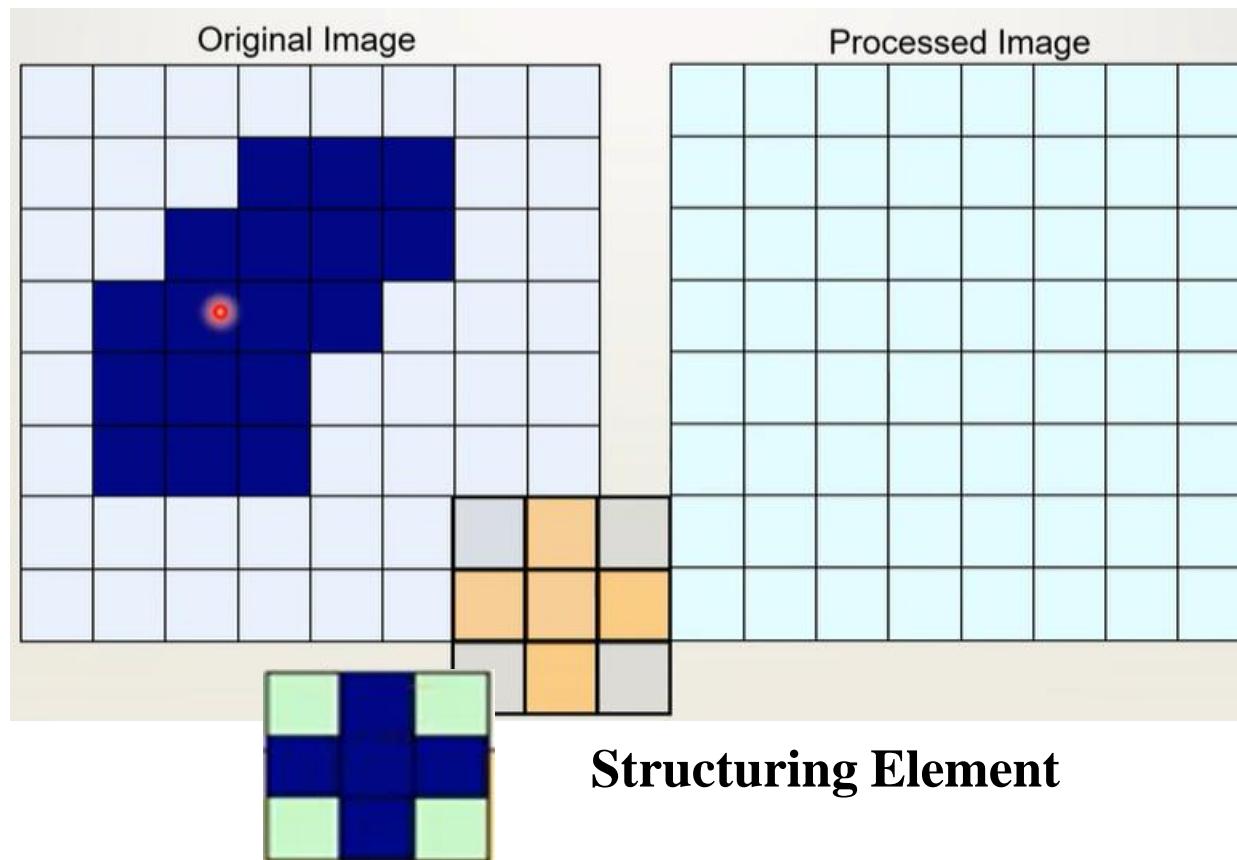
Structuring Element

Dilation Example

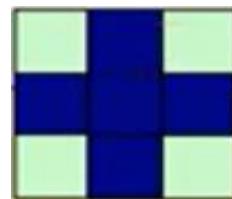
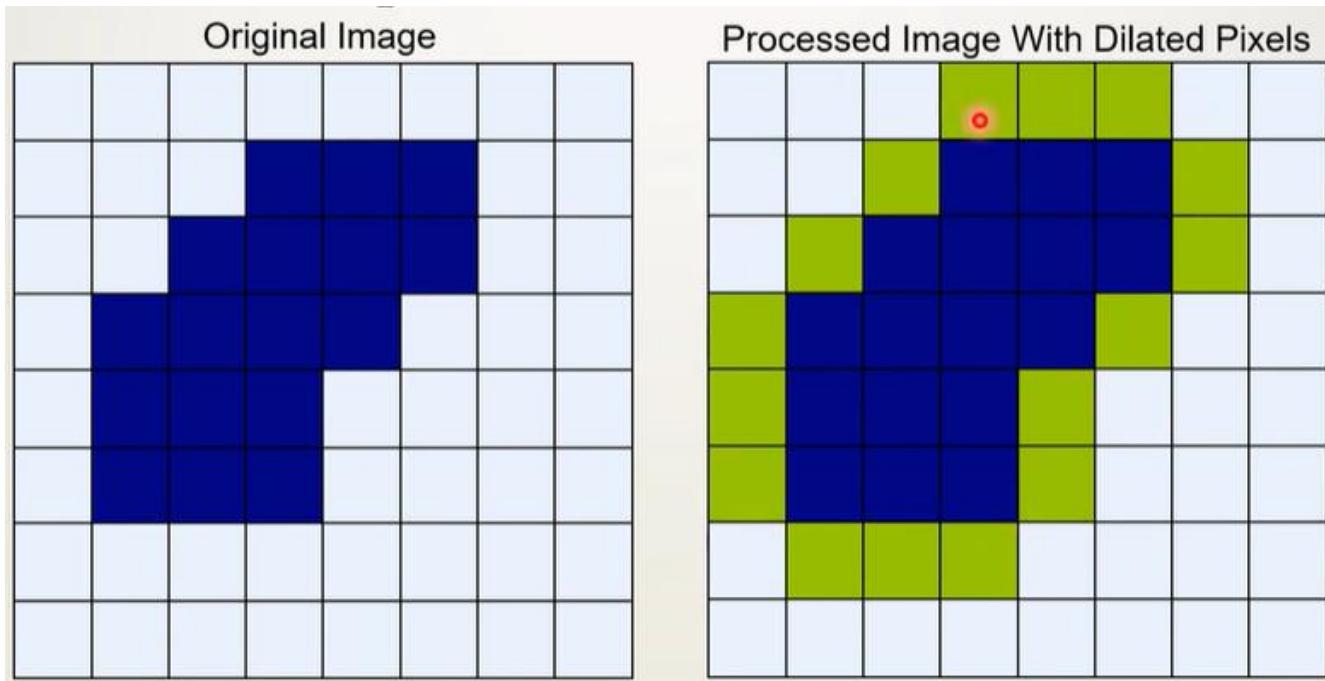


Structuring Element

Dilation Example



Dilation Example



Structuring Element

Dilation Example 1



Original image



Dilation by 3*3
square structuring
element

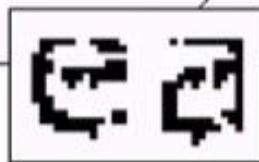


Dilation by 5*5
square structuring
element

Dilation Example 2

Original image

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



After dilation

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

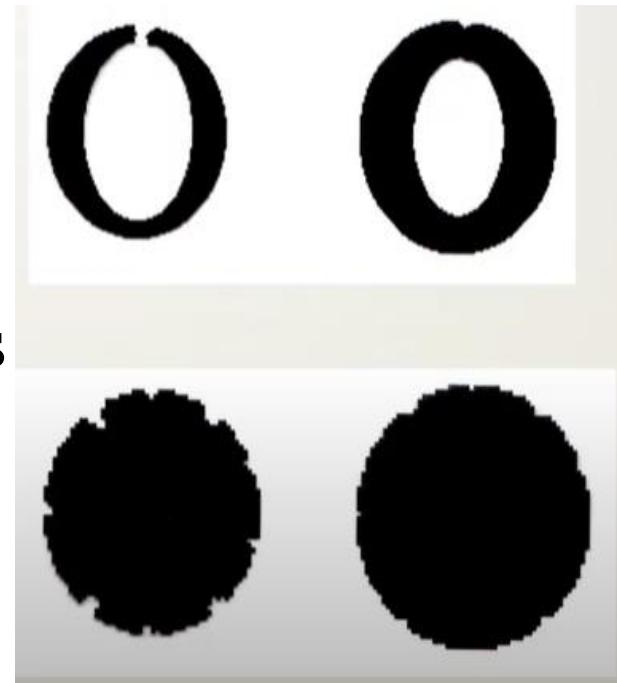


0	1	0
1	1	1
0	1	0

Structuring Element

What is Dilation For?

- Dilation can repair breaks or bridges the gap
- Dilation can repair intrusions
- Watch out: Dilation enlarges objects



Dilation

- Advantage of dilation over low pass filter used to bridge the gap is that
- Dilation results directly in binary mage
- Low pass filter start with a binary image and produce a gray scale image, which would require a pass with a threholding function to convert it back to binary form

Compound Operations

Compound Operations

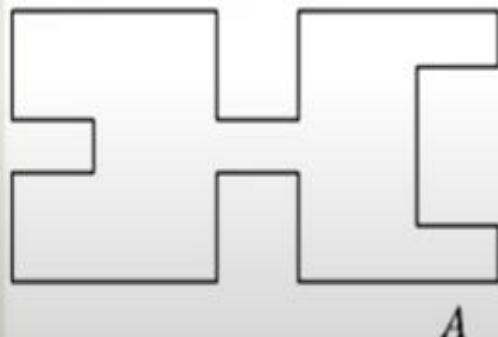
- More interesting morphological operations can be performed by performing combinations of erosion and dilations
- The most widely used of these compound operations are:
 1. Opening
 2. Closing

Opening

Opening

- The opening of image f by structuring element s , denoted $f \circ s$ is simply an erosion followed by a dilation

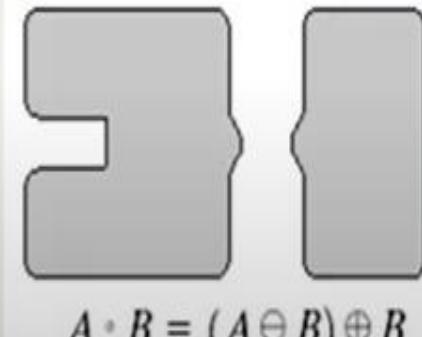
$$f \circ s = (f \ominus s) \oplus s$$



Original shape
 A



After erosion
 $A \ominus B$



After dilation
(opening)
 $A \circ B = (A \ominus B) \oplus B$

Opening Example

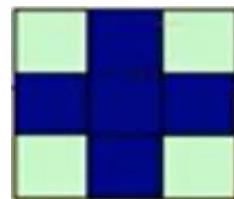
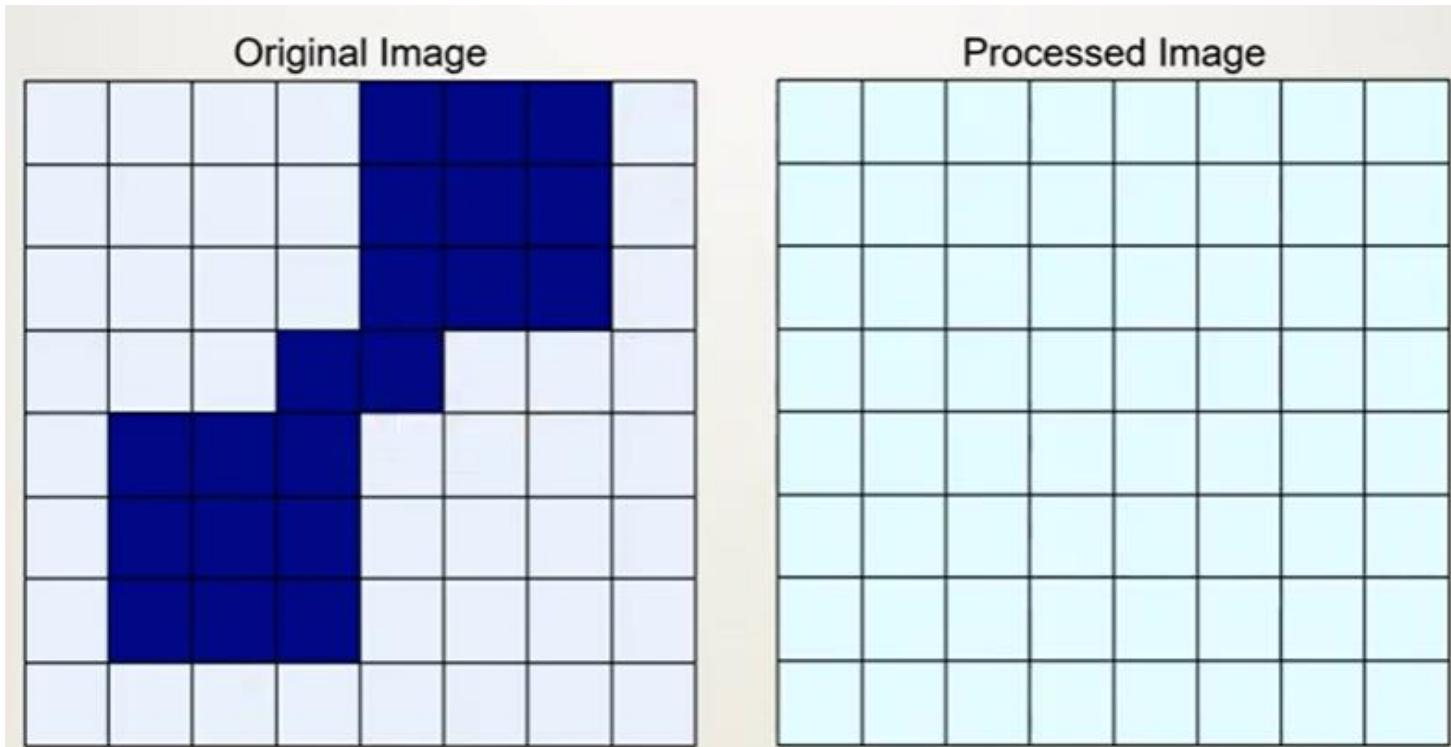
Original Image



Image After Opening

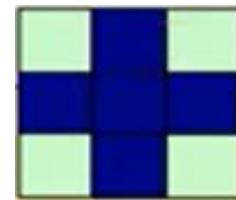
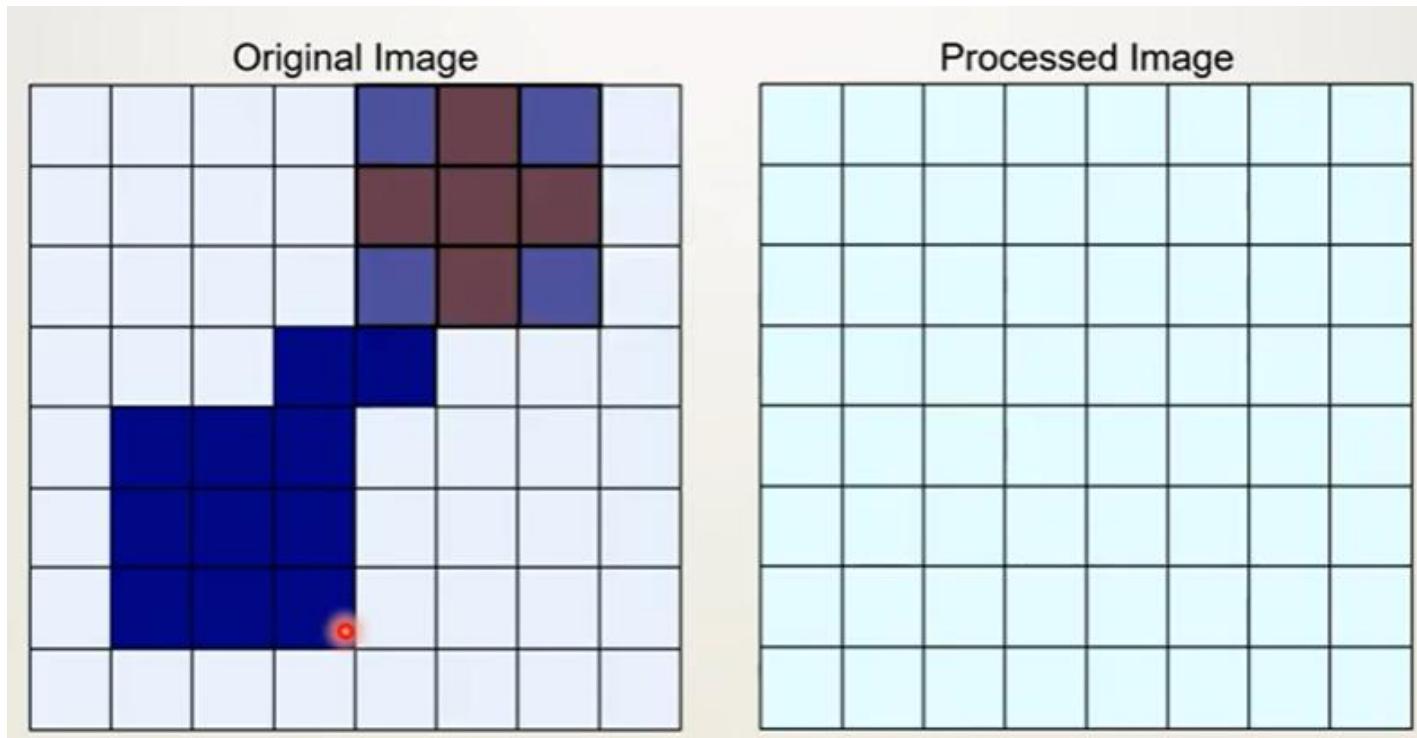


Opening Example



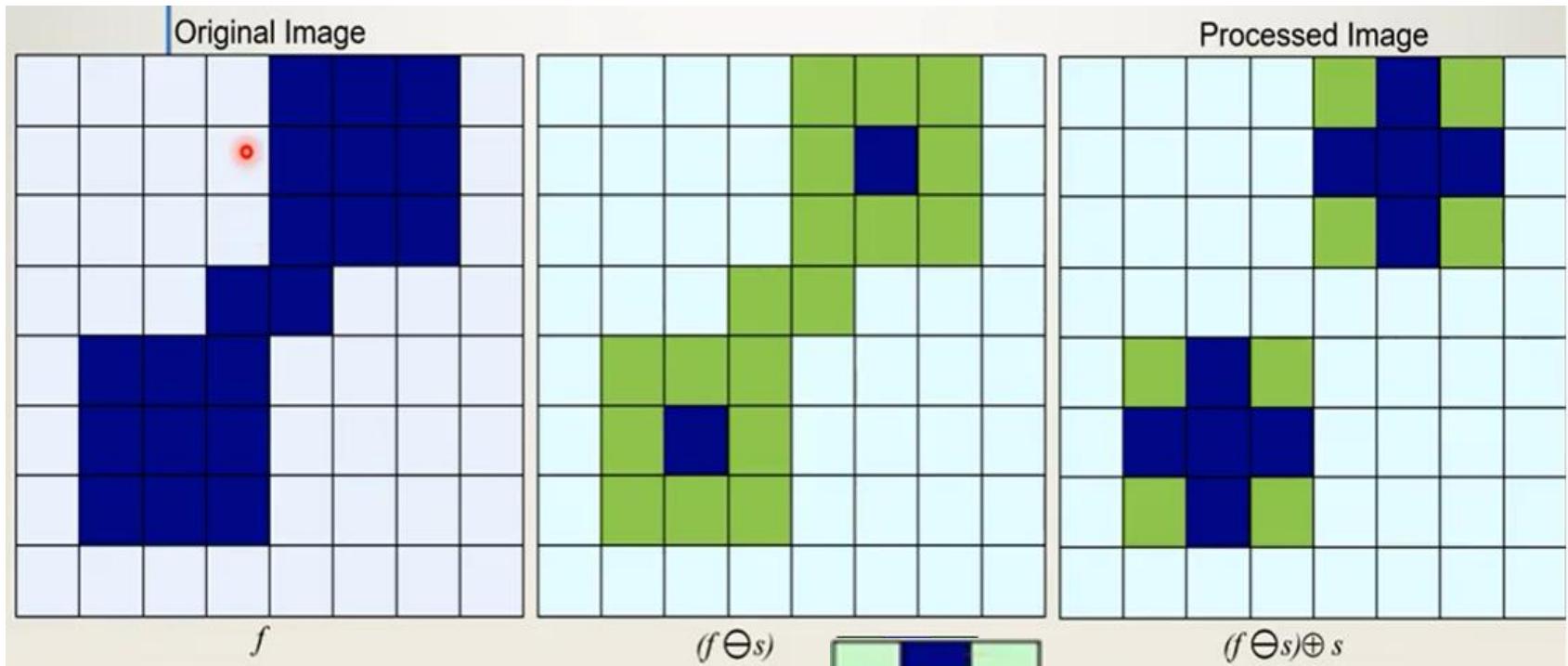
Structuring Element

Opening Example

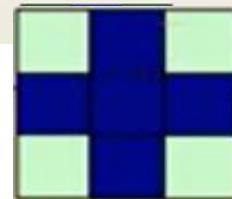


Structuring Element

Opening Example



Structuring Element



Advantages of opening

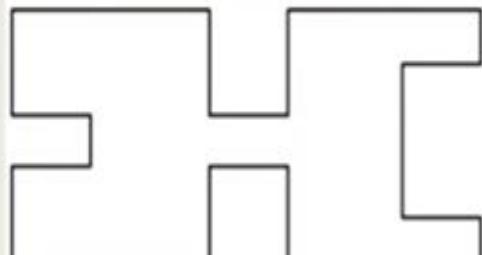
- Opening generally smoothing the contour of an object
- Breaks narrow isthmuses
- And eliminates thin protrusion

Closing

Closing

- The closing of image f by structuring element s , denoted $f \bullet s$ is simply a dilation followed by an erosion

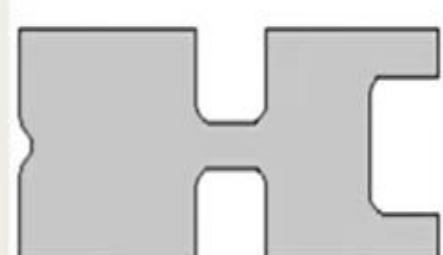
$$f \bullet s = (f \ominus s) \oplus s$$



Original shape
 A



After dilation
 $A \oplus B$



After erosion
(closing)
 $A \bullet B = (A \oplus B) \ominus B$

Closing Example

Original Image

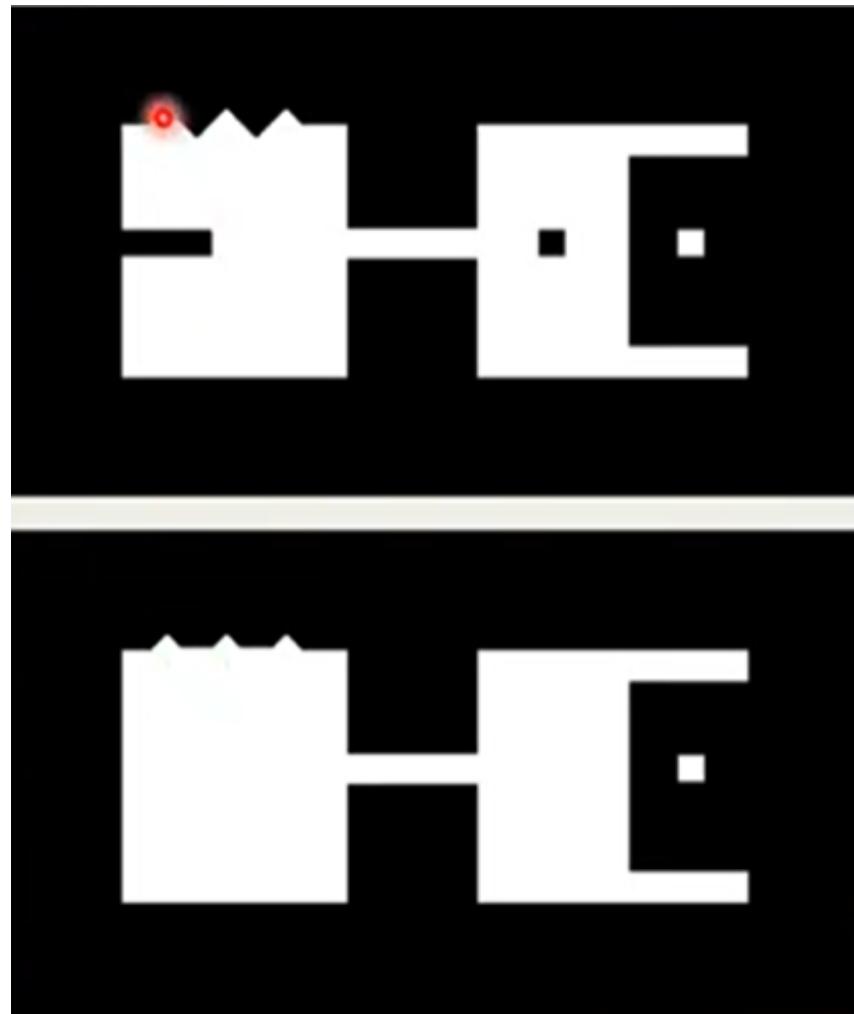
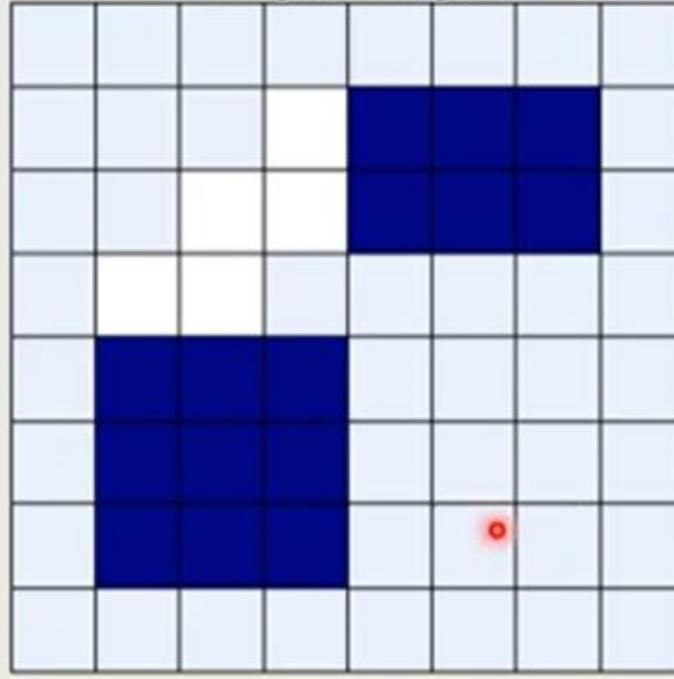


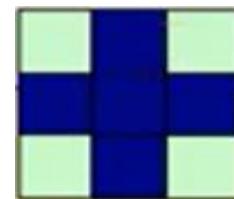
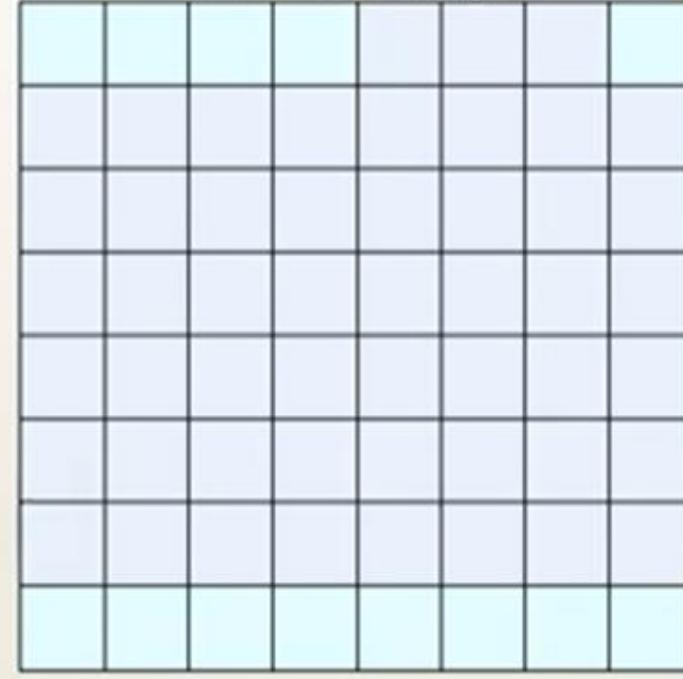
Image After Closing

Closing Example

Original Image

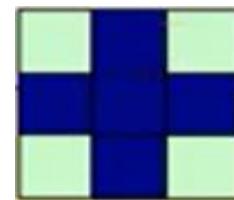
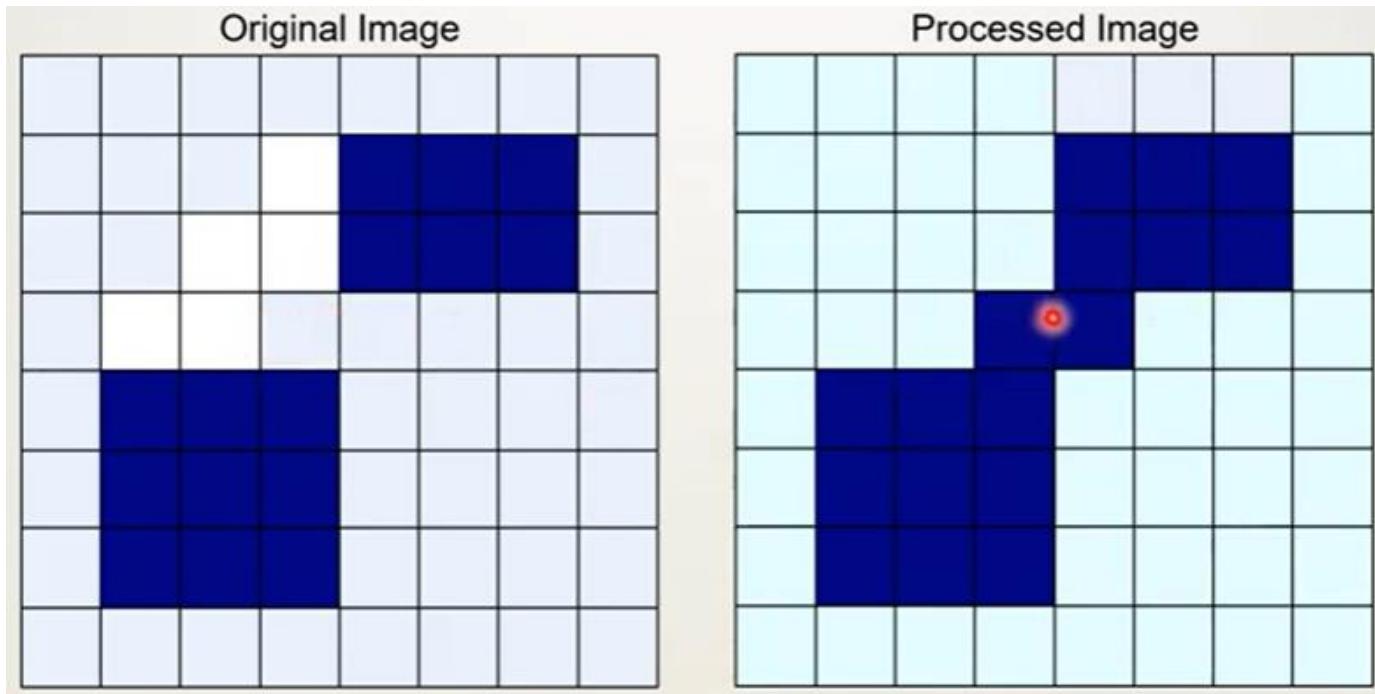


Processed Image



Structuring Element

Closing Example



Structuring Element

Advantages of closing

- Closing also tends to smooth sections of contours
- It generally fuses narrow breaks and long thin gulfs
- It eliminates small holes and fills gaps in the contour

Morphological Processing Example



Image Segmentation (Thresholding)

Thresholding

- Thresholding is usually the first in any segmentation approach
- We have talked about simple single value thresholding already
- Thresholding is used to produce region of uniformity within the given image based on some threshold criteria T .

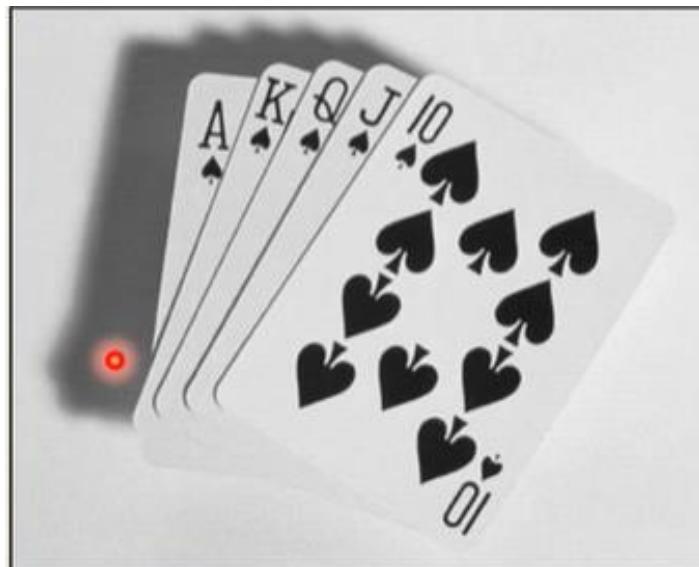
Thresholding

- Single value thresholding can be given mathematically as follows:

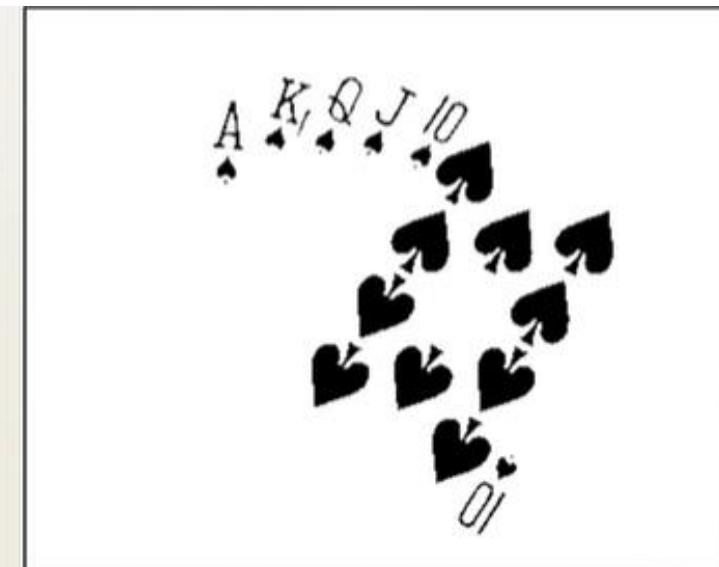
$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

Thresholding Example

- Imagine a poker playing robot that needs to visually interpret the cards in its hand.



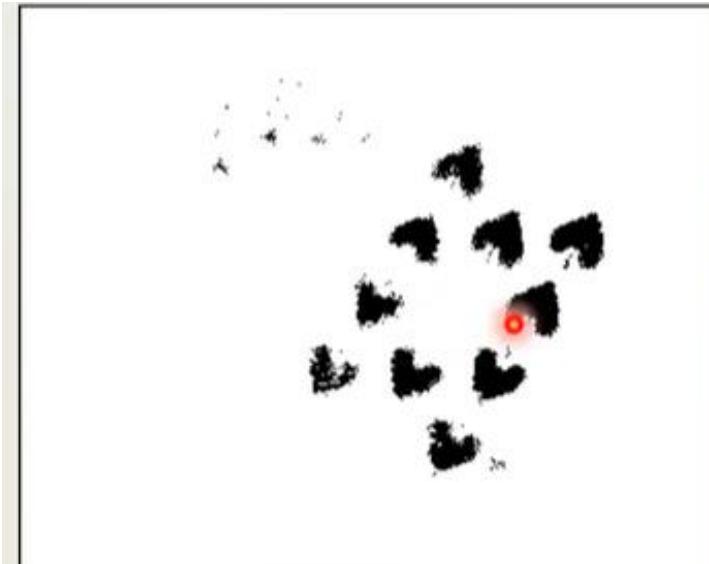
Original Image



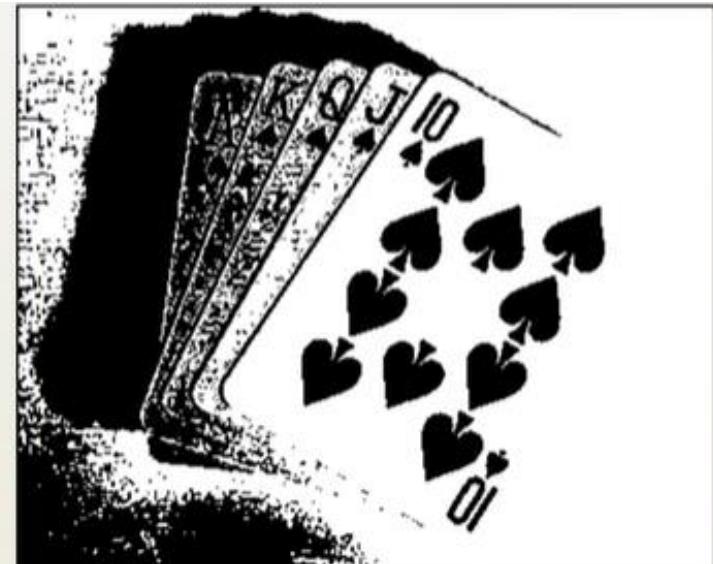
Thresholded Image

But be Careful

- If you get the threshold wrong , the results can be disastrous



Threshold Too Low



Threshold Too High

Basic Global Thresholding

- Based on the histogram of an image
- Partition the image histogram using a single global threshold
- The success of this technique very strongly depends on how well the histogram can be partitioned
- Types of thresholding:
 1. Global thresholding
 2. Local thresholding

Basic Global Thresholding Algorithm

- The basic global threshold , T , is calculated as follows
 1. Select an initial estimate for T (typically the grey level in the image)
 2. Segment the image using T to produce two group of pixels
 1. G_1 consisting of pixels with grey levels $> T$
 2. G_2 consisting of pixels with grey levels $\leq T$
 3. Compute the average grey levels of pixels in G_1 to give μ_1 and G_2 to give μ_2

Basic Global Thresholding Algorithm

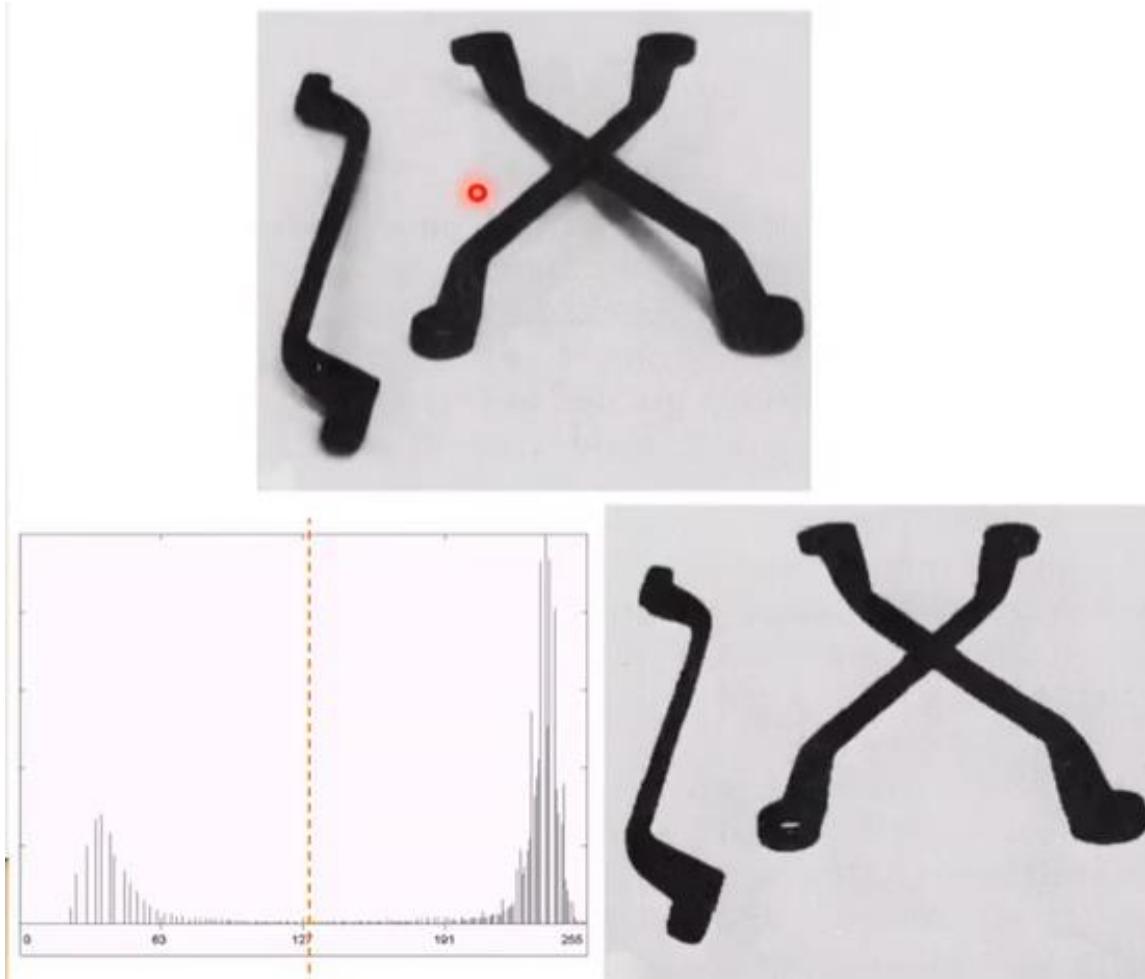
4. Compute a new threshold value:

$$T = \frac{\mu_1 + \mu_2}{2}$$

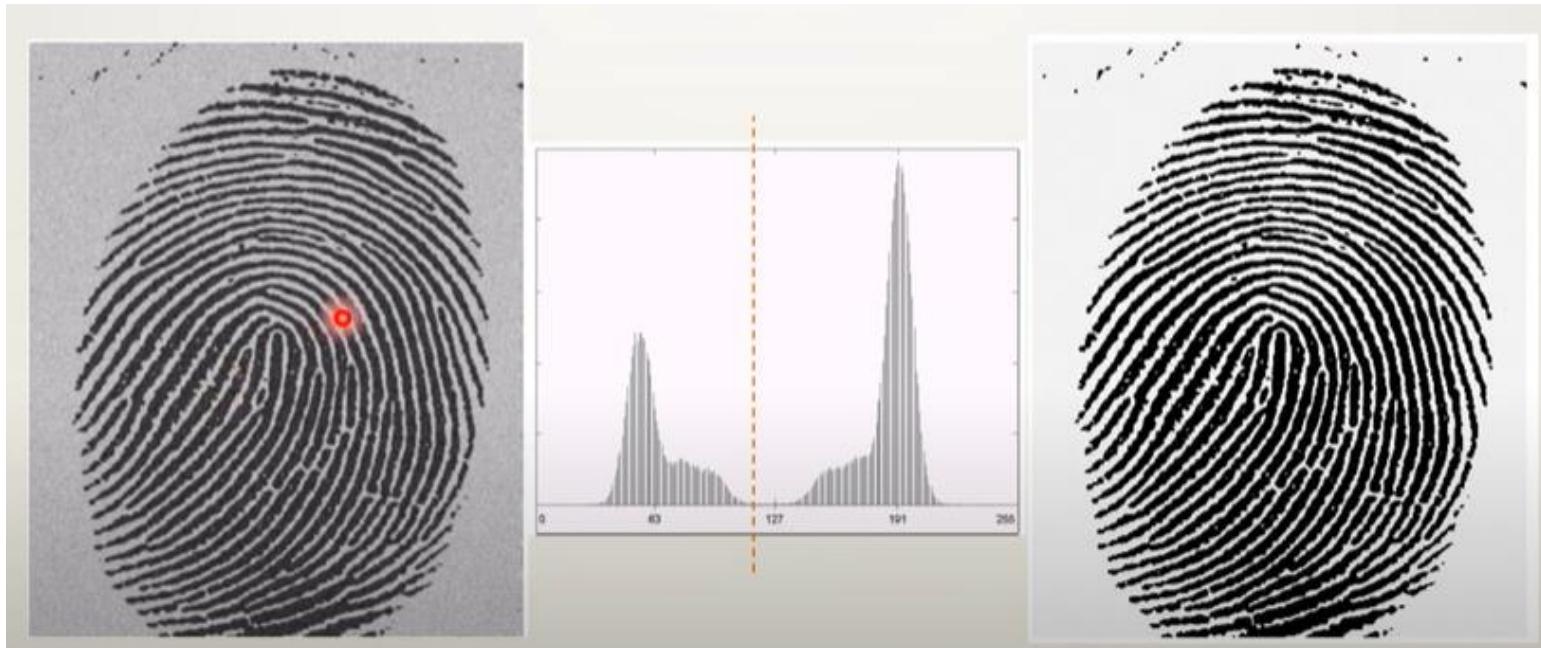
5. Repeat steps 2-4 until the difference in T in successive iterations is less than a predefined limit $T_{\text{--}}$

- This algorithm works very well for finding thresholds when the histogram is suitable

Thresholding Example 1



Thresholding Example 2



Problems with single value Thresholding(cont...)

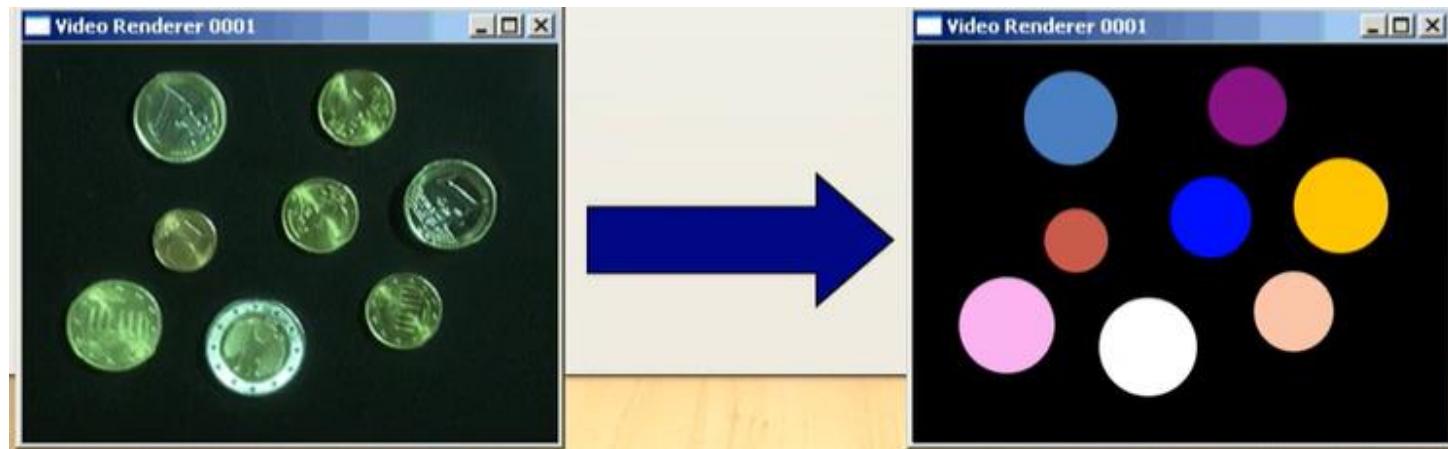
- Let's say we want to isolate the contents of the bottles
- Think about what the histogram for this image would look like
- What would happen if we used a single threshold value?



Image Segmentation

The segmentation problem

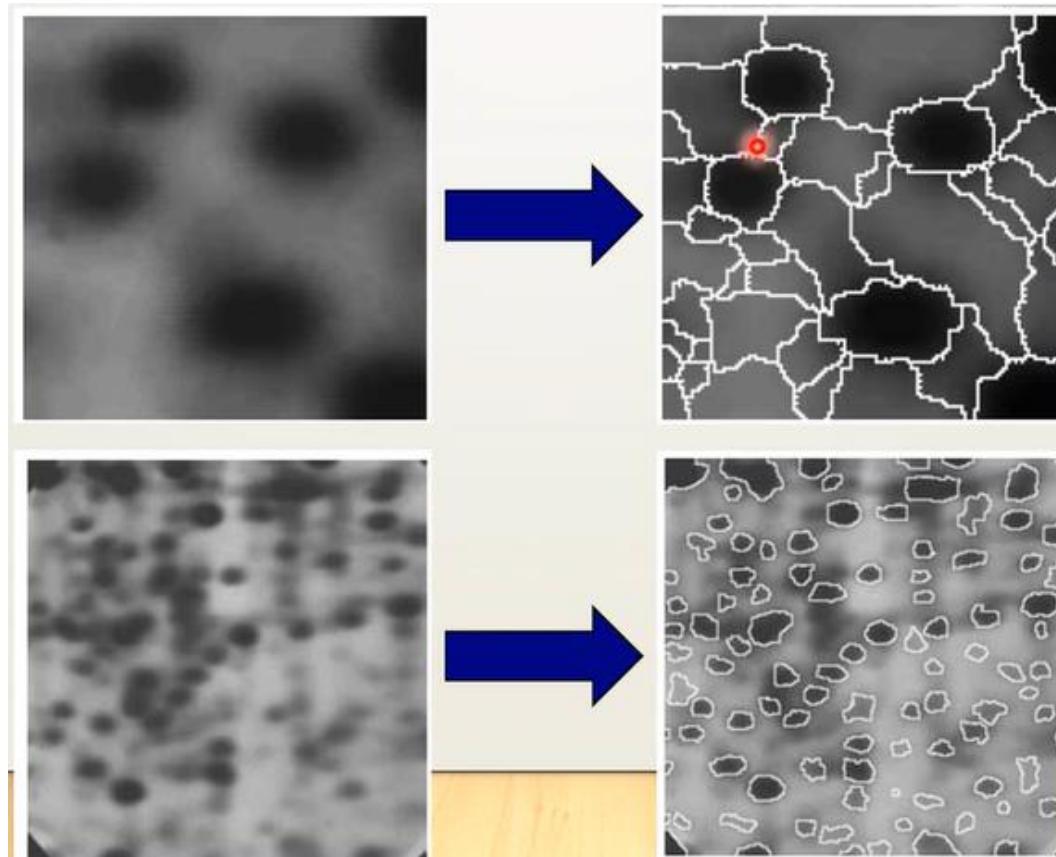
- Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image
- Typically the first step in any automated computer vision application



The segmentation problem

- Image segmentation can be achieved in any one of two ways:
 1. Segmentation based on discontinuities in intensity it partitions the image based on abrupt changes in intensity such as edges
 2. Segmentation based on similarities in intensity it divides the image into regions that are similar according to a set of predefined criteria

Segmentation Examples



Detection of Discontinuities

- There are three basic types of grey level discontinuities that we tend to look for in digital images:
 1. Points
 2. Lines
 3. Edges
- We typically find discontinuities using masks and correlation
- These all are high frequency components hence we need mask which are basically high pass

Point Detection

Point Detection

- Point detection can be achieved simply using the mask below:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Points are detected at those pixels in the subsequent filtered image that are above a set threshold

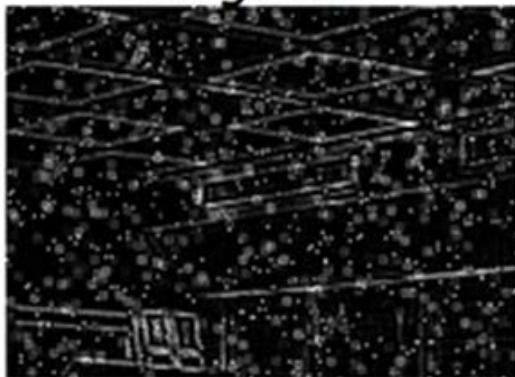
Point Detection



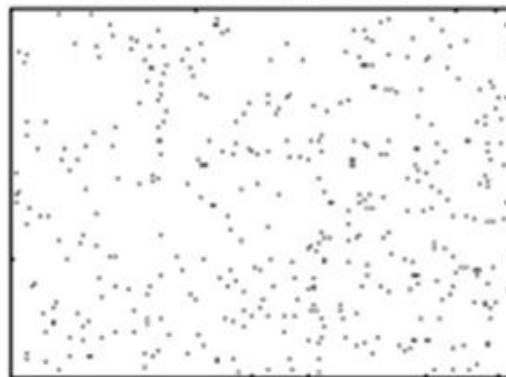
Original



Noise-added

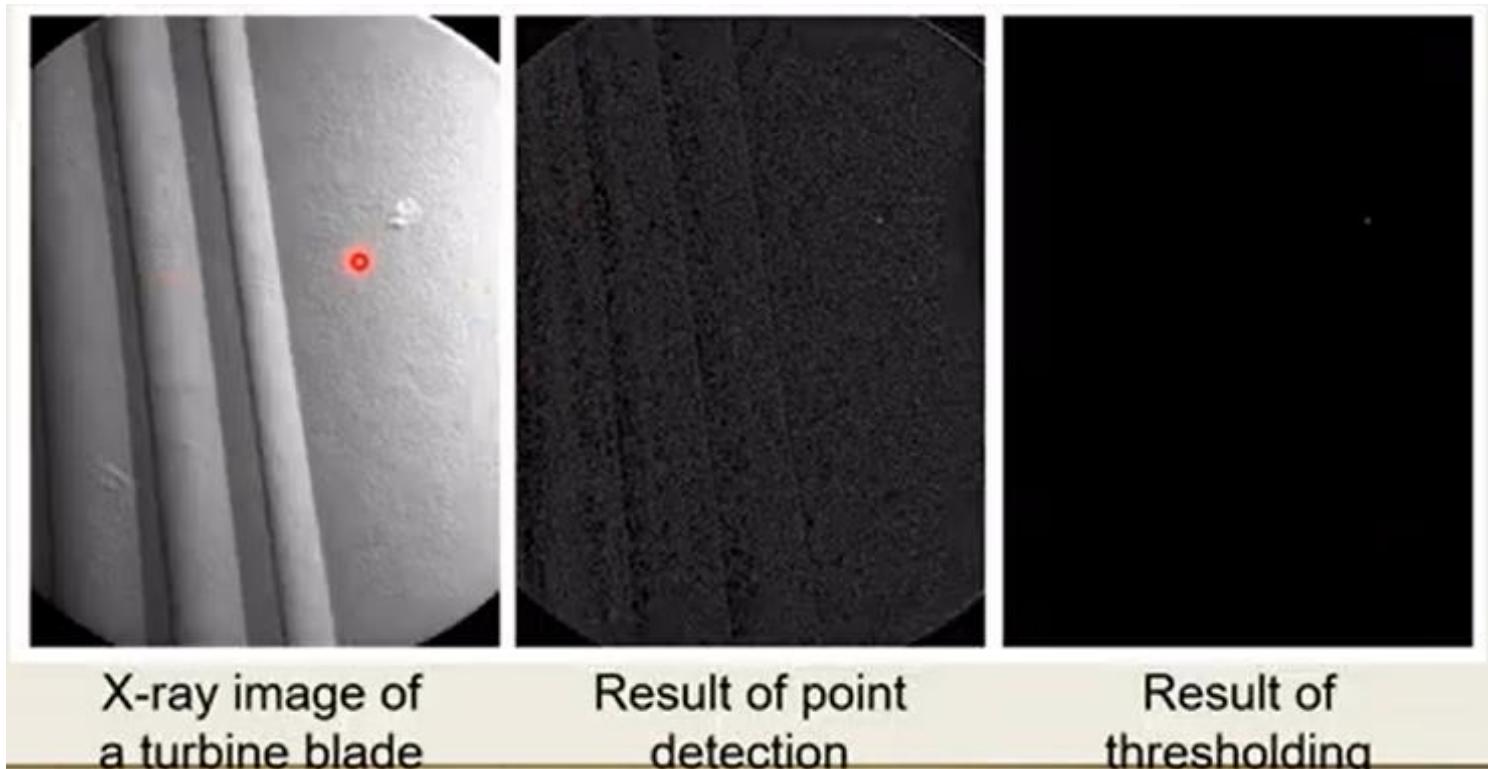


Filtered o/p

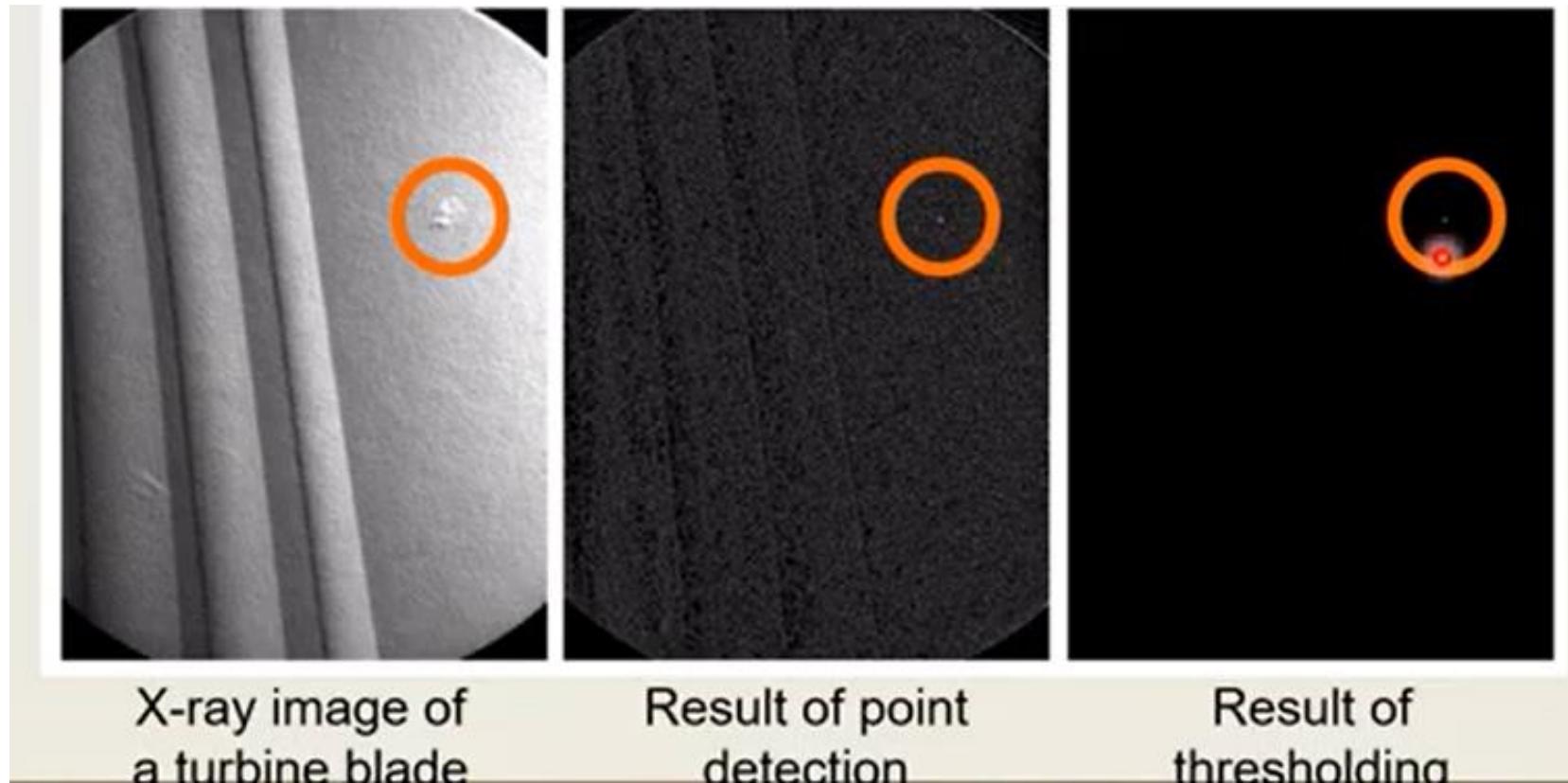


Thresholded o/p

Point Detection



Point Detection



Line Detection

Line Detection

- The next level of complexity is to try to detect lines
- The masks below will extract lines that are one pixel thick and running in a particular direction

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+45°

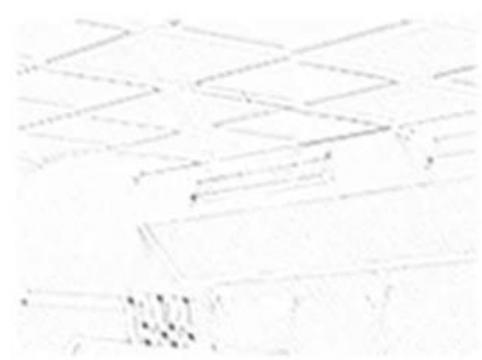
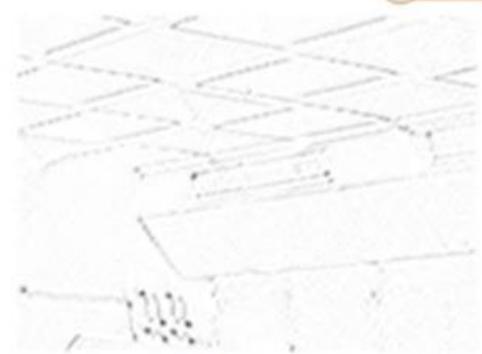
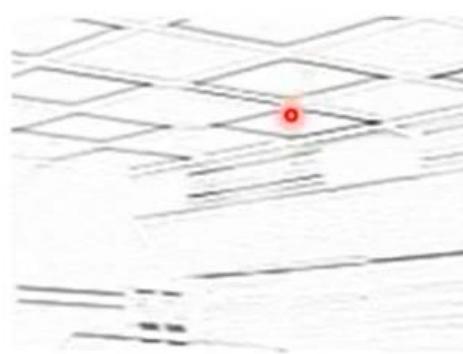
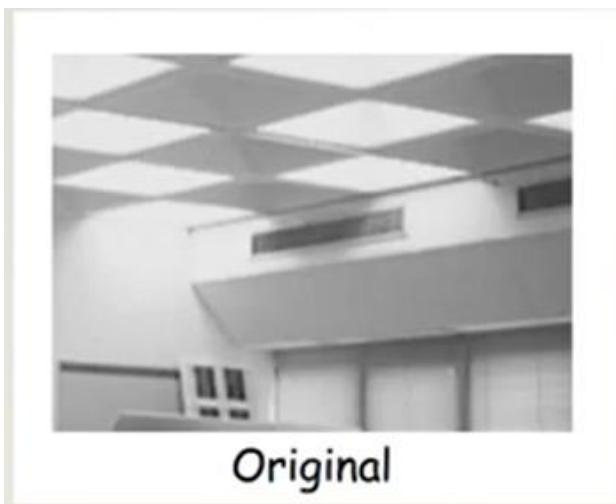
-1	2	-1
-1	2	-1
-1	2	-1

Vertical

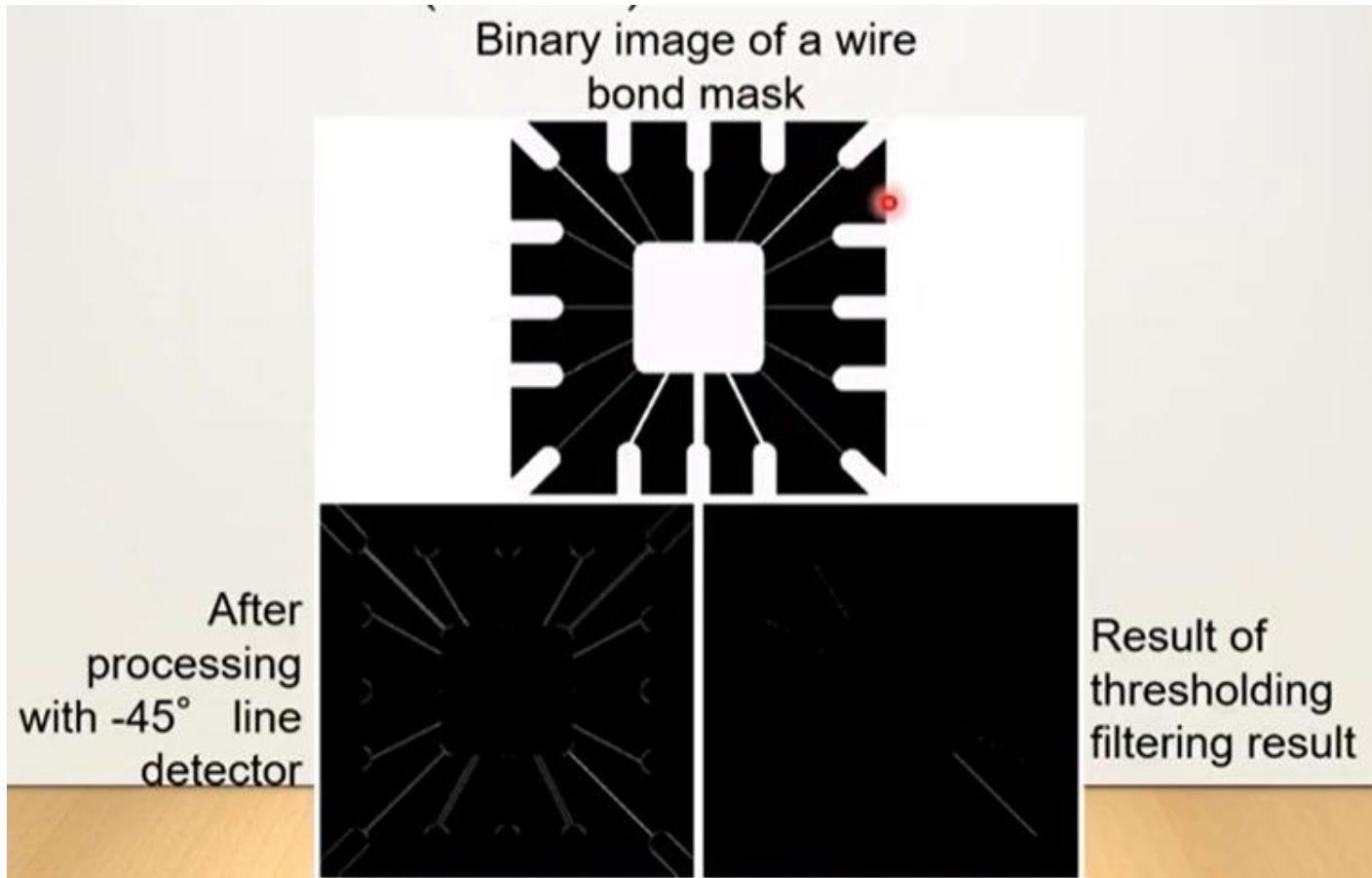
2	-1	-1
-1	2	-1
-1	-1	2

-45°

Line Detection



Line Detection

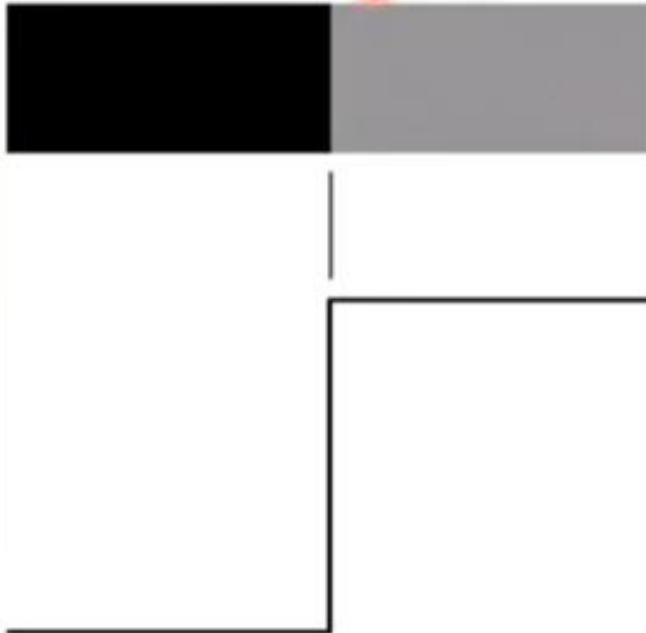


Edge Detection

Edge Detection

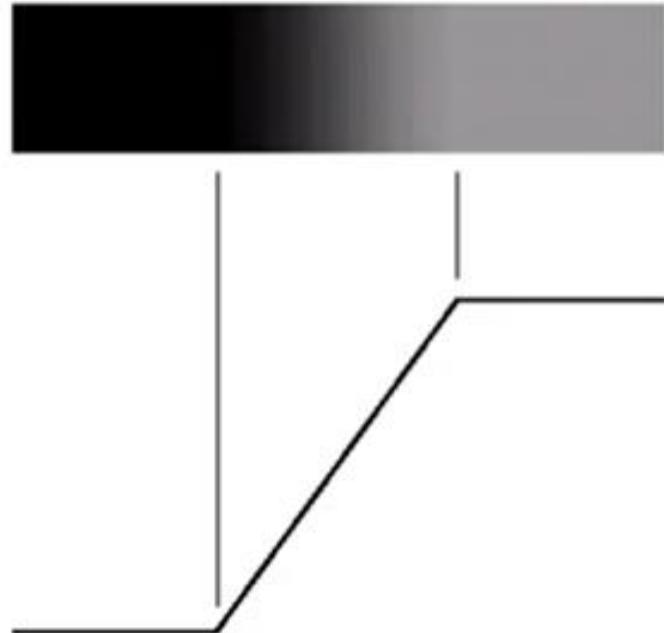
- An edge is a set of connected pixels that lie on the boundary between two regions

Model of an ideal digital edge



Gray-level profile
of a horizontal line
through the image

Model of a ramp digital edge

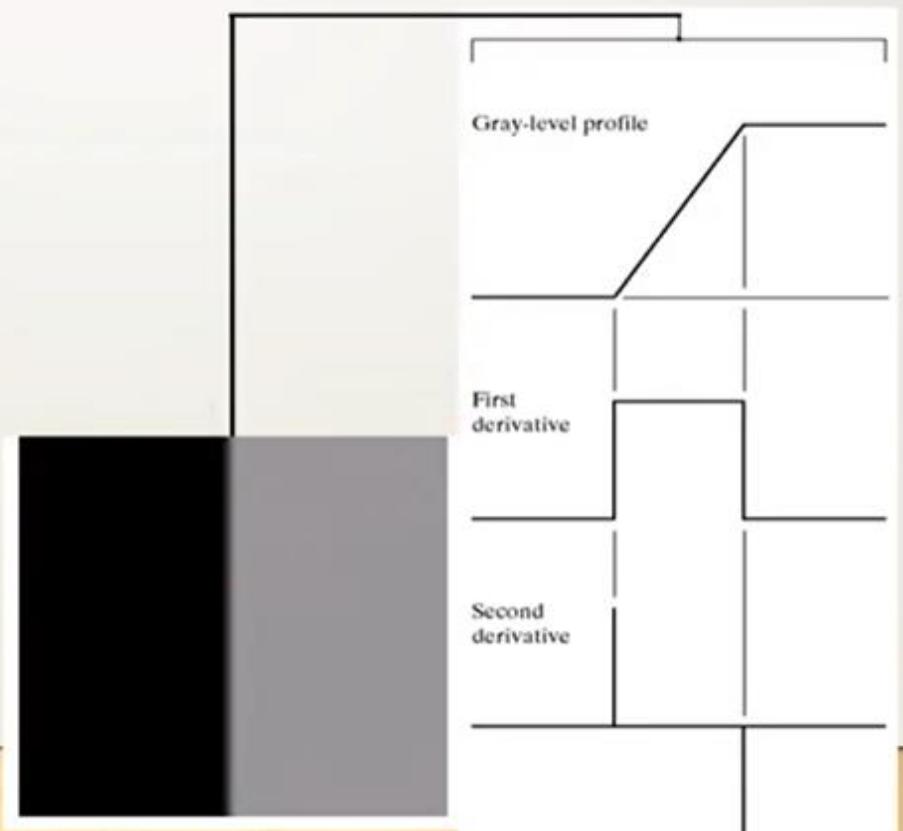


Gray-level profile
of a horizontal line
through the image

Edges & Derivatives

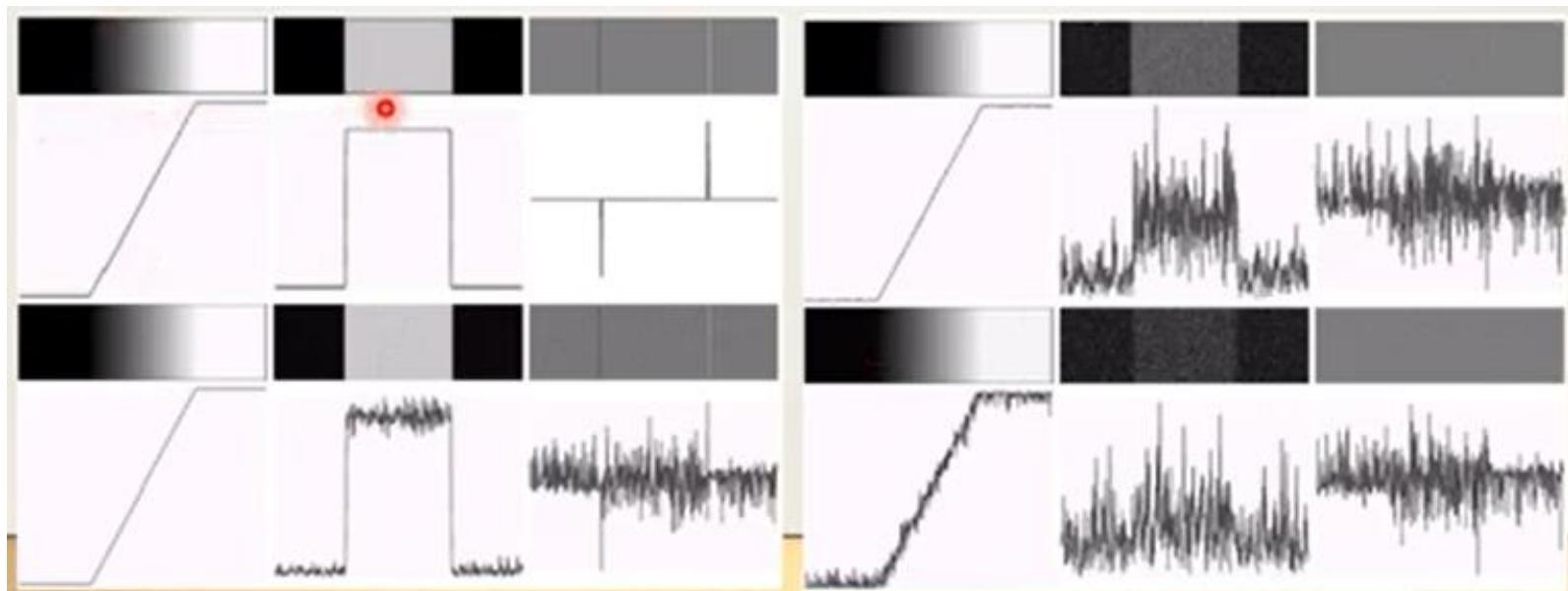
Edges & Derivatives

- We have already spoken about how derivatives are used to find discontinuities
- 1st derivative **tells us** where an edge is
- 2nd derivative can be used to show edge direction



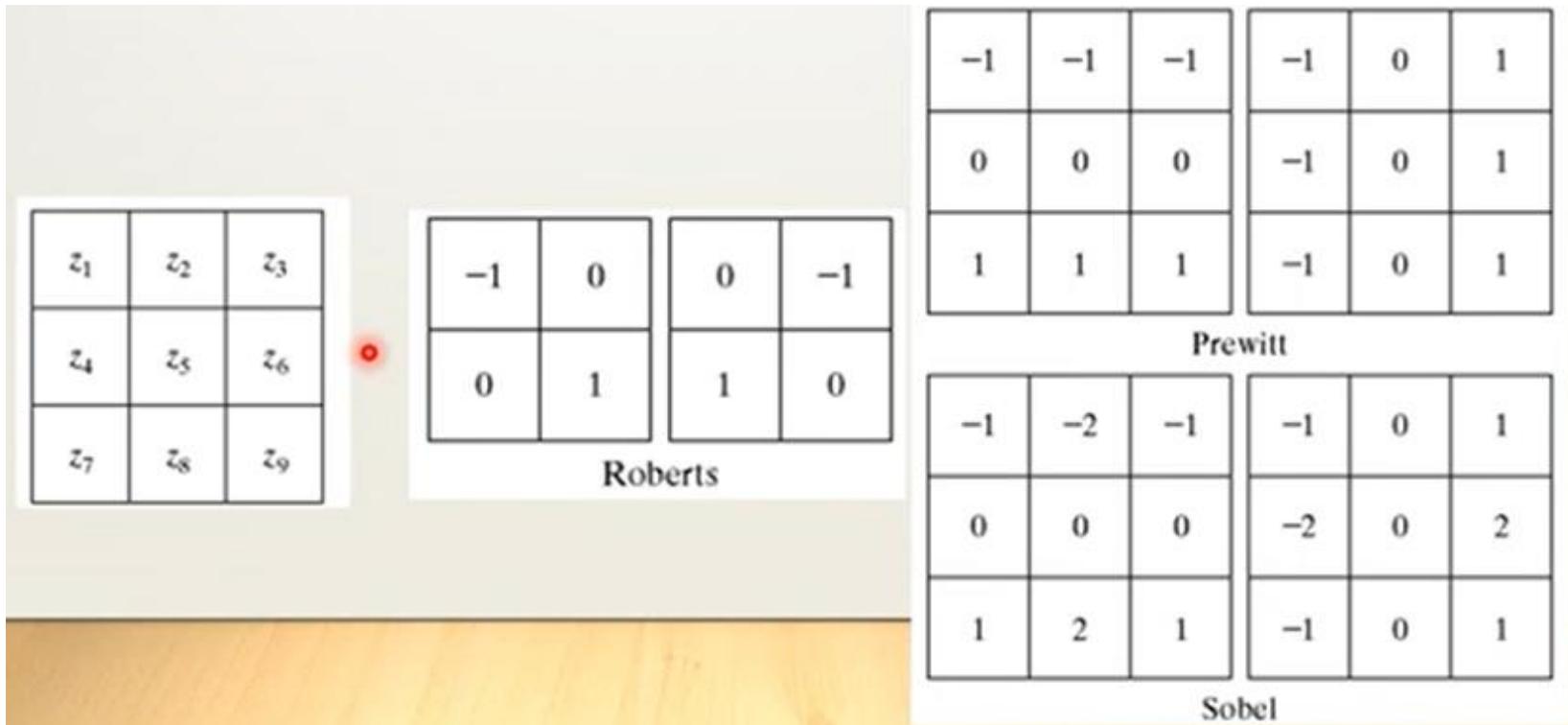
Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise
- We need to keep this in mind



Common Edge Detectors

- Given a 3*3 region an image the following edge detection filters can be used





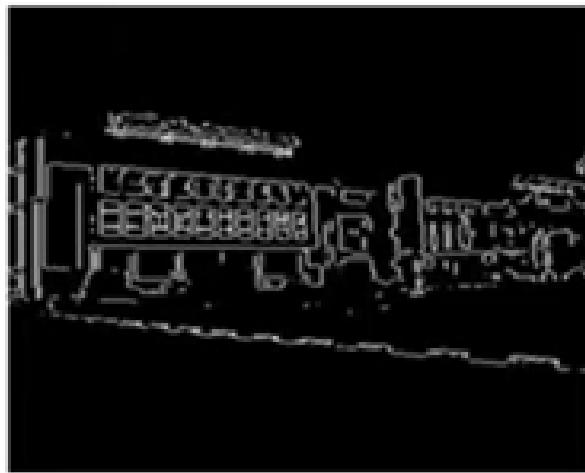
Original



Roberts

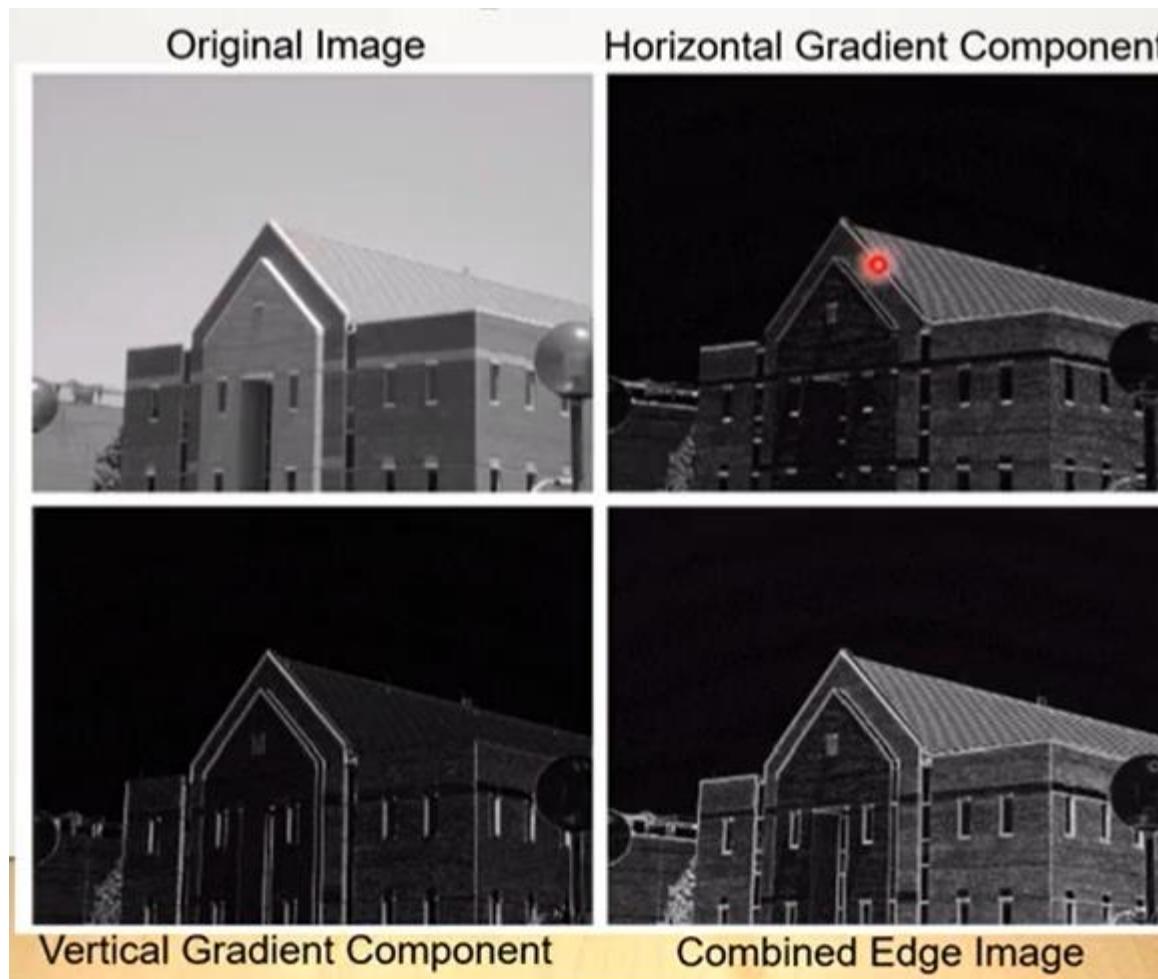


Sobel

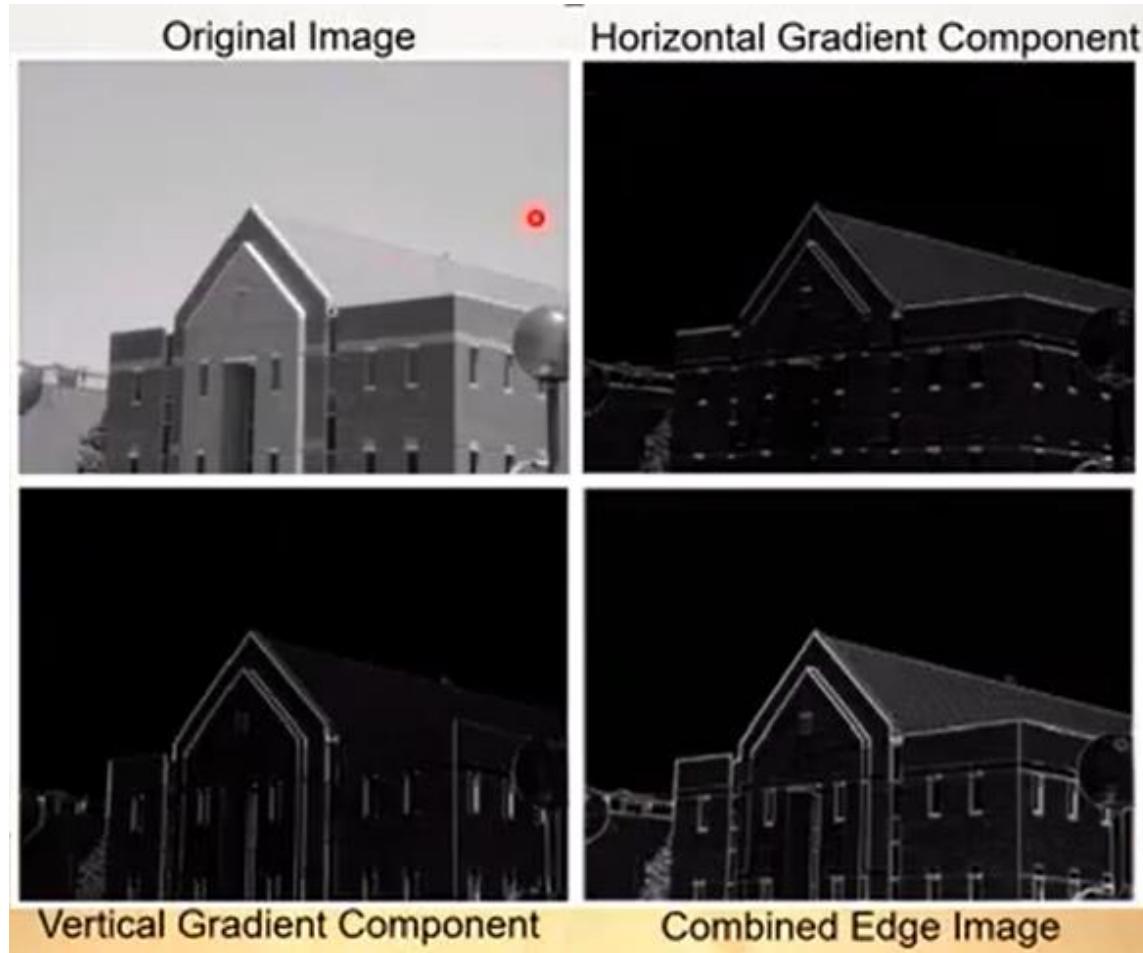


Prewitt

Edge Detection Example



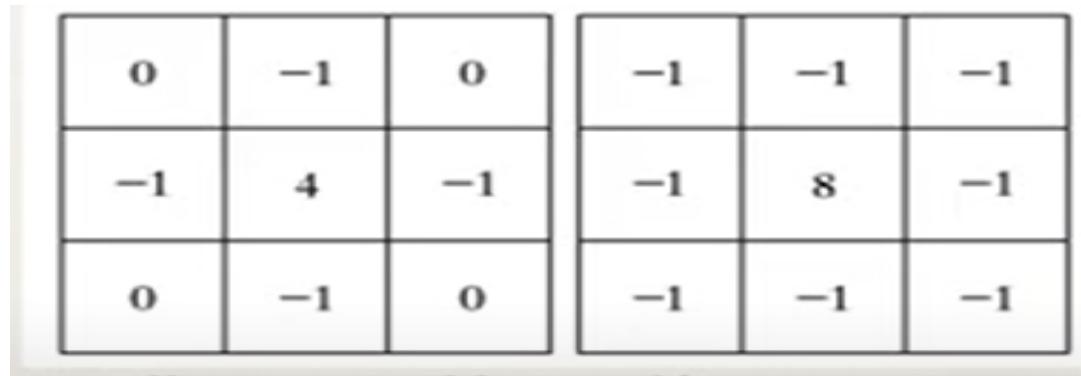
Edge Detection Example with Smoothing



Laplacian Edge Detection

Laplacian Edge Detection

- We encounter the 2nd order derivative based laplacian filter already



The image displays two 3x3 matrices representing Laplacian filters. The left matrix has values: 0, -1, 0 in the top row; -1, 4, -1 in the middle row; and 0, -1, 0 in the bottom row. The right matrix has values: -1, -1, -1 in the top row; -1, 8, -1 in the middle row; and -1, -1, -1 in the bottom row.

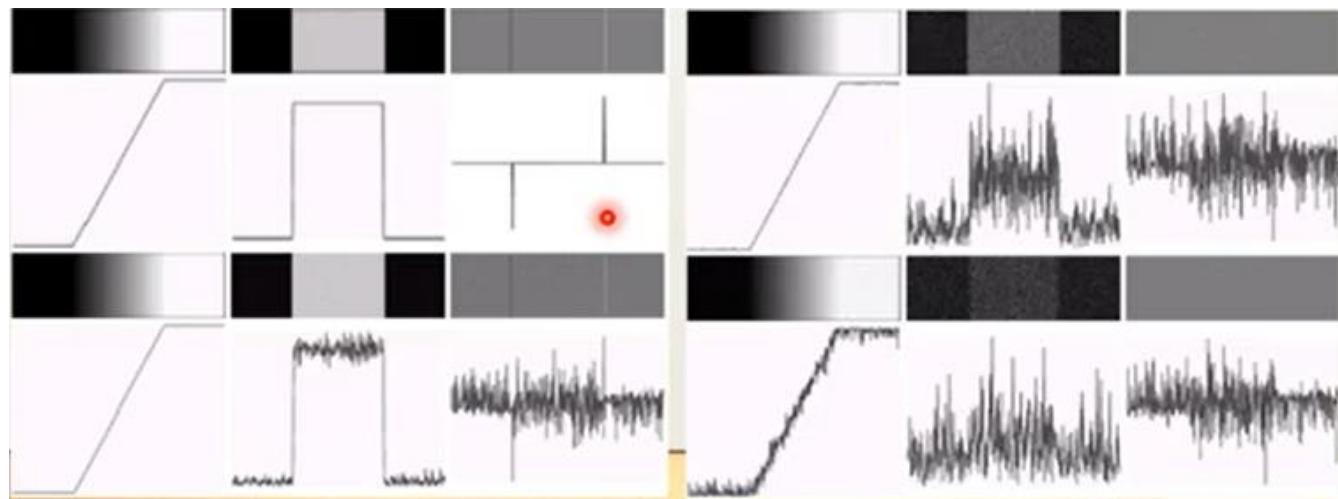
0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

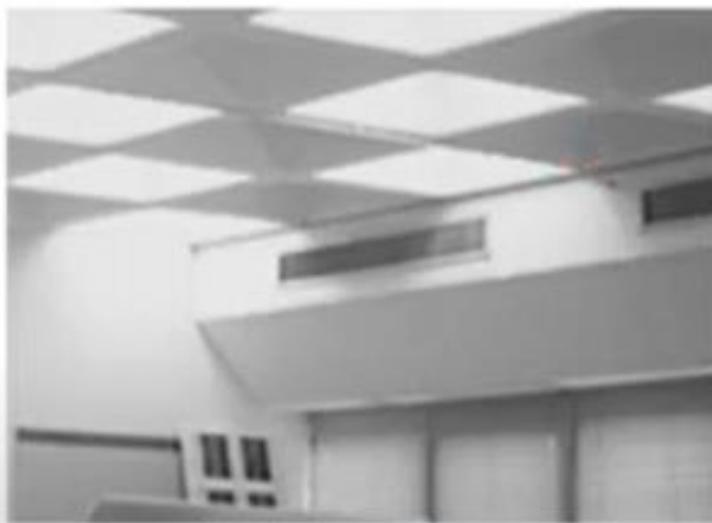
- The laplacian is typically not used by itself as it is too sensitive to noise
- Usually when used for edge detection the Laplacian is combined with a smoothing Gaussian filter

Derivatives & Noise

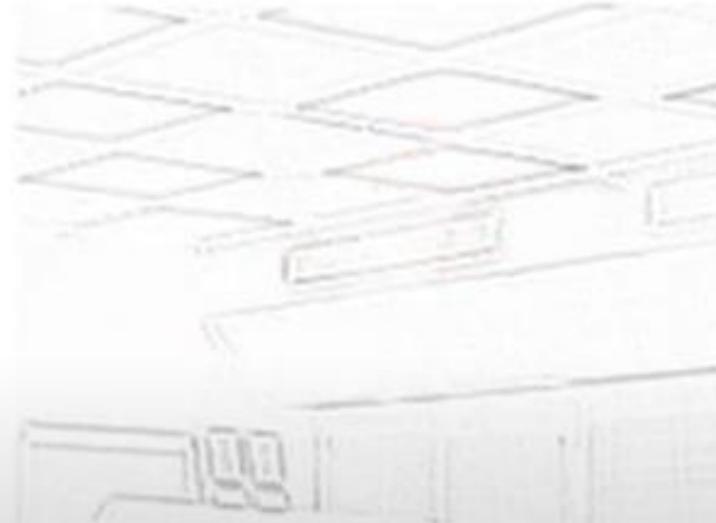
- Derivative based edge detectors are extremely sensitive to noise
- We need to keep this in mind



Laplacian Edge Detection



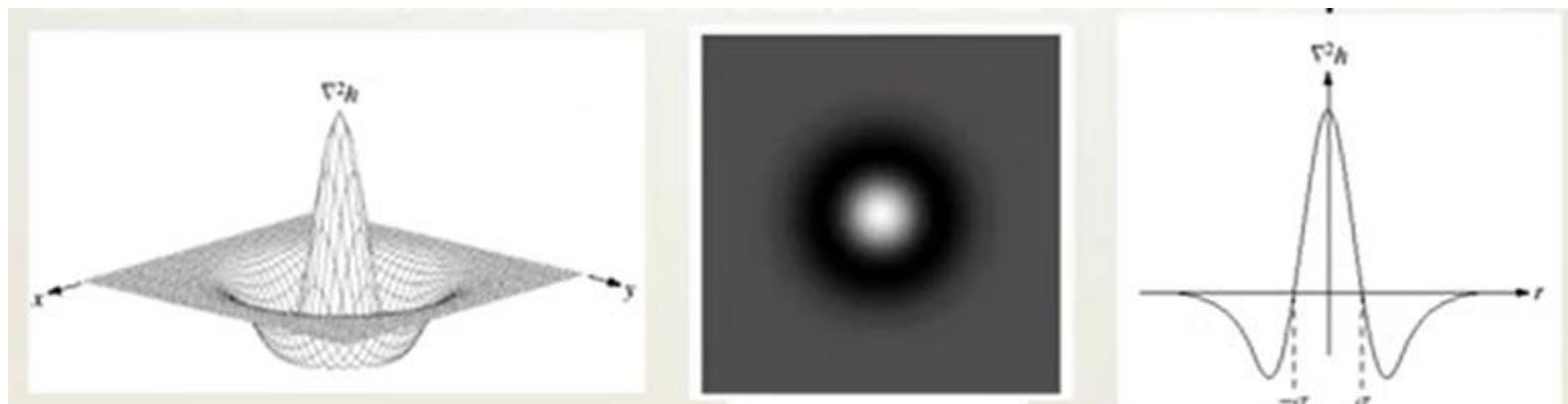
Original



Processed image

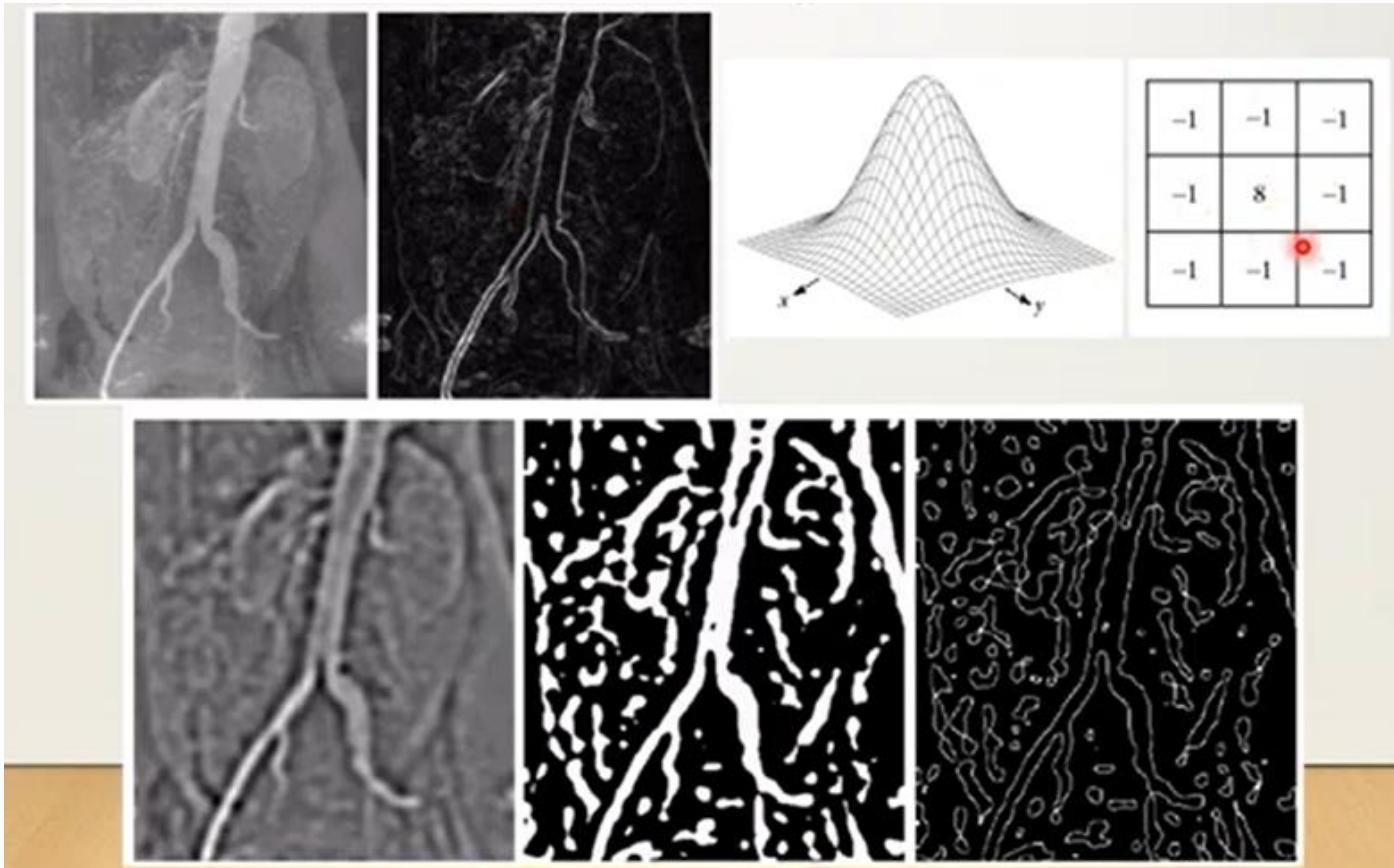
Laplacian of Gaussian

- The Laplacian of Gaussian (or Mexican hat) filter uses the Gaussian for noise removal and the laplacian



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Laplacian of Gaussian



Active Contours

Active Contour

- Segmentation is a section of image processing for separation of information from the required target region of the image. There are different techniques used for segmentation of pixels of interest from the image.
- Active contour is one of the active models in segmentation techniques, which makes use of the energy constraints and forces in the image for separation of regions of interest. Active contour defines a separate boundary or curvature for the regions of target object for segmentation.

Active Contour

- Application of Active Contour
- Medical Imaging
 - Brain CT Images
 - Cardiac Image
 - MRI Image
- Motion Tracking
- Stereo Tracking

What is Active Contour?

Given: Approximate boundary (contour) around the object

Task: Evolve (move) the contour to fit exact object boundary

Active Contour:

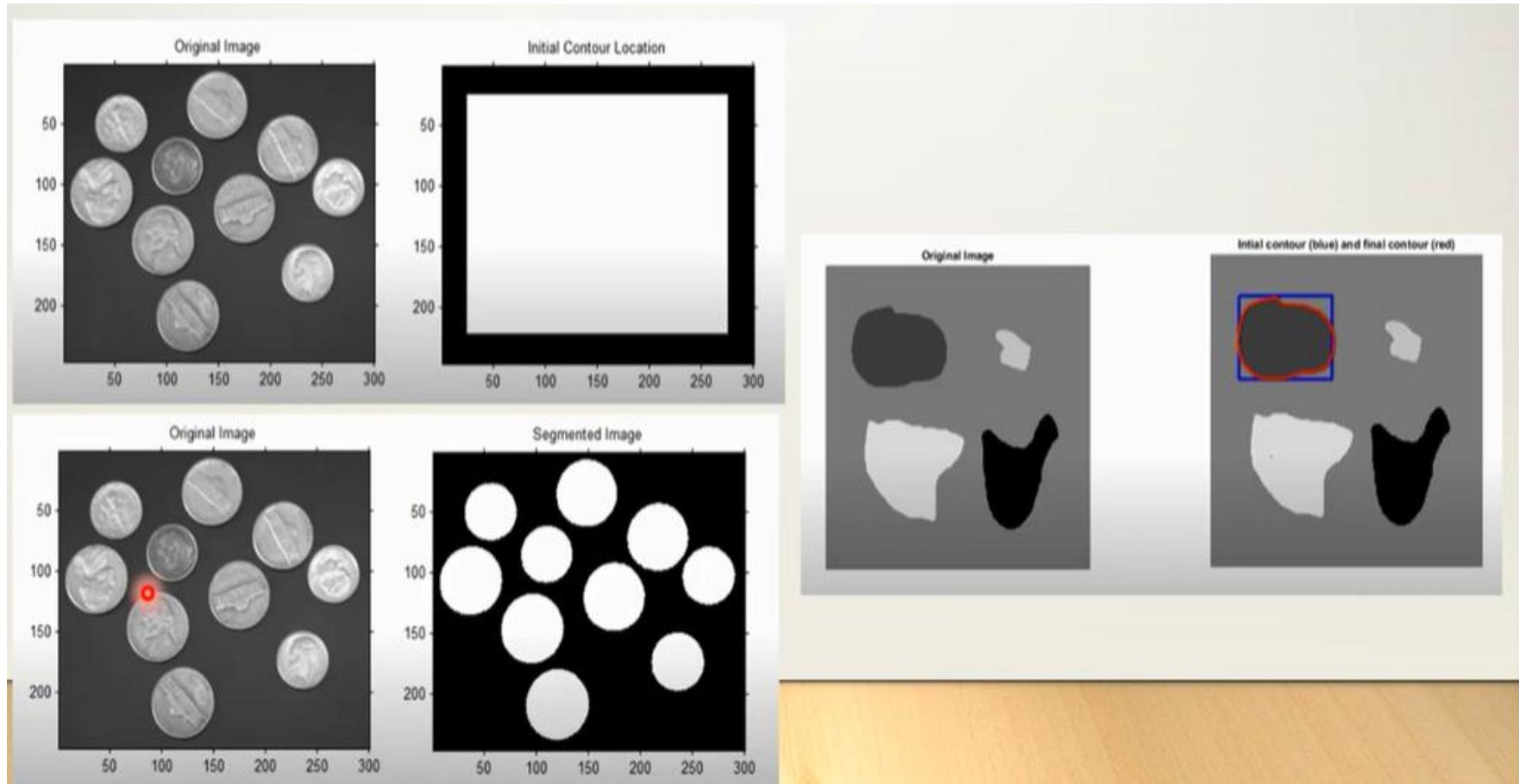
Iteratively “deform” the initial contour so that:

- It is near pixels with high gradient (edges)
- It is smooth



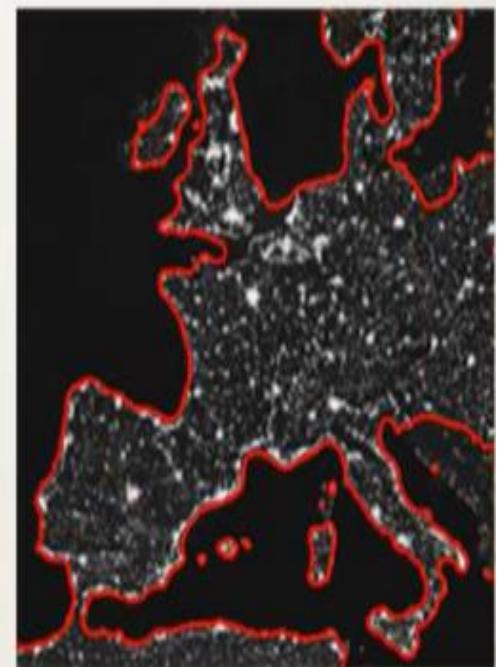
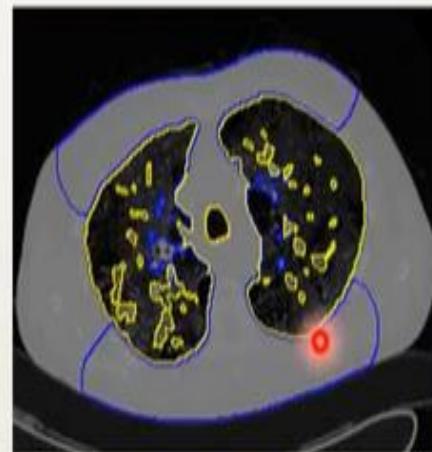
Image

Example of Active Contour



Example of Active Contour

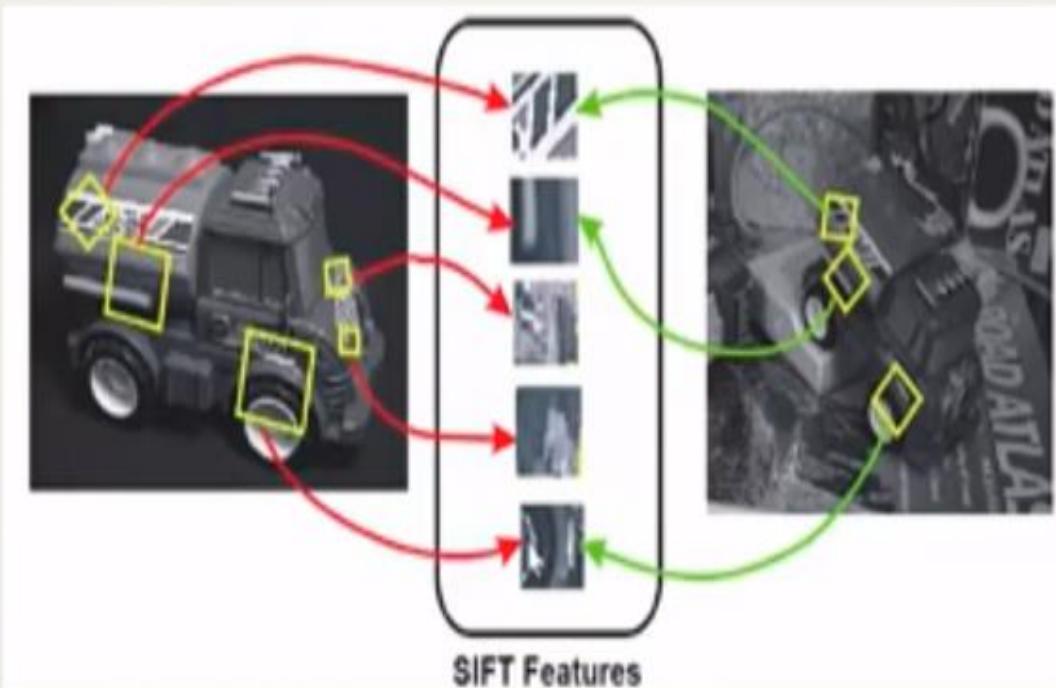
Medical Imaging



SIFT Features

SIFT Features

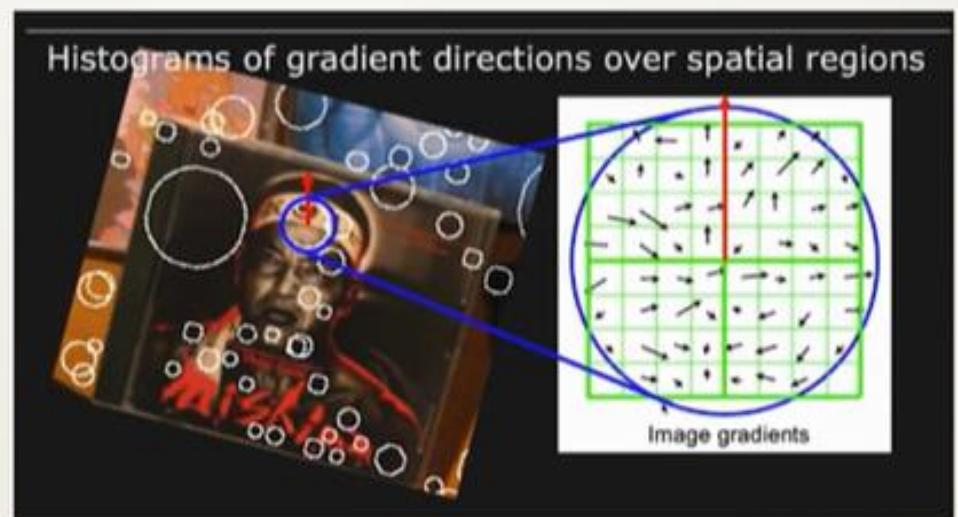
- SIFT stands for Scale-Invariant Feature Transform and was first presented in 2004, by D.Lowe, University of British Columbia. SIFT is invariance to image scale and rotation.



SIFT Features

- Major advantages of SIFT are
- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to a wide range of different feature types, with each adding robustness

SIFT Features



Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

L2 Distance:


$$d(H_1, H_2) = \sqrt{\sum_k (H_1(k) - H_2(k))^2}$$

Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

Normalized Correlation:

$$d(H_1, H_2) = \frac{\sum_k [(H_1(k) - \bar{H}_1)(H_2(k) - \bar{H}_2)]}{\sqrt{\sum_k (H_1(k) - \bar{H}_1)^2} \sqrt{\sum_k (H_2(k) - \bar{H}_2)^2}}$$

where: $\bar{H}_l = \frac{1}{N} \sum_{k=1}^N H_l(k)$

Larger the distance metric, better the match.

Perfect match when $d(H_1, H_2) = 1$

Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

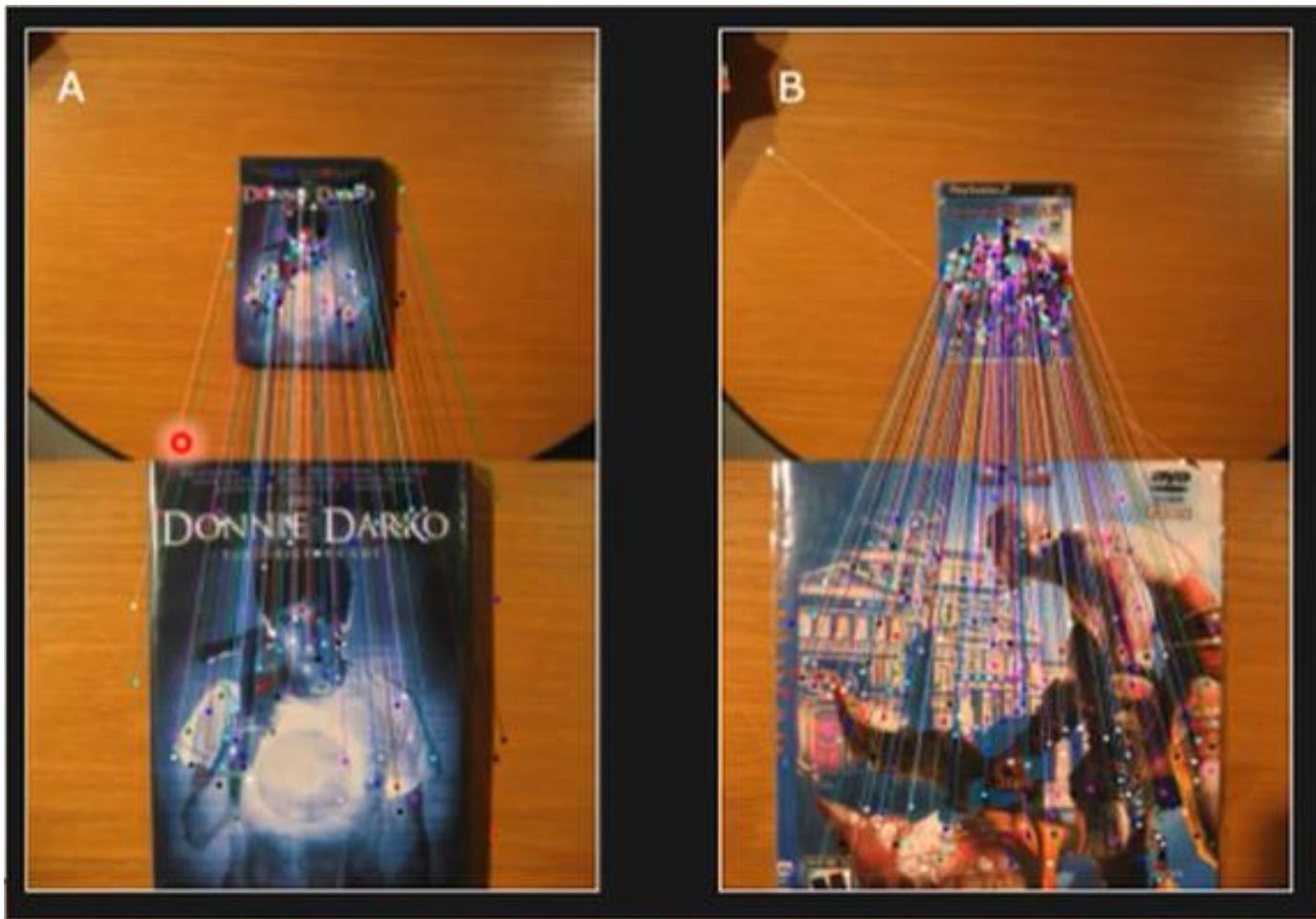
Intersection:



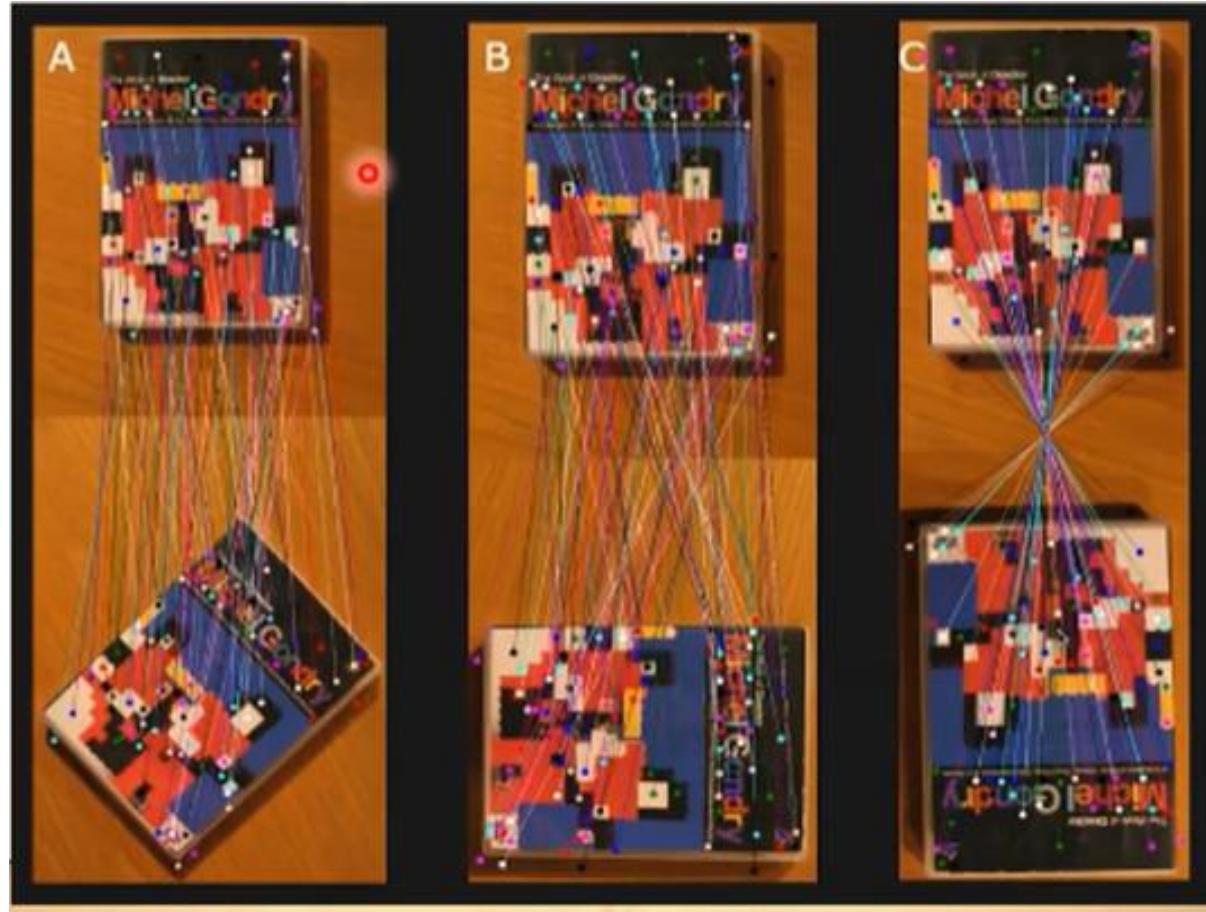
$$d(H_1, H_2) = \sum_k \min(H_1(k), H_2(k))$$

Larger the distance metric, better the match.

SIFT Results: Scale Invariance



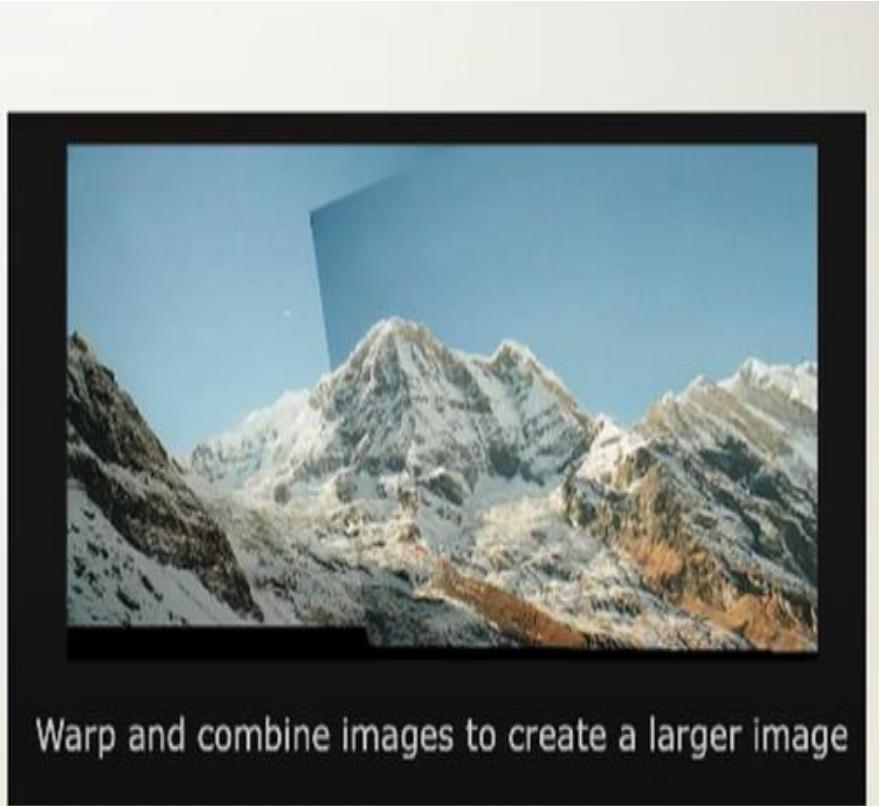
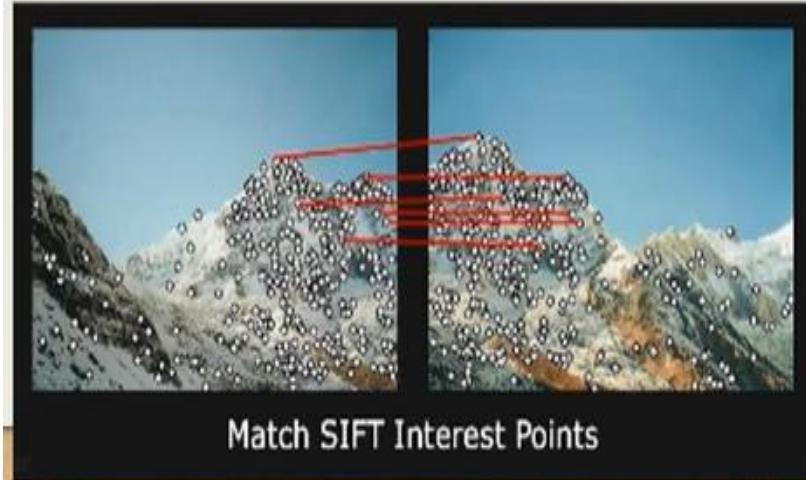
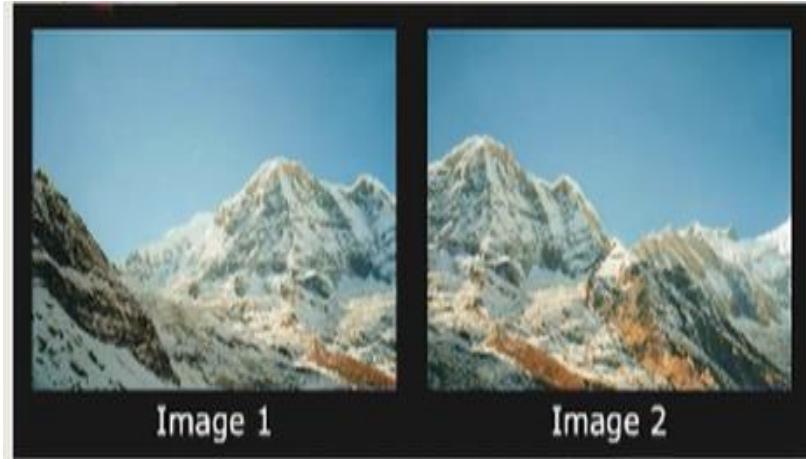
SIFT Results: Rotation Invariance



SIFT Results: Robustness to Clutter



Panorama Stitching using SIFT

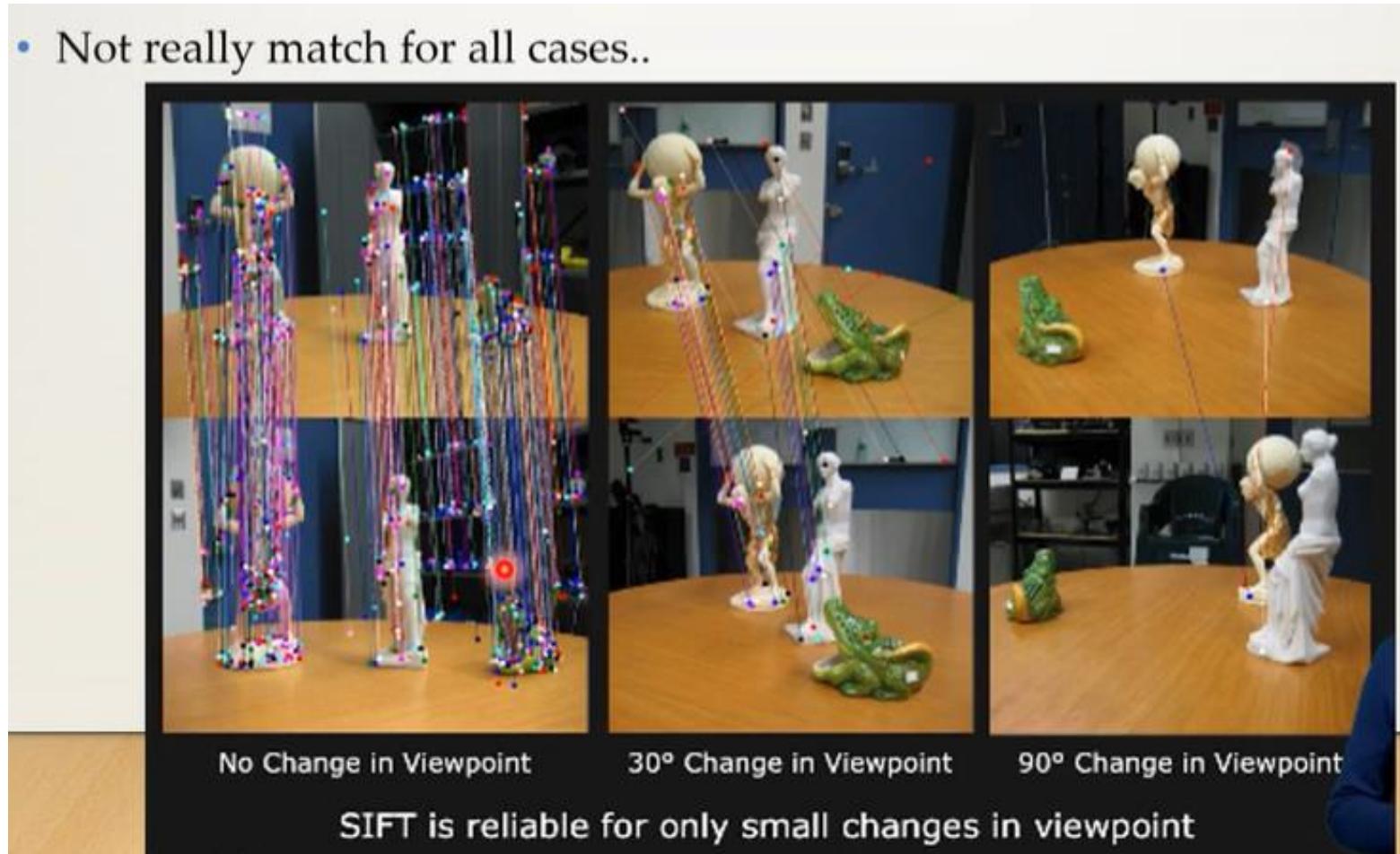


Auto Collage using SIFT



SIFT for 3D Objects?

- Not really match for all cases..



The Algorithm for SIFT

The Algorithm

- SIFT is quite an involved algorithm. There are mainly four steps involved in the SIFT algorithm. We will see them one-by-one.
- **Scale-space peak selection:** Potential location for finding features.
- **Keypoint Localization:** Accurately locating the feature keypoints.
- **Orientation Assignment:** Assigning orientation to keypoints.
- **Keypoint descriptor:** Describing the keypoints as a high dimensional vector.
- **Keypoint Matching**



Introduction to SIFT

- *SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm in Computer Vision.*
- SIFT helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.
- For example, here is another image of the Eiffel Tower along with its smaller version. The keypoints of the object in the first image are matched with the keypoints found in the second image. The same goes for two images when the object in the other image is slightly rotated.

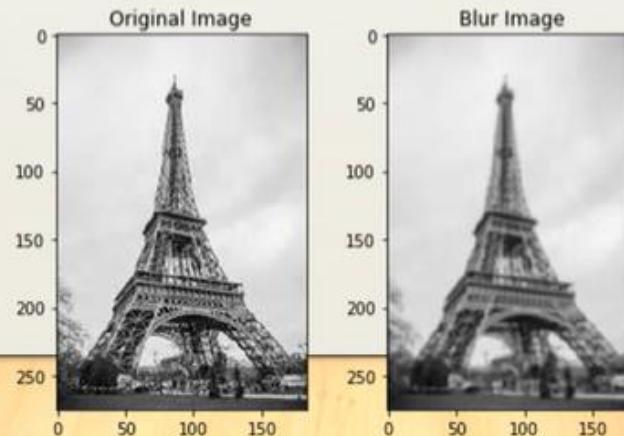


Introduction to SIFT

- **Constructing a Scale Space:** To make sure that features are scale-independent
- **Keypoint Localisation:** Identifying the suitable features or keypoints
- **Orientation Assignment:** Ensure the keypoints are rotation invariant
- **Keypoint Descriptor:** Assign a unique fingerprint to each keypoint

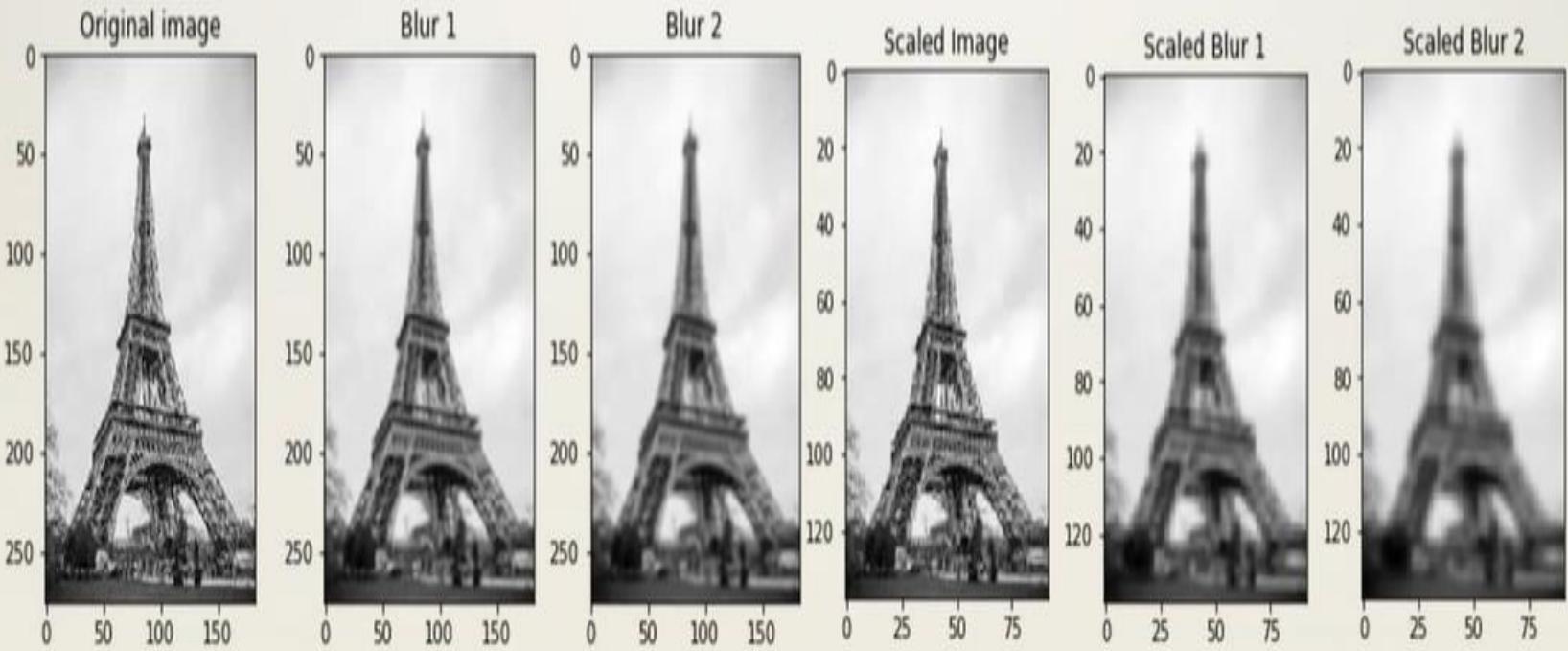
Constructing the scale space

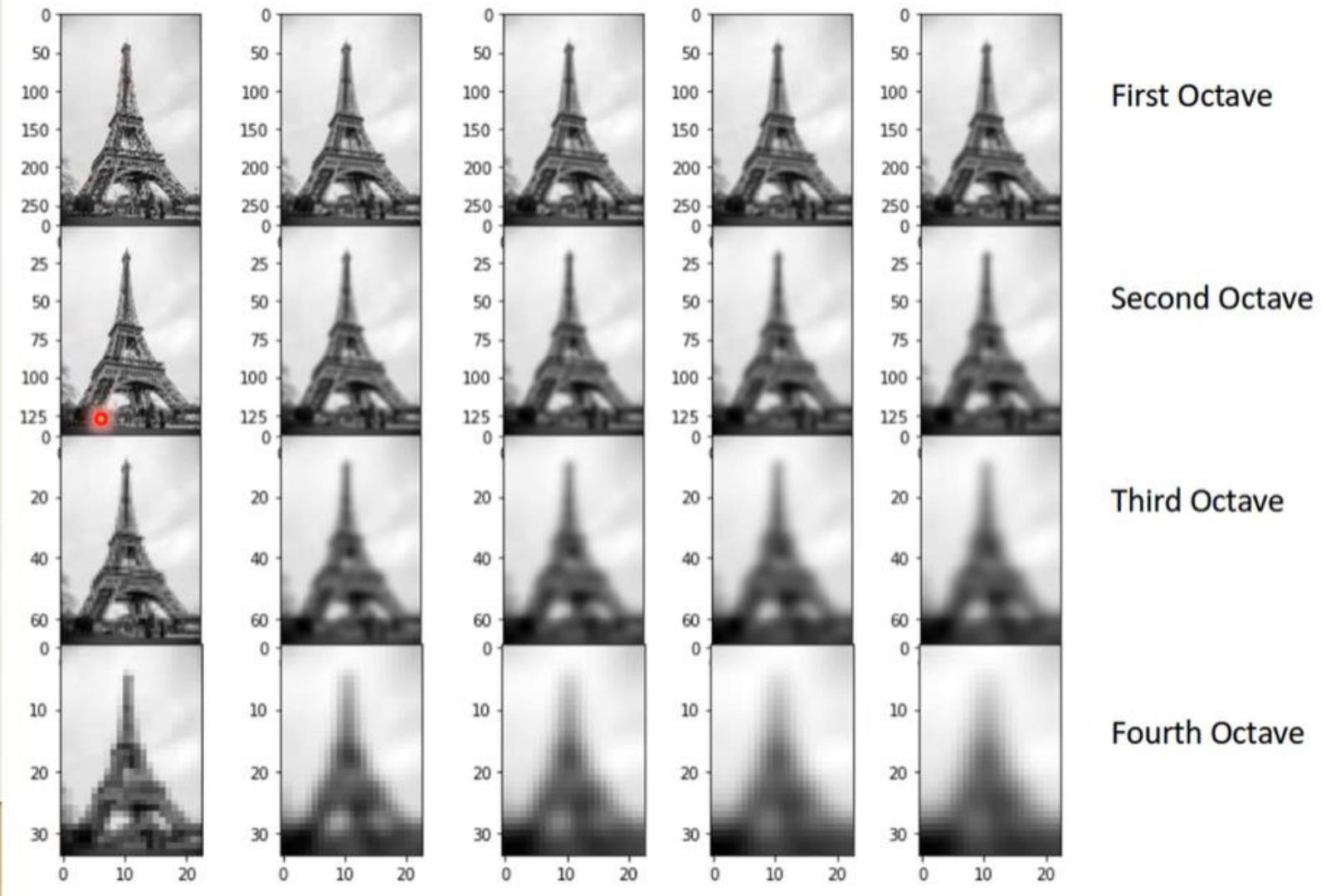
- We use the *Gaussian Blurring technique* to reduce the noise in an image.
- So, for every pixel in an image, the Gaussian Blur calculates a value based on its neighboring pixels. Below is an example of image before and after applying the Gaussian Blur. As you can see, the texture and minor details are removed from the image and only the relevant information like the shape and edges remain:



Constructing the scale space

- Hence, these blur images are created for multiple scales. To create a new set of images of different scales, we will take the original image and reduce the scale by half. For each new image, we will create blur versions as we saw above.





First Octave

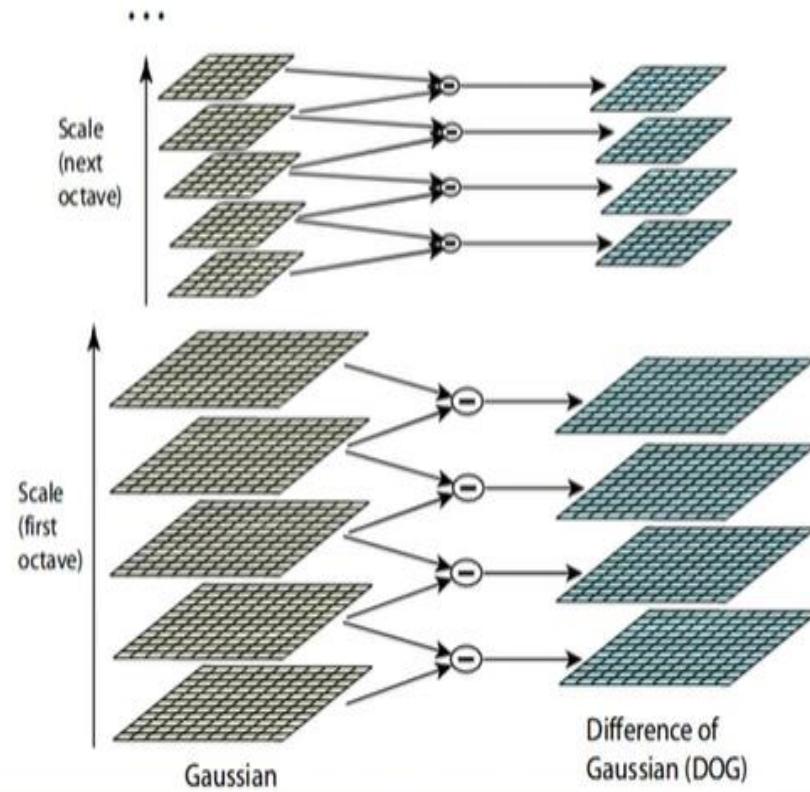
Second Octave

Third Octave

Fourth Octave

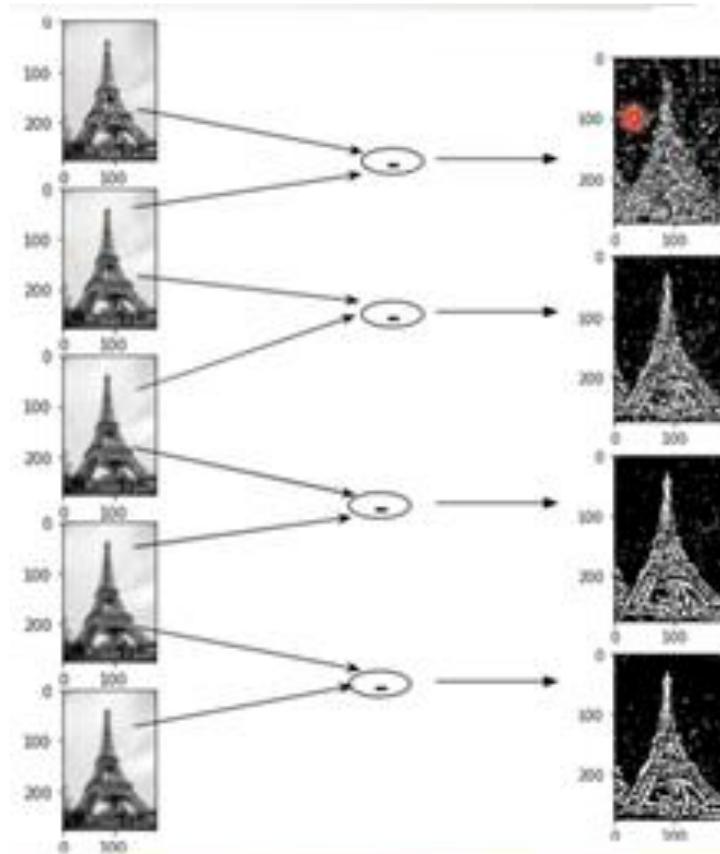
Difference of Gaussian

- Difference of Gaussian is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original.



Difference of Gaussian

- Let us create the DoG for the images in scale space. Take a look at the below diagram. On the left, we have 5 images, all from the first octave (thus having the same scale). Each subsequent image is created by applying the Gaussian blur over the previous image.
- On the right, we have four images generated by subtracting the consecutive Gaussians.

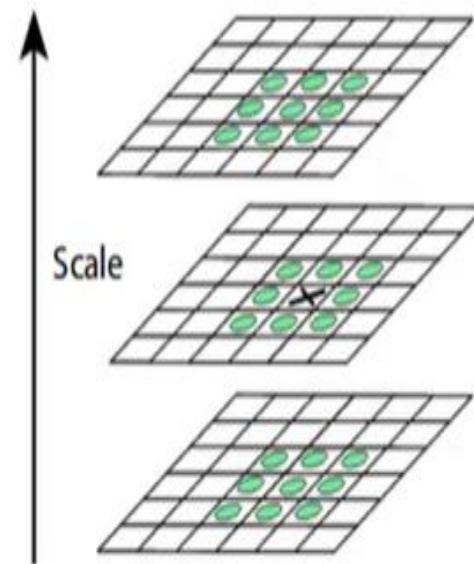


Key point Localization

- Once the images have been created, the next step is to find the important keypoints from the image that can be used for feature matching. **The idea is to find the local maxima and minima for the images.** This part is divided into two steps:
 1. Find the local maxima and minima
 2. Remove low contrast keypoints (keypoint selection)

Key point Localization

- This means that every pixel value is compared with 26 other pixel values to find whether it is the local maxima/minima. For example, in the below diagram, we have three images from the first octave. The pixel marked x is compared with the neighboring pixels (in green) and is selected as a keypoint if it is the highest or lowest among the neighbors:
- We now have potential keypoints that represent the images and are scale-invariant. We will apply the last check over the selected keypoints to ensure that these are the most accurate keypoints to represent the image.



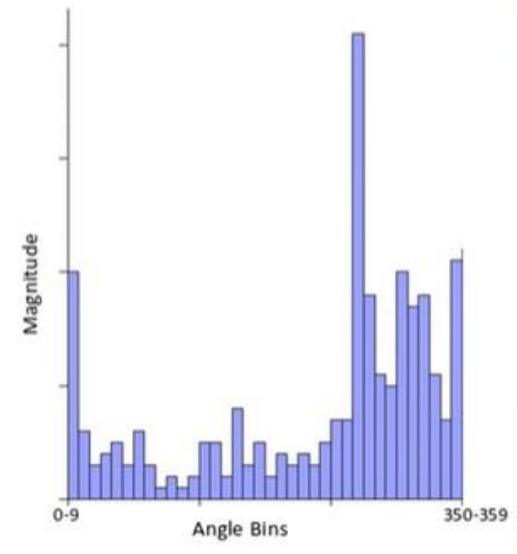
Orientation Assignment

- At this stage, we have a set of stable keypoints for the images. We will now assign an orientation to each of these keypoints so that they are invariant to rotation. We can again divide this step into two smaller steps:
 1. Calculate the magnitude and orientation
 2. Create a histogram for magnitude and orientation

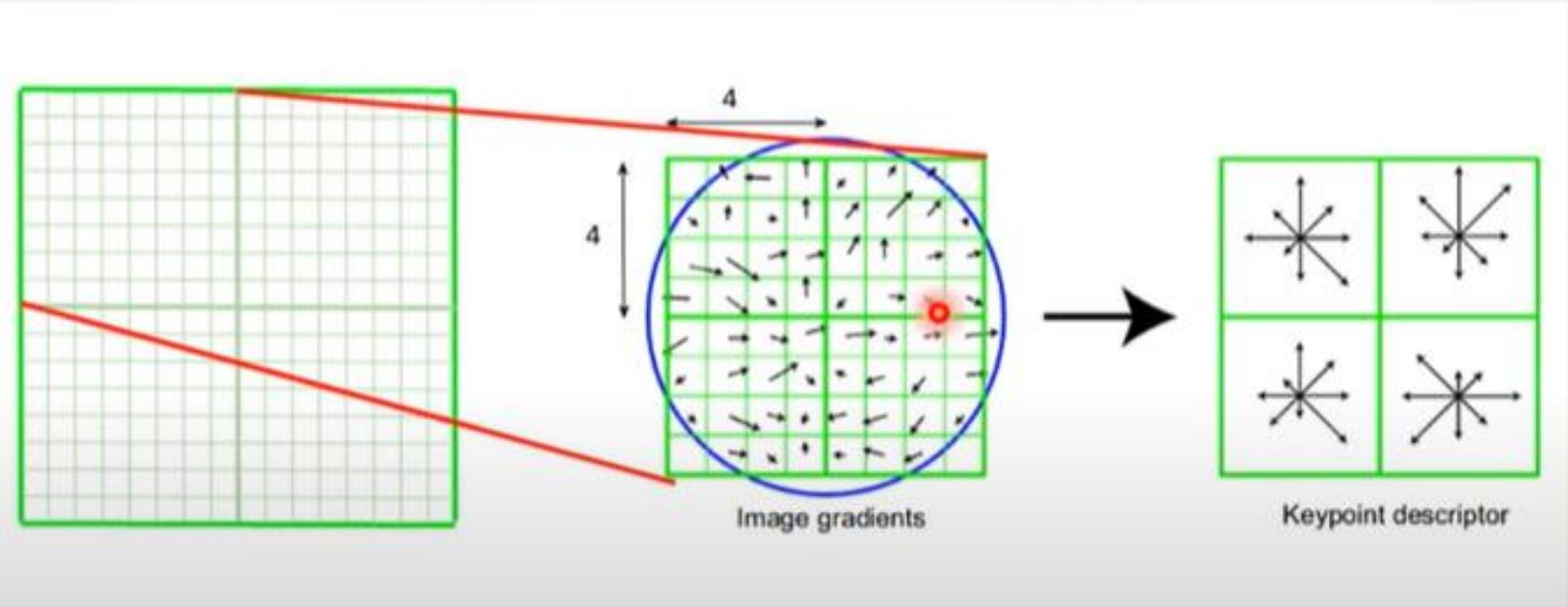
Orientation Assignment

- Let's say we want to find the magnitude and orientation for the pixel value in red. For this, we will calculate the gradients in x and y directions by taking the difference between 55 & 46 and 56 & 42. This comes out to be $G_x = 9$ and $G_y = 14$ respectively.
- Once we have the gradients, we can find the magnitude and orientation using the following formulas:
 - Magnitude = $\sqrt{(G_x)^2 + (G_y)^2} = 16.64$
 - $\Phi = \text{atan}(G_y / G_x) = \text{atan}(1.55) = 57.17$

35	40	41	45	50
40	40	42	46	52
42	46	50	55	55
48	52	56	58	60
56	60	65	70	75



Key point Descriptor

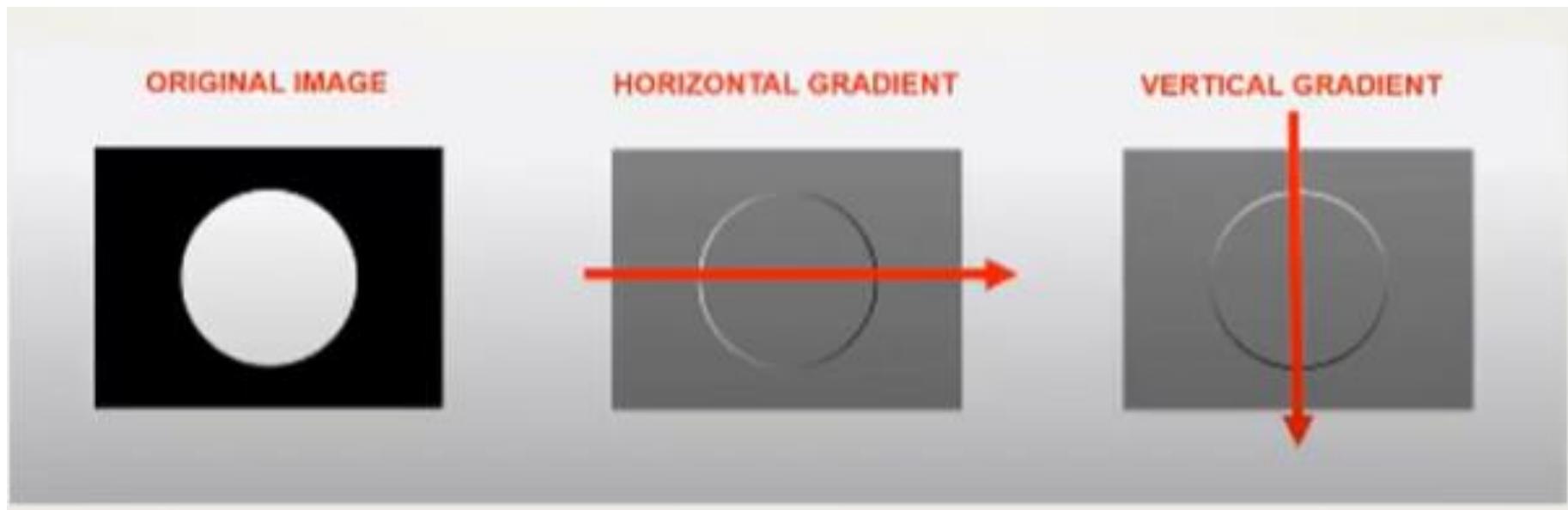


Key point Descriptor

- This is the final step for SIFT. So far, we have stable keypoints that are scale-invariant and rotation invariant. In this section, we will use the neighboring pixels, their orientations, and magnitude, to generate a unique fingerprint for this keypoint called a ‘descriptor’.
- Additionally, since we use the surrounding pixels, the  descriptors will be partially invariant to illumination or brightness of the images.
- We will first take a 16×16 neighborhood around the keypoint. This 16×16 block is further divided into 4×4 sub-blocks and for each of these sub-blocks, we generate the histogram using magnitude and orientation.

HOG Features

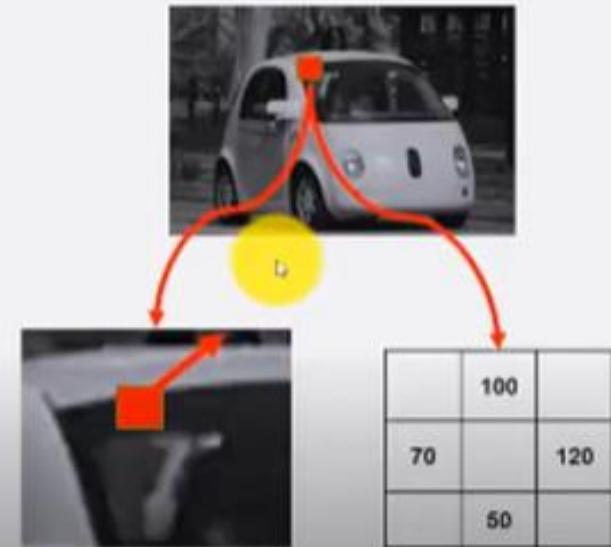
HOG(Histogram of Oriented Gradients)



Gradient Magnitude and Angle

- The gradient value in the X-direction is $120-70=50$
- Y-direction is $100-50=50$.
- Putting it together we will have $[50\ 50]$ feature vector.
- The magnitude and direction are calculated as follows:

$$\text{Gradient Magnitude} = \sqrt{(50)^2 + (50)^2} = 70.1$$
$$\text{Gradient Angle} = \tan^{-1}(50/50) = 45^\circ$$



HOG(Histogram of Oriented Gradients)

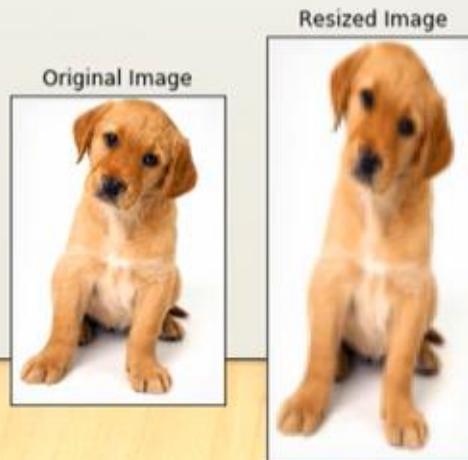
- Consider the below image of size (180 x 280). Let us take a detailed look at how the HOG features will be created for this image:



HOG(Histogram of Oriented Gradients)

- **Step:1 Preprocessed Data (64 x128)**

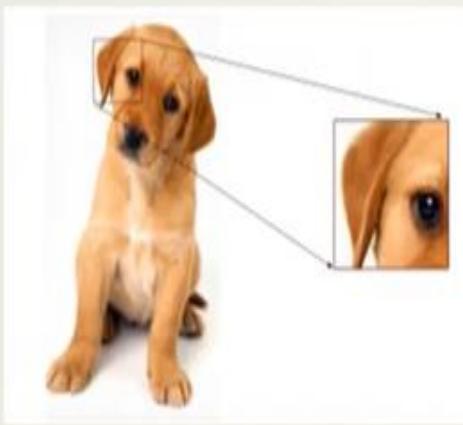
- We need to preprocess the image and bring down the width to height ratio to 1:2. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations pretty simple. In fact, this is the exact value used in the original paper.



HOG(Histogram of Oriented Gradients)

- Step:2 Calculating Gradients (direction x and y)

- The next step is to calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions. Here, I am going to take a small patch from the image and calculate the gradients on that:



121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

HOG(Histogram of Oriented Gradients)

- I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.
- Hence the resultant gradients in the x and y direction for this pixel are:
 - Change in X direction(G_x) = $89 - 78 = 11$
 - Change in Y direction(G_y) = $68 - 56 = 8$

HOG(Histogram of Oriented Gradients)

- Step 3: Calculate the Magnitude and Orientation

- The gradients are basically the base and perpendicular here. So, for the previous example, we had G_x and G_y as 11 and 8. Let's apply the Pythagoras theorem to calculate the total gradient magnitude:

- Total Gradient Magnitude = $\sqrt{(G_x)^2 + (G_y)^2}$

- Total Gradient Magnitude = $\sqrt{(11)^2 + (8)^2} = 13.6$

- Next, calculate the orientation (or direction) for the same pixel. We know that we can write the tan for the angles:

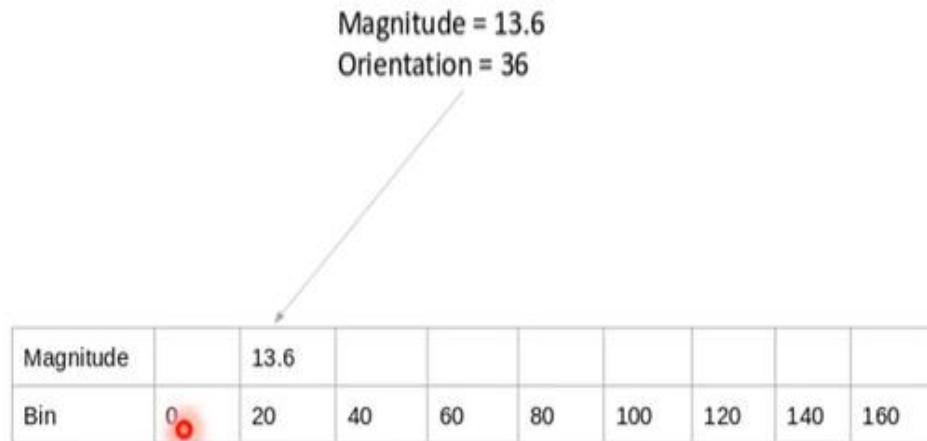
- $\tan(\Phi) = G_y / G_x$

- Hence, the value of the angle would be:

- $\Phi = \arctan(G_y / G_x) = \arctan(8/11) = 36$

HOG(Histogram of Oriented Gradients)

- **Step 4: Create Histograms using Gradients and Orientation**
 - A histogram is a plot that shows the frequency distribution of a set of continuous data. We have the variable (in the form of bins) on the x-axis and the frequency on the y-axis. Here, we are going to take the angle or orientation on the x-axis and the frequency on the y-axis.
 - we can use the gradient magnitude to fill the values in the matrix.



HOG(Histogram of Oriented Gradients)

- Step: 5 Calculate Histogram of Gradients in 8×8 cells (9×1)

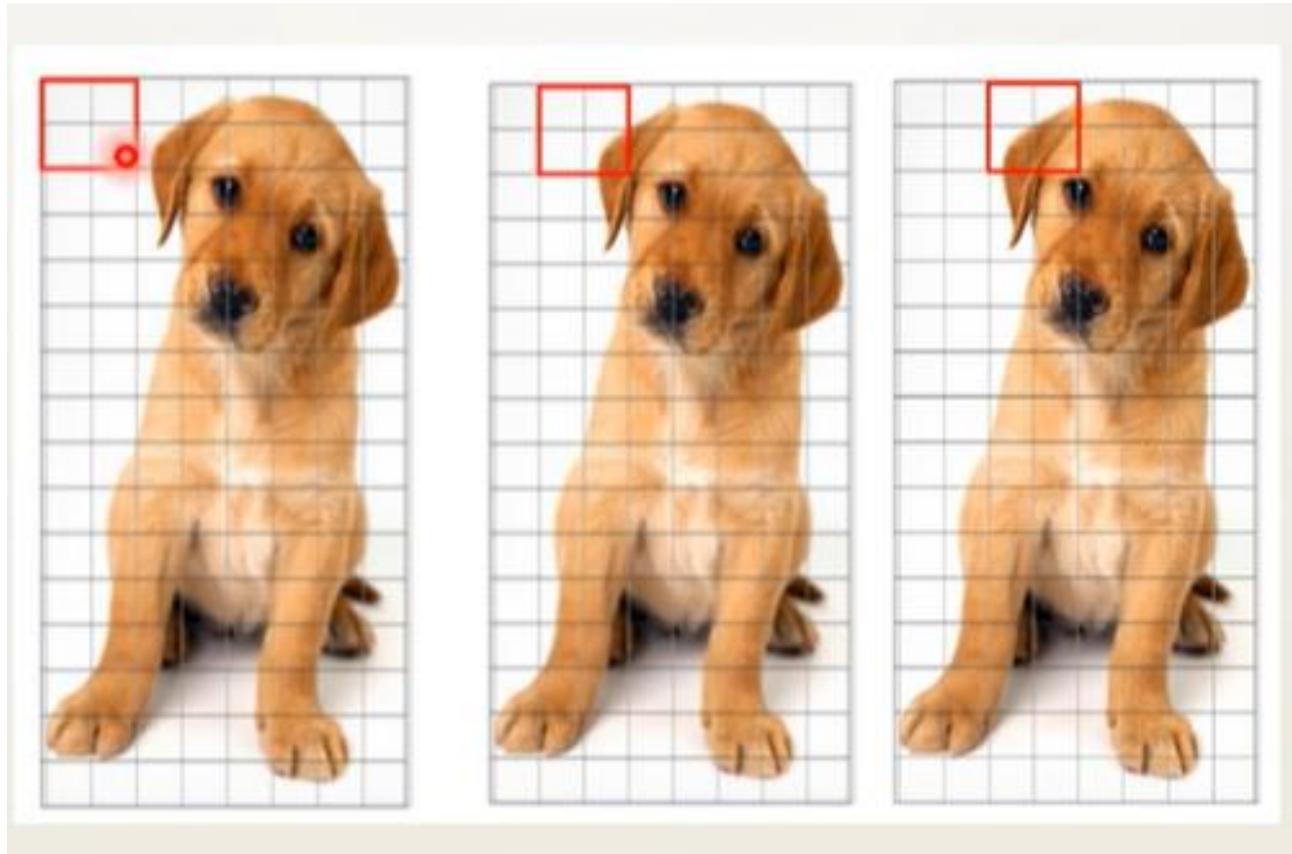
- The histograms created in the HOG feature descriptor are not generated for the whole image. Instead, the image is divided into 8×8 cells, and the histogram of oriented gradients is computed for each cell. Why do you think this happens?
- By doing so, we get the features (or histogram) for the smaller patches which in turn represent the whole image. We can certainly change this value here from 8×8 to 16×16 or 32×32 .
- If we divide the image into 8×8 cells and generate the histograms, we will get a 9×1 matrix for each cell. This matrix is generated using method 4 that we discussed in the previous section.



HOG(Histogram of Oriented Gradients)

- Step: 6 Normalize gradients in 16×16 cell (36×1)
 - Before we understand how this is done, it's important to understand why this is done in the first place.
 - Although we already have the HOG features created for the 8×8 cells of the image, the gradients of the image are sensitive to the overall lighting. This means that for a particular picture, some portion of the image would be very bright as compared to the other portions.
 - We cannot completely eliminate this from the image. But we can reduce this lighting variation by normalizing the gradients by taking 16×16 blocks. Here is an example that can explain how 16×16 blocks are created:

HOG(Histogram of Oriented Gradients)

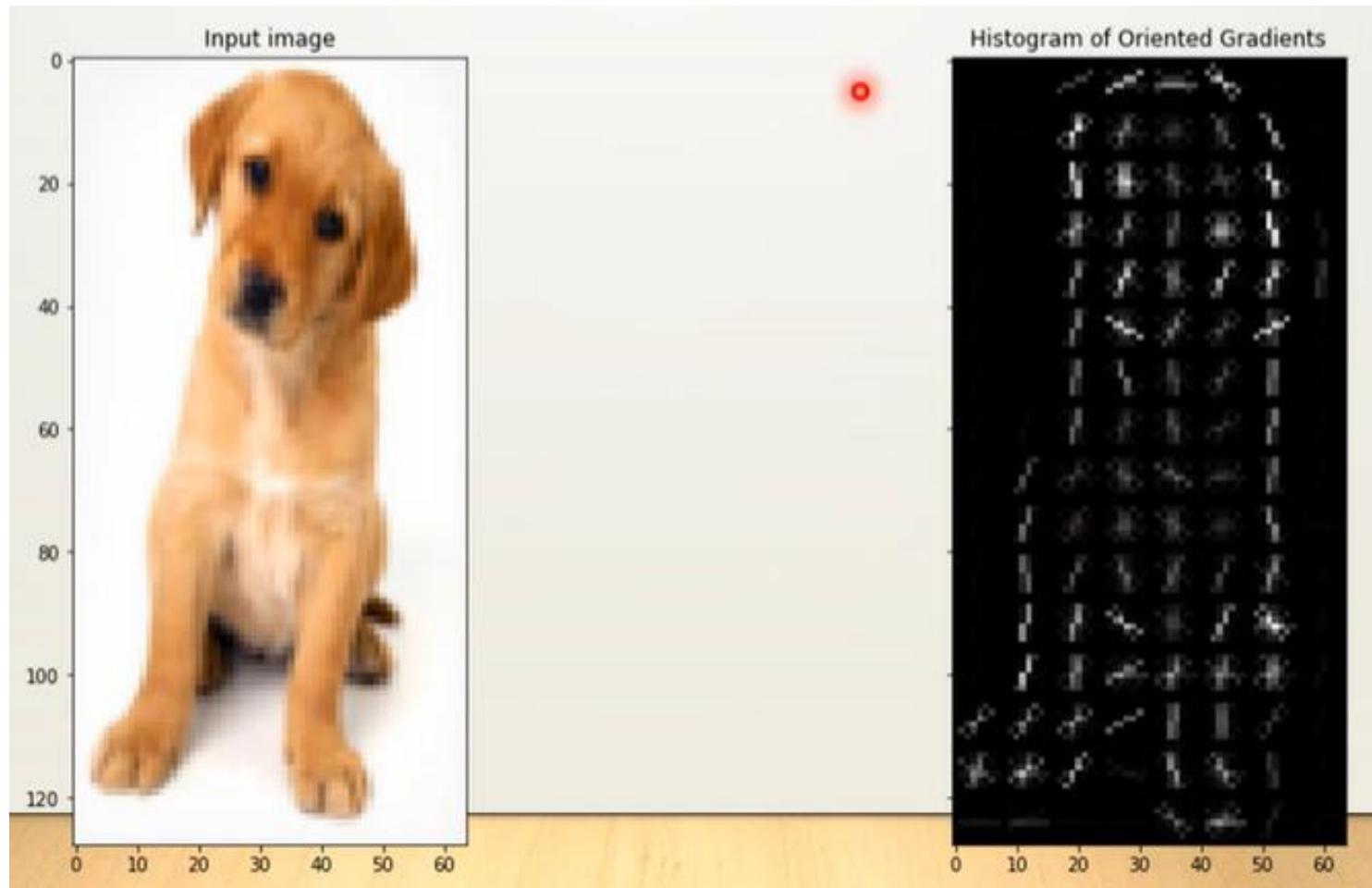


HOG(Histogram of Oriented Gradients)

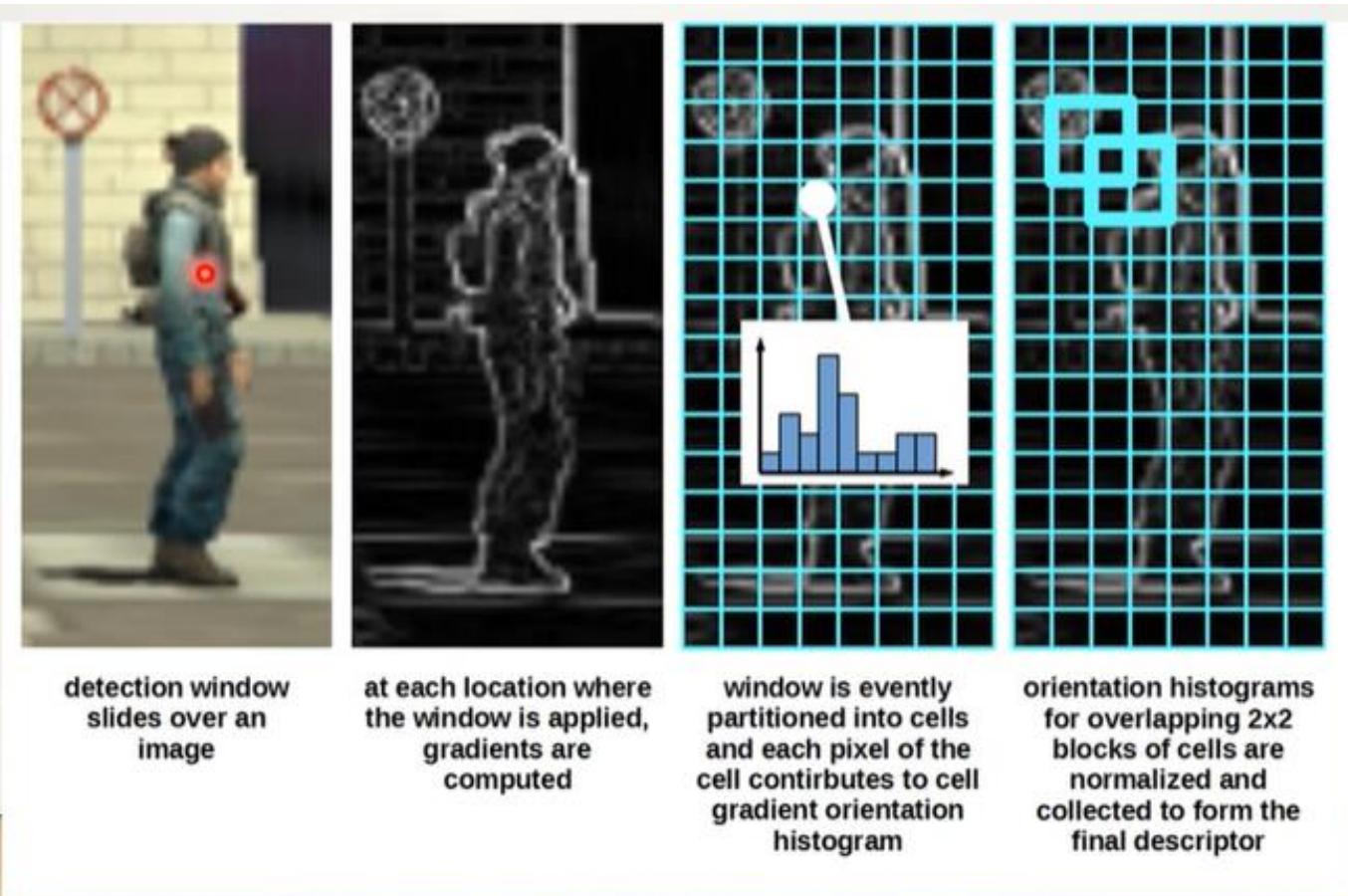
- Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V:
 - $V = [a_1, a_2, a_3, \dots, a_{36}]$
- We calculate the root of the sum of squares:
 - $k = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2 + \dots + (a_{36})^2}$
- And divide all the values in the vector V with this value k:

$$\text{Normalised Vector} = \left(\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \dots, \frac{a_{36}}{k} \right)$$

HOG(Histogram of Oriented Gradients)



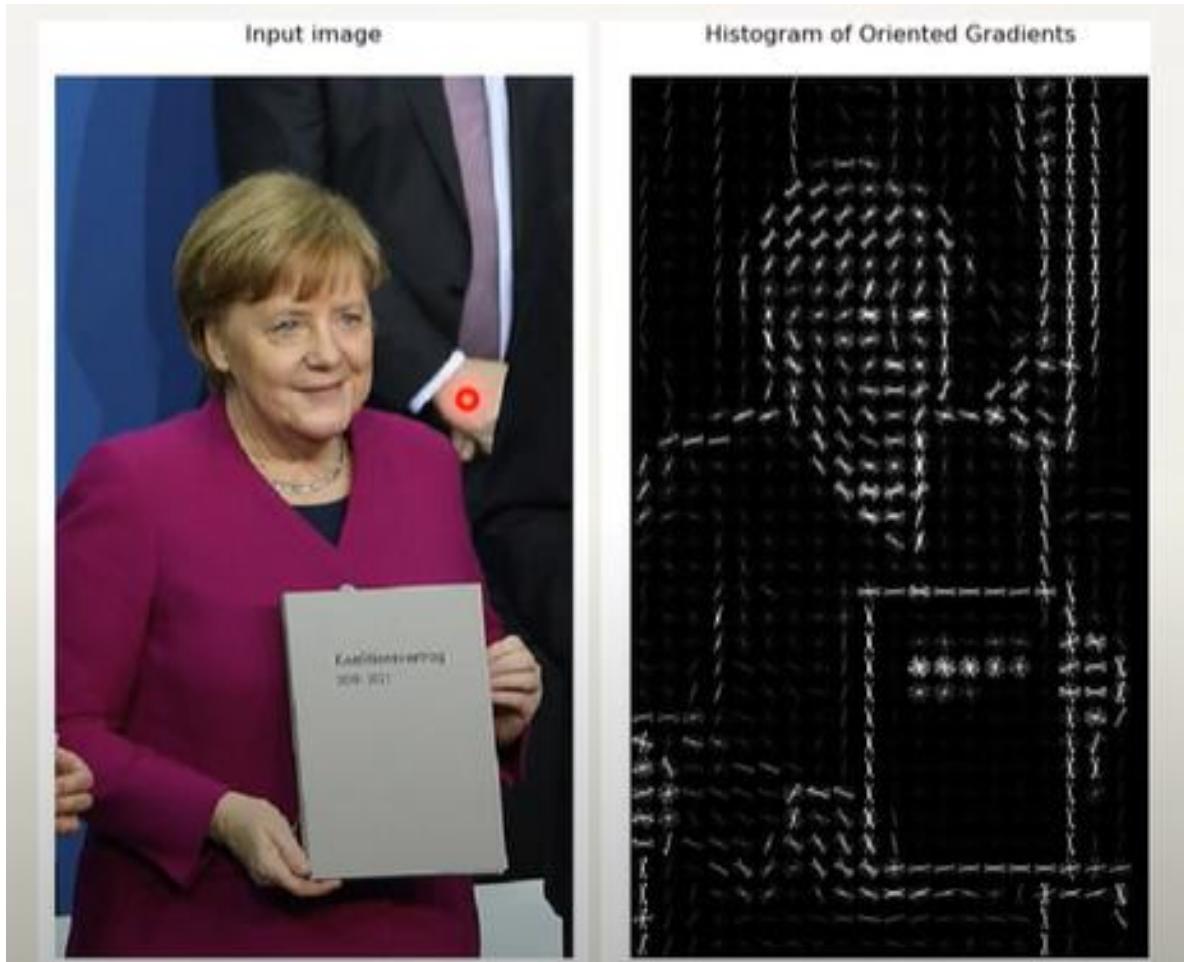
HOG(Histogram of Oriented Gradients)



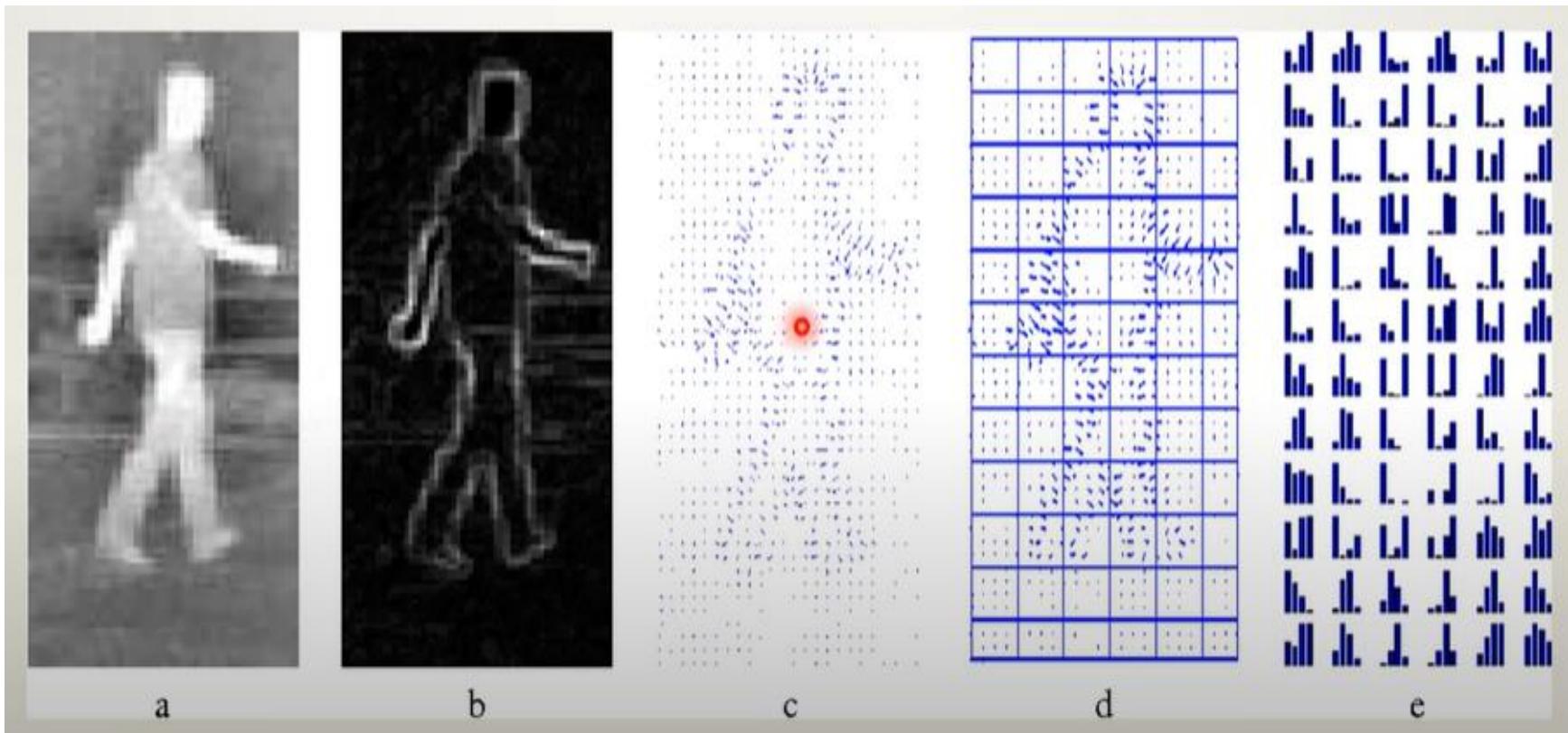
HOG(Histogram of Oriented Gradients)



HOG(Histogram of Oriented Gradients)



HOG(Histogram of Oriented Gradients)



Thank U!!!!

Unit 4

Segmentation

Prof. Janki Patel
Department of IT
SPCE

Outline

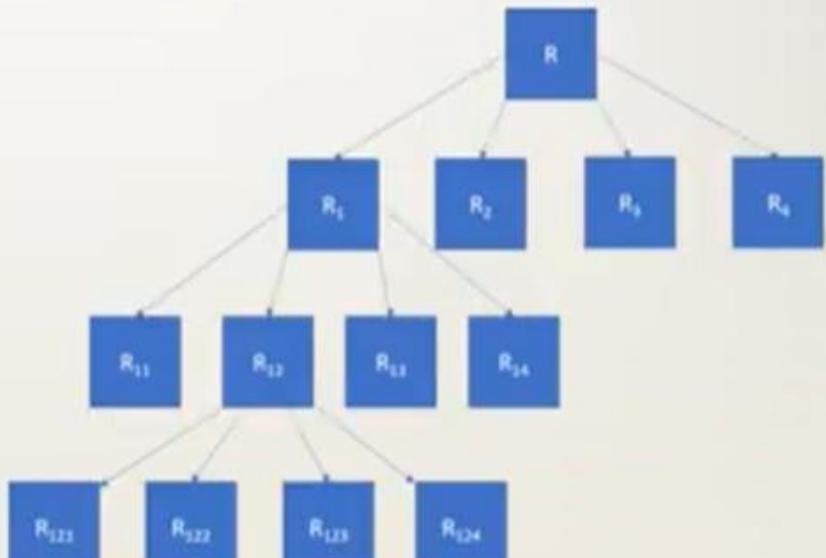
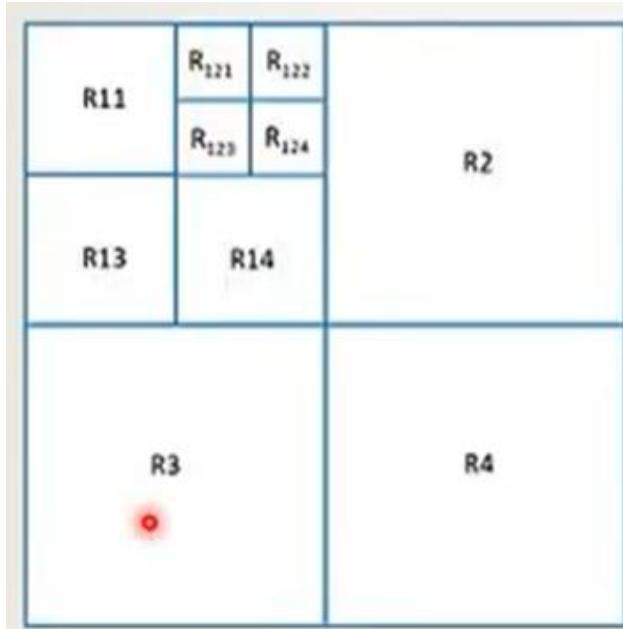
- Active Contours
- Split and Merge
- Watershed
- Region Splitting and Merging
- Graph-based Segmentation
- Mean shift and Model finding
- Normalized Cut

Split and Merge

Split and Merge

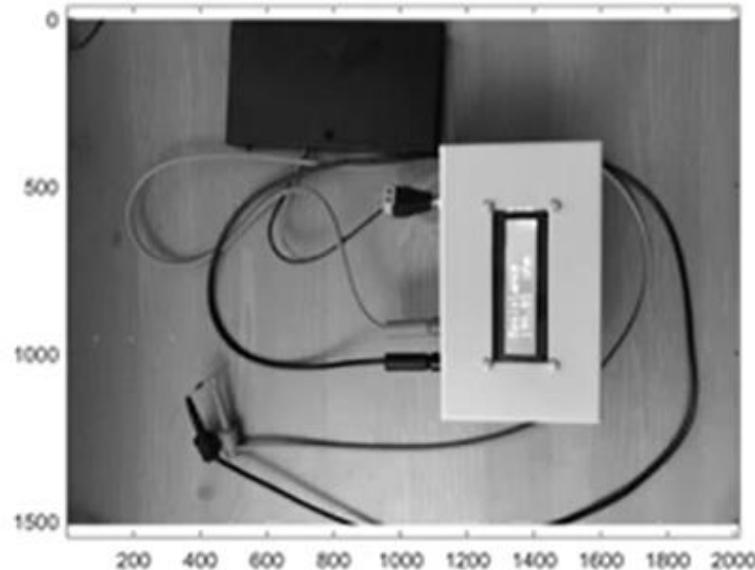
- Split and merge segmentation is an image processing technique used to segment an image. The image is successively split into quadrants based on a homogeneity criterion and similar regions are meaning that there is a parent child node relationship. The total region is a parent, and each of the four splits is a child.

Split and Merge



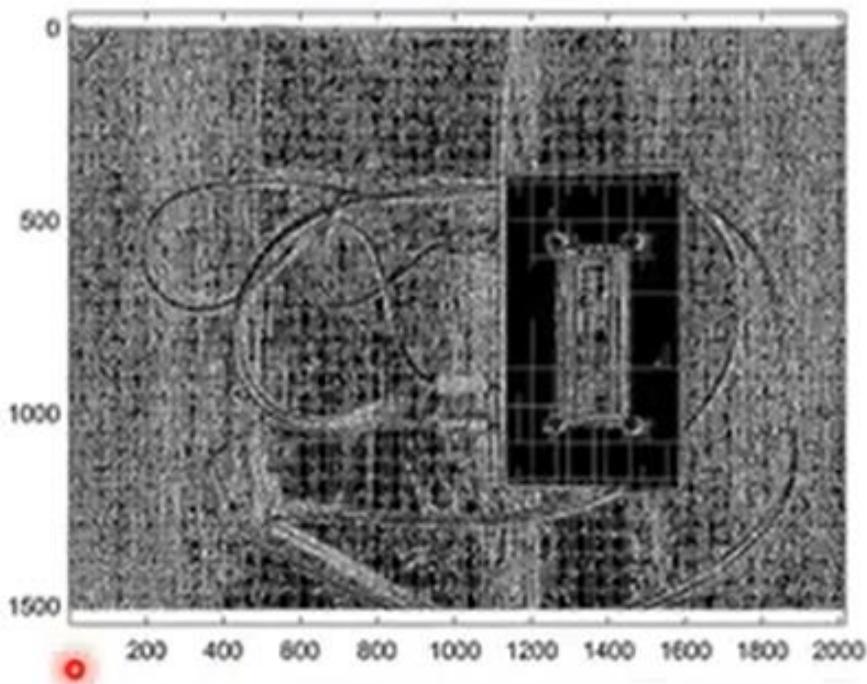
Split and Merge Example

- The following example shows the segmentation of a gray scale image using matlab. The homogeneity criterion is thresholding $\max(\text{region}) - \min(\text{region}) < 10$ for a region to be homogeneous



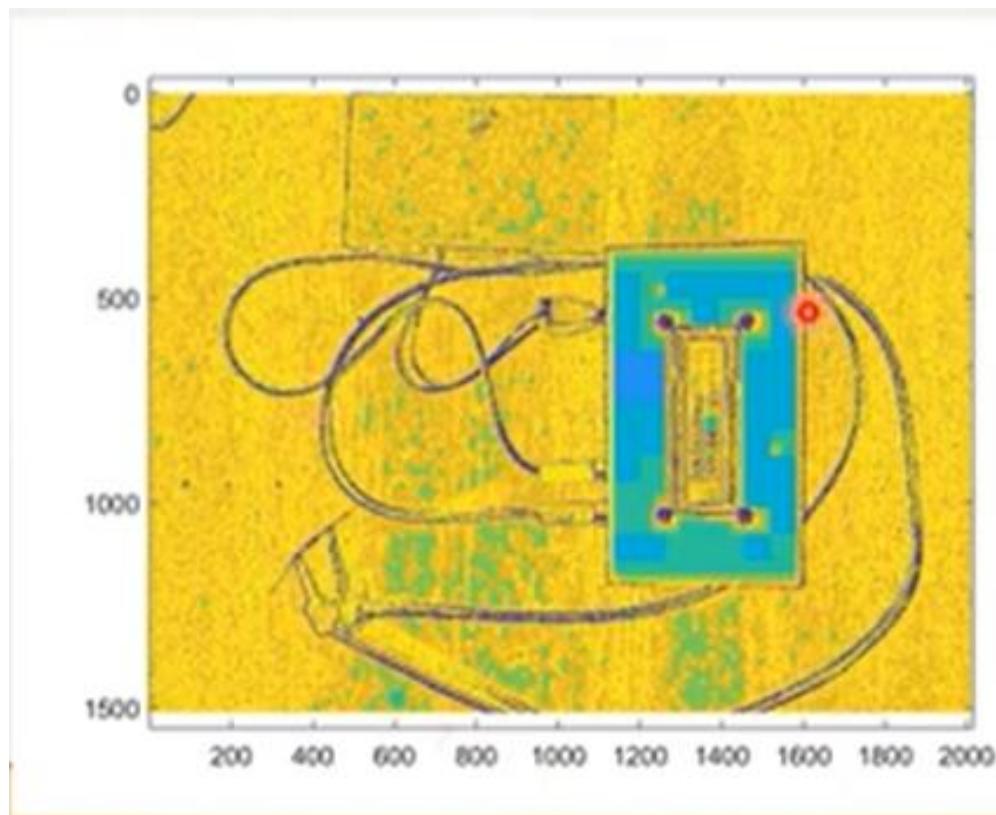
Split and Merge Example

- The blocks created during splitting are shown in the following picture:



Split and Merge Example

- And the segmented image is below.



Region Split and Merge

Region Split and Merge Example

- Apply Region split on following image. Assume that threshold value be ≤ 4 .

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

Region Split and Merge Example

- Step 1: identify Max and Min pixel value from the whole image
- $\text{Max} = 7$
- $\text{Min} = 0$
- $\text{Max-Min} = 7$
- $7 \leq 4$ (condition false)
 - Split image

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

Region Split and Merge Example

- Step 1: Identify Max and Min pixel value from the whole image.
 - Max = 7
 - Min = 0
 - Max-Min=7
 - $7 \leq 4$ (Condition false)
 - Split R image R1a,R2b,R3c,R4d

R1a				R2b				
5	6	6	6	7	7	6	6	6
6	7	6	7	5	5	4	7	7
6	6	4	4	3	2	5	6	6
5	4	5	4	2	3	4	6	6
0	3	2	3	3	2	4	7	7
0	0	0	0	2	2	5	6	6
1	1	0	1	0	3	4	4	4
1	0	1	0	2	3	5	4	4
R3c				R4d				

Region Split and Merge Example

- Step 2: Compute for R1a region. Max and Min
 - Max = 7
 - Min = 4
 - Max-Min=7-4=3
 - $3 \leq 3$ (Condition true)
 - No need to split R1a

R1a				R2b				
5	6	6	6	7	7	6	6	
6	7	6	7	5	5	4	7	
6	6	4	4	3	2	5	6	
5	4	5	4	2	3	4	6	
0	3	2	3	3	2	4	7	
0	0	0	0	2	2	5	6	
1	1	0	1	0	3	4	4	
1	0	1	0	2	3	5	4	
R3c				R4d				

Region Split and Merge Example

- Step 3: Compute for R3c region. Max and Min

- Max = 3
- Min = 0
- Max-Min=3-0=3
- $3 \leq 3$ (Condition true)
- No need to split R3c

R1a				R2b				
5	6	6	6	7	7	6	6	
6	7	6	7	5	5	4	7	
6	6	4	4	3	2	5	6	
5	4	5	4	2	3	4	6	
0	3	2	3	3	2	4	7	
0	0	0	0	2	2	5	6	
1	1	0	1	0	3	4	4	
1	0	1	0	2	3	5	4	
R3c				R4d				

Region Split and Merge Example

- Step 4: Compute for R2b region. Max and Min
 - Max = 7
 - Min = 2
 - Max-Min=7-2=5

R1a					R2b			
5	6	6	6	7	7	7	6	6
6	7	6	7	5	5	5	4	7
6	6	4	4	3	2	2	5	6
5	4	5	4	2	3	3	4	6
0	3	2	3	3	2	2	4	7
0	0	0	0	2	2	2	5	6
1	1	0	1	0	3	3	4	4
1	0	1	0	2	3	3	5	4
R3c					R4d			

Region Split and Merge Example

- Step 4: Compute for R2b region. Max and Min
 - Max = 7
 - Min = 2
 - Max-Min=7-2=5
 - $5 \leq 3$ (Condition true)
 - Split R2b into R2b1, R2b2, R2b3, R2b4

R1a				R2b			
5	6	6	6	7	7	6	6
6	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
0	3	2	3	3	2	4	7
0	0	0	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4
R3c				R4d			

Region Split and Merge Example

- Step 5: Compute for R2b1, R2b2, R2b3, R2b4 region. Max and Min
- For all R2b1, R2b2, R2b3, R2b4
- Condition have satisfied so no need to splitting

R1a				R2b				
5	6	6	6	7	7	6	6	
6	7	6	7	5	5	4	7	
6	6	4	4	3	2	5	6	
5	4	5	4	2	3	4	6	
0	3	2	3	3	2	4	7	
0	0	0	0	2	2	5	6	
1	1	0	1	0	3	4	4	
1	0	1	0	2	3	5	4	
				R3c				

Region Split and Merge Example

- Step 6: Compute for R4d region. Max and Min
 - Max = 7
 - Min = 0
 - Max-Min=7-0=7
 - $7 \leq 3$ (Condition false)
 - Split R4d into R4b1, R4b2, R4b3, R4b4

R1a				R2b				
5	6	6	6	7	7	6	6	
6	7	6	7	5	5	4	7	
6	6	4	4	3	2	5	6	
5	4	5	4	2	3	4	6	
0	3	2	3	3	2	4	7	
0	0	0	0	2	2	5	6	
1	1	0	1	0	3	4	4	
1	0	1	0	2	3	5	4	

R3c R4d

Region Split and Merge Example

- Step 7: Compute for R4b1, R4b2, R4b3, R4b4 region. Max and Min
- For all R4b1, R4b2, R4b3, R4b4
- Condition have satisfied so no need to splitting

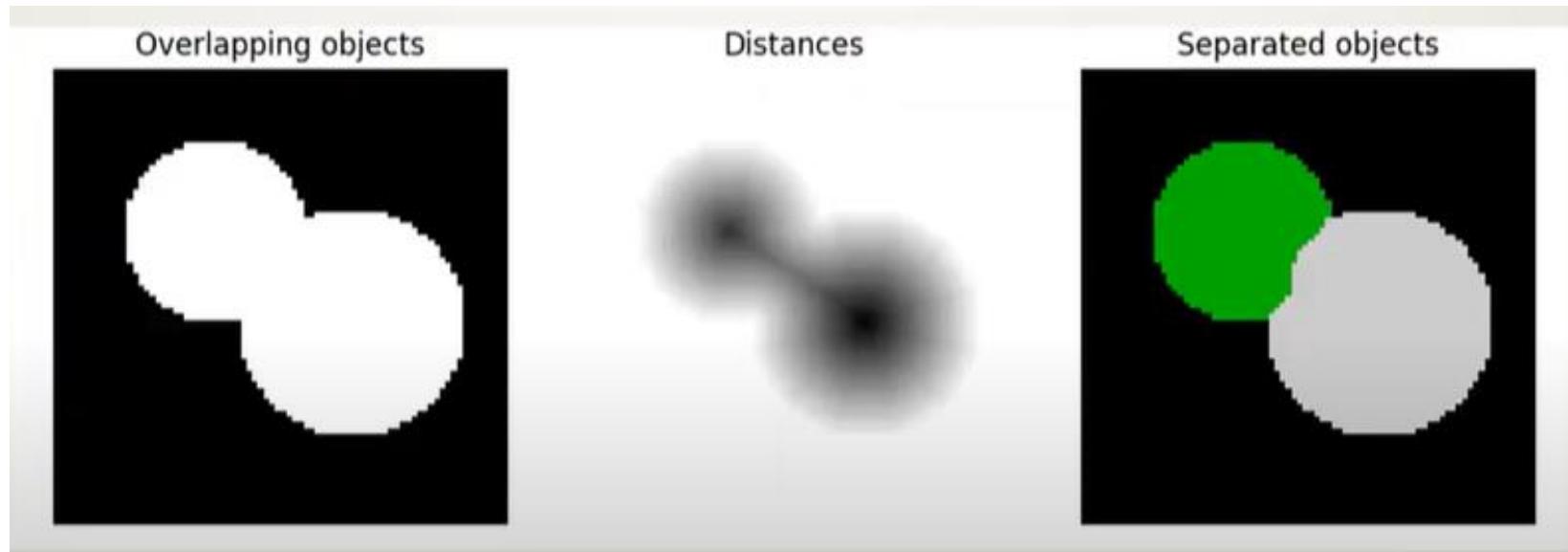
R1a				R2b				
5	6	6	6	7	7	6	6	
6	7	6	7	5	5	4	7	
6	6	4	4	3	2	5	6	
5	4	5	4	2	3	4	6	
0	3	2	3	3	2	4	7	
0	0	0	0	2	2	5	6	
1	1	0	1	0	3	4	4	
1	0	1	0	2	3	5	4	
R3c				R4d				

Watershed Segmentation

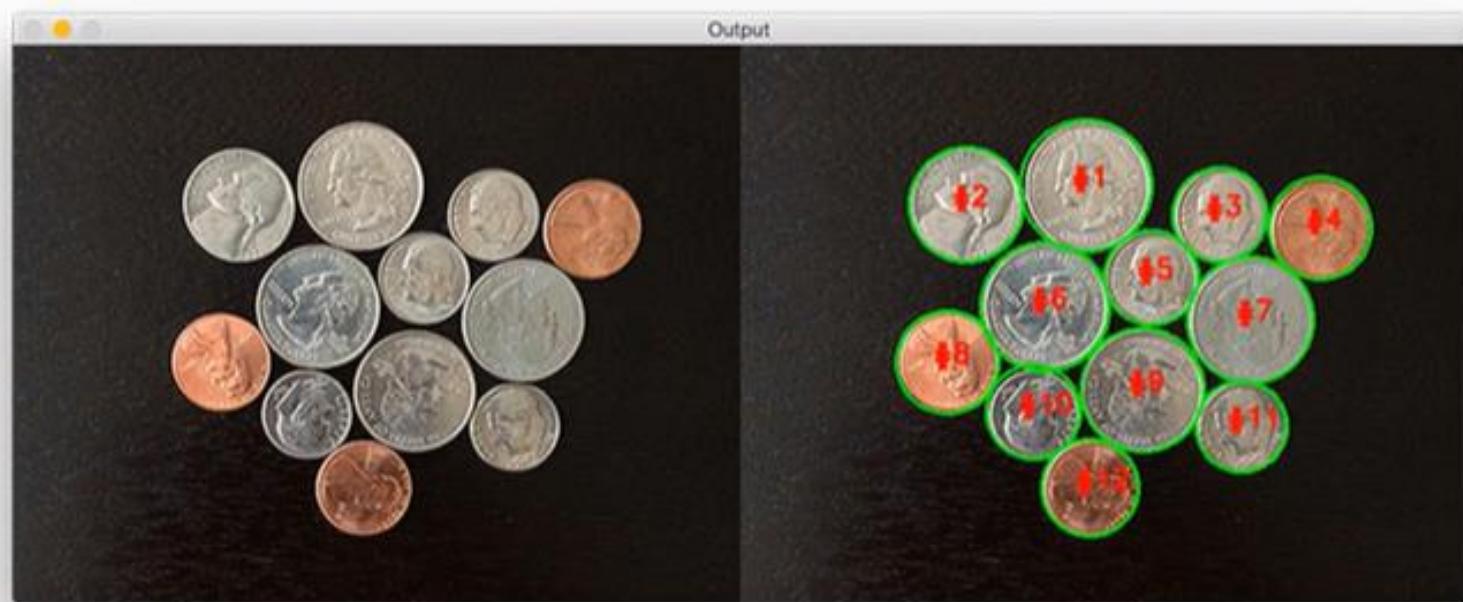
Watershed Segmentation

- The watershed algorithm is a classic algorithm used for segmentation and is especially used when extracting touching or overlapping objects in images, such as the coins in the figure above.
- Using traditional image processing methods such as thresholding and contour detection, we would be unable to extract each individual coin from the image but by leveraging the watershed algorithm, we are able to detect and extract each coin without a problem.
- When utilizing the watershed algorithm we must start with user-defined markers. These markers can be either manually defined via point-and-click, or we can automatically define them using methods such as thresholding and morphological operations

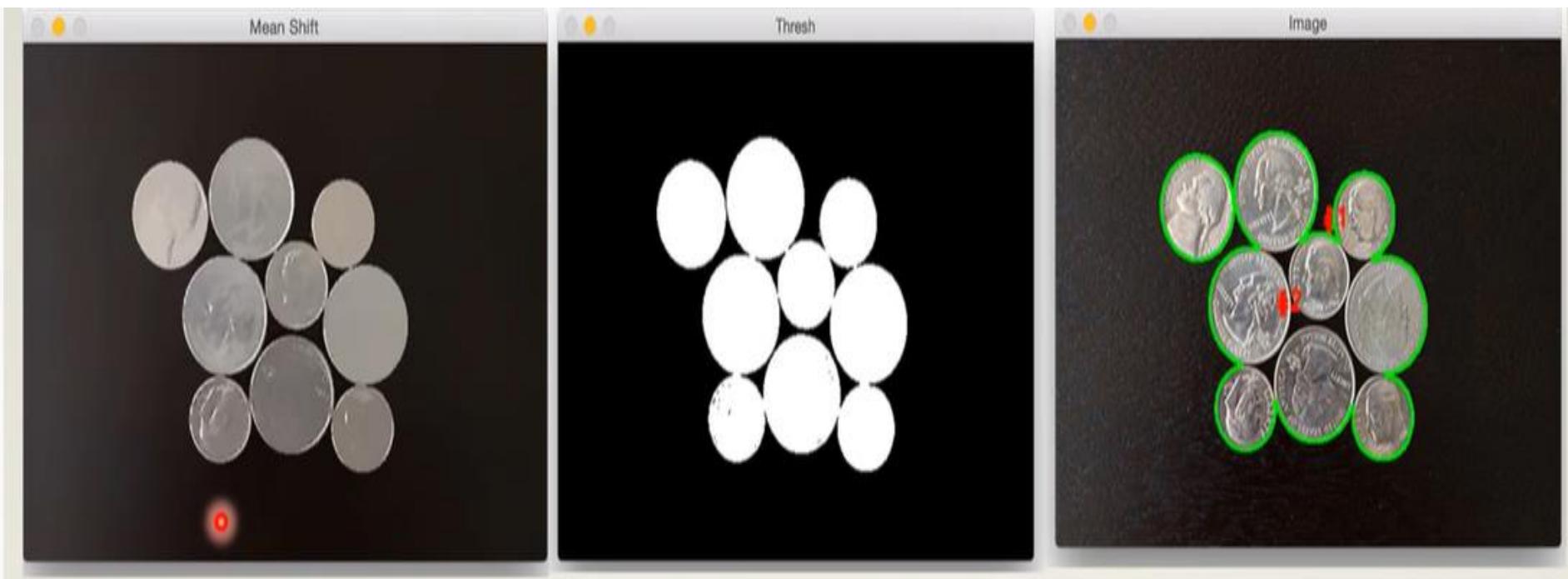
Watershed Segmentation



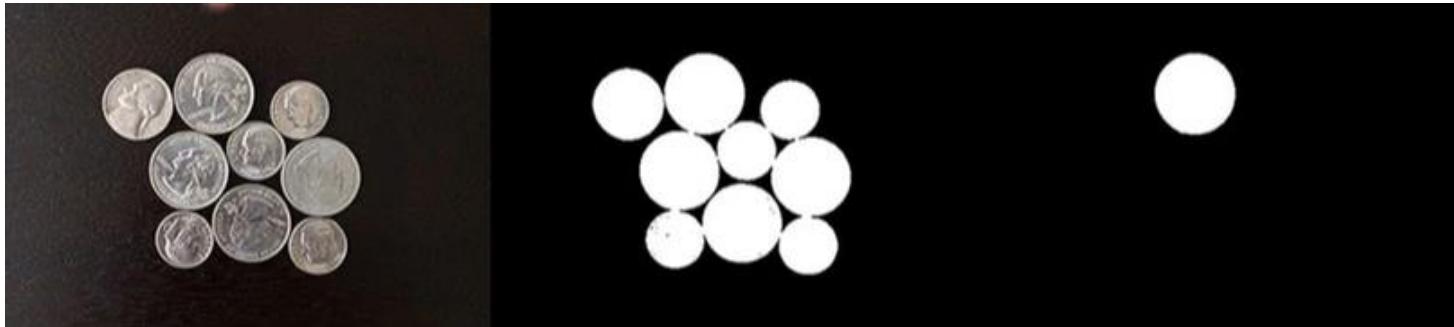
Watershed Segmentation



The problem with basic thresholding and contour extraction



Using the watershed algorithm for segmentation

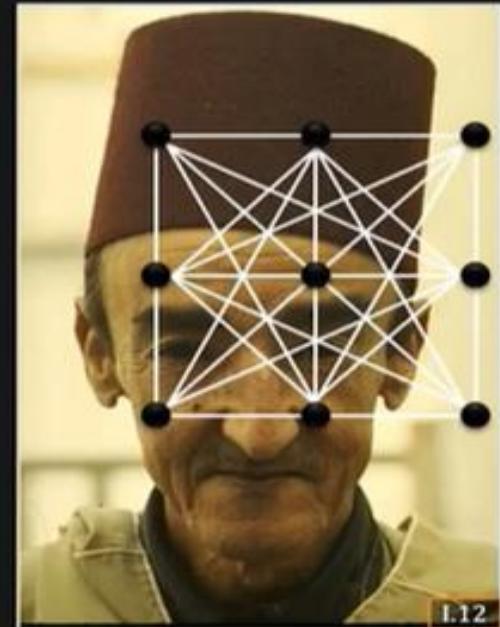


Graph based Segmentation

Graph based segmentation

Images as Graphs:

- A vertex for each pixel.
- An edge between each pair of pixels.
- Graph Notation: $G = (V, E)$ where V and E are the sets of vertices and edges, respectively.
- Each edge is weighted by the **affinity** or similarity between its two vertices.



Input Image

Measuring Affinity

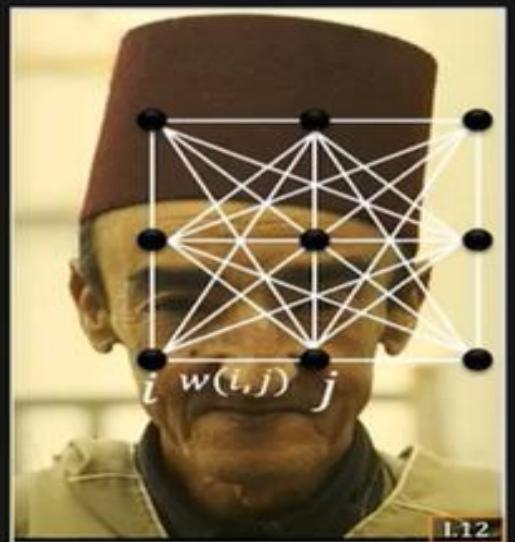
Let i and j be two pixels whose features are \mathbf{f}_i and \mathbf{f}_j .

Pixel Dissimilarity:

$$S(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{\left(\sum_k (f_{ik} - f_{jk})^2 \right)}$$

Pixel Affinity:

$$w(i, j) = A(\mathbf{f}_i, \mathbf{f}_j) = e^{\left\{ \frac{-1}{2\sigma^2} S(\mathbf{f}_i, \mathbf{f}_j) \right\}}$$



Input Image

Smaller the Dissimilarity, Larger the Affinity

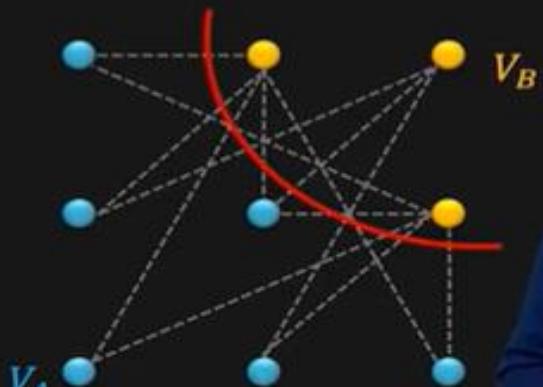
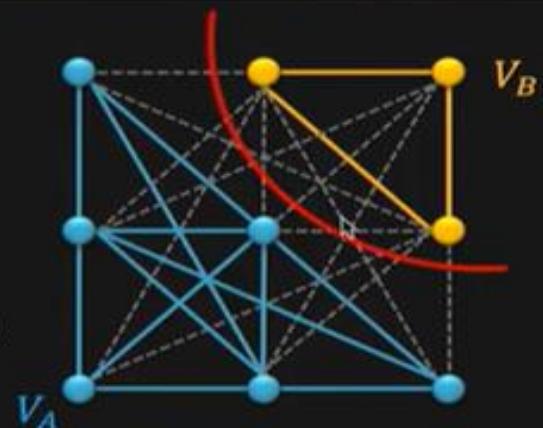
Graph Cut Segmentation

Cut $C = (V_A, V_B)$ is a partition of vertices V of a graph $G = (V, E)$ into two disjoint subsets V_A and V_B .

Cut-Set: Set of edges whose vertices are in different subsets of partition.

Cost of Cut: Sum of weights of cut-set edges.

$$cut(V_A, V_B) = \sum_{u \in V_A, v \in V_B} w(u, v)$$



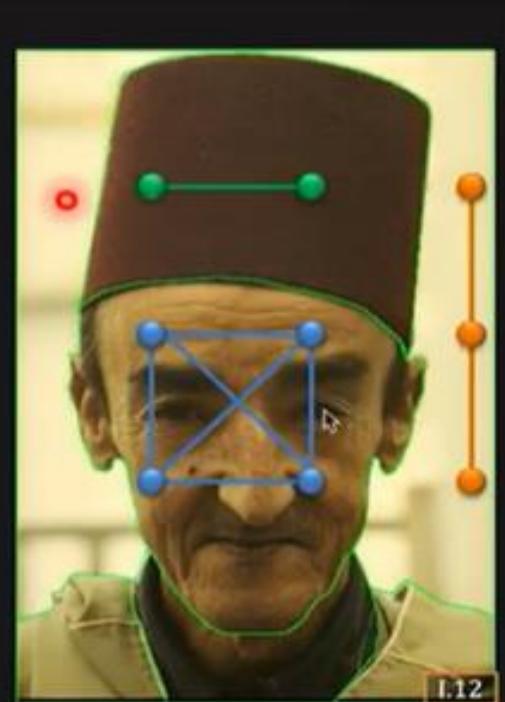
Graph Cut Segmentation

Criteria for Graph Cut:

- A pair of vertices (pixels) within a subgraph have **high affinity**.
- A pair of vertices from two different subgraphs have **low affinity**.

That is, minimize the cost of cut.

Also called **Min-Cut**.

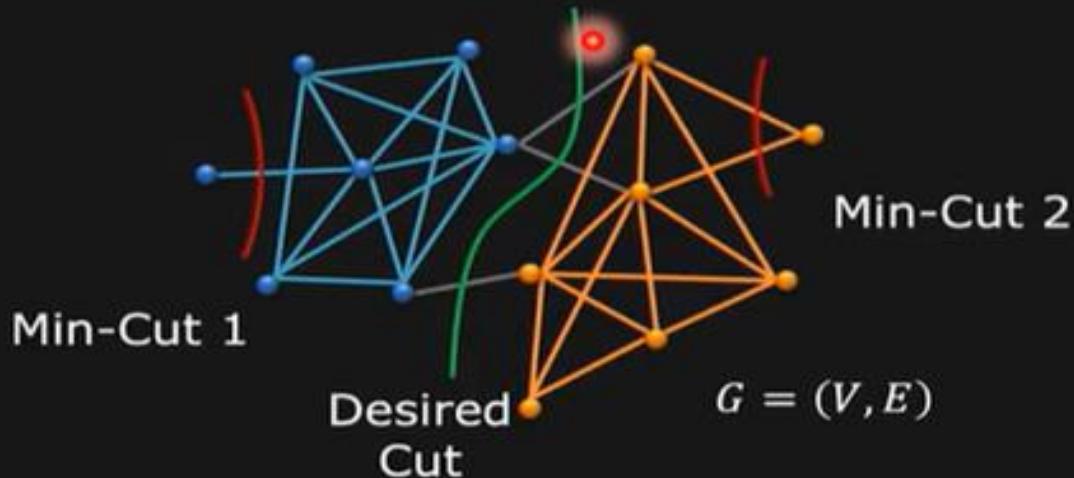


Input Image

Each subgraph is an image segment.

Problem with Min-Cut

There is a bias to cut small, isolated segments.

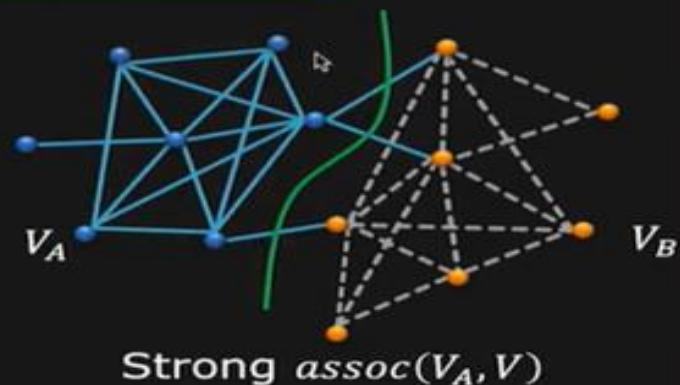
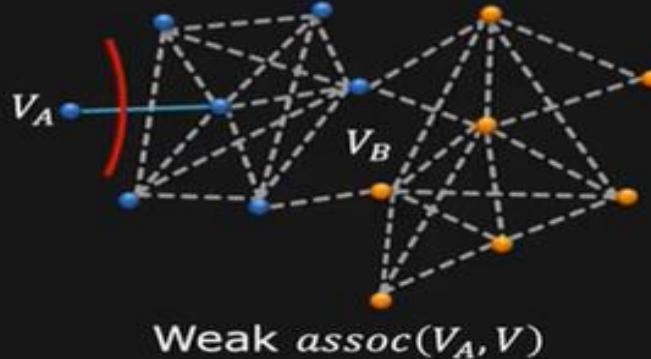


Solution: Normalize Cut to favor larger subgraphs.

Measure of Subgraph Size

Compute how strongly vertices V_A are associated with vertices V .

$$assoc(V_A, V) = \sum_{u \in V_A, v \in V} w(u, v)$$



$assoc()$ is the sum of the weights of the solid edges

Normalized Cut(Ncut)

Minimize Cost of Normalized Cut during Partition

$$NCut(V_A, V_B) = \frac{cut(V_A, V_B)}{assoc(V_A, V)} + \frac{cut(V_A, V_B)}{assoc(V_B, V)}$$



Minimizing NCut has no known polynomial time solution.
It is NP-Complete.

Fast eigenvector-based approximations exist [Shi 2000].

Ncut Segmentation Results

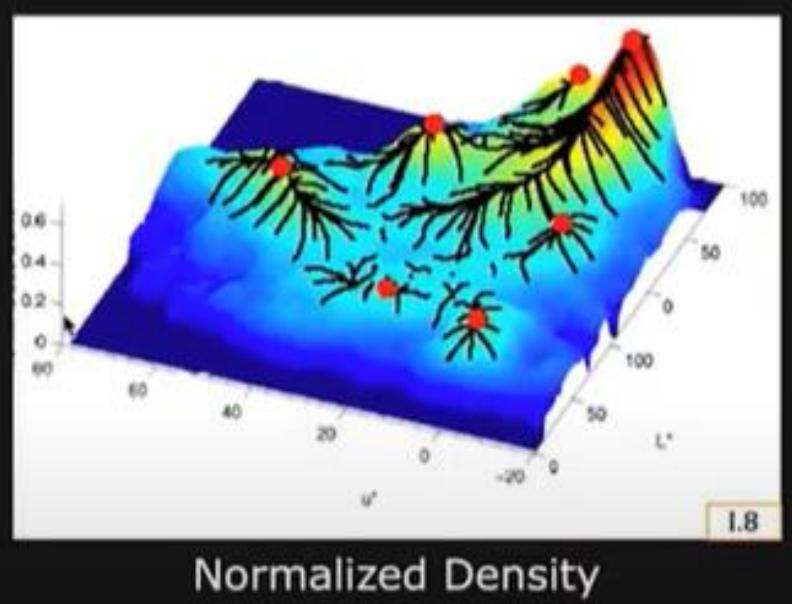
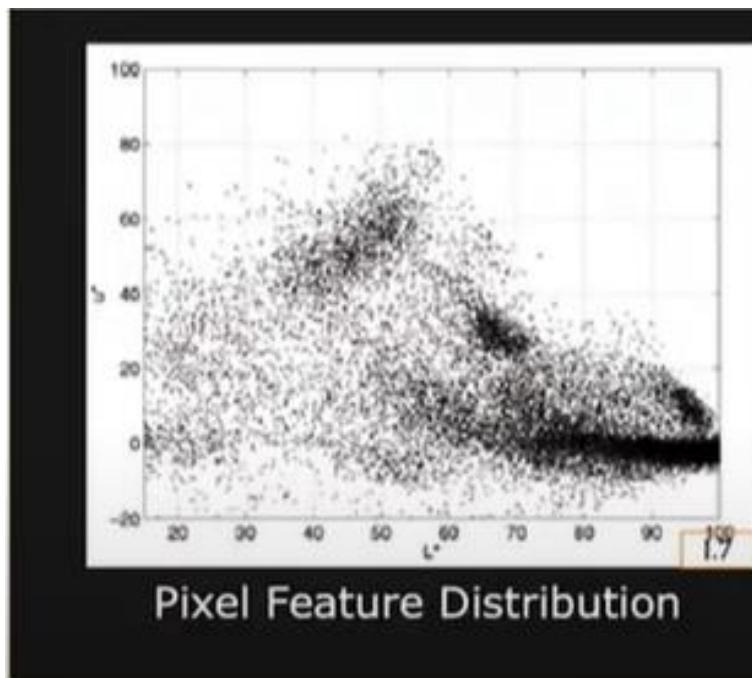


Segmented Images

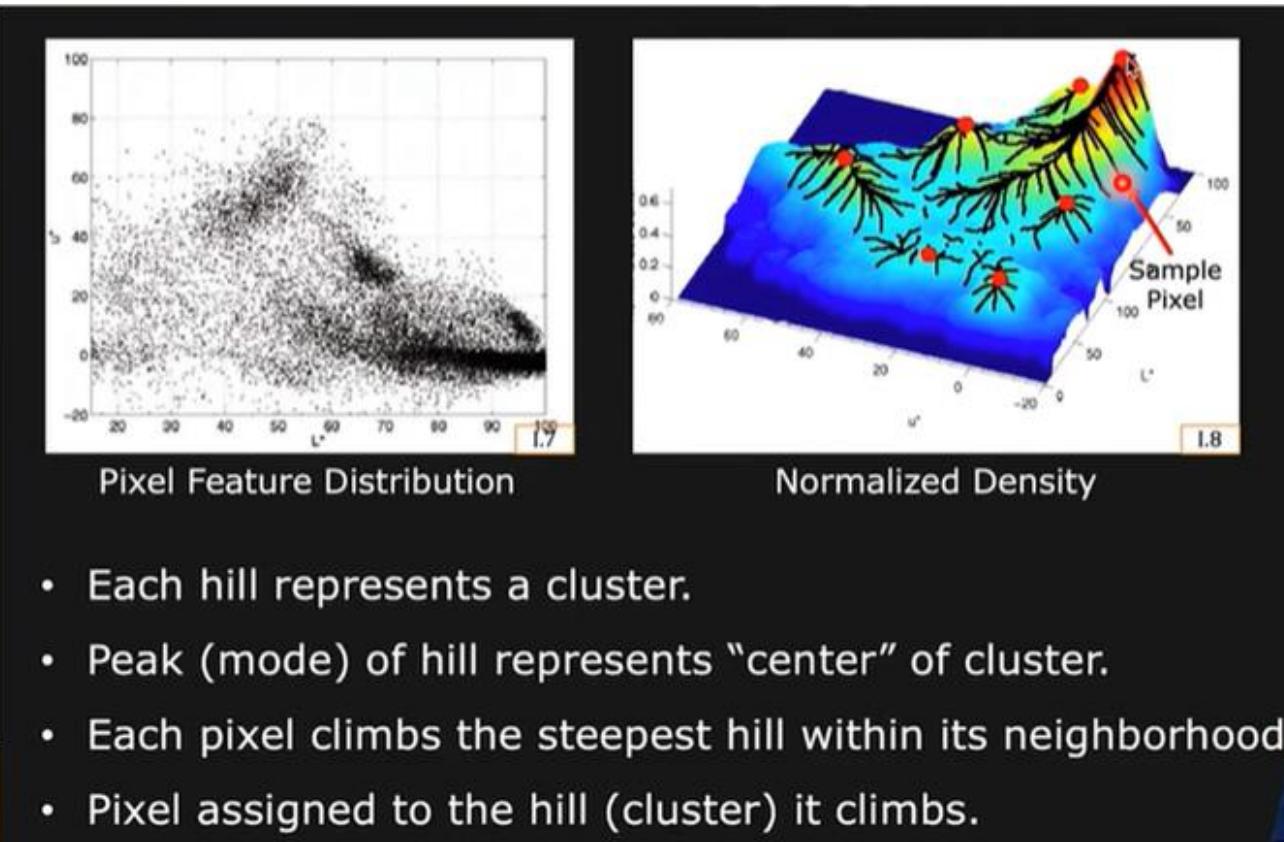
Pixel Feature: {*Brightness, Location*}

The Concept of Mean Shift

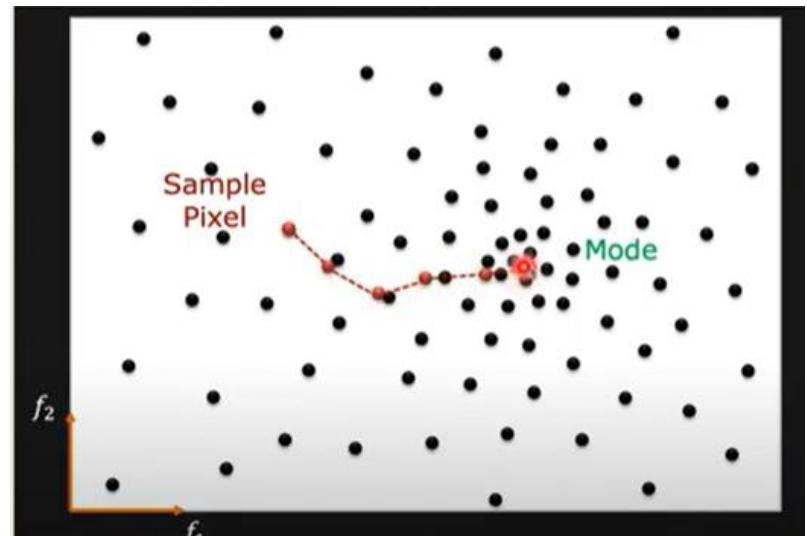
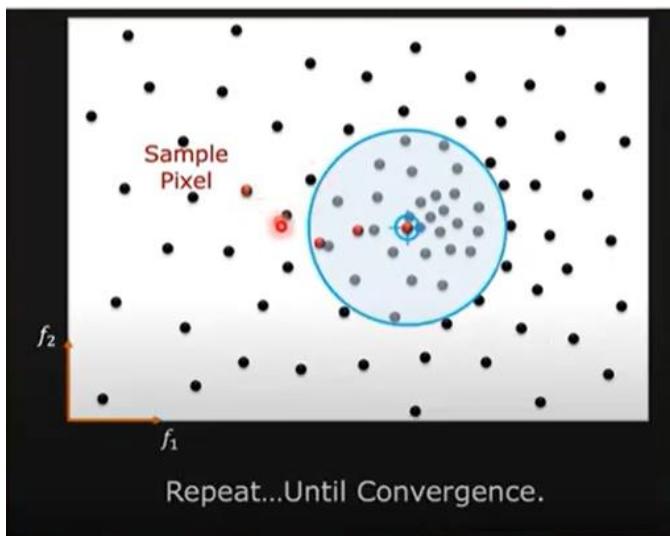
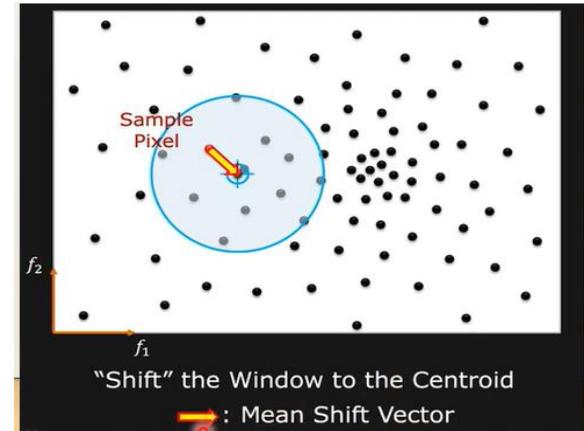
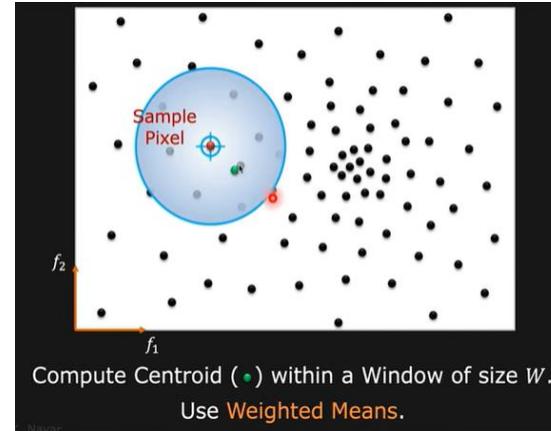
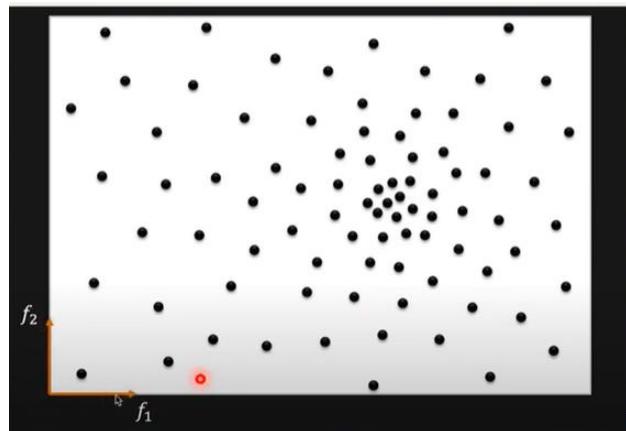
The Concept of Mean Shift



The Concept of Mean Shift



Hill Climbing using Mean Shift



Mean Shift Algorithm

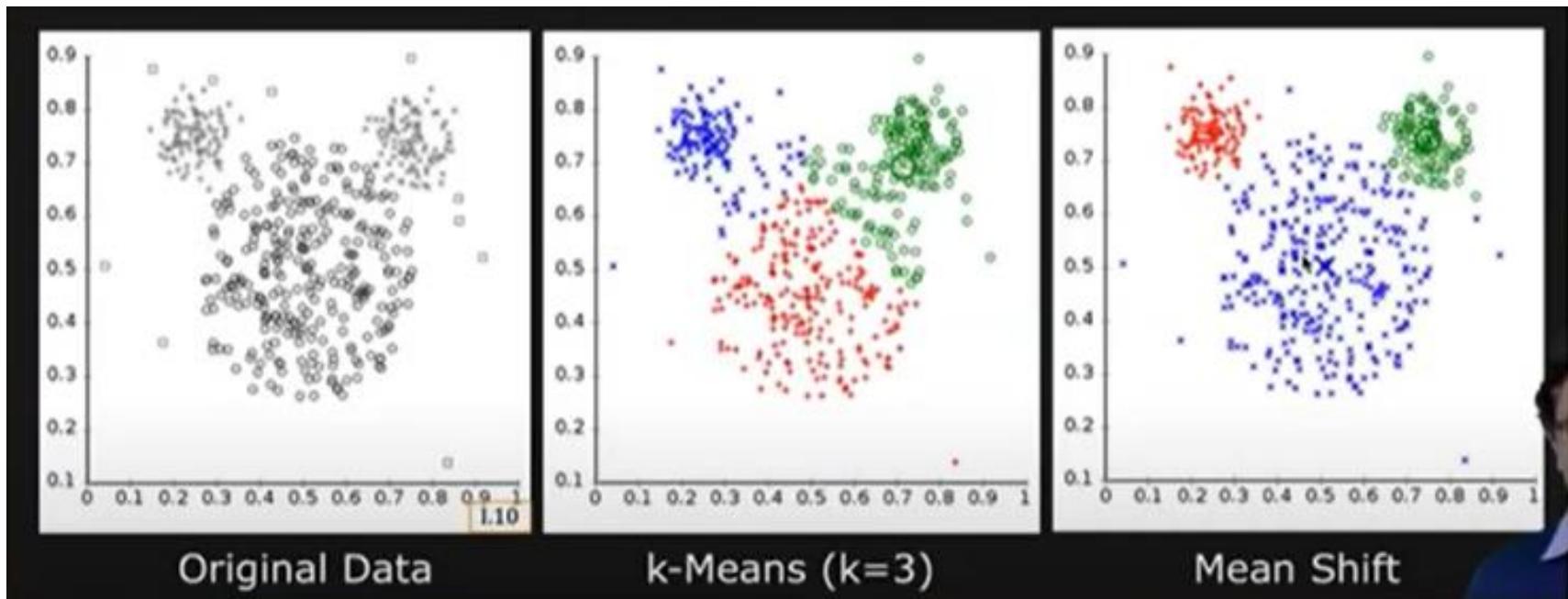
Given: Distribution of N pixels in feature space.

Task: Find modes (clusters) of distribution.

Clustering:

- 1: Set $\mathbf{m}_i = \mathbf{f}_i$ as initial mean for each pixel i .
- 2: Repeat the following for each mean \mathbf{m}_i :
 - a: Place window of size W around \mathbf{m}_i .
 - b: Compute centroid \mathbf{m} within the window. Set $\mathbf{m}_i = \mathbf{m}$.
 - c: Stop if shift in mean \mathbf{m}_i is less than a threshold ε .
 \mathbf{m}_i is the mode.
- 3: Label all pixels that have same mode as belonging to same cluster.

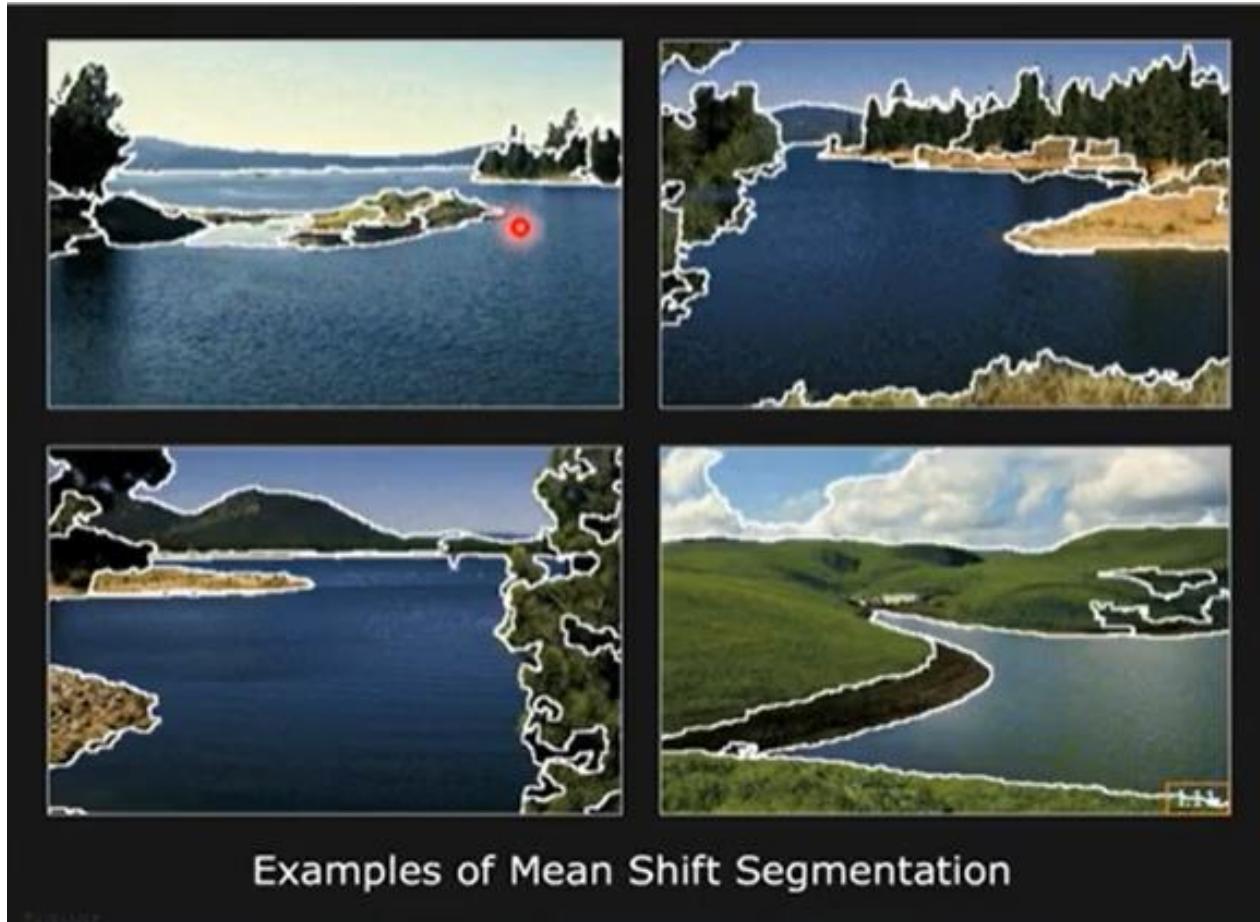
K-mean Vs. Mean Shift



Example K-mean Vs. Mean Shift



Result of Mean Shift Segmentation



Thank U!!!

Unit 5

Camera Calibration

Prof. Janki Patel
Department of IT
SPCE

Camera Models

The Computer Vision Toolbox™ contains calibration algorithms for the pinhole camera model and the fisheye camera model.

You can use the fisheye model with cameras up to a field of view (FOV) of 195 degrees.

- Pinhole camera model
- Fisheye camera model
- Orthographic projection
- Scaled orthographic projection
- Paraperspective projection
- Perspective projection

Fisheye Camera Models

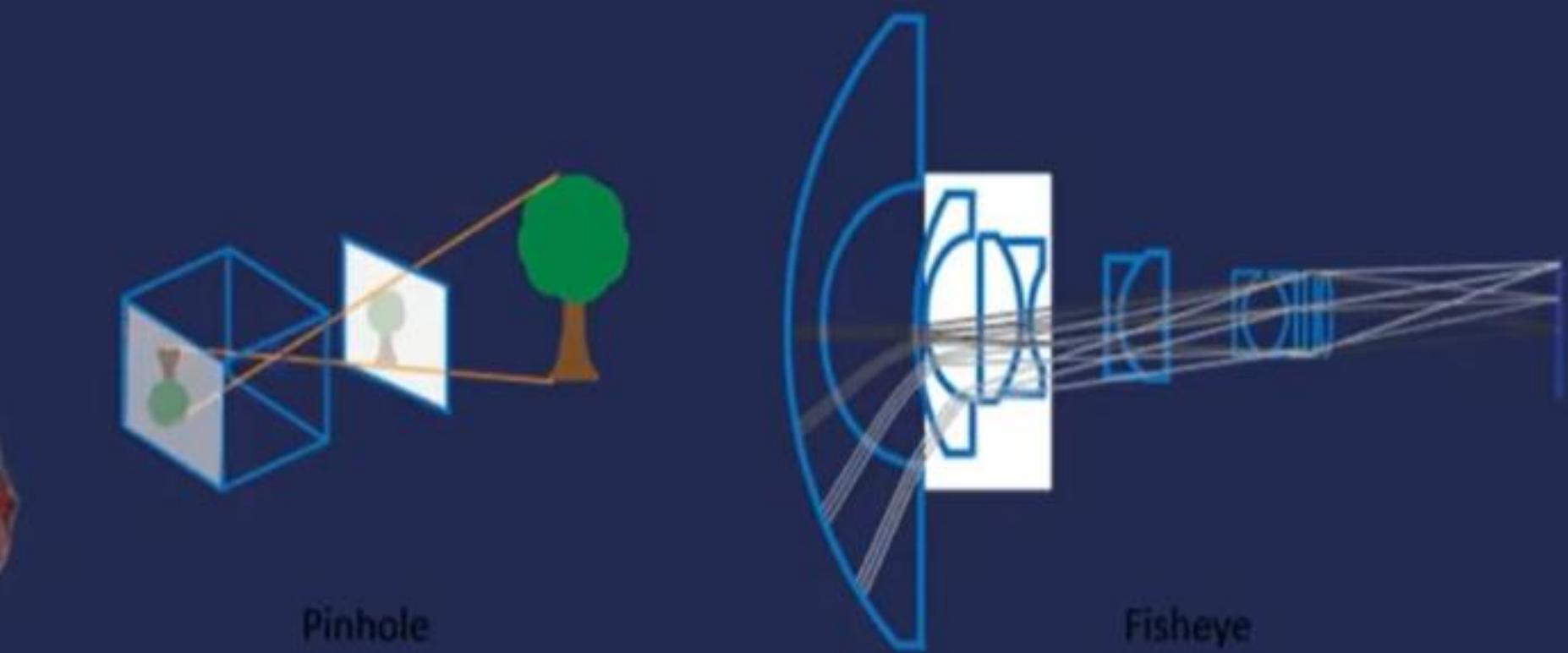
Fisheye cameras are used in odometry and to solve the simultaneous localization and mapping (SLAM) problems visually.

Other applications include, surveillance systems, GoPro, virtual reality (VR) to capture 360 degree field of view (fov), and stitching algorithms.

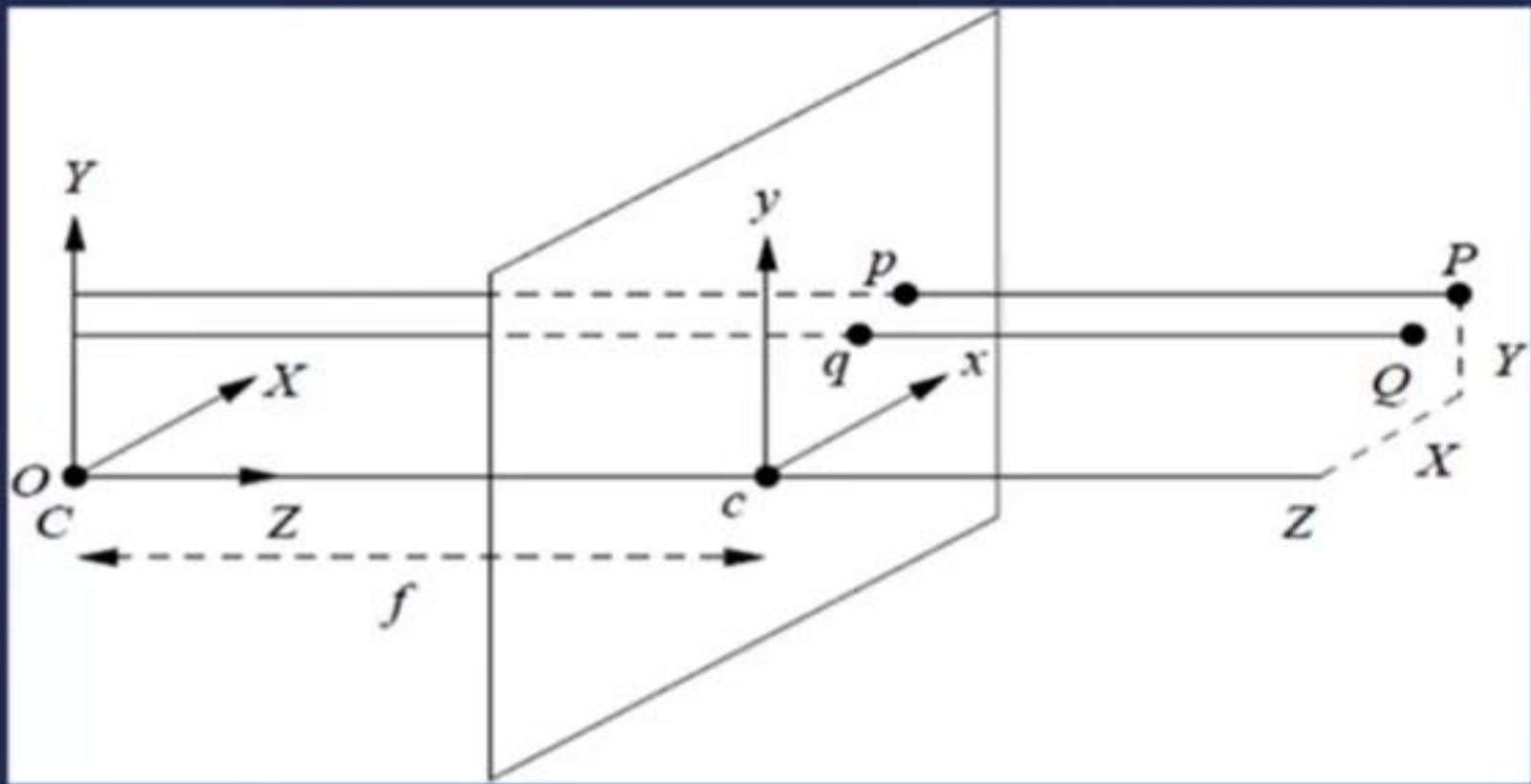
These cameras use a complex series of lenses to enlarge the camera's field of view, enabling it to capture wide panoramic or hemispherical images. However, the lenses achieve this extremely wide angle view by distorting the lines of perspective in the images

Fisheye Camera Models

Because of the extreme distortion a fisheye lens produces, the pinhole model cannot model a fisheye camera.

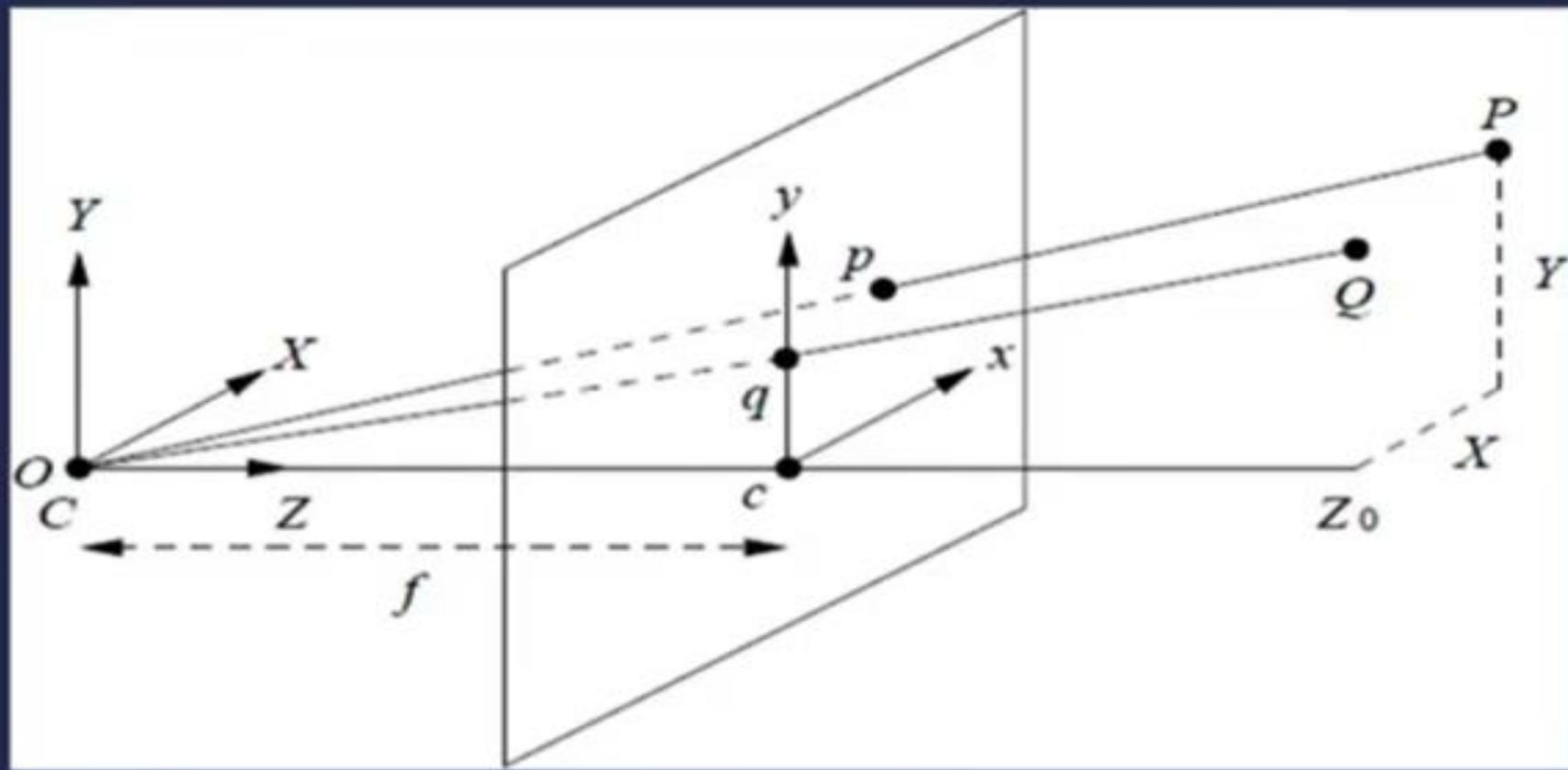


Orthographic Projection



- 3D scene is at infinite distance from camera.
- All projection lines are parallel to optical axis.
- So, $x = X, \quad y = Y$

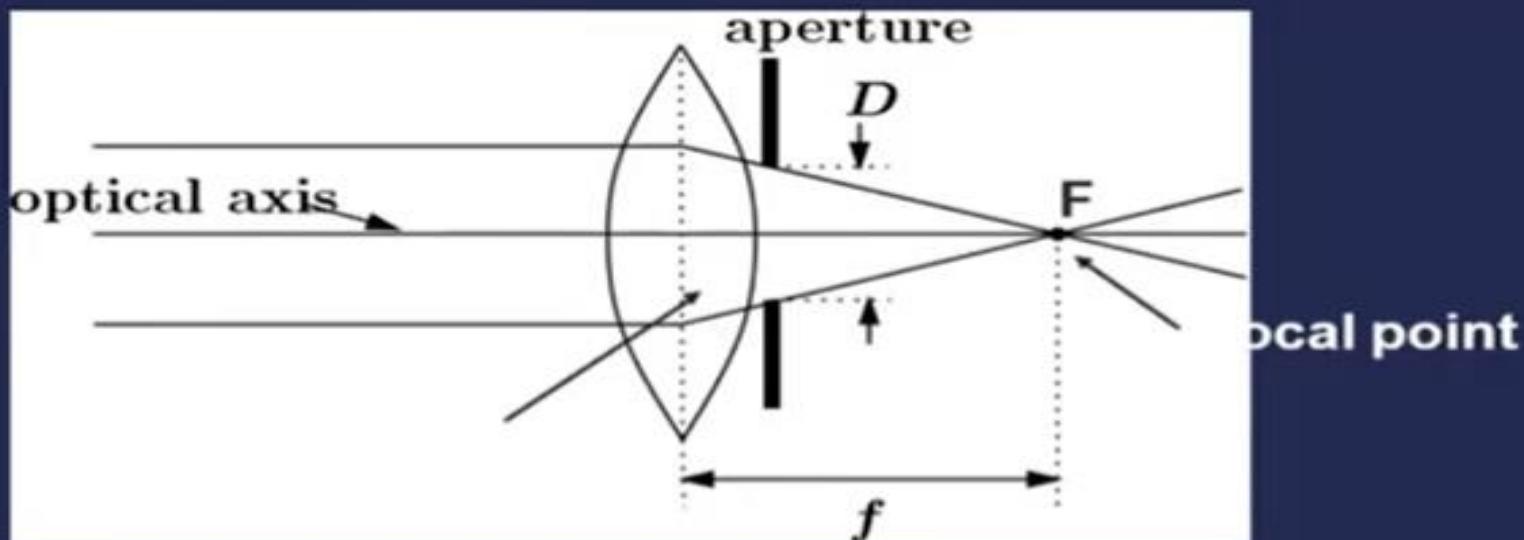
Scaled Orthographic Projection



- Scene depth \ll distance to camera.
- Z is the same for all scene points, say Z_0

$$x = sX, \quad y = sY, \quad s = \frac{f}{Z_0} \text{ for all scene points.}$$

Lenses

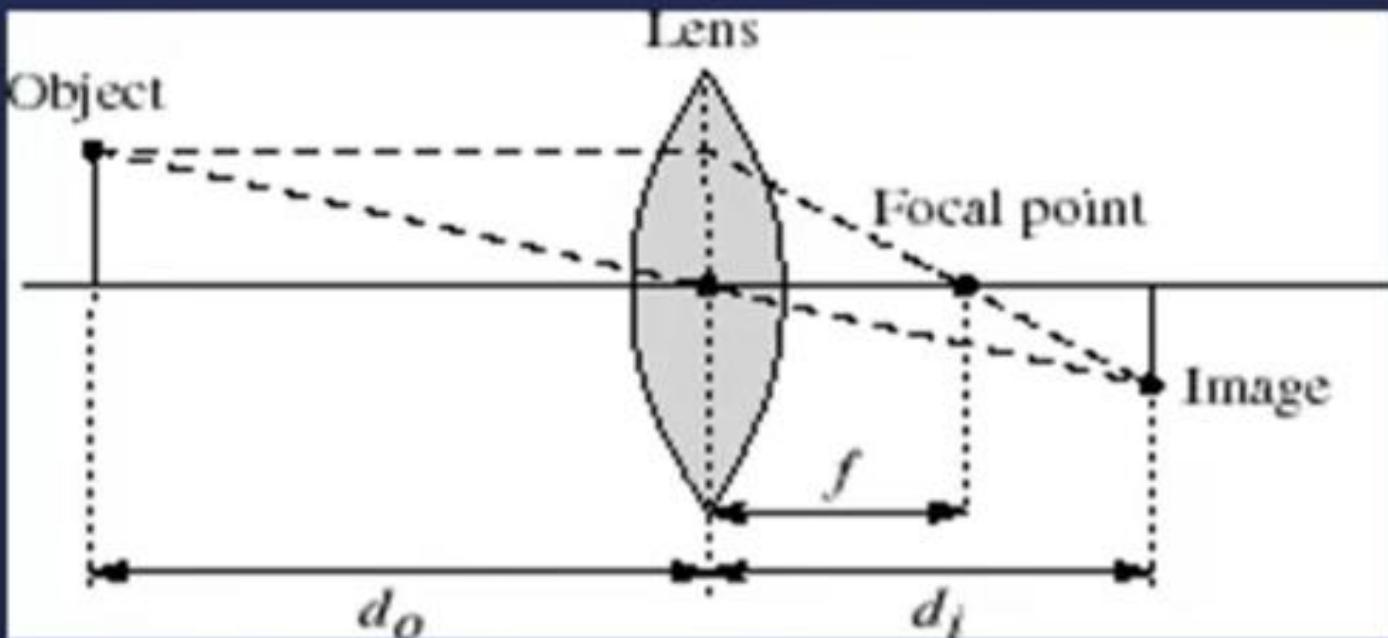


A lens focuses parallel rays onto a single focal point focal point at a distance f beyond the plane of the lens f is a function of the shape and index of refraction of the lens

Aperture of diameter D restricts the range of rays aperture may be on either side of the lens.

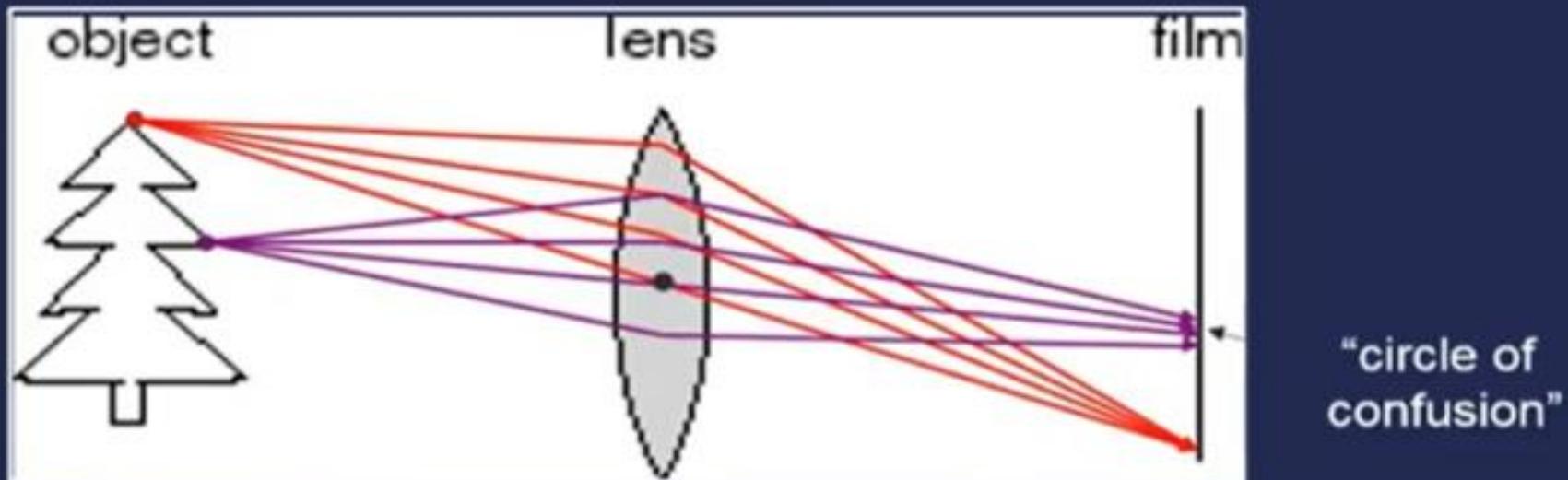
Lenses used to be typically spherical but now many “aspherical” elements – designed to improve variety of “aberrations”...

Thin lenses



- Thin lens equation: $\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$
- Any object point satisfying this equation is in focus
- What is the shape of the focus region?
- How can we change the focus region?

What's in focus and what's not?



- A lens focuses light onto the film
 - There is a specific distance at which objects are “in focus”
 - other points project to a “circle of confusion” in the image
 - Aside: could actually compute distance from defocus Changing the shape or relative locations of the lens elements changes this distance

Extrinsic Parameters

The extrinsic parameters represent a rigid transformation from 3-D world coordinate system to the 3-D camera's coordinate system.

The extrinsic parameters consist of a rotation, R , and a translation, t .

The origin of the camera's coordinate system is at its optical center and its x- and y-axis define the image plane.

Extrinsic Parameters

Camera frame is not aligned with world frame.

- Rigid transformation between them:

$${}^C \mathbf{X} = {}^W_R {}^W \mathbf{X} + {}^W_T$$

frame

object

- Coordinates of 3D scene point in camera frame.
- Coordinates of 3D scene point in world frame.
- Rotation matrix of world frame in camera frame.
- Position of world frame's origin in camera frame.

Extrinsic Parameters

- Translation matrix:

$$\mathbf{T} = \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

- Rotation matrix:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Intrinsic Parameters

The intrinsic parameters represent a projective transformation from the 3-D camera's coordinates into the 2-D image coordinates.

Pinhole camera model:

$$x = f \frac{X}{Z}, \quad y = f \frac{Y}{Z}$$

Camera sensor's pixels not exactly square:

$$x = k f \frac{X}{Z}, \quad y = l f \frac{Y}{Z}$$

- x, y : coordinates (pixels)
- k, l : scale parameters (pixels/m)
- f : focal length (m or mm)

Intrinsic Parameters

- f, k, l are not independent.
- Can rewrite as follows (in pixels):

$$f_x = kf \quad f_y = lf$$

- So,

$$x = f_x \frac{X}{Z}, \quad y = f_y \frac{Y}{Z}$$

- Image centre or principal point c may not be at origin. Denote location of c in image plane as cx, cy .

- Then,

$$x = f_x \frac{X}{Z} + c_x, \quad y = f_y \frac{Y}{Z} + c_y$$

- Along optical axis, $X = Y = 0$, and $x = c_x, y = c_y$.

Intrinsic Parameters

- Image frame may not be exactly rectangular.
- Let θ denote skew angle between x- and y-axis.
- Then,

$$x = f_x \frac{X}{Z} - f_x \cot \theta \frac{Y}{Z} + c_x, \quad y = \frac{f_y}{\sin \theta} \frac{Y}{Z} + c_y$$

- Combine all parameters yield

$$\tilde{\mathbf{x}} = \frac{1}{Z} \mathbf{K} \mathbf{X}, \quad \mathbf{K} = \begin{bmatrix} f_x & -f_x \cot \theta & c_x \\ 0 & \frac{f_y}{\sin \theta} & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic
parameter
matrix

$\tilde{\mathbf{x}} = [x, y, 1]^\top$ homogeneous coordinates

Radial lens Distortion

- Radial distortion is modelled as:

$$x_d = x (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6)$$

$$y_d = y (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6)$$

distorted
coordinates

undistorted
coordinates

distortion
parameters

- With $r^2 = x^2 + y^2$

- Actual image coordinates

$$x_a = f_x x_d + c_x$$

$$y_a = f_y y_d + c_y$$

Direct parameter calibration

Direct recovery of the intrinsic and extrinsic camera parameters.

$$M_{in} = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{ex} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T T \\ r_{21} & r_{22} & r_{23} & -R_2^T T \\ r_{31} & r_{32} & r_{33} & -R_3^T T \end{bmatrix}$$

Direct parameter calibration

- Review of basic equations
 - From world coordinates to camera coordinates:

$$P_c = R(P_w - T) \text{ or } P_c = RP_w - RT \text{ or } P_c = RP_w - T'$$

- For simplicity, we will replace $-T'$ with T
- – Warning: this is NOT the same T as before!

$$P_c = RP_w + T$$



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

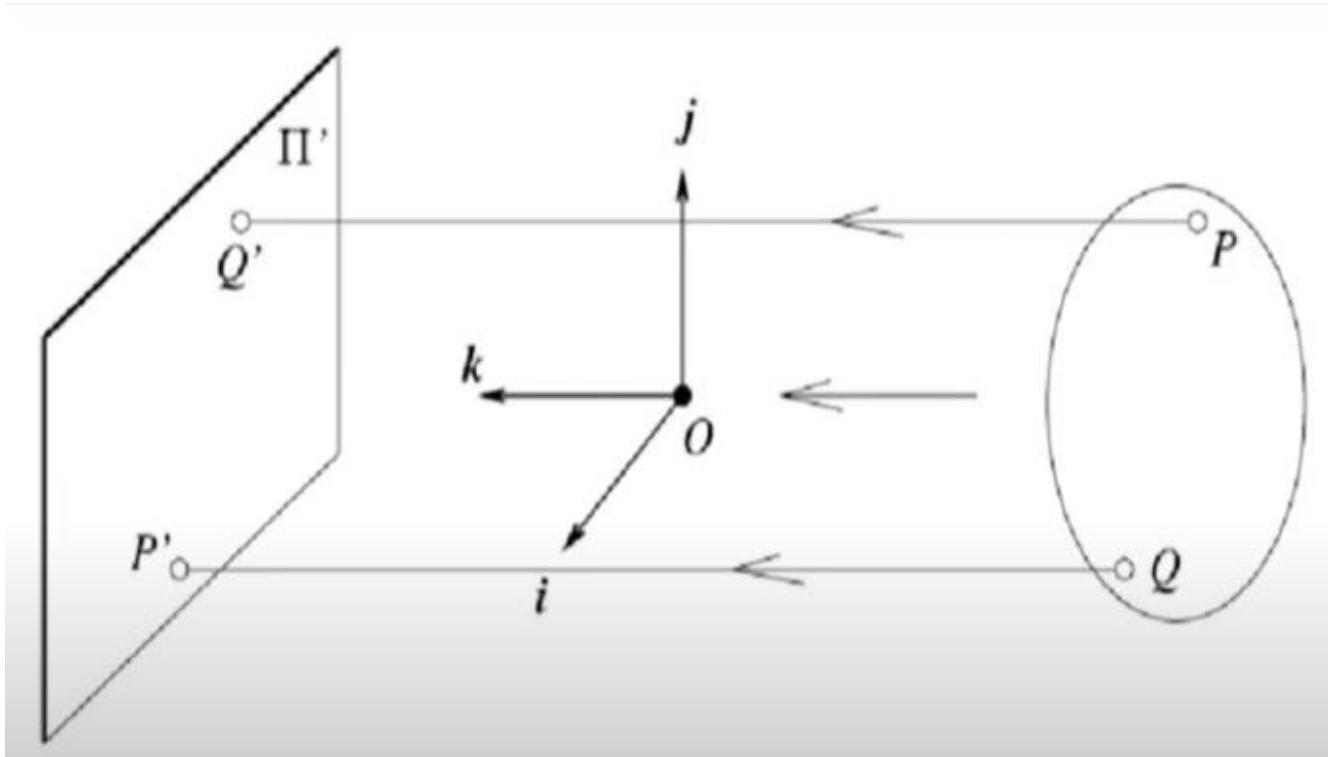
Orthographic Projection

As a special case of the weak perspective projection, when f/z_c factor equals 1, we have $u = xc$ and $v = yc$, i.e., the lens (rays) of projection are parallel to the optical axis, i.e., the projection rays meet in the infinite instead of lens center.

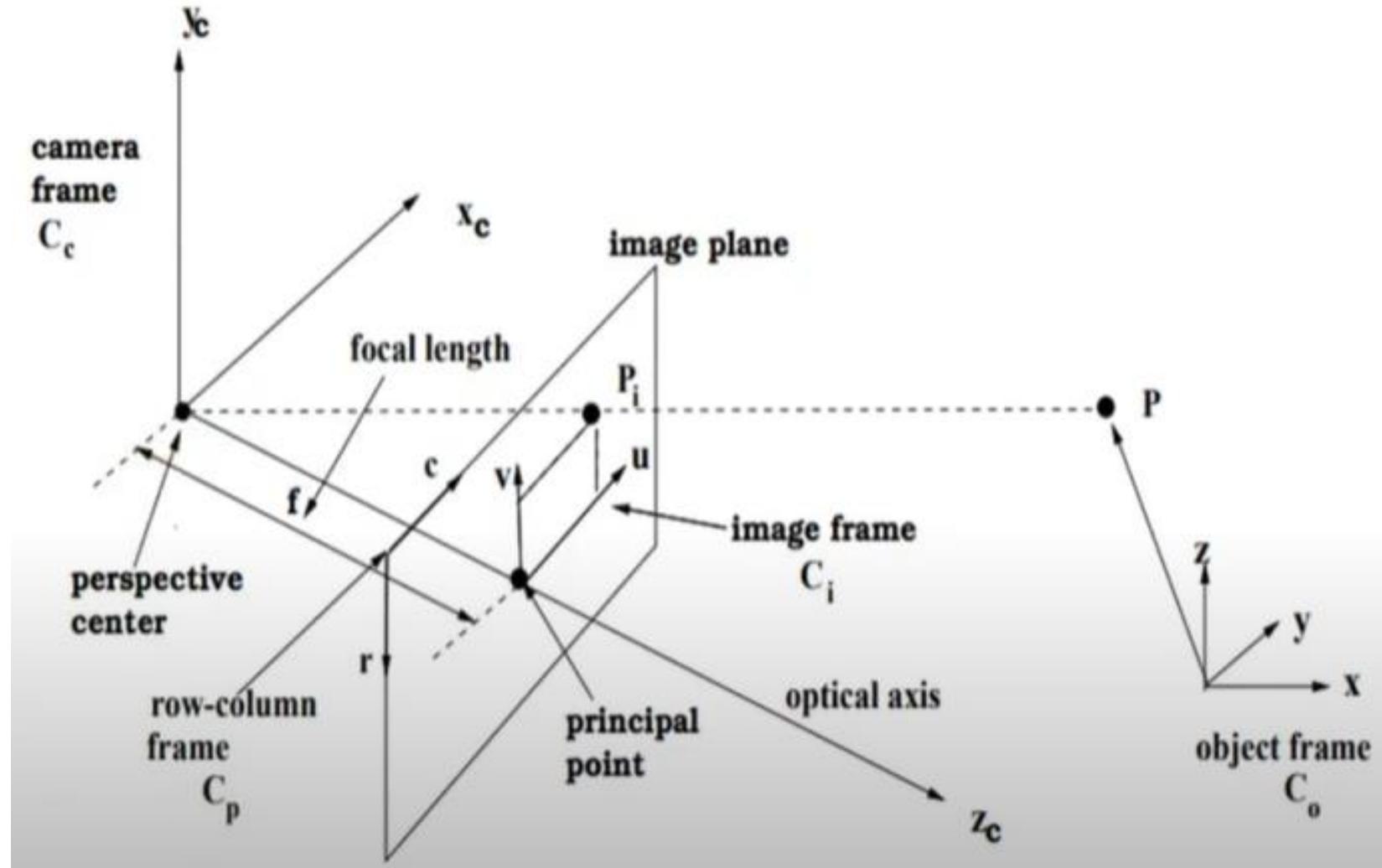
This leads to the sizes of image and the object are the same.

This is called orthographic projection.

Orthographic Projection

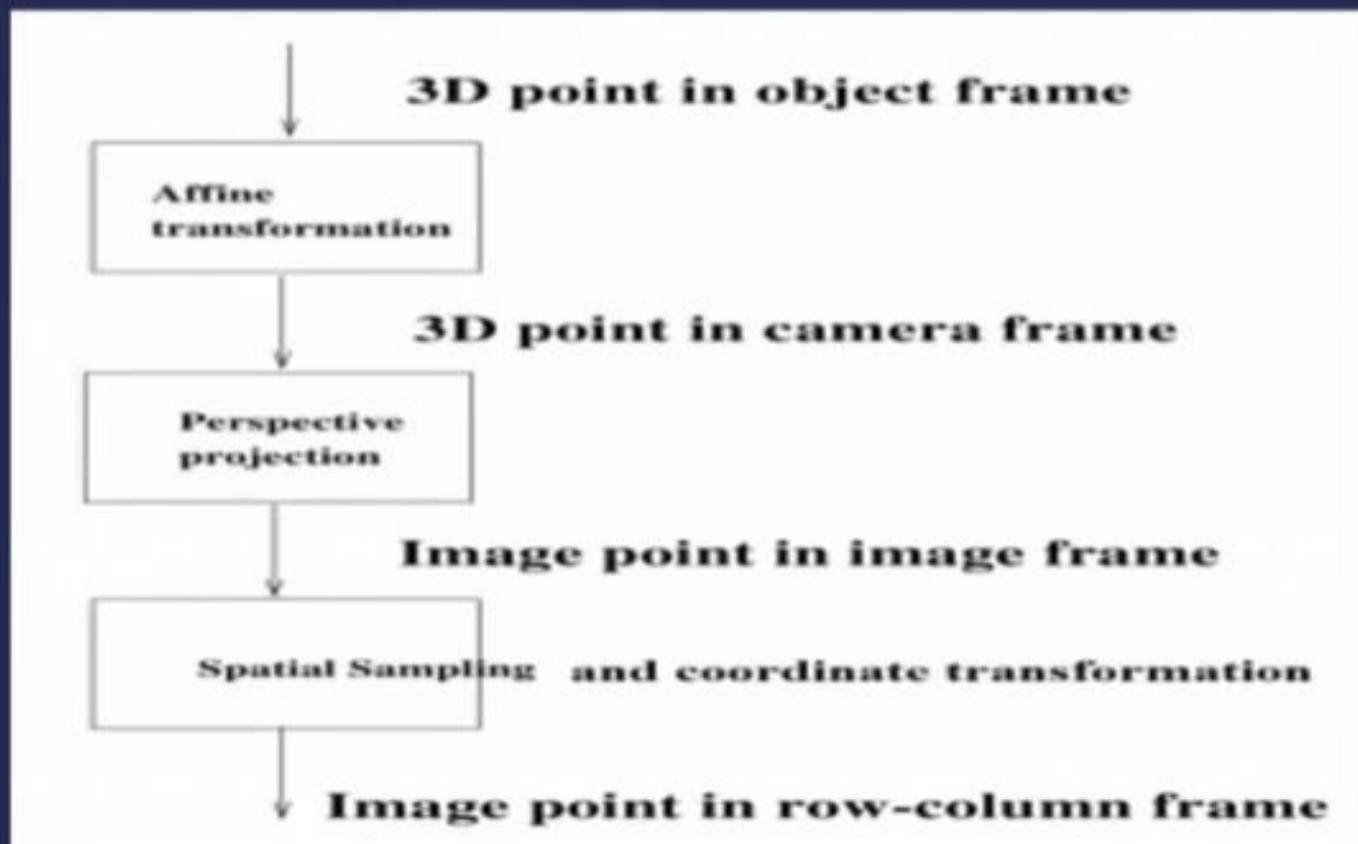


Perspective Projection



Projection Process

Our goal is to go through the projection process to understand how an image point (c, r) is generated from the 3D point (x, y, z) .



Relationships between different frames

Between camera frame (C_c) and object frame (C_o)

$$X_c = RX + T$$

X is the 3D coordinates of P w.r.t the object frame.
 R is the rotation matrix and T is the translation vector.

R and T specify the orientation and position of the object frame relative to the camera frame.
They are often collectively called the pose of the object,

Relationships between different frames

Substituting the parameterized T and R into equation 1 yields

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Relationships between different frames

Between image frame (C_i) and row-col frame (C_p)
(spatial quantization process)

$$\begin{pmatrix} c \\ r \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} c_0 \\ r_0 \end{pmatrix}$$

where s_x and s_y are scale factors (pixels/mm) due to spatial quantization. c_0 and r_0 are the coordinates of the principal point in pixels relative to C_p .

Relationships between different frames

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

$r_i = (r_{i1}, r_{i2}, r_{i3})$ be a 1×3 row vector, R can be written as :

$$R = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{pmatrix}$$

Affine Camera Model

A further simplification from weak perspective camera model is the affine camera model, which is often assumed by computer vision researchers due to its simplicity.

The affine camera model assumes that the object frame is located on the centroid of the object being observed.

$$\mathbf{X}' = \mathbf{X} - \bar{\mathbf{X}} = \Delta \mathbf{X} \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}$$

Thank u!!!

Unit 6

Motion Representation

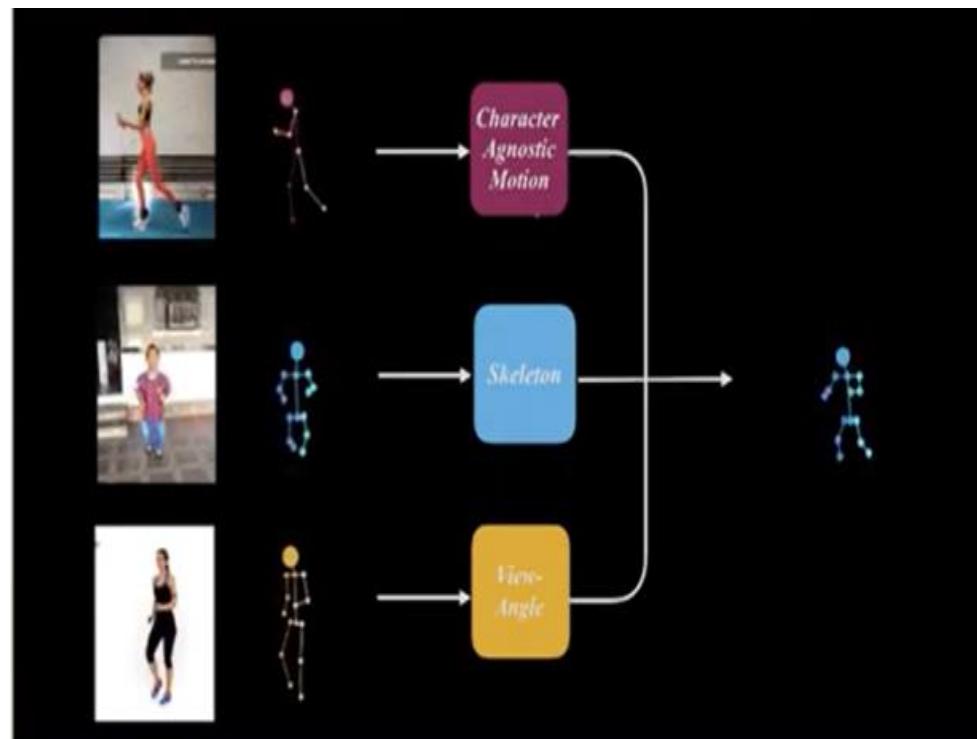
Prof. Janki Patel
Department of IT
SPCE

Content

- The motion field of rigid objects
- Motion parallax
- Optical flow, the image brightness constancy equation, affine flow
- Differential techniques
- Feature-based techniques
- Regularization and robust estimation

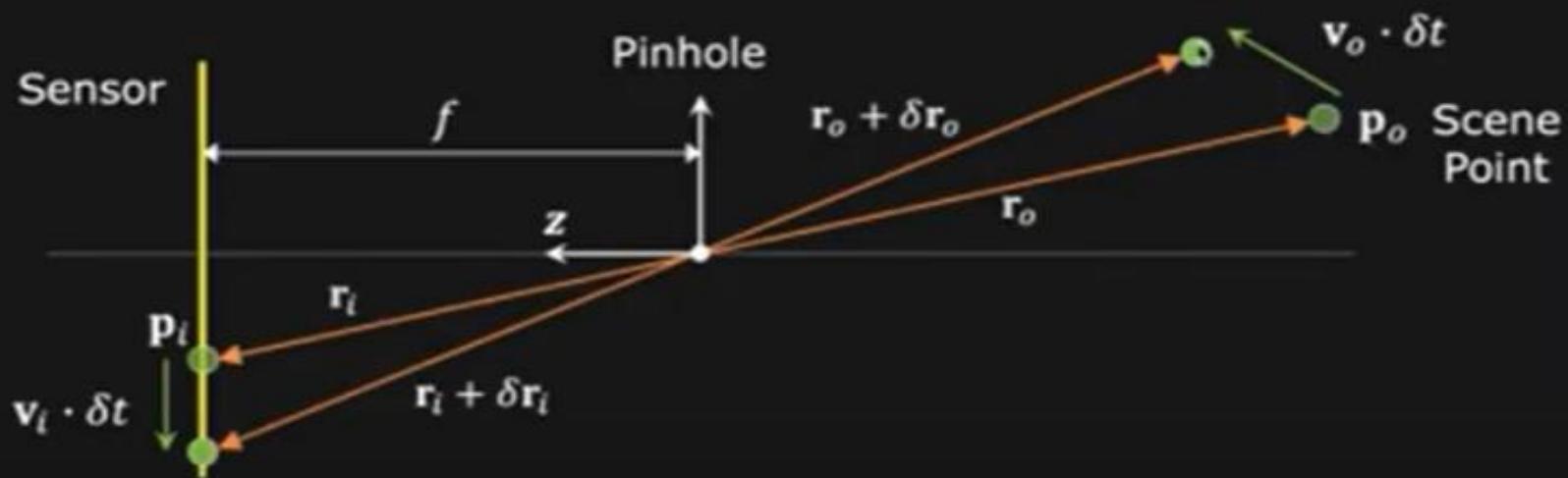
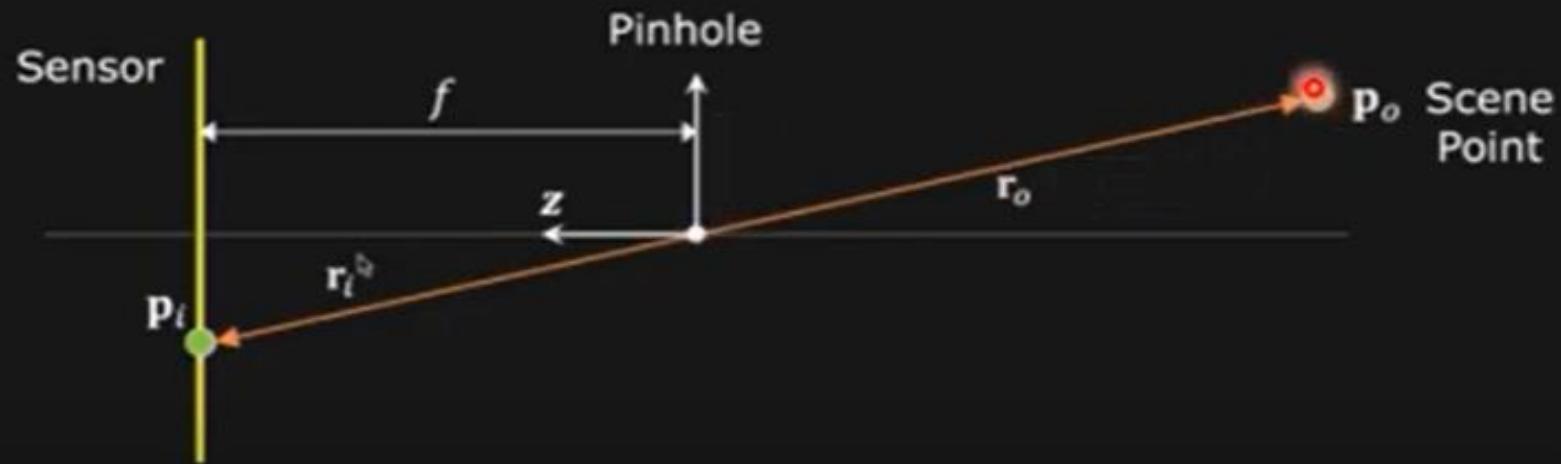
What is Motion Representation?

- Motion analysis was motivated by the need for tracking an object and advancement in image processing hardware.
- Analyzing human motion is a challenging task with a wide variety of applications in computer vision and in graphics. One such applications, of particular importance in computer animation, is the retargeting of motion from one performer to another. While human motions are captured using video, requiring 2D-to-3D pose and camera recovery, before existing retargeting approaches may be applied.



The Motion Field

- In computer vision the **motion field** is an ideal representation of 3D motion as it is projected onto a camera image. Given a simplified camera model, each point (y_1, y_2) in the image is the projection of some point in the 3D scene but the position of the projection of a fixed point in space can vary with time.
- The motion field can formally be defined as the time derivative of the image position of all image points given that they correspond to fixed 3D points. This means that the motion field can be represented as a function which maps image coordinates to a 2-dimensional vector. The motion field is an ideal description of the projected 3D motion in the sense that it can be formally defined but in practice it is normally only possible to determine an approximation of the motion field from the image data.



The Motion Field

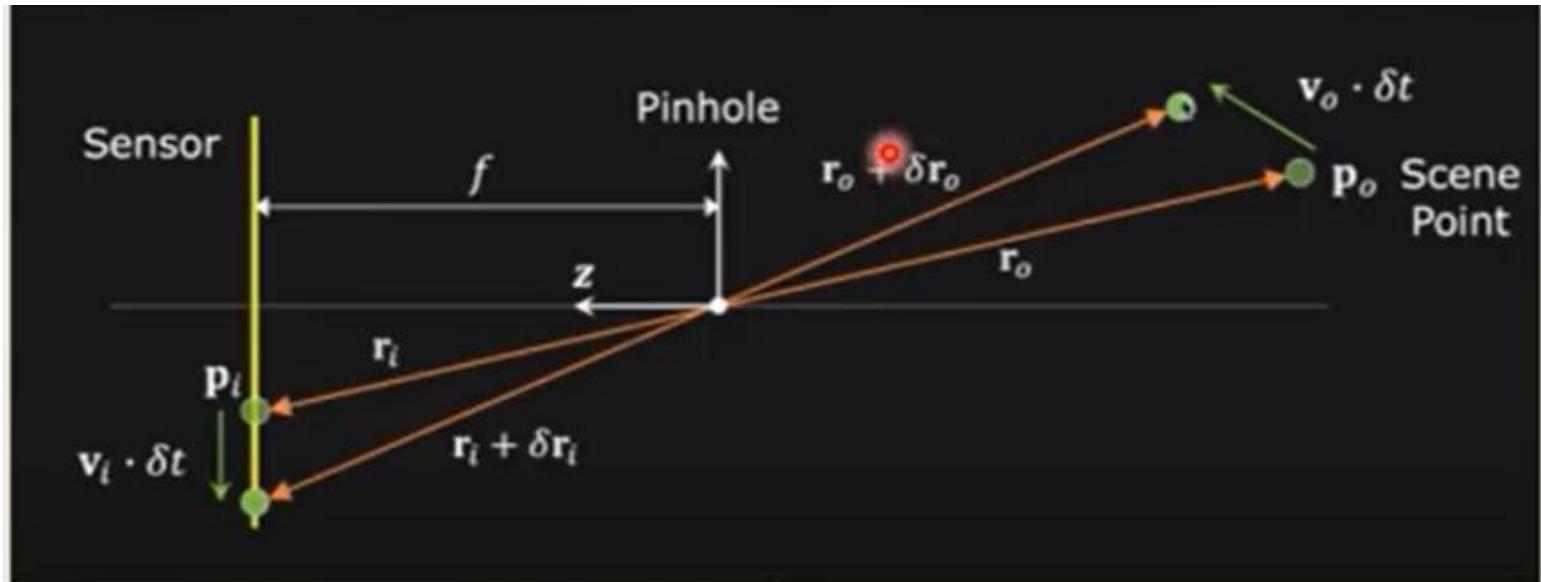
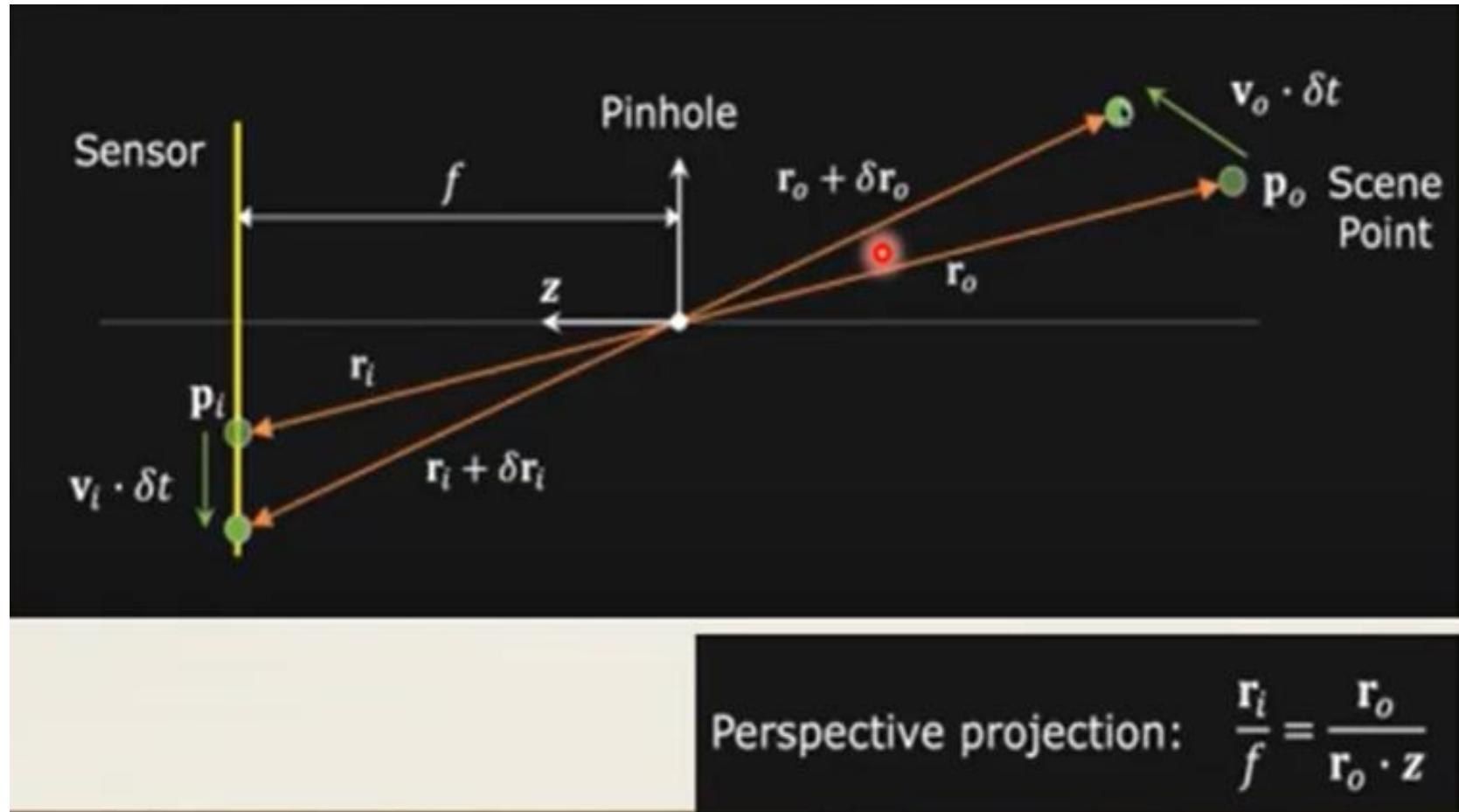


Image Point Velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt}$
(Motion Field)

Scene Point Velocity: $\mathbf{v}_o = \frac{d\mathbf{r}_o}{dt}$

The Motion Field



The Motion Field

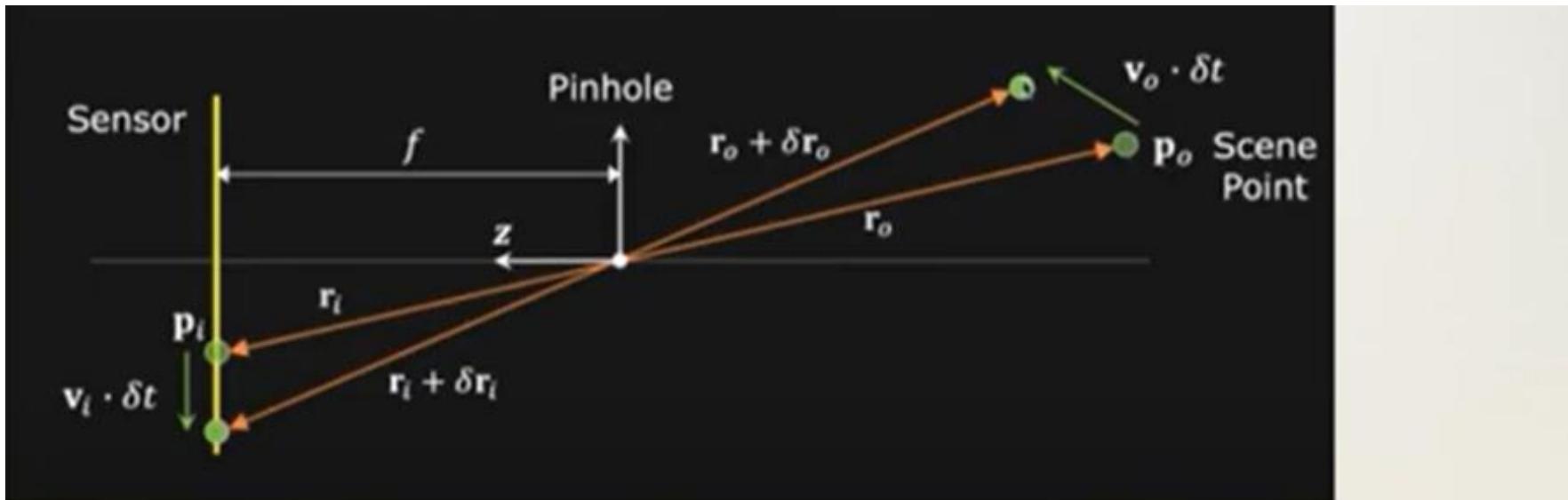
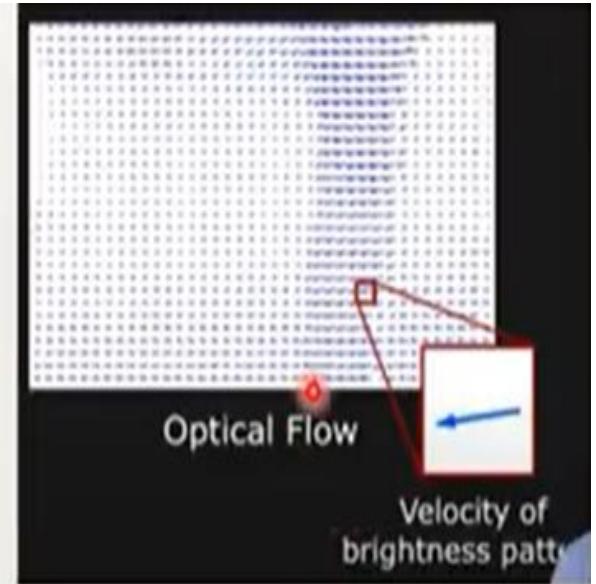
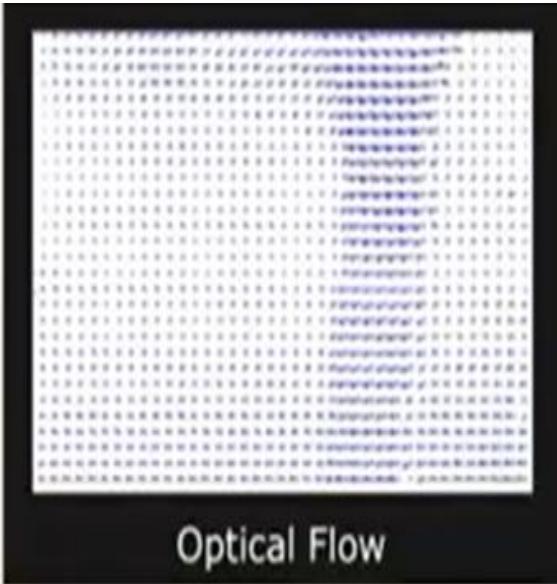
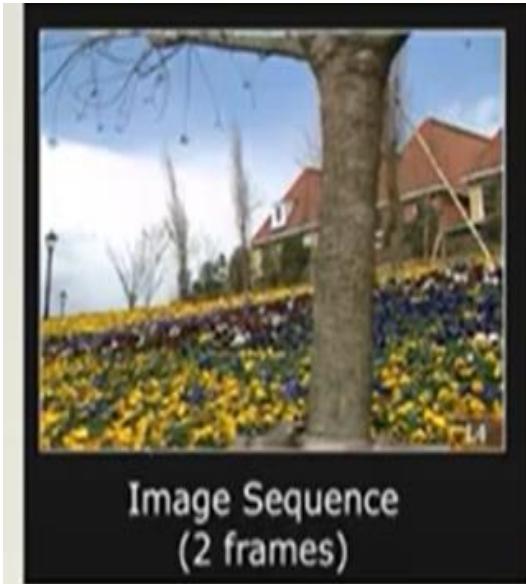


Image Point Velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt} = f \frac{(\mathbf{r}_o \cdot \mathbf{z})\mathbf{v}_0 - (\mathbf{v}_o \cdot \mathbf{z})\mathbf{r}_0}{(\mathbf{r}_o \cdot \mathbf{z})^2}$

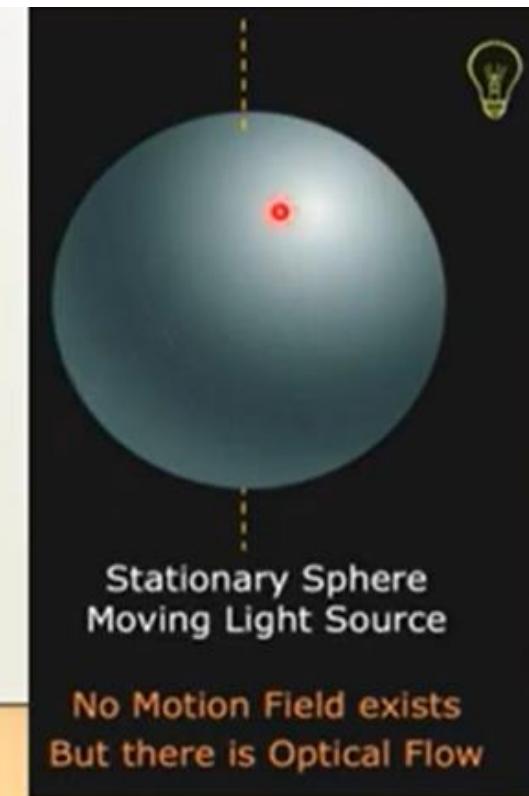
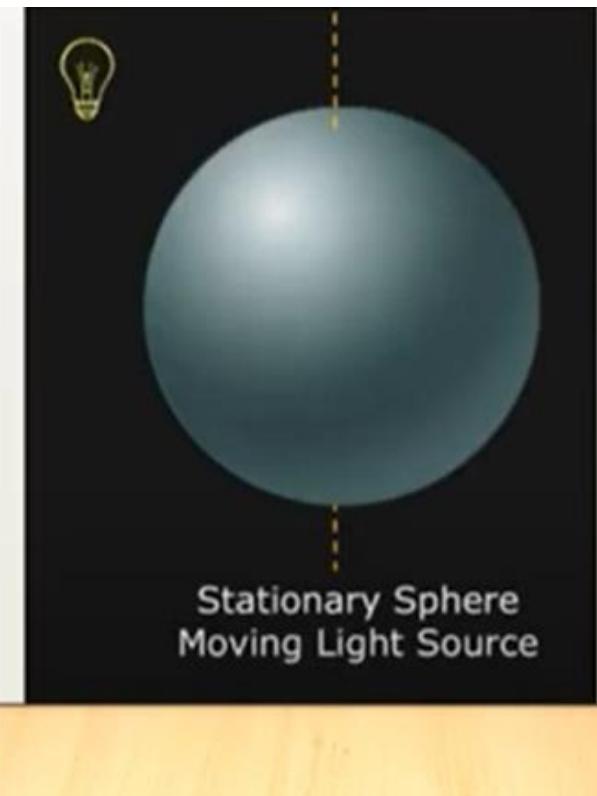
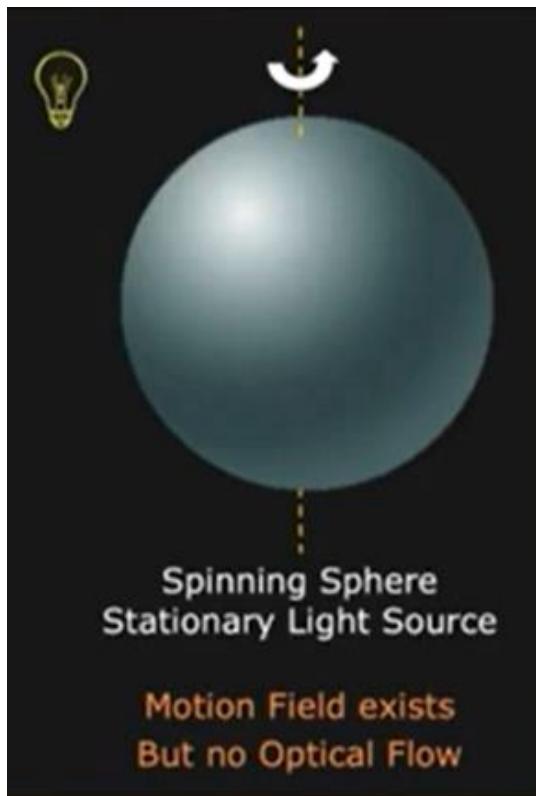
$$\mathbf{v}_i = f \frac{(\mathbf{r}_o \times \mathbf{v}_0) \times \mathbf{z}}{(\mathbf{r}_o \cdot \mathbf{z})^2}$$

Optical Flow

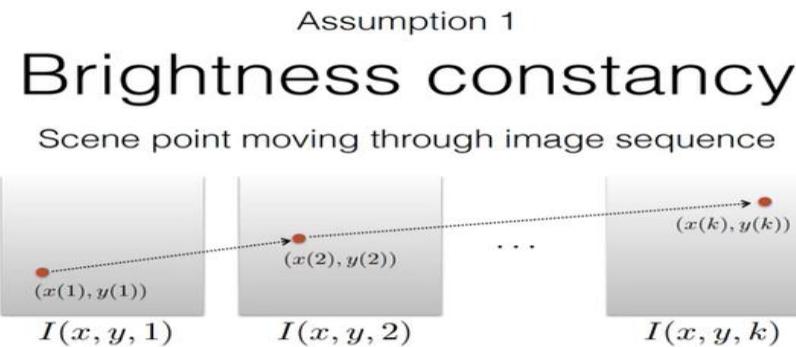
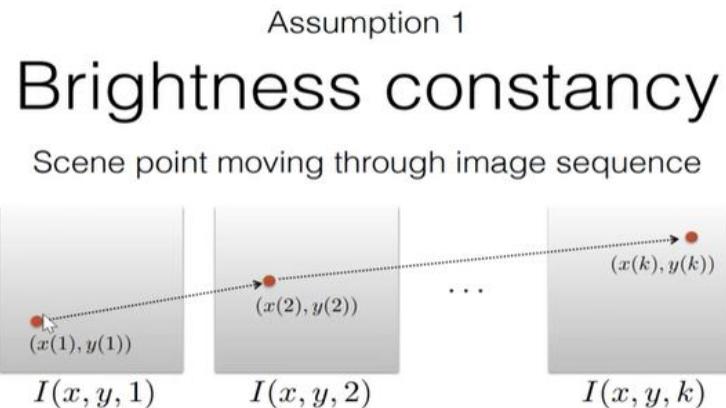
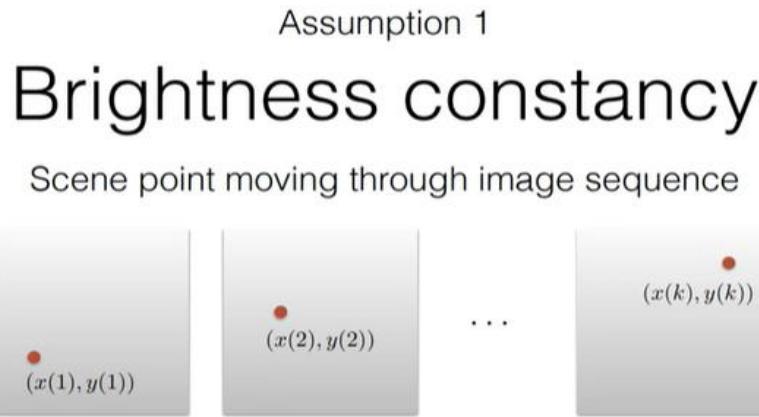
- Motion of brightness patterns in the image.



When Optical Flow \neq Motion Filed?



Brightness Constancy



Assumption: Brightness of the point will remain the same

$$I(x(t), y(t), t) = C$$

constant

Small motion

Assumption 2

Small motion



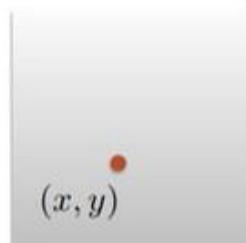
$I(x, y, t)$



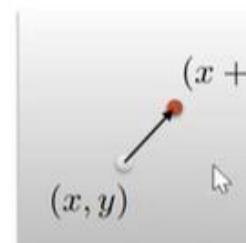
$I(x, y, t + \delta t)$

Assumption 2

Small motion

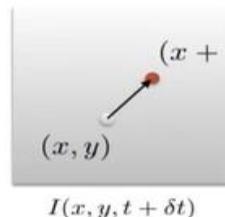
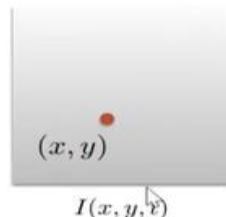


(x, y)



$I(x, y, t + \delta t)$

Assumption 2
Small motion



Optical flow (velocities): (u, v)

Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

Feature-based Techniques

Feature-based Techniques

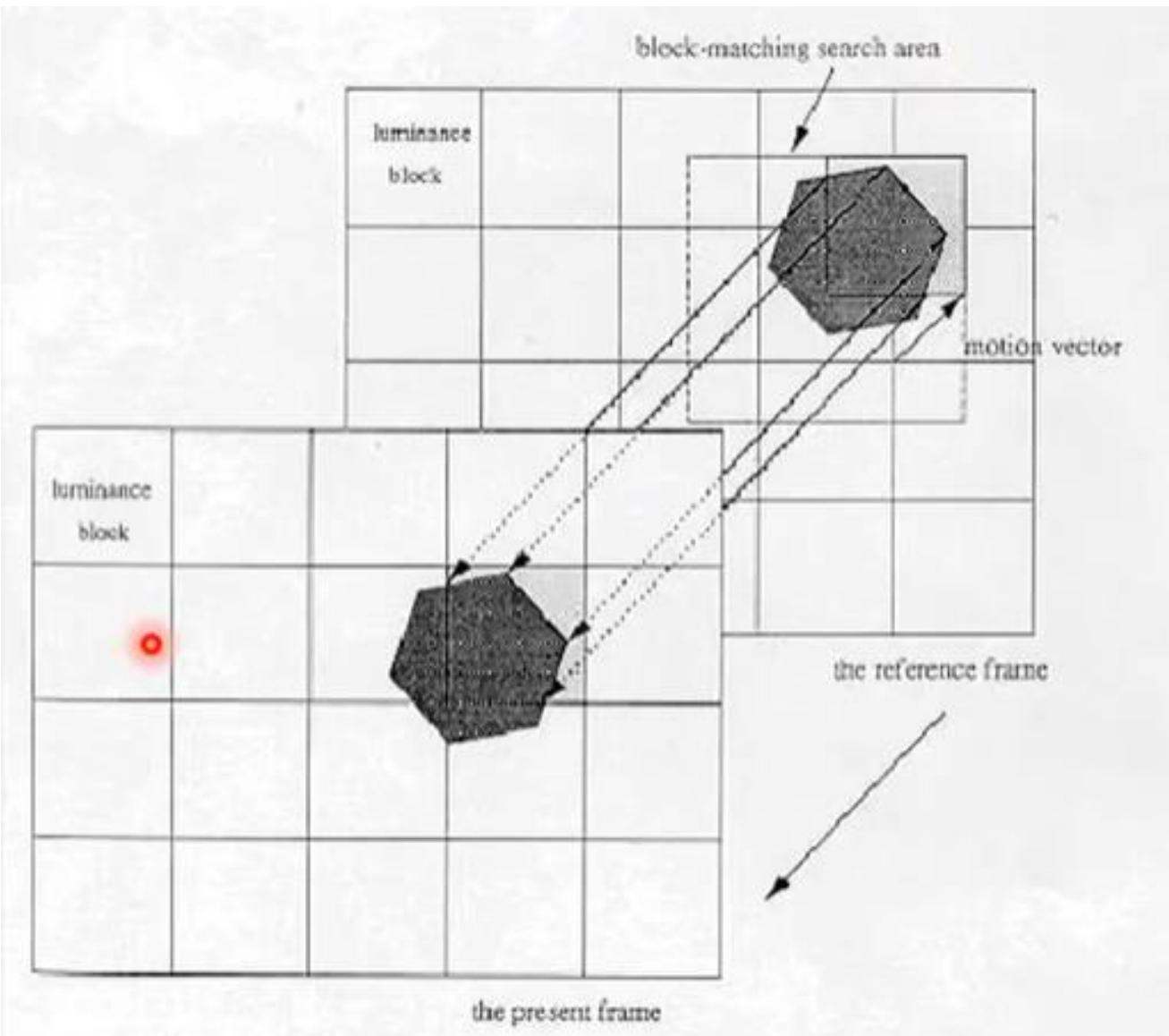
- The method of finding image displacements which is easiest to understand is the feature-based approach. This finds features (for example, image edges, corners, and other structures well localized in two dimensions) and tracks these as they move from frame to frame. This involves two stages. Firstly, the features are found in two or more consecutive images.
- The act of feature extraction, if done well, will both reduce the amount of information to be processed (and so reduce the workload), and also go some way towards obtaining a higher level of understanding of the scene, by its very nature of eliminating the unimportant parts. Secondly, these features are matched between the frames. In the simplest and commonest case, two frames are used and two sets of features are matched to give a single set of motion vectors.

Feature-based Techniques

- Most of the motion estimation algorithms make the following assumptions:
 1. Objects move in translation in a plane that is parallel to the camera plane, i.e., the effects of camera zoom, and object rotations are not considered.
 2. Illumination is spatially and temporally uniform.
 3. Occlusion of one object by another, and uncovered background are neglected.

Feature-based Techniques

- There are two mainstream techniques of motion estimation:
 1. pel-recursive algorithm (PRA)
 2. block-matching algorithm (BMA).
- RAs are iterative refining of motion estimation for individual pels by gradient methods. BMAs assume that all the pels within a block has the same motion activity. BMAs estimate motion on the basis of rectangular blocks and produce one motion vector for each block. PRAs involve more computational complexity and less regularity, so they are difficult to realize in hardware. In general, BMAs are more suitable for a simple hardware realization because of their regularity and simplicity.



Feature-based Techniques

- Figure illustrates a process of block-matching algorithm. In a typical BMA, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block.
- The best (lowest distortion, i.e., most matched) candidate block is found and its displacement (motion vector) is recorded. In a typical interframe coder, the input frame is subtracted from the prediction of the reference frame. Consequently the motion vector and the resulting error can be transmitted instead of the original luminance block; thus interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames. The summation gives an exact replica of the current frame. The better the prediction the smaller the error signal and hence the transmission bit rate.

Thank U!!!

Unit 7

Motion tracking

Prof. Janki Patel
Department of IT
SPCE

Introduction

- An important area of computer vision is real-time object tracking, which is now widely used in intelligent transportation and smart industry technologies.
- Although the correlation filter object tracking methods have a good real-time tracking effect, it still faces many challenges such as scale variation, occlusion, and boundary effects.
- Many scholars have continuously improved existing methods for better efficiency and tracking performance in some aspects.
- To provide a comprehensive understanding of the background, key technologies and algorithms of single object tracking, this session focuses on the correlation filter-based object tracking algorithms.
- Computer vision refers to the use of cameras and computers instead of human eyes to visually recognize, track, and measure targets. First, image processing is performed so that the processed image is more suitable for human eye observation or instrument detection.

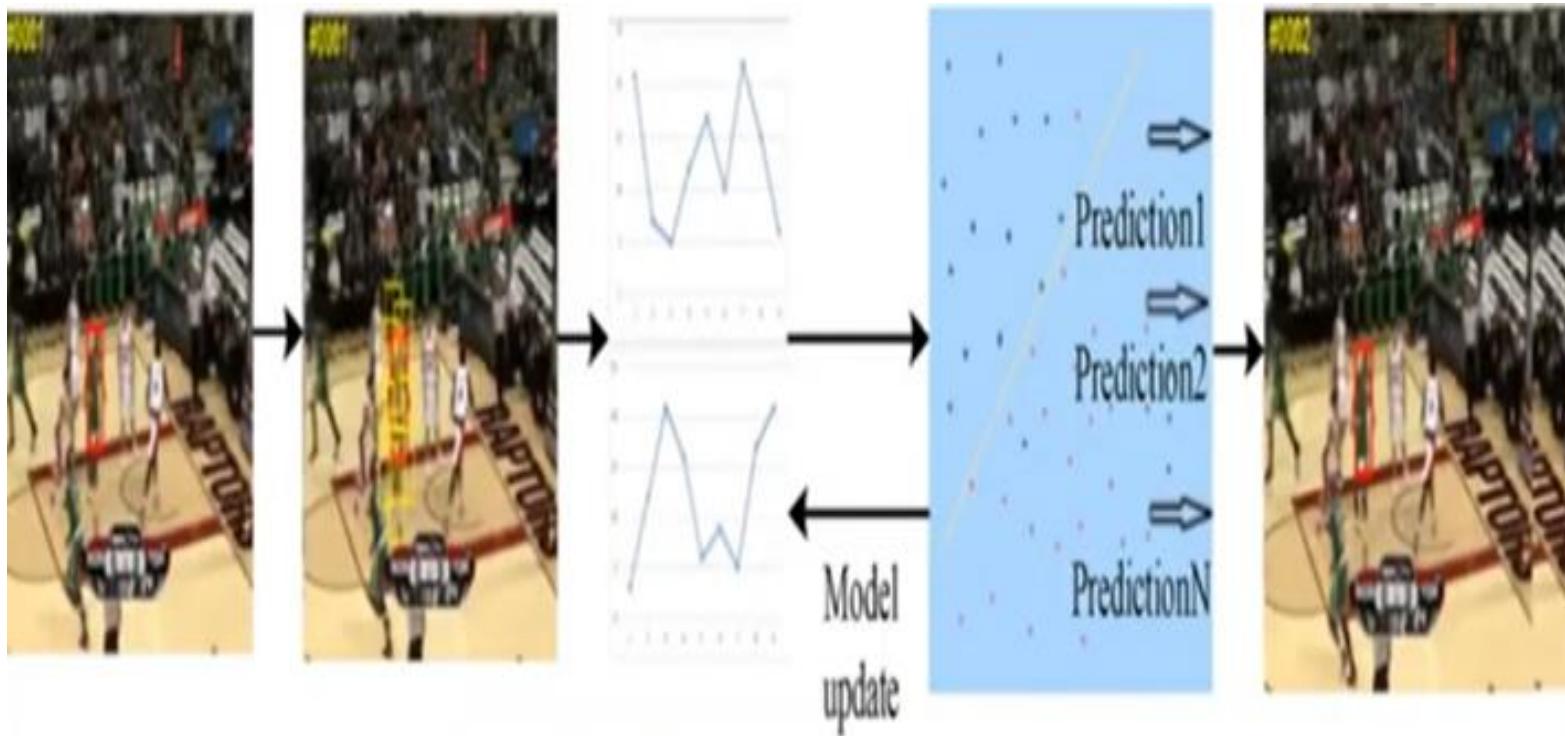
Applications of motion tracking

- The various applications are:
 1. Application of object tracking in unmanned aerial vehicles (UAVs). UAVs is commonly known as drones. Compared with the human eye, the drone has the advantages of stability and accurately capture. Therefore, drones using object tracking technology can achieve more robust tracking results.
 2. Application of object tracking in industrial automation. All industries are developing intelligently, and object tracking is also widely used in industrial production.
 3. Application of object tracking in intelligent transportation. Today, an efficient transportation selection system is vital for the masses. In intelligent transportation, real-time monitoring and tracking of vehicles can be achieved using object detection and tracking technology

Motion tracking

- Object tracking is an important component of computer vision.
- Object tracking can be divided into single object tracking and multi-object tracking.
- The process of single object tracking is to find a calibration target in the next sequence based on the given first frame information.
- In single object tracking, there are two non-depth methods: generative method and discriminative method.
- At present, the five steps of the discriminative method commonly used are motion model, feature extraction, observation model, model update and integration method.

Motion Tracking



Input image

Motion model

Feature extraction

Observation model

Integration method

Output image

Motion Tracking

- At first, the image to be processed needs to be input and the motion model also needs to be selected.
- The motion model is used to generate candidate sample information which may contain the target.
- The speed and quality of sample generation affect the speed and effectiveness of the entire tracking.
- Motion models typically include particle filtering and sliding windows.
- Particle filtering uses particle sets to represent probabilities.
- This method approximates the probability density function by finding a set of random samples propagating in the state space, and replaces the integral operation with the sample mean.
- The sliding window generates a series of candidate samples on the calibration target frame through the cyclic matrix method.
- Next, feature extraction is performed. Feature extraction is to find features in the candidate region which can uniquely identify the target. In object tracking, the quality of the feature has the most direct impact on tracking results.

Motion Tracking

- The observation model is used to determine whether the current sample is matching tracked object.
- The generative model and the discriminative model are divided according to the observation model.
- The generated model is tracked by searching for the object most similar to the target in the current area.
- This is a template matching process, where sparse representation is the most commonly used.
- Finally, a discriminant model is obtained through learning as a discriminator, and the discriminator is used to discriminate whether the current sample is the target.

Iterated Estimation

- Pose estimation algorithms utilize frames of a video feed from a moving camera to generate hypotheses for how the camera has moved over the course of each consecutive frame.
- Until now, algorithms used for pose estimation required the generation of up to five to ten hypotheses for how the camera was moving given the data in the video feed.
- These hypotheses were then scored by how well they fit the data, with the highest scoring hypothesis being chosen as the best pose estimate.
- Unfortunately, the generation of multiple hypotheses is computationally expensive and results in slower return time for robust pose estimation.

Iterated Estimation

- Researchers have found a way to seed, or give hints to, an already used algorithm in computer vision by feeding it information in between each frame, thus greatly reducing the need for generating many hypotheses.
- "At each iteration, we use the current best hypothesis to seed the algorithm."
- The reduction of required hypotheses directly results in reduced computation time and complexity; "we show that this approach significantly reduces the number of hypotheses that must be generated and scored to estimate the pose, thus allowing real-time execution of the algorithm."

Observability and linear systems

- Observability requirements previously established for bearings-only tracking in two dimensions are extended to a class of three-dimensional estimation algorithms capable of processing any pairwise combination of azimuth bearing, conical bearing, and depth/elevation angle measurements.
- Although these algorithms are intrinsically nonlinear, it is shown that they can be analyzed in a linear framework without sacrificing mathematical rigor.

Observability and linear systems

- A simplified observability criterion, applicable to both autonomous and nonautonomous linear systems, is presented and utilized to specify conditions on own-ship motion which are both necessary and sufficient for a unique tracking solution.
- Further analysis reveals that observability dependence on own-ship maneuvers for the three-dimensional algorithms considered here parallels the concomitant two-dimensional requirements. An interesting difference, however, is that under certain conditions, a unique tracking solution can be obtained in three dimensions for unaccelerated own-ship motion.

The Kalman filter

The Kalman filter has long been regarded as the optimal solution to many applications in computer vision for example the tracking objects, prediction and correction tasks.

Its use in the analysis of visual motion has been documented frequently, we can use in computer vision and open cv in different applications in reality for example robotics, military image and video, medical applications, security in public and privacy society, etc.

The Kalman filter

In this, we investigate the implementation of a Matlab code for a Kalman Filter using three algorithm for tracking and detection objects in video sequences (block-matching (Motion Estimation) and Camshift Meanshift (localization, detection and tracking object)).

The Kalman filter is presented in three steps:

1. Prediction
2. estimation (correction)
3. update.

The Kalman filter

- The first step is a prediction for the parameters of the tracking and detection objects.
- The second step is a correction and estimation of the prediction parameters.
- The important application in Kalman filter is the localization and tracking mono-objects and multi-objects are given in results.
- This work presents the extension of an integrated modeling and simulation tool for the tracking and detection objects in computer vision described at different models of algorithms in implementation systems.

The kalman filter algorithm

- In an object tracking algorithm, there are generally four steps: detection, location, association, and trajectory estimation [1, 2, 3].
- The algorithms are composed by three important modules: block matching and meanshift, camshift, Kalman filter.
- The Kalman filter is used in a wide range of technological fields.
- It is a major theme in automation and frame and signal processing.
- The Kalman filter “KF” is a set of mathematical equations which provide an efficient (recursive) computation of the means for estimation the state of a process.

The kalman filter algorithm

The KF is very powerful in several aspects: it supports estimates of past, present and even future states and it can do so even when the exact nature of the modeled system tracking and detection objects.

The Kalman filter is a corrective predictor filter. In the tracking system objects, this filter looks at an object as it moves, that is, it takes information on the state of the object at the precise moment. Then, it uses this information to predict where the object is in the next frame.

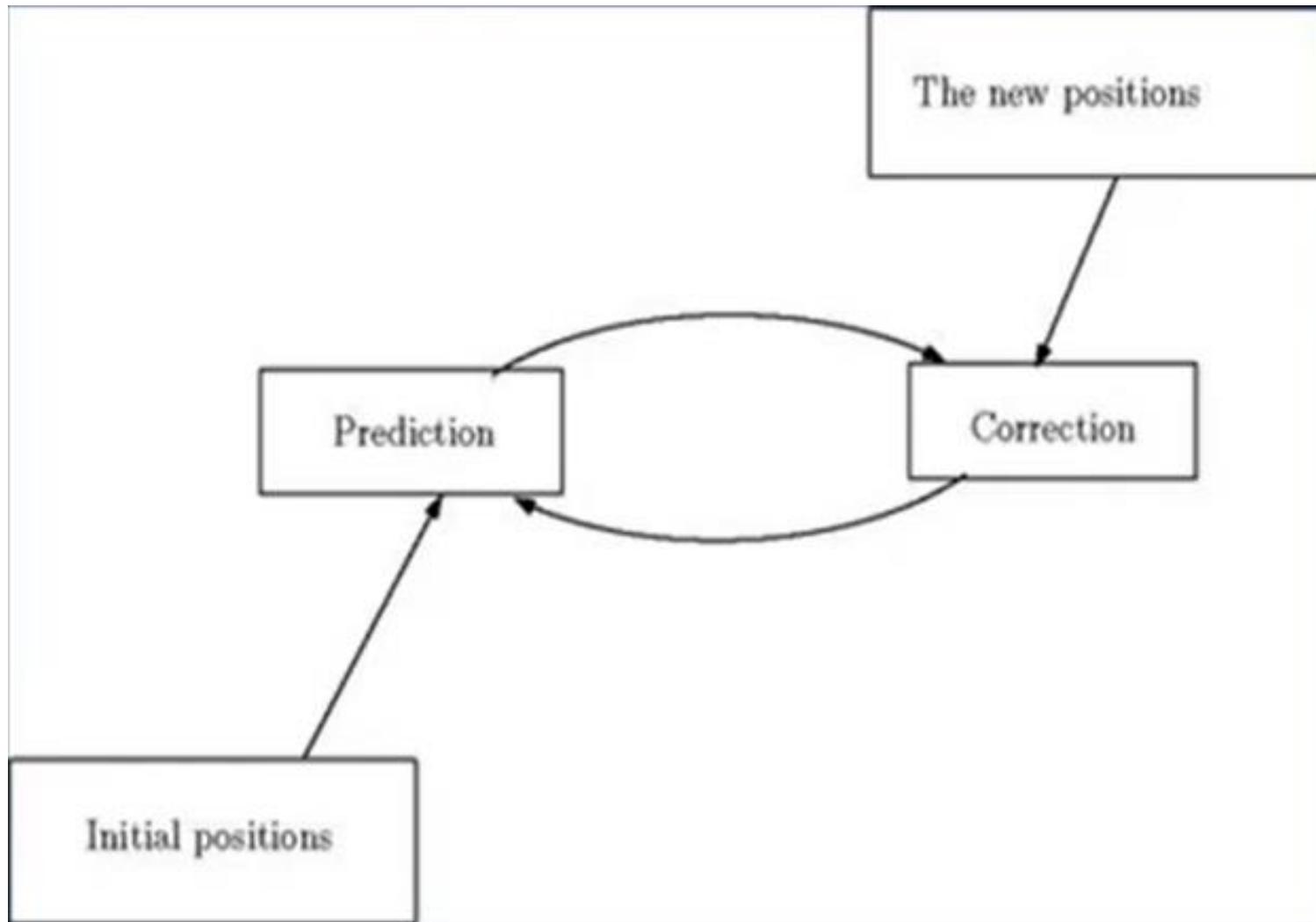
For this, it takes as input a measurement vector (position in x, in y, width and height of the object).

The kalman filter algorithm

In the tracking process, this filter looks at an object as it moves, that is it takes information on the state of the object at the precise moment.

In the case of tracking an object in motion, the Kalman filter allows us to estimate the states of motion of the object (and therefore predetermine the areas of motion in the following frames with using the combination for three algorithms (block-matching, Camshift and Kalman Filter)) and thus adds robustness tracking objects.

The kalman filter algorithm



Formulation and modeling of the kalman filter

- The main objective of the Kalman filter is to estimate the vector of states in a discrete time.
- This process is illustrated by Eq. with stochastic linear differences:

$$x_k = A \times x_{k-1} + w_{k-1}$$

- With a measurement vector which has the following form:

$$z_k = H \times x_k + v_k$$

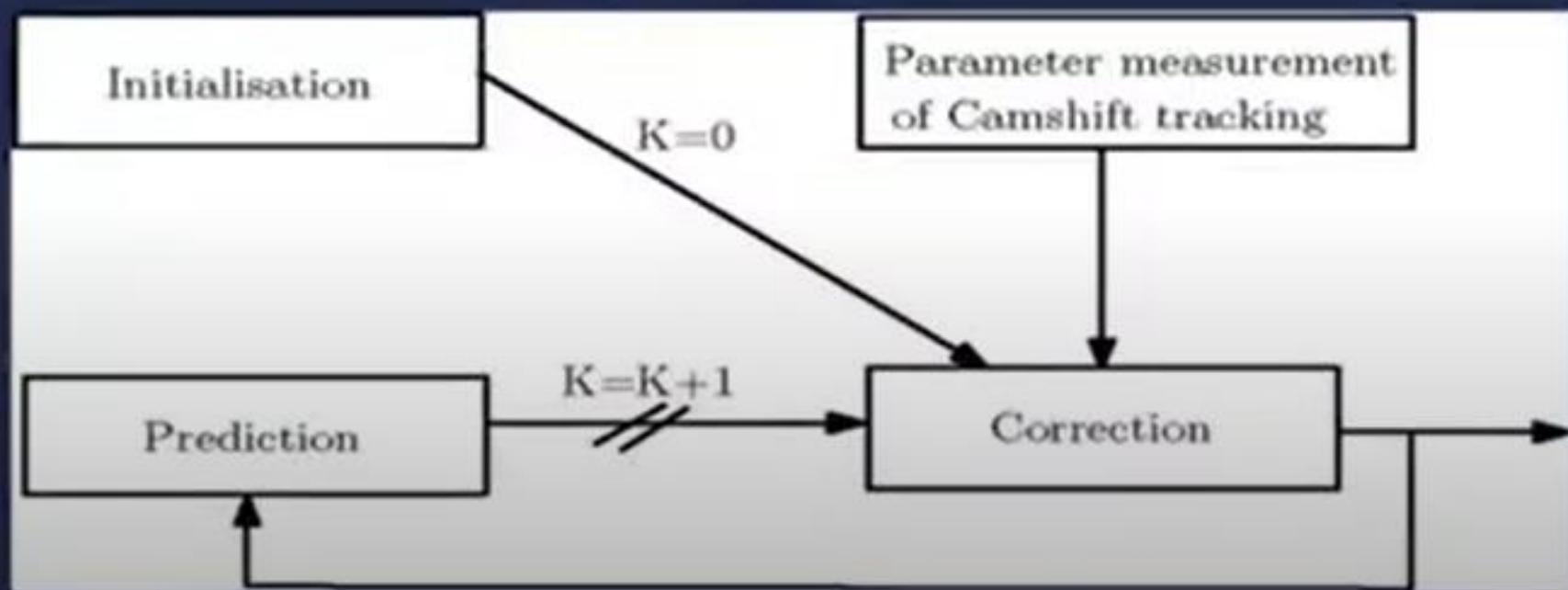
- A is the transition matrix and H presents the measurement matrix.
- Random variables w_k and v_k present Gaussian and measurement noise (respectively)

Different function of the Kalman filter

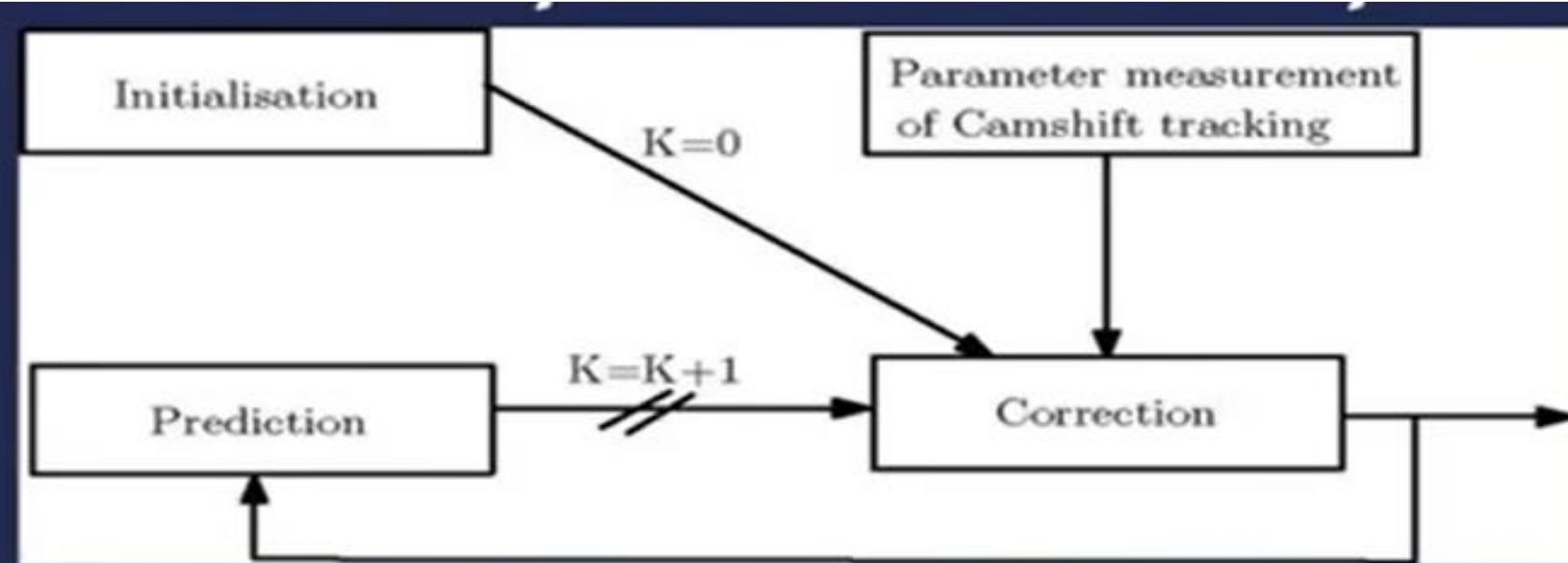
- The Kalman filter is an optimal recursive estimator to the linear filtering problem Data.
- This filter has two necessary modules, a prediction module and a correction or estimation module.
- The Kalman filter then makes it possible to estimate the position of the object by achieving a compromise between the position observed in the frame and the predicted position.
- The input parameters of the Kalman filter are respectively, the position of the object in the frame at time “k,” the size of the object and the width and length of the object search window which are variable due to the mobility of the object during the sequence.

Different function of the Kalman filter

- These parameters represent the state vector and the filter measurement vector from Kalman filter.
- Figure shows the Kalman filter cycle [8, 9, 10]. Generally the estimation of the tracking parameters with a Kalman filter is a process requires the following steps: itemize.



Different function of the Kalman filter



The measure which consists in taking the tracking parameters computed in the Camshift algorithm.
The estimate, which updates the position of the object.

The prediction, which computes the position of the object in the next frame.

Different function of the Kalman filter

The state vector is composed by the initial position, the width and the length of the search window as well as the center of gravity of the object ($x_c; y_c$) at time t_k . This vector is presented by the following Eq.:

$$s_k = (x_k; y_k; W_k; L_k; x_c; y_c)$$

The measurement vector of the Kalman filter is composed of the initial position, the length and the width of the search window for the object at time t_k . This vector is given by the Eq.:

$$z_k = (x_k; y_k; W_k; L_k)$$

Thank u!!!

Unit 8

Object recognition and

Shape representation

Prof. Janki Patel
Department of IT
SPCE

Image Eigen spaces

Object recognition methods can be classified according to a number of characteristics.

We focus on model acquisition (learning) and invariance to image formation conditions.

Historically, two main trends can be identified.

In the so called geometry- or model-based object recognition, the knowledge of an object appearance is provided by the user as an explicit CAD-like model. Typically, such a model describes only the 3D shape, omitting other properties such as colour and texture.

Image Eigen spaces

On the other end of the spectrum are the appearance-based methods, where no explicit user provided model is required.

The object representations are usually acquired through an automatic learning phase (but not necessarily), and the model typically relies on surface reflectance (albedo) properties.

Recently, methods which put local image patches into correspondence emerged.

Models are learned automatically, objects are represented by appearance of small local elements.

Global arrangement of the representation is constrained by weak or strong geometric models.

Appearance Based Methods

The central idea behind appearance-based methods is the following.

Having seen all possible appearances of an object, can recognition be achieved by just efficiently remembering all of them?

Could recognition be thus implemented as an efficient visual (pictorial) memory?

The answer obviously depends on what is meant by "all appearances".

The approach has been successfully demonstrated for scenes with unoccluded objects on black background.

Appearance Based Methods

- Appearance based methods typically include two phases.
- In the first phase, a model is constructed from a set of reference images.
- The set includes the appearance of the object under different orientations, different illuminants and potentially multiple instances of a class of objects, for example faces.
- The images are highly correlated and can be efficiently compressed using e.g. Karhunen-Loeve transformation (also known as Principal Component Analysis - PCA).

Appearance Based Methods

In the second phase, "recall", parts of the input image (subimages of the same size as the training images) are extracted, possibly by segmentation (by texture, colour, motion) or by exhaustive enumeration of image windows over whole image.

The recognition system then compares an extracted part of the input image with the reference images (e.g. by projecting the part to the Karhunen-Loeve space).

Appearance Based Methods

- The family of appearance-based object recognition methods includes global histogram matching methods.
- Swain and Ballard proposed to represent an object by a colour histogram.
- Objects are identified by matching histograms of image regions to histograms of a model image.
- While the technique is robust to object orientation, scaling, and occlusion, it is very sensitive to lighting conditions, and it is not suitable for recognition of objects that cannot be identified by colour alone.

Appearance Based Methods

The approach has been later modified by Healey and Slater and Funt and Finlayson to exploit illumination invariants.

Recently, the concept of histogram matching was generalised by Schiele ,where, instead of pixel colours, responses of various filters are used to form the histograms (called then receptive field histograms).

Appearance Based Methods

1) Edge matching

- Uses edge detection techniques, such as the Canny edge detection, to find edges.
- Changes in lighting and color usually don't have much effect on image edges
- Strategy:
 - Detect edges in template and image
 - Compare edges images to find the template
 - Must consider range of possible template positions

Appearance Based Methods

Measurements:

- Good – count the number of overlapping edges. Not robust to changes in shape
- Better – count the number of template edge pixels with some distance of an edge in the search image
- Best – determine probability distribution of distance to nearest edge in search image (if template at correct position). Estimate likelihood of each template position generating image

Appearance Based Methods

2) Divide-and-Conquer search

- Strategy:
 - Consider all positions as a set (a cell in the space of positions)
 - Determine lower bound on score at best position in cell
 - If bound is too large, prune cell
 - If bound is not too large, divide cell into subcells and try each subcell recursively
 - Process stops when cell is “small enough”

Unlike multi-resolution search, this technique is guaranteed to find all matches that meet the criterion (assuming that the lower bound is accurate)

Appearance Based Methods

Finding the Bound:

- To find the lower bound on the best score, look at score for the template position represented by the center of the cell
- Subtract maximum change from the “center” position for any other position in cell (occurs at cell corners)
- Complexities arise from determining bounds on distance

Appearance Based Methods

3) Greyscale matching

- Edges are (mostly) robust to illumination changes, however they throw away a lot of information
- Must compute pixel distance as a function of both pixel position and pixel intensity
- Can be applied to color also

4) Gradient matching

- Another way to be robust to illumination changes without throwing away as much information is to compare image gradients
- Matching is performed like matching greyscale images

Limitation

A major limitation of the appearance-based approaches is that they require isolation of the complete object of interest from the background.

They are thus sensitive to occlusion and require good segmentation.

A number of attempts have been made to address recognition with occluded or partial data.

Invariants Introduction

- One example for the importance of invariance in image recognition is depicted in Figure.
- Here an observation pattern is shown, which contains the object of a handwritten digit ‘7’.
- If it is compared with the two references on the right side, a classifier based on Euclidean distance would find that it is closer to the first reference, showing an image of the digit ‘9’, because the sum of squared gray value differences is smaller than the one for the second, ‘correct’ reference.

Invariants Introduction



- Note that if a sufficiently large set of training data is available, it would probably contain also versions of the digit ‘7’ with modified line thickness, such that the advantage of invariance would be reduced.

Invariants Introduction

- There is an inherent connection between invariant recognition and image registration.
- The term registration refers to the mapping of images with the same or nearly identical content onto each other, such that the important structures are in the same image position.
- This is an important paradigm for example in medical imaging, when images have to be compared that were produced at different points in time.
- Usually, registration assumes images of the same content.

Invariants Introduction

- But when a powerful registration algorithm is at hand, it can be applied to invariant recognition, by using it for normalization or determining the mapping function, hypothesizing each class in question and comparing the results.
- On the other hand, when an invariant classification algorithm is known that can return the transformation that connects two given patterns (which is the case for tangent distance), the registration problem is solved as a by-product.

Invariant Classification

- There exists a variety of techniques for solving the problem of invariant pattern recognition: “Such techniques include integral transforms, construction of algebraic moments and the use of structured neural networks.
- In all cases we assume that the nature of the invariance group is known *a priori*.”
- The last statement is quite essential in most approaches.
- One approach not mentioned here is the use of invariant distance measures, which play an important role for this work.

Invariant Classification

- Consider patterns as functions on some set, e.g. in image recognition $x : (i, j) \in I \times J \rightarrow x_{ij}$, furthermore there exists a classification function which maps patterns onto class numbers,
e.g. $r : x \rightarrow 1, \dots, K$
- and a transformation group G which acts on the set the pattern is defined on and therefore on the pattern space, e.g. $g \in G : (gx)_{ij} = x_{g^{-1}(i,j)}$ and does not affect class membership.

Invariant Classification

Thus the desired classification function should be invariant under the action of the group, that is
 $r(gx) = r(x) \forall g \in G.$

That is, the patterns with the same invariant content form an equivalence class with respect to a group operation describing the geometric transform.

In practice, in some cases one may want to restrict the actions of the group, e.g. in digit recognition in order to distinguish between the digits '6' and '9'.

This is sometimes referred to as 6-9-problem

Invariant Classification : Normalization

With the term normalization one usually refers to the construction of a canonical representation for each pattern with respect to the regarded transformations.

These representations can then be compared without the influence of the differences of the transformations.

One drawback of such methods is that they may be very sensitive to noise and artifacts in the patterns.

For example one may use the following normalization procedure in order to achieve invariance with respect to rotation, translation and scale for images:

Invariant Classification : Normalization

1. compute the center of gravity and translate the origin to that point (translation-invariance)
2. normalize for average radius (scale-invariance)
3. rotate such that direction of maximum variance coincides with x-axis (rotation-invariance)

Invariant Classification: Invariant Features

If one wants to obtain a classification procedure that is invariant with respect to certain transformations, another approach is to calculate a set of features from the pattern, which is not affected by these transformations but still contains all information relevant for classification.

This ideal view of invariant features can be expressed as: “A complete system of invariants must be able to distinguish with arbitrary precision between any two vectors not in the same orbit under G ; i.e. the system must possess perfect discriminability.”

Invariant Classification: Invariant Distance Measures

While normalization and the extraction of invariant features aim at the elimination of the considered transformations before the actual classification process, invariance can also be incorporated directly into the classifier.

This can be done by using invariant distance measures.

An invariant distance measure would ideally have the property that the distance between two patterns is always equal to the minimum distance between the 'best matching' transformed instances of those patterns.

Invariant Classification: Invariant Distance Measures

Since the orbits that arise from regarding the set of all possible transformations of a pattern form a manifold in pattern space, this ideal invariant distance is called manifold distance.

A definition of a manifold is given for example: A manifold is a locally Euclidean space together with a differential structure.

“One can think about a manifold as a way to piece together “bent” pieces of \mathbb{R}^n .

Thus the manifold has the same local properties as \mathbb{R}^n , but may have different global properties.

Eigenspace

The image set is normalised in brightness and the background noise removed in order to eliminate the redundant information.

The eigenspace for the image set is built by computing the biggest eigenvalues (eigenvectors) of the set.

The next step is to project all the images onto eigenspace and we obtain a set of points that characterise the object and the pose in space.

In order to increase the resolution for position estimation we connected these points in eigenspace and an unknown position can be better approximated.

The first step could be considered by analogy with neural networks as “learning” and the recognition by space partitioning as “matching”.

Eigenspace

An image $I(x,y)$ is represented by a two-dimensional 256 by 256 array of 8-bit intensity values.

This image can also be considered as a vector of dimension 65536.

This is obtained by reading pixel brightness values in a raster scan manner.

To reconstruct an image we map a collection of points (in our case eigenvalues) in this huge space.

Eigenspace

The main idea of the principal component analysis (Karhunen-Loeve expansion) is to find the vectors that can better describe the image (or the entire space).

These vectors describe an orthonormal space and this property is presented below:

$$u_i u_j^T = \gamma_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

where u_i , u_j are two eigenvectors and γ_{ij} is the scalar product.

Thank u!!!