Unit 3
Feature Detection

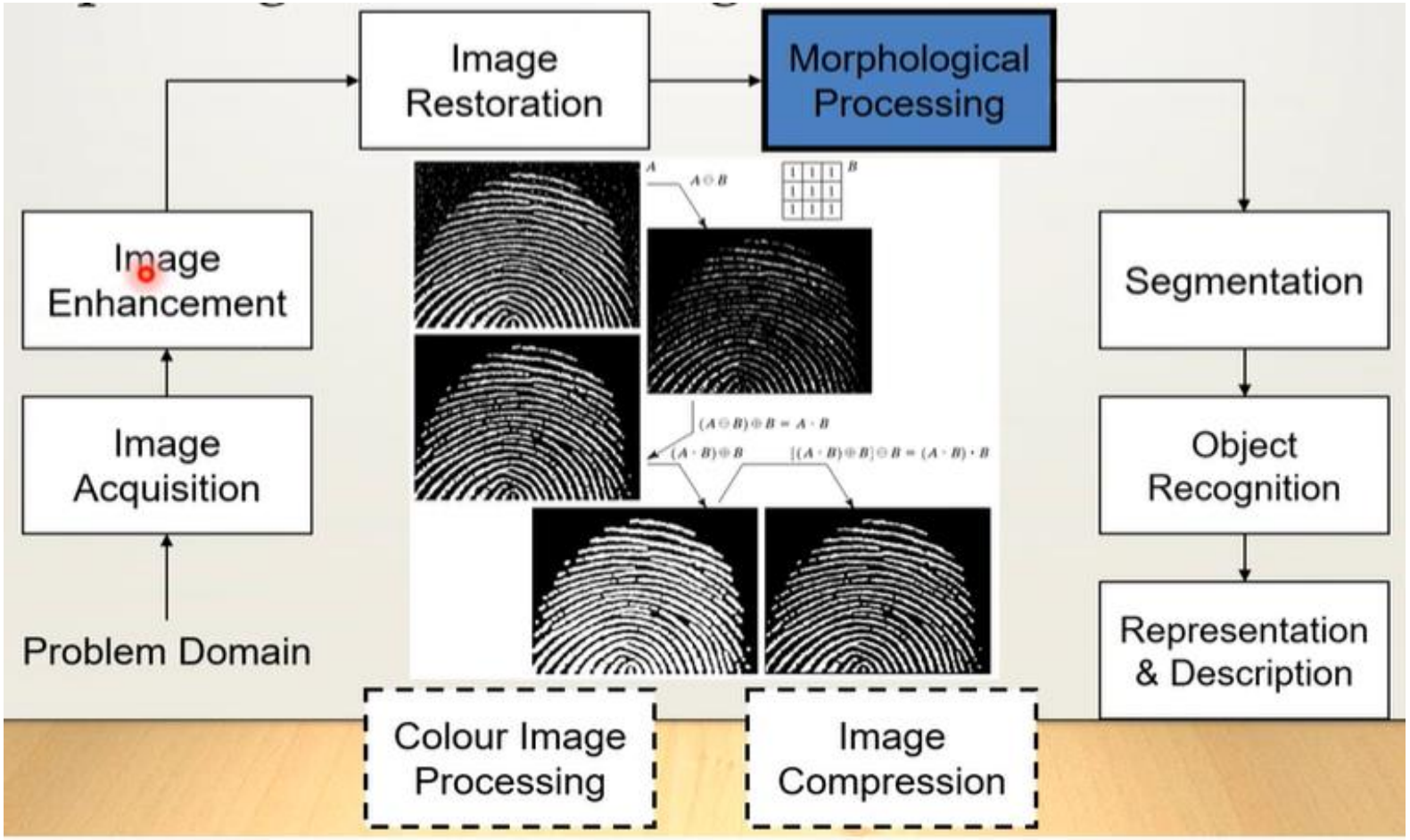
Prof. Janki Patel
Department of IT
SPCE

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



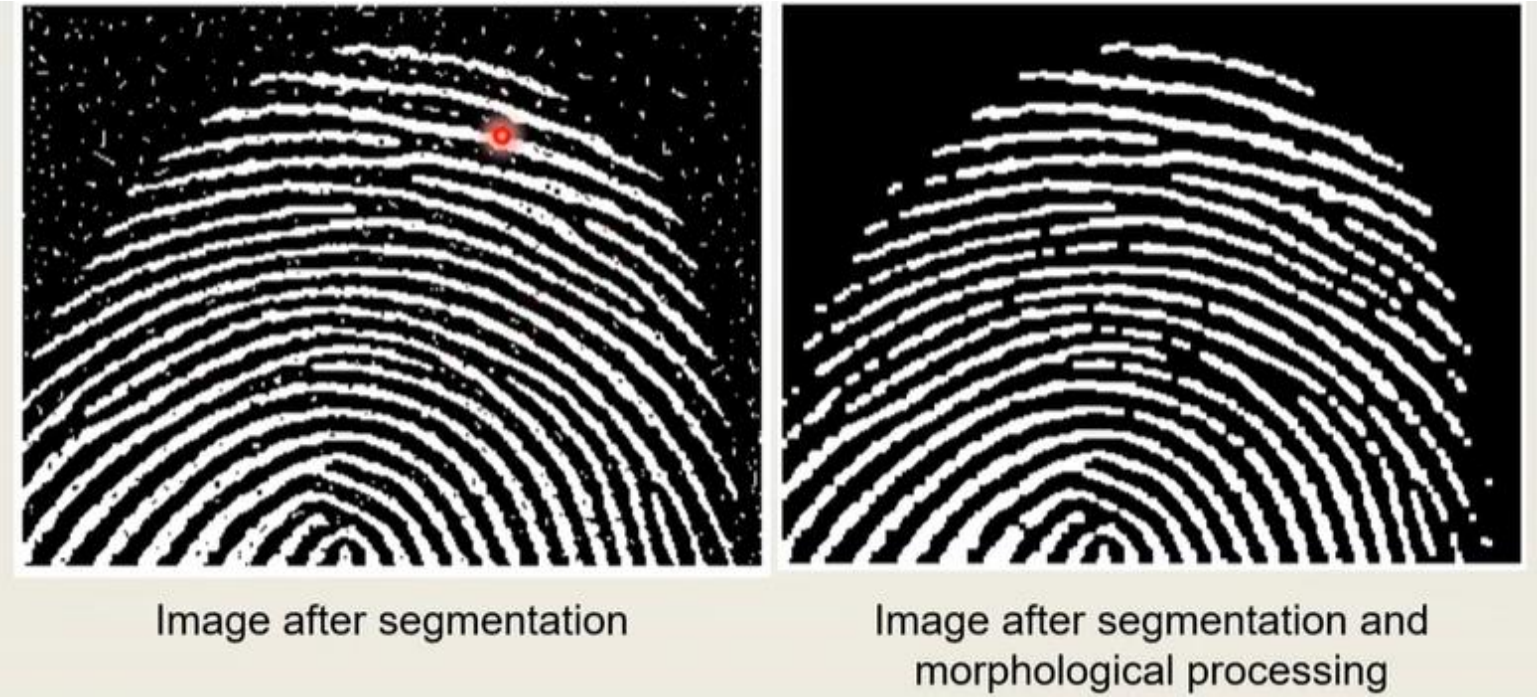
1, 0, Black, White?

- 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….

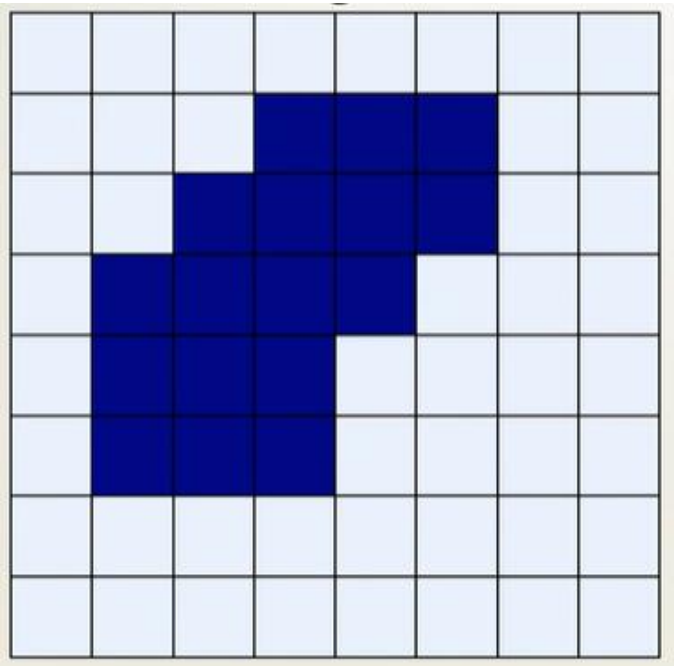
What is Morphology?

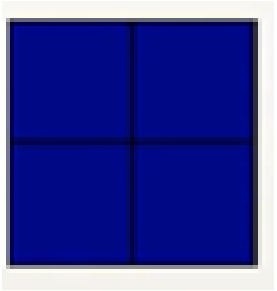
- Morphological image processing(or morphology) describes a range of image processing techniques that deal with the shape(or morphology) of features in an image
- Morphological operations are typically applied to remove imperfections introduced during segmentation and so typically operate on bi-level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

Quick Example



Structuring Elements , Hits &Fits



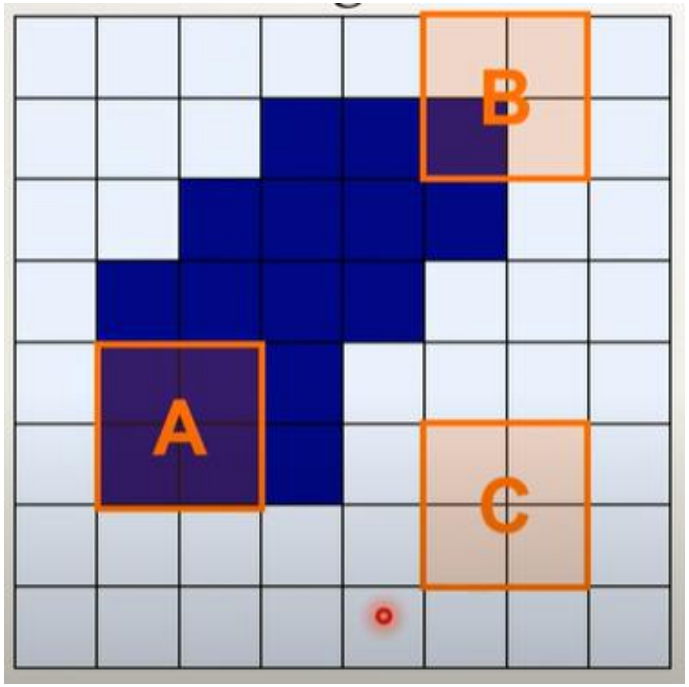


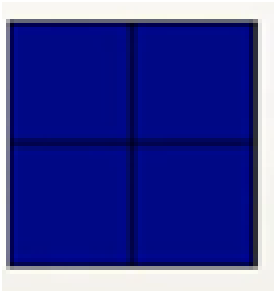
Structuring Element

- 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





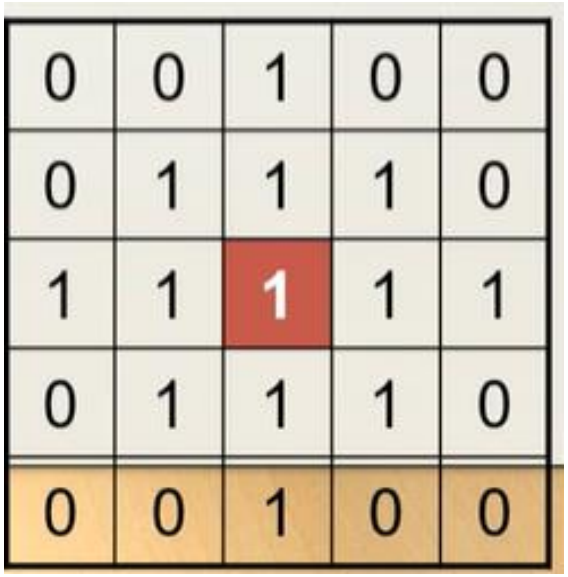
Structuring Element

- 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

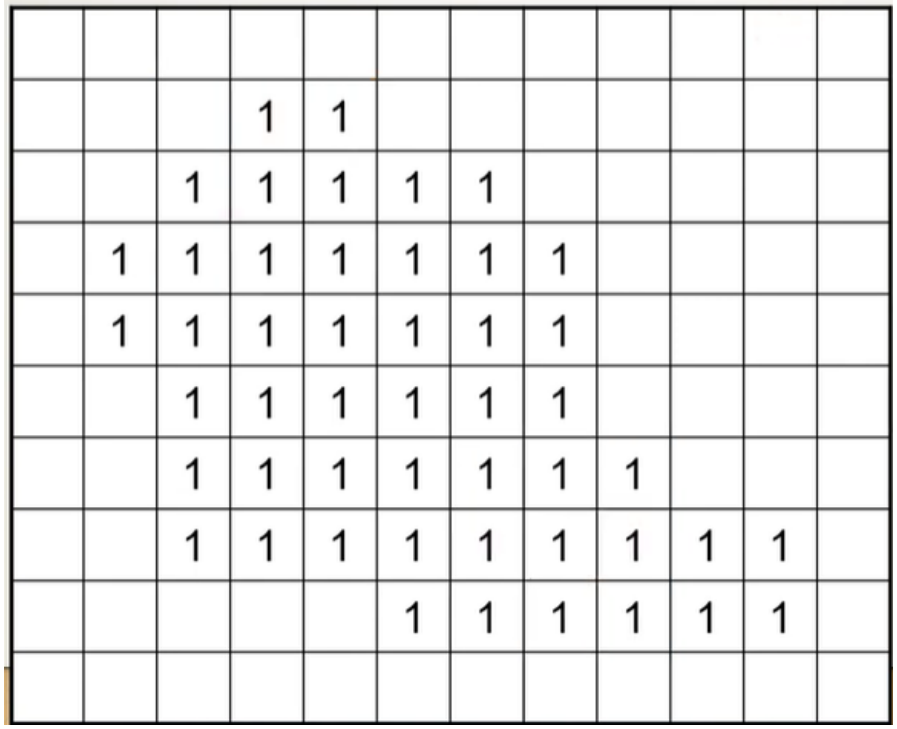
- Structuring elements can be of any size and make any shape
- However, for simplicity we will use rectangular structuring elements with their origin at the middle pixel

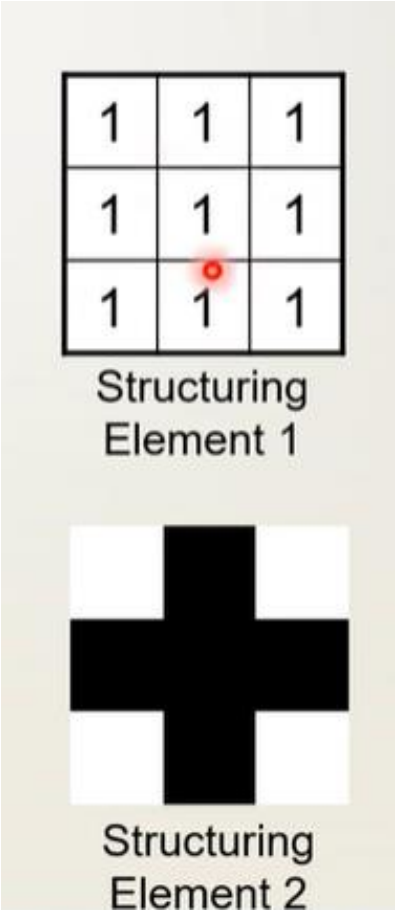


What is use of Hit, Miss and Fit?

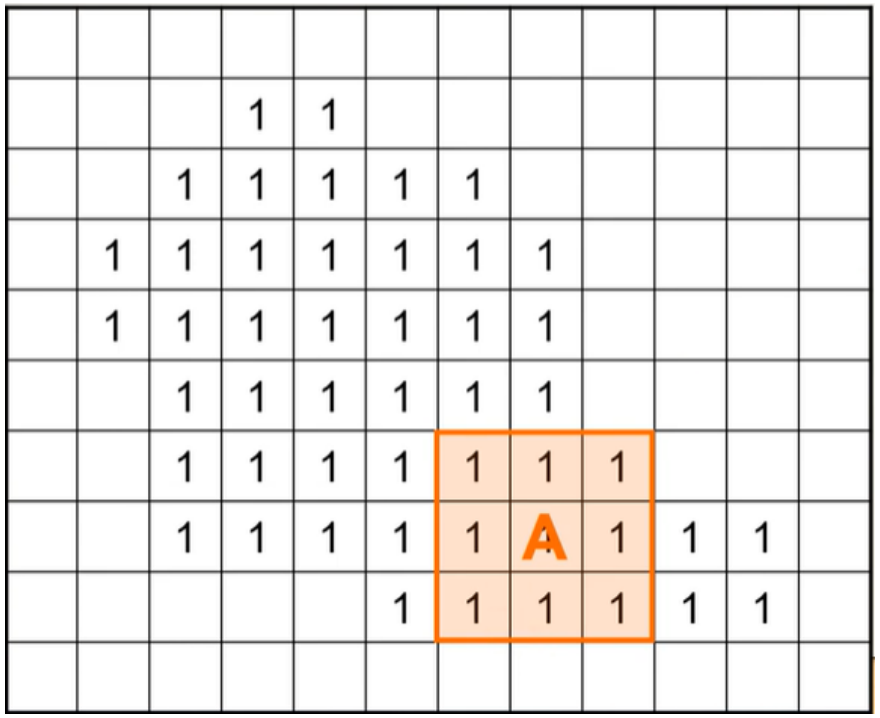
- Hit and miss algorithm can be used to thin and skeletonize a shape in a binary image.

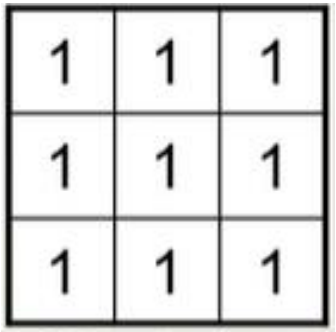
Fitting & Hitting





Fitting & Hitting

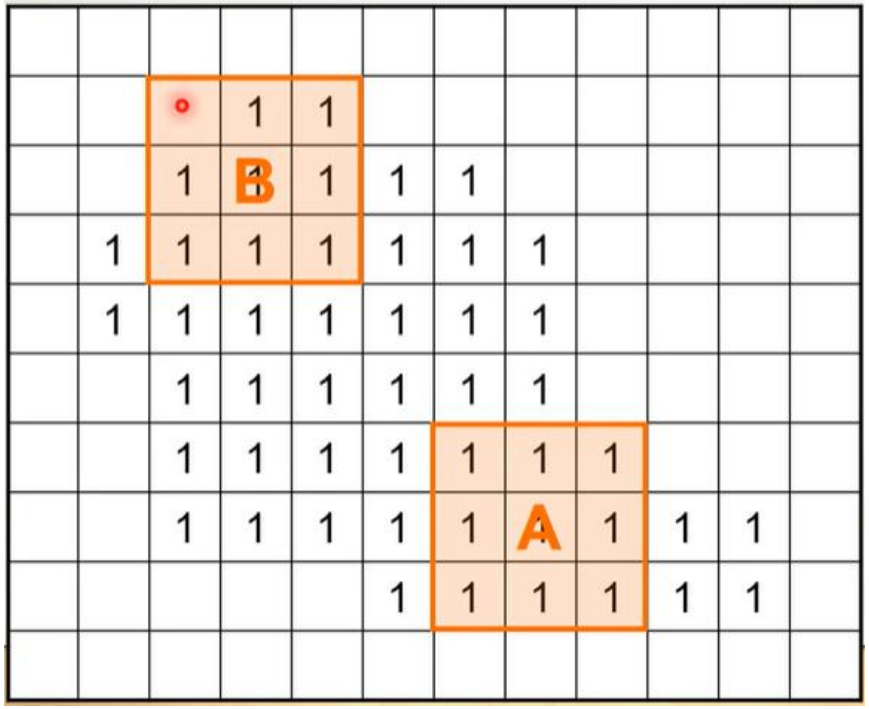


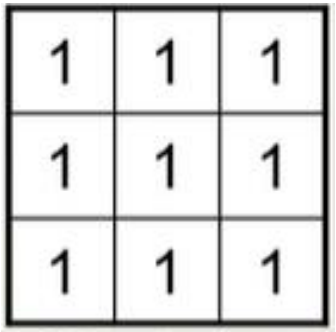


Structuring Element 1

Structuring Element 2

Fitting & Hitting

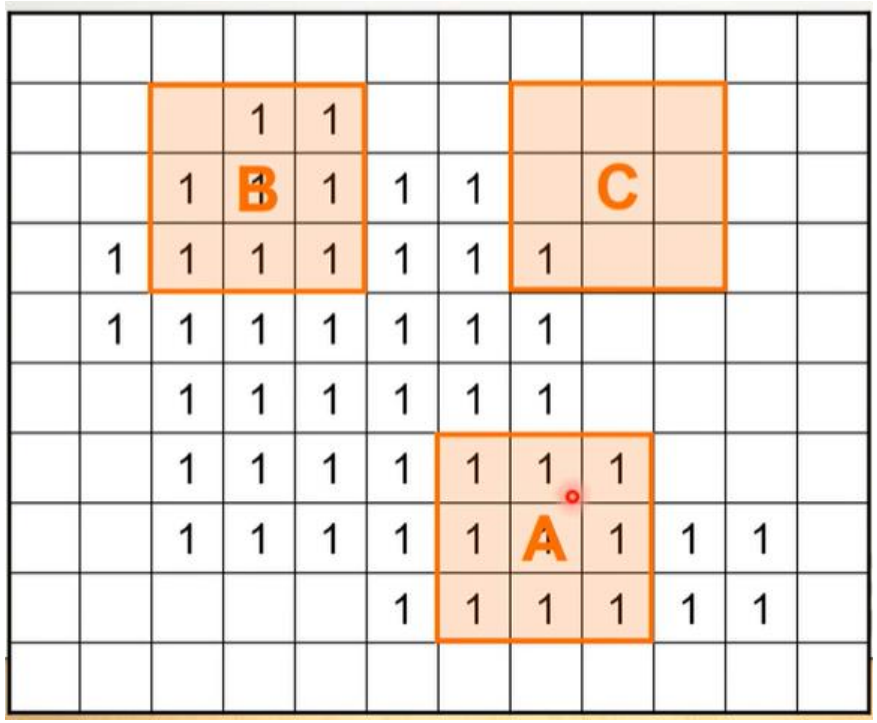


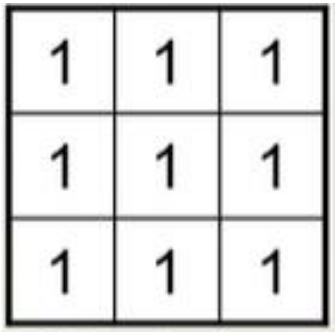


Structuring Element 1

Structuring Element 2

Fitting & Hitting





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**

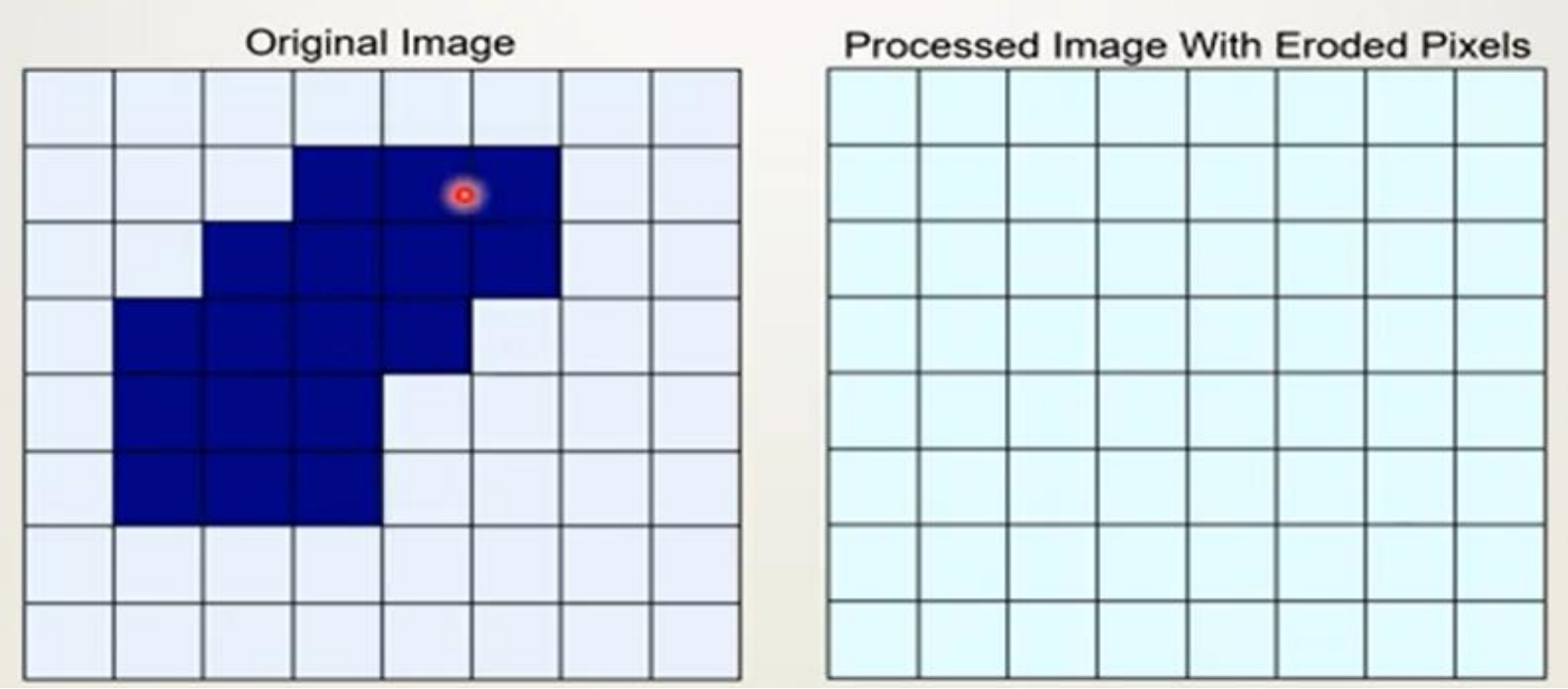
Erosion

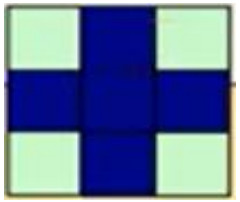
Erosion

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

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

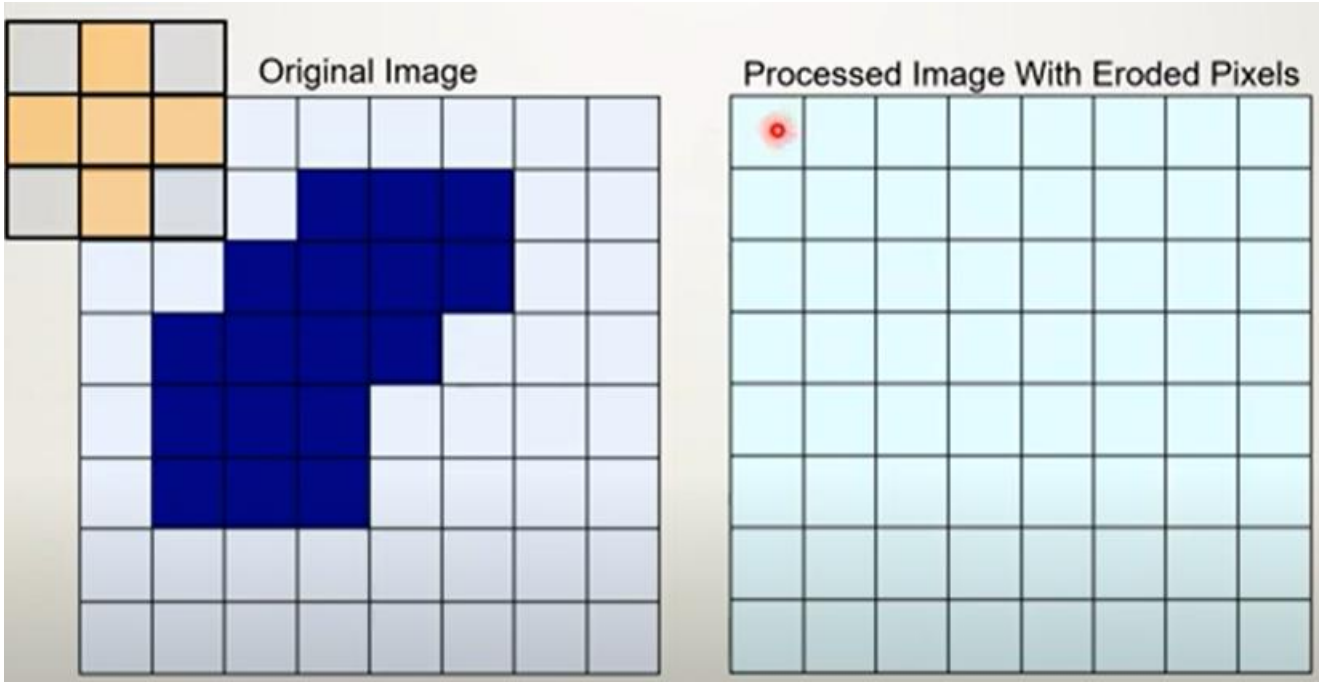
Erosion Example

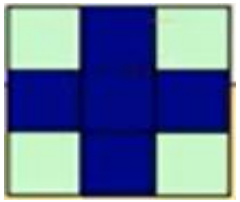




Structuring Element

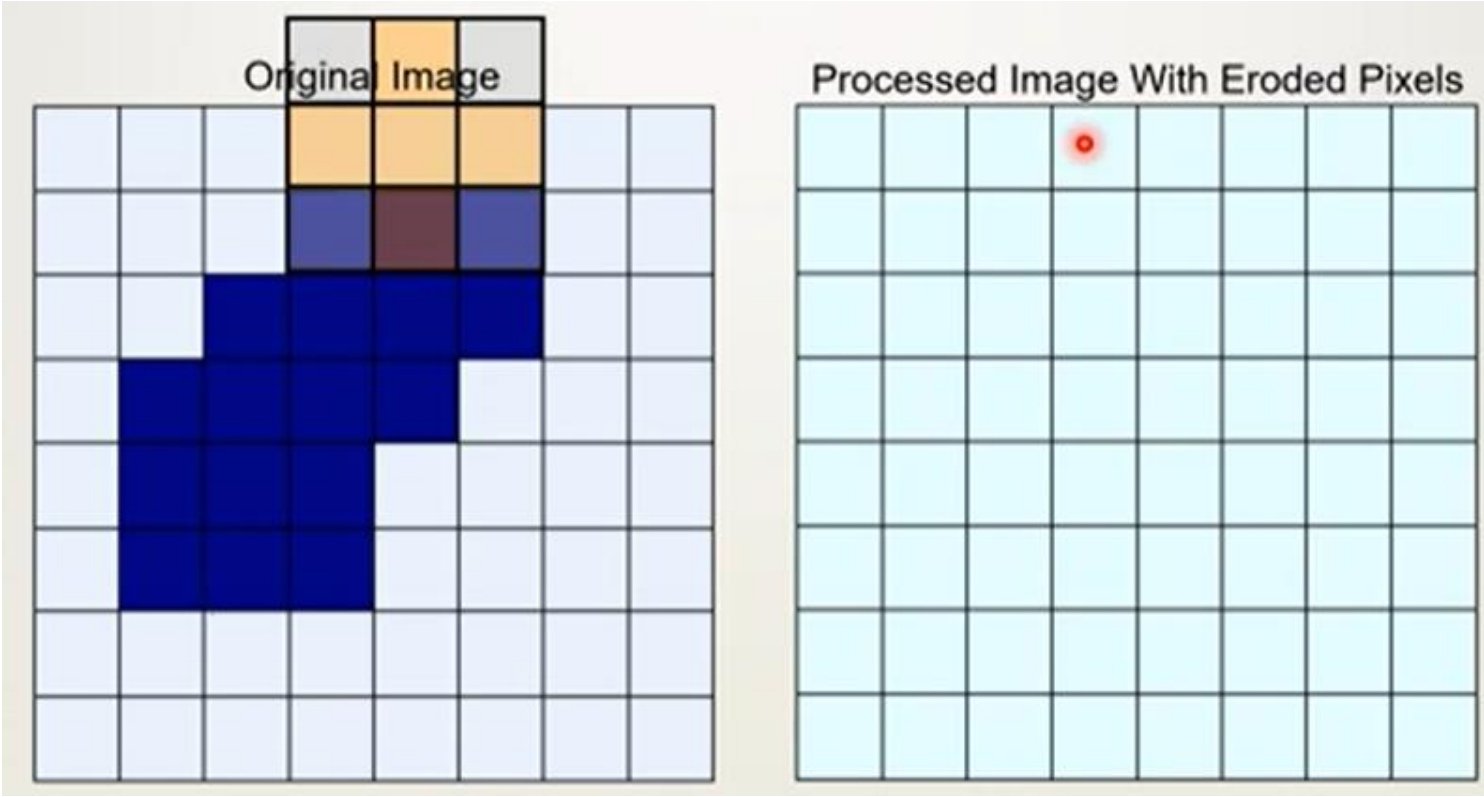
Erosion Example

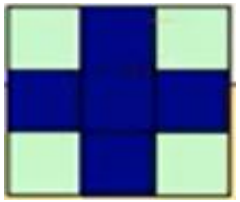




Structuring Element

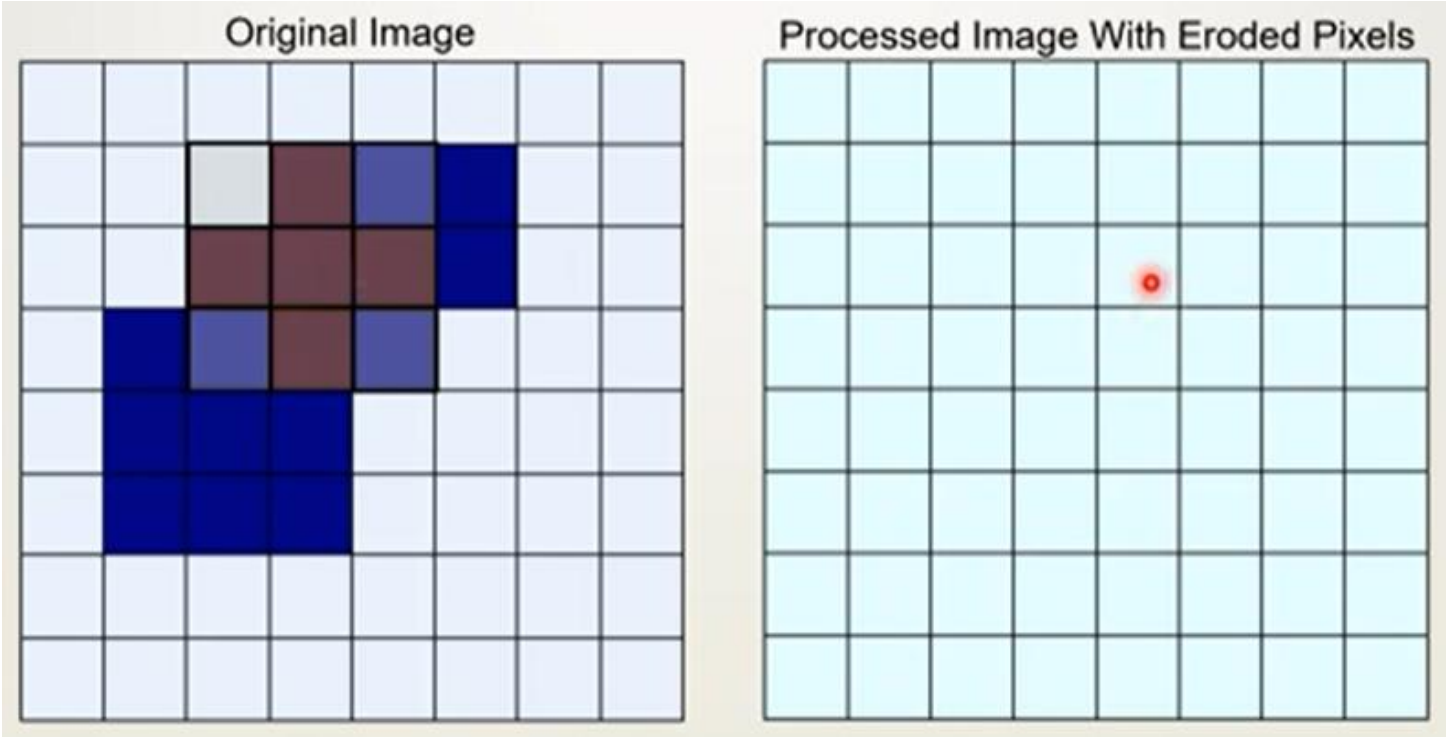
Erosion Example

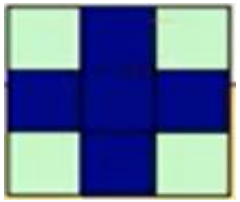




Structuring Element

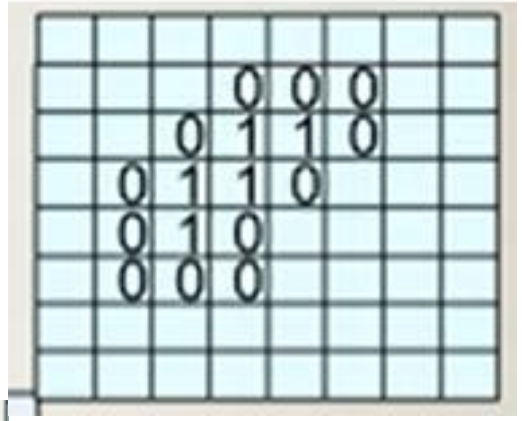
Erosion Example

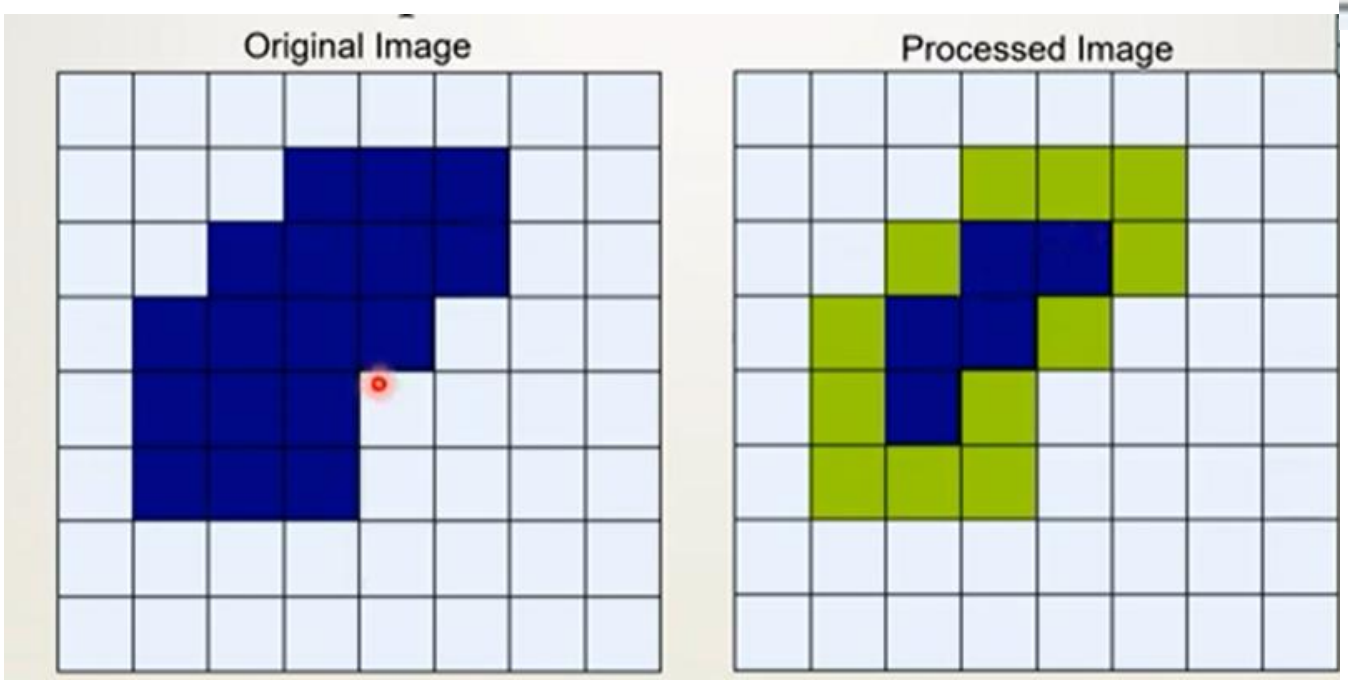


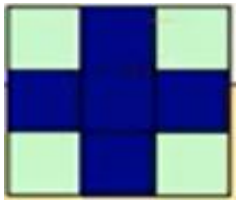


Structuring Element

Erosion Example

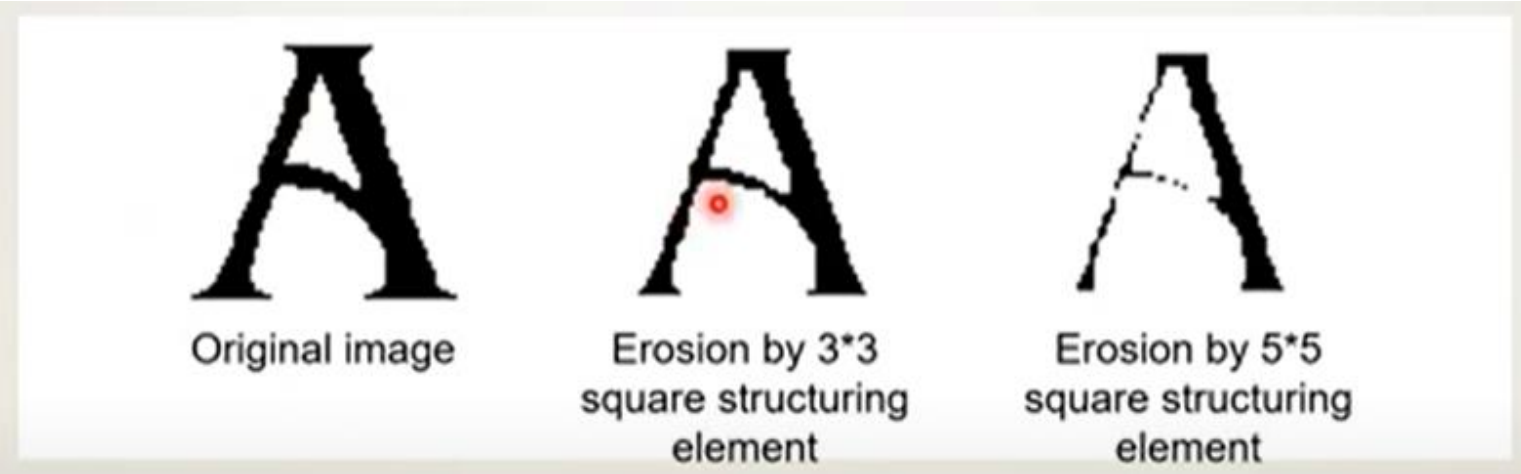






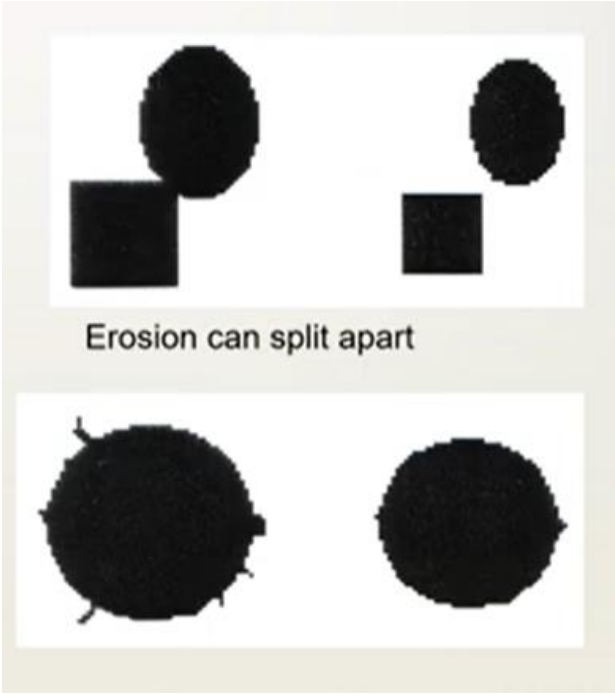
Structuring Element

Erosion Example 1



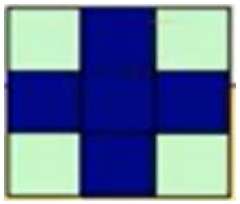
What is Erosion For?

- 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	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

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	0	0	0	0	0	0	0
0	0	0	0	**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	0
0	0	0	0	0	0	0	0	0	0

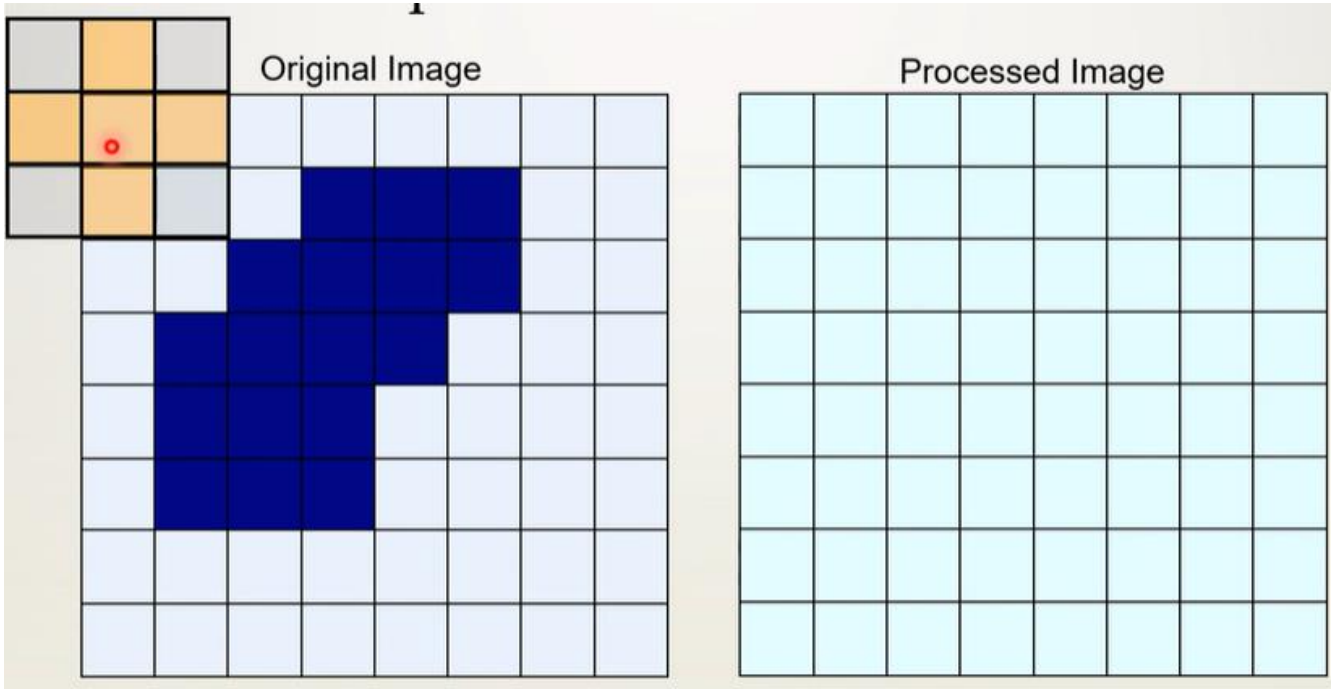
Dilation

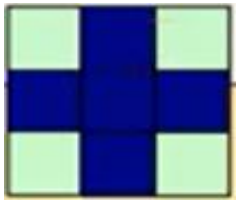
Dilation

- 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:

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

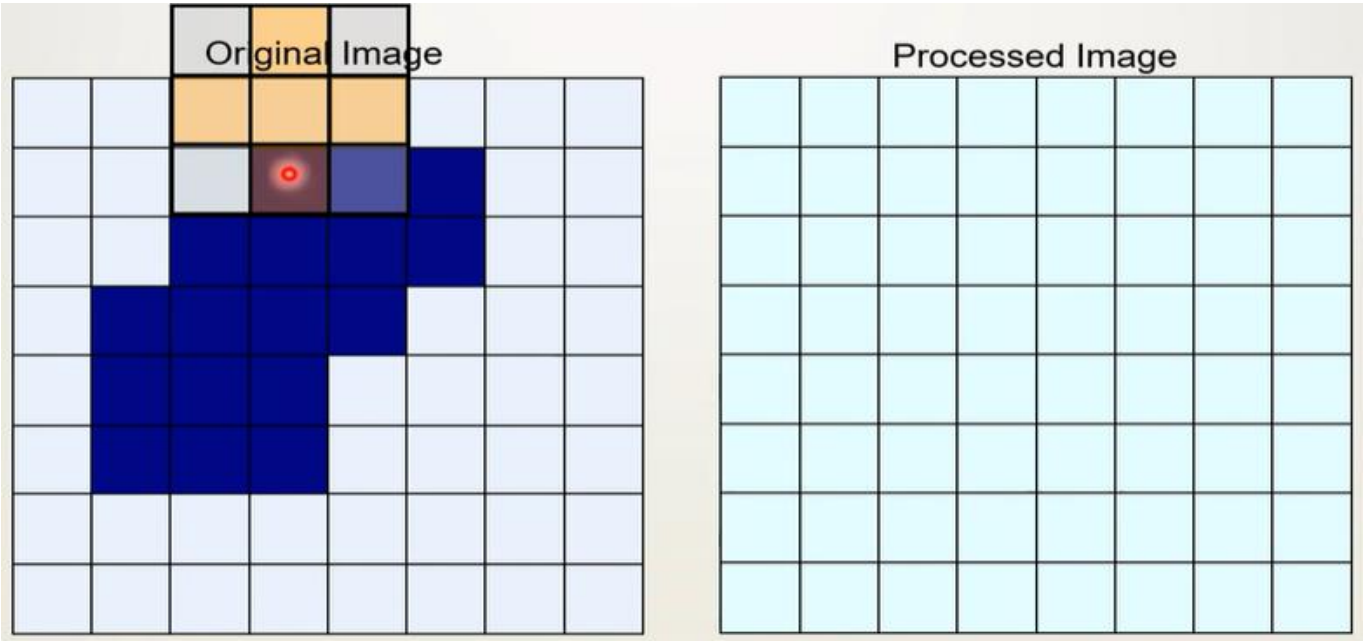
Dilation Example

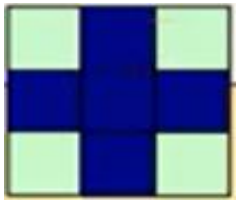




Structuring Element

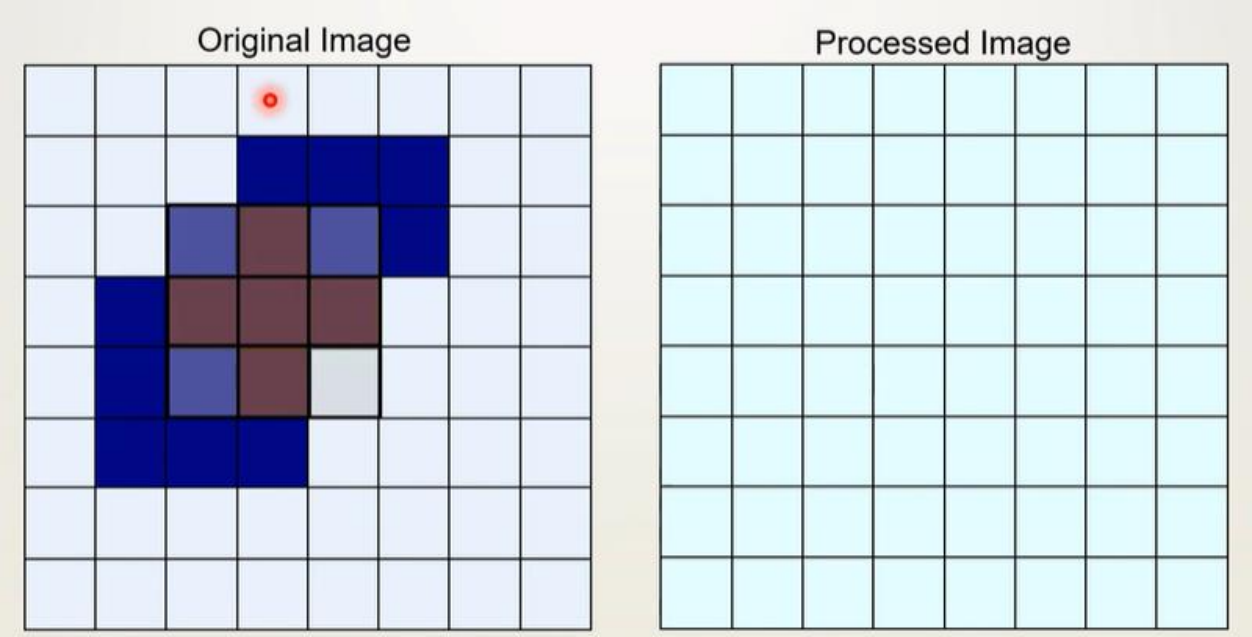
Dilation Example

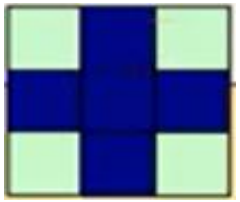




Structuring Element

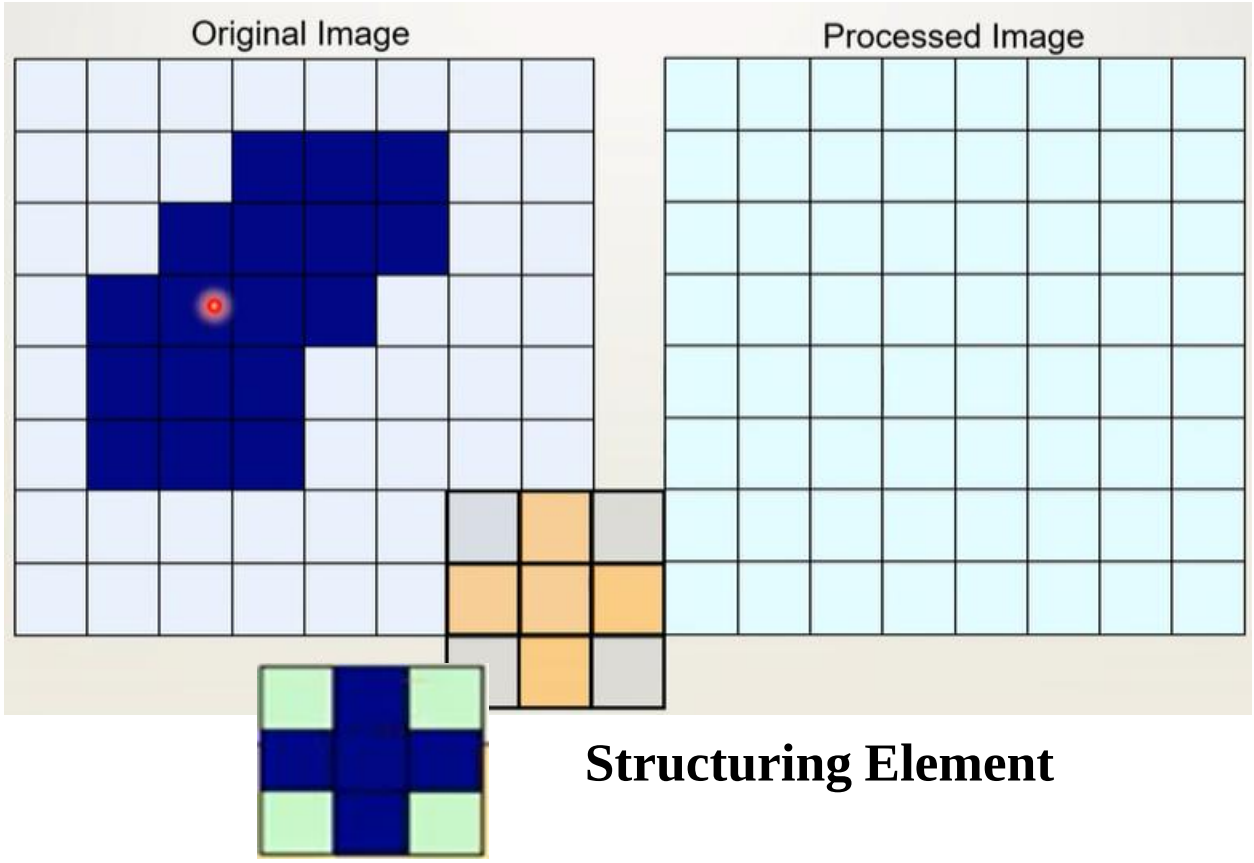
Dilation Example



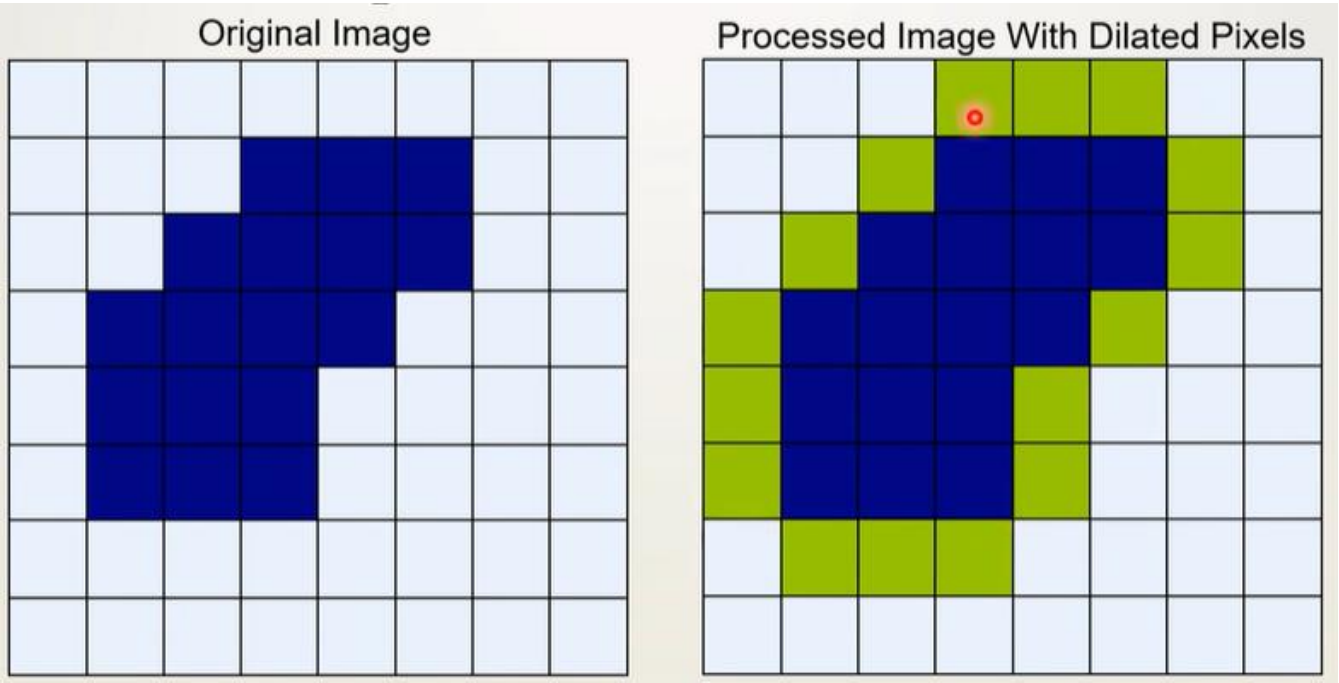


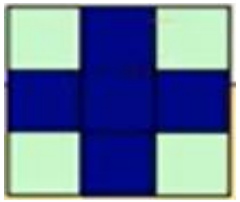
Structuring Element

Dilation Example



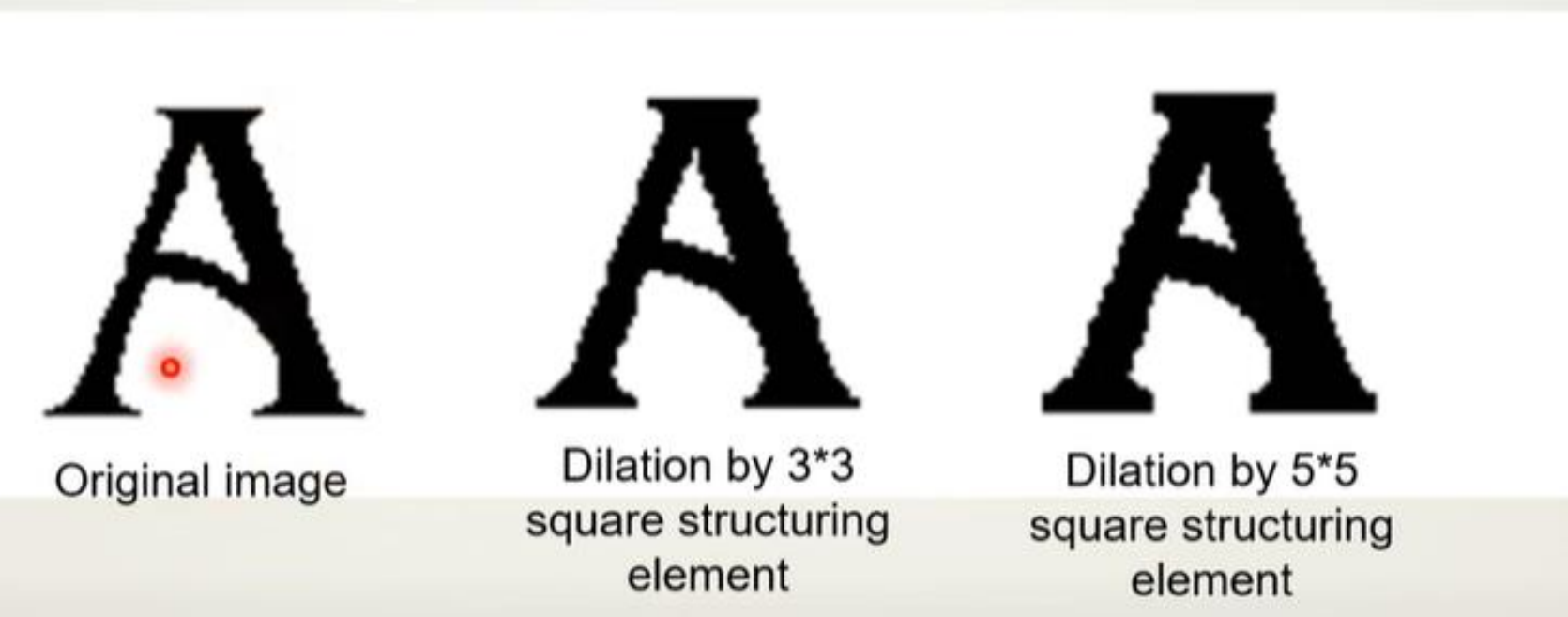
Dilation Example



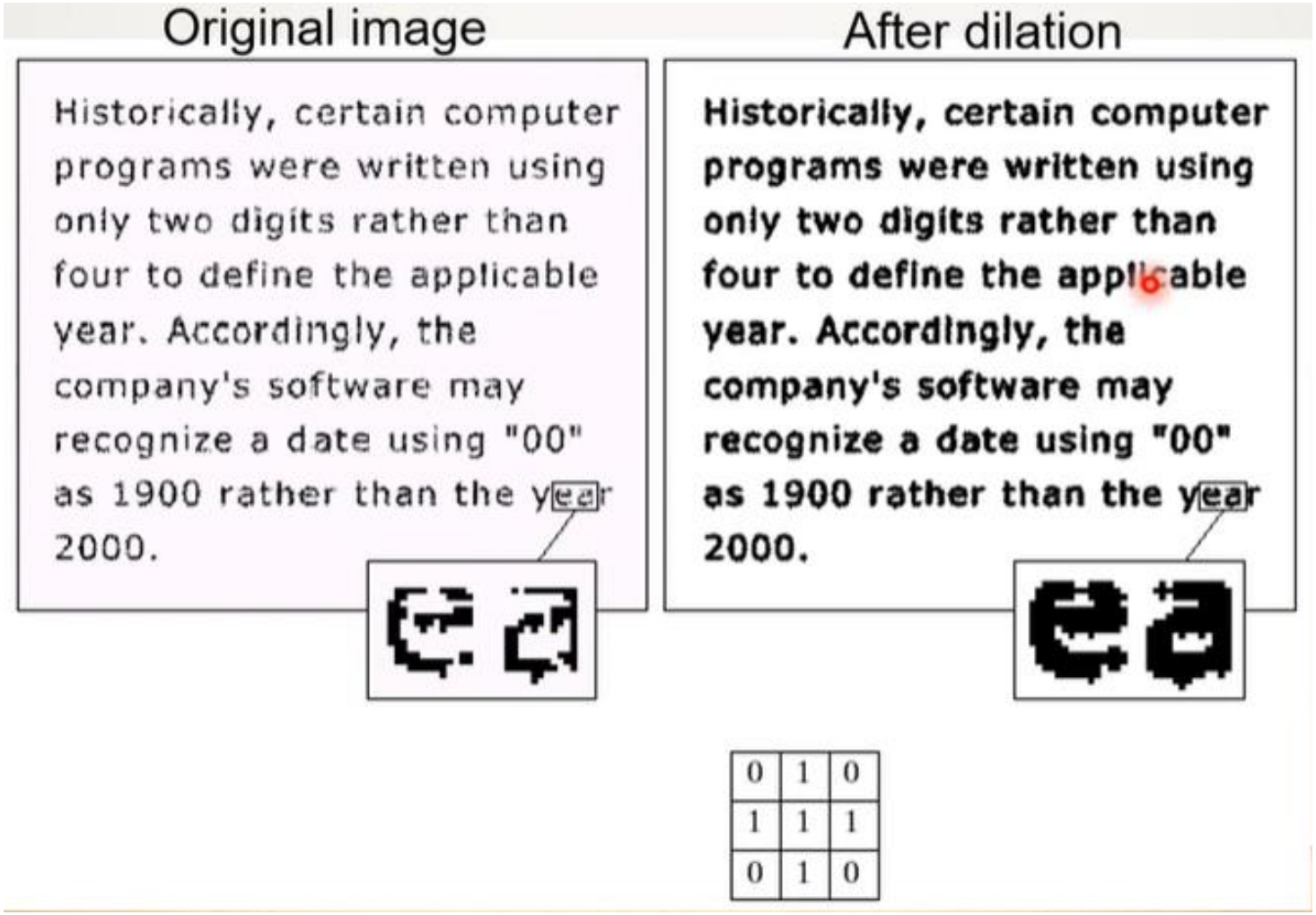


Structuring Element

Dilation Example 1



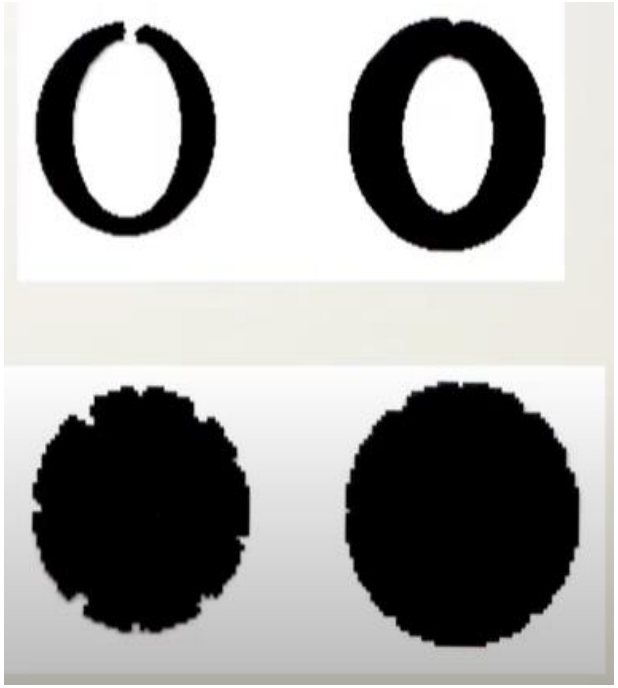
Dilation Example 2



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

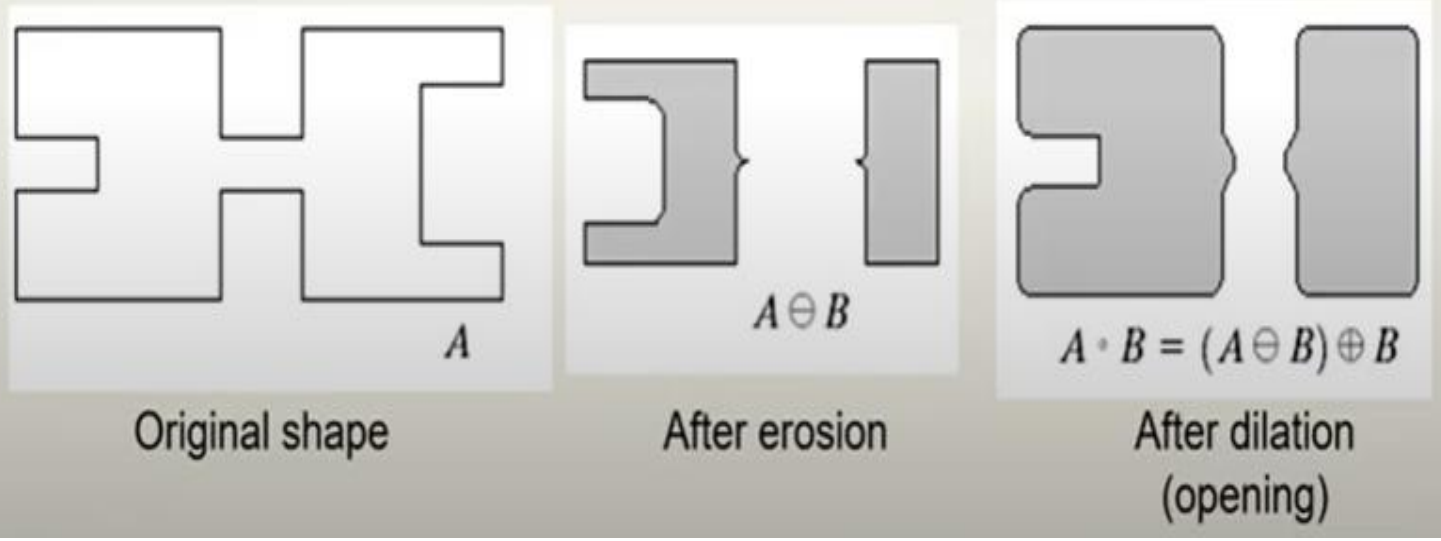
- More interesting morphological operations can be performed by performing combinations of erosion and dilations
- The most widely used of these compound operations are:
  1. Opening
  2. Closing

Opening

Opening

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

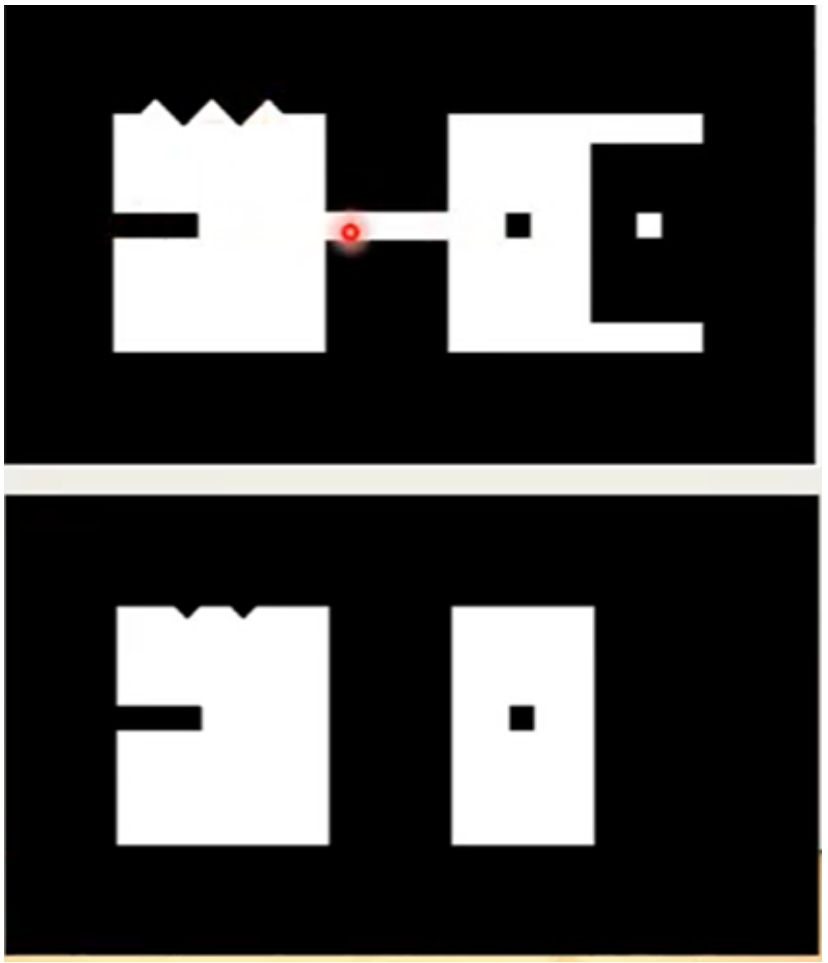
$$f \circ s = (f \ominus s) \oplus s$$



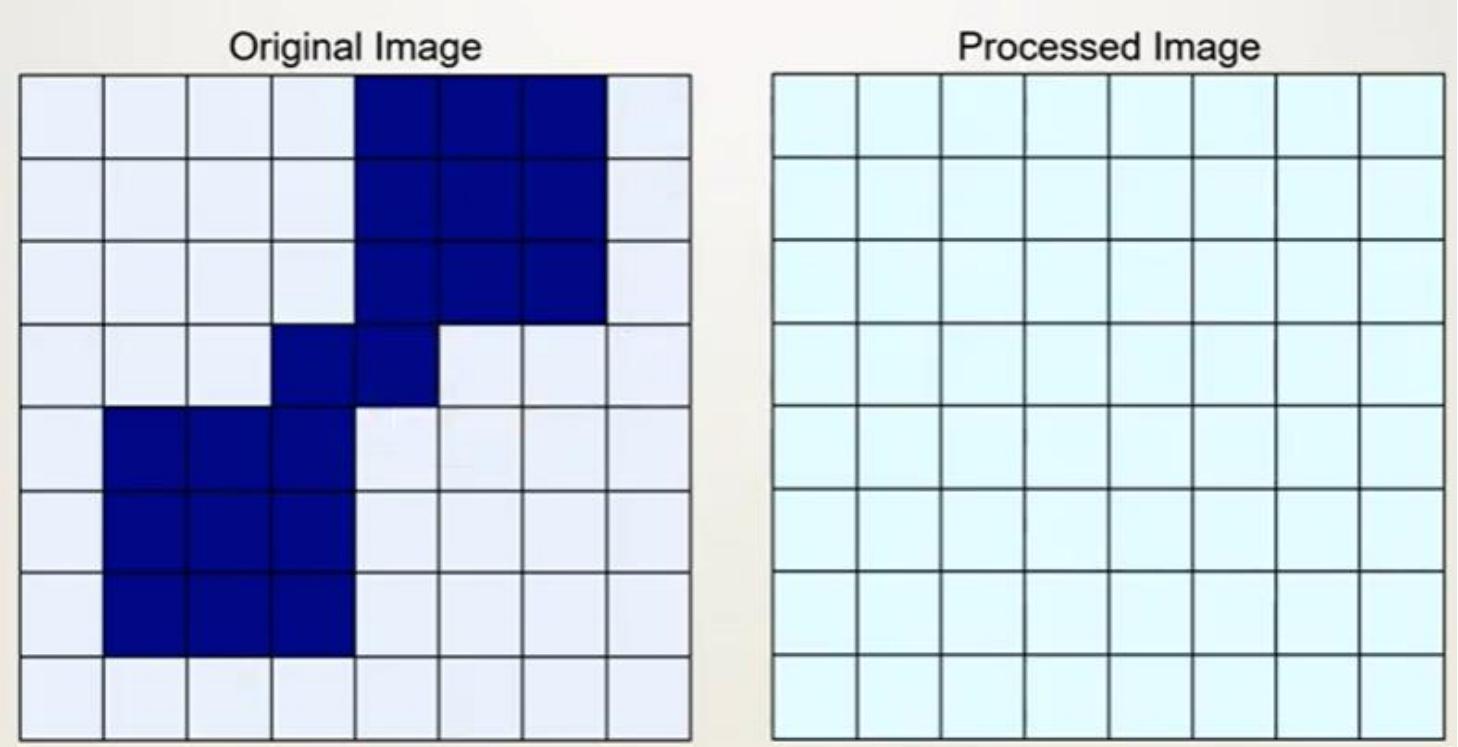
Opening Example

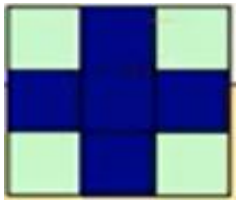
Original Image

Image After Opening



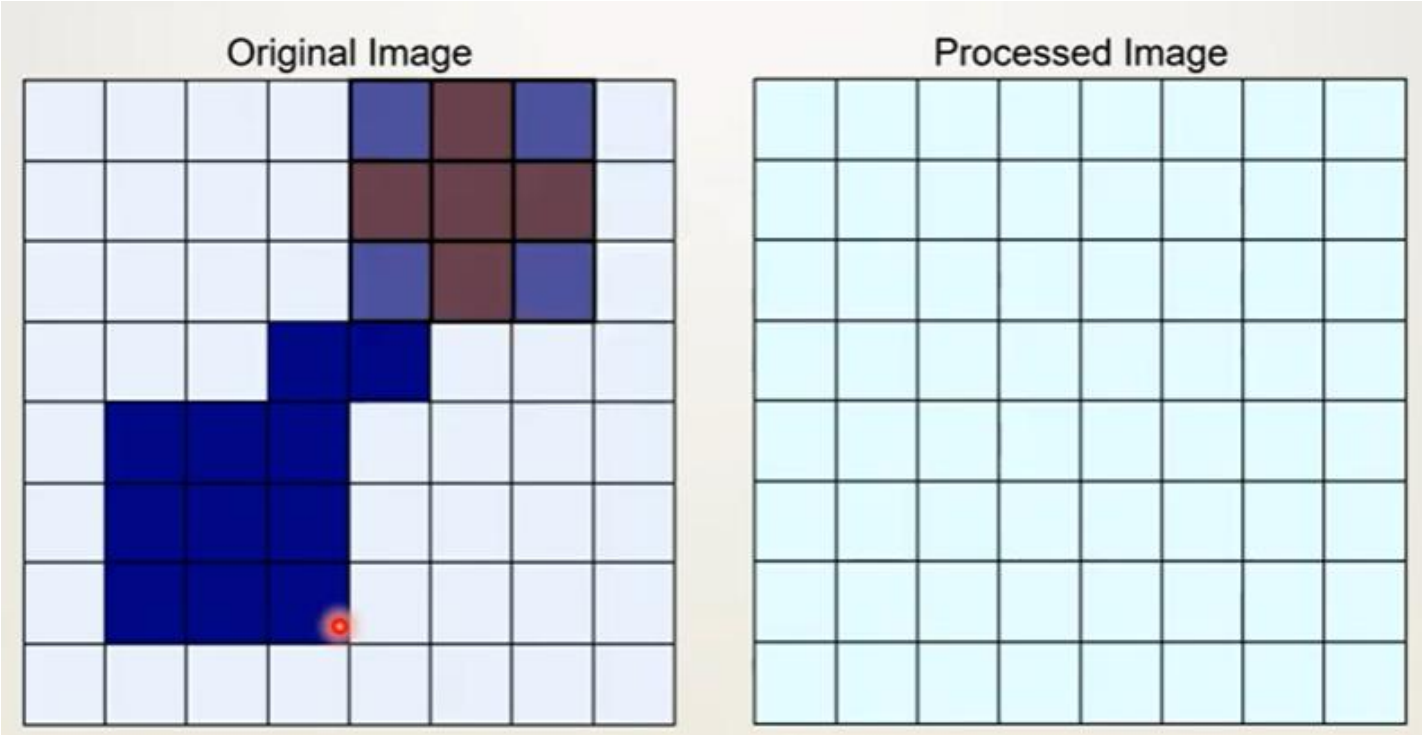
Opening Example

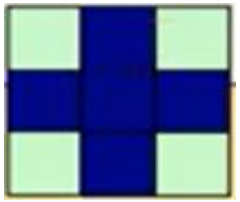




Structuring Element

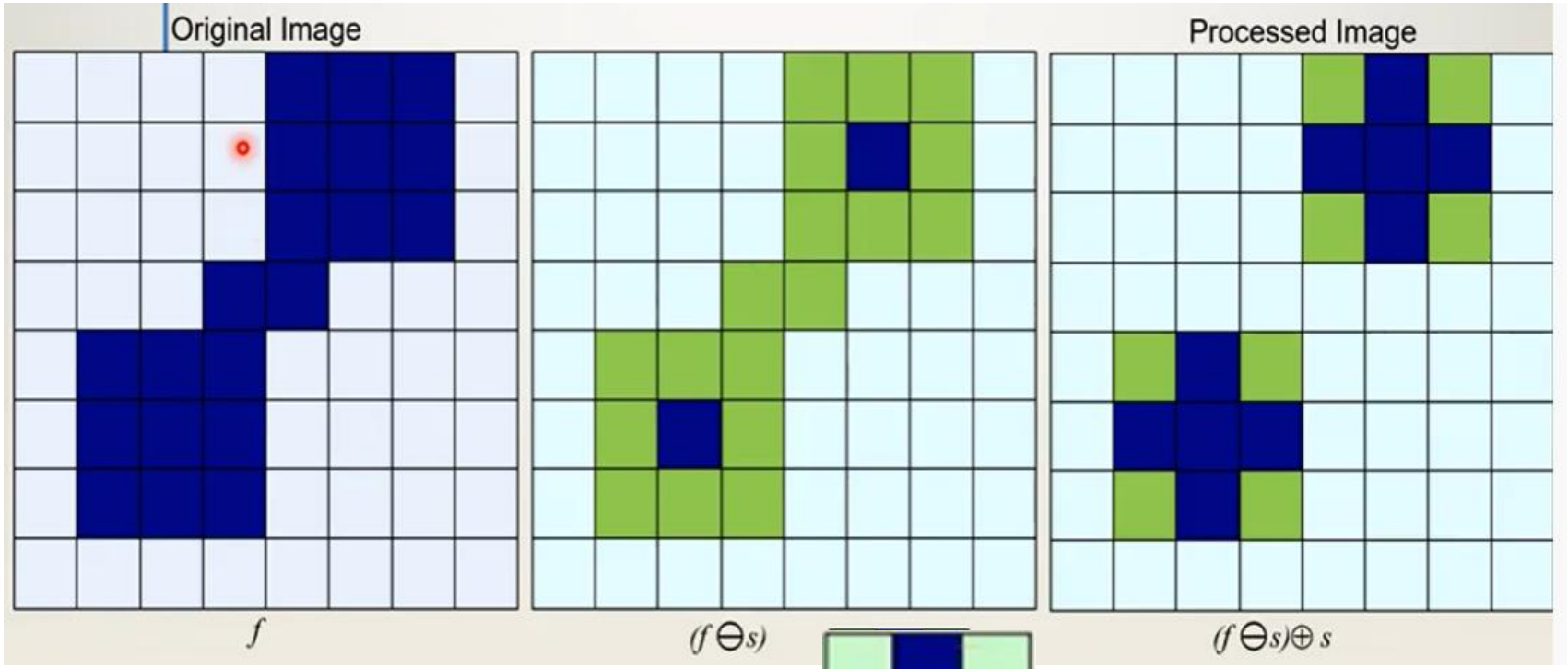
Opening Example



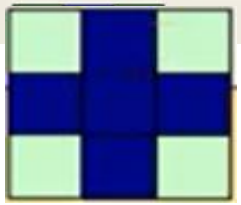


Structuring Element

Opening Example



Structuring Element



Advantages of opening

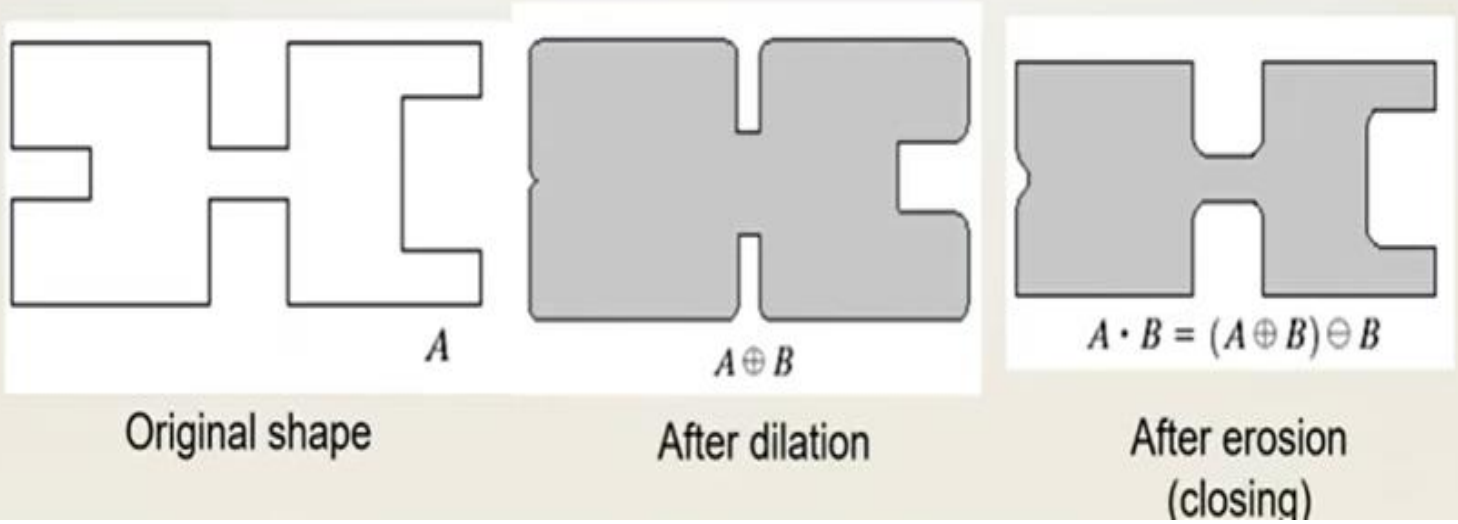
- Opening generally smoothing the contour of an object
- Breaks narrow isthmuses
- And eliminates thin protrusion

Closing

Closing

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

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

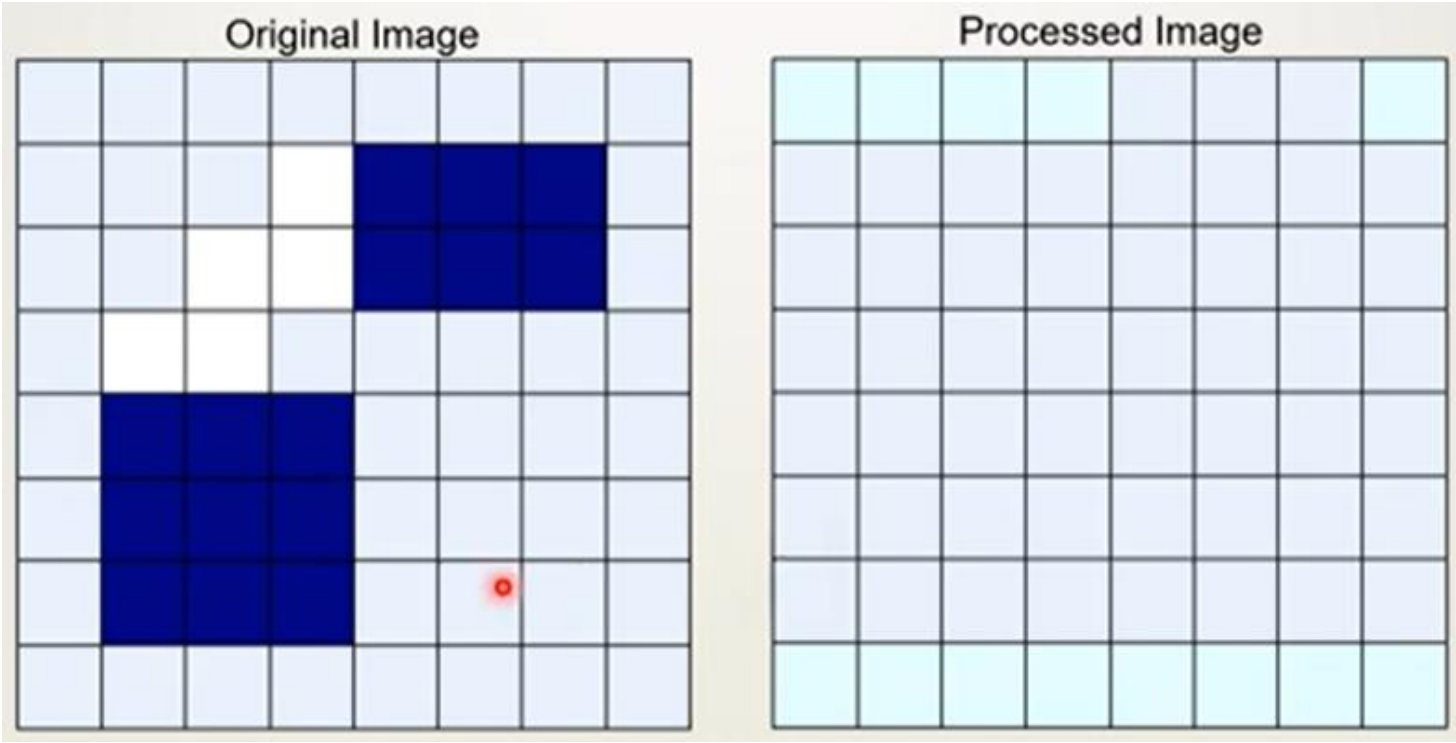


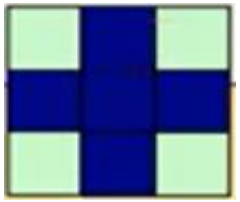
Closing Example

Original Image

Image After Closing

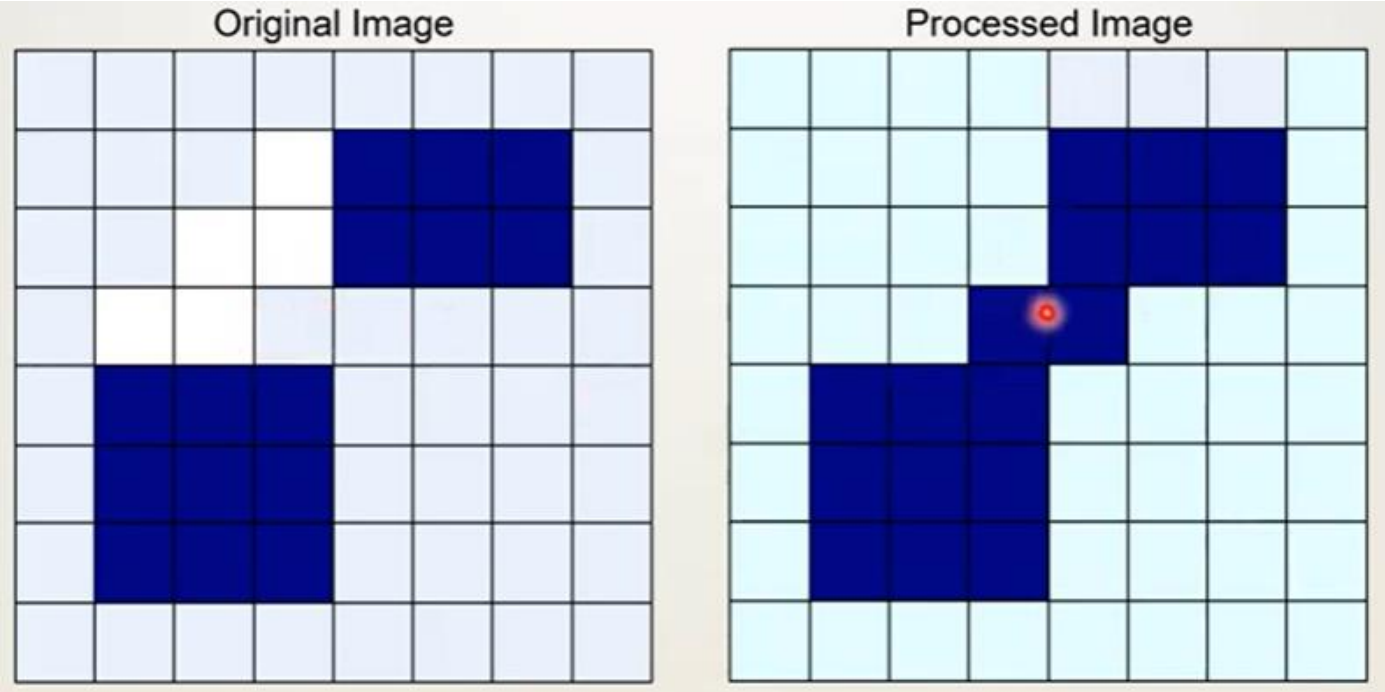
Closing Example

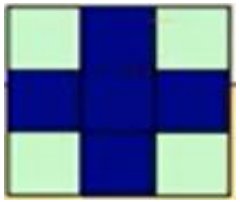




Structuring Element

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.

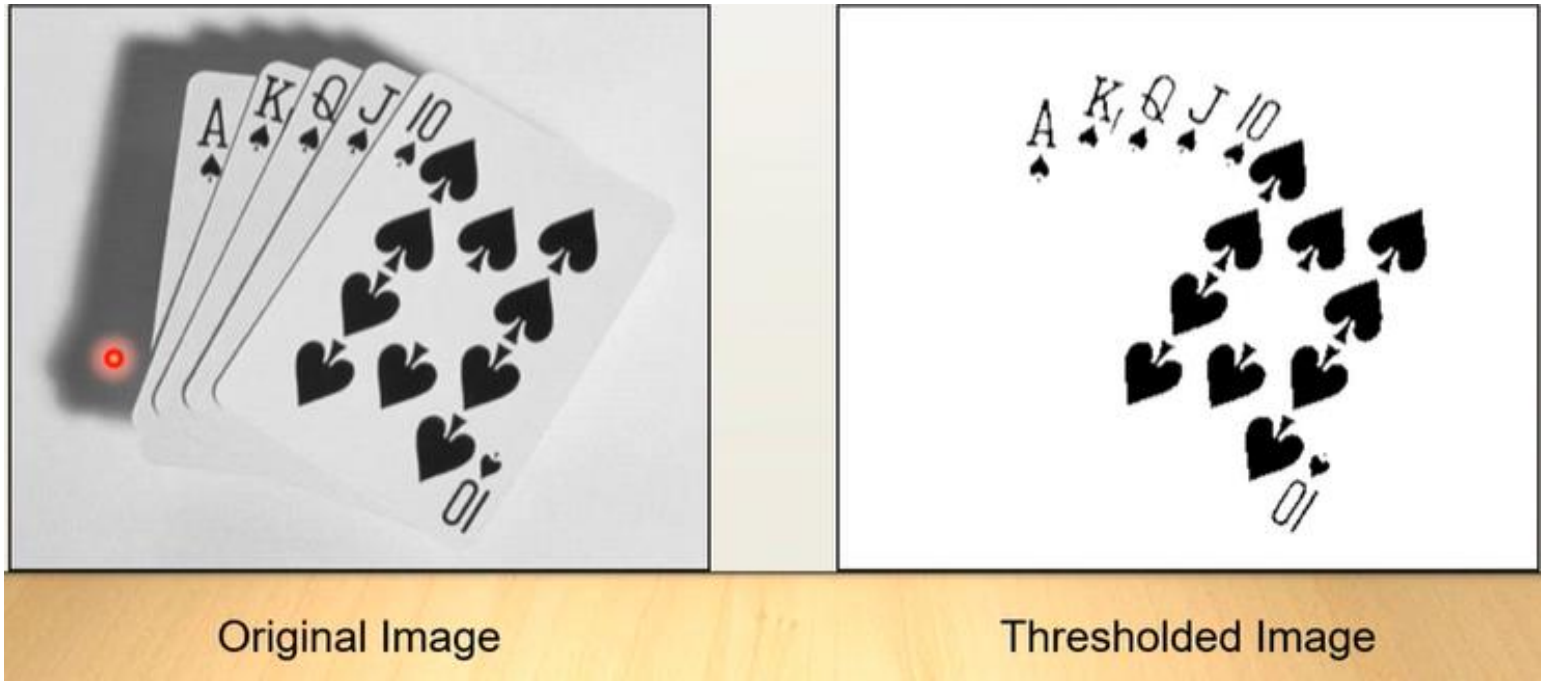
Thresholding

- Single value thresholding can be given mathematically as follows:

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

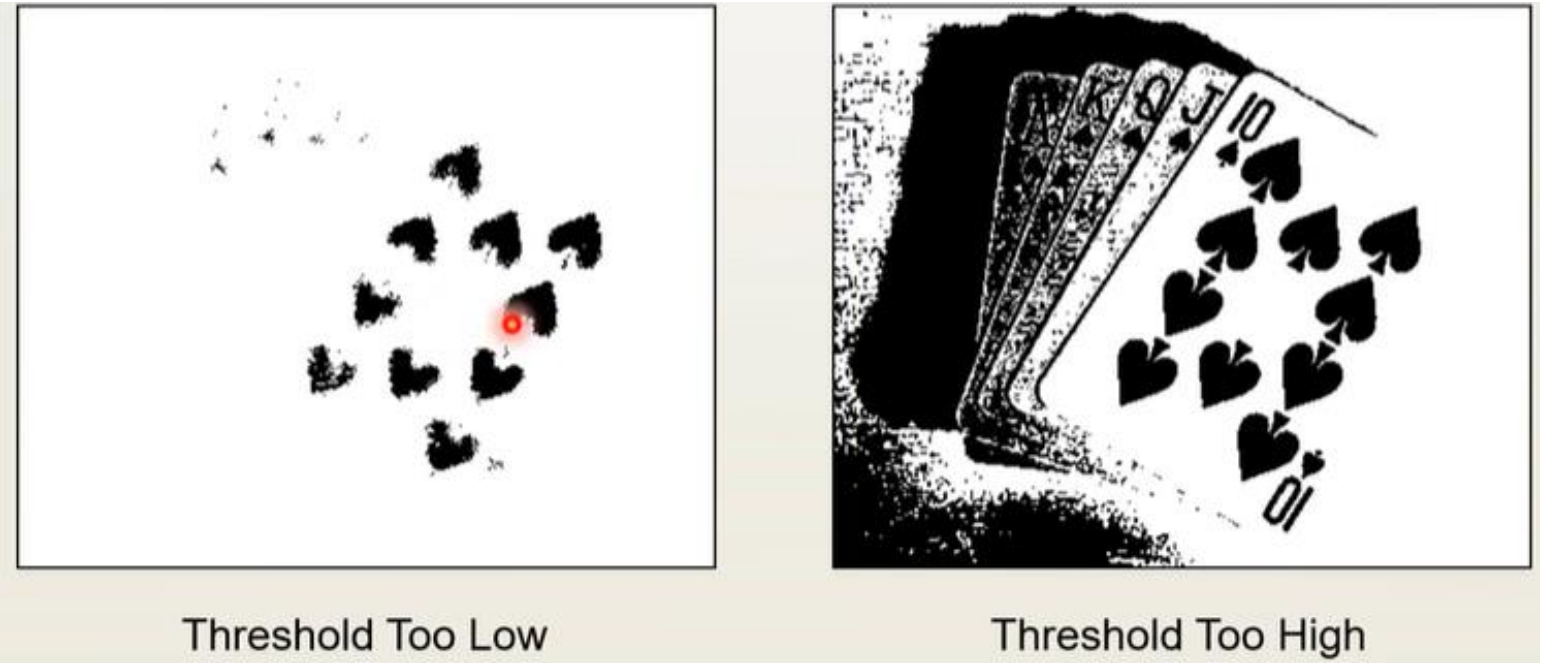
Thresholding Example

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



But be Careful

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



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 $\mu_1$and G_2 to give μ_2

Basic Global Thresholding Algorithm

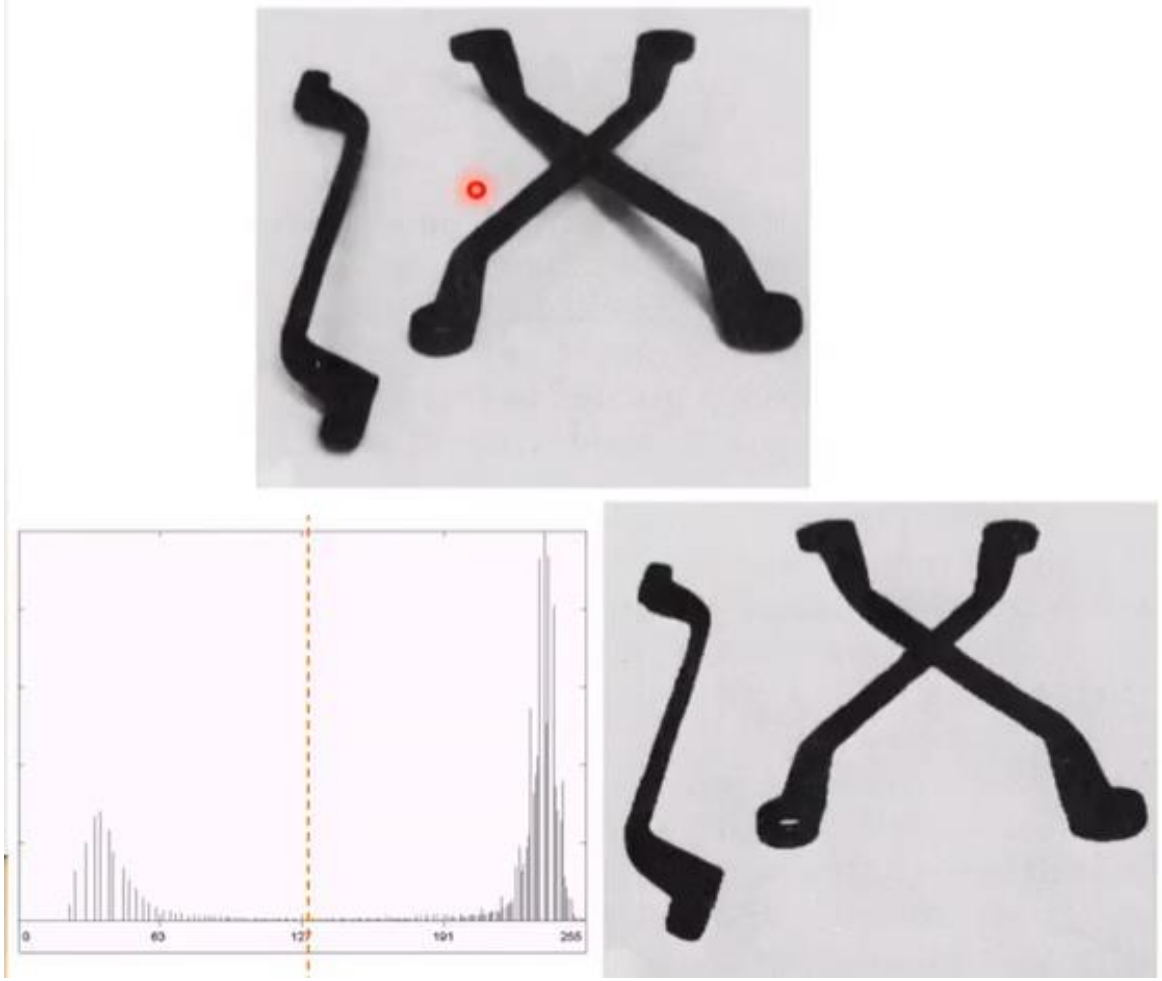
4. Compute a new threshold value:

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

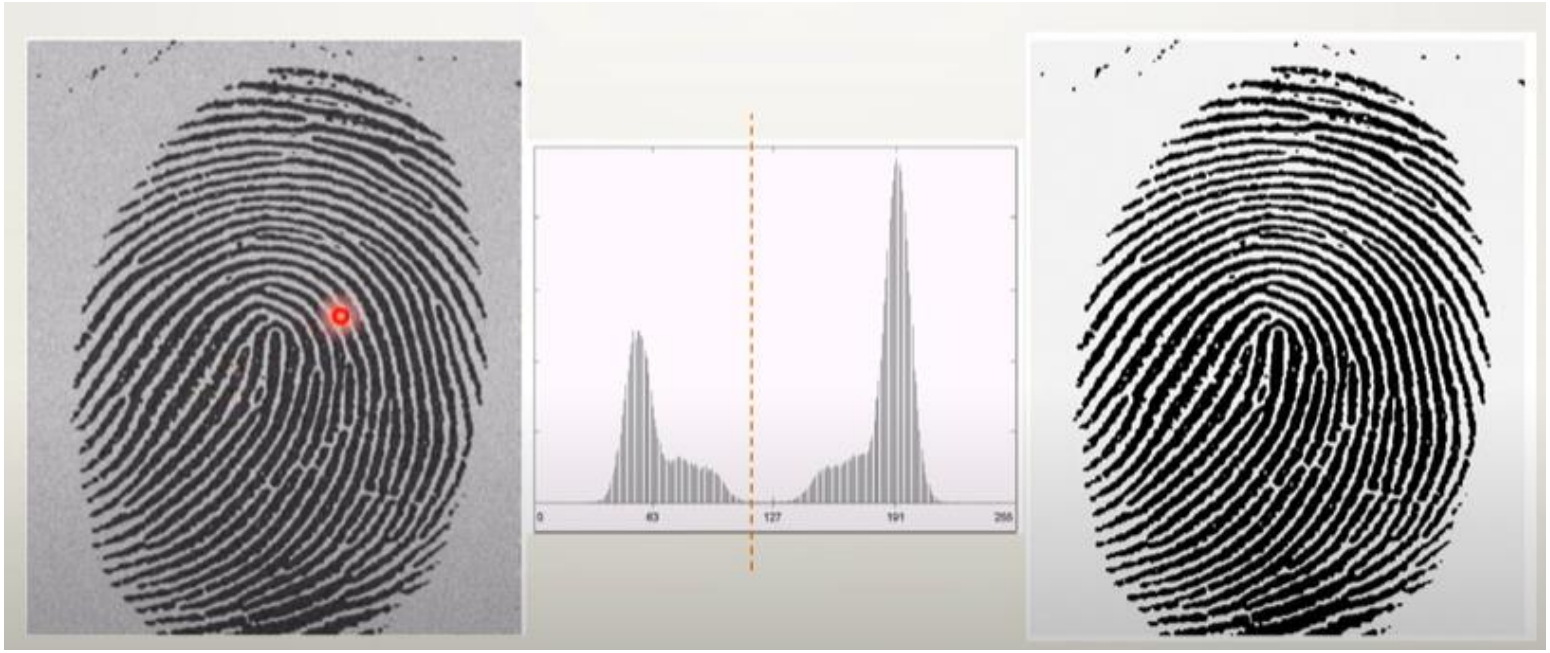
5. Repeat steps 2-4 until the difference in T in successive iterations is less than a predefined limit T_

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

Thresholding Example 1



Thresholding Example 2



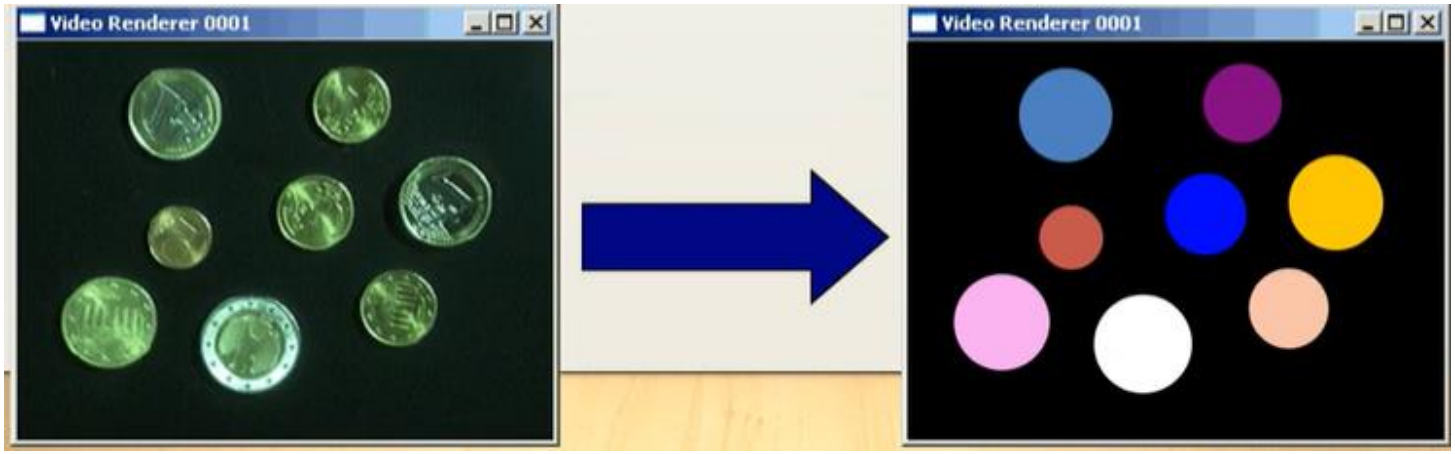
Problems with single value Thresholding(cont…)

- 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



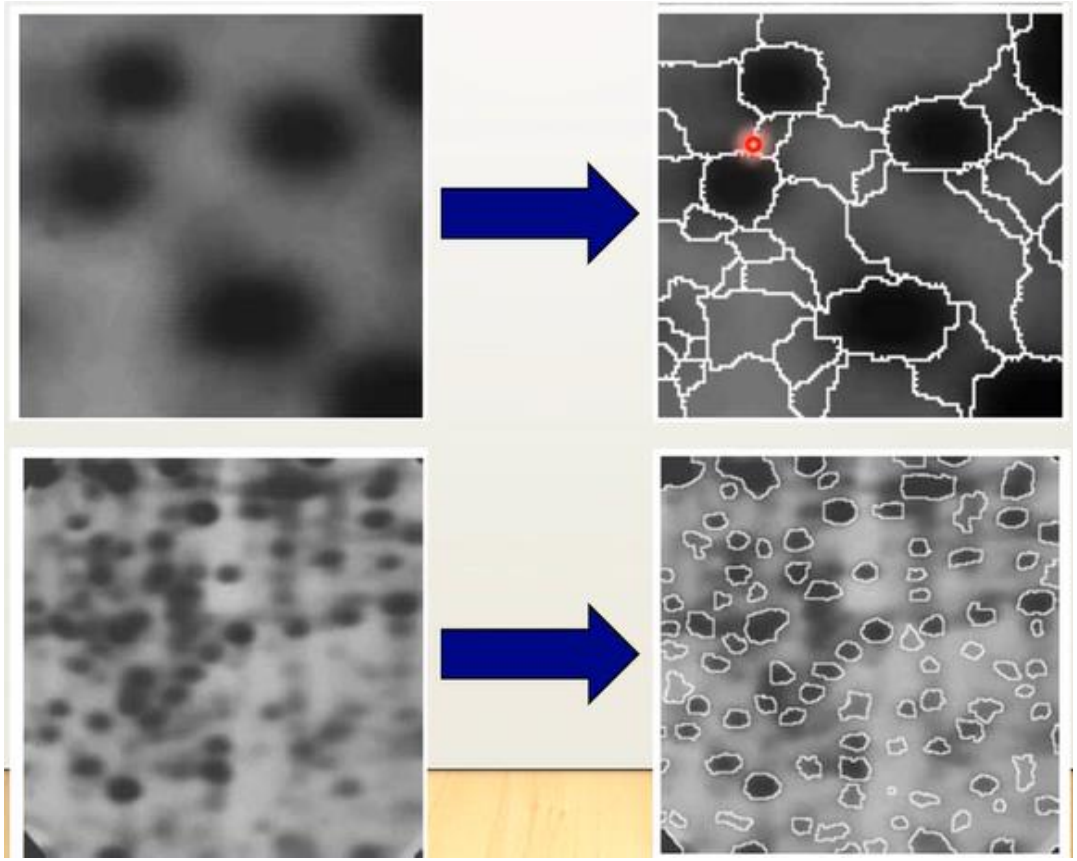
The segmentation problem

- 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

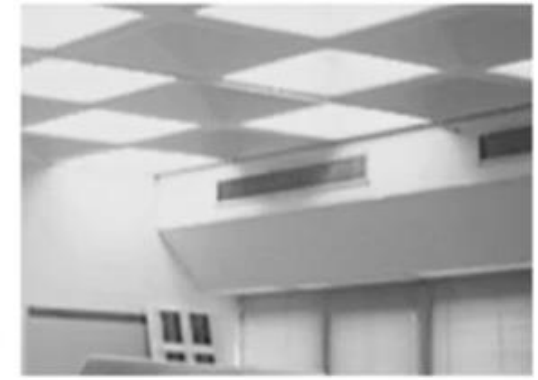
Point Detection

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

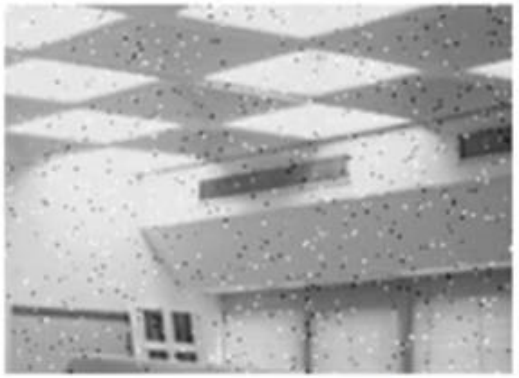
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

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

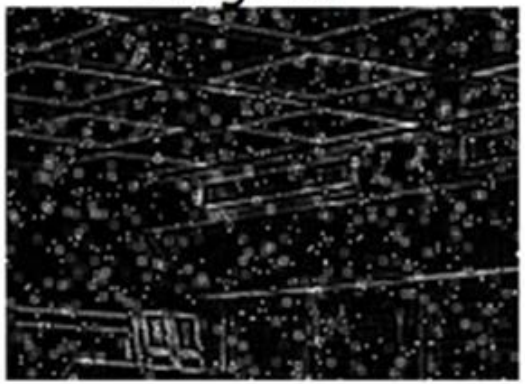
Point Detection



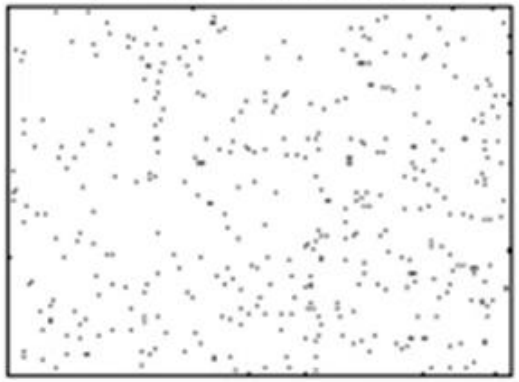
Original



Noise-added

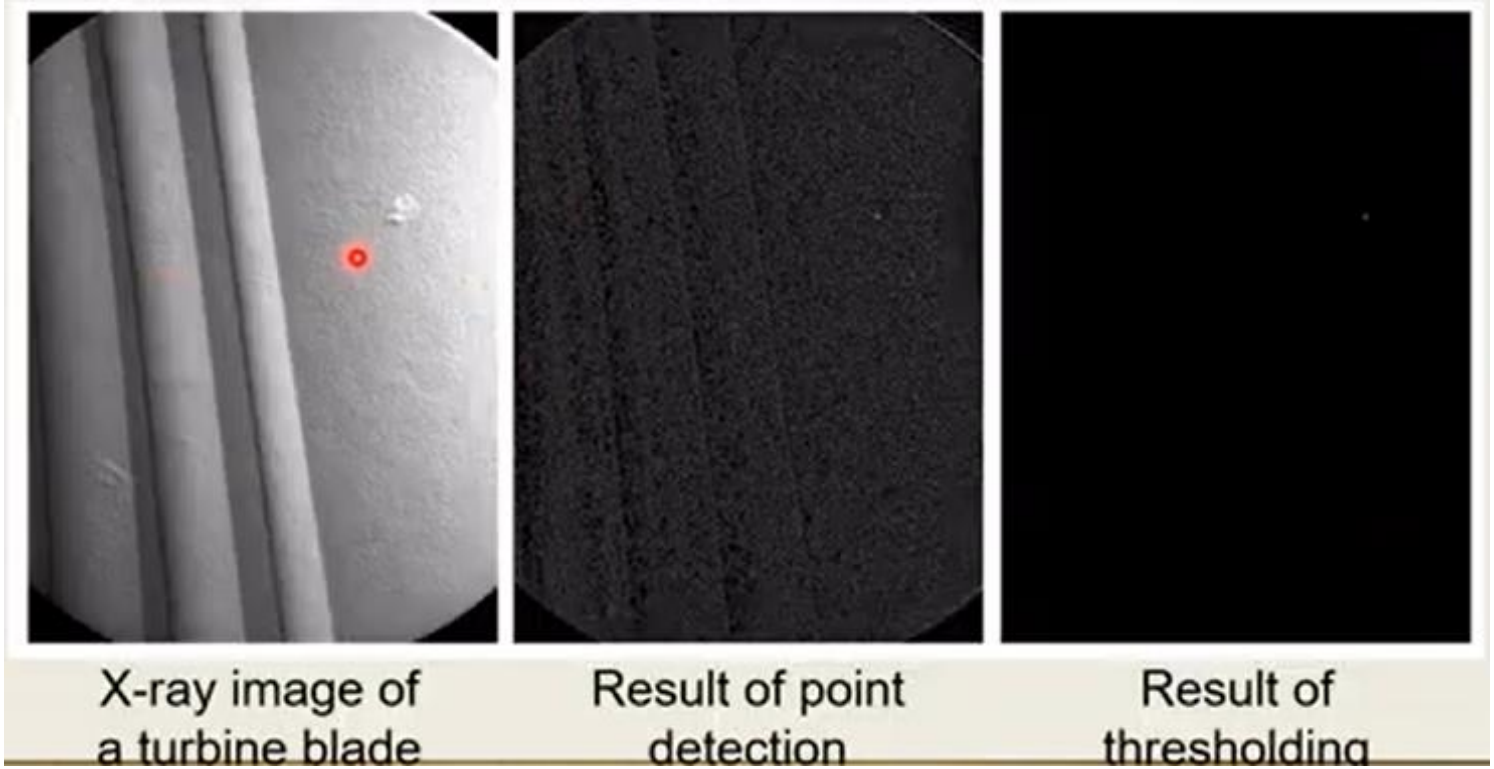


Filtered o/p

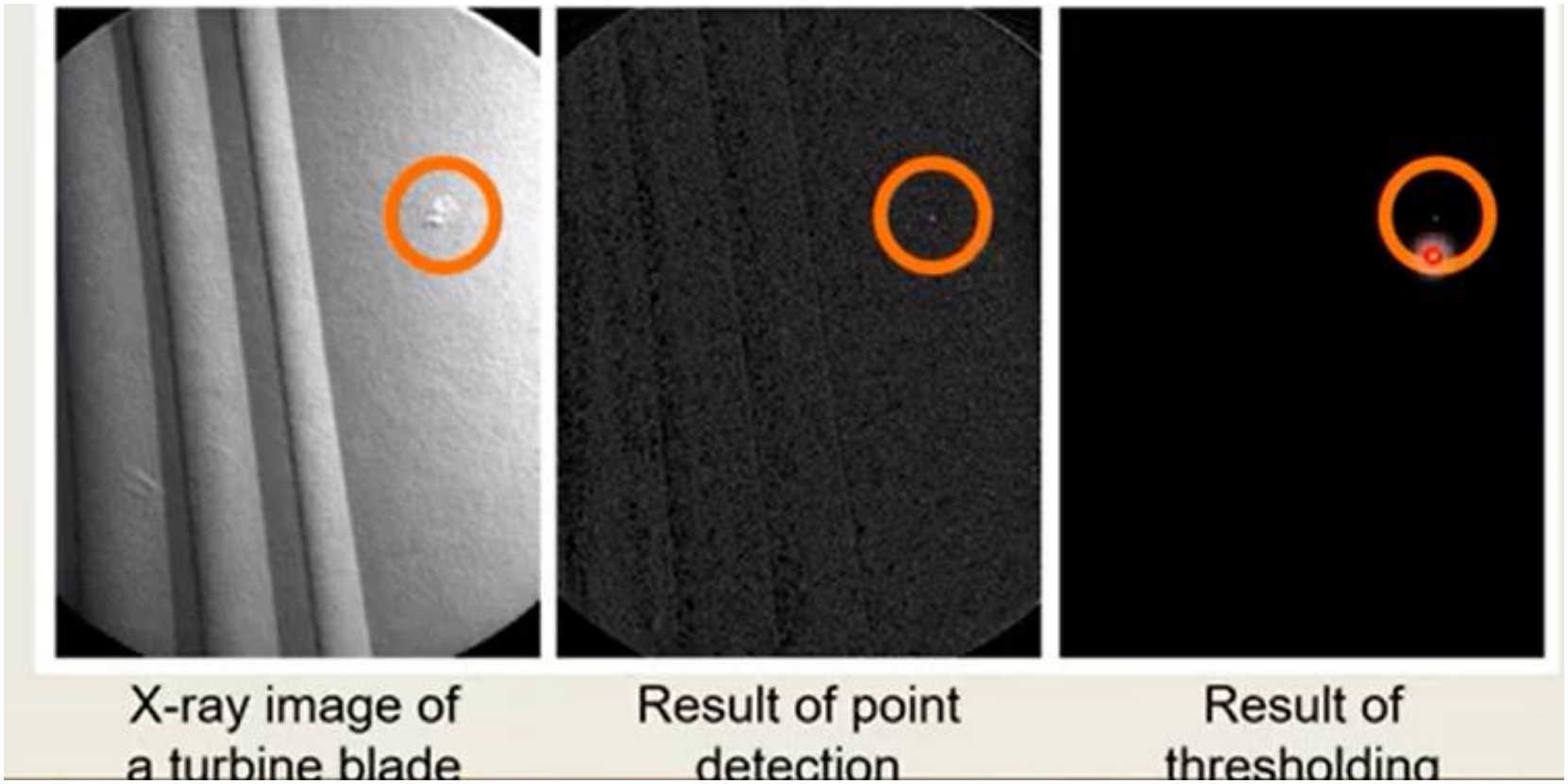


Thresholded o/p

Point Detection



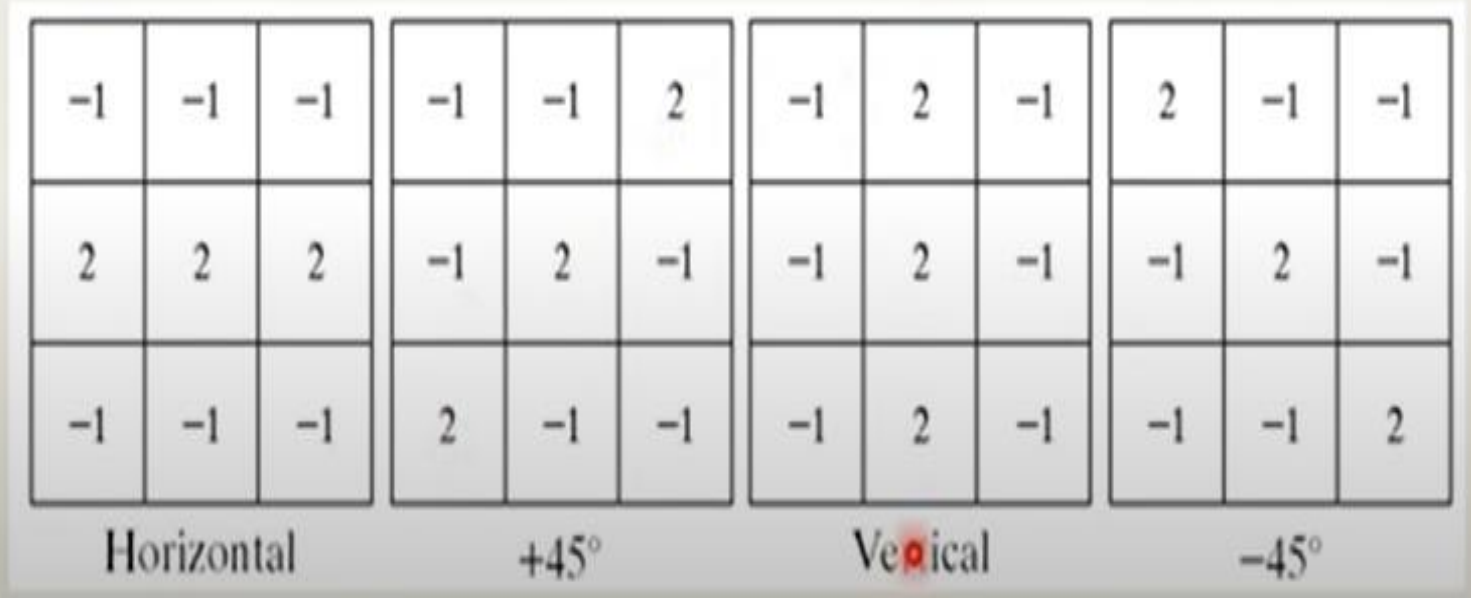
Point Detection



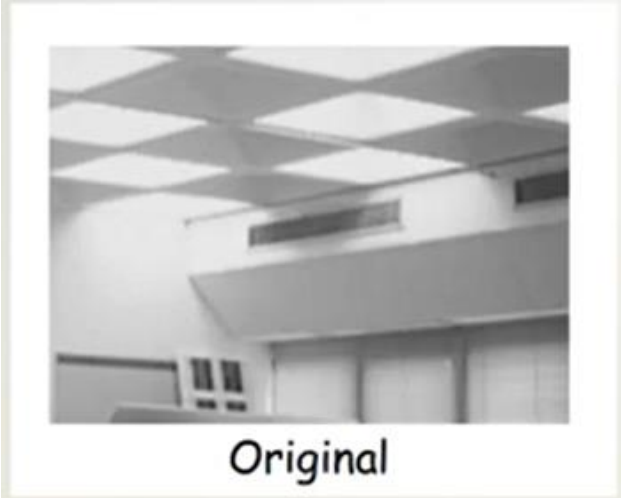
Line Detection

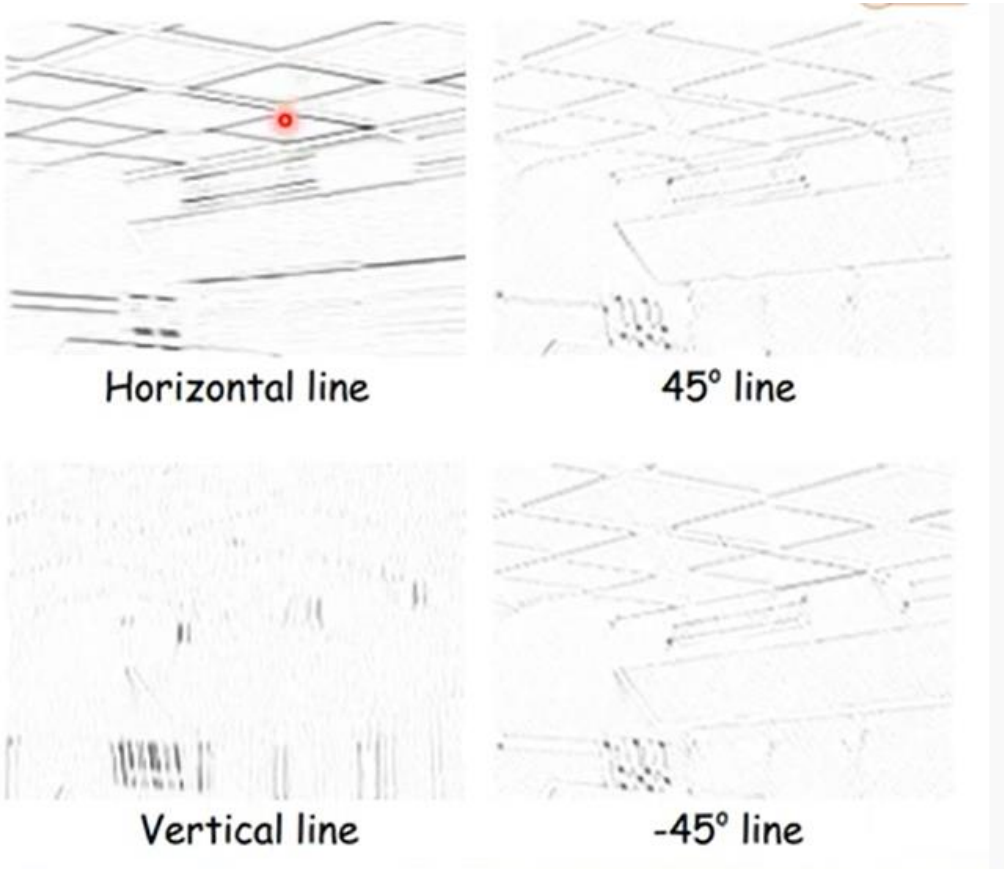
Line Detection

- 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

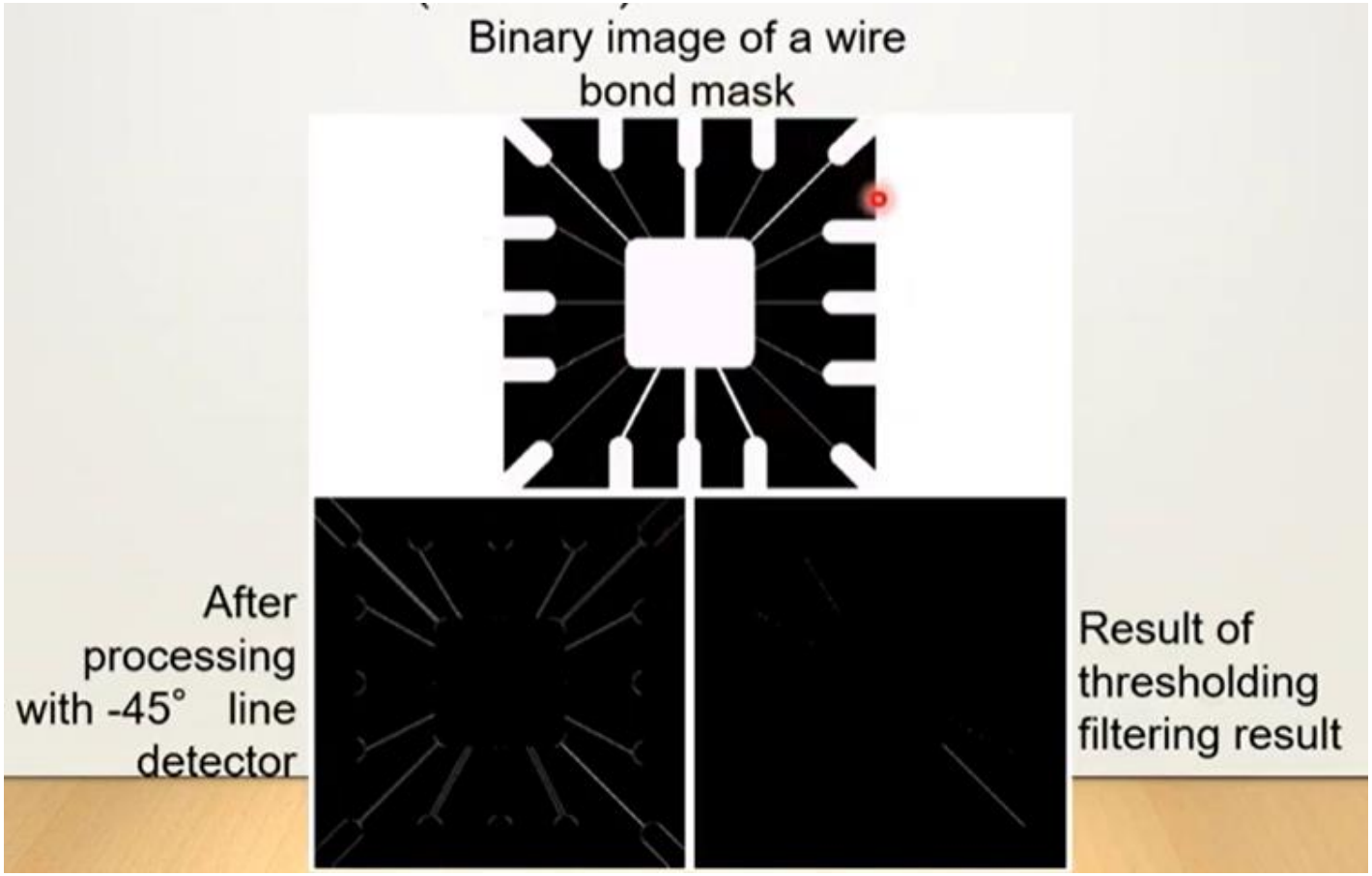


Line Detection





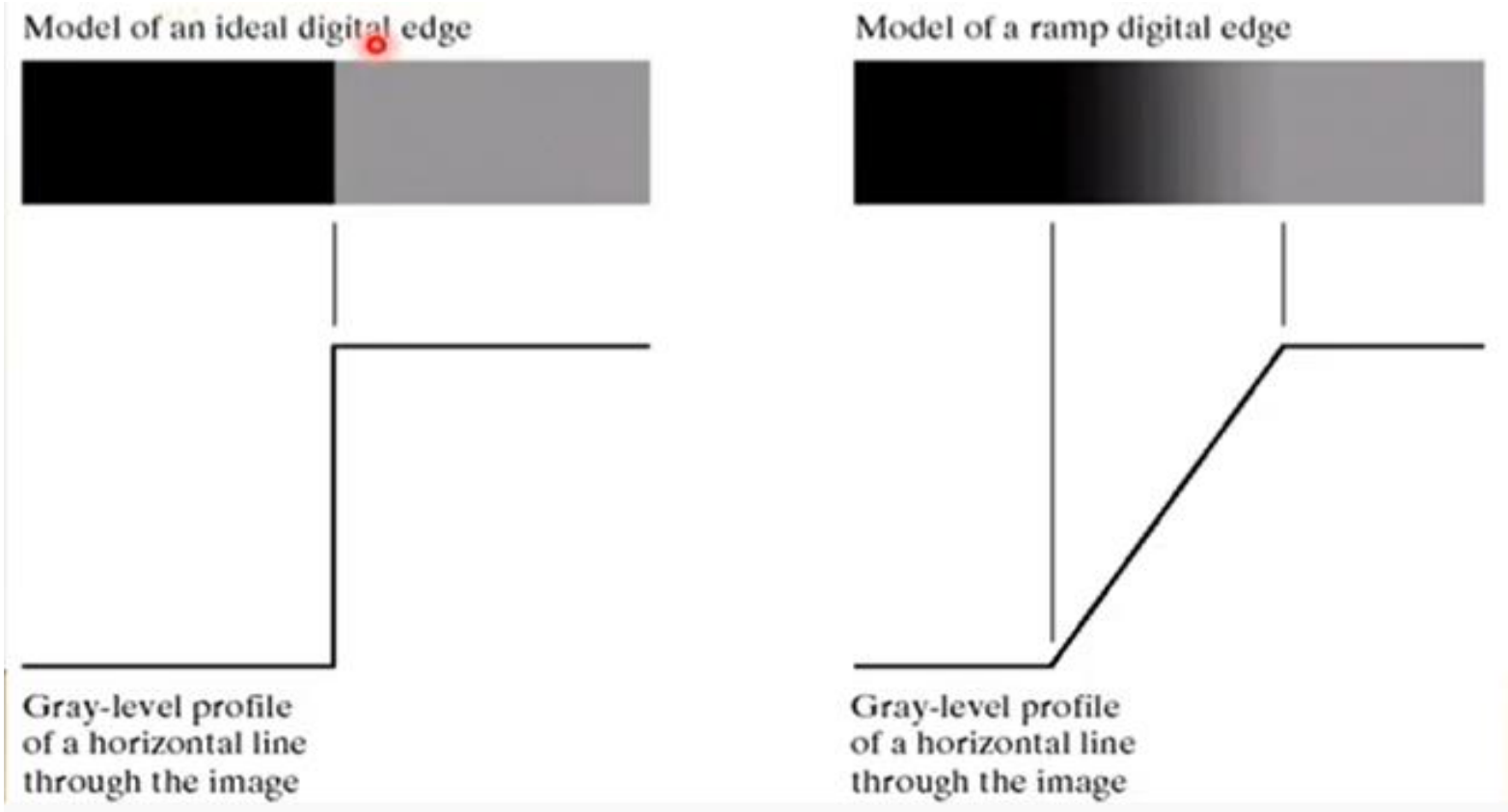
Line Detection



Edge Detection

Edge Detection

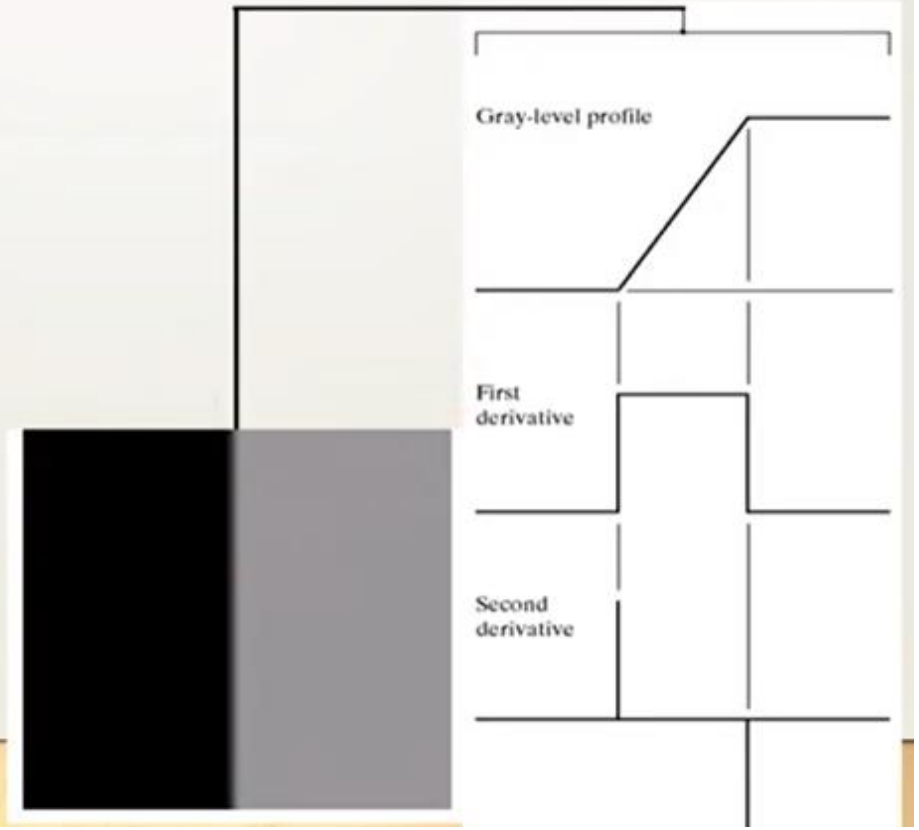
- An edge is a set of connected pixels that lie on the boundary between two regions



Edges & Derivatives

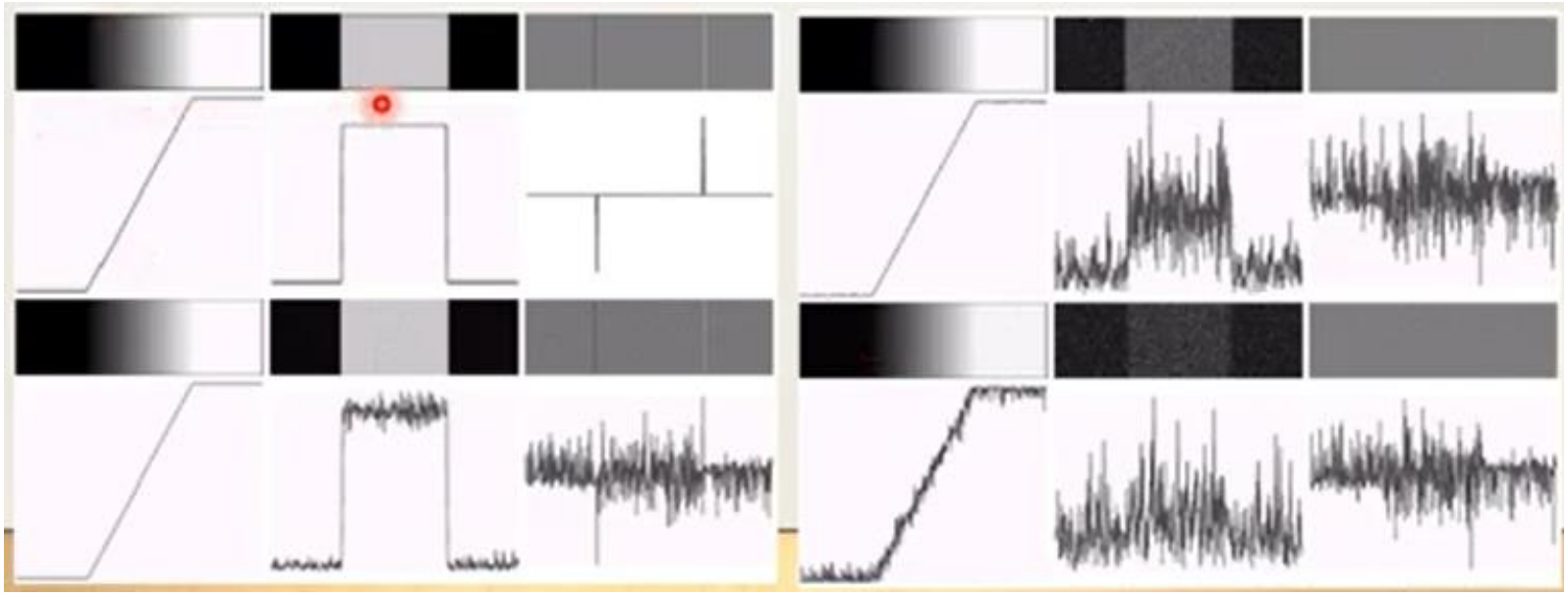
Edges & Derivatives

- 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



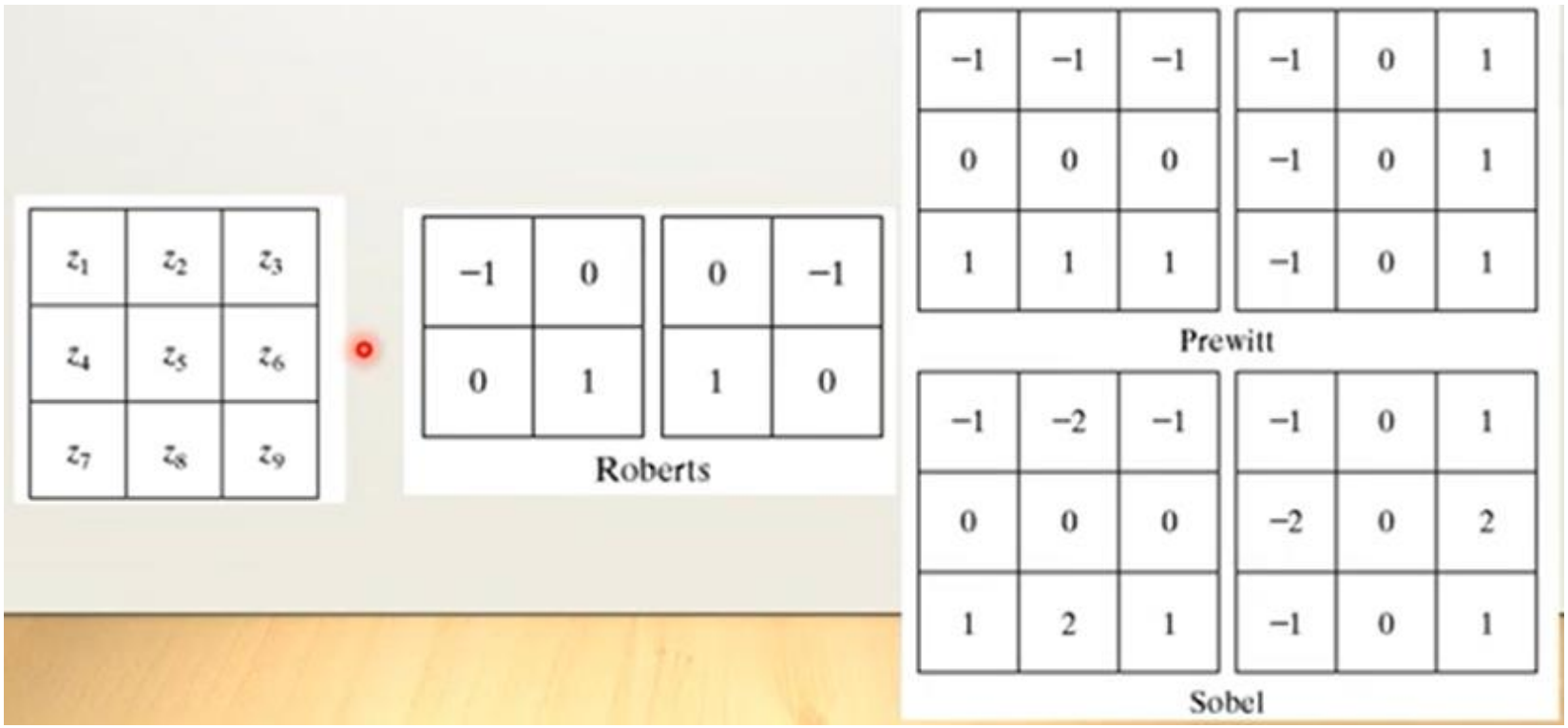
Derivatives &Noise

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



Common Edge Detectors

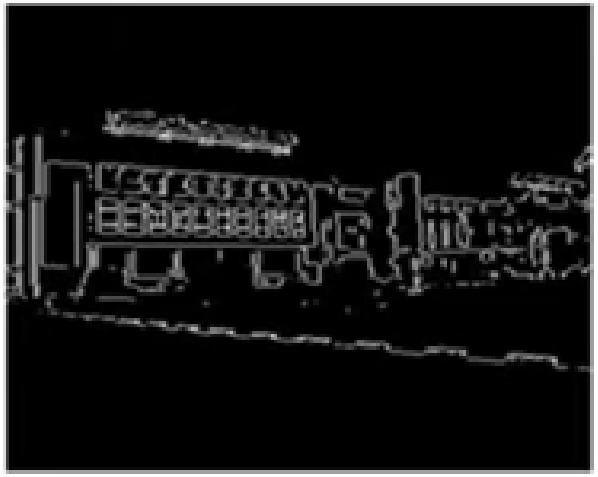
- Given a 3*3 region an image the following edge detection filters can be used



Original

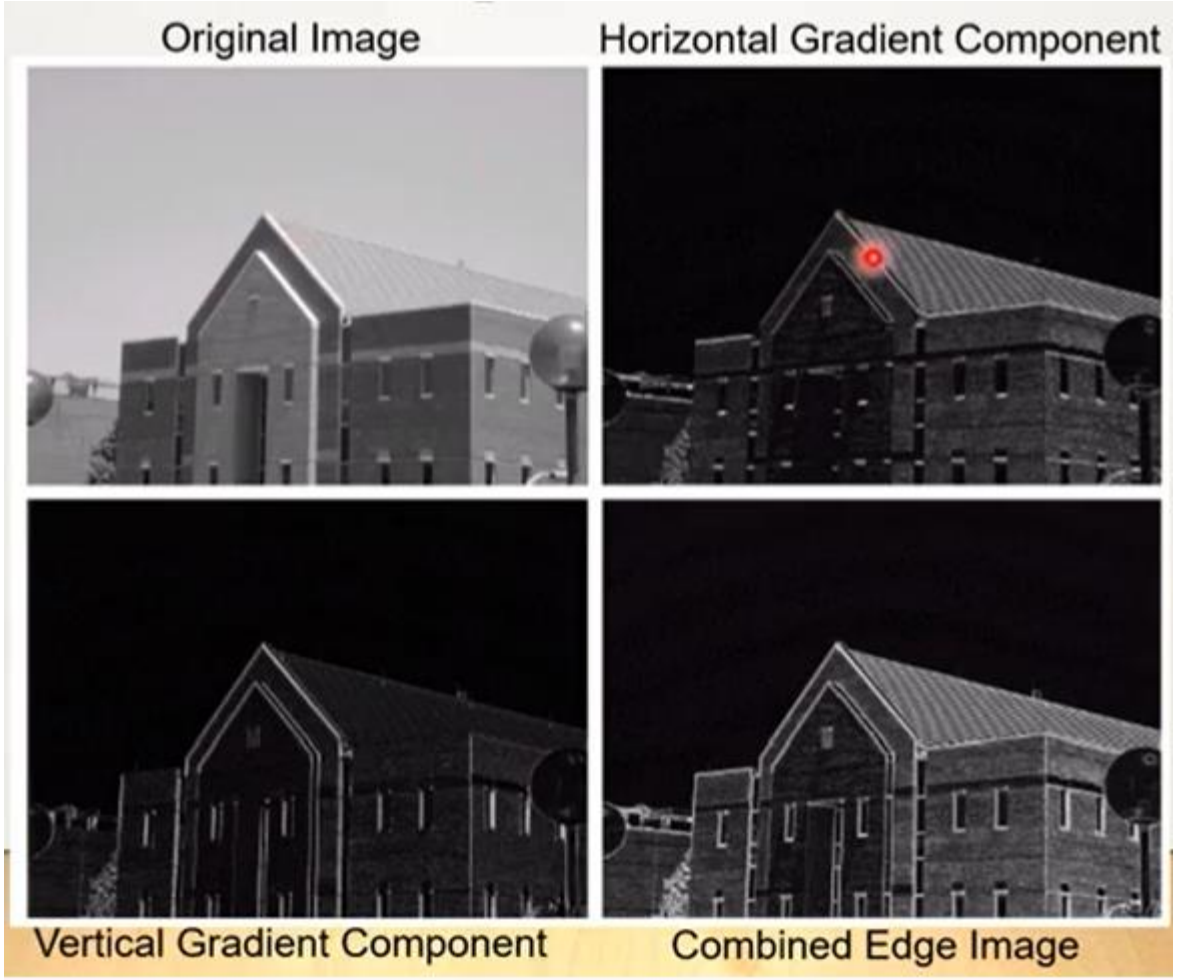
Roberts

Sobel

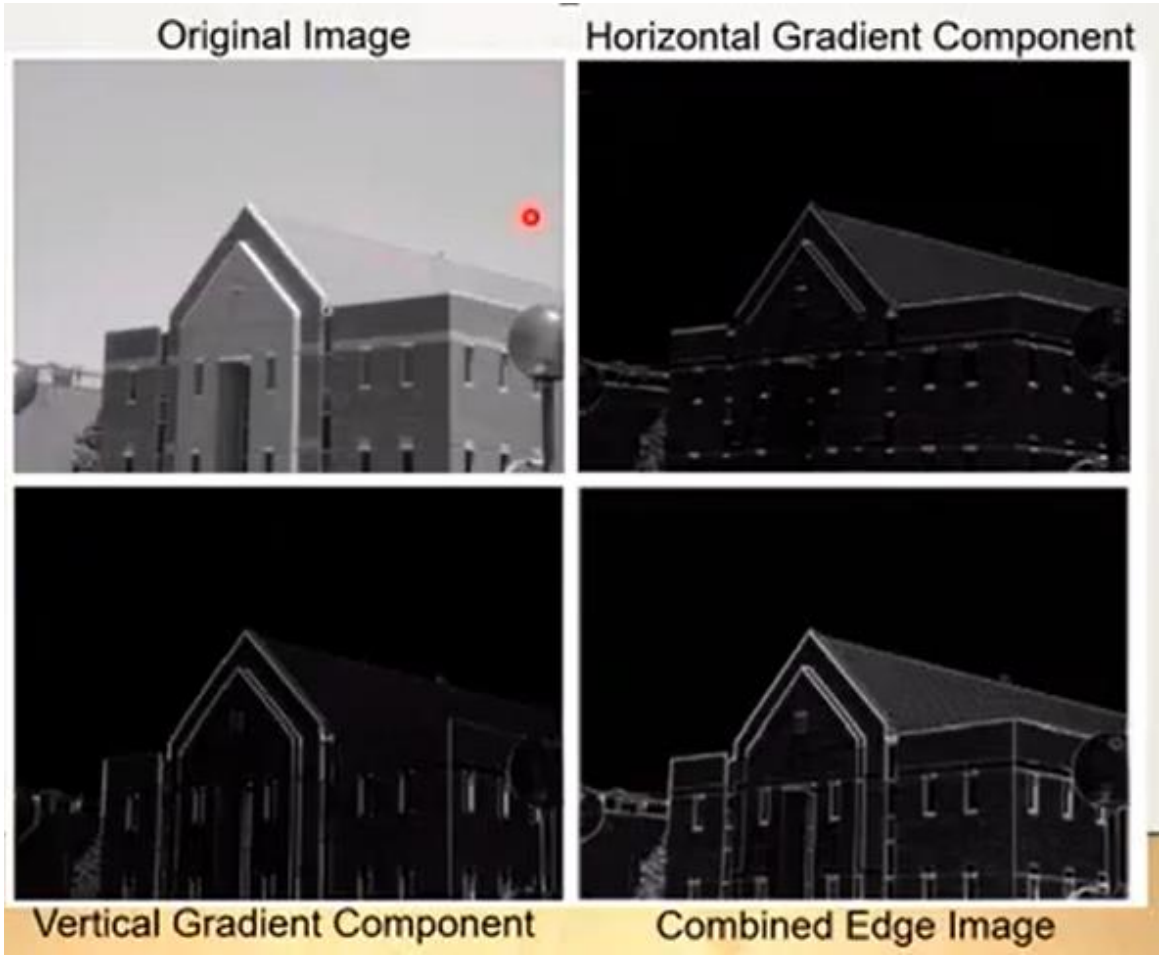


Prewitt

Edge Detection Example



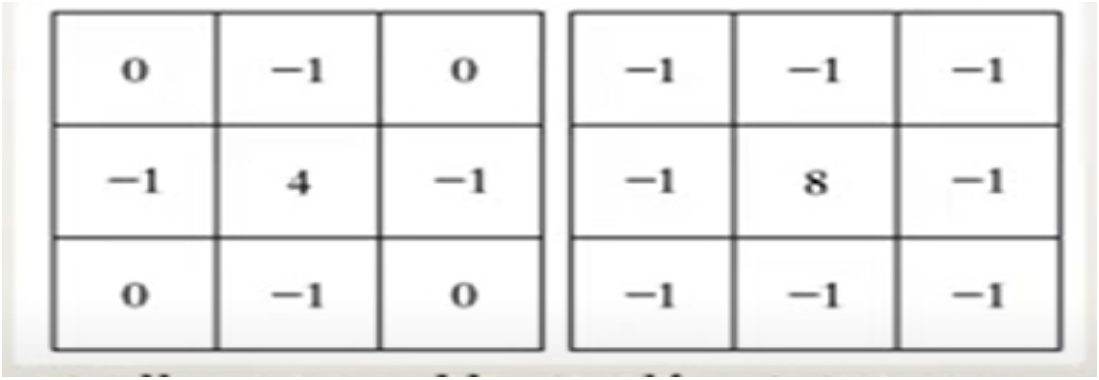
Edge Detection Example with Smoothing



Laplacian Edge Detection

Laplacian Edge Detection

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



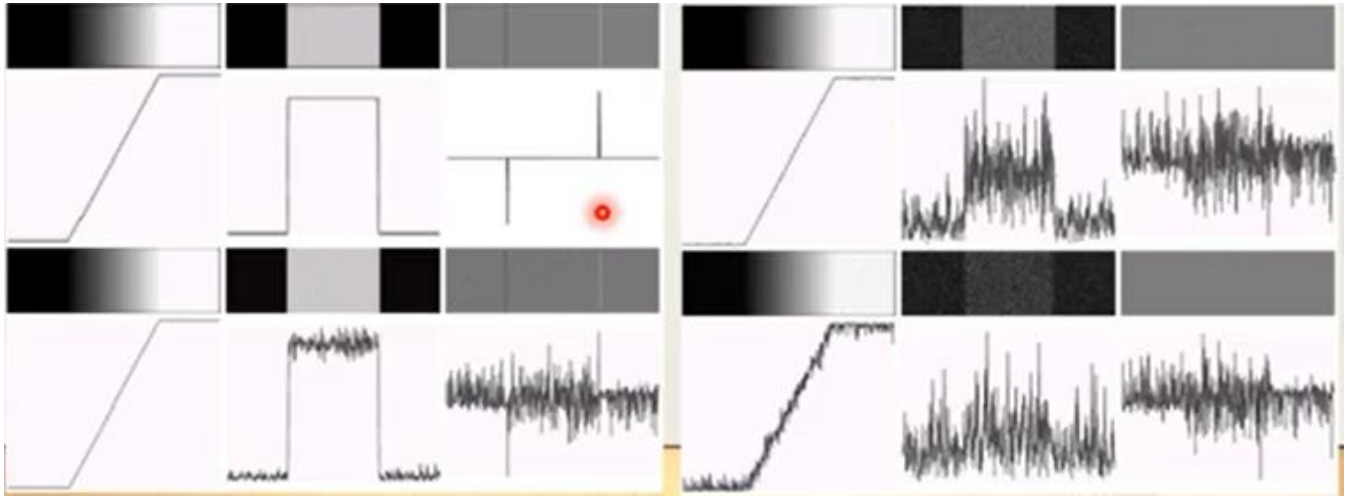
0	−1	0
−1	4	−1
0	−1	0

−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

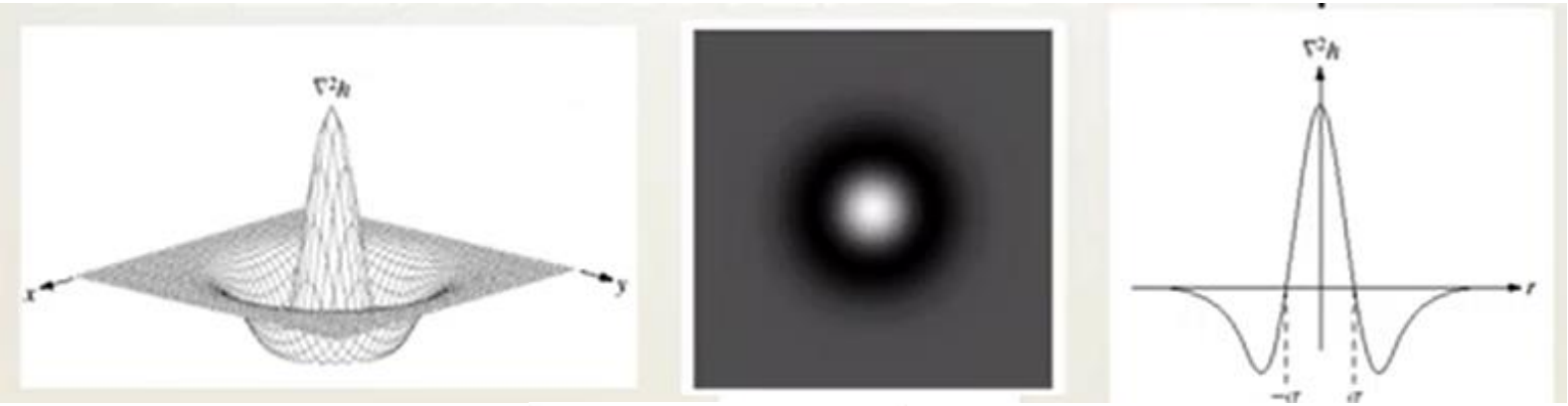
- Derivative based edge detectors are extremely sensitive to noise
- We need to keep this in mind



Laplacian Edge Detection

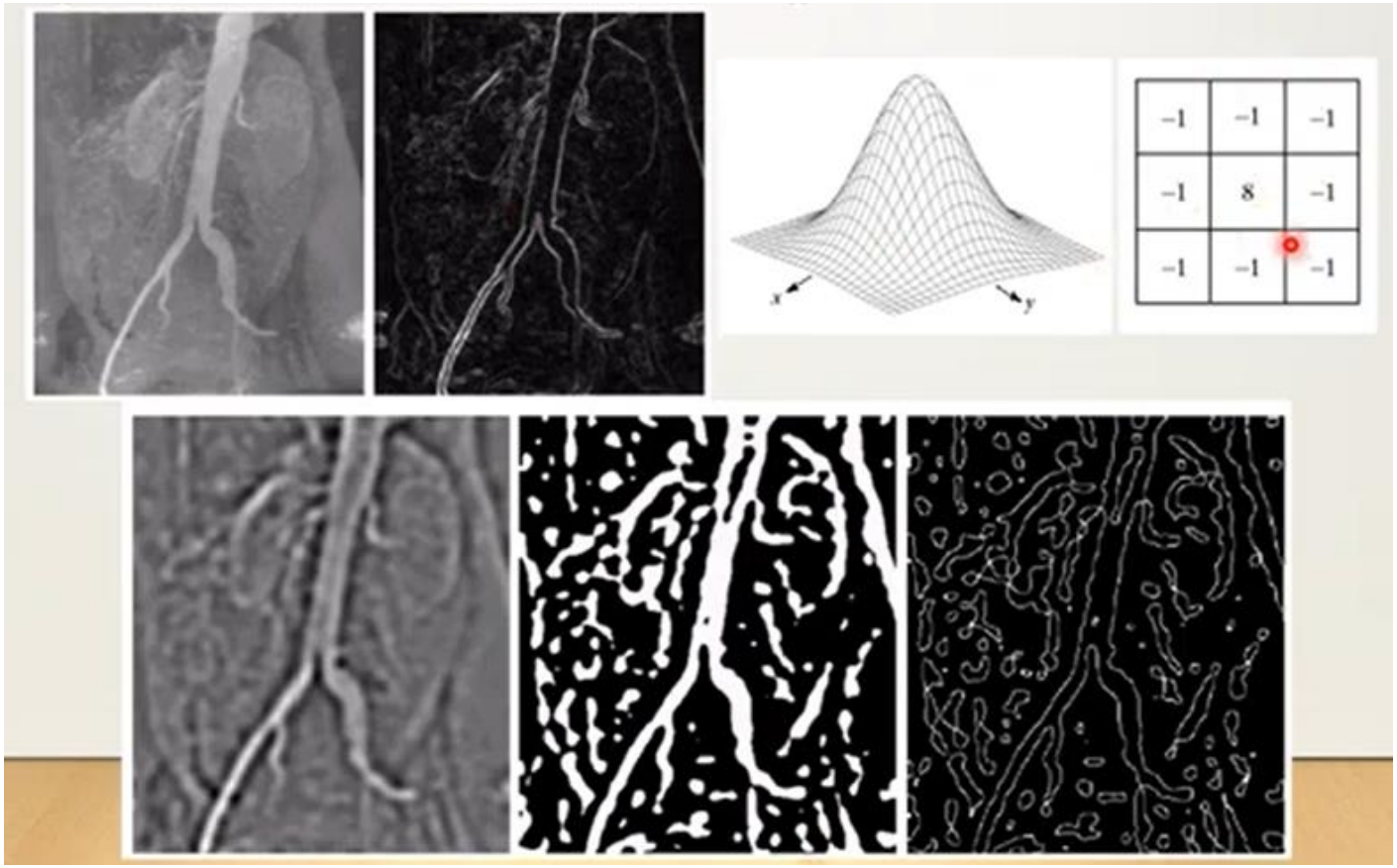
Laplacian of Gaussian

- 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

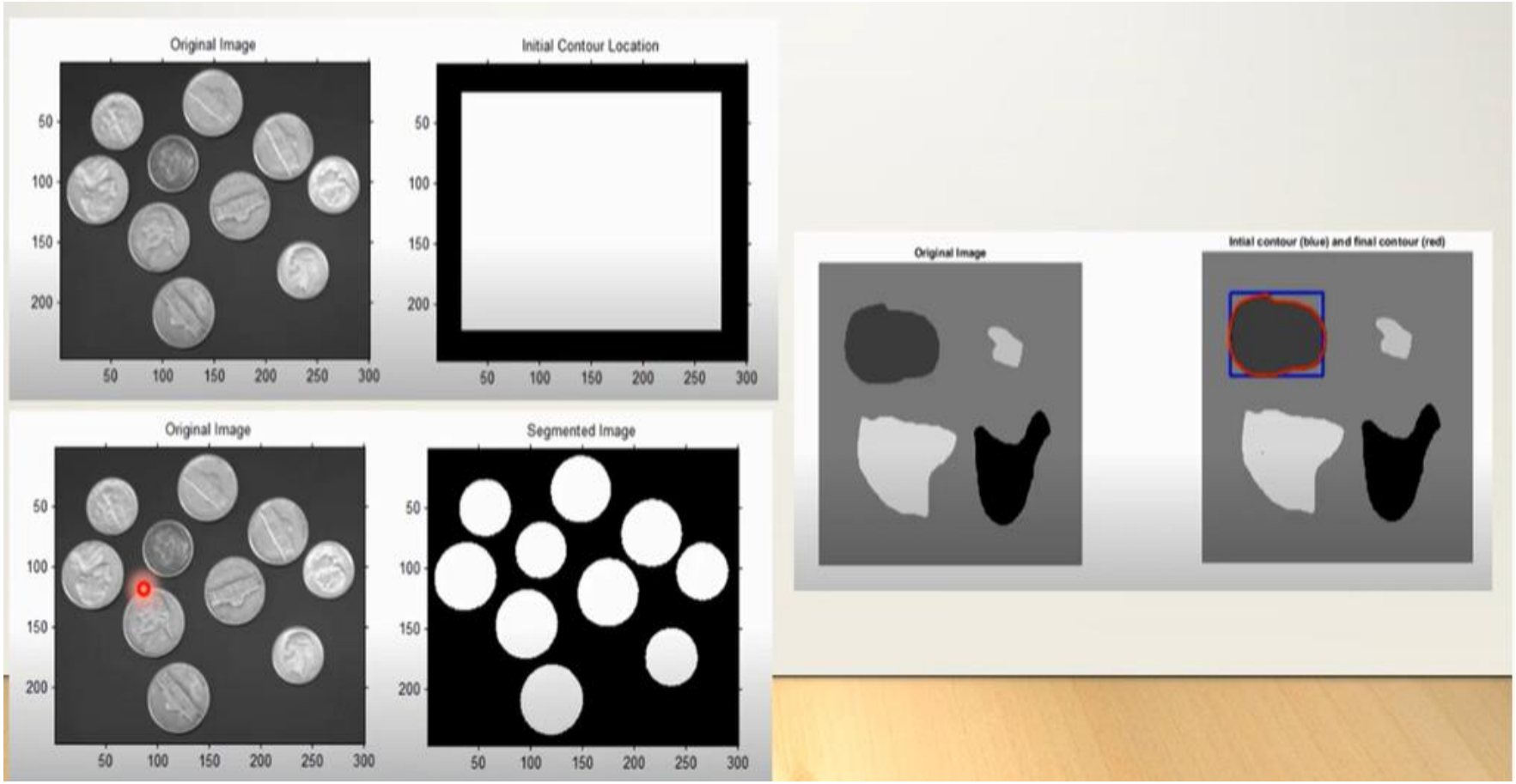
Image

Active Contour:

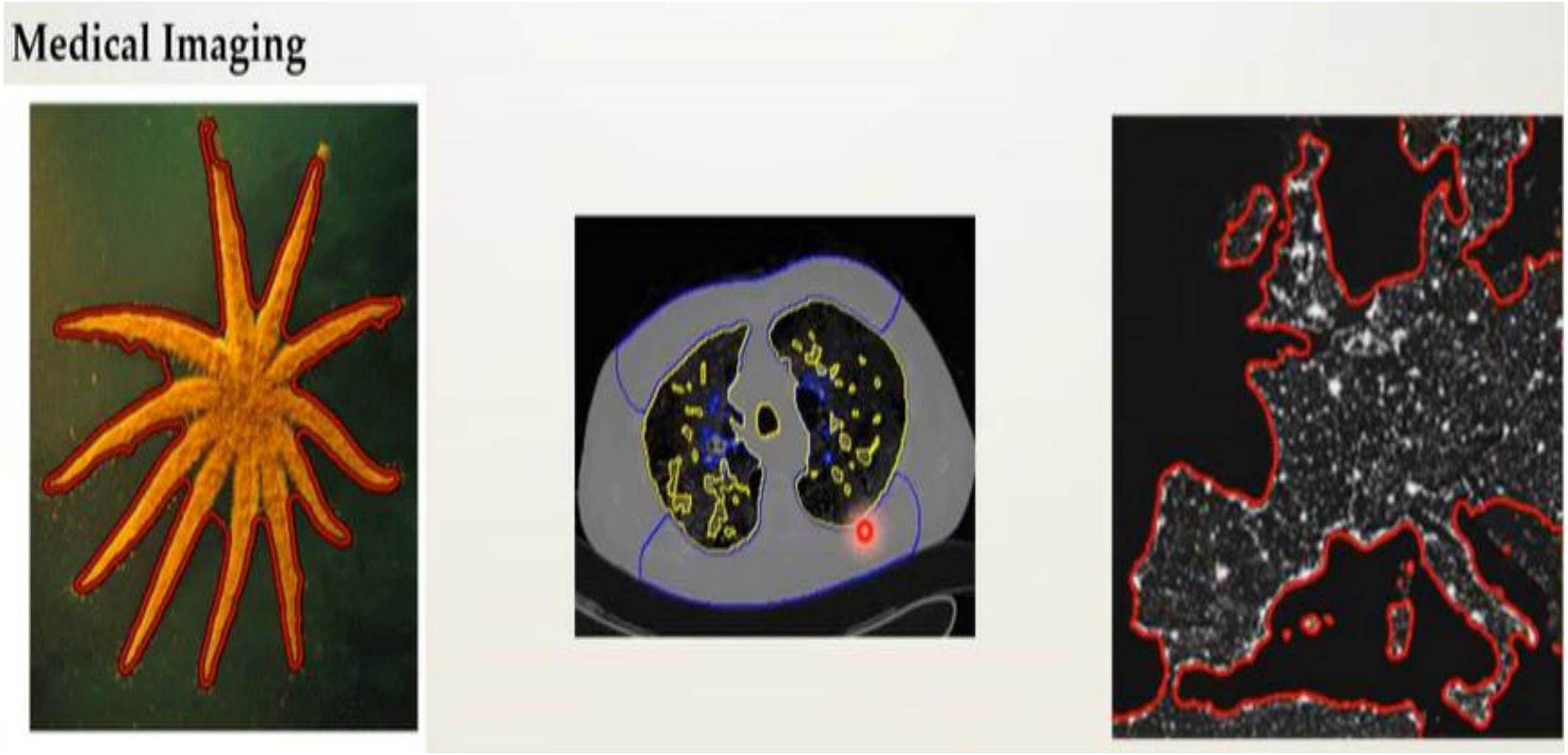
Iteratively "deform" the initial contour so that:

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

Example of Active Contour



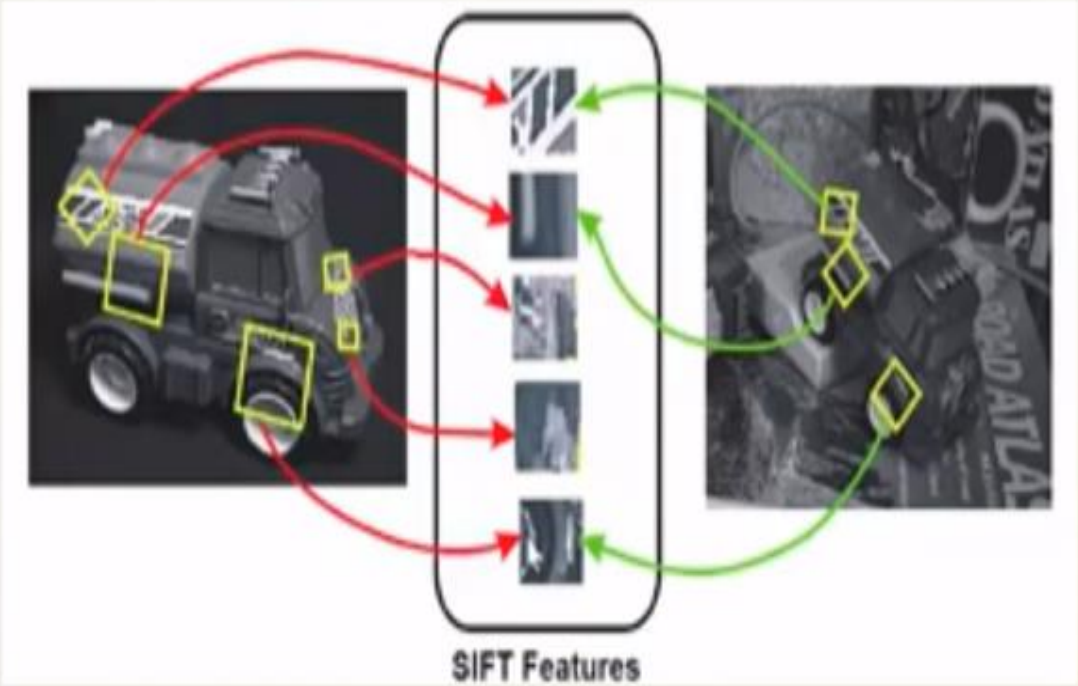
Example of Active Contour



SIFT Features

SIFT Features

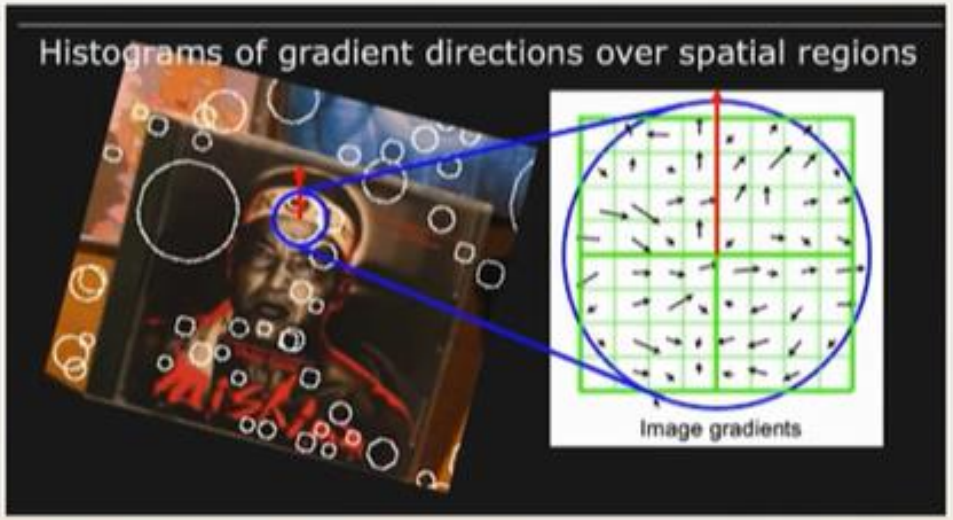
- SIFT stands for Scale-Invariant Feature Transform and was first presented in 2004, by **D.Lowe**, University of British Columbia. SIFT is invariance to image scale and rotation.

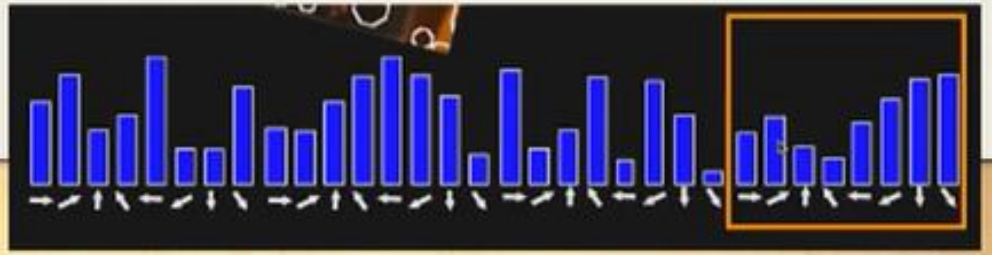


SIFT Features

- 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:

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

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: $\bar{H}_i = \frac{1}{N} \sum_{k=1}^{N} H_i(k)$

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.

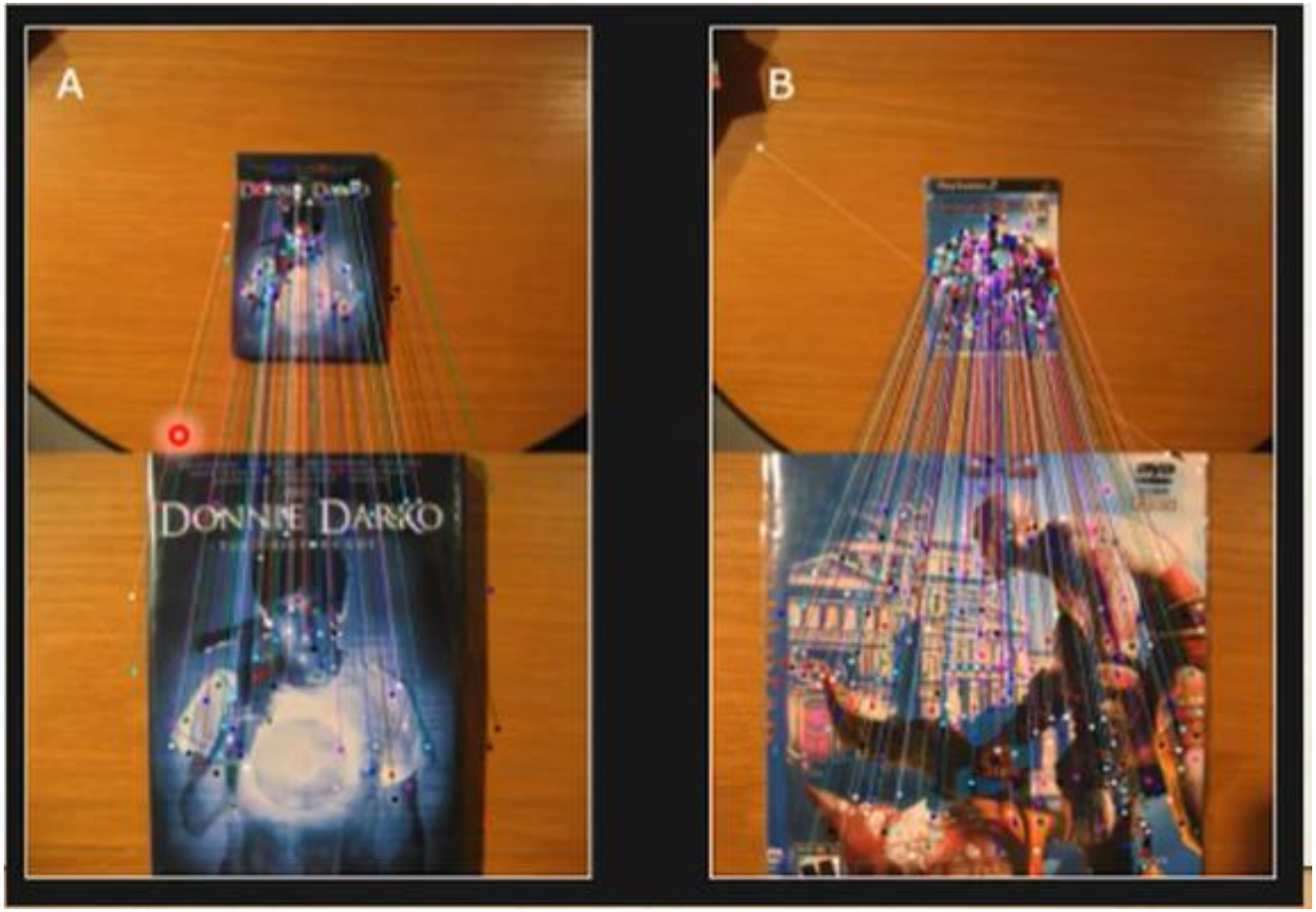
Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N.

Intersection:

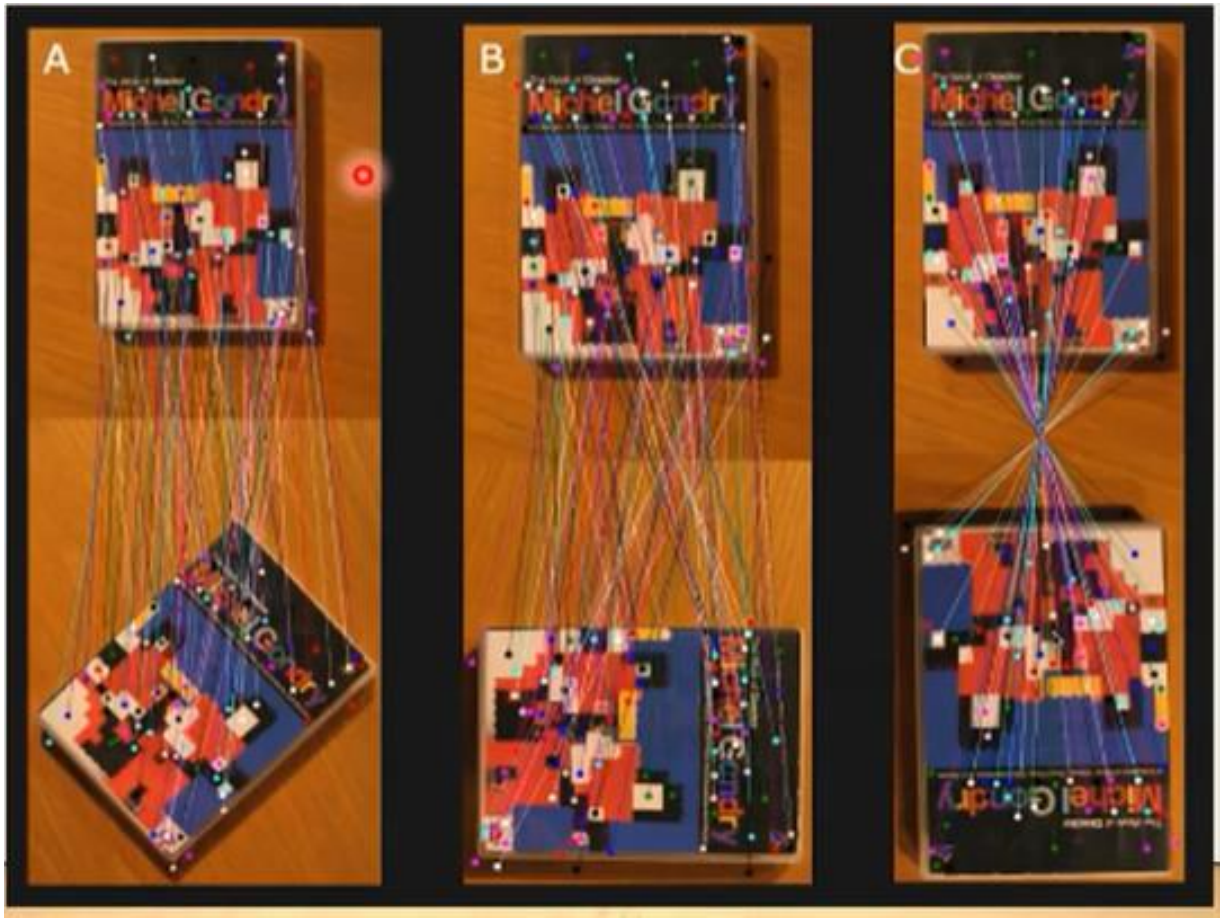
$$d(H_1, H_2) = \sum_k \min\left(H_1(k), H_2(k)\right)$$

Larger the distance metric, better the match.

SIFT Results: Scale Invariance



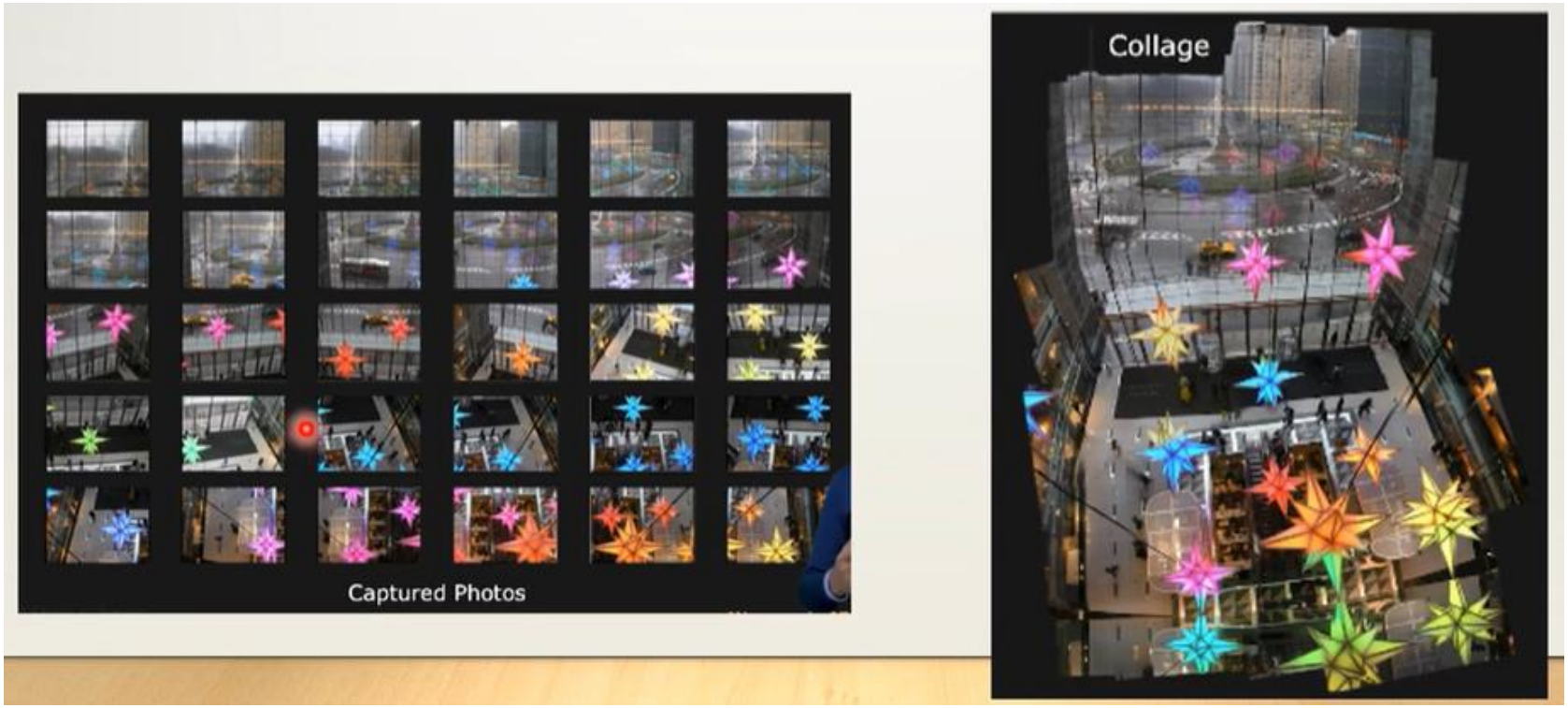
SIFT Results: Rotation Invariance



SIFT Results: Robustness to Clutter

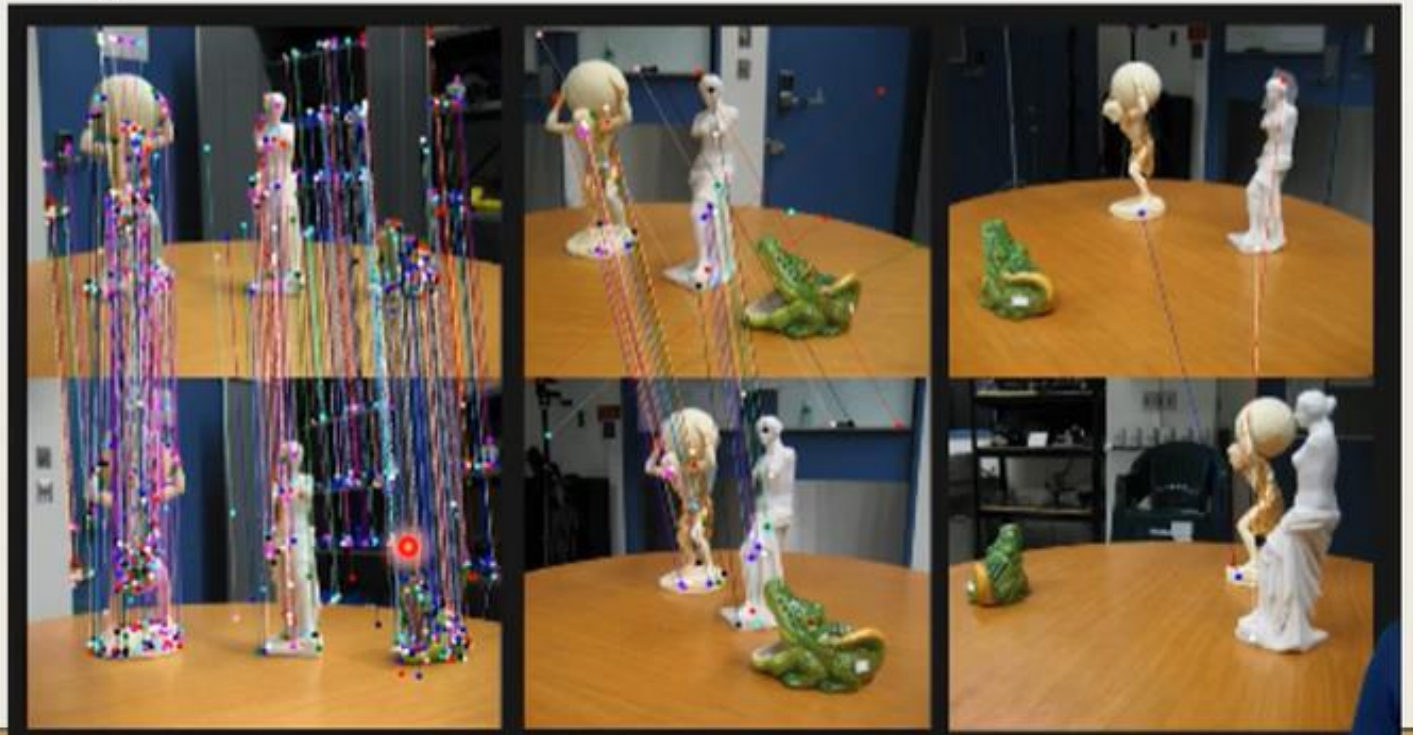
Panorama Stitching using SIFT

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

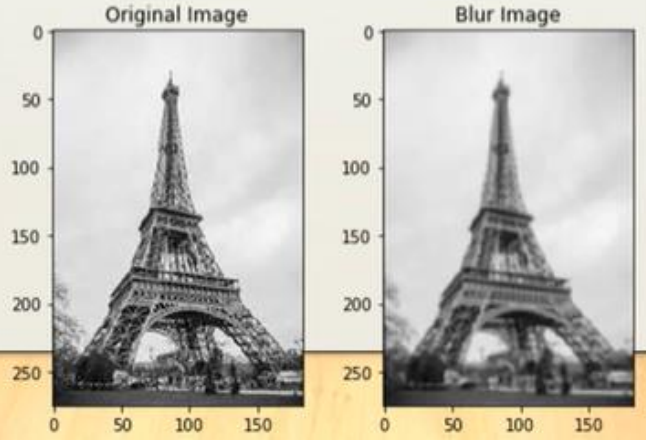
- *SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm in Computer Vision.*
- SIFT helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, 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

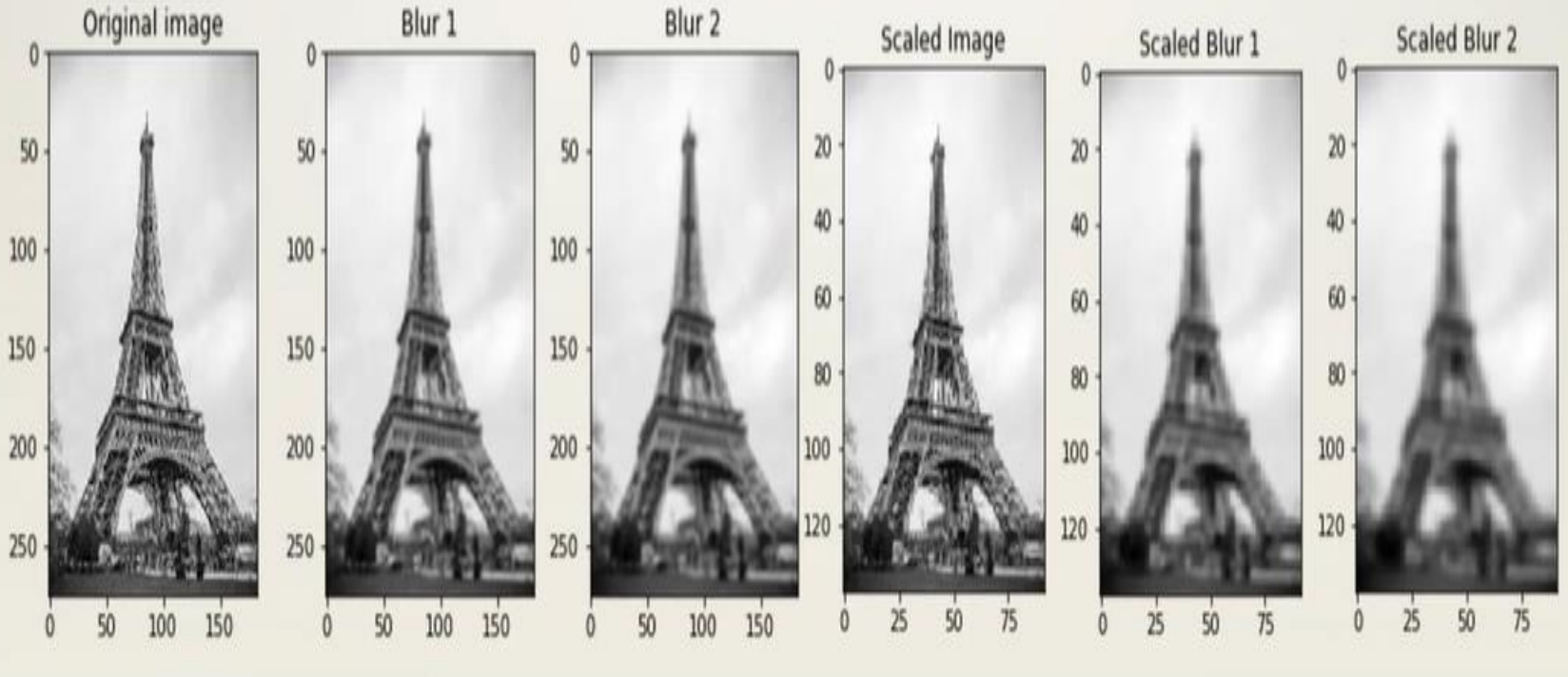
Constructing the scale space

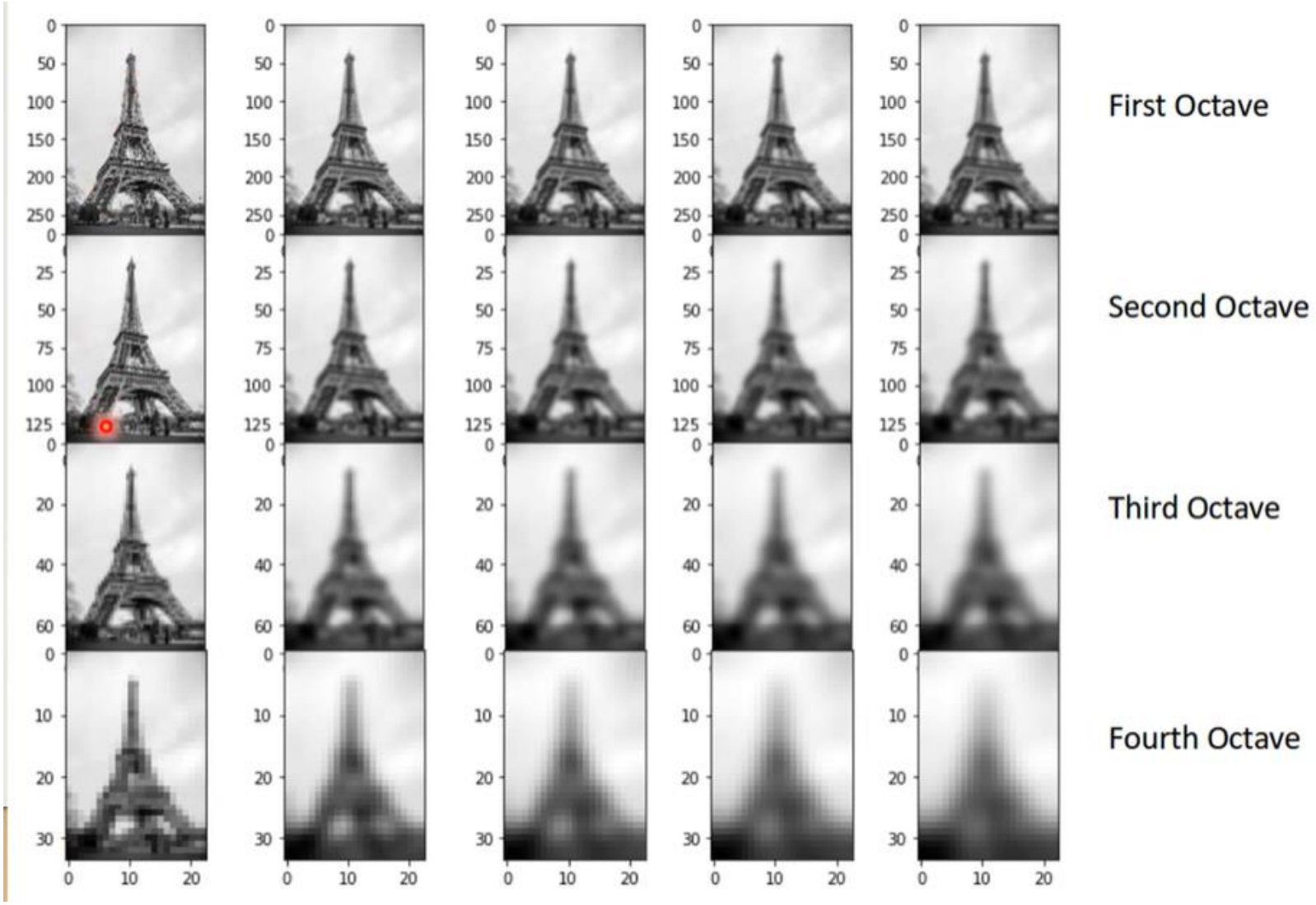
- *We use the **Gaussian Blurring technique** to reduce the noise in an image.*
- So, for every pixel in an image, the Gaussian Blur calculates a value based on its neighboring pixels. Below is an example of image before and after applying the Gaussian Blur. As you can see, the texture and minor details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



Constructing the scale space

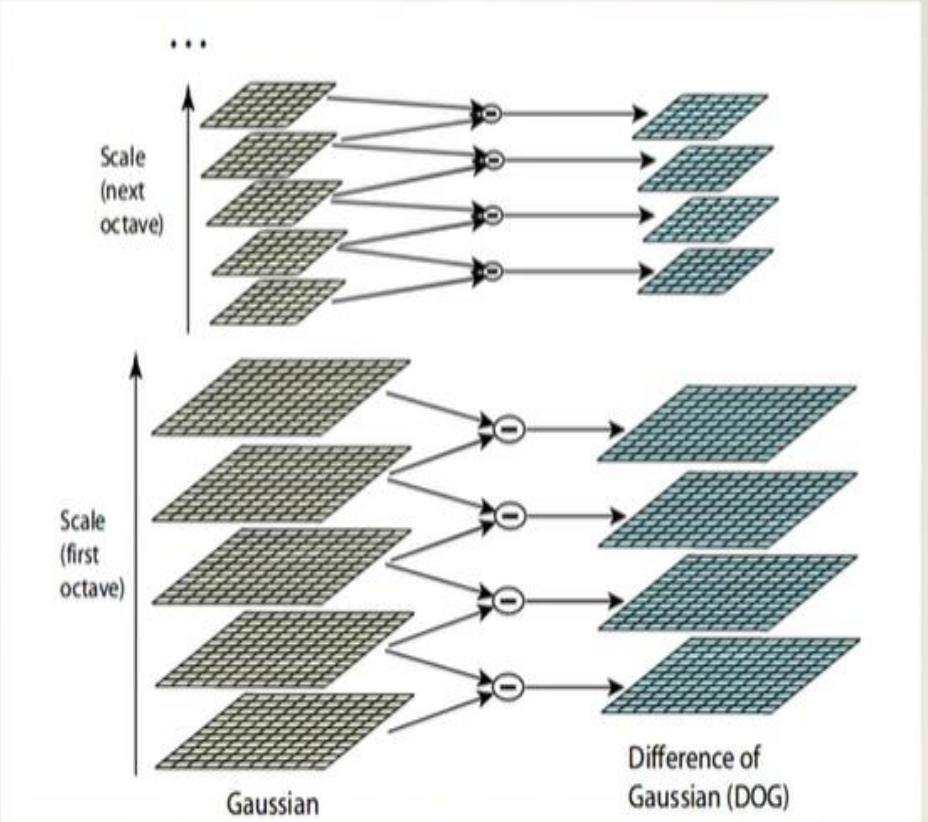
- Hence, these blur images are created for multiple scales. To create a new set of images of different scales, we will take the original image and reduce the scale by half. For each new image, we will create blur versions as we saw above.




First Octave
Second Octave
Third Octave
Fourth Octave

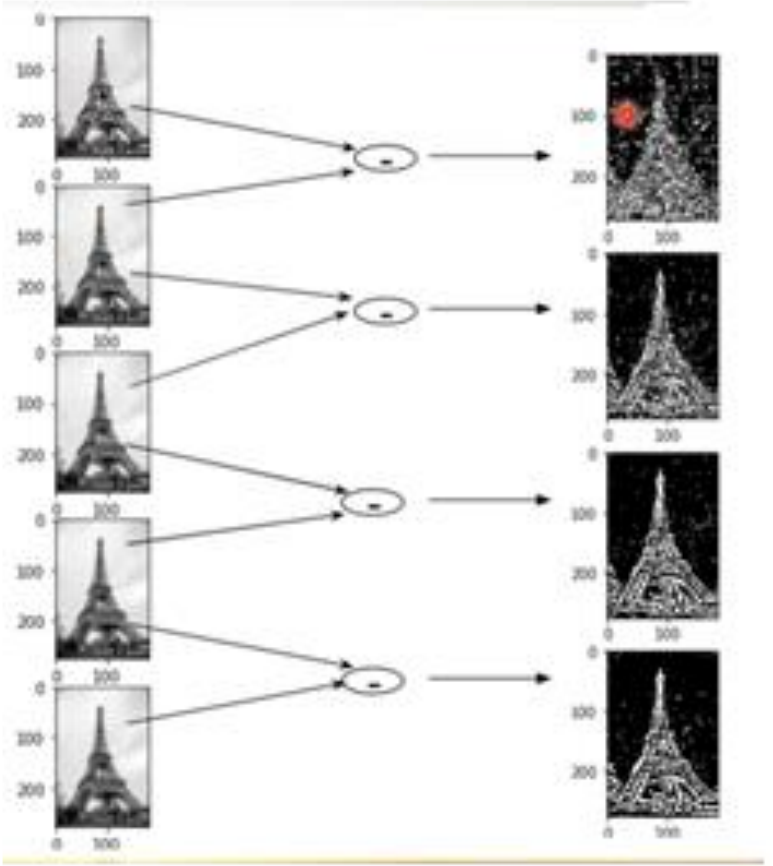
Difference of Gaussian

- *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.



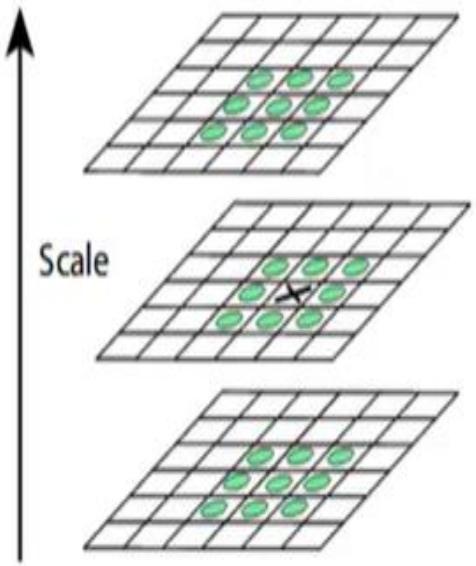
Key point Localization

- 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

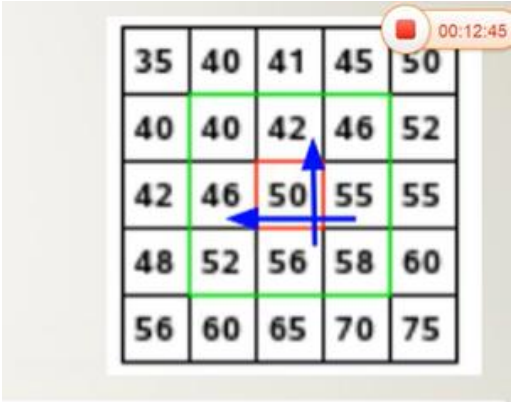
- At this stage, we have a set of stable keypoints for the images. We will now assign an orientation to each of these keypoints so that they are invariant to rotation. We can again divide this step into two smaller steps:

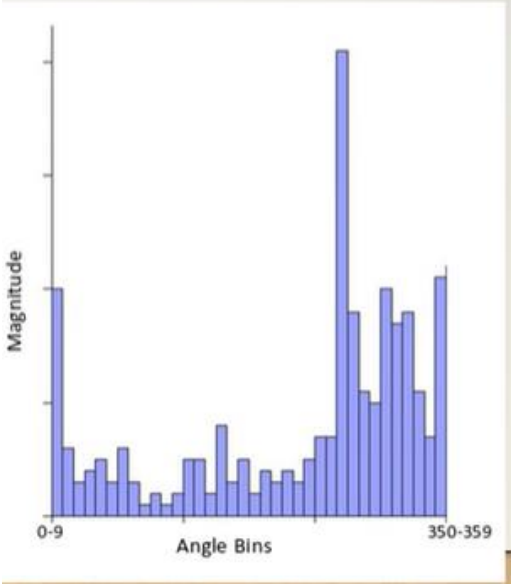
  1. Calculate the magnitude and orientation

  2. Create a histogram for magnitude and orientation

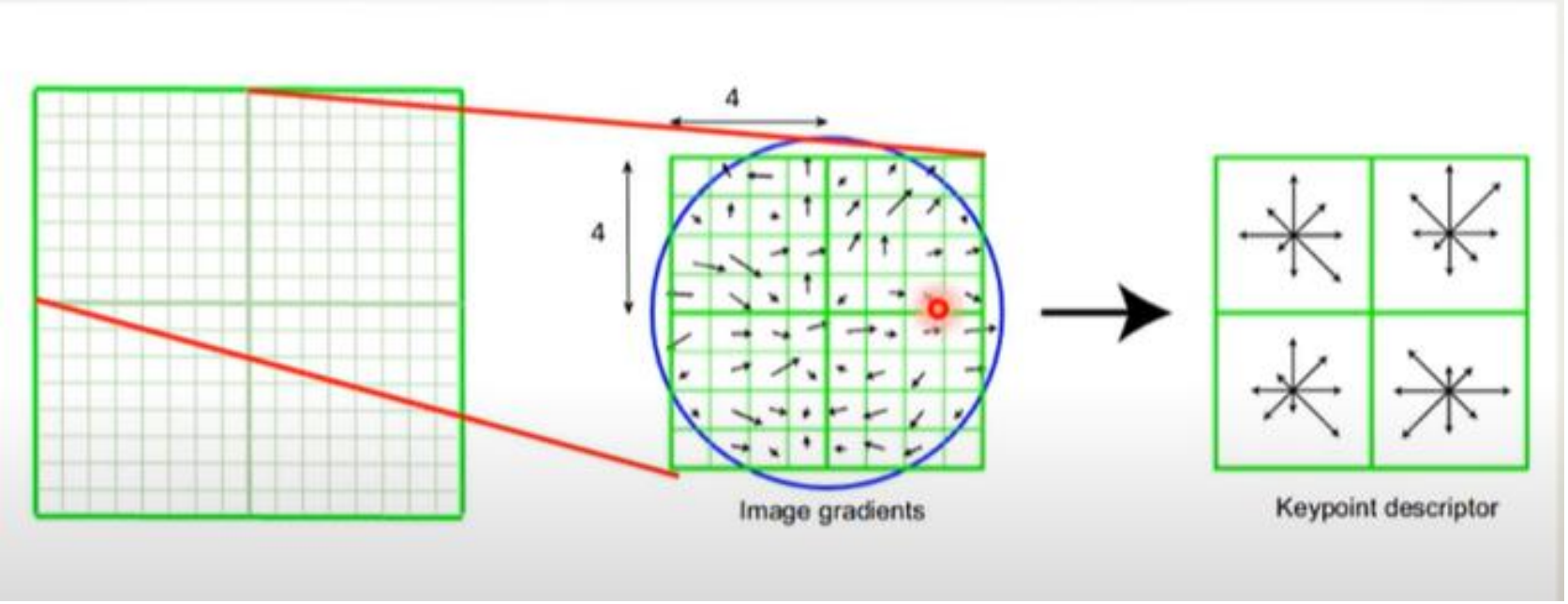
Orientation Assignment

- 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 Gx = 9 and Gy = 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
  - Φ = atan(Gy / Gx) = atan(1.55) = 57.17





Key point Descriptor

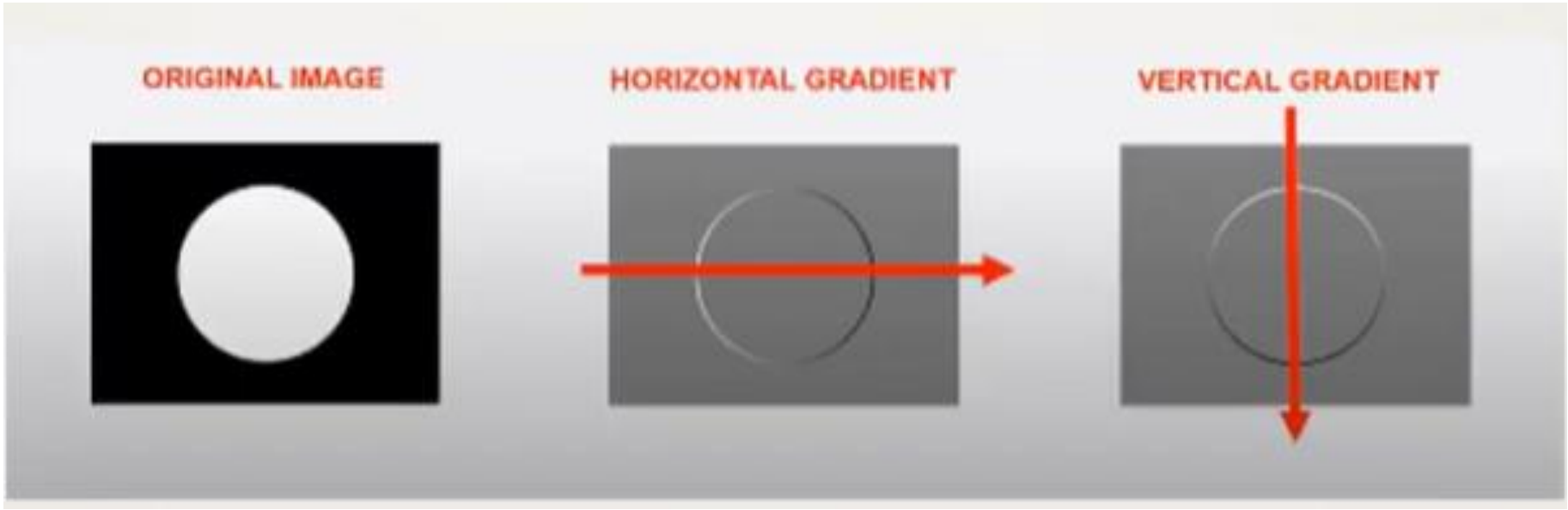


Key point Descriptor

- 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:

Gradient Magnitude = $\sqrt{(50)^2 + (50)^2} = 70.1$

Gradient Angle = $\tan^{-1}(50/50) = 45°$

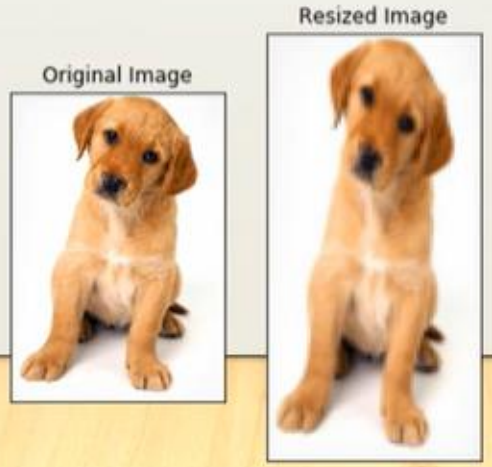
	100	
70		120
	50	

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:

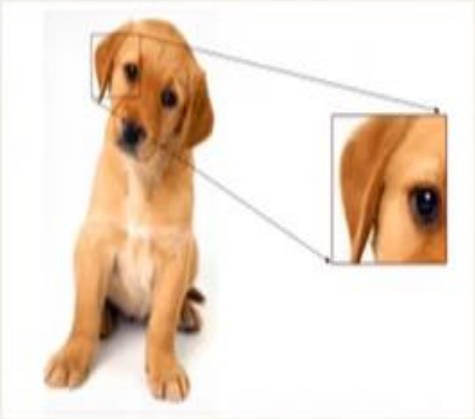
HOG(Histogram of Oriented Gradients)

- **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)

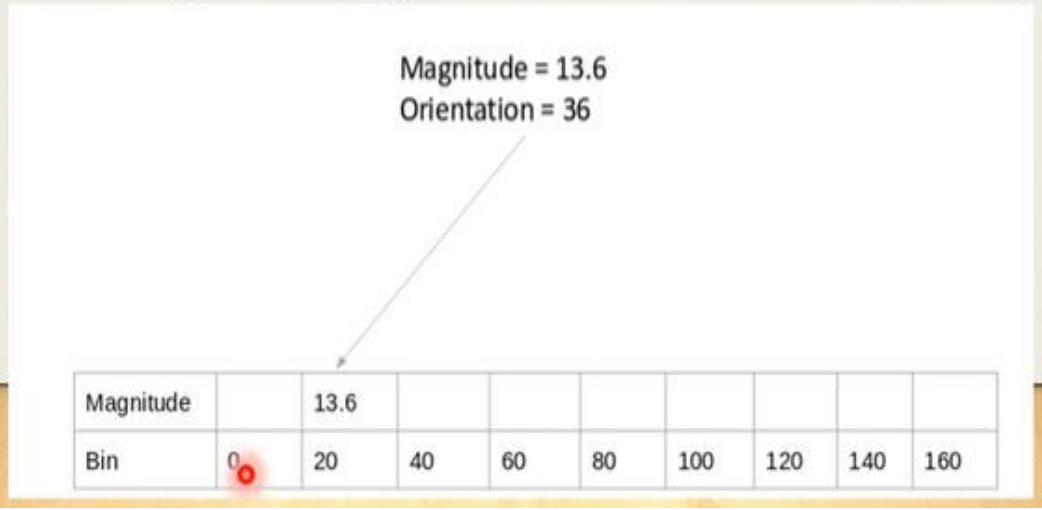
- I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above 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 Gx and Gy 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:
    - $\tan(\Phi) = Gy / Gx$
  - Hence, the value of the angle would be:
    - $\Phi = \text{atan}(Gy / Gx) = \text{atan}(8/11) = 36$

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.



Magnitude		13.6							
Bin	0	20	40	60	80	100	120	140	160

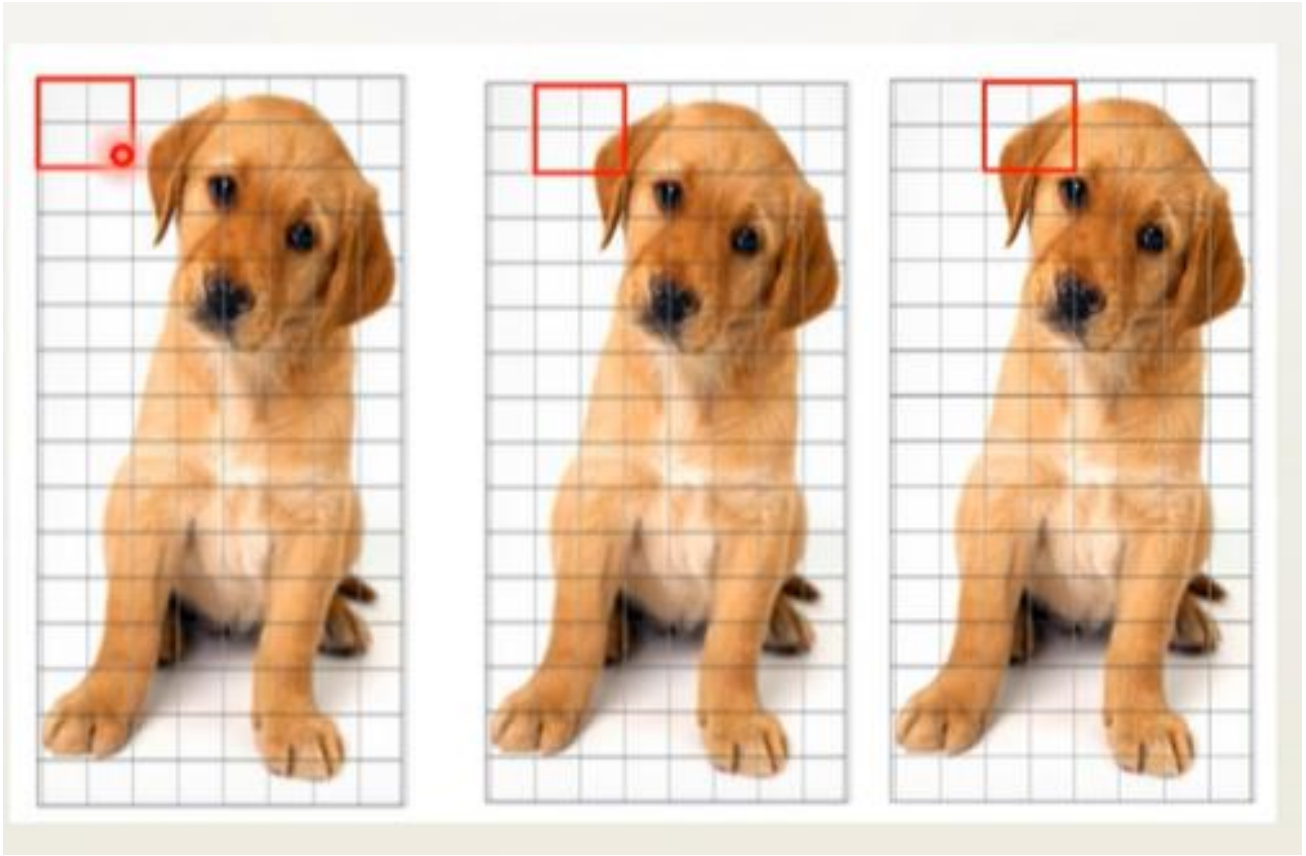
HOG(Histogram of Oriented Gradients)

- **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 x 8 to 16 x 16 or 32 x 32.
  - If we divide the image into 8×8 cells and generate the histograms, we will get a 9 x 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)

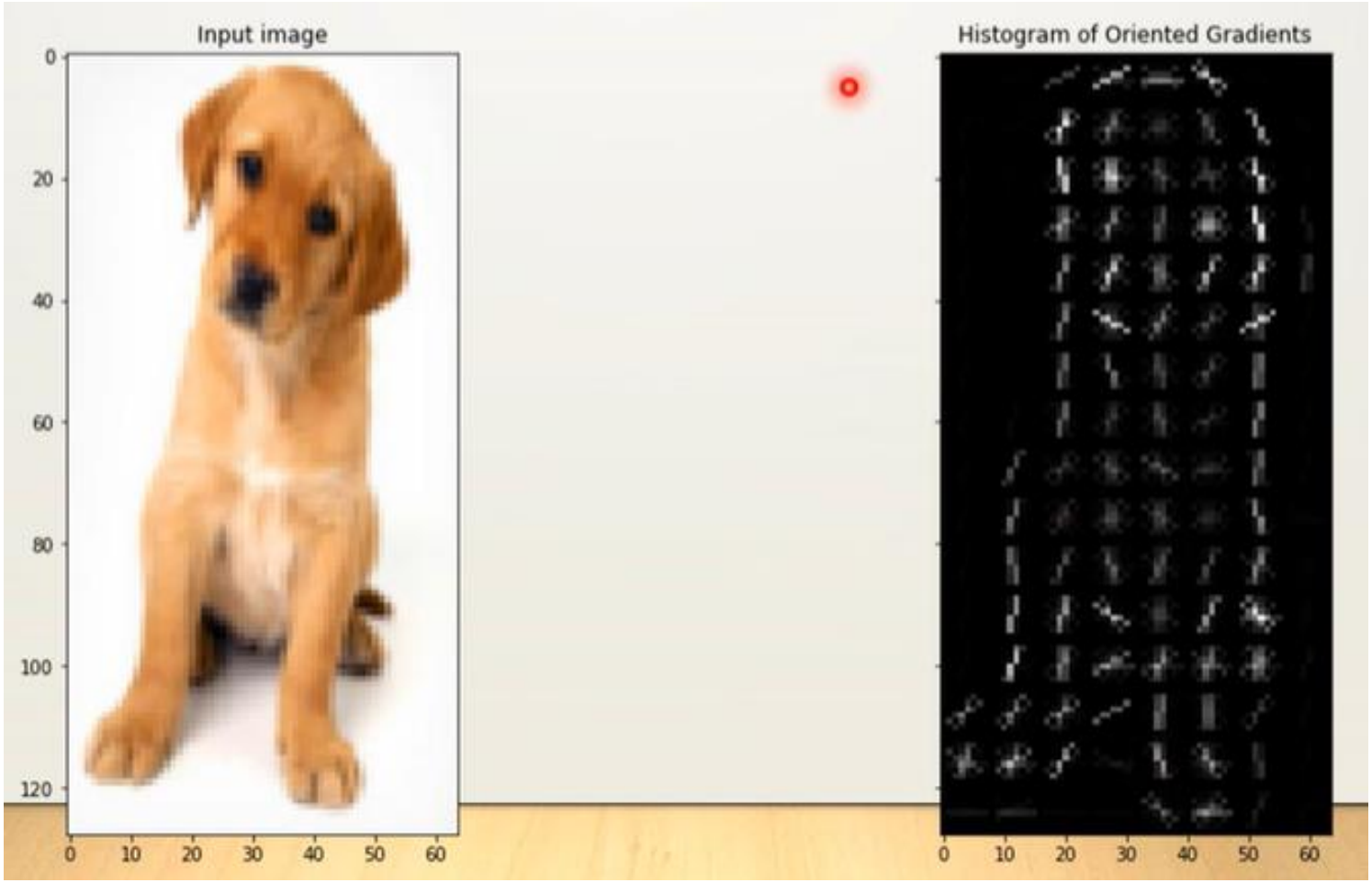


HOG(Histogram of Oriented Gradients)

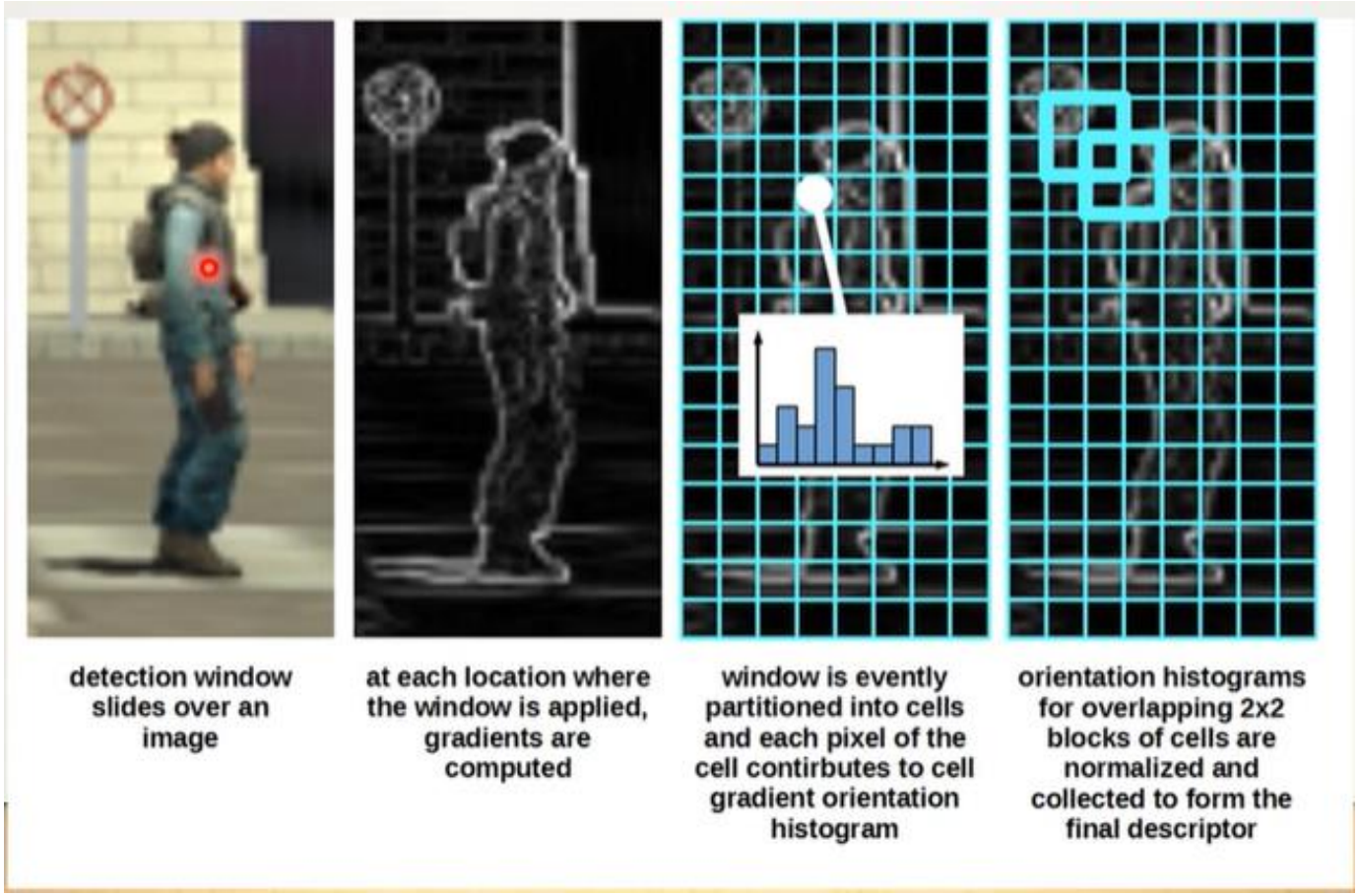
- Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V:
  - V = [a1, a2, a3, ….a36]
- We calculate the root of the sum of squares:
  - $k = \sqrt{(a1)2+ (a2)2+ (a3)2+ \ldots. (a36)2}$
- And divide all the values in the vector V with this value k:

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

HOG(Histogram of Oriented Gradients)



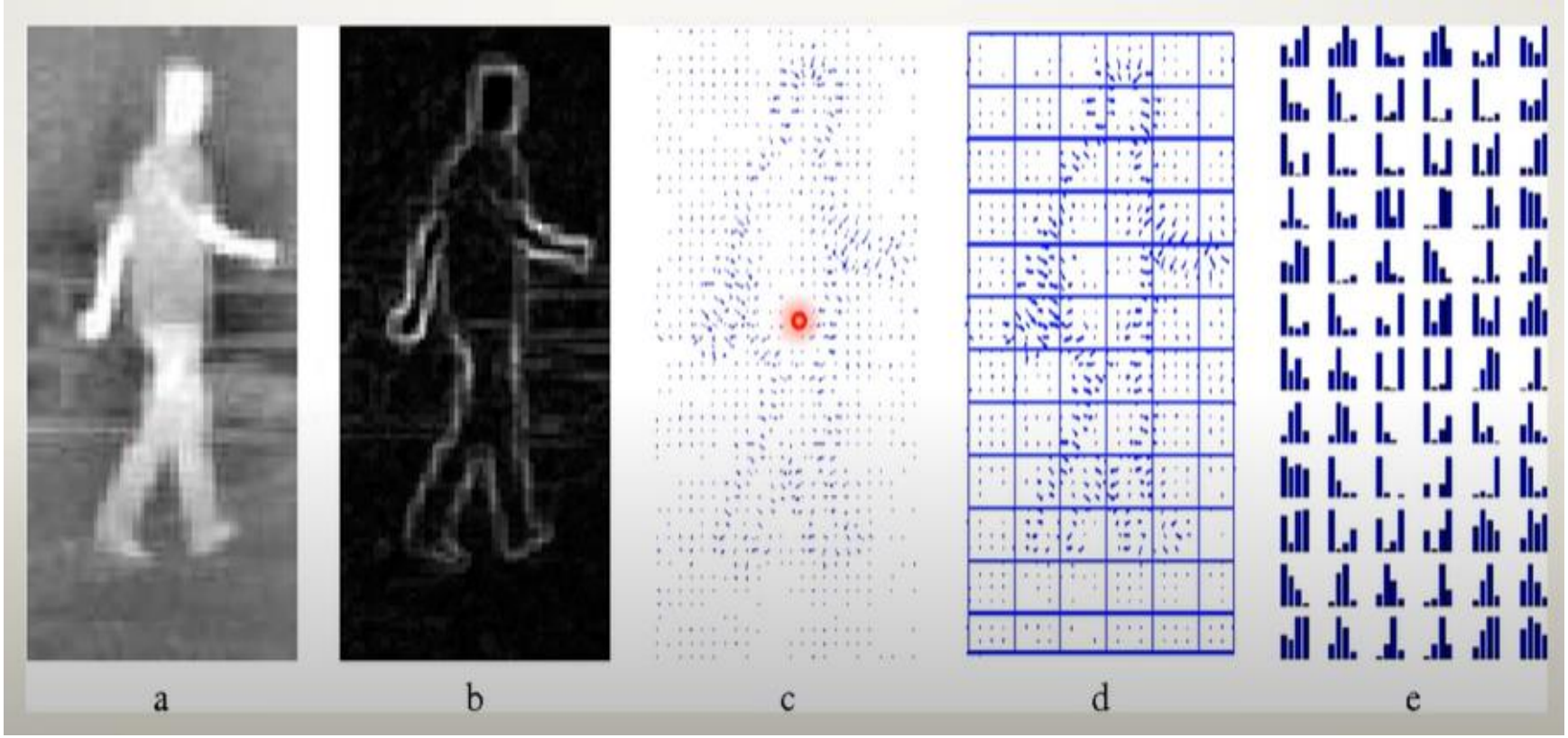
HOG(Histogram of Oriented Gradients)



HOG(Histogram of Oriented Gradients)

HOG(Histogram of Oriented Gradients)

HOG(Histogram of Oriented Gradients)



Thank U!!!!