Image Processing

Lecture 12: Image Segmentation – I

(Ch10 Image Segmentation)

Zhiguo Zhang

zhiguozhang@hit.edu.cn

Review of Last Lecture

- In the last lecture we learnt:
  - Boundary extraction
  - Region filling
  - Extraction of connected components
  - Convex hull
  - Thinning/thickening
  - Skeletons
  - Grayscale morphological processing

Contents of This Lecture

- In this lecture we will learn:
  - The segmentation problem
  - Edge detection
  - Edge linking and boundary detection

Segmentation

- The purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application.
- Segmentation should stop when the objects of interest in an application have been isolated.
- The segmentation is based on measurements taken from the image and might be grey level, colour, texture, depth or motion.

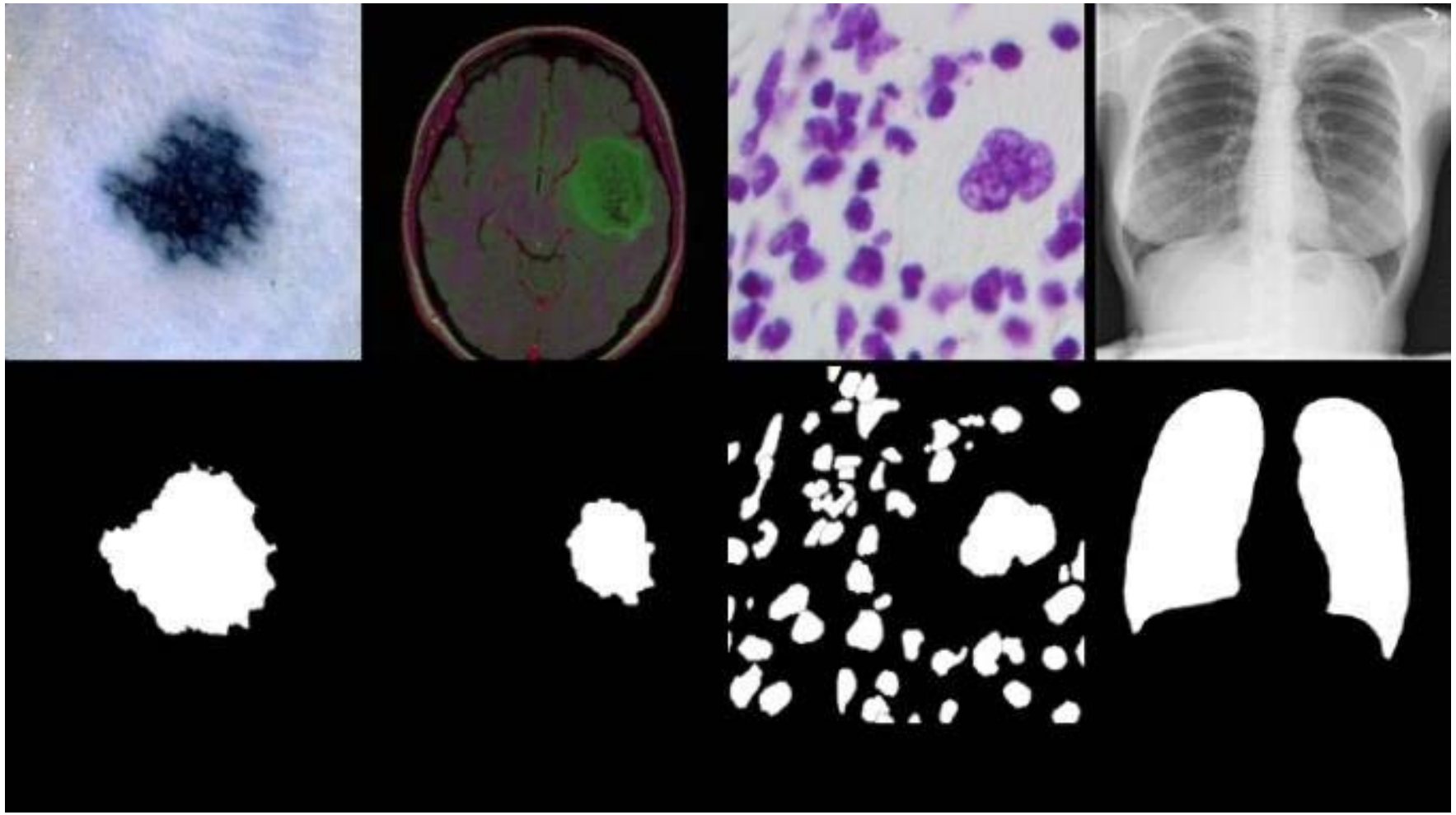
Segmentation

- Typically it is the first step in any automated computer vision application.
- Applications of image segmentation include:
  - Object detection for image analysis
  - Object identification in a scene
  - Path planning for a mobile robot
  - Medical image segmentation for auxiliary diagnosis
  - and many others…

Segmentation

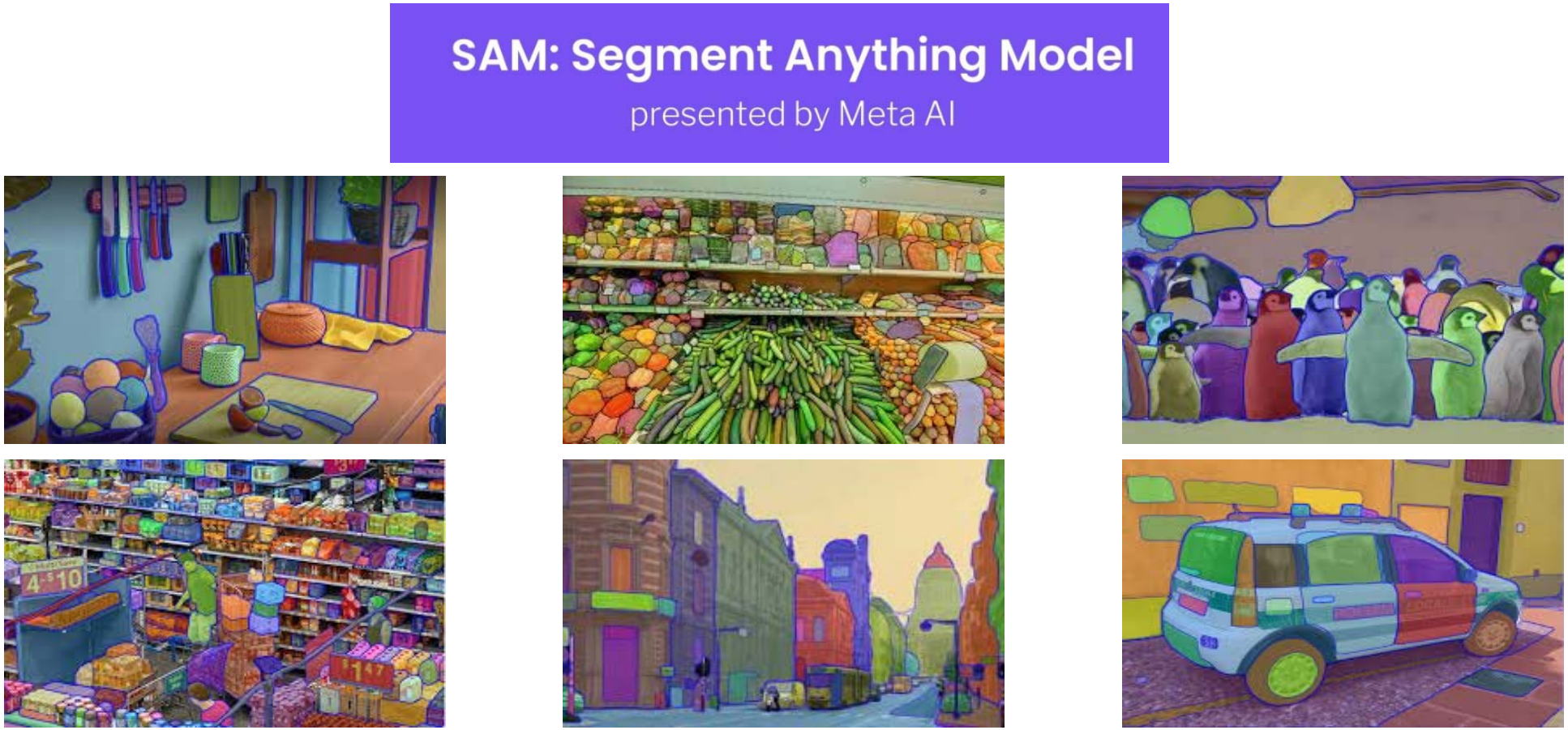
Segmentation

Medical image segmentation



Segmentation

- Image segmentation is one of the most popular and the most difficult tasks in image processing.



Segmentation

- Segmentation algorithms are generally based on 2 basic properties of intensity values:
  - **Discontinuity:** to partition an image based on violent changes in intensity (such as edges).
  - **Similarity:** to partition an image into regions that are similar according to a set of predefined criteria.

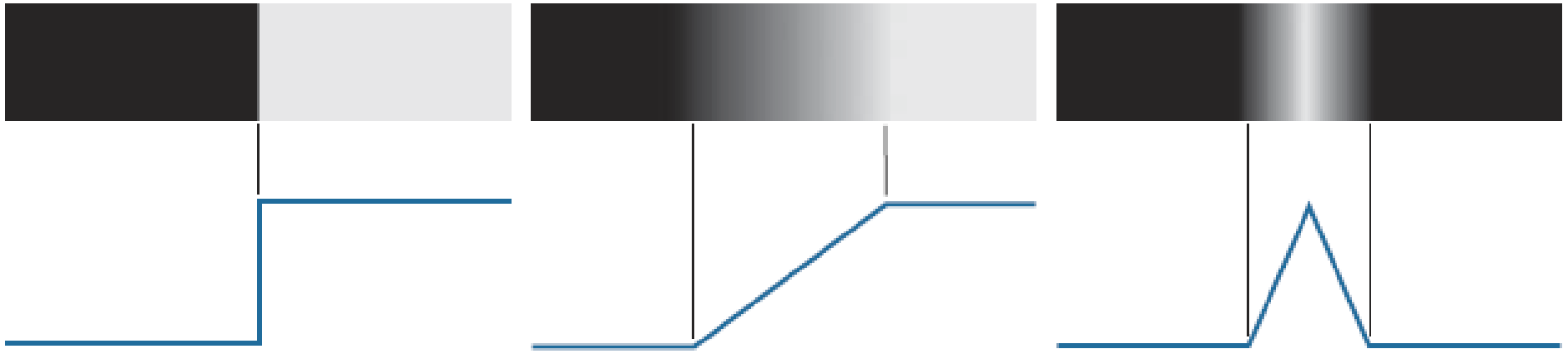
- In this lecture, we will look at segmentation based on *discontinuity*.

Detection of Discontinuities

- There are three basic types of grey level discontinuities that we tend to look for in digital images:
  - Points
  - Lines
  - Edges
- We typically find discontinuities using masks and correlation.

Edge Detection

- Edge detection is the most common approach for detecting meaningful discontinuities in gray level.

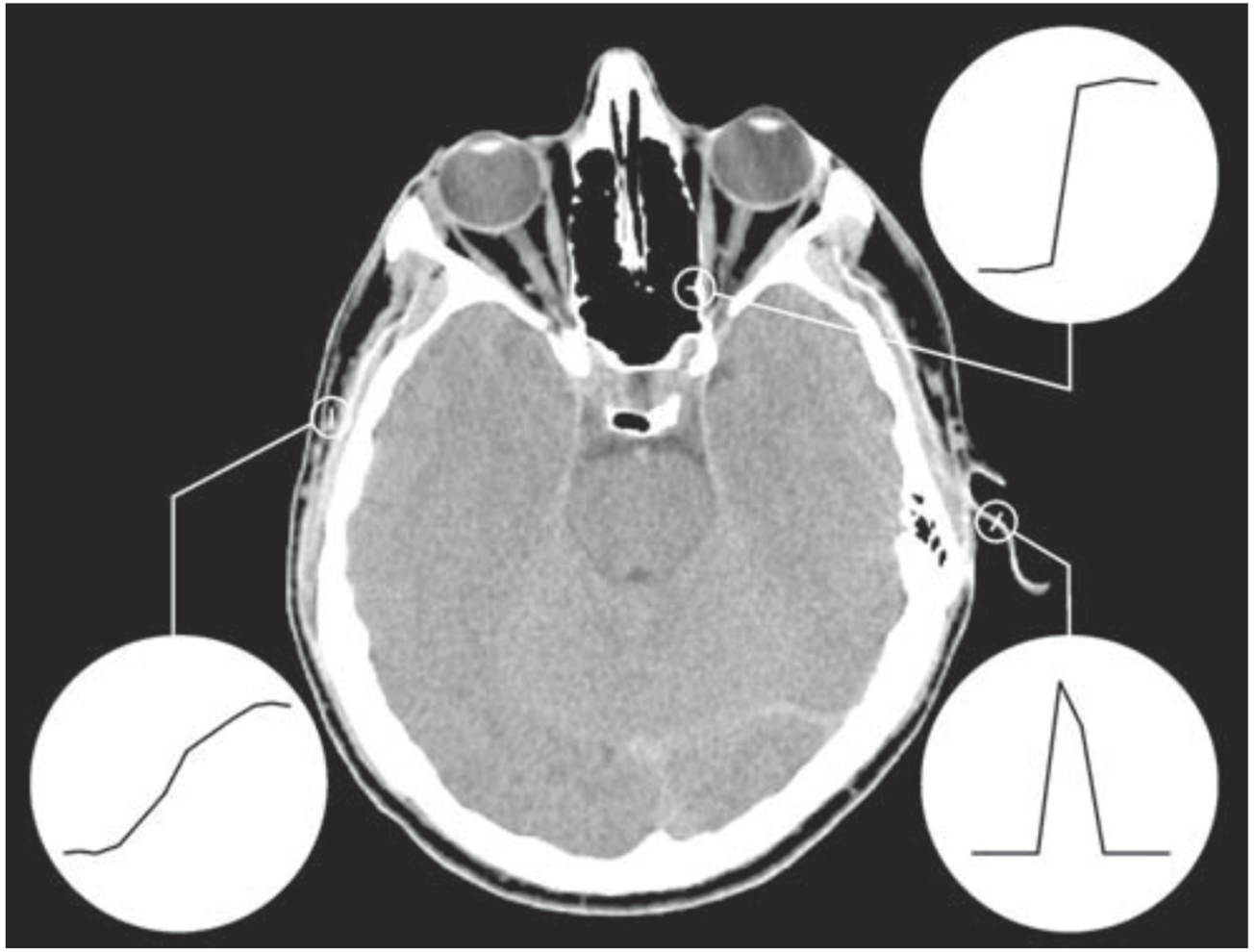


Step edge Ramp edge Roof edge

Thin Edge vs. Thick Edge!

The reasons for blur: optics, sampling, illumination conditions.

Edge Detection

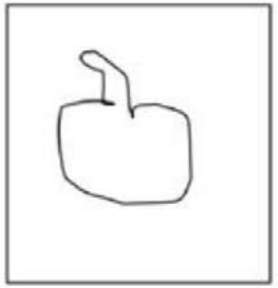


Edges in an actual image (head CT slice)

Edge Detection

- Edge and Region Boundary:
  - An edge is a set of connected pixels that lie on the boundary between two regions.
  - An edge is a "local" concept whereas a region boundary is a more global idea.
  - Typically our goal is to reconstruct the boundary from local edge elements.

Local edge point or edge element



Object boundary

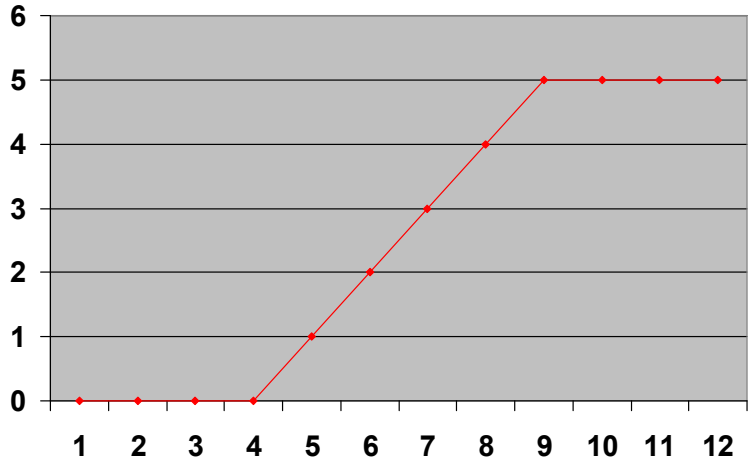
May need domain or object model

Edges
- local intensity discontinuities
- points
- not dependent on models

Boundaries
- extensive
- composed of many points
- maybe dependent on models

Spatial Differentiation

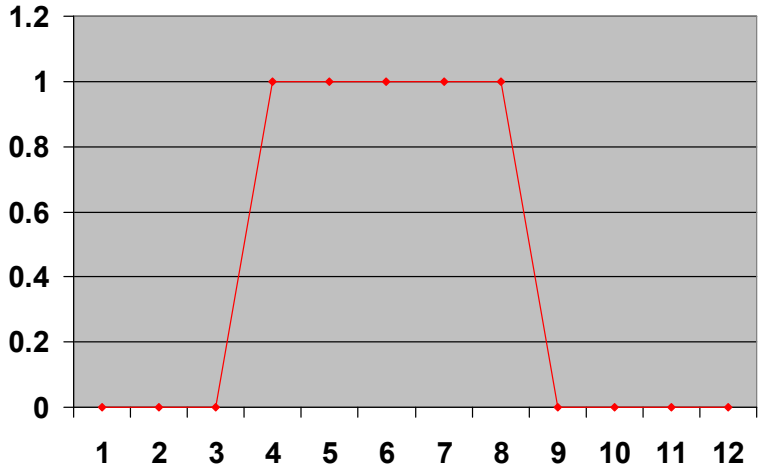
- Spatial differentiation approaches for implementing edge detection:
  - First-order derivative (Gradient operator)
  - Second-order derivative (Laplacian operator)



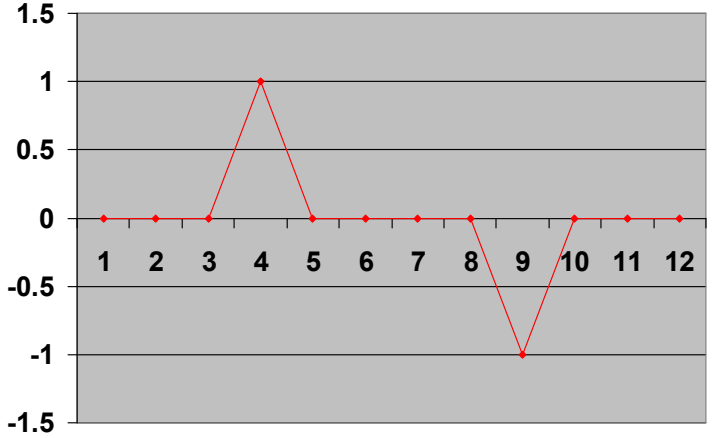
0	0	0	0	1	2	3	4	5	5	5	5

0	0	0	1	1	1	1	1	0	0	0	

	0	0	1	0	0	0	0	-1	0	0	

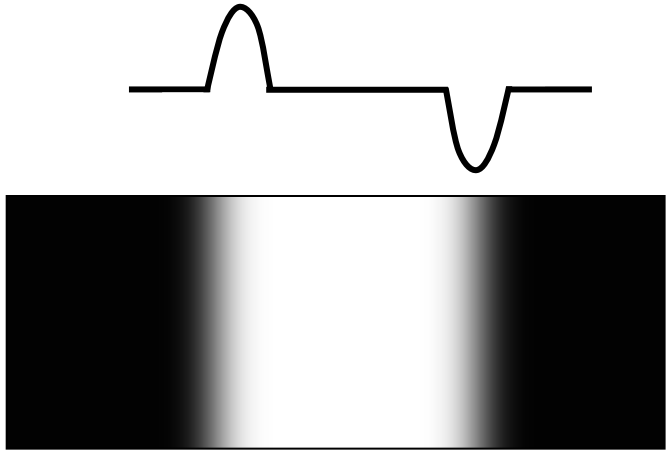


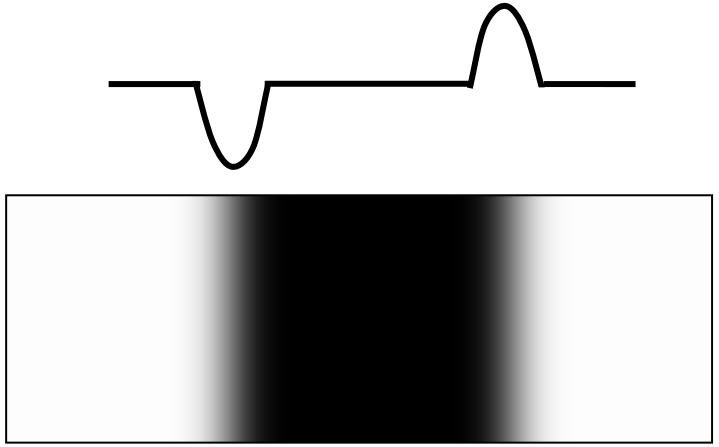
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$



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

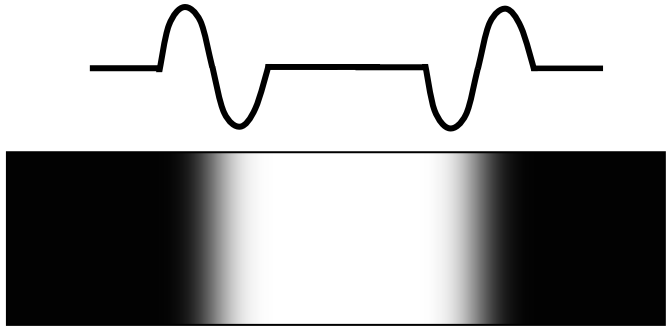
Edges & Derivatives





1st derivative

For the left graph, the left side is positive (dark to light) and the right side is negative (light to dark). For the right graph, the observations are reversed. The constant part is always zero.

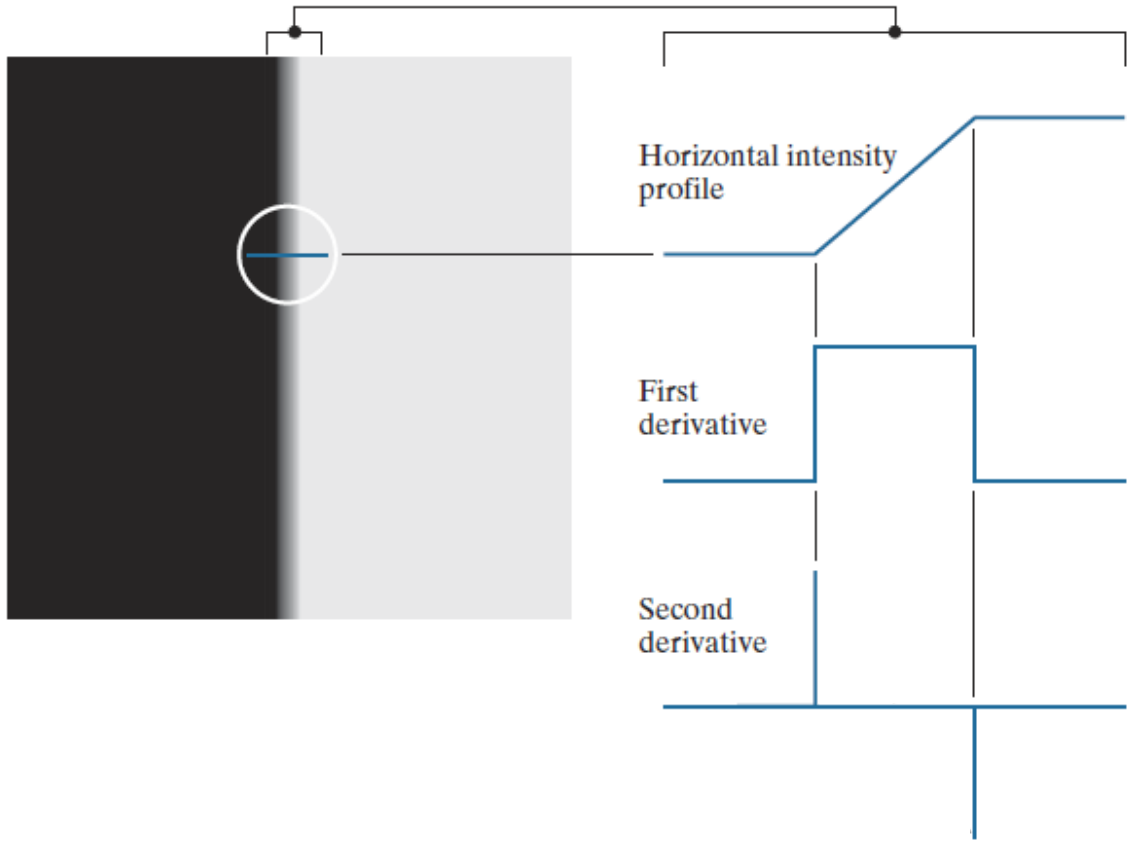


2nd derivative

The second order derivative is negative on the light side and positive on the dark side. The constant part is zero.

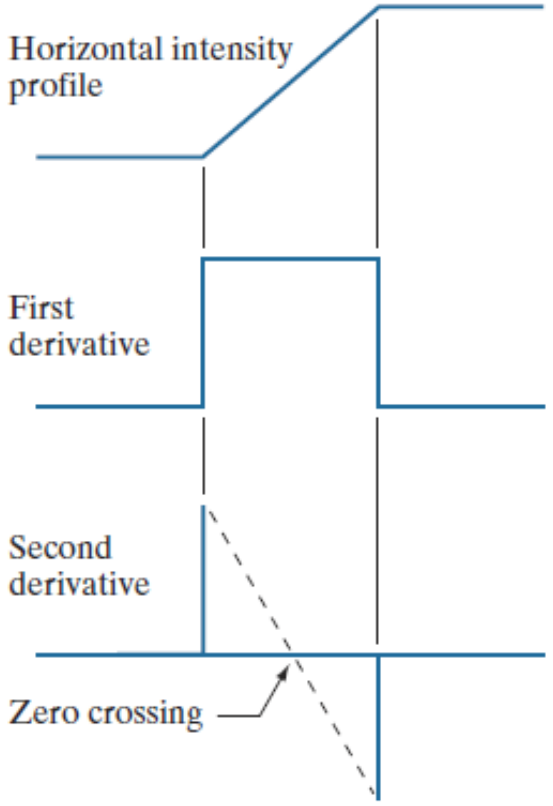
Edges & Derivatives

- We can see how derivatives can be used to find discontinuities.
  - The magnitude of the 1st derivative tells us where an edge is.
  - The sign of the 2nd derivative can be used to show edge direction.



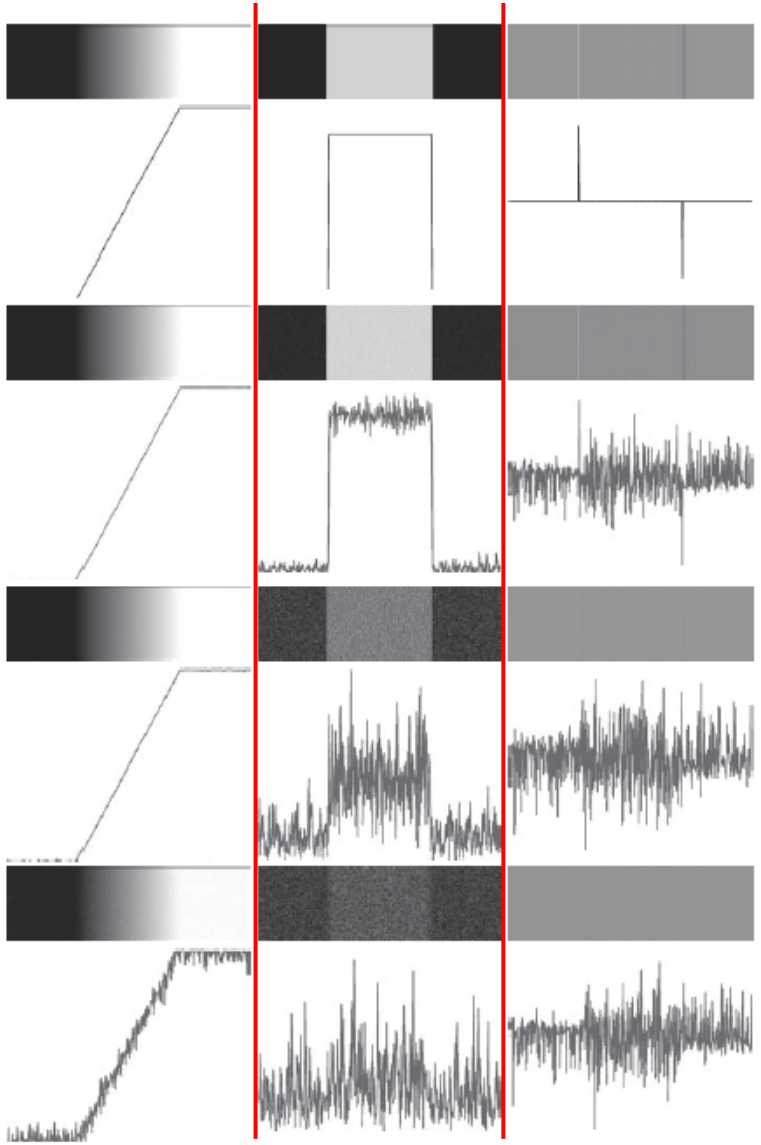
Edges & Derivatives

- Second derivatives:
  - Produces 2 values for every edge in an image (an undesirable feature).
  - An imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge (zero-crossing property).
  - This zero-crossing property is quite useful for locating the centers of thick edges.



Derivatives & Noise

- Derivative-based edge detectors are extremely sensitive to noise.
  - First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and σ = 0.0, 0.1, 1.0, and 10.0.
  - Second column: first derivative images and gray-level profiles.
  - Third column: second first derivative images and gray-level profiles.



Derivatives & Noise

- Fairly little noise can have a significant impact on the two key derivatives used for edge detection in images.

- Image smoothing should be seriously considered prior to the use of derivatives in applications where noise is likely to be present.

Derivatives & Noise

- How to determine a point as an edge point:
  - The transition in grey level associated with the point has to be significantly stronger than the background at that point.
  - Use a threshold to determine whether a value is “significant” or not.
  - The point’s two-dimensional 1st derivative must be greater than a specified threshold.

Firstly, we discuss ***gradient operators*** and ***Laplacian operators***

Gradient Operator

- The gradient vector points in the direction of maximum rate of change of f at (x, y).

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- First derivatives are implemented using the magnitude of the gradient.

$$M(x, y) = \|\nabla f\| = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

- For practical reasons this can be approximated as:

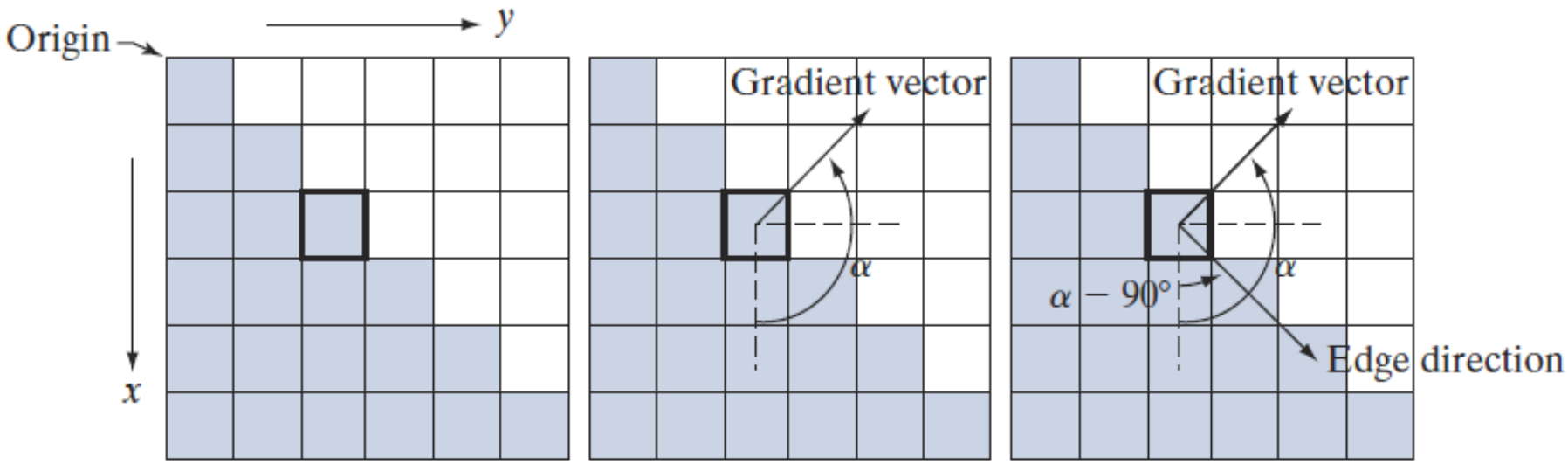
$$M(x, y) \approx |g_x| + |g_y|$$

Gradient Direction

- Let $a(x, y)$ represent the direction angle of the vector ∇f at (x, y):

$$a(x, y) = \tan^{-1}(g_y / g_x)$$

- The direction of an edge at (x, y) is perpendicular to the direction of the gradient vector at that point.



Common Edge Detectors

- The following gradient-based edge detection spatial filters are often used.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

−1	0	0	−1		
0	1	1	0		

Roberts

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

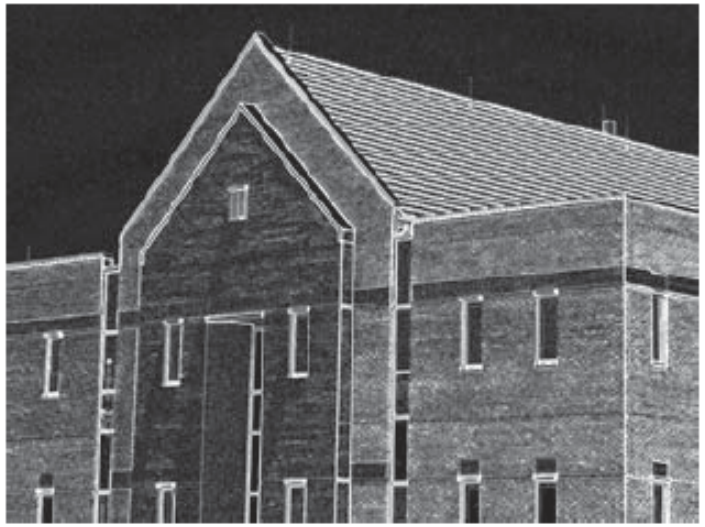
Sobel

Edge Detection Example

(a) Image of size 834 × 1114 pixels, with intensity values scaled to the range [0,1].

(b) g_x, the component of the gradient in the x-direction, obtained using the Sobel kernel.

(c) g_y, the component of the gradient in the y-direction, obtained using the Sobel kernel.



(d) The gradient image, $|g_x| + |g_y|$.

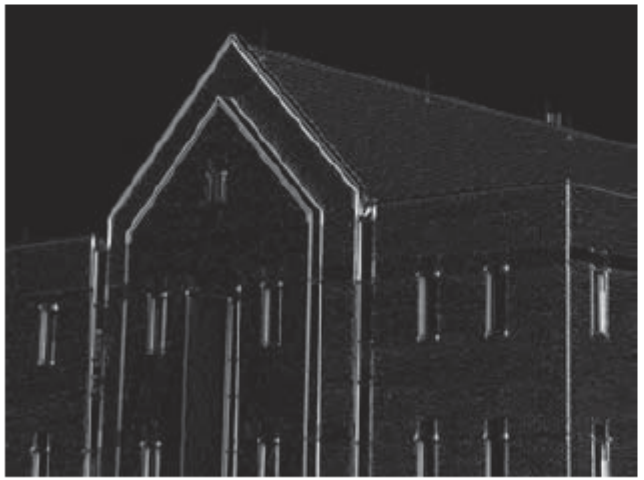
Edge Detection Example

- Often, a problem in edge detection is that there is too much detail.
- One way to overcome this problem is to smooth images prior to edge detection.

Same image as the previous slide,, but with the original image smoothed using a 5 × 5 averaging kernel prior to edge detection.

Original Image

Horizontal Gradient Component



Vertical Gradient Component

Combined Edge Image

Laplacian

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

- The Laplacian is seldom used in practice, because:
  - it is unacceptably sensitive to noise (as second-order derivative)
  - it produces double edges
  - it is unable to detect edge direction

Laplacian

- Usually when used for edge detection the Laplacian is combined with a smoothing Gaussian filter.
- An important use of the Laplacian: to find the location of edges using its zero-crossings property.
- Plus, the Laplacian can play only the role of detector of whether a pixel is on the dark or light side of an edge.

Laplacian of Gaussian

- Laplacian of Gaussian (LoG), which is also known as the Marr-Hildreth edge detector, is a Laplacian combined with smoothing as a precursor to find edges via zero-crossing.

- 2D Gaussian function

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

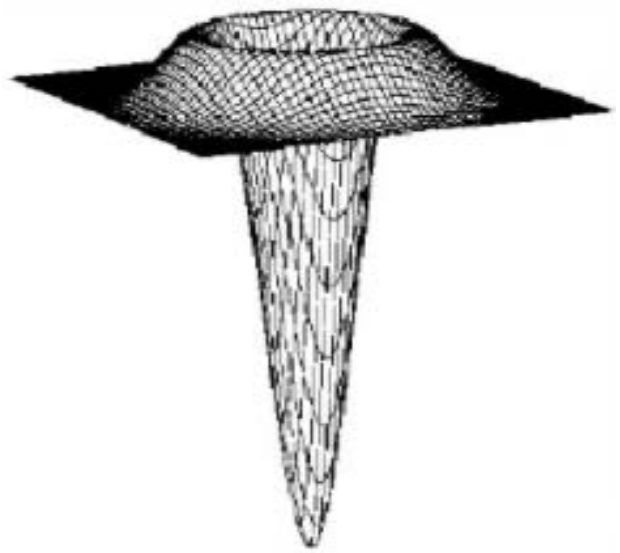
  where σ is the standard deviation.

- Convolving a Gaussian function with an image blurs the image, with the degree of blurring being determined by the value of σ.

Laplacian of Gaussian

- The Laplacian of the above Gaussian is:

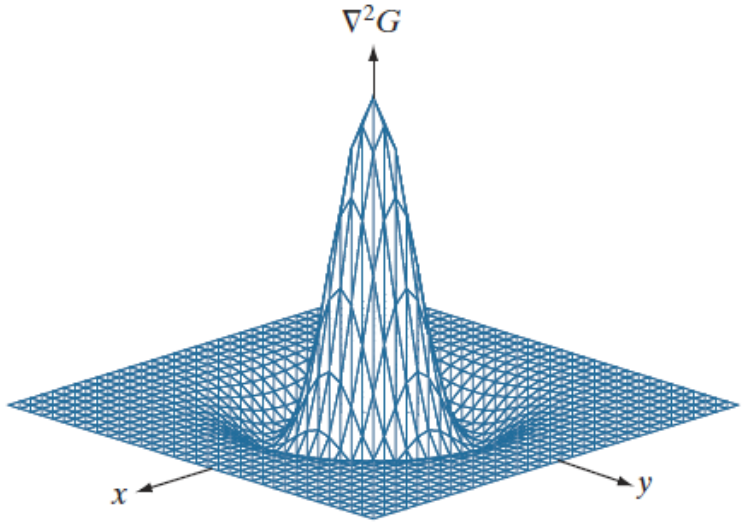
$$\nabla^2 G(x,y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

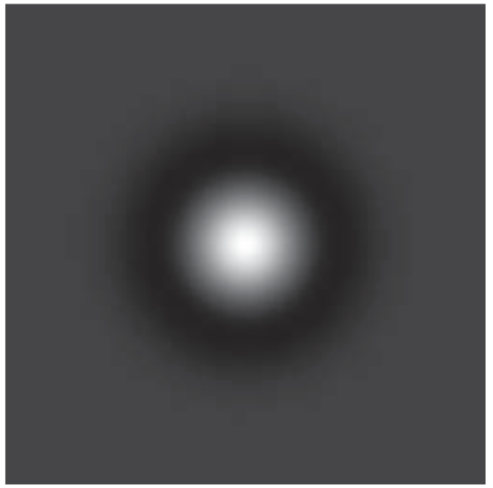


σ determines the degree of blurring that occurs.

Laplacian of Gaussian

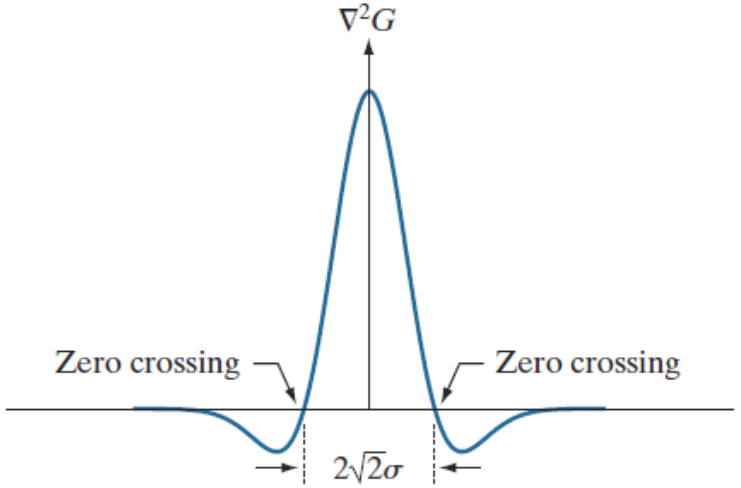
(a) 3-D plot of the *negative* of the LoG.





(b) Negative of the LoG displayed as an image.

(c) Cross section of (a) showing zero crossings.



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

(d) 5 × 5 kernel approximation to the shape in (a).

Laplacian of Gaussian

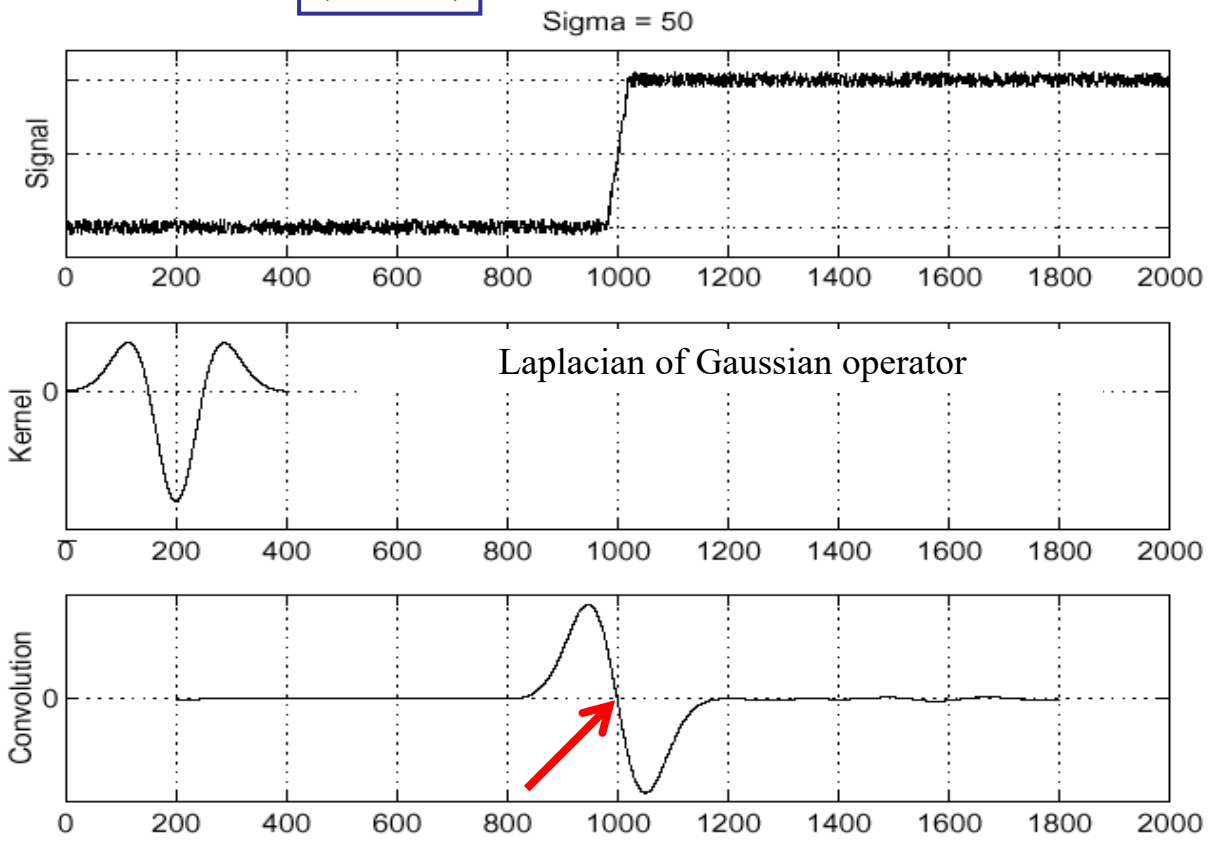
$$\frac{\partial^2}{\partial x^2}(g * f) = \left(\frac{\partial^2}{\partial x^2} g\right) * f$$

Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2} g$

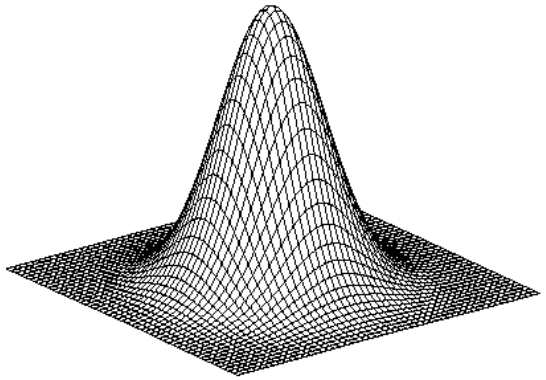
$\left(\frac{\partial^2}{\partial x^2} g\right) * f$



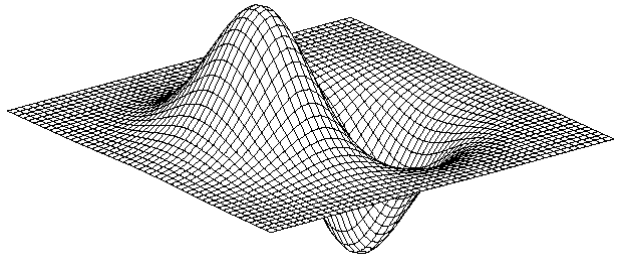
Where is the edge?

Zero-crossings of bottom graph !

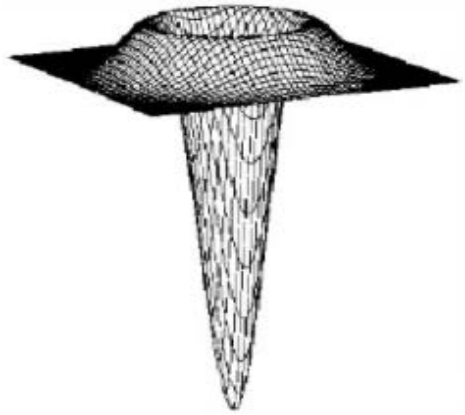
Laplacian of Gaussian



Gaussian



Derivative of Gaussian (DoG)



Laplacian of Gaussian

Mexican Hat (Sombrero)

- The Laplacian of Gaussian (LoG) filter uses the Gaussian for noise removal and the Laplacian for edge detection

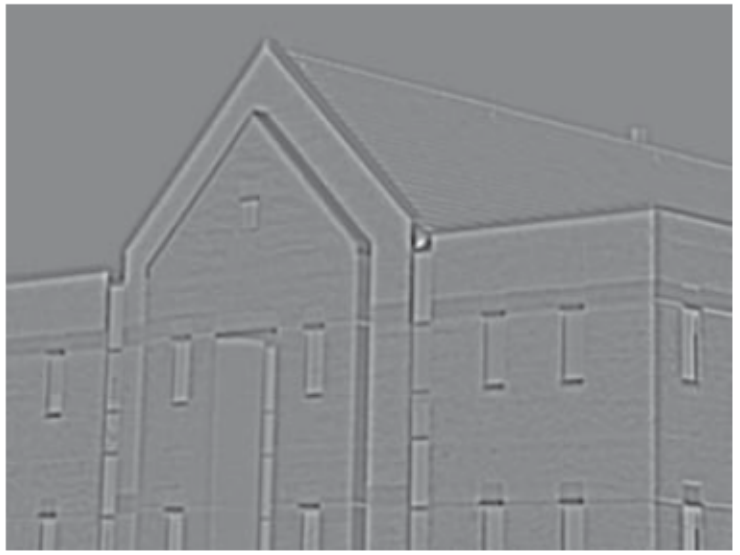
Laplacian of Gaussian

Remarks:

- Because the second derivative is a linear operation, convolving an image with LoG is the same as convolving the image with the Gaussian smoothing function first, and then computing Laplacian.
- The purpose of Gaussian in the LoG is to smooth the image.
- The purpose of Laplacian is to establish the location of edges (zero crossing).

Laplacian of Gaussian

(a) Image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].



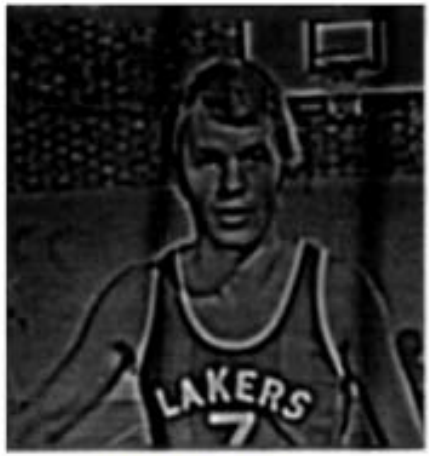
(b) Result of LoG using $\sigma = 4$ and $n = 25$.

(c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges).

(d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

Laplacian of Gaussian

Original Image



LoG Filter

Zero Crossings

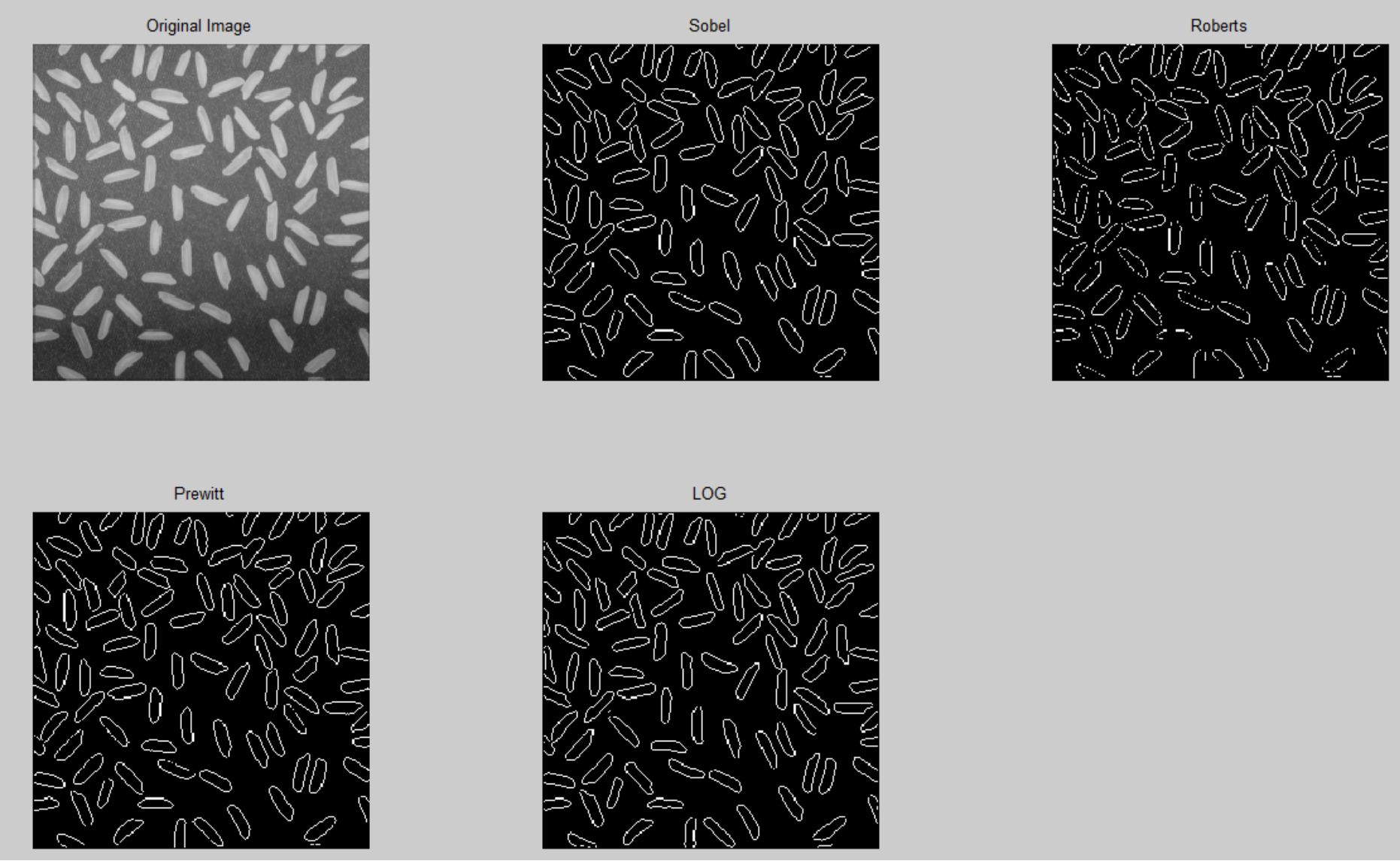
Small

Scale (σ)

Large

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian



The Canny Edge Detector

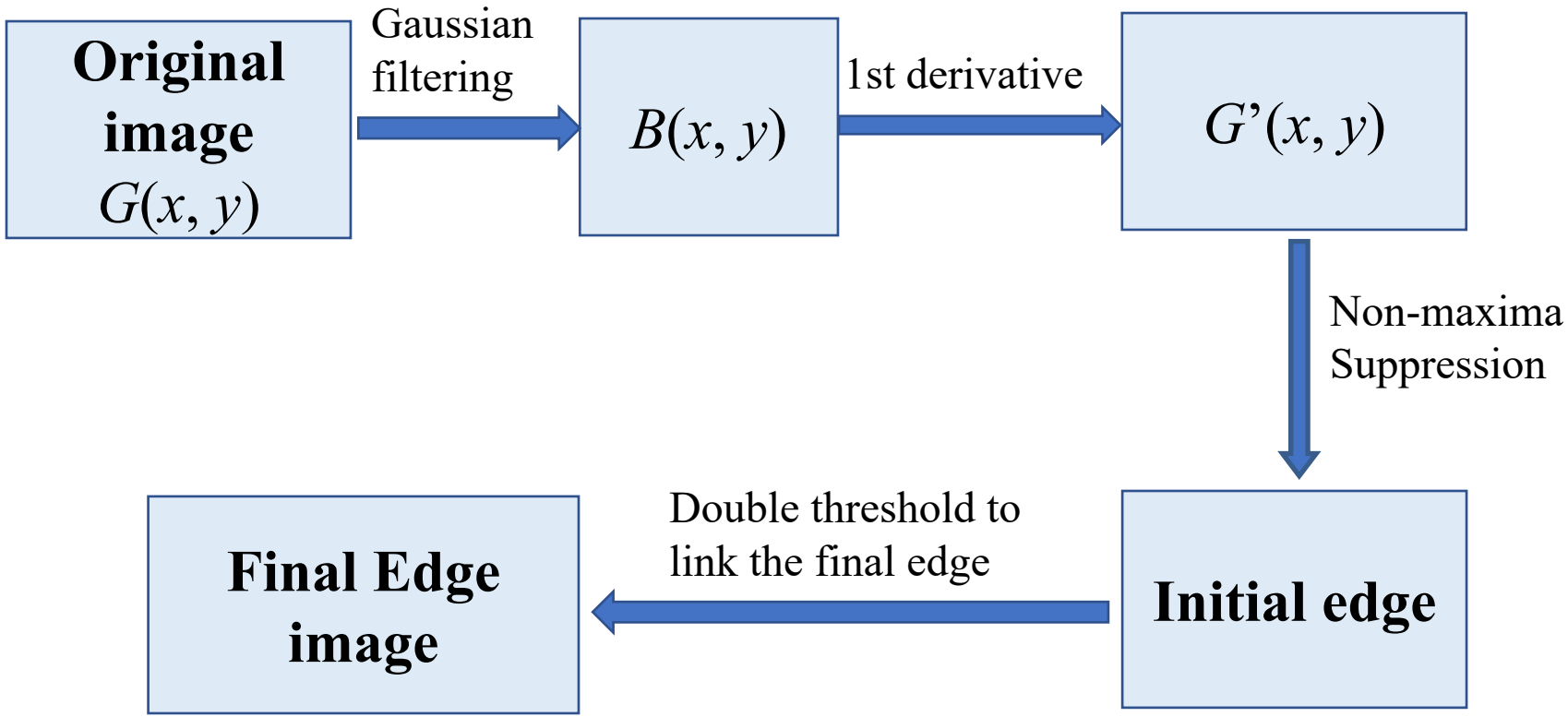
- Canny's approach is based on three basic objectives:

  1. *Low error rate.* All edges should be found, and there should be no spurious responses.

  2. *Edge points should be well localized.* The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.

  3. *Single edge point response.* The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exists.

The Canny Edge Detector



The Canny Edge Detector

Steps:

1. Apply directional derivatives of Gaussian

2. Compute gradient magnitude and gradient direction

3. Non-maximum suppression

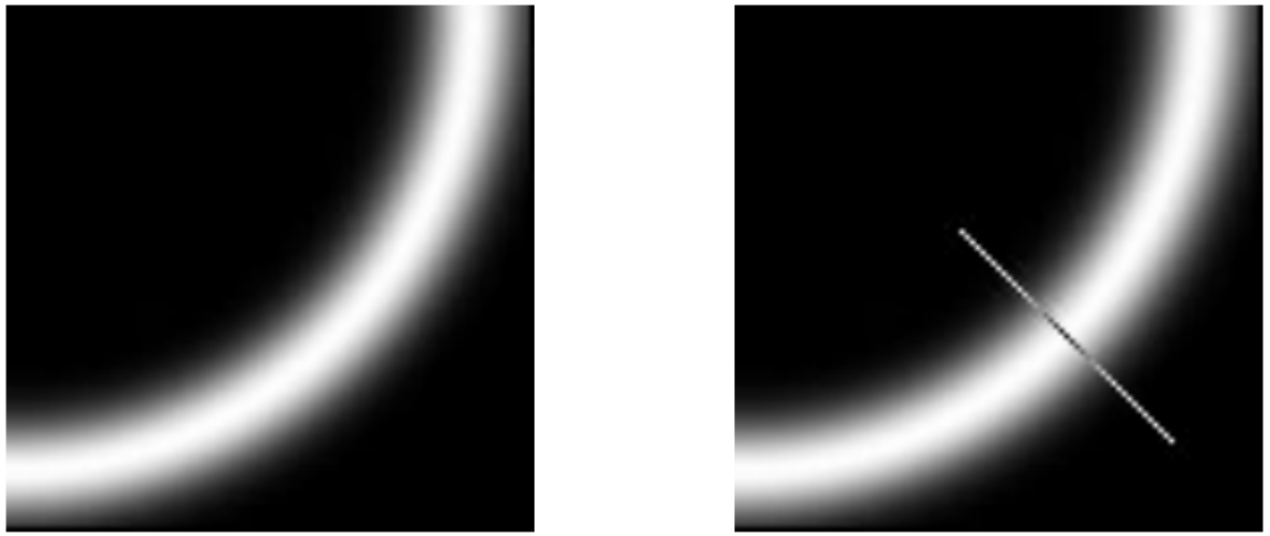
- Thin multi-pixel wide “ridges” down to single pixel width

4. Thresholding and linking

- Low, high edge-strength thresholds
- Accept all edges over low threshold that are connected to edge over high threshold

The Canny Edge Detector

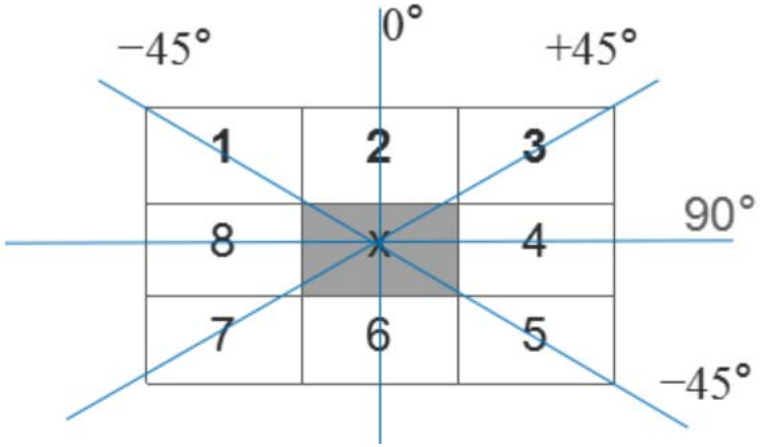
- Gradient image typically contains wide ridges around local maxima. The next step is to thin those ridges. One approach is to use *non-maxima suppression*.



Select the image maximum point across the width of the edge

The Canny Edge Detector

- Non-maxima Suppression (NMS): find the max gradient along the angle, which is the edge point



- At each point, the center of the neighborhood x is compared to two pixels along its corresponding gradient direction, and if the center pixel is the maximum, it is retained, otherwise the center is set to 0. This suppresses non-maxima and retains the point with the largest local gradient to get a refined edge.
- Note: Non-maximum suppression of magnitude along the gradient direction, not the edge direction.

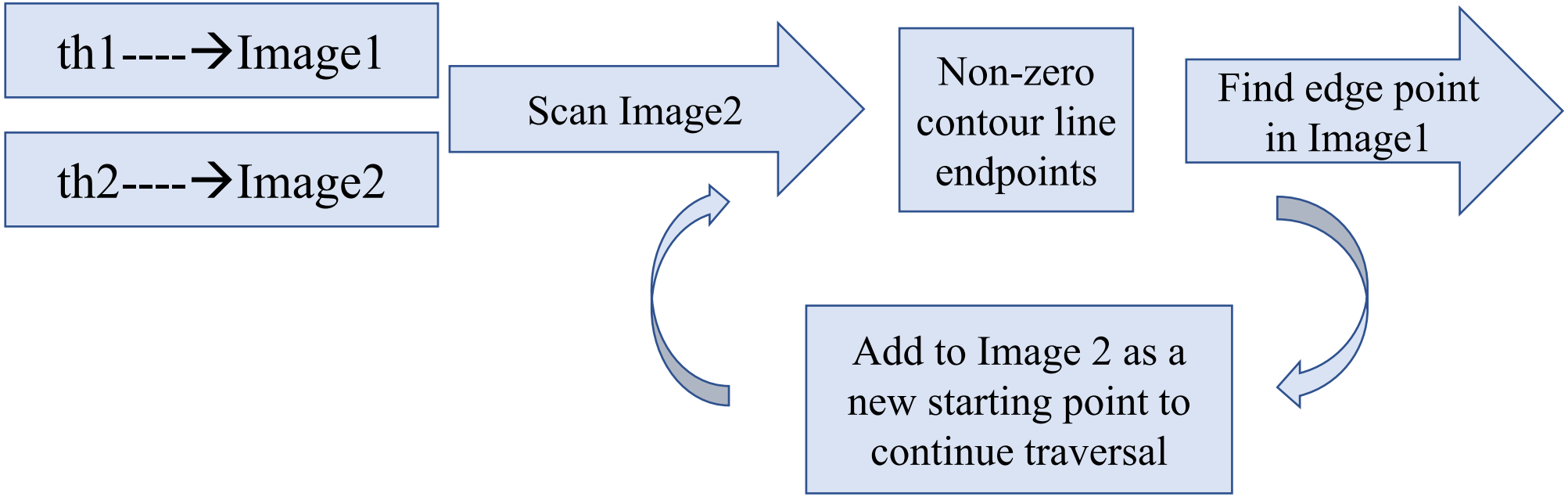
The Canny Edge Detector

- Step 4 - 1. Double thresholding：
  - Usually, single threshold cannot get good results, since the perfect threshold is difficult to choose.
  - The double thresholds are the better way: th1, th2, th1=0.4th2
  - Then we can get two edge image: I1, I2.
  - I2 is gotten by the bigger threshold th2, which removes more noise, and also loses some useful edge. I1 contains more detail edge, so we can combine them together.

The Canny Edge Detector

- Step 4 - 2. Edge linking:
  - Scan I2, when find a non-zero point p, take p as a start point to track the edge to its end point q.
  - In I1, consider the same location (q') of q (in I2) and its corresponding 8-neighbourhood. If there is any non-zero point, then add it into I2, and mark it as r. Then take r as a start point, repeat step a until no more points can be found.
  - Until now, we finish the edge detection which includes point p, and mark it as scanned. Then, go to find next edge by above steps.

The Canny Edge Detector



The Canny Edge Detector

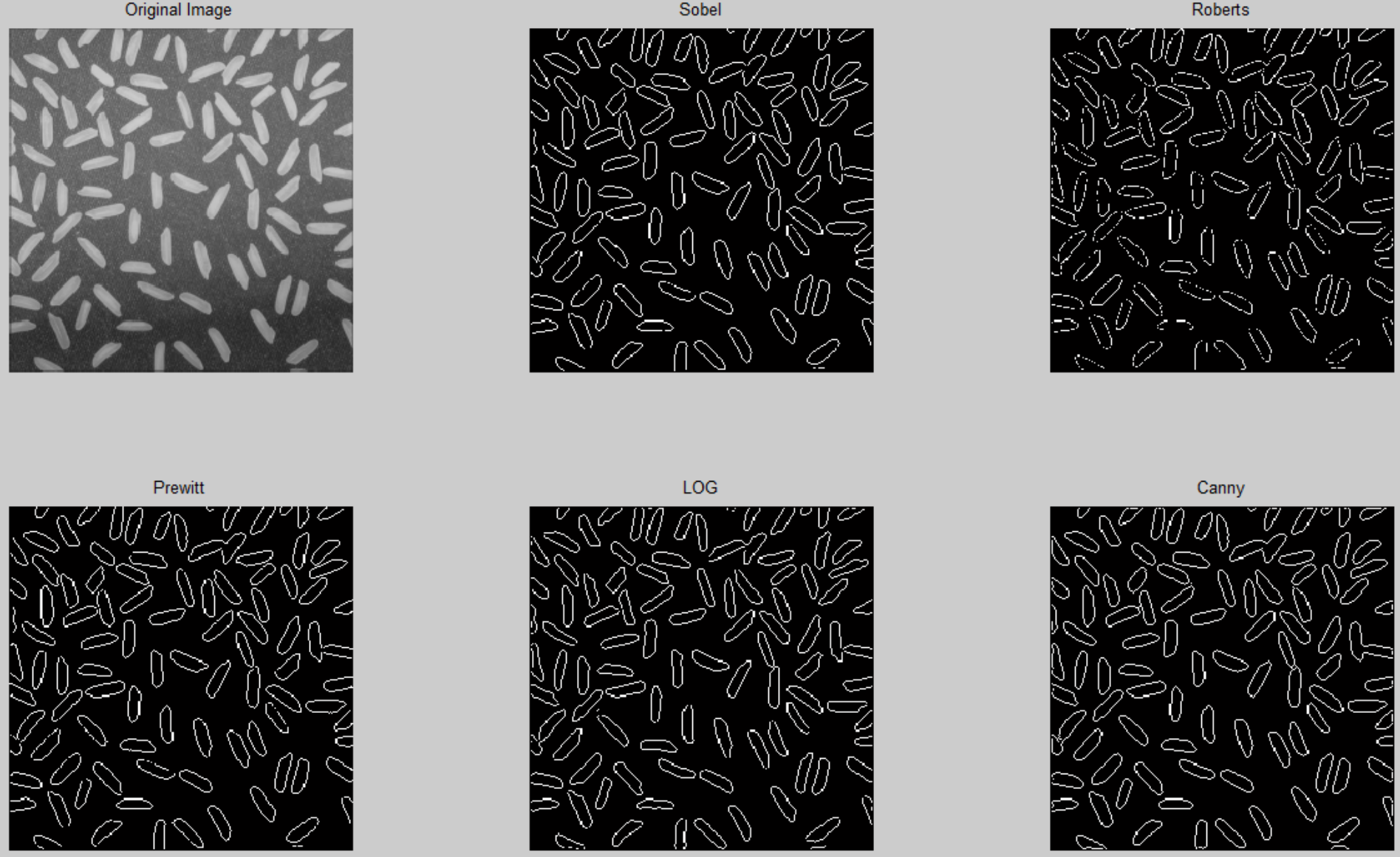
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].

(b) Thresholded gradient of the smoothed image.

(c) Image obtained using the LoG algorithm.

(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

The Canny Edge Detector



Edge Link and Boundary Detection

- Edge detection algorithm are followed by linking procedures to assemble edge pixels into meaningful edges.
- Basic approaches:
  - Local processing
  - Global processing via the Hough Transform

Local Processing

- Analyze the characteristics of pixels in a small neighborhood (e.g. 3×3, 5×5) about every edge pixels (x, y) in an image.
- All points that are similar according to a set of predefined criteria are linked, forming an edge of pixels that share those criteria.

Local Processing

- The strength of the response of the gradient operator
  - An edge pixel with coordinates (x_0, y_0) in a predefined neighborhood of (x, y) is similar in magnitude to the pixel at (x, y) if

$$\left|\nabla f(x,y) - \nabla f(x_0,y_0)\right| \le E$$

- The direction of the gradient vector
  - An edge pixel with coordinates (x_0, y_0) in a predefined neighborhood of (x, y) is similar in angle to the pixel at (x, y) if

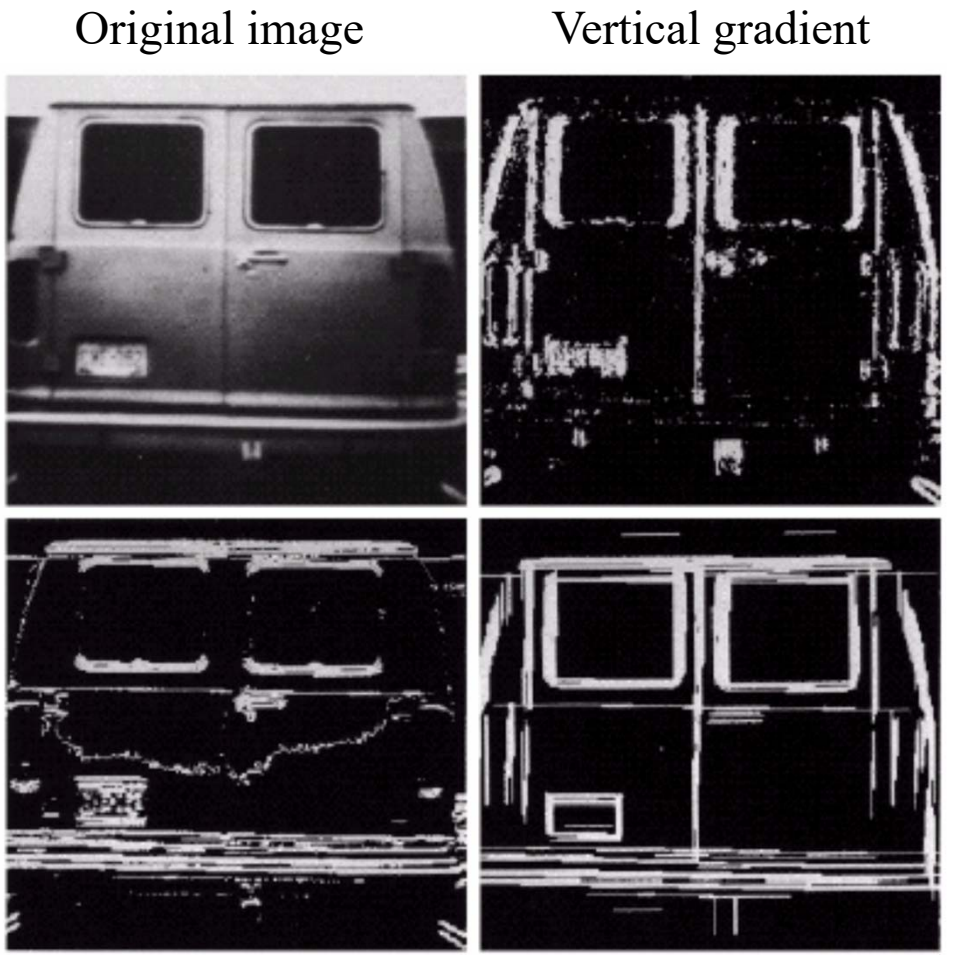
$$\left|\alpha(x,y) - \alpha(x_0,y_0)\right| \le A$$

Local Processing

- Criteria
  - A point in the predefined neighborhood of (x, y) is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied.
  - The process is repeated at every location in the image.
  - As the center of the neighborhood is moved from pixel to pixel, a record of linked points is kept.
  - A simple bookkeeping procedure is to assign a different intensity value to each set of linked edge pixels.

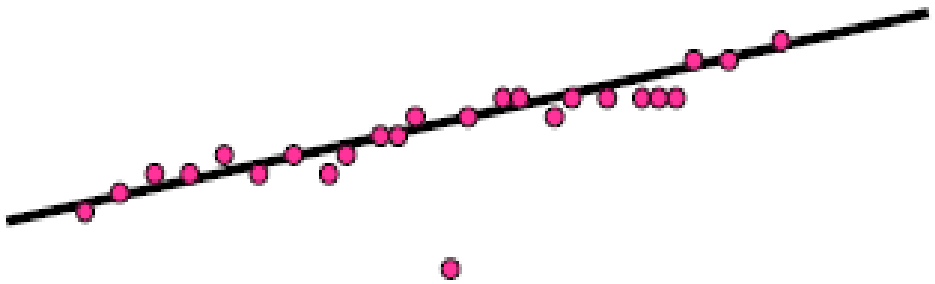
Local Processing

- Use vertical and horizontal Sobel operators.
- Link conditions: gradient value > 25; gradient direction difference < 15.
- Eliminates isolated short segments, links little breaks.



Global Processing

- Find lines, curves, or parts of lines or curves in an input image.
- Such an image might be the output of a edge detector discussed in the previous lectures.
- Model fitting: given a group points, find the curve or line that explains the data points best.
- For example: Line.



Global Processing

- Solution:
  - Find all lines determined by every pair of points
  - Find all subsets of points close to particular lines

  - Involves: $n(n-1)/2 \sim n^2$ lines (given n points in an image)

    $(n-2)(n(n-1))/2 \sim n^3$ computations for comparing every point to all lines.

*A solution: Hough Transform

Hough Transform

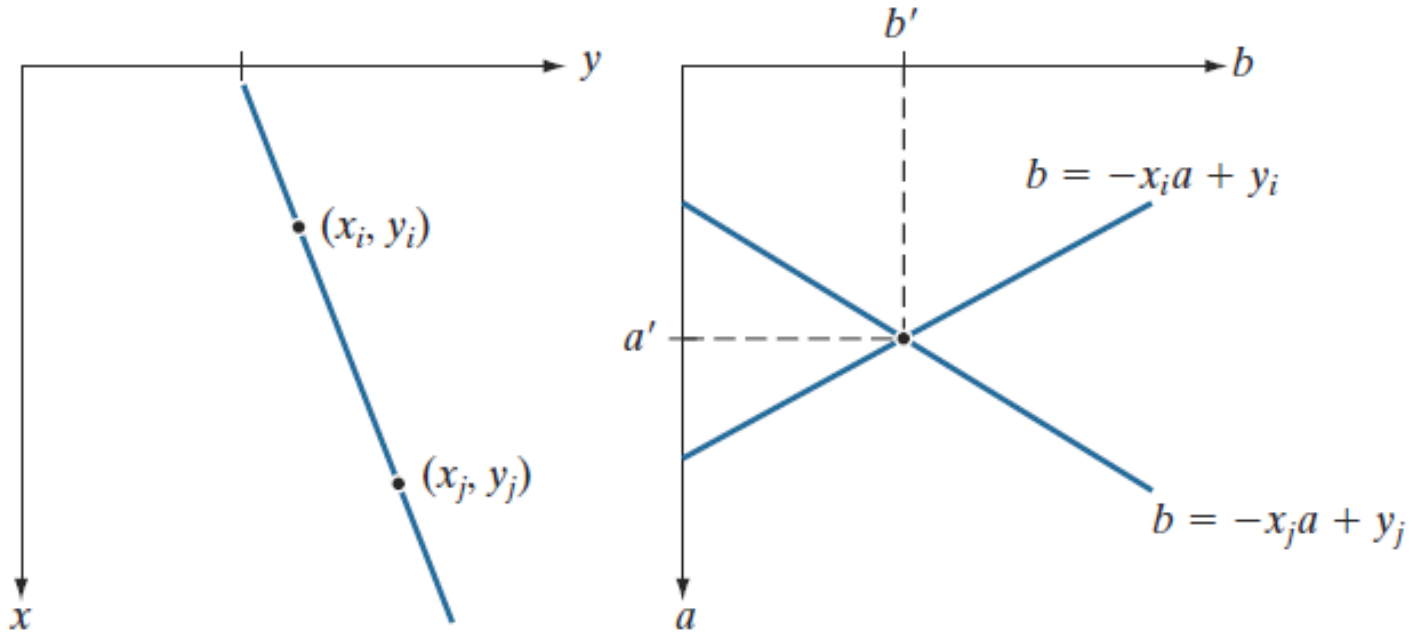
$$y_i = ax_i + b$$
$$y_j = ax_j + b$$

Hough Transform

$$b = -x_i a + y_i$$
$$b = -x_j a + y_j$$

Original space

Parameter Space



$$b' = -x_i a' + y_i$$
$$b' = -x_j a' + y_j$$

$$y_i = x_i a' + b'$$
$$y_j = x_j a' + b'$$

Points-Lines Duality

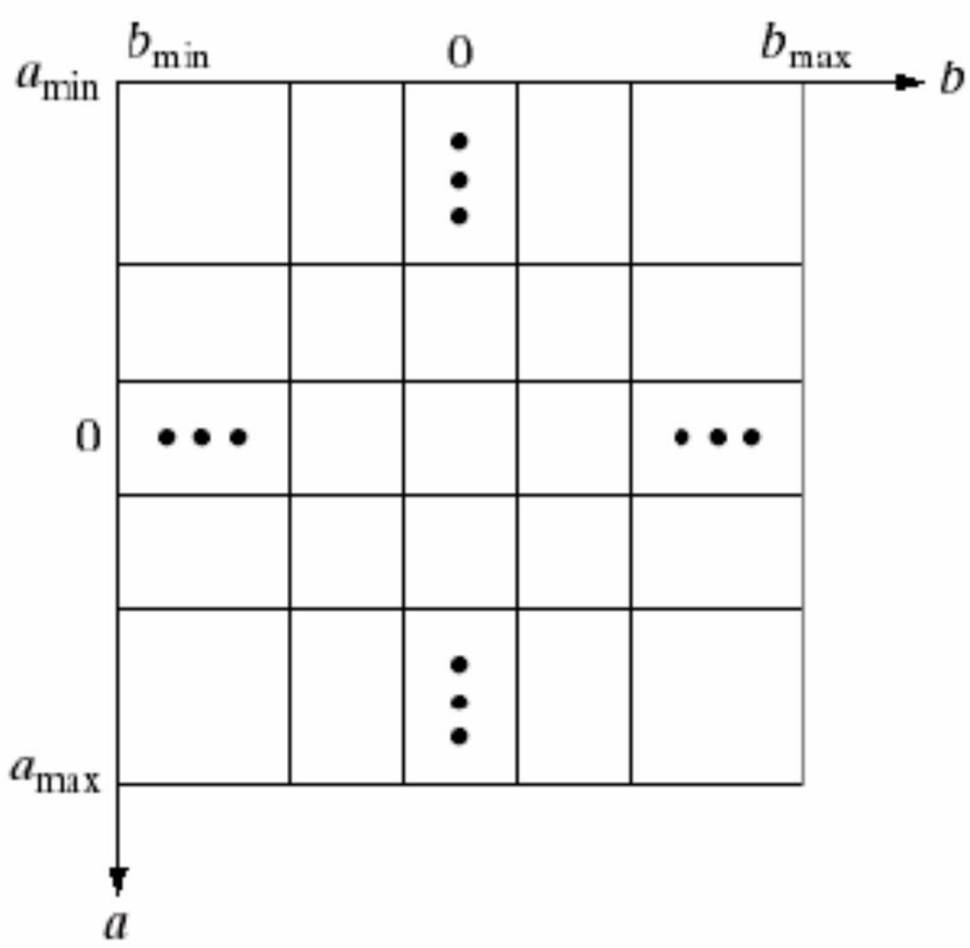
Hough Transform

- Consider a straight line. The HT maps all the points on the line onto a **single point** in the parameter space.
- A line in image space and a point in parameter space are a Hough Transform pair.
- A point in image space and a line in parameter space are a Hough Transform pair.

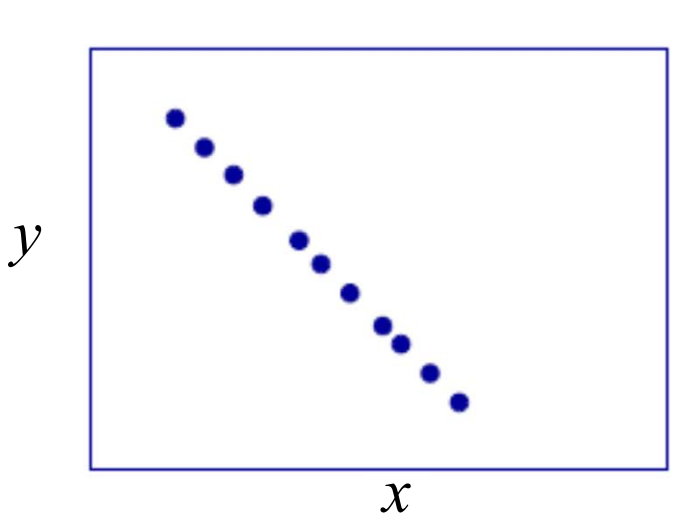
Hough Transform

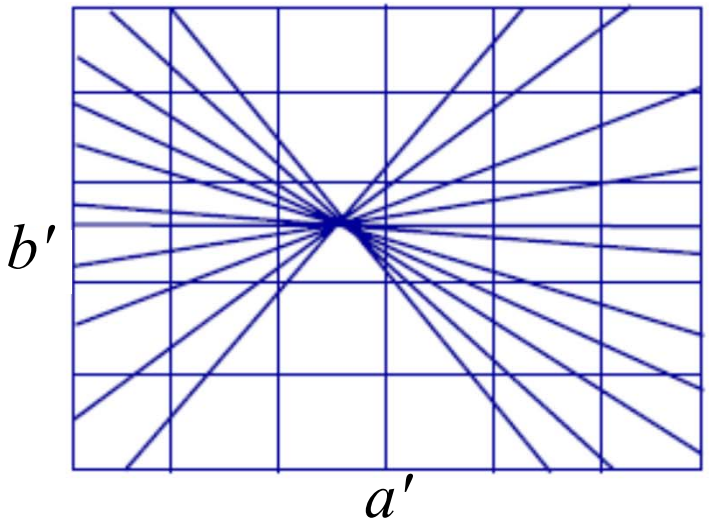
Steps of Hough transform:

1. In parameter space, quantize a and b, and give out (a_{min}, a_{max}) and (b_{min}, b_{max}).
2. Set an accumulator A: A is $A(a_{min}:a_{max}, b_{min}:b_{max})$, set A to zero at the beginning.
3. For a given point (x_i, y_i) in the original space, let a equal each allowed value on the a-axis, and solve for the corresponding b using the equation: $b = -x_i a + y_i$
4. The resulting b is then rounded off to the nearest allowed value in the b-axis. If a choice of a_p results in a solution b_q, we let $A(p, q) = A(p, q) + 1$.
5. At the end of this procedure, a value Q in $A(i, j)$ corresponds to Q points in the xy space lying on the line $y=a_i x+b_j$.



Hough Transform





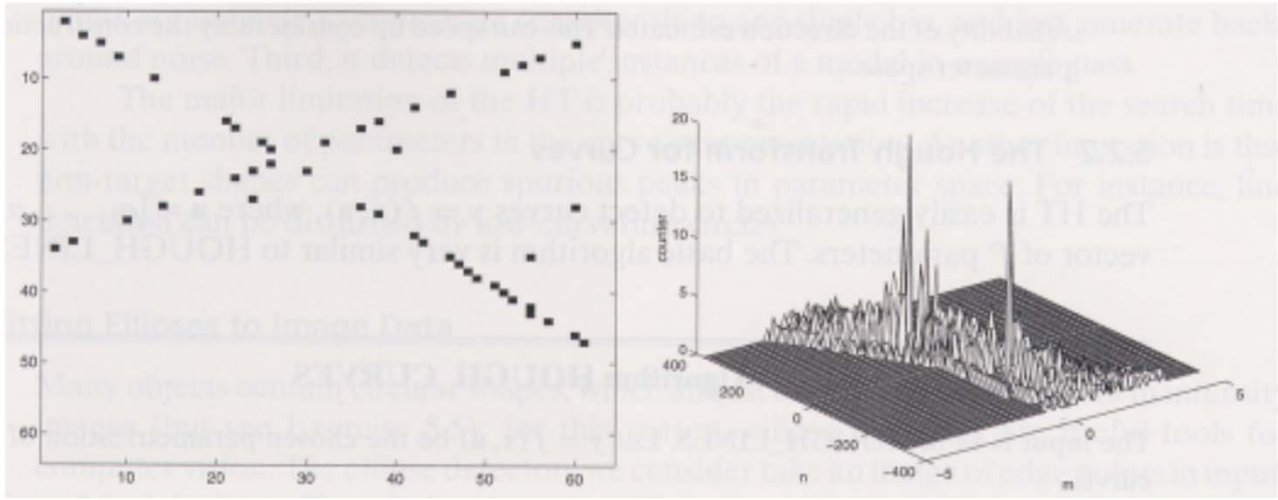
b′

2	2	0	1	2	2
3	5	3	3	2	2
3	7	11	10	4	3
2	3	1	4	5	2
2	1	0	1	3	3

a′

Hough Transform

- Quantize the parameter space (an array) and let the generation of a line be the increment of a counter at all cells in the parameter space that lie along the line.
- Histogram the resulting parameter space (accumulators).
- Peaks in this accumulator array occur at parameter locations corresponding to lines with significant support.



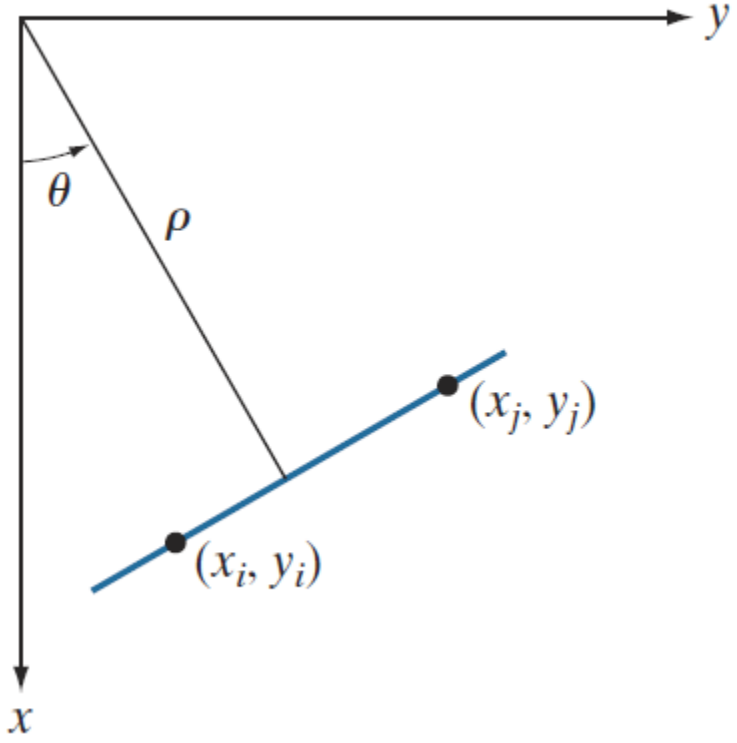
Hough Transform

- *a* can be huge for near vertical lines, and there is no representation for a vertical line.
- Solution – using polar representation for lines.

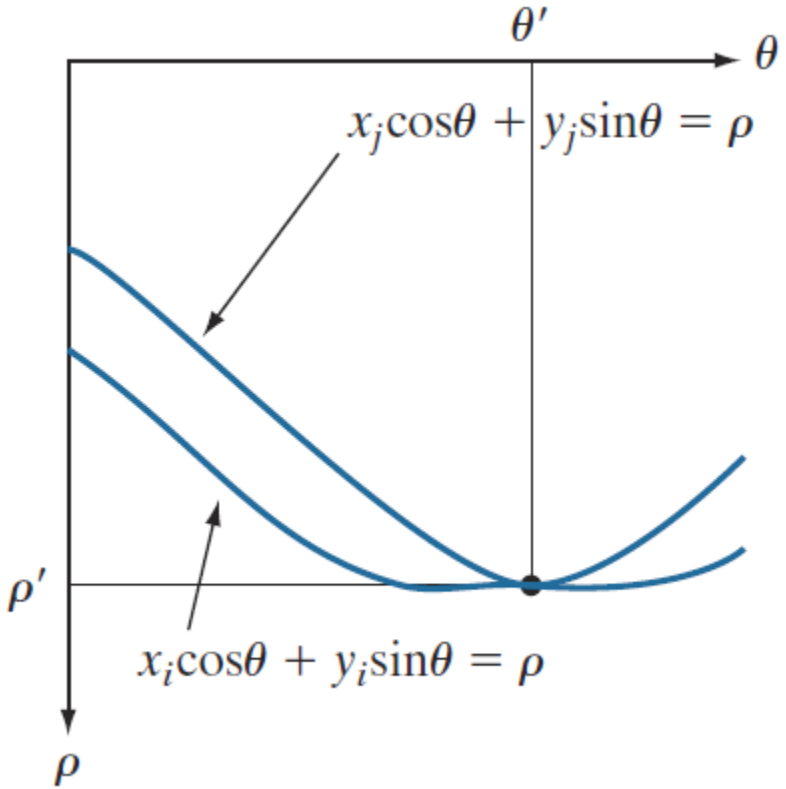
$$x\cos\theta + y\sin\theta = \rho$$

- In polar representation, a point in the image correspond to sinusoidal curve in the parameter space.

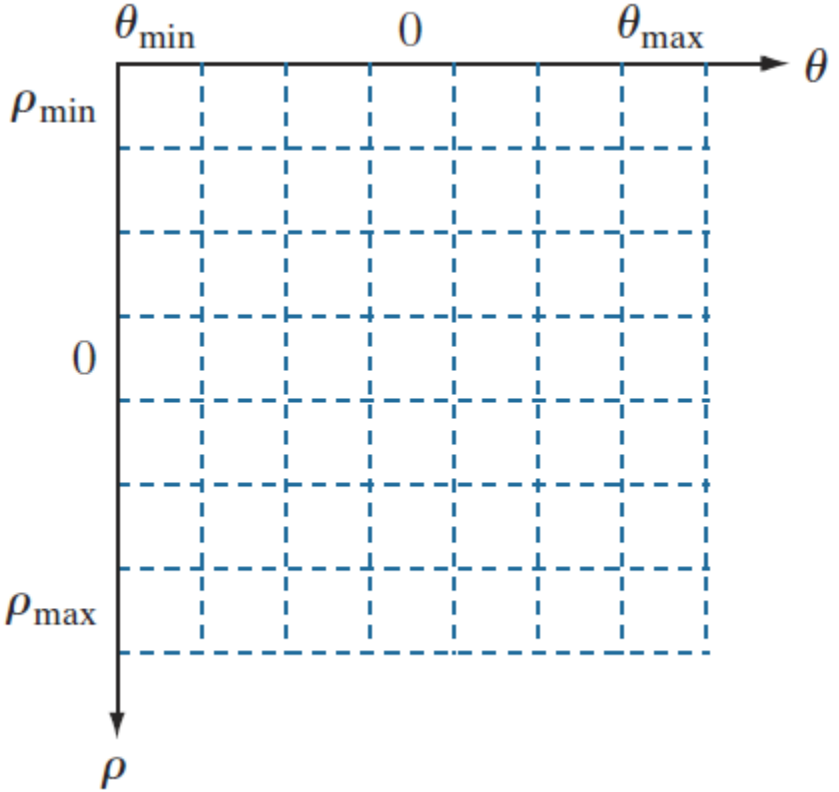
Hough Transform



(a) (ρ, θ) parameterization of a line in the xy-plane.



(b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane.

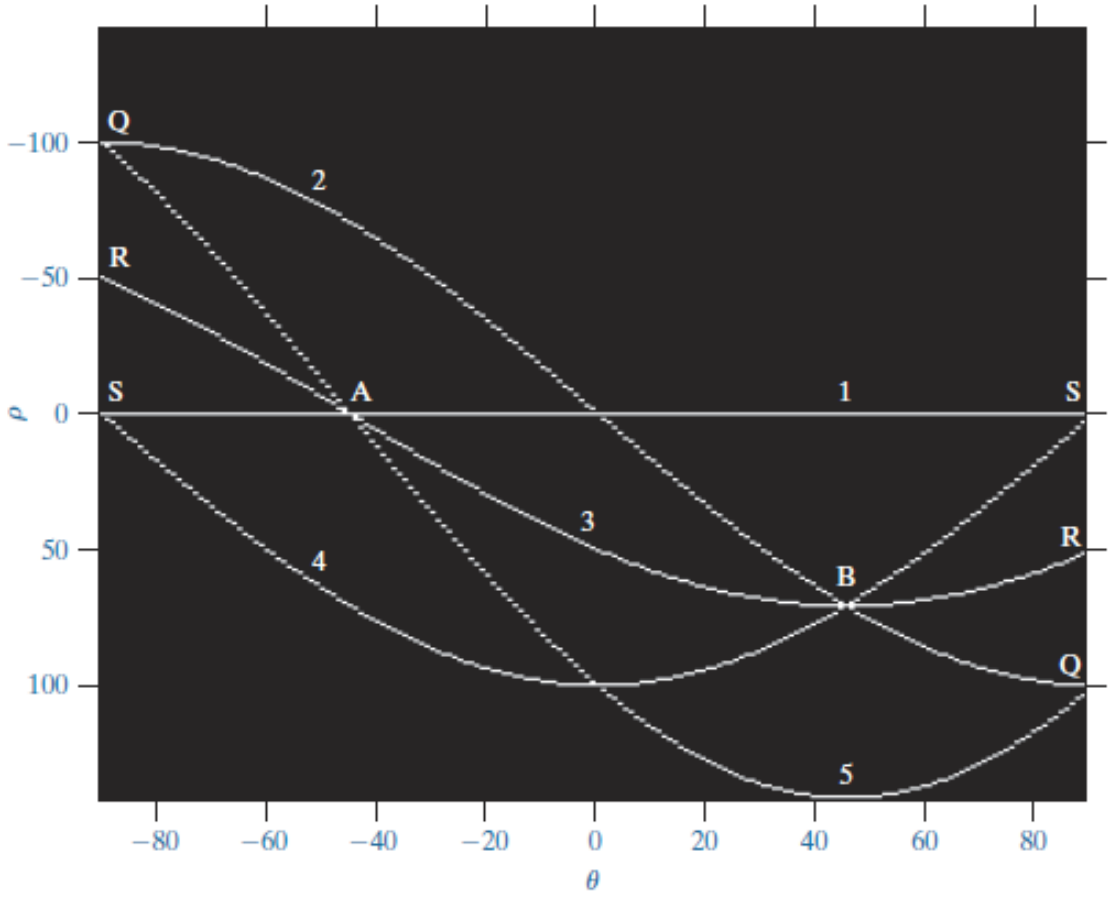


(c) Division of the $\rho\theta$-plane into accumulator cells.

Hough Transform

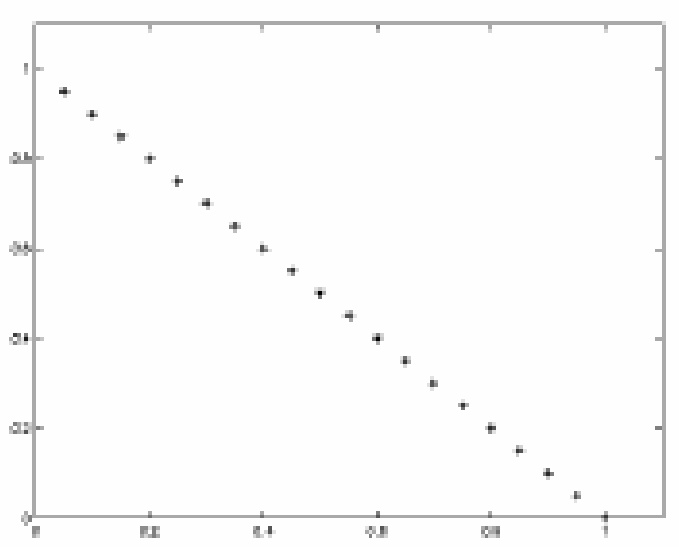
Image of size 101 × 101 pixels, containing five white points (four in the corners and one in the center).

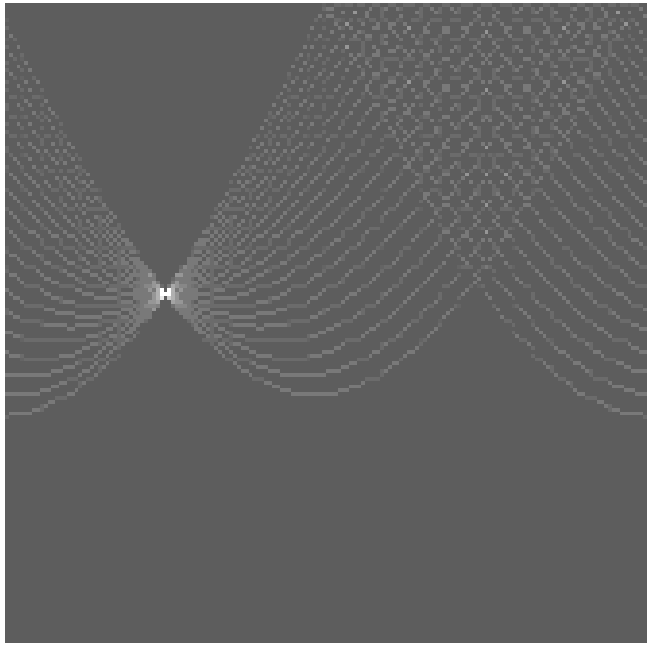
Corresponding parameter space.



Hough Transform

Hough Transform: 20 colinear points

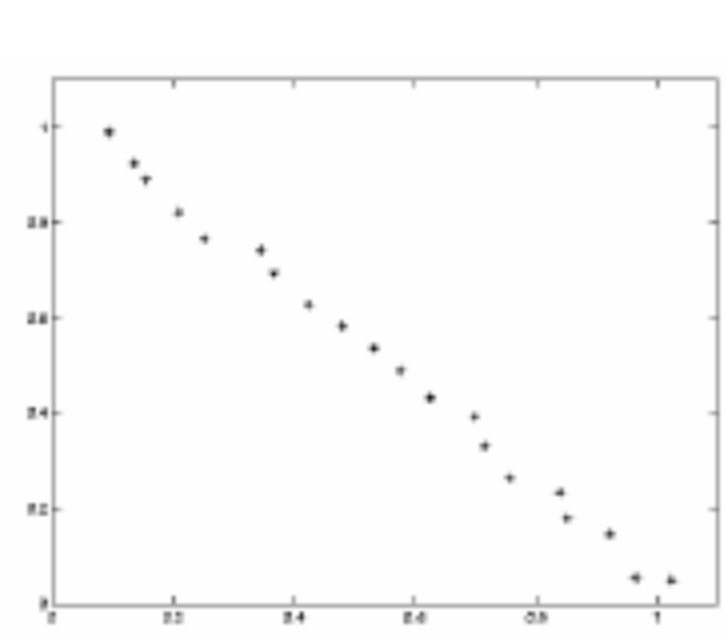


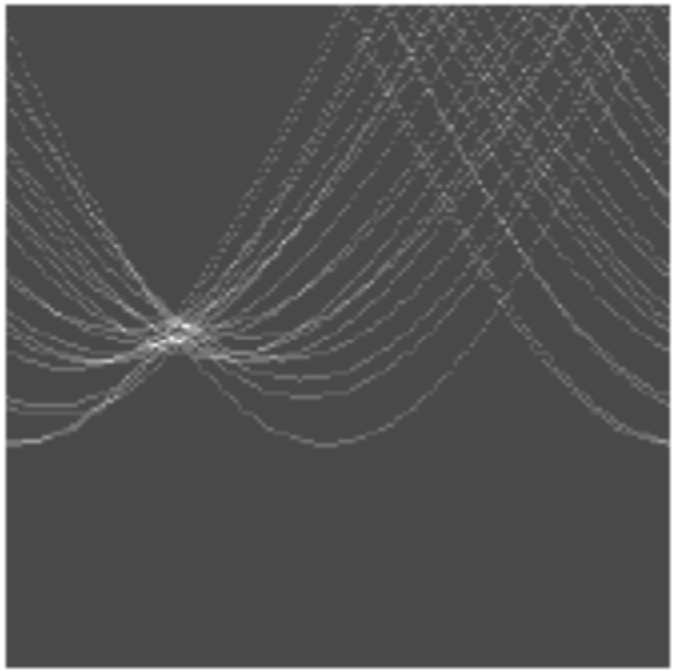


- ρ, θ representation of the line
- Maximum accumulator value is 20

Hough Transform

Hough Transform: “Noisy line”





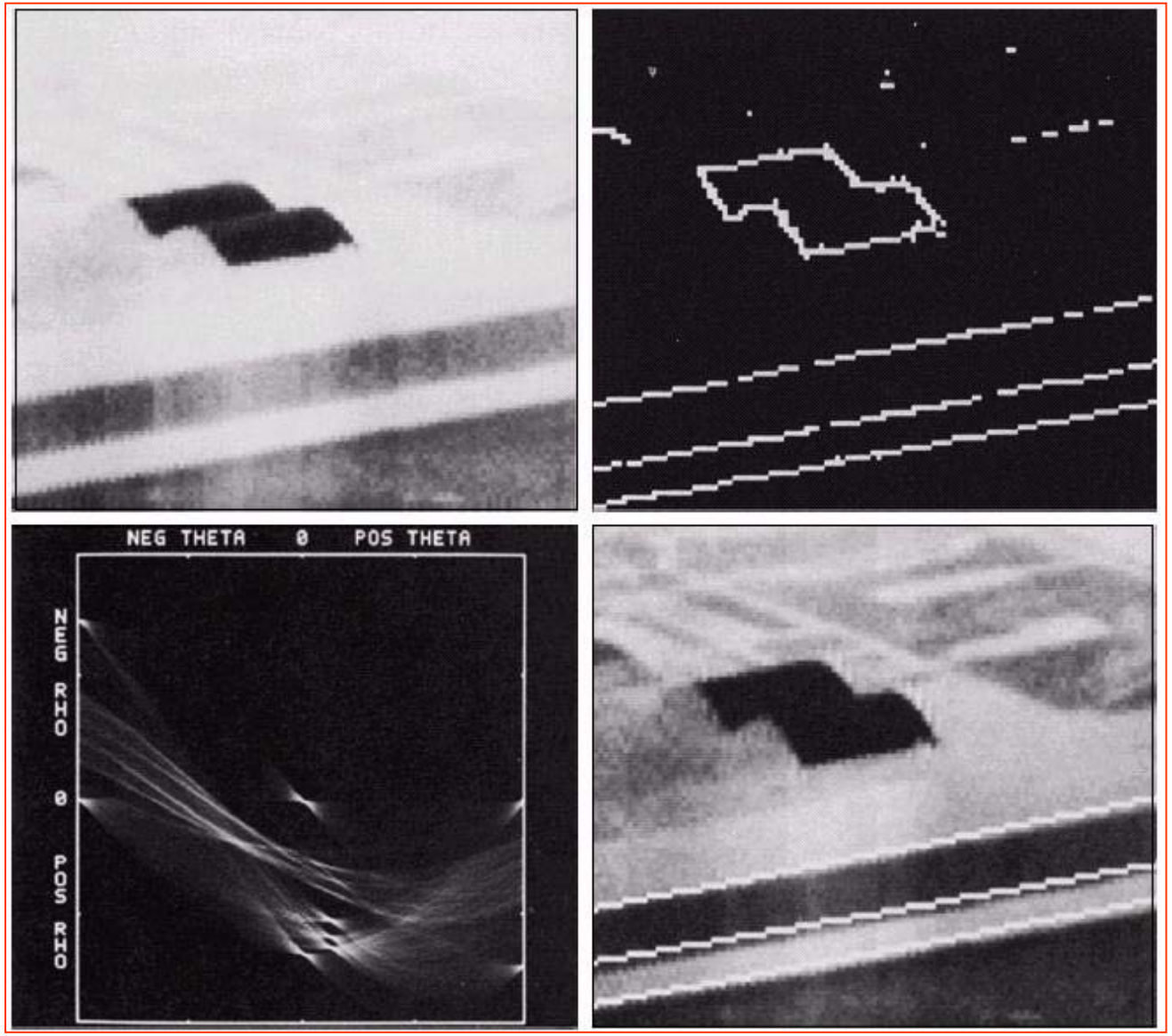
- ρ, θ representation of the line
- Maximum accumulator value is 6

Hough Transform

- Now return to the edge linking problem:

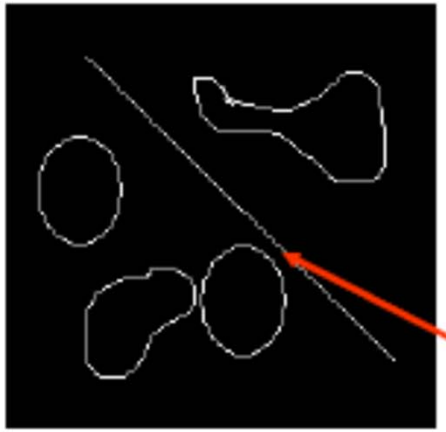
1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the parameter plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

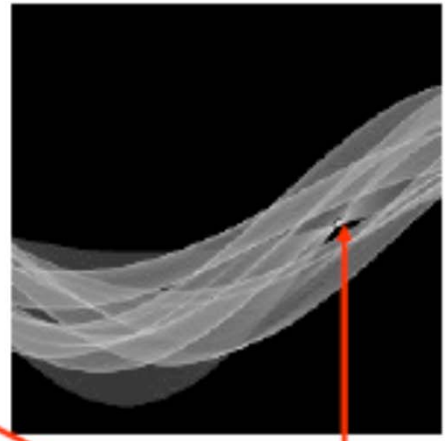
Hough Transform



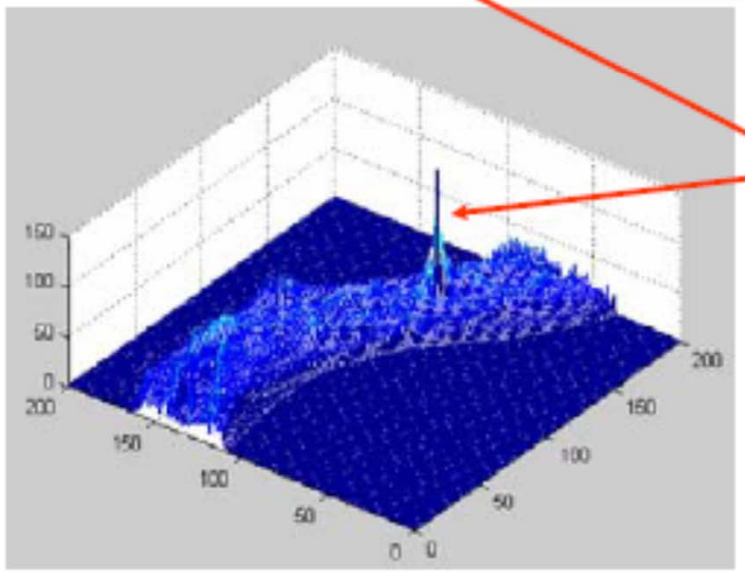
1. They belonged to one of the three accumulator cells with the highest count
2. No gaps were longer than five pixels

Hough Transform





Note: the parametric solution represents the whole line and not the segment.

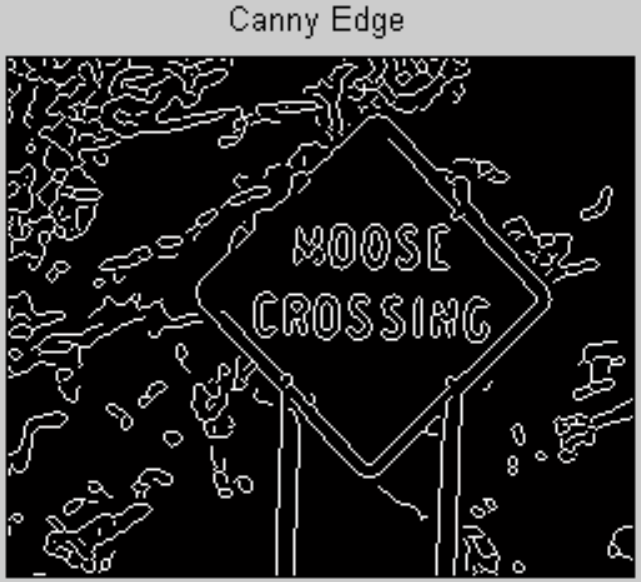


Peak corresponding to this line using normal parameters

It is easy to determine the segment points.

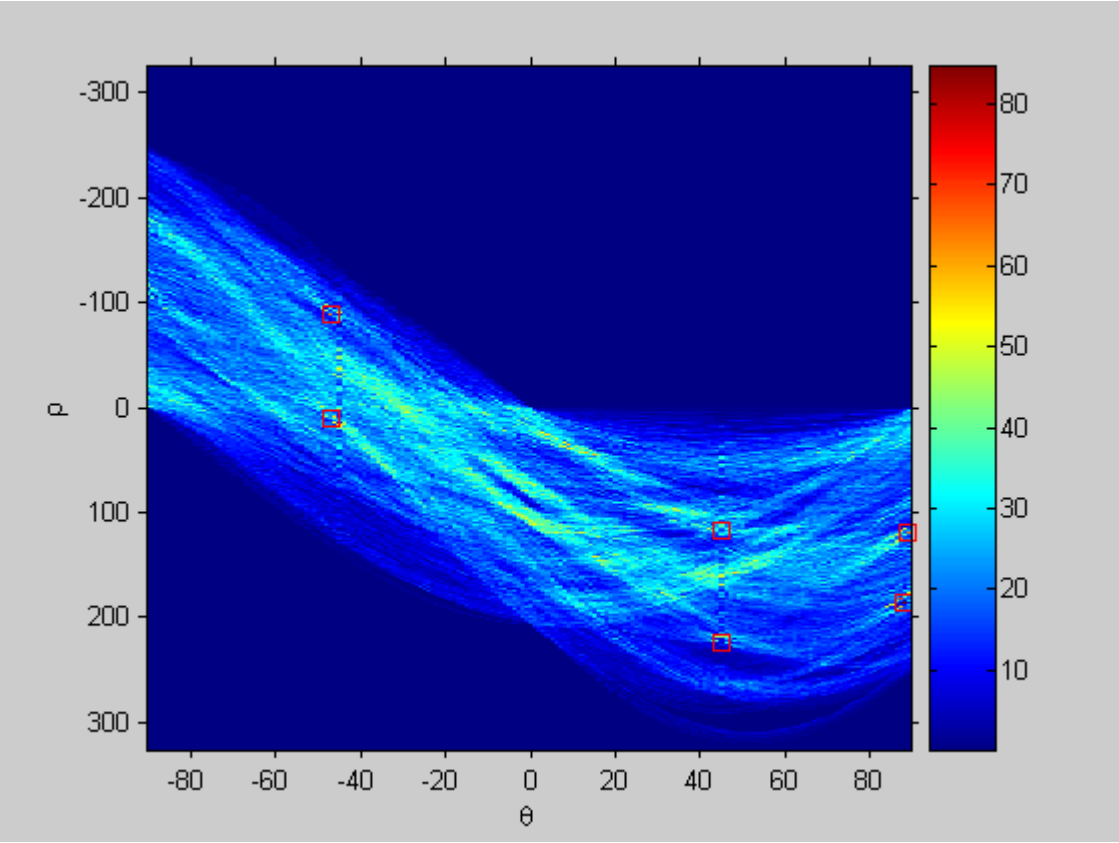
Hough Transform

Gray Scale Image



Edge Image (Canny)

Hough Transform



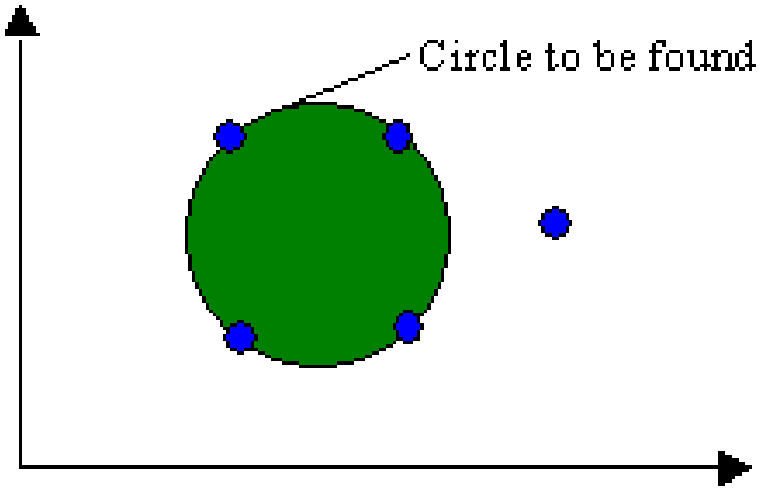
Hough Transform for Detecting Circles

- Although the focus so far has been on straight lines, the Hough transform is applicable to any function of the form $g(v, c)=0$, where v is a vector of coordinates and c is a vector of coefficients. For example, the points lying on the circle

$$(x-c_1)^2+(y-c_2)^2=c_3^2$$

  can be detected by using the approach just discussed.

- The only difference is that there are three parameters.



Hough Transform for Detecting Circles

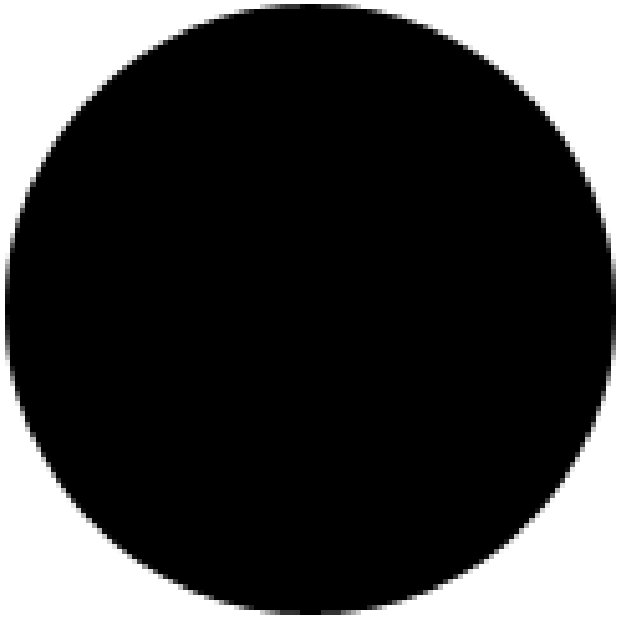
- The procedure is to increment c_1 and c_2, solve for the c_3 that satisfy:

$$(x-c_1)^2+(y-c_2)^2=c_3^2$$

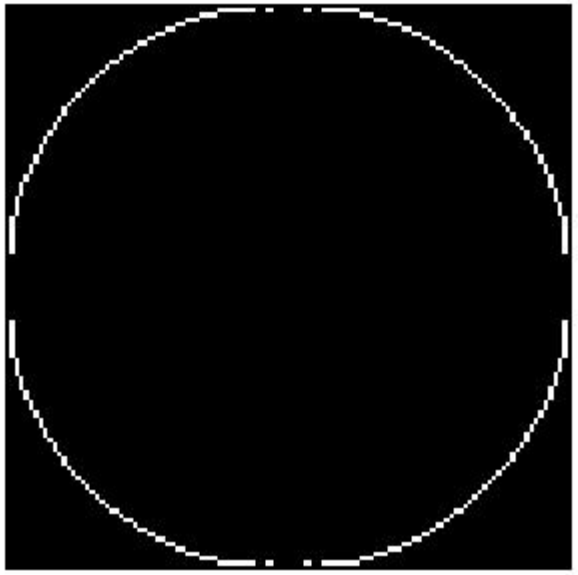
- Then updated the accumulator corresponding to the cell associated with the triplet (c_1,c_2,c_3).

- Clearly, the complexity of the Hough transform is proportional to the number of coordinates and coefficients in a given functional representation.

Hough Transform for Detecting Circles



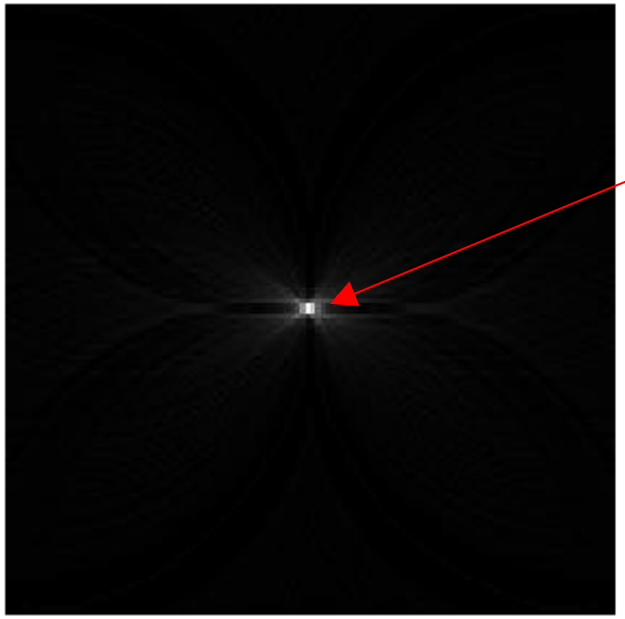
Input image



Detected circles

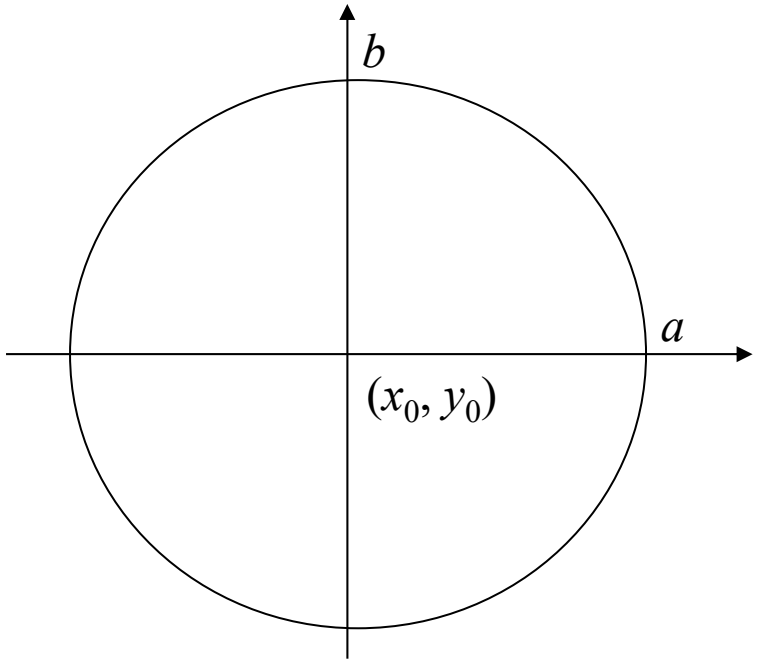
Hough Transform for Detecting Circles

- Find the center of the circle (the biggest value in the accumulator), since the radius is known, then we can find the circle we want.

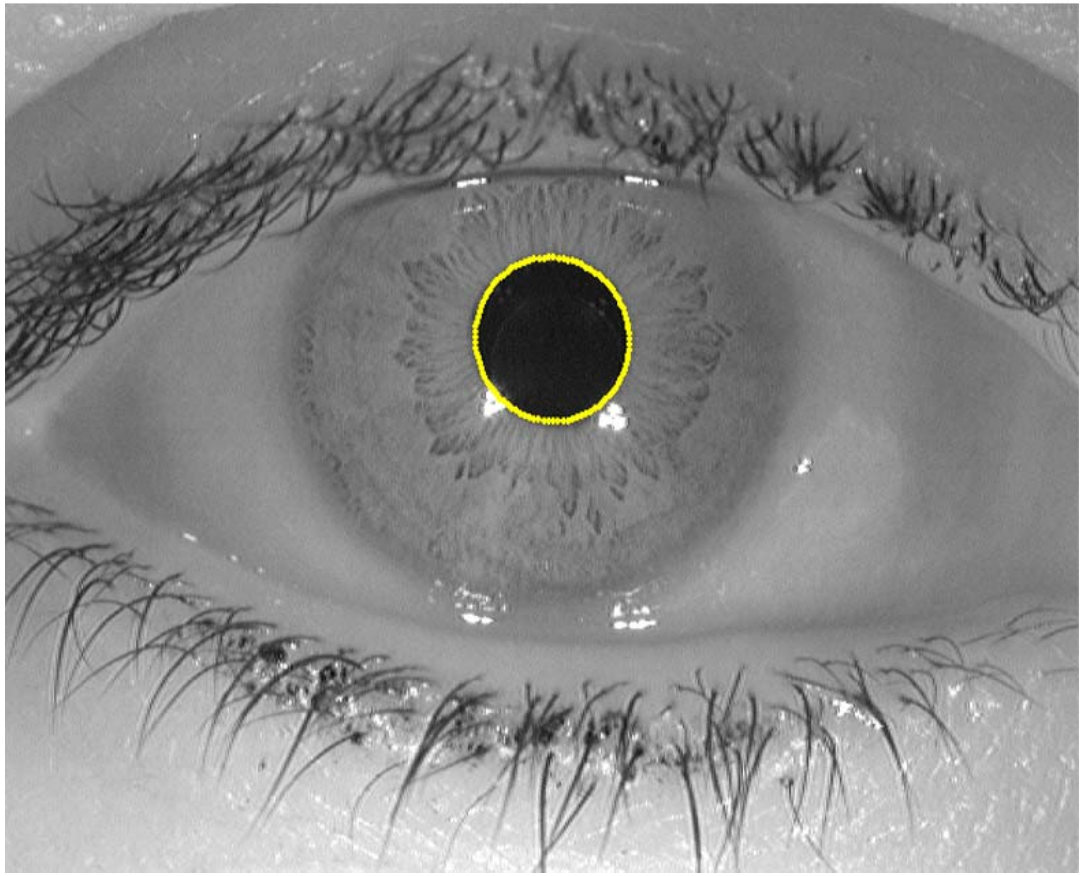


The center of the circle (the biggest value in the accumulator)

Hough Transform for Detecting Circles



$$\text{Min } E = (x - x_0)^2/a^2 + (y - y_0)^2/b^2 - 1$$



Hough Transform

1) The original Hough transform was designed to detect straight lines and curves.
2) This method can be used if analytic equations of object borderlines are known.
3) Since each point is handled independently, parallel implementations are possible.
4) It becomes difficult when the dimension of the parameter space is large.
5) Now hardware of Hough transform for line detection is available.

Summary

- In this lecture we have learnt:
  - The segmentation problem
  - Edges detection
  - Edge linking and boundary detection

Optional Homework

Check the Textbook!

- **Chapter 10: Problems 10.3, 10.6(a), 10.9, 10.11(b), 10.26(a)(b)**

- Homework answers will be provided at the end of each week.