Image Segmentation-II

Point, Line, and Edge Detection

Contents

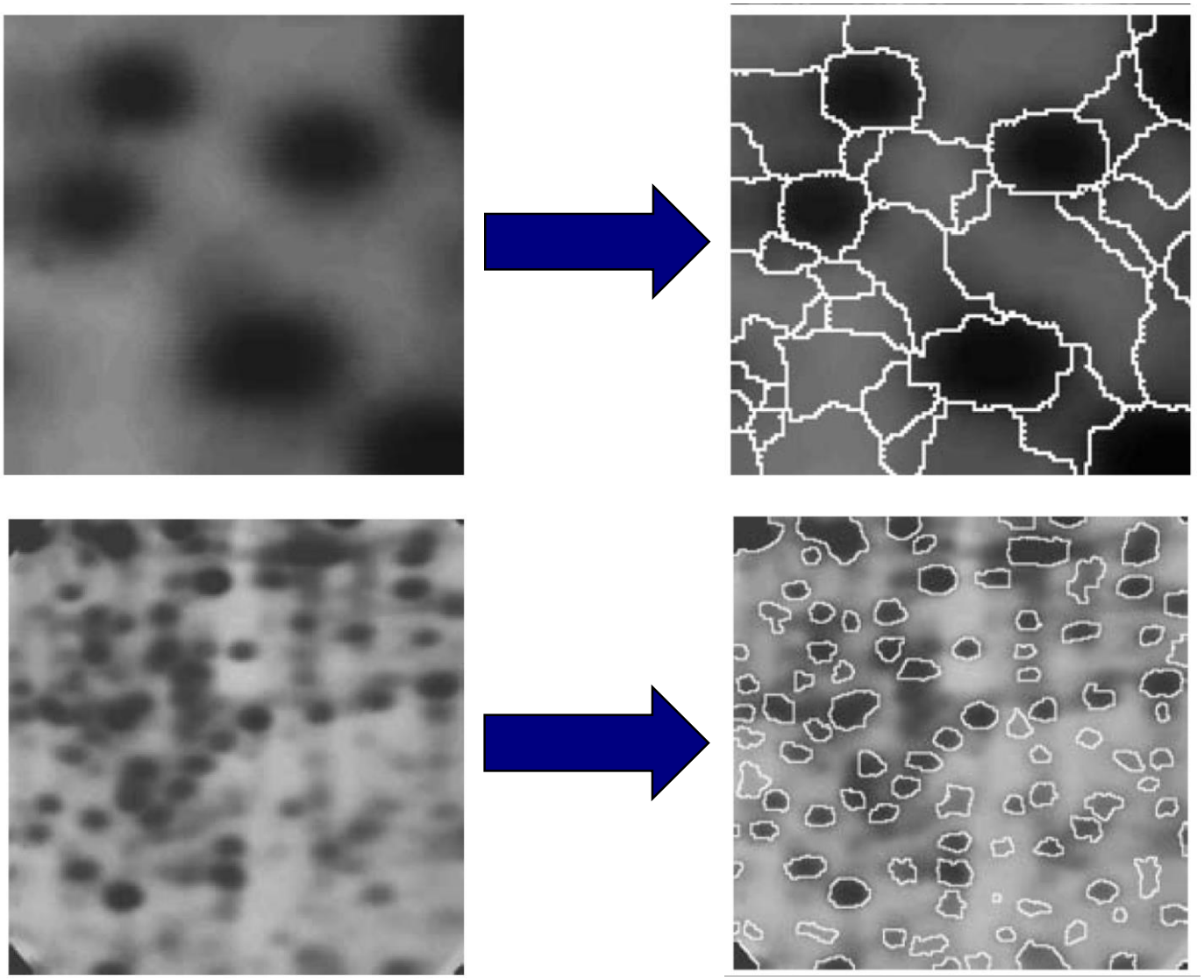
- ✓ The Segmentation Problem
- ✓ Finding Points and Lines
- ✓ Edge Detection
  - Derivative Operators
  - Hough Transform

Introduction

Edge detection is massively important as it is in many cases the first step to object recognition.

- The segmentation problem
- Finding points, lines, and edges

Segmentation Examples



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

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.

Point Detection

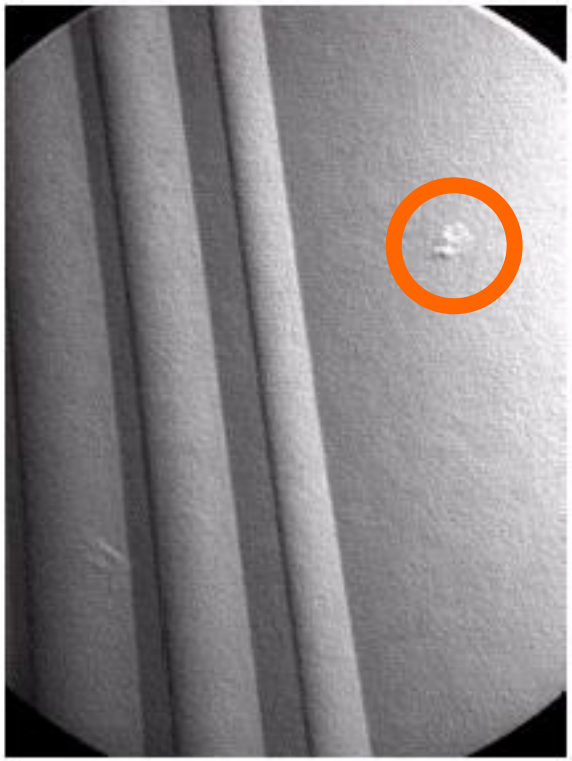
Point Detection

Point detection can be achieved simply using the mask below:

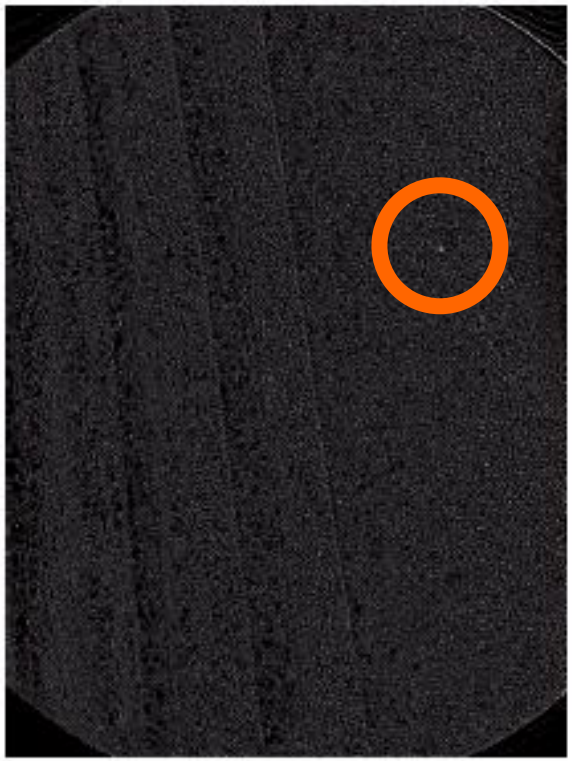
−1	−1	−1
−1	8	−1
−1	−1	−1

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

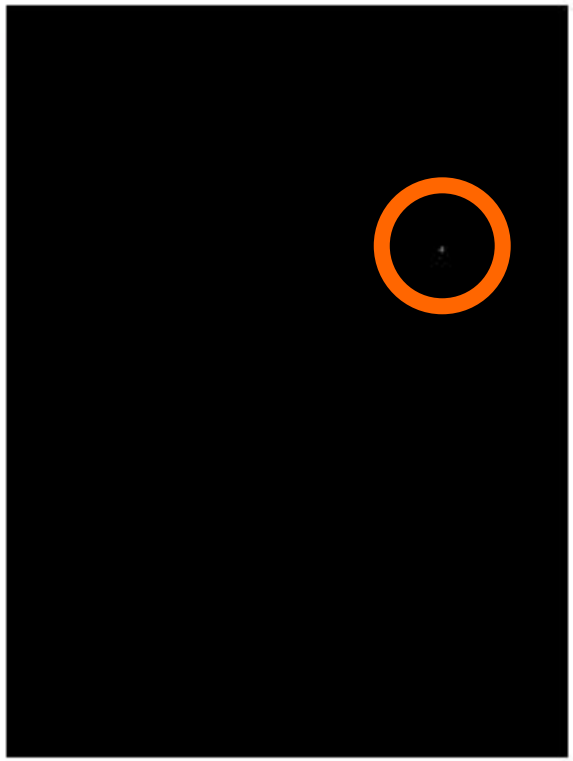
Point Detection



X-ray image of
a turbine blade



Result of point
detection



Result of
thresholding

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Line Detection

Line Detection

The next level of complexity is to try to detect lines.

The masks below will extract lines that are one pixel thick and running in a particular direction.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

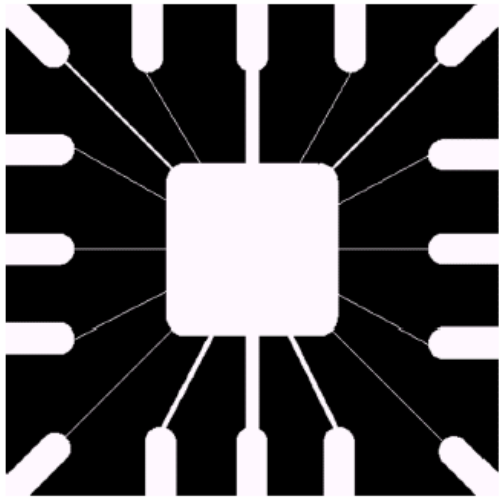
Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

Line Detection

Binary image of a wire bond mask



After processing with -45° line detector

Result of thresholding filtering result

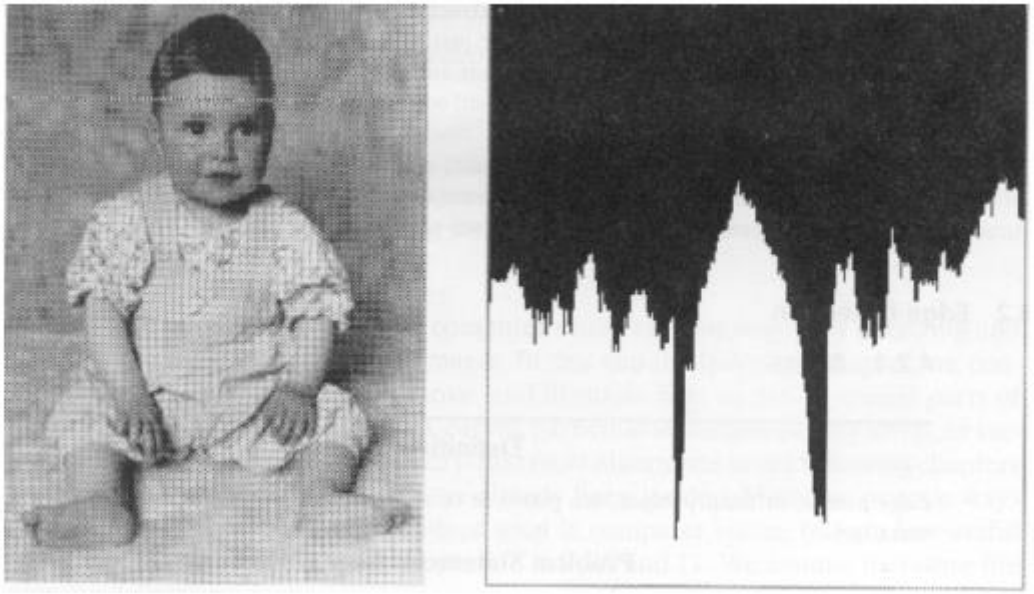
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection

Definition of Edges

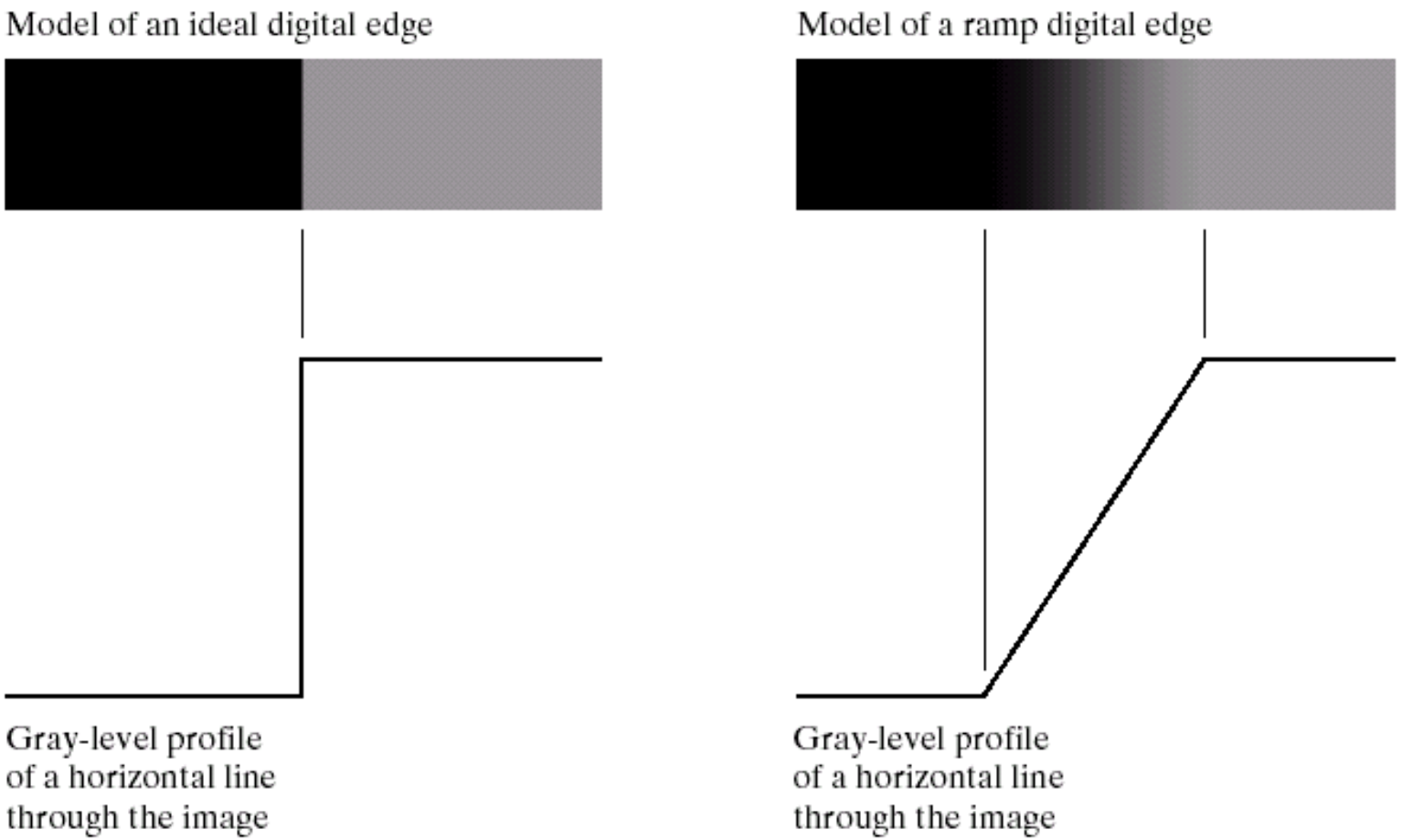
Edges are significant local changes of intensity in an image.

Edges typically occur on the boundary between two different regions in an image.



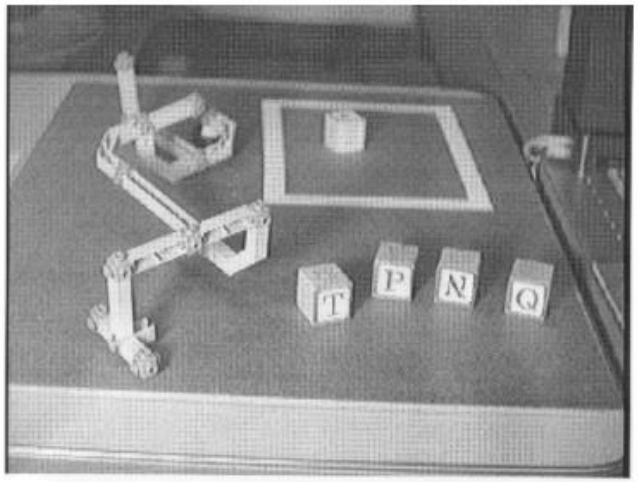
Definition of Edges

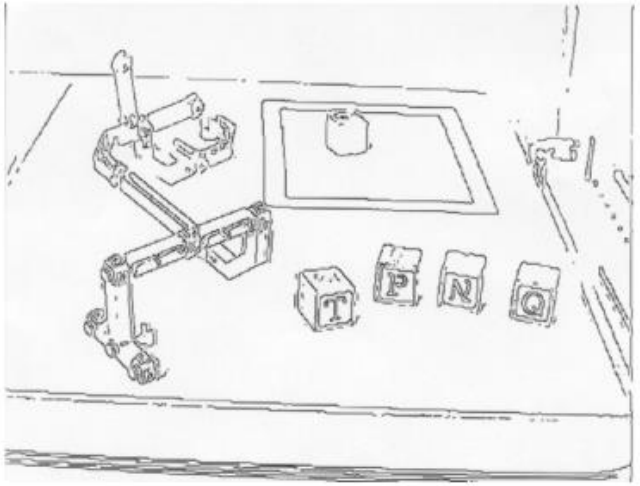
An edge is a set of connected pixels that lie on the boundary between two regions.



Goal of Edge Detection

- Produce a line drawing of a scene from an image of that scene.
- Important features can be extracted from the edges of an image (e.g., corners, lines, curves).
- These features are used by higher-level computer vision algorithms (e.g., recognition).



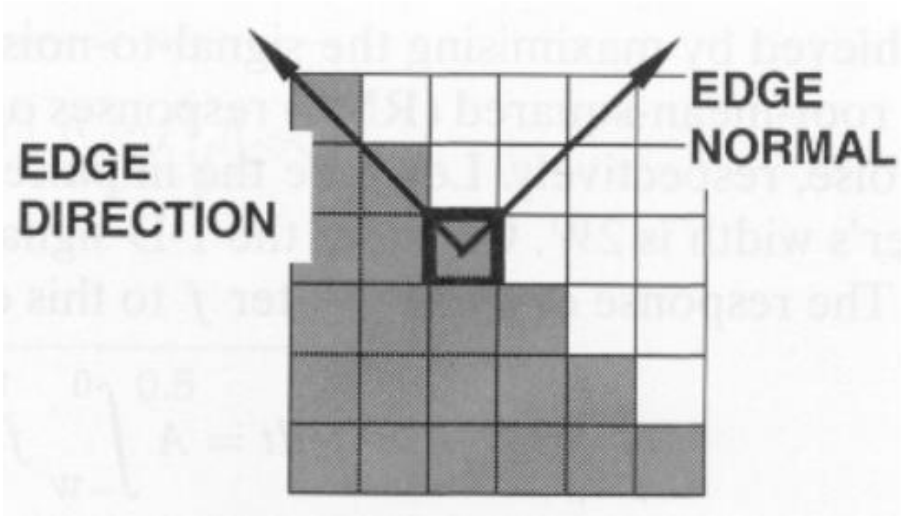


Reason of Intensity Changes

- Various physical events cause intensity changes.
- Geometric events:
  - Object boundary (discontinuity in depth and/or surface color and texture)
  - Surface boundary (discontinuity in surface orientation and/or surface color and texture)
- Non-geometric events:
  - Specularity (direct reflection of light, such as a mirror)
  - Shadows (from other objects or from the same object)
  - Inter-reflections

Edge Descriptors

- Edge normal: unit vector in the direction of maximum intensity change.
- Edge direction: unit vector perpendicular to the edge normal.
- Edge position or center: the image position at which the edge is located.
- Edge strength: related to the local image contrast along the normal.



Types of Edges

- **Step edge:** the image intensity abruptly changes from one value to one side of the discontinuity to a different value on the opposite side.

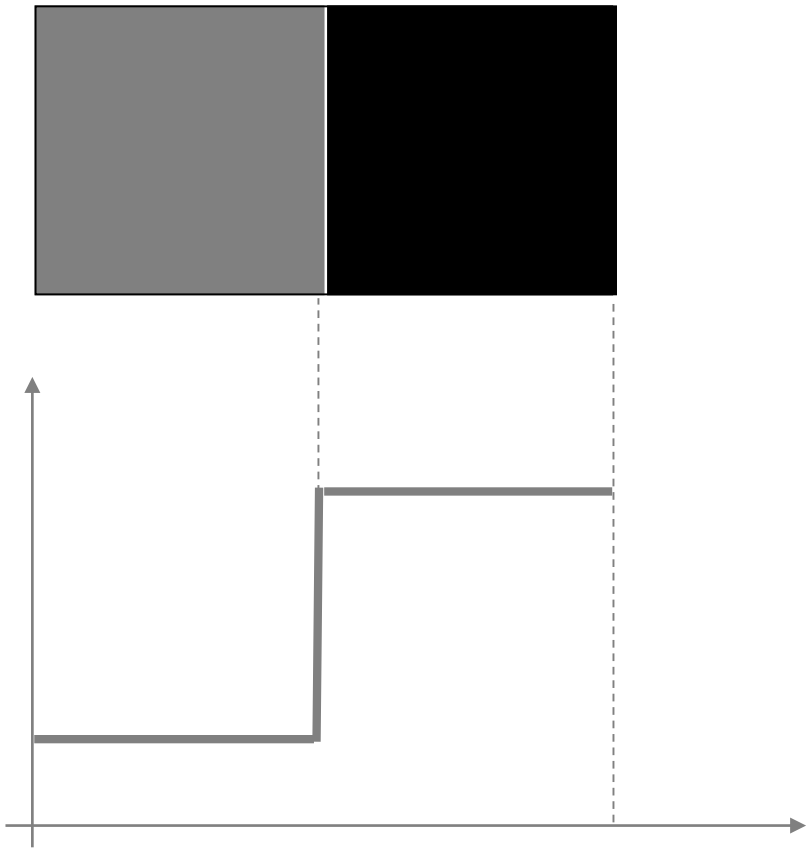
- **Ramp edge:** a step edge where the intensity change is not instantaneous but occur over a finite distance.

- **Ridge edge:** the image intensity abruptly changes value but then returns to the starting value within some short distance (generated usually by lines).

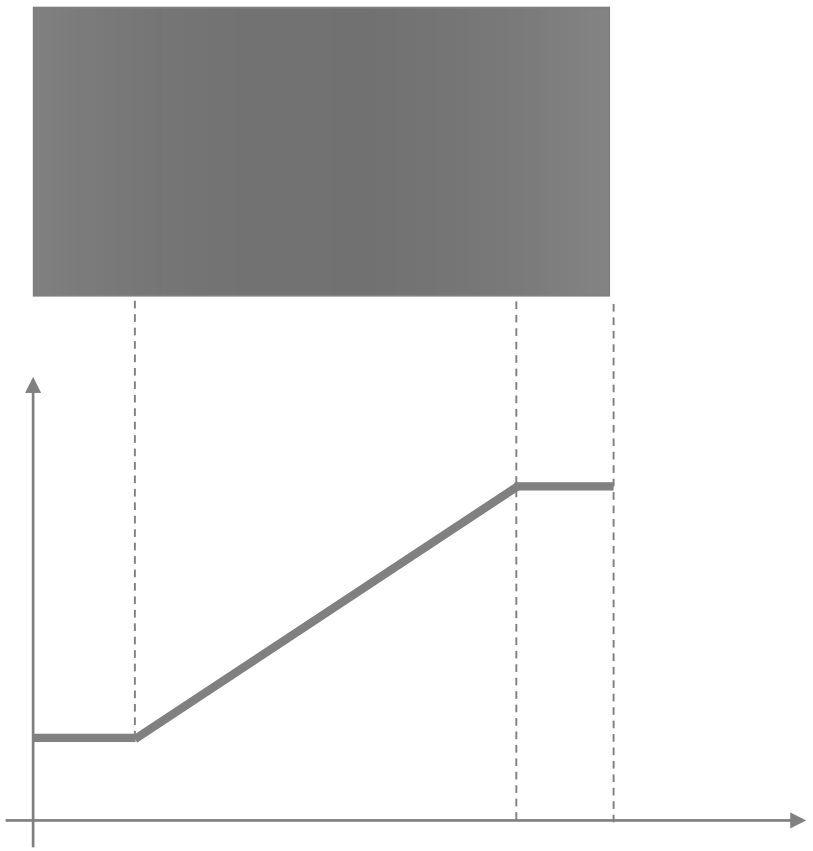
- **Roof edge:** a ridge edge where the intensity change is not instantaneous but occur over a finite distance (generated usually by the intersection of surfaces).

Gray-Level Transition

Step

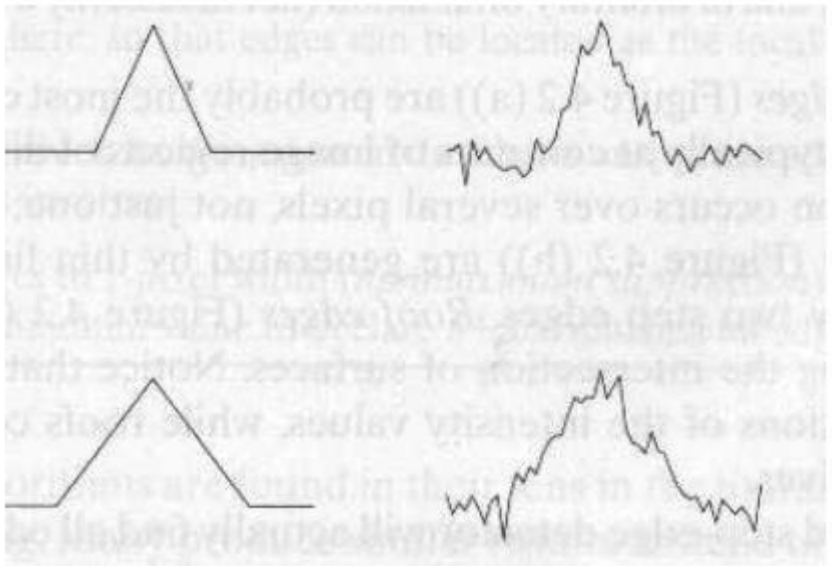


Ramp



Gray-Level Transition

Ridge Edge



Roof Edge

Four Steps of Edge Detection

(1) Smoothing: suppress as much noise as possible, without destroying the true edges.

(2) Enhancement: apply a filter to enhance the quality of the edges in the image (sharpening).

(3) Detection: determine which edge pixels should be discarded as noise and which should be retained (usually,thresholding provides the criterion used for detection).

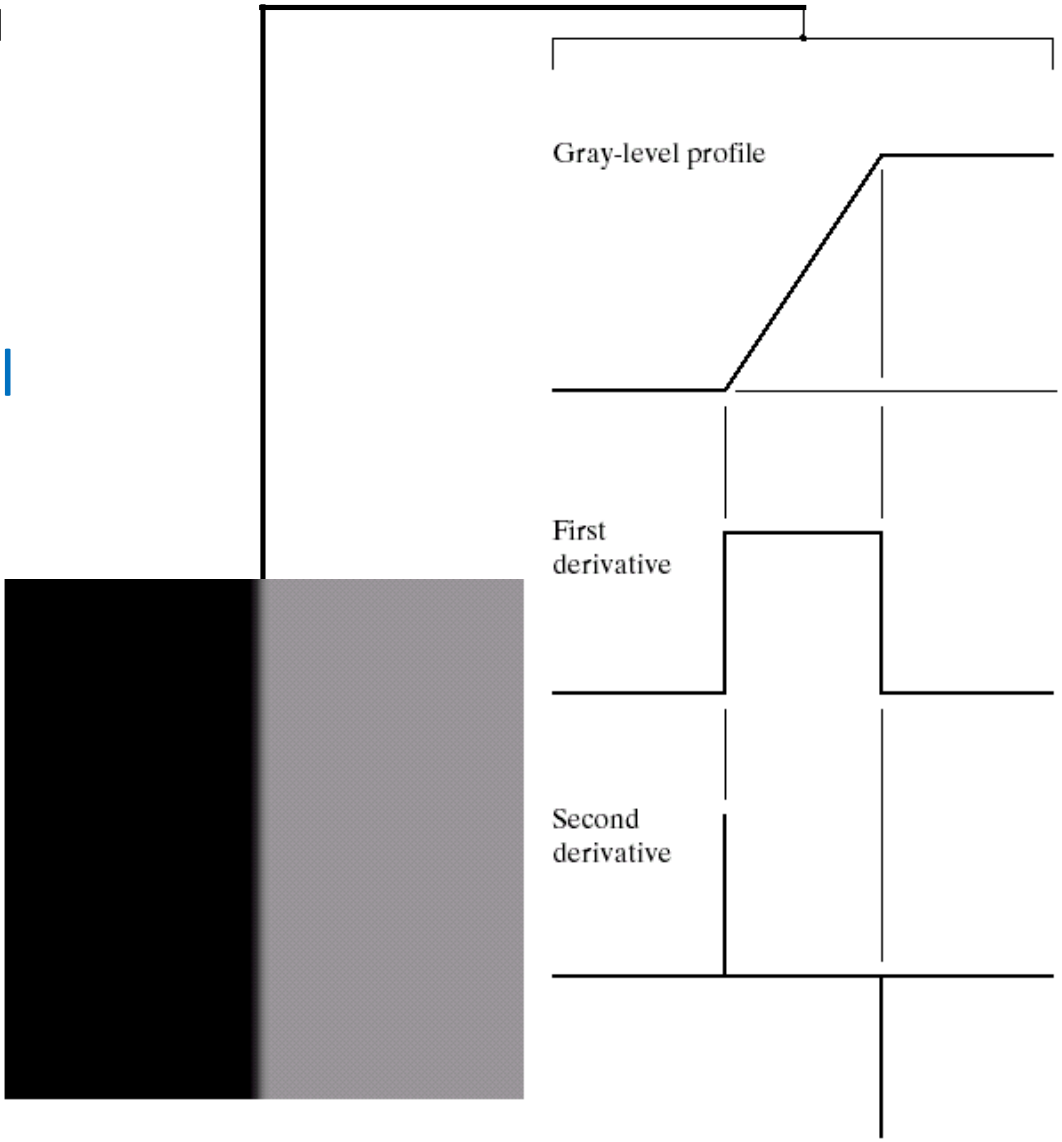
(4) Localization: determine the exact location of an edge (sub-pixel resolution might be required for some applications, that is, estimate the location of an edge to better than the spacing between pixels). Edge thinning and linking are usually required in this step.

Edges & Derivatives

It is already known to you about how derivatives are used to find discontinuities.

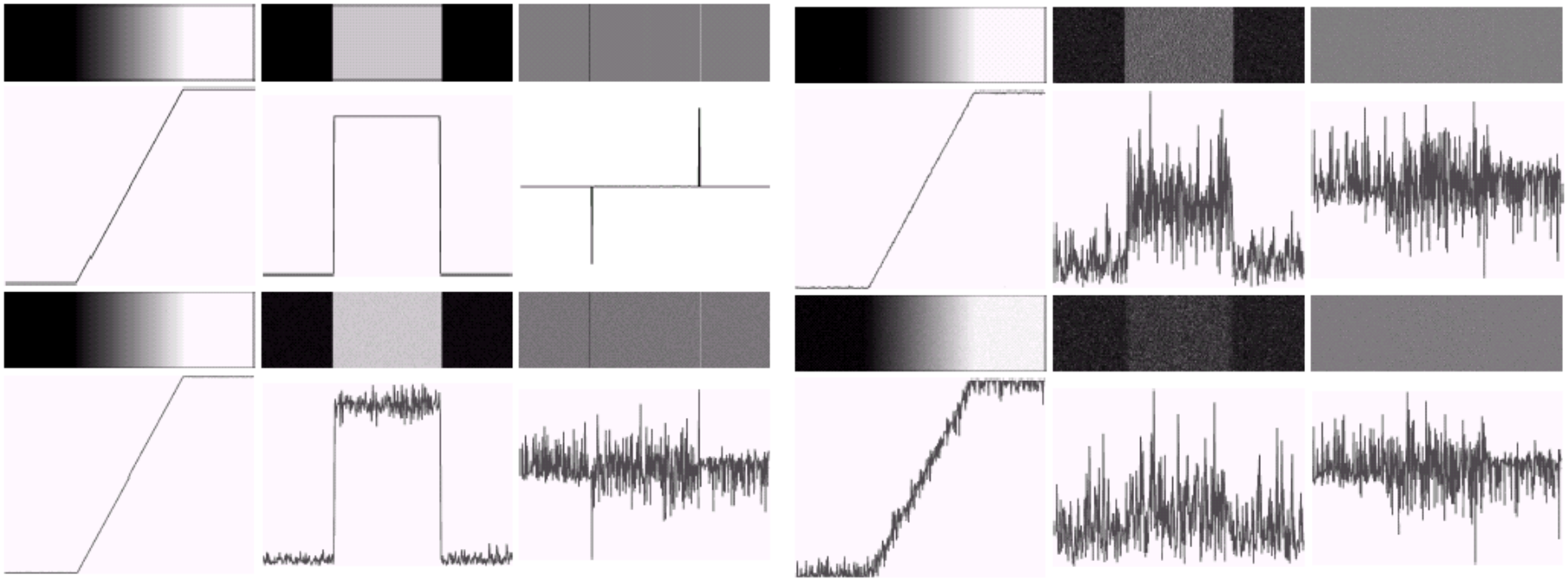
1st derivative detects local Maxima/minima.

2nd derivative Detects *zero crossing*.



Derivatives & Noise

Derivative based edge detectors are extremely sensitive to noise.



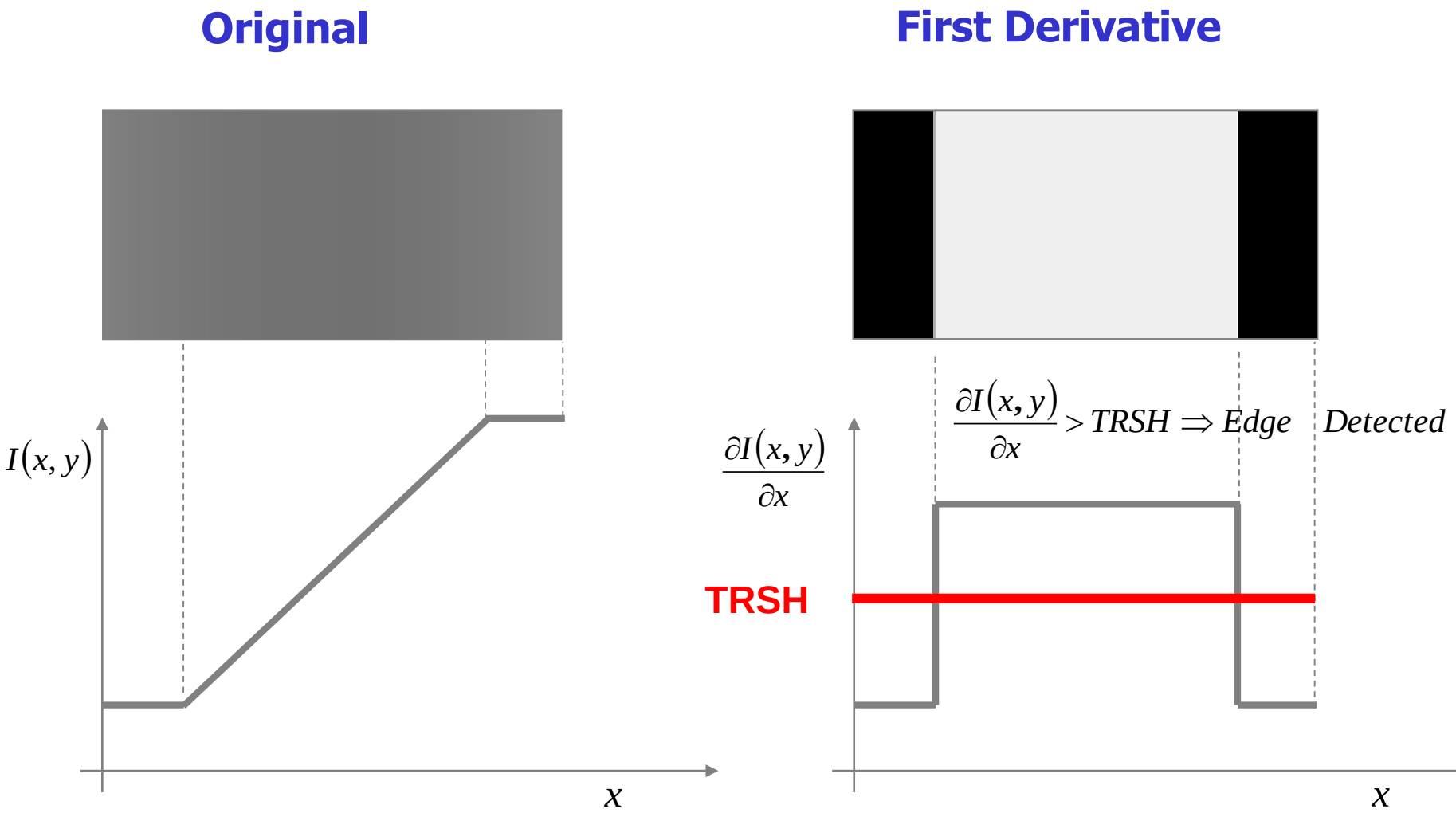
Criteria for Optimal Edge Detection

(1) Good Detection: the optimal detector must minimize the probability of false positives(detecting spurious edges caused by noise), as well as that of false negatives(missing real edges).

(2) Good Localization: the edges detected must be as close as possible to the true edges.

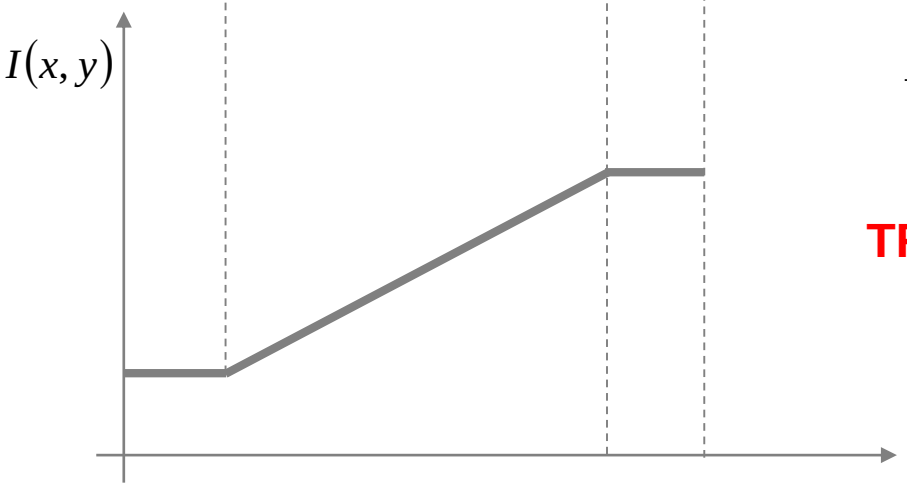
(3) Single Response Constraint: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge (created by noise).

Detecting the Edge (1)

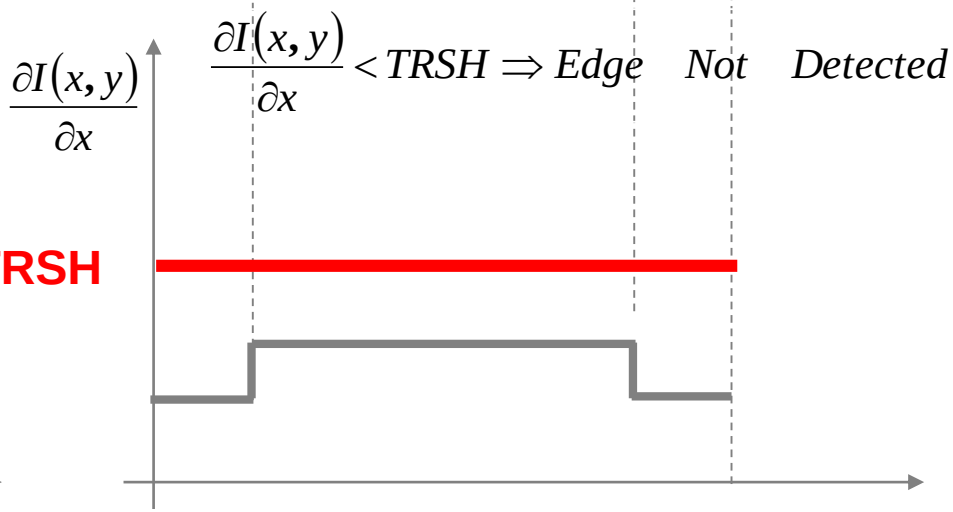


Detecting the Edge (2)

Original



First Derivative



Gradient Operators

The gradient of the image ***I(x,y)*** at location ***(x,y)***, is the vector:

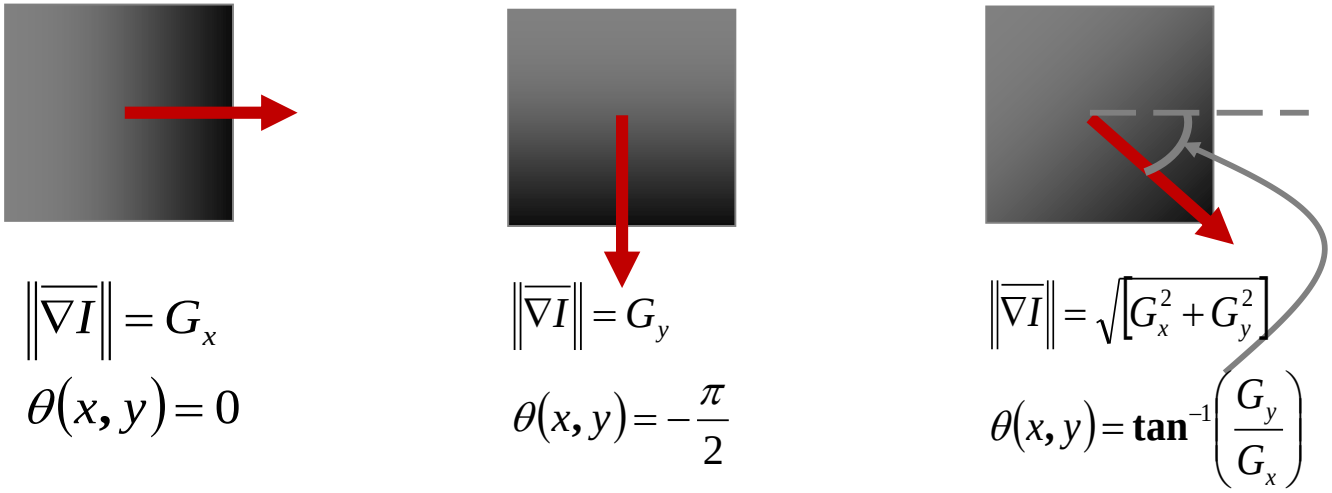
$$\overline{\nabla I} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

The magnitude of the gradient: $\nabla I = \left\| \overline{\nabla I} \right\| = \sqrt{\left[G_x^2 + G_y^2 \right]}$

The direction of the gradient vector: $\theta(x,y) = \tan^{-1}\left(\frac{G_y}{G_x} \right)$

The Meaning of the Gradient

It represents the direction of the strongest variation in intensity.



$$\left\|\overline{\nabla I}\right\| = G_x \qquad \theta(x,y) = 0$$

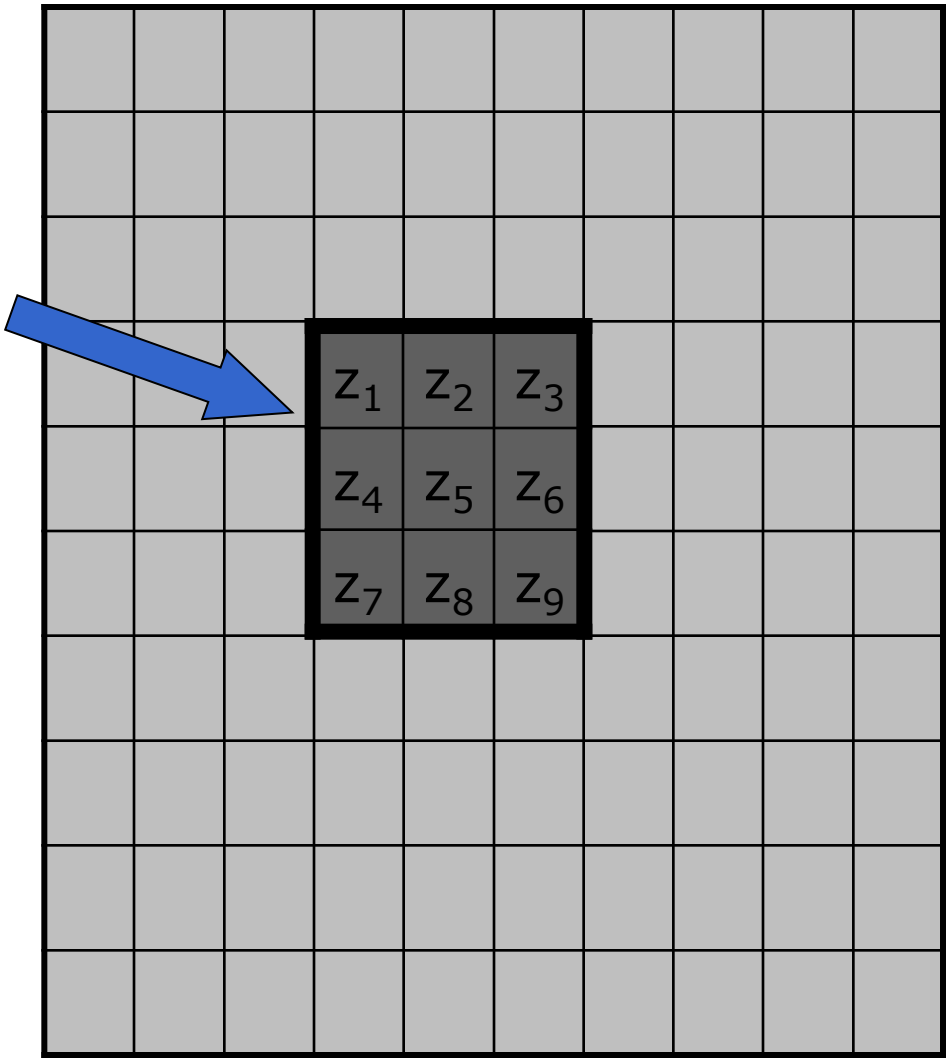
$$\left\|\overline{\nabla I}\right\| = G_y \qquad \theta(x,y) = -\frac{\pi}{2}$$

$$\left\|\overline{\nabla I}\right\| = \sqrt{\left[G_x^2 + G_y^2\right]} \qquad \theta(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

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

Calculating the Gradient

For each pixel the gradient is calculated, based on a 3x3 neighborhood around this pixel.



Ordinary Operator

0	-1	0
0	1	-1
0	0	1

$$G_x \approx z_5 - z_2$$

0	0	0
-1	1	-1
0	1	0

$$G_y \approx z_5 - z_4$$

As the difference is taken between adjacent pair of pixels, gradient computed by this operator is not symmetric about the central pixel.

Robert Operator [Robert'1965]

-1	0	0
0	1	-1
0	0	1

$$G_x \approx z_5 - z_1$$

0	1	0
-1	0	-1
0	1	0

$$G_y \approx z_2 - z_4$$

Gradient value obtained to the central pixel (r-1/2, c-1/2) and is symmetric about this pixel.

4-Neighbour Operator [Chaudhuri and Chanda'1984]

0	-1	0
0	0	0
0	1	0

$$G_x \approx z_8 - z_2$$

0	0	0
-1	0	1
0	0	0

$$G_y \approx z_6 - z_4$$

Prewitt Operator [Prewitt'1970]

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

$$G_x \approx (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$G_y \approx (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Sobel Operator [Duda and Hart'1973]

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

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

$$G_x \approx (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y \approx (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Performance of Different Operators

Prewitt operator can detect diagonal edges better than that by Sobel operator.

Sobel operator is superior than Prewitt in detecting vertical edges.

Robert operator is very much sensitive to noise. The effect of noise is reduced in case of Prewitt and Sobel operators.

Edge Detection Example

Original Image

Horizontal Gradient Component

Vertical Gradient Component

Combined Edge Image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Problems

Often, problems arise in edge detection when there is too much detail.

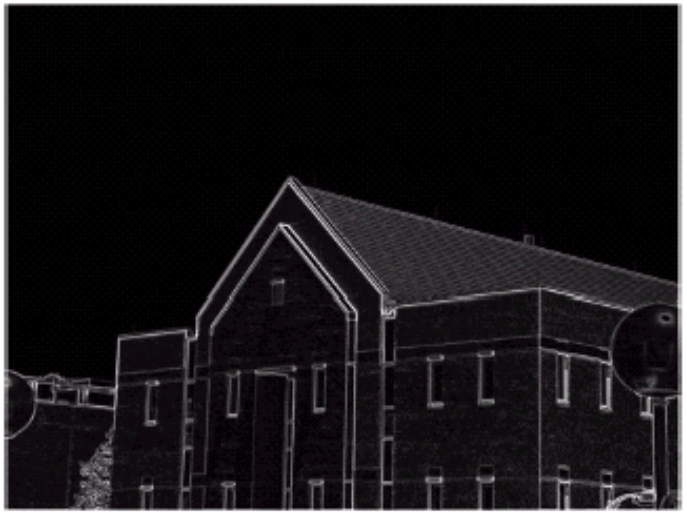
One way to overcome this is to smooth images prior to edge detection.

Edge Detection Example With Smoothing

Original Image

Horizontal Gradient Component

Vertical Gradient Component

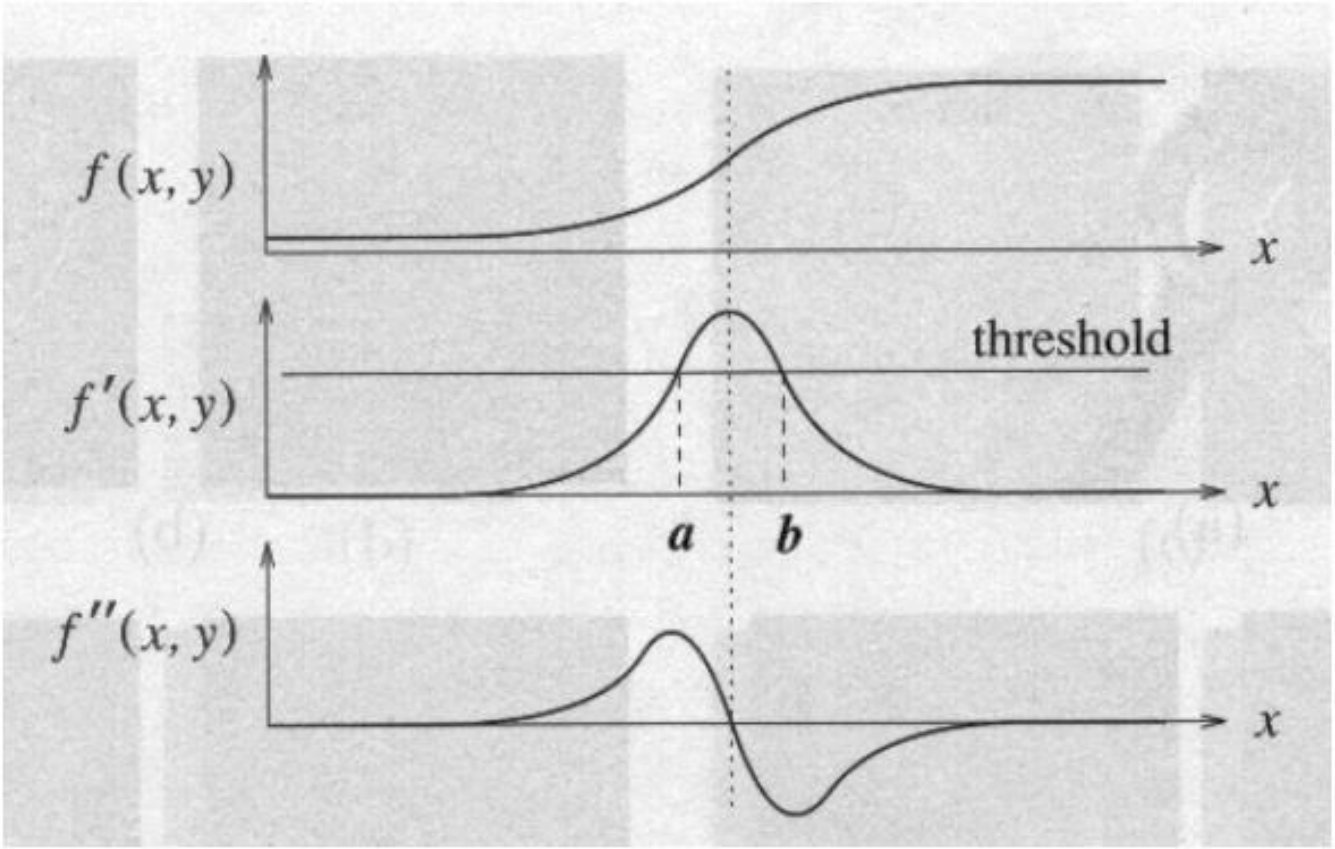


Combined Edge Image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Edge Detection Using Second Derivative

Edge points can be detected by finding the zero-crossings of the second derivative.



Laplacian Edge Detection

We encountered the 2nd-order derivative based Laplacian filter.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Example of Laplacian Mask

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

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

Properties of Laplacian Mask

- It is an isotropic operator.
- It is cheaper to implement (one mask only).
- It does not provide information about edge direction.
- It is more sensitive to noise.

  The Laplacian is typically not used by itself as it is too sensitive to noise.

  Usually Laplacian is combined with a smoothing Gaussian filter for edge detection.

Laplacian of Gaussian (LoG)

The Laplacian is combined with smoothing as a precursor to find edges via zero-crossing.

Consider the function

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}}, \quad \text{where} \quad r^2 = x^2 + y^2$$

and σ is the standard deviation.

Convolving this function with an image blurs the image, with the degree of blurring being measured by σ.

Laplacian of Gaussian (LoG)

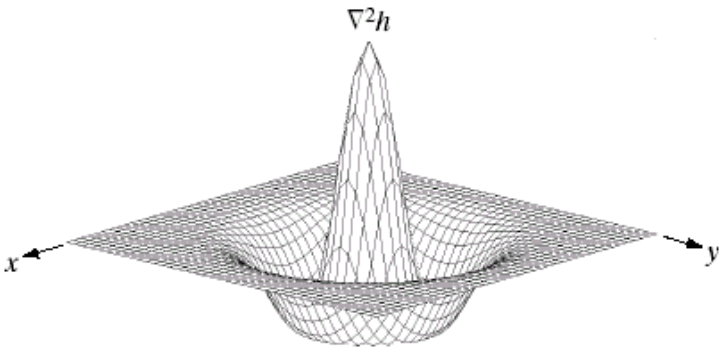
The Laplacian of h (the second derivative of h with respect to r) is

$$\Delta^2 h(r) = -\left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}$$

This function is commonly referred to as the Laplacian of Gaussion (LoG).

Laplacian of Gaussian (LoG)

The Laplacian of Gaussian (or Mexican hat) filter uses the Gaussian for noise removal and the Laplacian for edge detection.



3-d plot

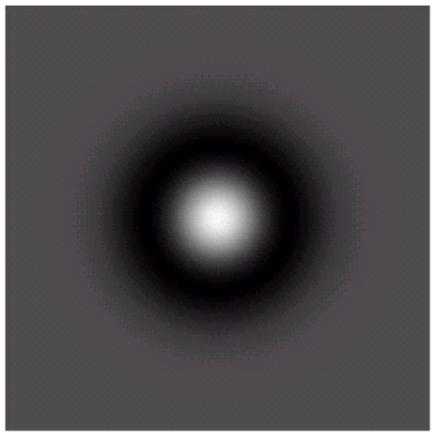
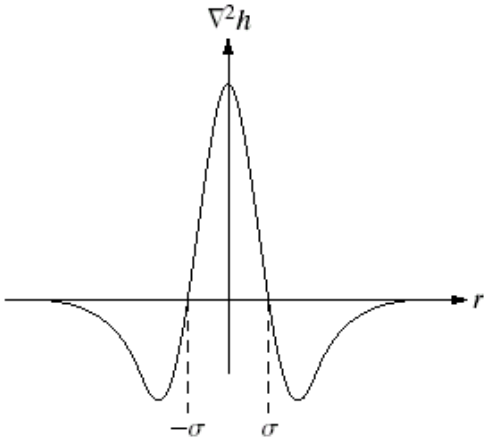


Image (black is negative, gray is the zero plane, and white is positive)



Cross section showing zero crossing

5×5 mask approximation to the shape of LoG

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

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Laplacian Of Gaussian (LoG)

17×17 Laplacian of Gaussian mask

0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0

Gradient vs LoG: A Comparison

- Gradient works well when the image contains sharp intensity transitions and low noise.
- Zero-crossings of LoG offer better localization, especially when the edges are not very sharp.

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

A sample image containing a vertical step edge.

0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

A sample image containing a vertical ramp edge.

0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0

The Canny Method [Canny'1986]

This is probably the most widely used edge detector in Computer Vision applications.

The analysis is based on "step-edges" corrupted by "additive Gaussian noise".

It is also based on first-derivative coupled with noise cleaning.

Canny edge detector tries to achieve an optimal trade-off between the accuracy of edge detection and uncertainty in localizing the edge.

The Canny Method

Algorithm

1. Compute f_x and f_y

$$f_x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G = f * G_x$$

$$f_y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y} G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2} G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2} G(x, y)$

2. Compute the gradient magnitude

$$magn(i, j) = \sqrt{f_x^2 + f_y^2}$$

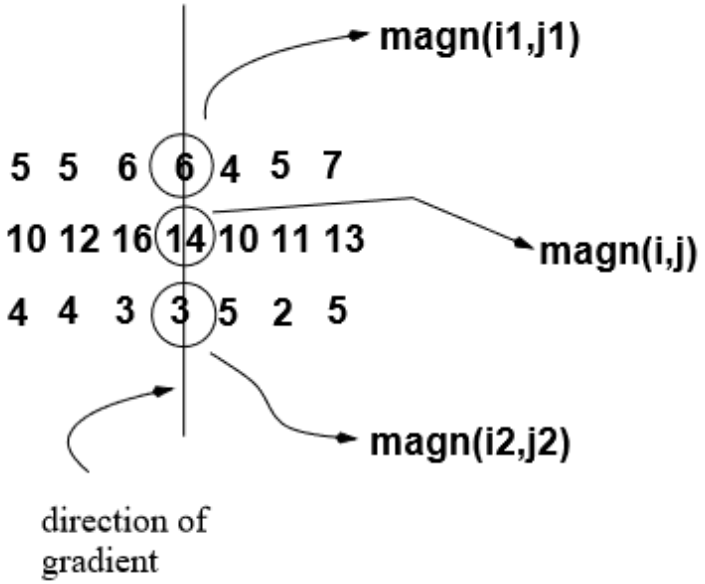
3. Apply non-maxima suppression.

4. Apply hysteresis thresholding/edge linking.

Non-Maxima Suppression

To find the edge points, we need to find the local maxima of the gradient magnitude.

All values along the direction of the gradient that are not peak values of a ridge are suppressed.



Algorithm

For each pixel (x,y) do:

if $magn(i,j) < magn(i_1,j_1)$ or $magn(i,j) < magn(i_2,j_2)$
then $I_N(i,j) = 0$
else $I_N(i,j) = magn(i,j)$

Hysteresis Thresholding/Edge Linking

- The output of non-maxima suppression still contains the local maxima created by noise.

- Can we get rid of them just by using a single threshold?

    * if we set a low threshold, some noisy maxima will be accepted too.

    * if we set a high threshold, true maxima might be missed (the value of true maxima will fluctuate above and below the threshold, fragmenting the edge).

- A more effective scheme is to use two thresholds:

    * a low threshold t_l
    * a high threshold t_h
    * usually, $t_h \approx 2t_l$

Hysteresis Thresholding/Edge Linking

Algorithm

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$.

(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

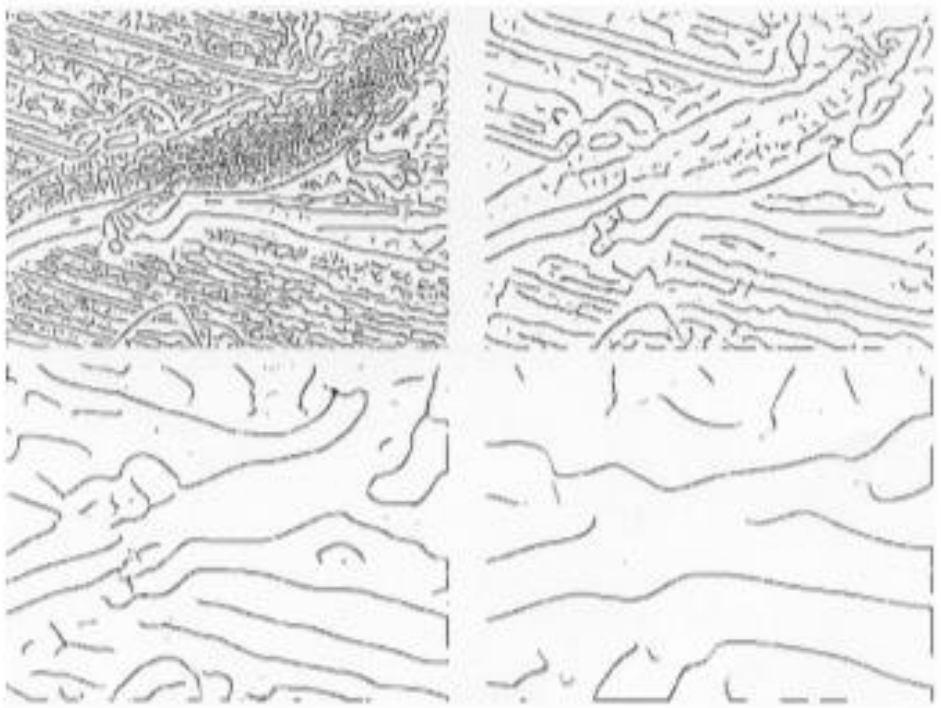
2. Link the edges in $I_2(i, j)$ into contours

    2.1 Look in $I_1(i, j)$ when a gap is found.

    2.2 By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $I_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

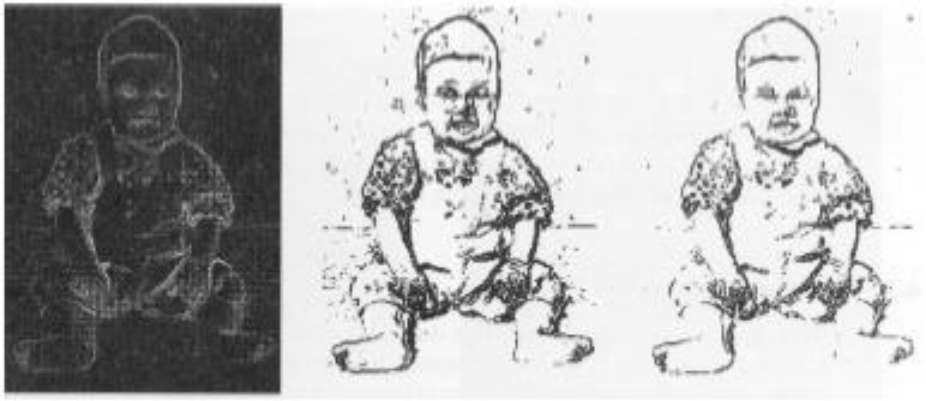
The algorithm performs edge linking as a by-product of double-thresholding.

Canny Results



(Canny edges at multiple scales of smoothing, σ=0.5, 1, 2, 4, 8, 16)

Test Results



(left:Sobel, middle: thresh=35, right: thersh=50)

(Canny - left:σ=1, middle: σ=2, right: σ=3)

Hough Transform

Overview

Performed after Edge Detection.

It is a technique to isolate the curves of a given shape / shapes in a given image.

Classical Hough Transform can locate regular curves like straight lines, circles, parabolas, ellipses, etc.

- Requires that the curve be specified in some parametric form.

Generalized Hough Transform can be used where a simple analytic description of feature is not possible.

Advantages of Hough Transform

The Hough Transform is tolerant of gaps in the edges.

It is relatively unaffected by noise.

It is also unaffected by obstruction in the image.

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$$

Parameter space also called Hough Space

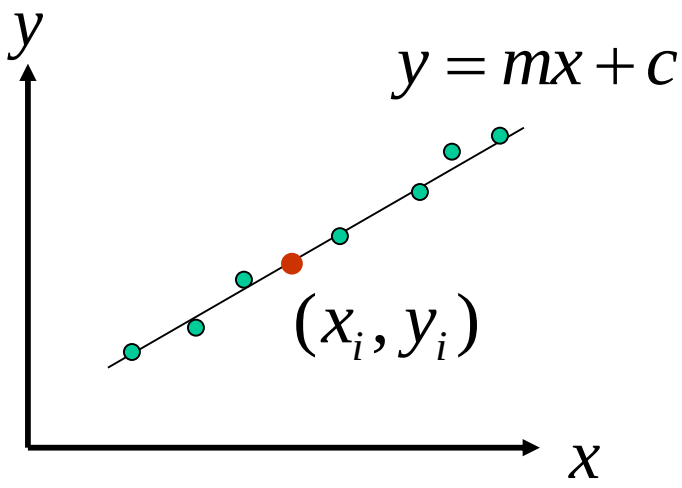
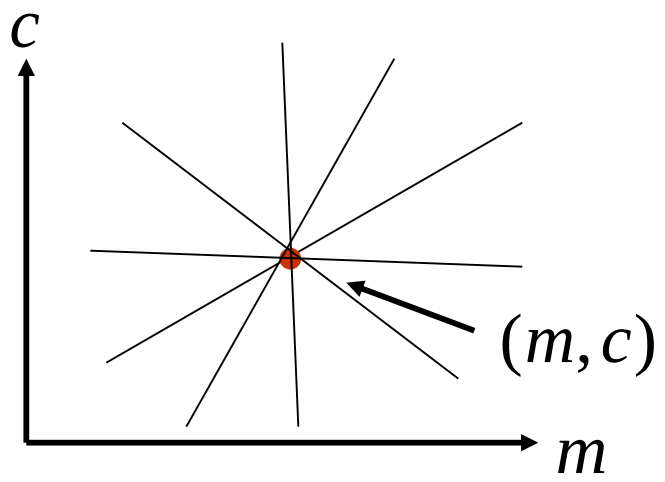


Image Space

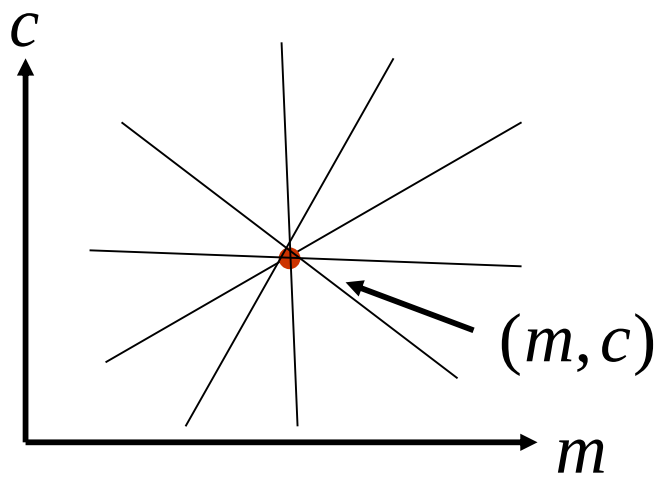


Parameter Space

Line Detection by Hough Transform

Algorithm:

- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m, c) = 0 \quad \forall m, c$
- For each image point (x_i, y_i)
  - If (m, c) lies on the line $c = -x_i m + y_i$
    - increment : $A(m, c) = A(m, c) + 1$
- Find local maxima in $A(m, c)$



Parameter Space

$A(m, c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Better Parameterization

NOTE: $-\infty \le m \le \infty$

Large Accumulator

More memory and computations

Improvement: (Finite Accumulator Array Size)

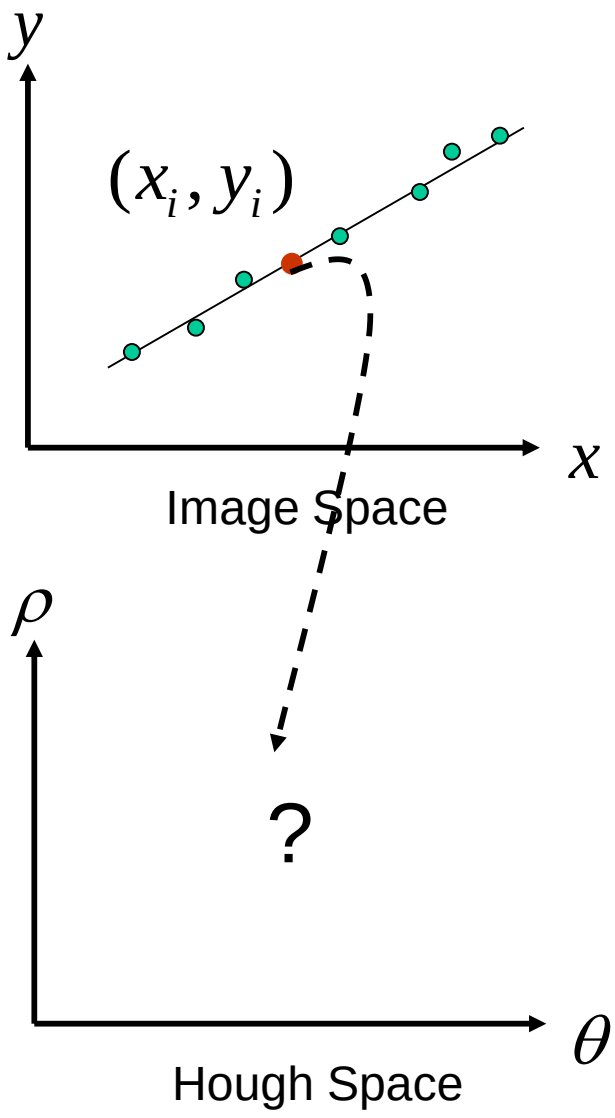
Line equation: $\rho = x\cos\theta + y\sin\theta$

Here $0 \le \theta \le \pi$

$0 \le \rho \le \rho_{\max}$

Given points (x_i, y_i) find (ρ, θ)

Hough Space Sinusoid



Hough Transform for Straight Line Detection

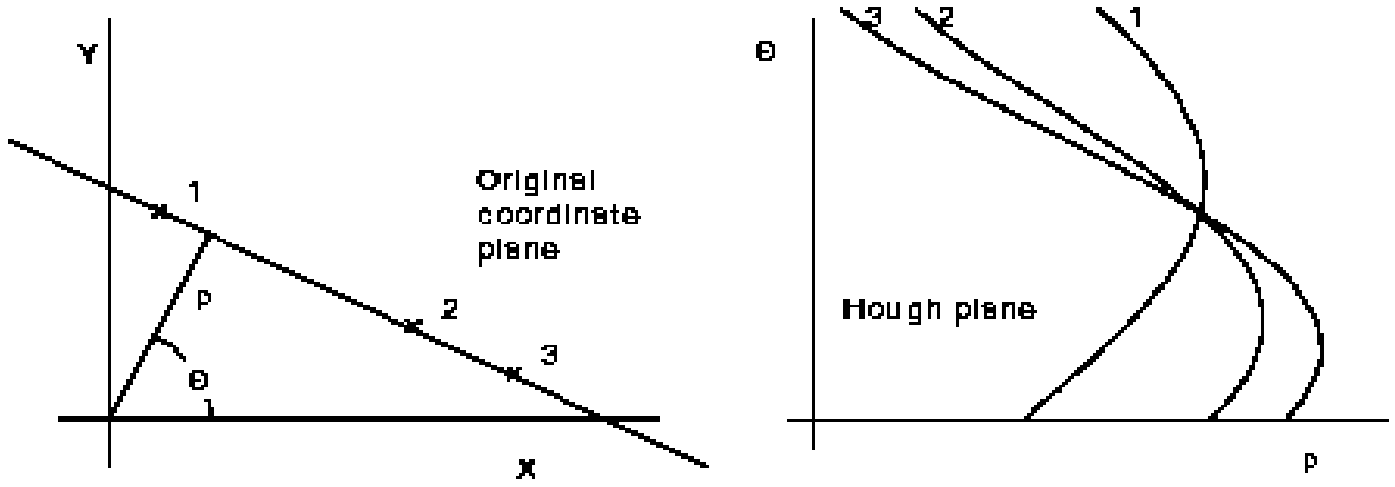
A straight line can be represented as

y = mx + c

This representation fails in case of vertical lines

A more useful representation in this case is

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



Hough Transform for Straight Lines

Advantages of Parameterization

- Values of 'ρ' and 'θ' become bounded.

How to find intersection of the parametric curves?

- Use of accumulator arrays – concept of 'Voting'.
- To reduce the computational load use Gradient information.

Hough Transform for Straight Lines - Algorithm

Quantize the Hough Transform space: identify the maximum and minimum values of ρ and θ.

Generate an accumulator array $A(\rho, \theta)$; set all values to zero.

For all edge points (x_i, y_i) in the image

- Use gradient direction for θ.
- Compute ρ from the equation.
- Increment $A(\rho, \theta)$ by 1.

For all cells in $A(\rho, \theta)$

- Search for the maximum value of $A(\rho, \theta)$.
- Calculate the equation of the line.

Hough Transform for Straight Lines - Algorithm

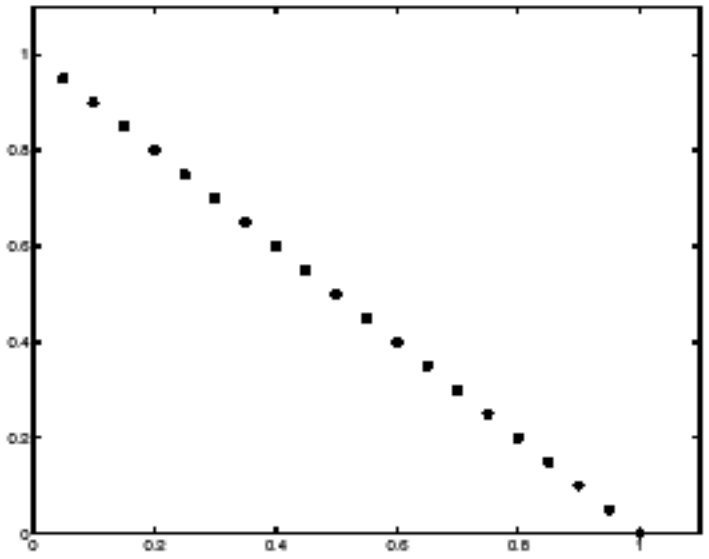
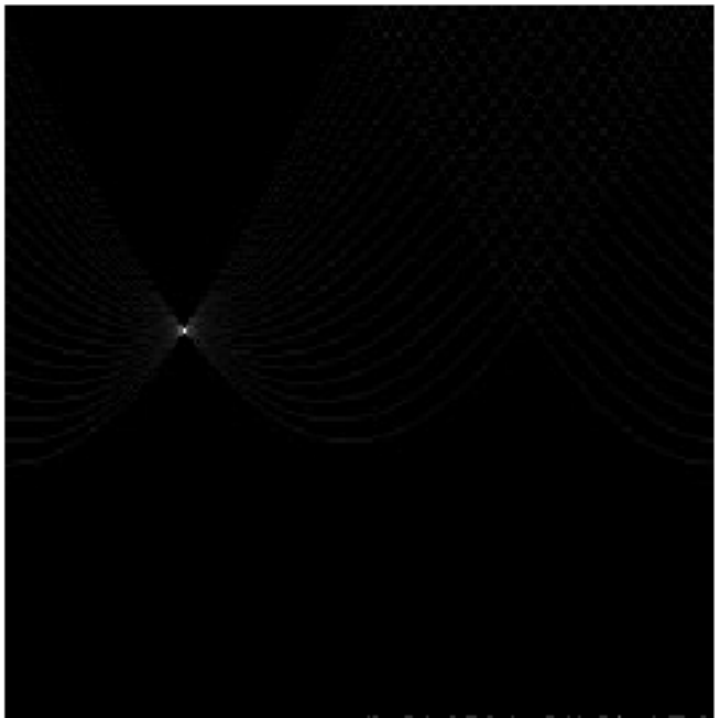


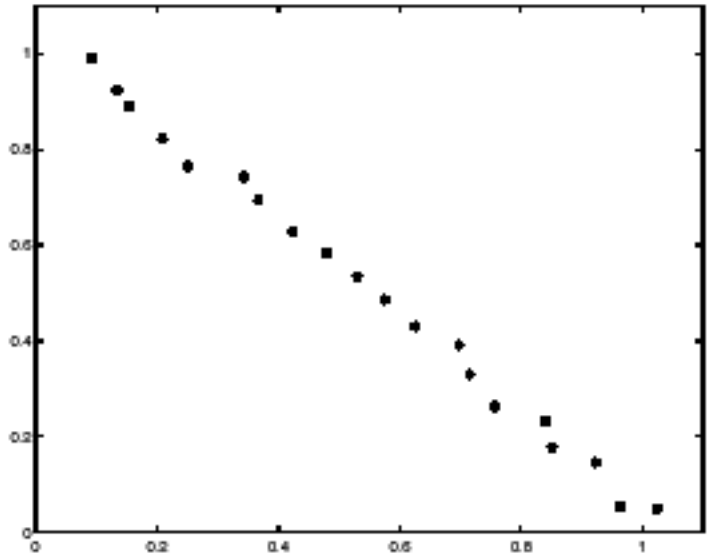
Image space

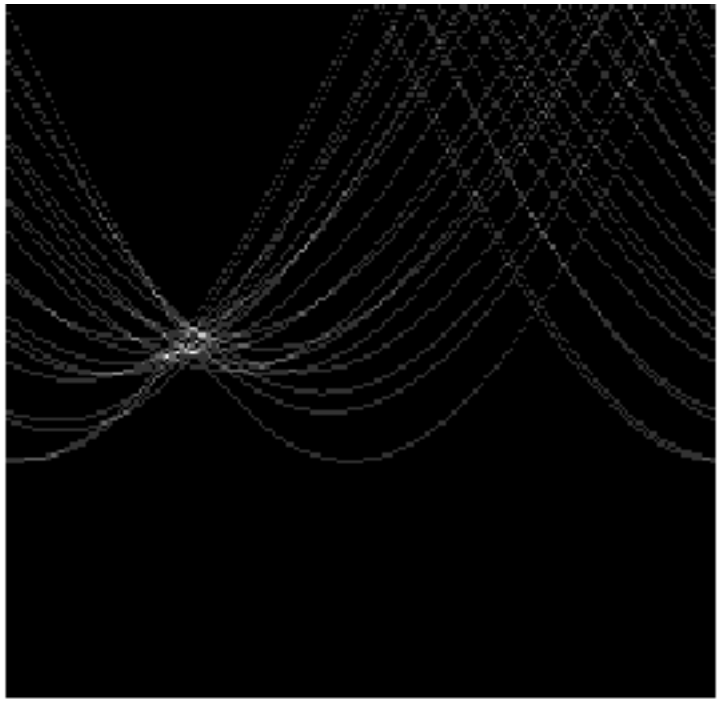


Votes

Horizontal axis is θ, vertical is rho.

Hough Transform for Straight Lines - Algorithm



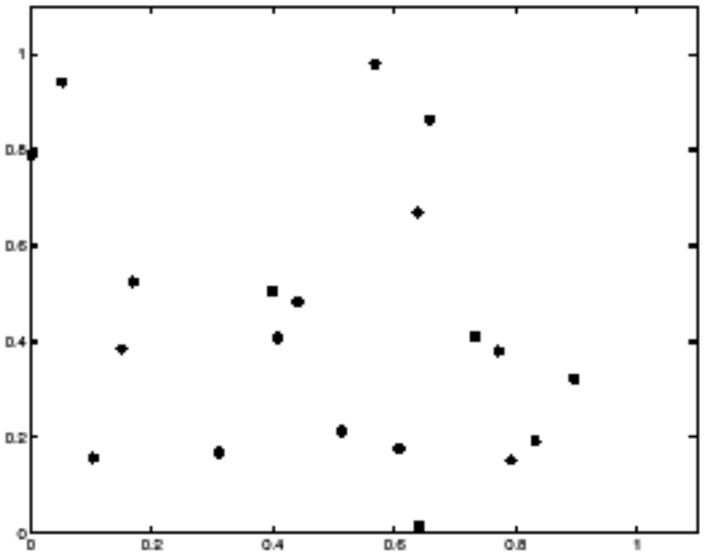


Votes

Image space

Horizontal axis is θ, vertical is rho.

Hough Transform for Straight Lines - Algorithm



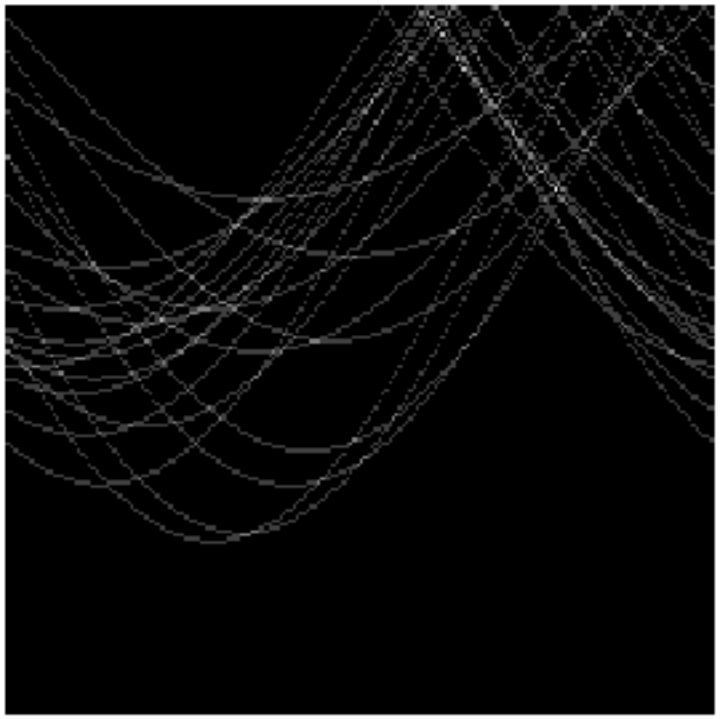
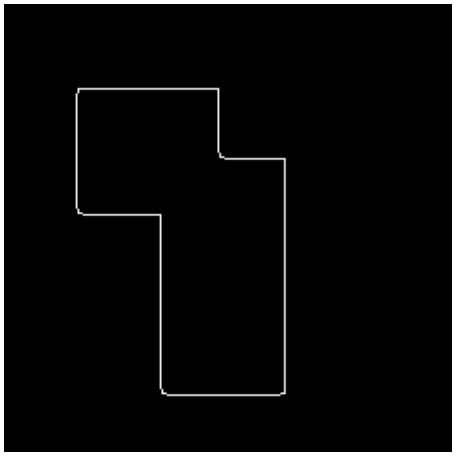


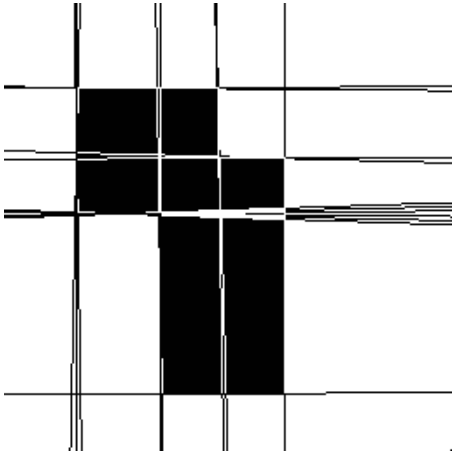
Image space

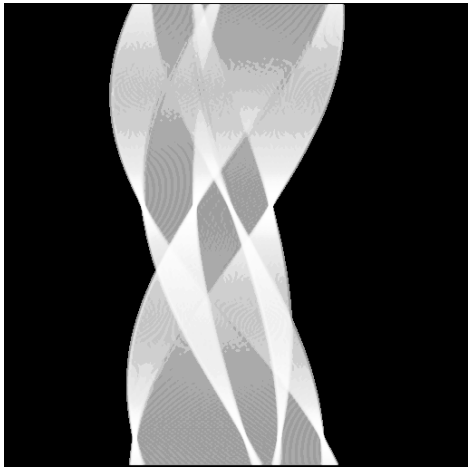
Votes

Horizontal axis is θ, vertical is rho.

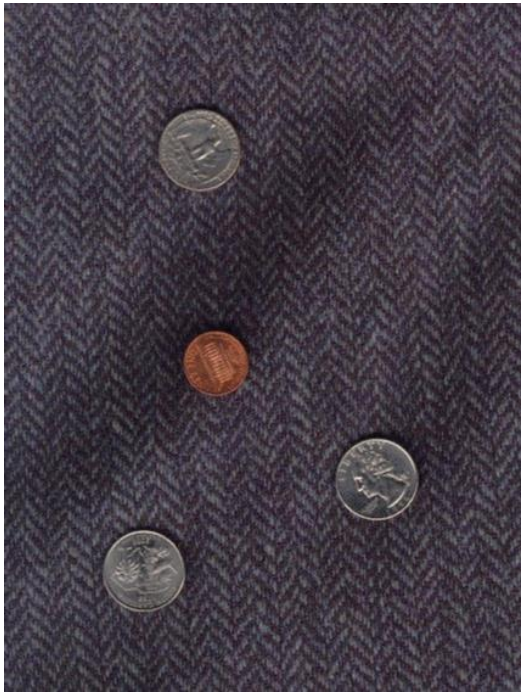
Line Detection by Hough Transform



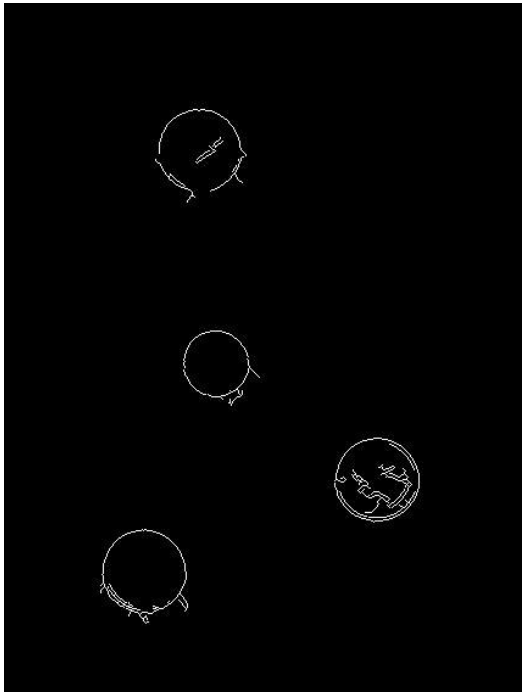




Detection of Circle by Hough Transform - Example

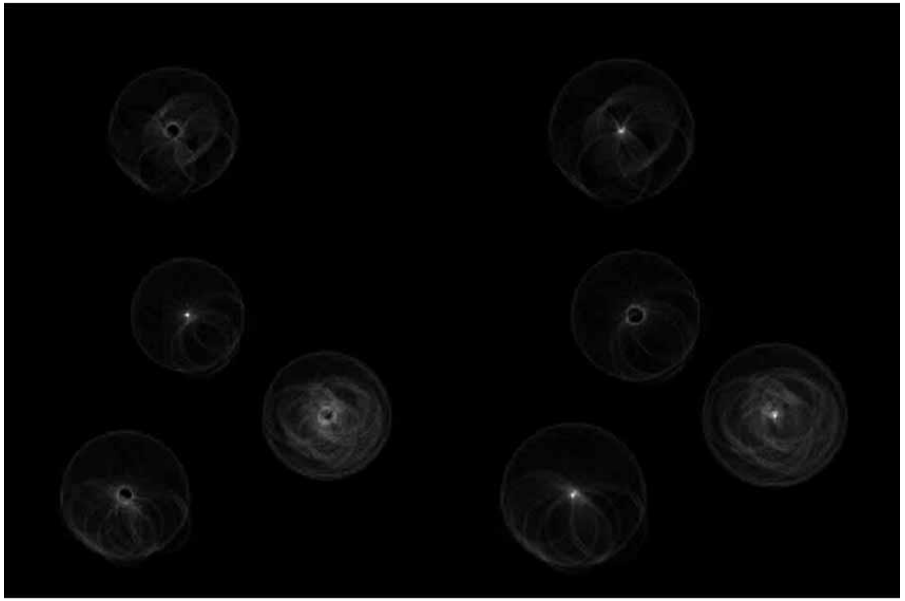


Original Image

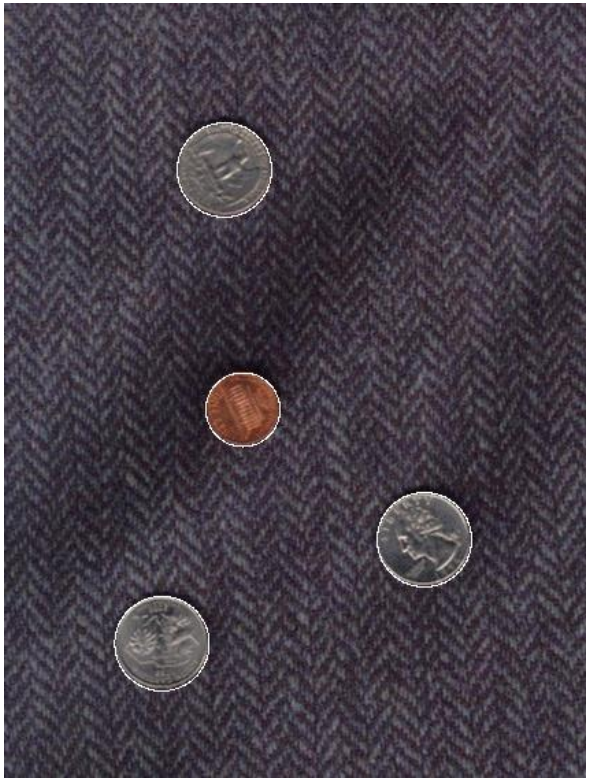


Circles detected by Canny Edge Detector

Detection of circle by Hough Transform



Hough Transform of the edge detected image

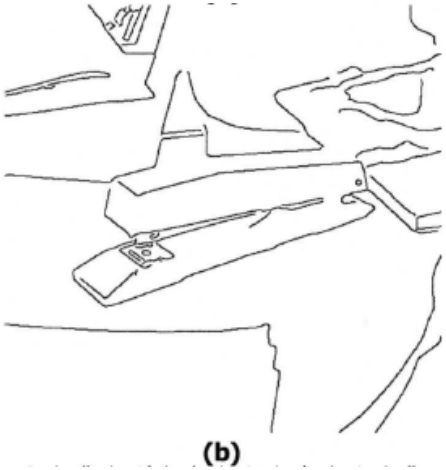


Detected Circles

Example

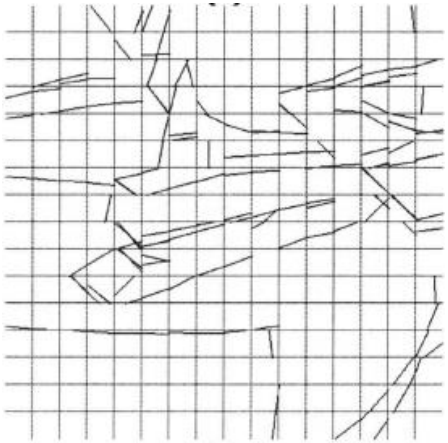
(a)

Original Image



(b)

Edge Detected Image



(c)

HT Results

Summary of Hough Transform

Hough Transform is a “voting” algorithm.

Since each point is handled independently, parallel implementations are possible.

It becomes difficult when the dimension of the parameter space is large.

Summary

We have discussed image segmentation and in particular point, line, and edge detection.

Edge detection is important as it is in many cases the first step to object recognition.