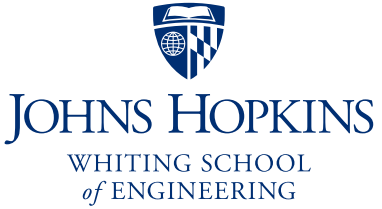
Johns Hopkins Engineering

Computer Vision

Edge and Boundary Detection



Edge and Boundary Detection

- Convert a 2D Image into a Set of Curves
- Finding **Object Boundaries** from **Edge Pixels**
- Topics
  - Theory of Edge Detection
  - Edge Detection Using Gradients and Laplacian
  - Noise suppression
  - Canny Edge Detector
  - Fitting Lines and Curves to Edges
  - The Hough Transform

Szelinsk (2011). Computer Vision: Algorithms and Applications. Chapter 3.2. (available online)

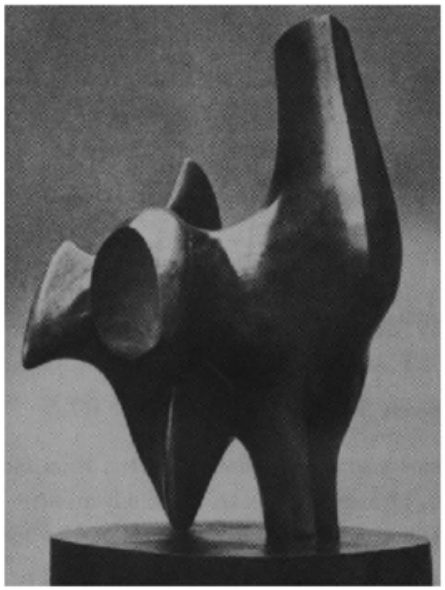
Edge and Boundary Detection

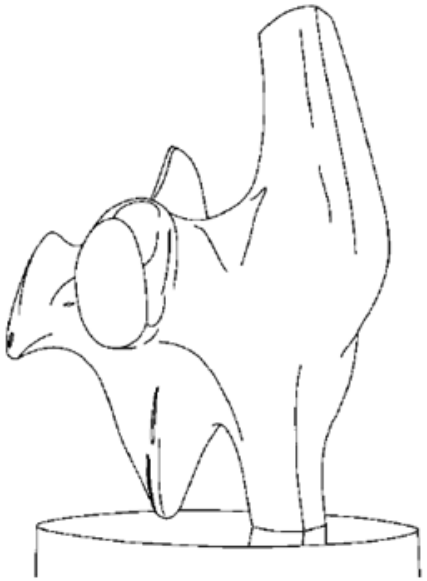
- Convert a 2D Image into a Set of Curves
- Finding **Object Boundaries** from **Edge Pixels**
- Topics
  - Theory of Edge Detection
  - Edge Detection Using Gradients

Szelinsk (2011). Computer Vision: Algorithms and Applications. Chapter 3.2. (available online)

What are Edges?

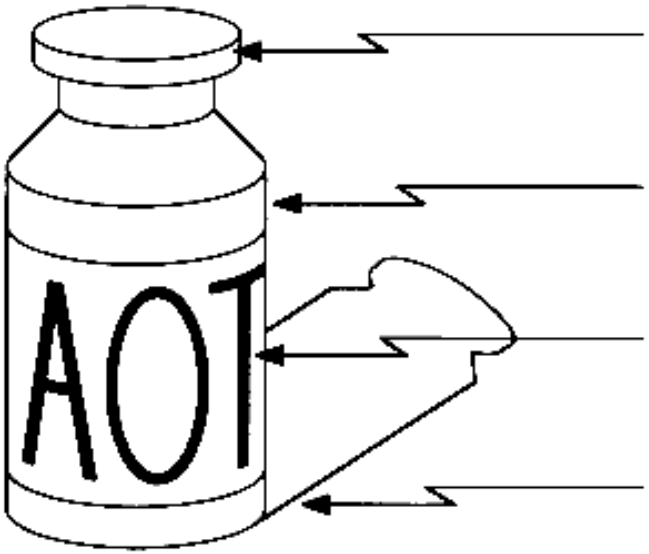
Rapid changes in image intensity within small region





Causes of Edges?

Edges are caused by a variety of factors



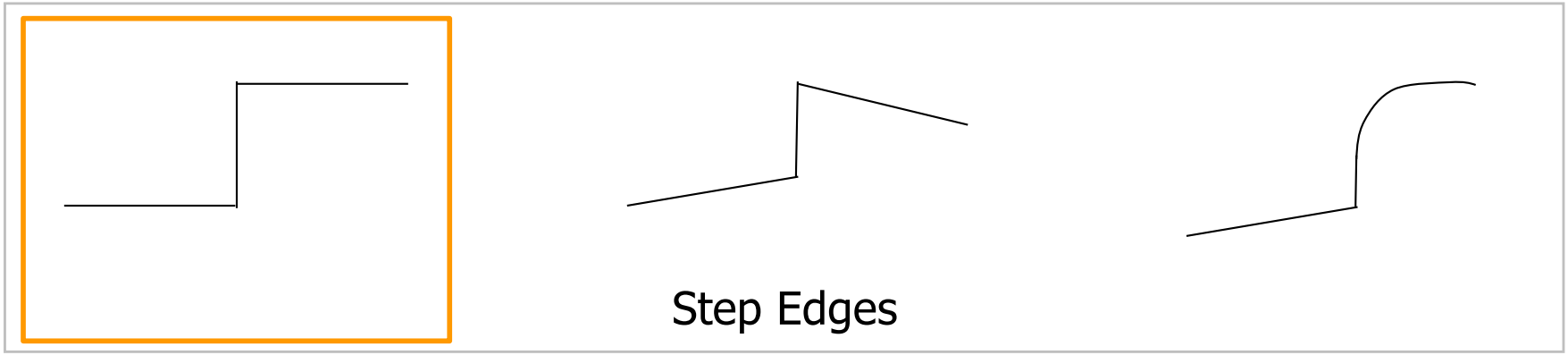
Surface Normal Discontinuity

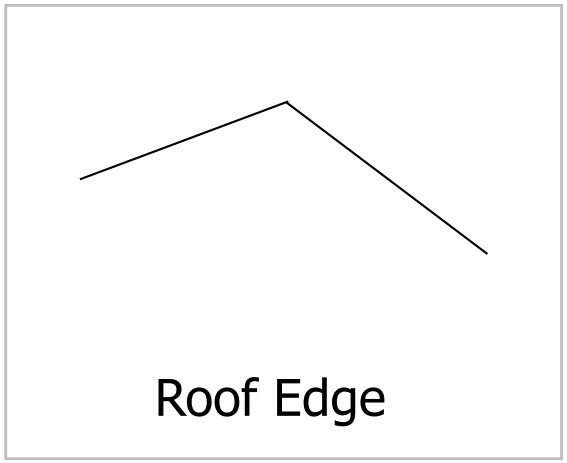
Depth Discontinuity

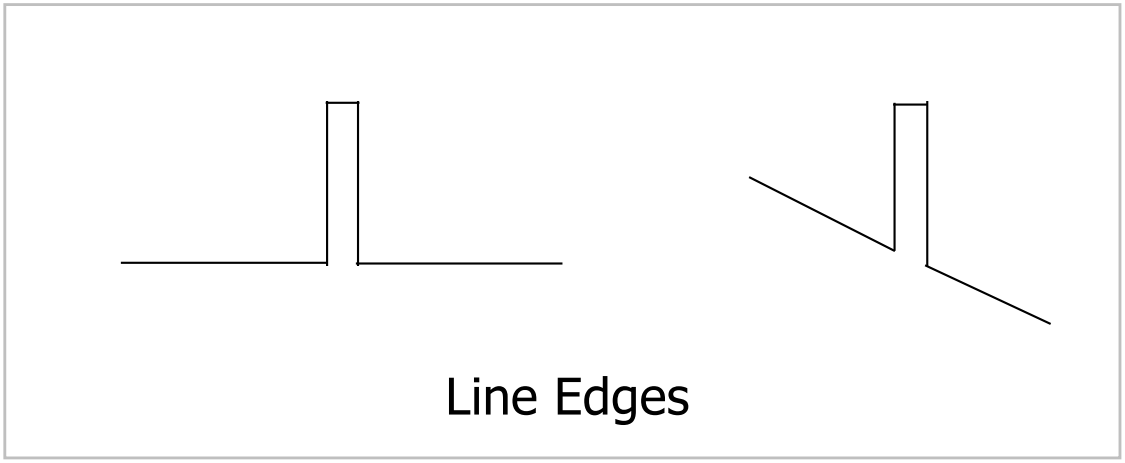
Surface Color Discontinuity

Illumination Discontinuity

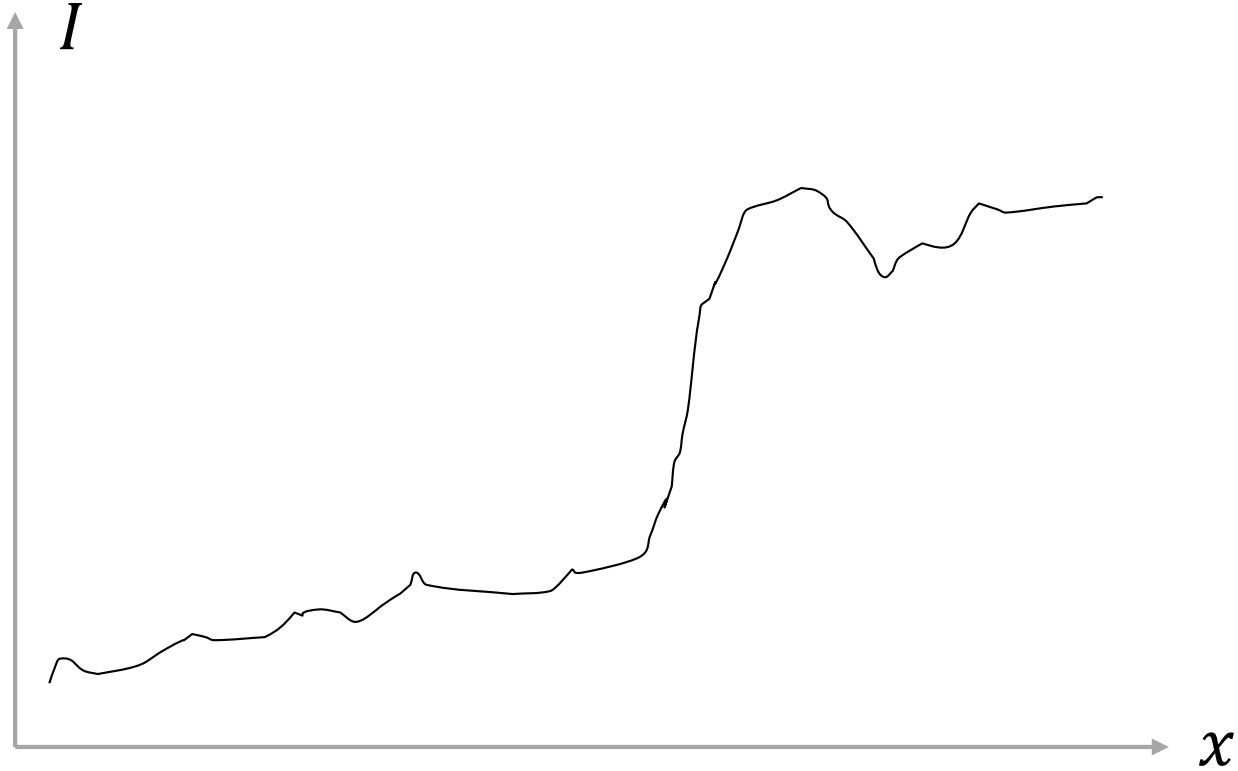
Types of Edges?







Real Edges



Problems: **Noisy** Images and **Discrete** Images

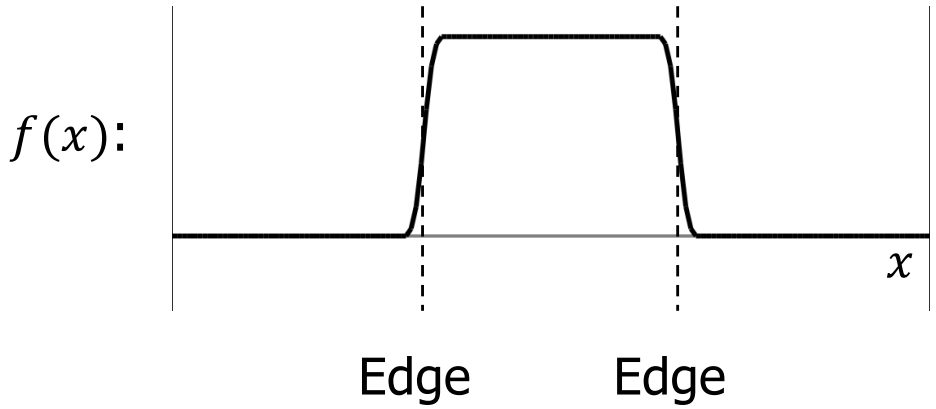
Edge Detector

- We want an **Edge Operator** that produces:
  - Edge **Position**
  - Edge **Magnitude** (Strength)
  - Edge **Orientation** (Direction)
- Crucial Requirements:
  - High **Detection Rate**
  - Good **Localization**
  - **Robust** to Noise

Szelinsk (2011). Computer Vision: Algorithms and Applications. Chapter 3.2. (available online)

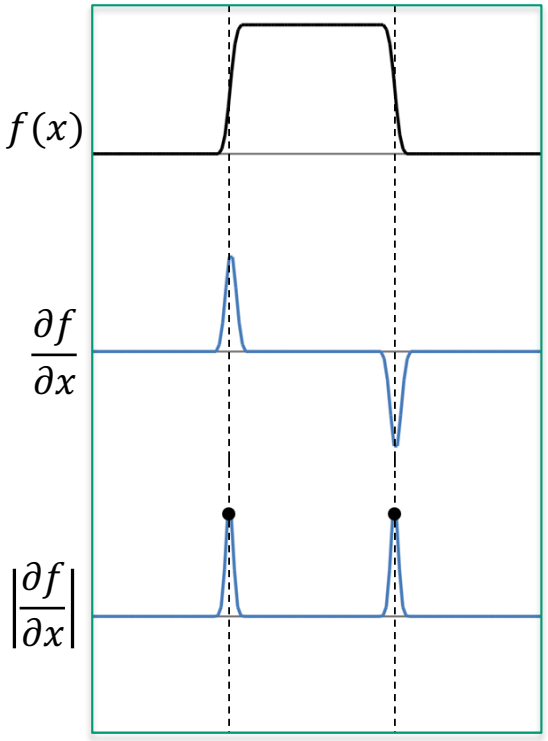
1D Edge Detection

Edges are rapid changes in image brightness in a small region.



Calculus 101: Derivative of a continuous function represents the amount of change in the function.

Edge Detection Using 1st Derivative



First Derivative: $\frac{\partial f}{\partial x}$

Local Extrema
Indicate Edges

First Derivative Absolute Value: $\left|\frac{\partial f}{\partial x}\right|$

Local Maxima
Indicate Edges

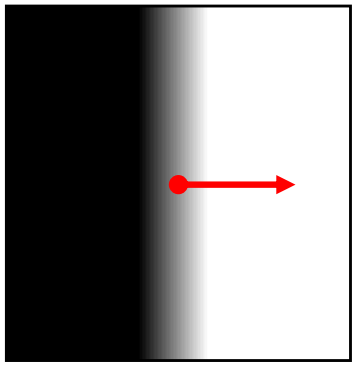
Provides Both Location and Strength of an Edge

Gradient (∇)

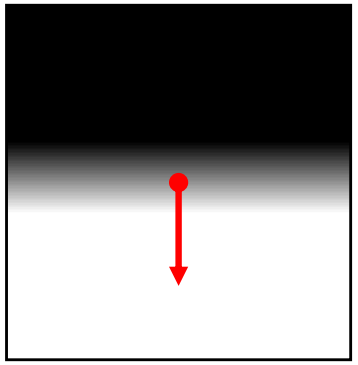
Gradient (Partial Derivative) Represents the Direction of Most Rapid Change in Intensity

$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

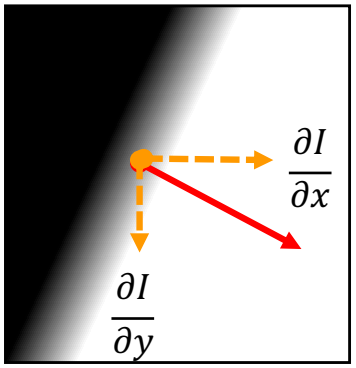
Pronounced as "Del I"



$$\nabla I = \left[\frac{\partial I}{\partial x}, 0\right]$$



$$\nabla I = \left[0, \frac{\partial I}{\partial y}\right]$$



$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

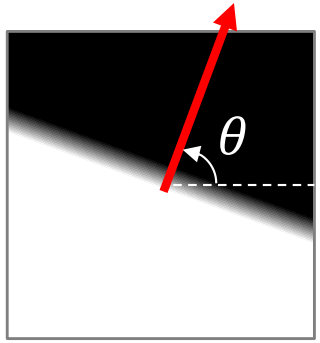
Gradient (∇) as Edge Detector

Gradient Magnitude

$$S = \|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Gradient Orientation

$$\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$$

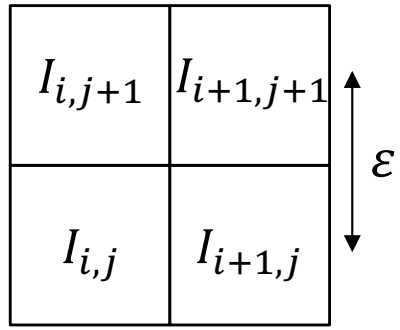


Discrete Gradient (∇) Operator

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$



Can be implemented as Convolution!

$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon}$$

-1	1
-1	1

$$\frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon}$$

1	1
-1	-1

Note: Convolution flips have been applied

Comparing Gradient (∇) Operators

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	0 1 -1 0	-1 0 1 -1 0 1 -1 0 1	-1 0 1 -2 0 2 -1 0 1	-1 -2 0 2 1 -2 -3 0 3 2 -3 -5 0 5 3 -2 -3 0 3 2 -1 -2 0 2 1
$\frac{\partial I}{\partial y}$	1 0 0 -1	1 1 1 0 0 0 -1 -1 -1	1 2 1 0 0 0 -1 -2 -1	1 2 3 2 1 2 3 5 3 2 0 0 0 0 0 -2 -3 -5 -3 -2 -1 -2 -3 -2 -1

Good Localization
Noise Sensitive
Poor Detection

Poor Localization
Less Noise Sensitive
Good Detection

Gradient (∇) Using Sobel Filter

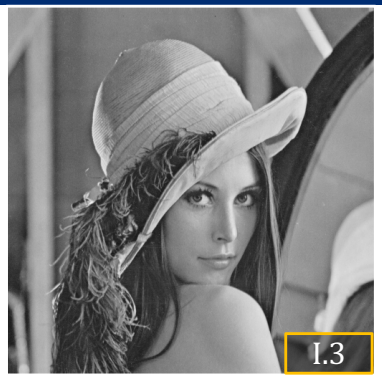


Image (I)

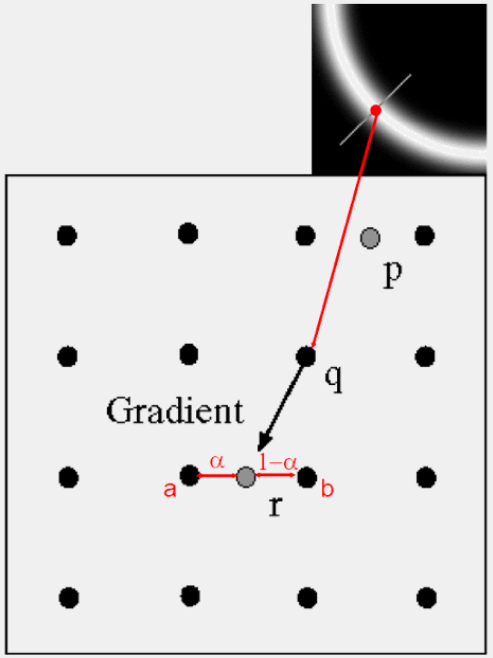
$\partial I/\partial x$

$\partial I/\partial y$

Gradient Magnitude

NonMaximal Suppression

- Edges after filtering will be "thick"
- Follow gradients and suppress (set to zero) pixels that are exceeded by a neighbor
- Can also interpolate to find subpixel maximum
- A common operation in many detection schemes, not just edge detection



With Interpolation (diagram from Nuno Vasconcelos)

Sobel Edge Detector

Image (I)

$\partial I/\partial x$

$\partial I/\partial y$

Gradient Magnitude

Thresholded Edge

Edge Detector Comparison

Image (I)

Roberts

Prewitt

Sobel

Edge Thresholding

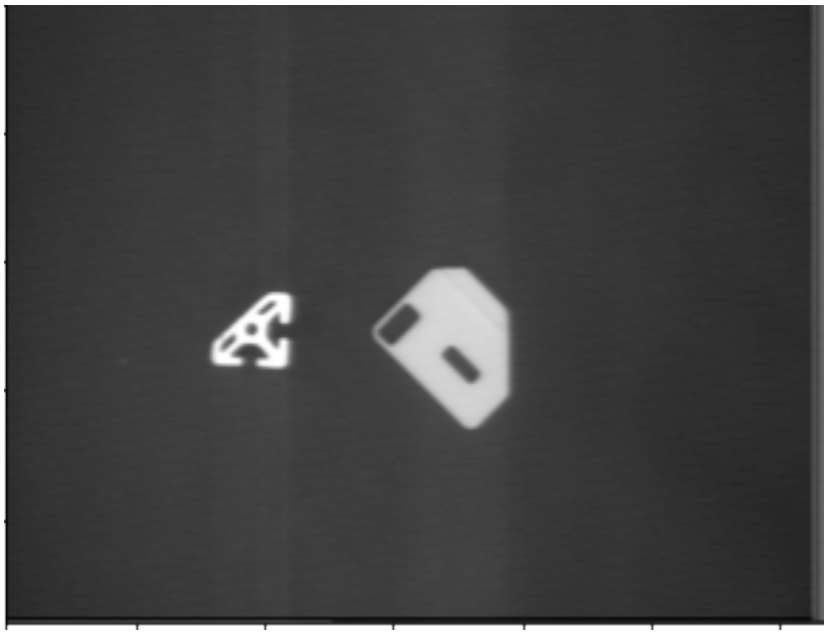
Standard: (Single Threshold T)

$\|\nabla I(x,y)\| < T$	Definitely Not an Edge
$\|\nabla I(x,y)\| \geq T$	Definitely an Edge

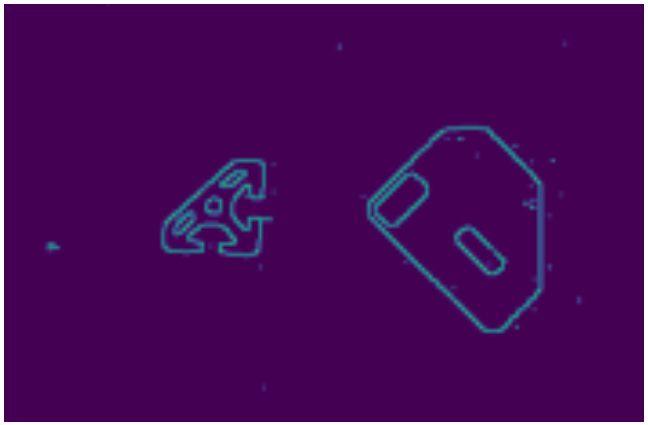
Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$\|\nabla I(x,y)\| < T_0$	Definitely Not an Edge
$\|\nabla I(x,y)\| \geq T_1$	Definitely an Edge
$T_0 \leq \|\nabla I(x,y)\| < T_1$	Is an Edge if a Neighboring Pixel if Definitely an Edge

Hysteresis Thresholding



Min=50
Max=50



Min=50
Max=100

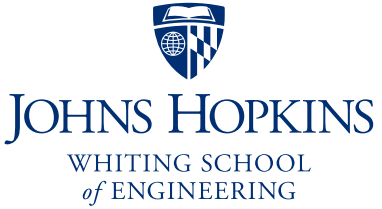
Summary

- **Derivative Operators**: A way to enhance signals to pull out content related to large brightness changes
- Essential concepts in this lecture:
  - The idea of using convolution to compute derivatives
  - Different types of derivative operators
  - The use of non-maximal suppression and hysteresis thresholding to improve detection

Johns Hopkins Engineering

Computer Vision

Edge and Boundary Detection

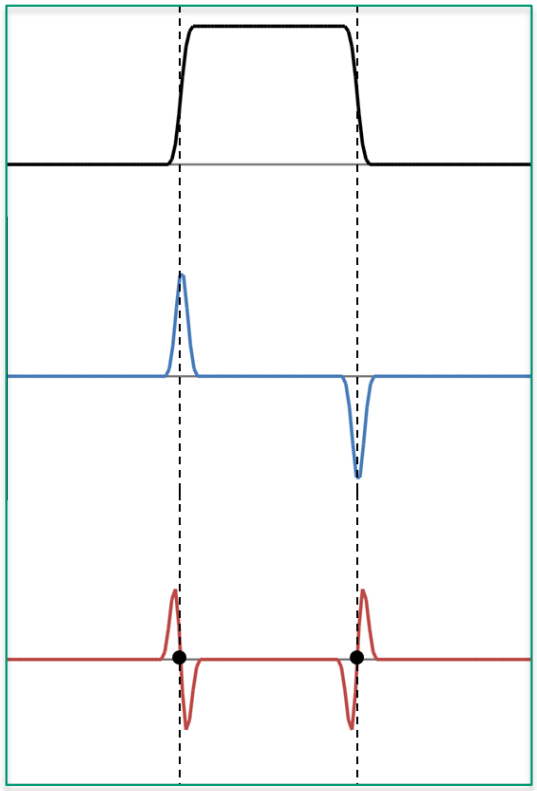


Edge and Boundary Detection

- Convert a 2D Image into a Set of Curves
- Finding **Object Boundaries** from **Edge Pixels**
- Topics
  - 2^{nd} derivative operators
  - Noise suppression
  - Canny Edge Detector

Szelinsk (2011). Computer Vision: Algorithms and Applications. Chapter 3.2. (available online)

Edge Detection Using 2nd Derivative



Local Extrema
Indicate Edges

Zero-Crossings
Indicate Edges

Laplacian (∇^2) as Edge Detector

Laplacian: Sum of Pure Second Derivatives

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Pronounced as "Del Square I"

"Zero-Crossings" in Laplacian of an image represent edges

[Marr1980]

Discrete Laplacian (∇^2) Operator

Finite difference approximations:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

ε

Convolution Mask :

$$\nabla^2 \approx \frac{1}{\varepsilon^2}$$

0	1	0
1	−4	1
0	1	0

OR

$$\nabla^2 \approx \frac{1}{6\varepsilon^2}$$

1	4	1
4	−20	4
1	4	1

(More Accurate)

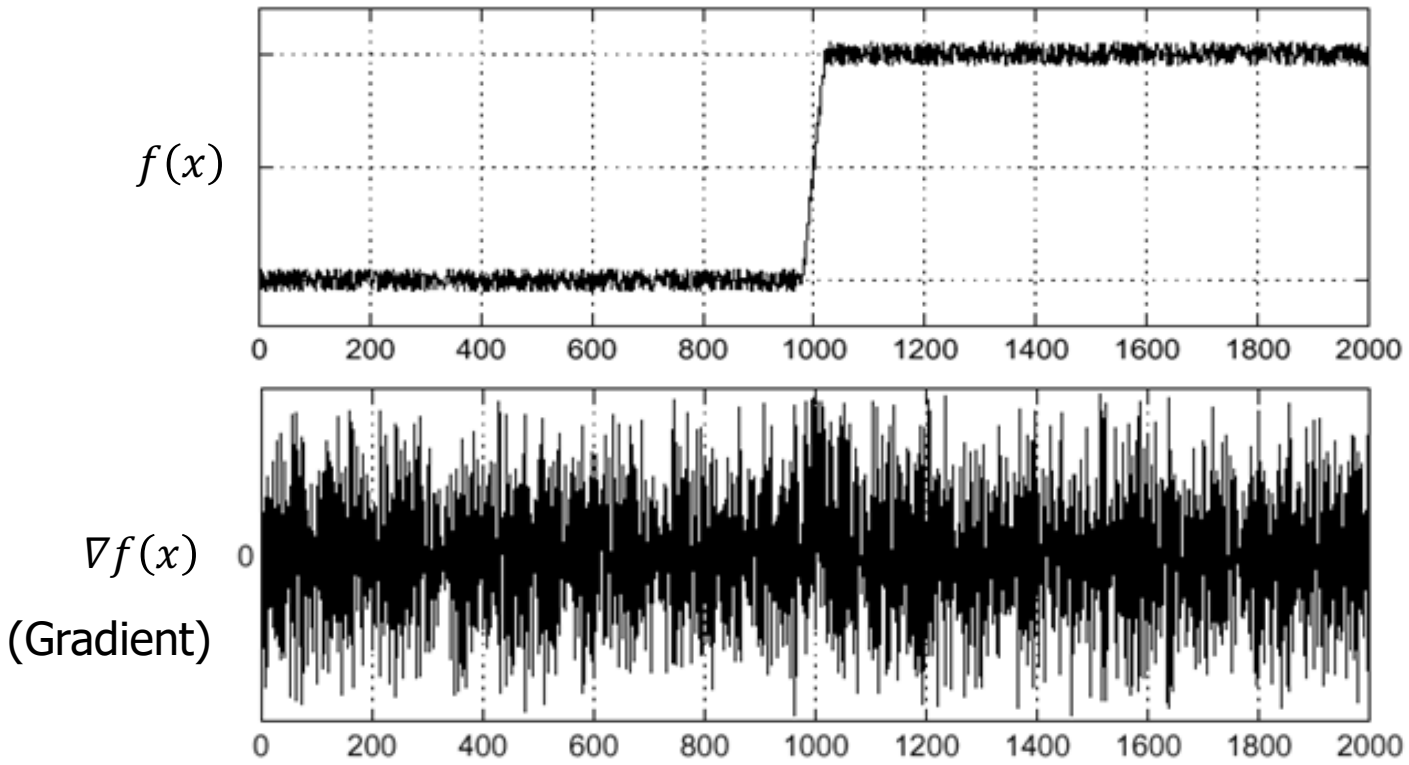
Laplacian Edge Detector

Image (I)

Laplacian
(0 maps to 128)

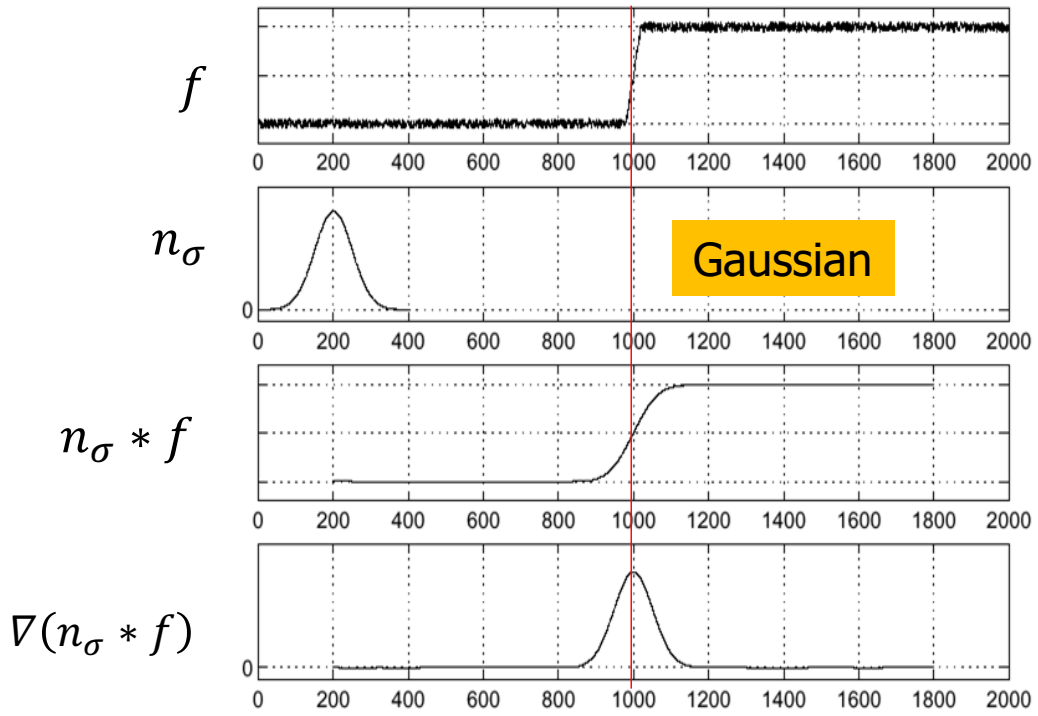
Laplacian
"Zero Crossings"

Effects of Noise



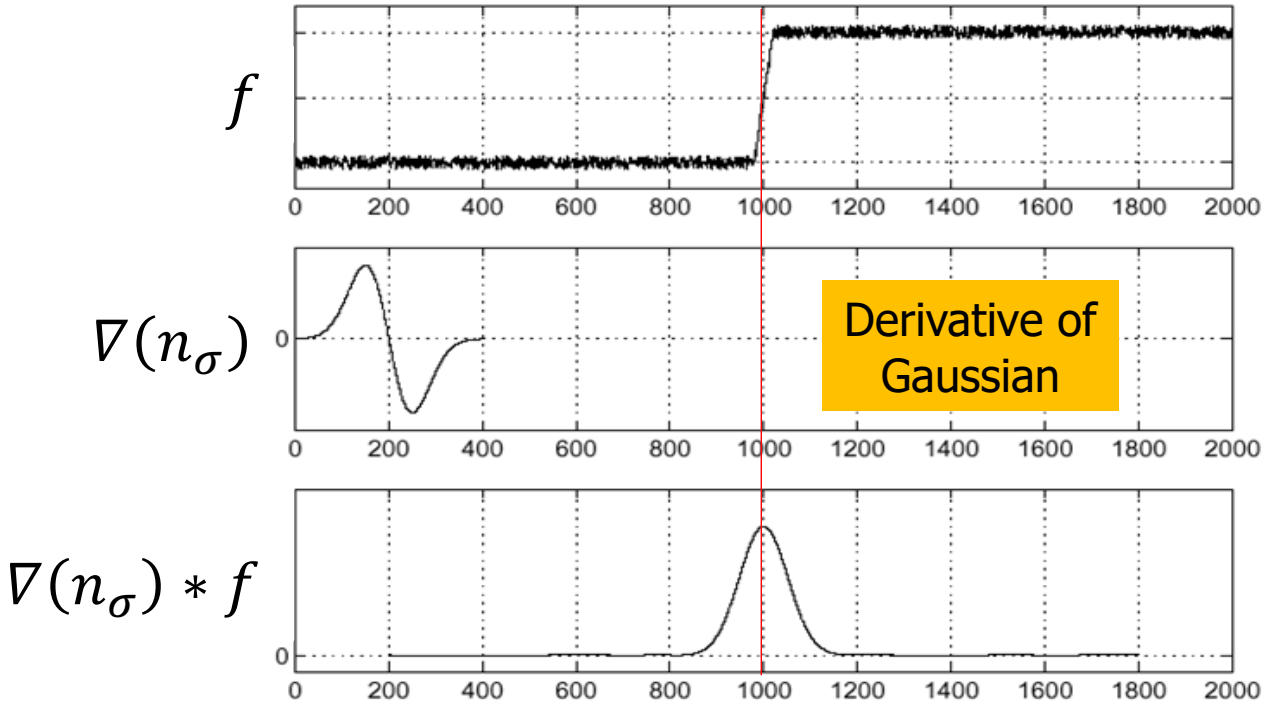
Where is the edge??

Solution: Gaussian Smooth First



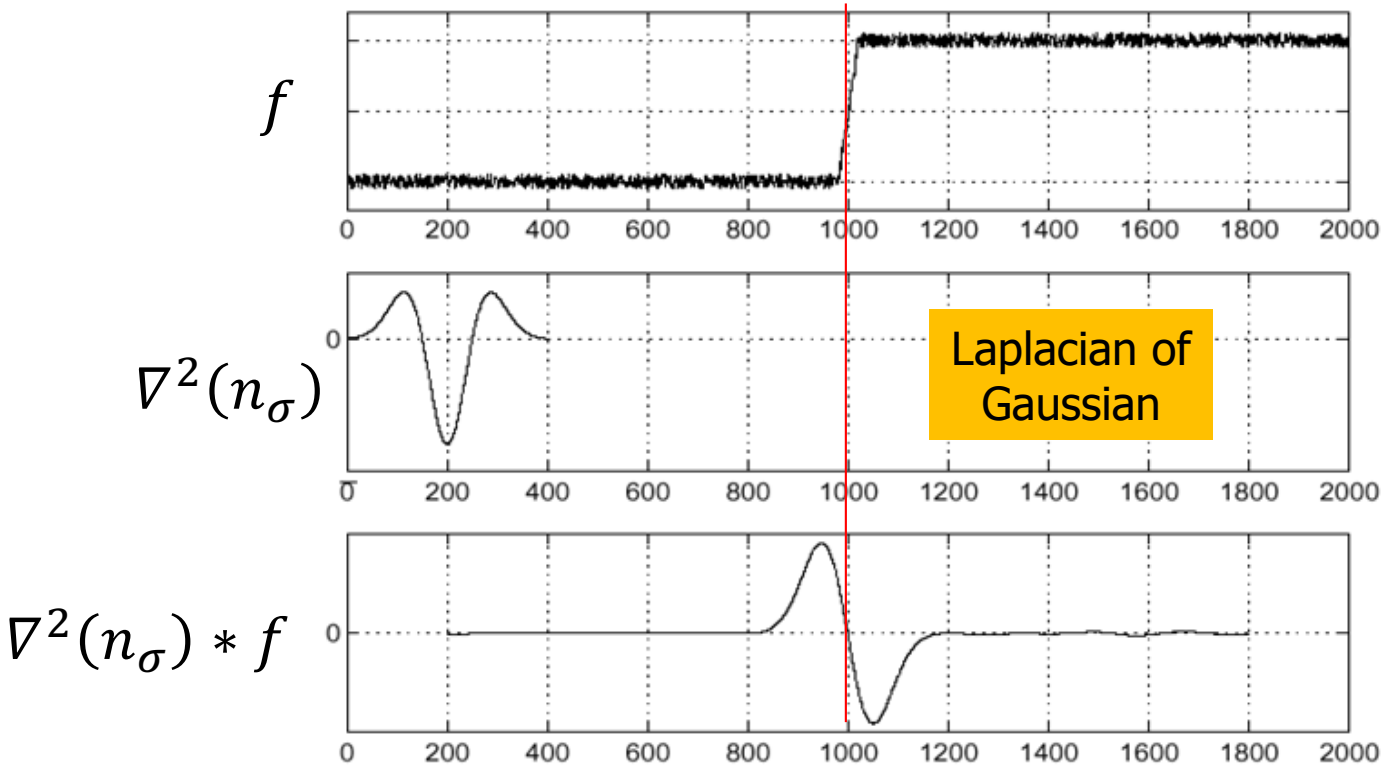
Derivative of Gaussian ($\nabla(n_\sigma)$)

$\nabla(n_\sigma * f) = \nabla(n_\sigma) * f$ …saves us one operation.



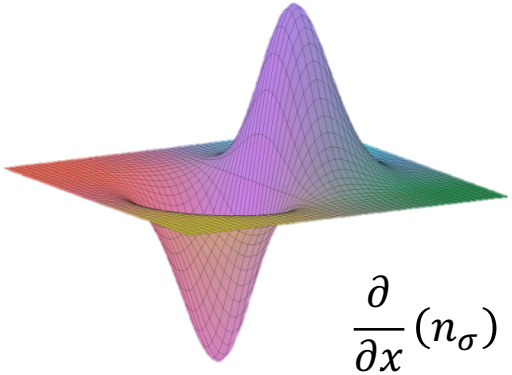
Laplacian of Gaussian ($\nabla^2 n_\sigma$ or $\nabla^2 G$)

$\nabla^2(n_\sigma * f) = \nabla^2(n_\sigma) * f$ …saves us one operation.

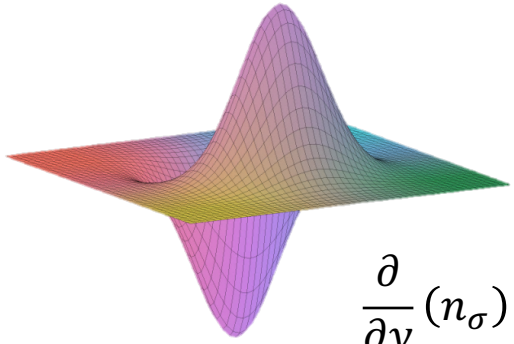


Gradient vs. Laplacian

Derivative of Gaussian (∇G)

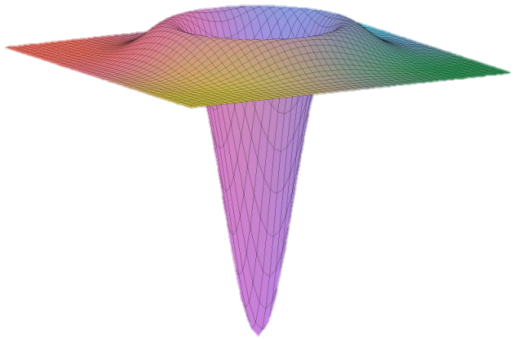


$$\frac{\partial}{\partial x}(n_\sigma)$$



$$\frac{\partial}{\partial y}(n_\sigma)$$

Laplacian of Gaussian ($\nabla^2 G$)



Inverted "Sombrero" (Mexican Hat)

$$\frac{\partial^2}{\partial x^2}(n_\sigma) + \frac{\partial^2}{\partial y^2}(n_\sigma)$$

Gradient vs. Laplacian

Provides location, magnitude and direction of the edge	Provides only location of the edge
Detection using Maxima Thresholding	Detection based on Zero-Crossing
Non-linear operation. Requires two convolutions.	Linear Operation. Requires only one convolution.

An operator that has the best of both?

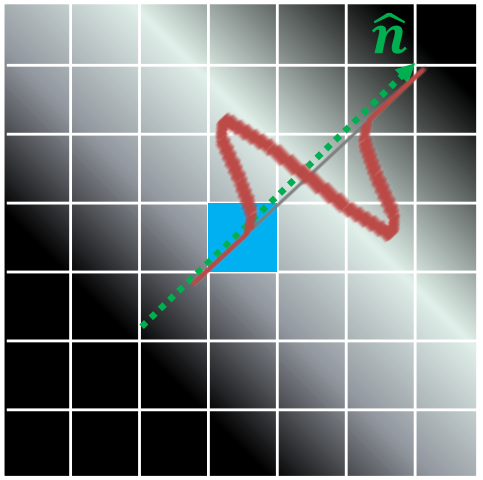
Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $|\nabla n_\sigma * I|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{|\nabla n_\sigma * I|}$$

- Compute Laplacian along the Gradient Direction $\hat{\boldsymbol{n}}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{\boldsymbol{n}}^2}$$



$|\nabla n_\sigma * I|$

Canny, 1986 http://justin-liang.com/tutorials/canny

Canny Edge Detector

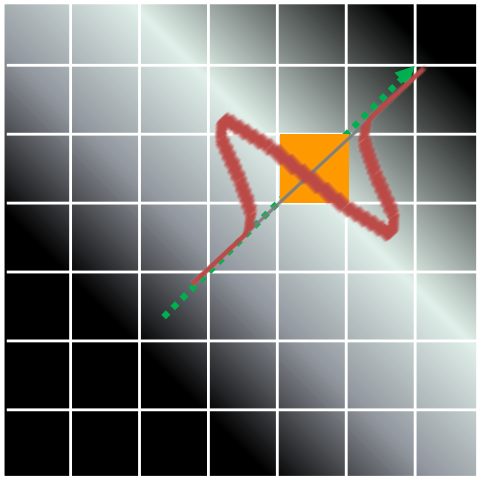
- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $|\nabla n_\sigma * I|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{|\nabla n_\sigma * I|}$$

- Compute Laplacian along the Gradient Direction $\hat{\boldsymbol{n}}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{\boldsymbol{n}}^2}$$

- Find Zero Crossings in Laplacian to find the edge location



$|\nabla n_\sigma * I|$

Canny, 1986

Canny Edge Detector

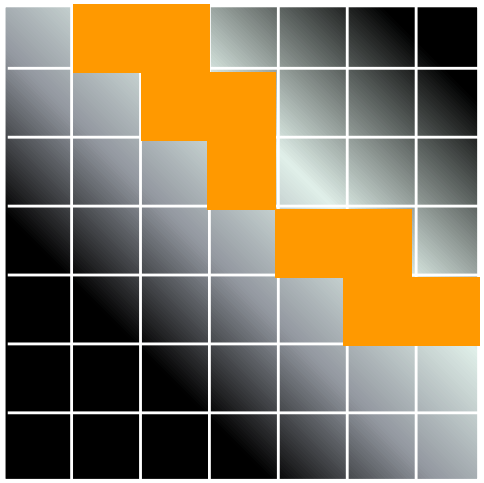
- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $|\nabla n_\sigma * I|$
- Find Gradient Orientation at each Pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla n_\sigma * I}{|\nabla n_\sigma * I|}$$

- Compute Laplacian along the Gradient Direction $\hat{\boldsymbol{n}}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{\boldsymbol{n}}^2}$$

- Find Zero Crossings in Laplacian to find the edge location
- Use hysteresis thresholding to choose true edges



$|\nabla n_\sigma * I|$

Canny, 1986

Canny Edge Detector Results

Image

$\sigma = 1$

$\sigma = 2$

$\sigma = 4$

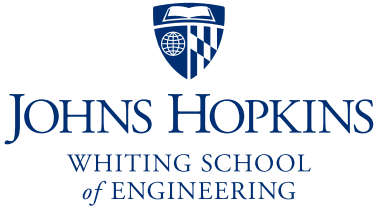
Summary

- **Edge Operators**: A way to enhance signals to pull out content related to large brightness changes
- Essential concepts in this lecture:
  - Creating band-pass filters using smoothing plus derivatives
  - Non-maximal suppression
  - The Canny Edge detector

Johns Hopkins Engineering

Computer Vision

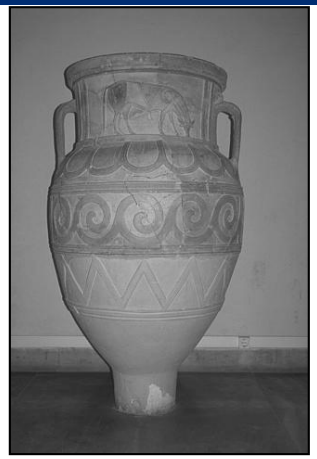
Edge and Boundary Detection



Edge and Boundary Detection

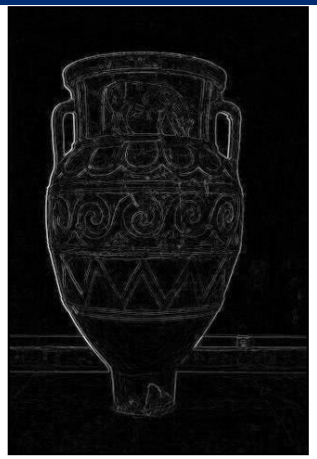
- Convert a 2D Image into a Set of Curves
- Finding **Object Boundaries** from **Edge Pixels**
- Topics
  - Fitting Lines and Curves to Edges
  - The Hough Transform

Preprocessing Edge Images

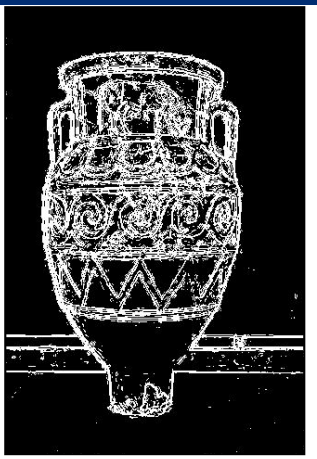


Manually Sketched

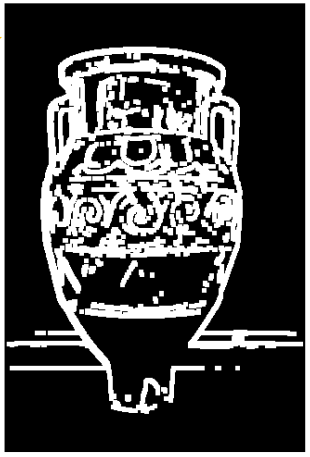
Edge Detection



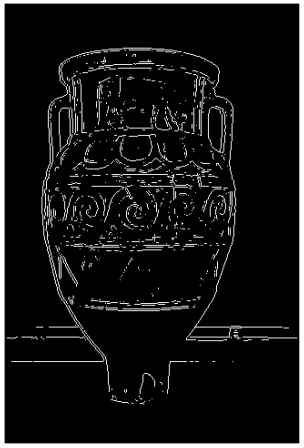
Thresholding



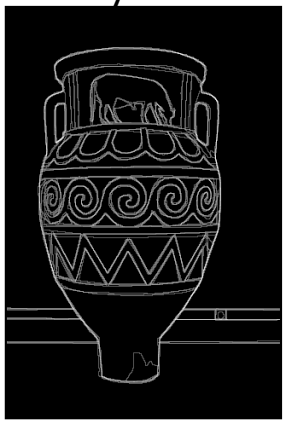
Shrink & Expand



Thinning



Boundary Detection



The Hough Transform

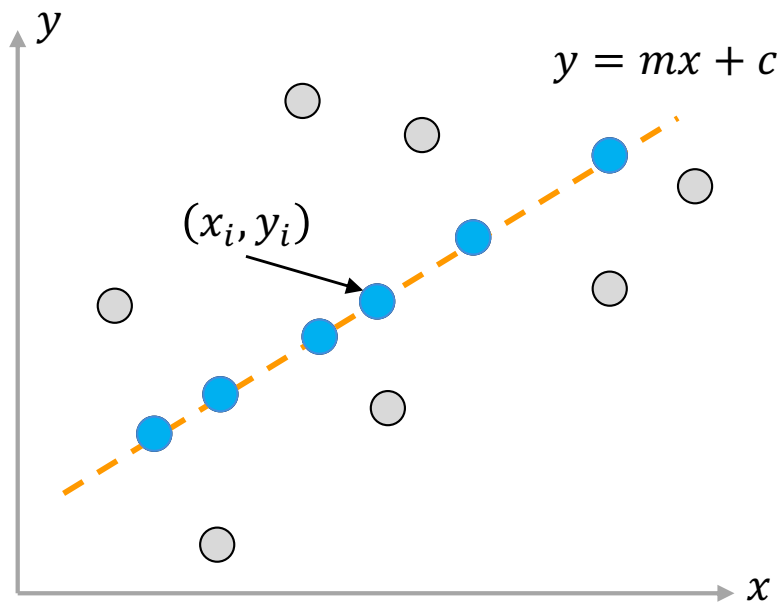
- Elegant method for Direct Object Recognition
  - Robust to disconnected edges
  - Complete object need not be visible
  - Relatively robust to noise

Hough Transform: Line Detection

Given: Edge Points (x_i, y_i)

Task: Detect line

$$y = mx + c$$

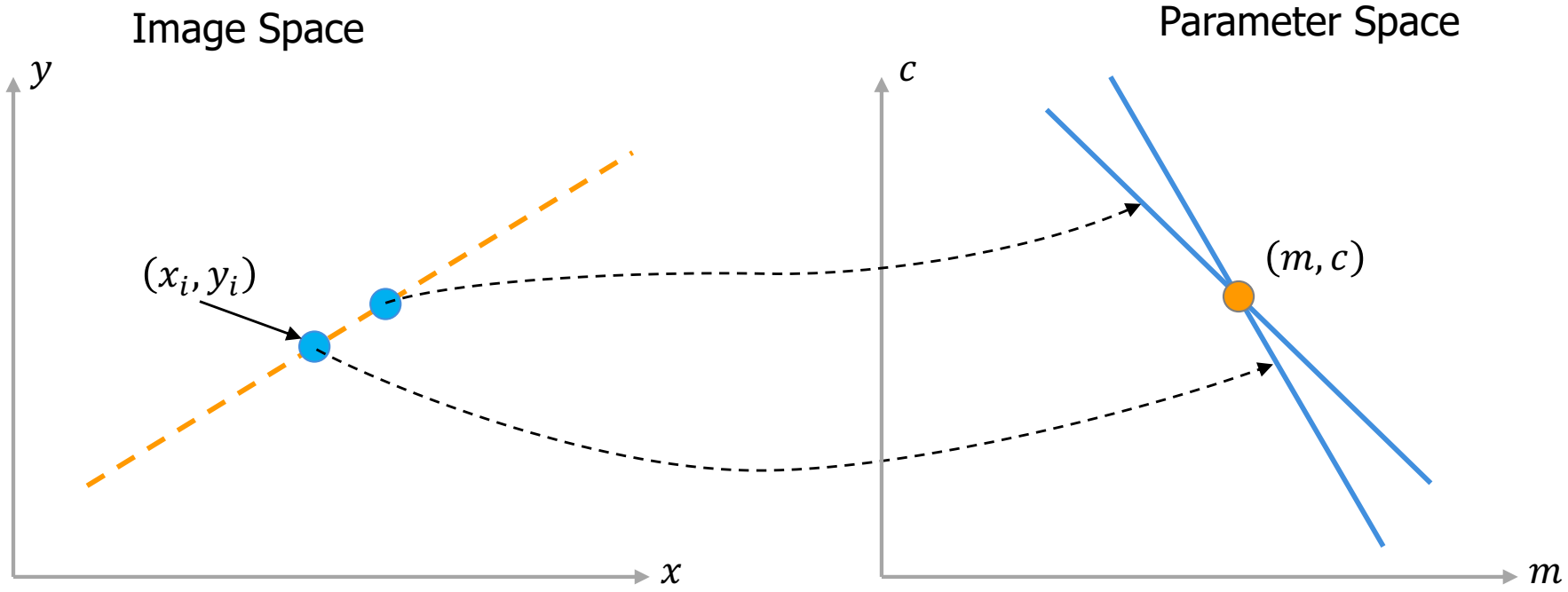


Consider point (x_i, y_i)

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

Hough, 1962

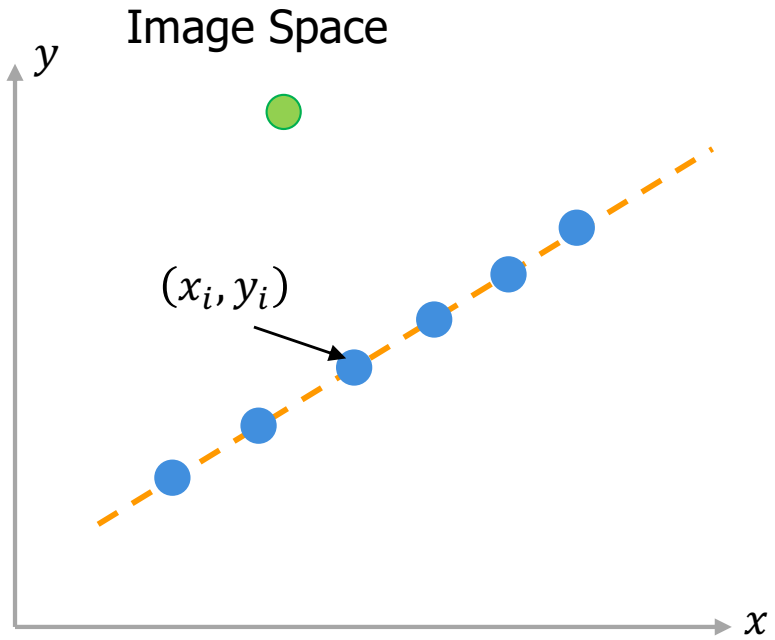
Hough Transform: Concept

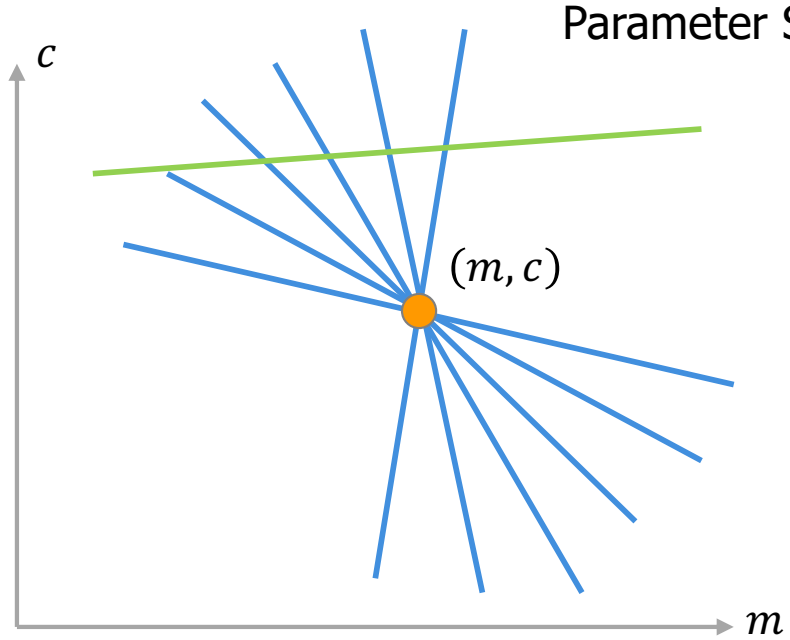


$y_i = mx_i + c$

$c = -x_i m + y_i$

Hough Transform: Concept





$y_i = mx_i + c$

$c = -x_i m + y_i$

Point ⟷ Line

Line ⟷ Point

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

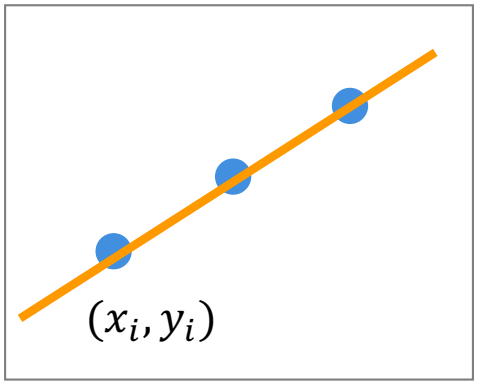
Step 4. For each edge point (x_i, y_i),

$$\boldsymbol{A(m, c) = A(m, c) + 1}$$

if (m, c) lies on the line: $\boldsymbol{c = -x_i m + y_i}$

Step 5. Find local maxima in $A(m, c)$

Image



$A(m, c)$

c					
	1	0	0	0	1
	0	1	0	1	0
	1	1	3	1	1
	0	1	0	1	0
	1	0	0	0	1

m

Multiple Line Detection

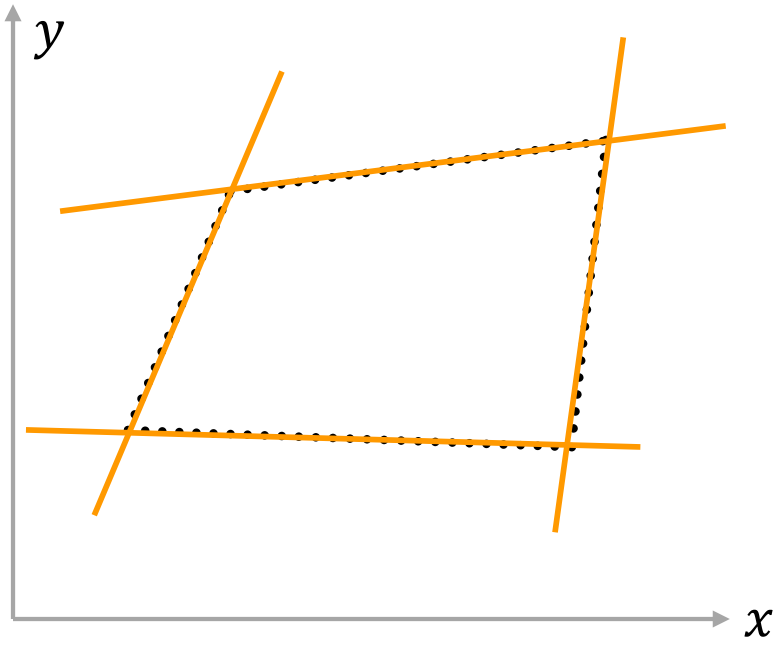
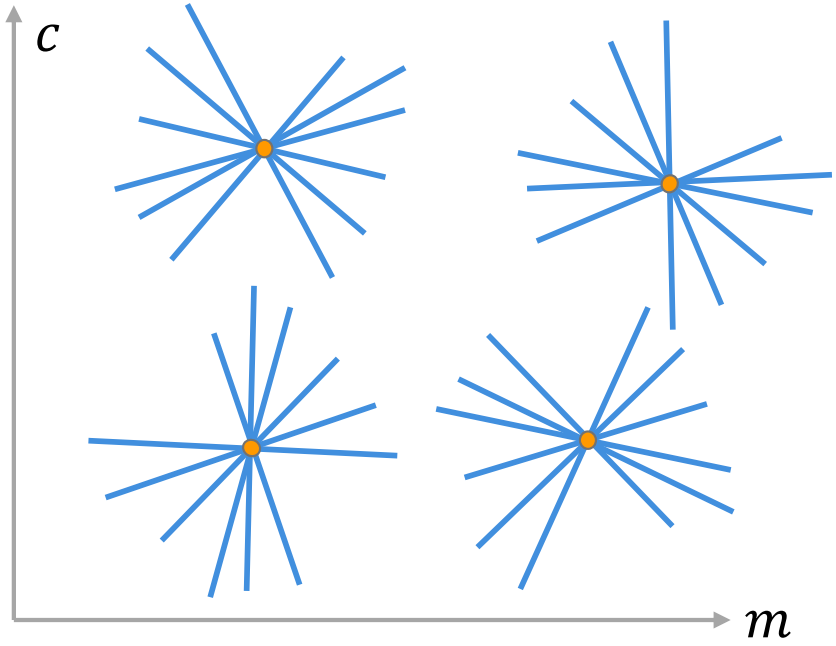


Image Space



Parameter Space

Better Parameterization

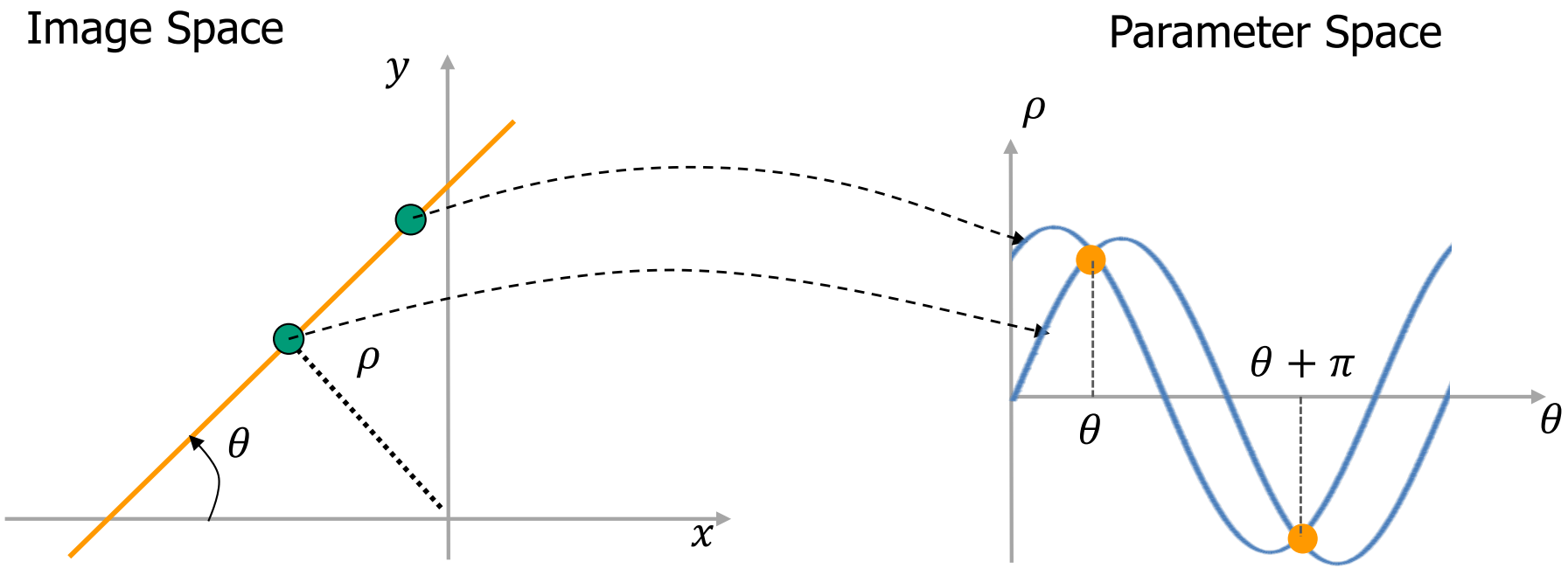
Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation
- Vertical lines cannot be represented

Solution: Use $x \sin\theta - y \cos\theta + \rho = 0$

- Orientation θ is finite: $0 \leq \theta \leq 2\pi$
- Distance ρ is finite: $0 \leq \rho \leq \rho_{max}$

Better Parameterization



$x \sin\theta - y \cos\theta + \rho = 0$

$x \sin\theta - y \cos\theta + \rho = 0$

For images: $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ and ρ_{max} = Image Diagonal

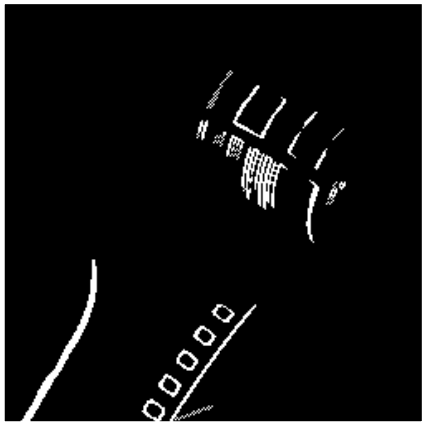
Hough Transform Mechanics

- **How big should the accumulator cells be?**
  - Too big, and different lines may be merged
  - Too small, and noise causes lines to be missed
- **How many lines?**
  - Count the peaks in the accumulator array
- **Handling inaccurate edge locations:**
  - Increment patch in accumulator rather than single point

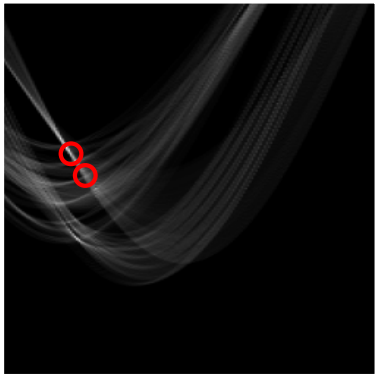
Line Detection Results

Original Image

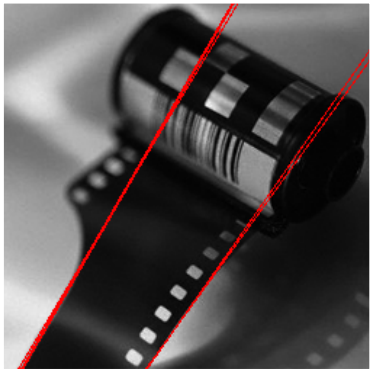
Gradient



Edge (Threshold)



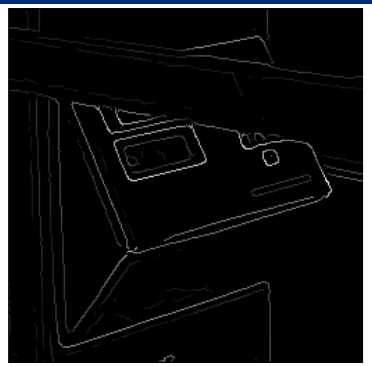
Hough Transform $A(\rho, \theta)$



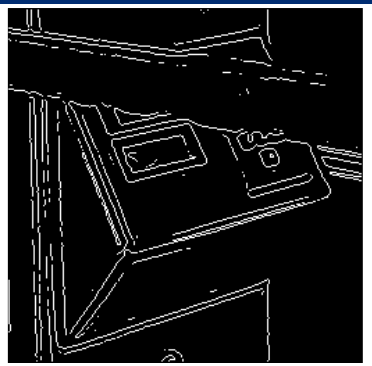
Detected Lines

Line Detection Results

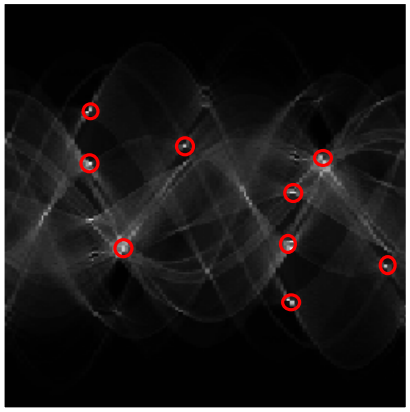
Original Image



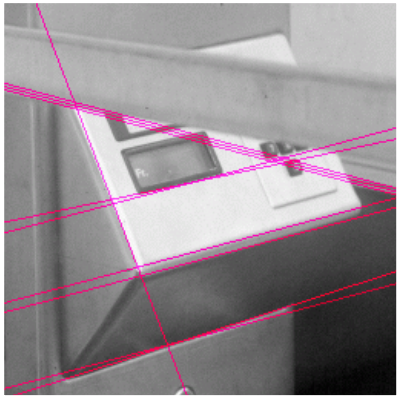
Gradient



Edge (Threshold)

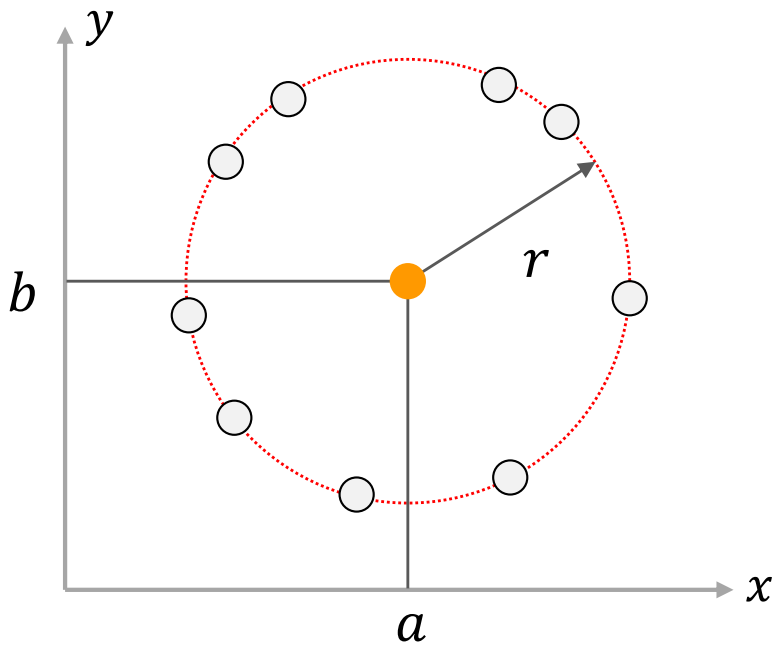


Hough Transform $A(\rho, \theta)$



Detected Lines

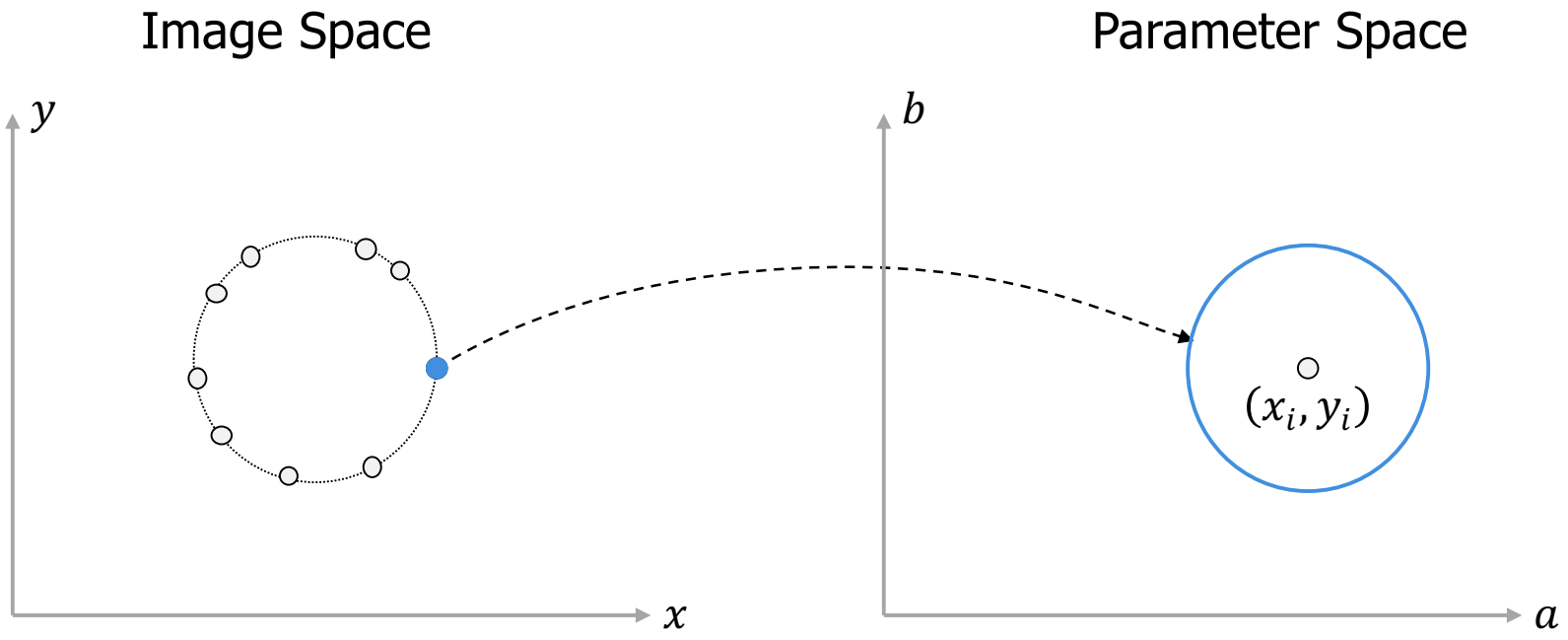
Hough Transform: Circle Detection



Equation of Circle: $(x_i - a)^2 + (y_i - b)^2 = r^2$

Hough Transform: Circle Detection

If radius r **is known**: Accumulator Array: $A(a, b)$



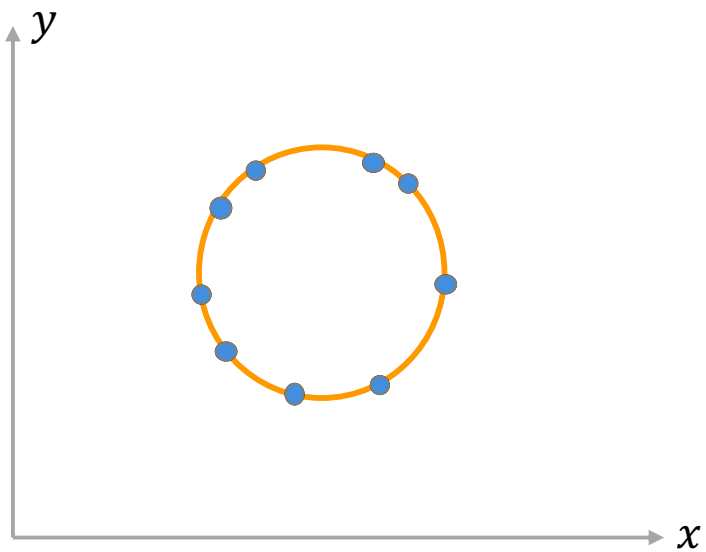
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

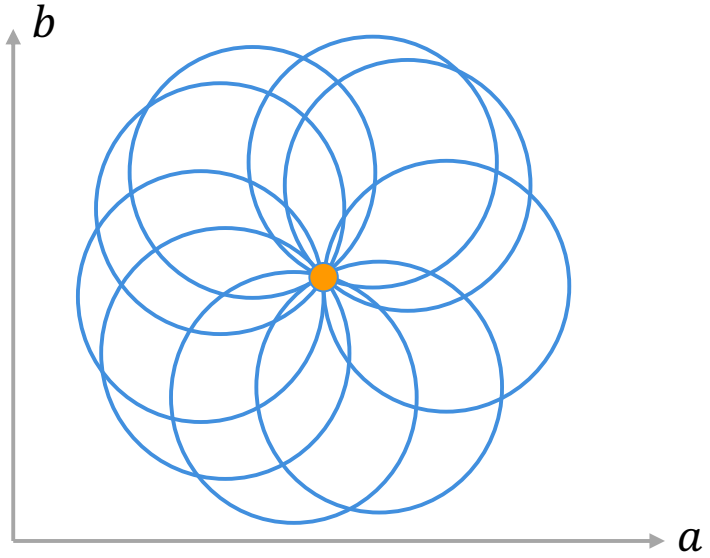
If radius r **is known**: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

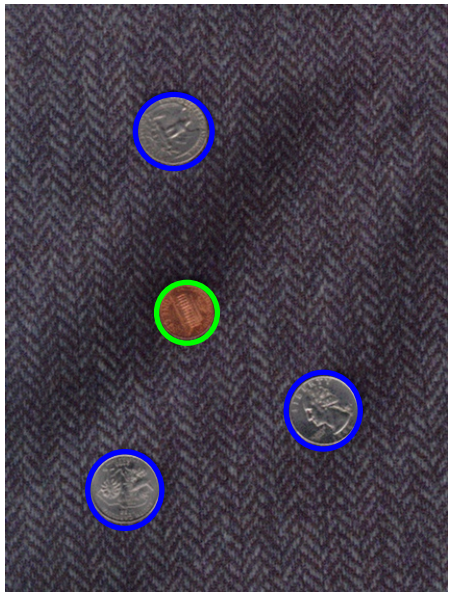
Parameter Space



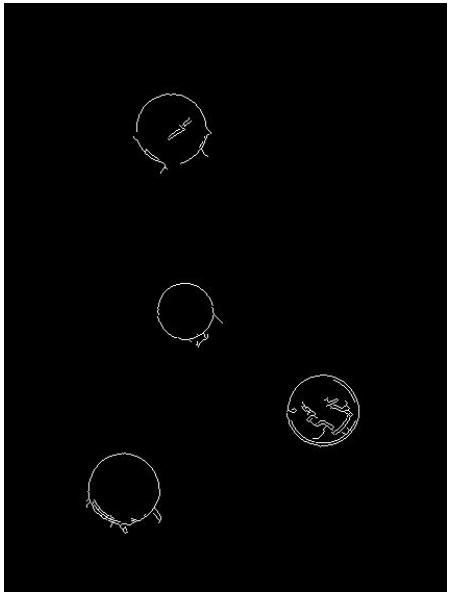
$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Circle Detection Results

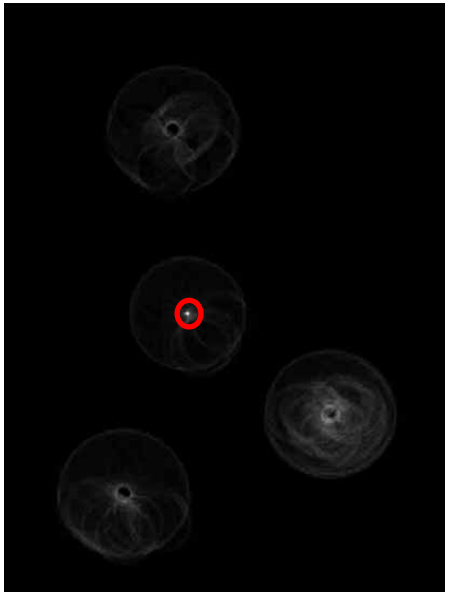
Penny $(r = r_1)$ Quarter $(r = r_2)$



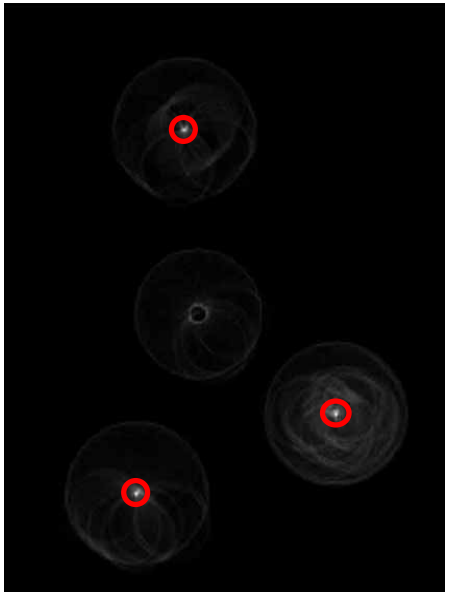
Original Image



Edge (Threshold)



Hough Transform
$A_1(a, b)$



Hough Transform
$A_2(a, b)$

Summary

- **Derivative Operators**: The Hough transform: moving from edge detection to detection of lines and circle
- Essential concepts in this lecture:
  - The essential idea of the Hough transform
  - Some of the design tradeoffs
  - How to apply it to lines and circles