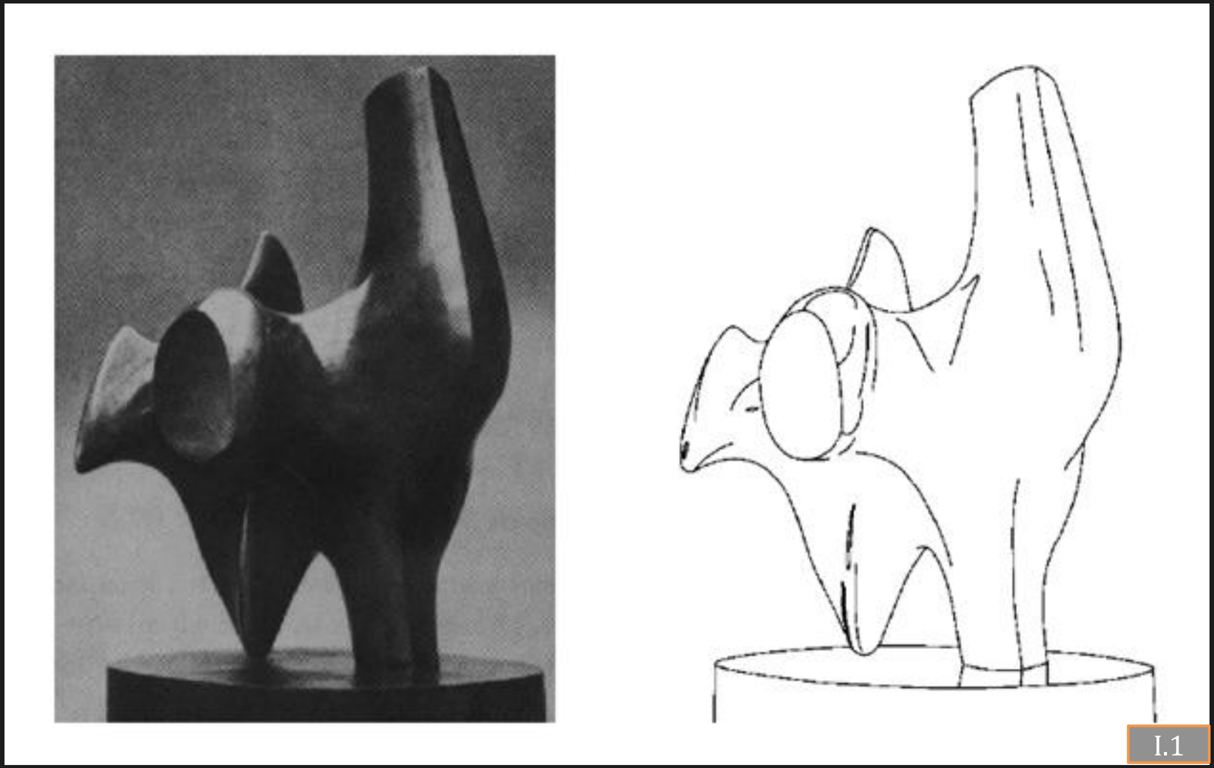
Edge And Boundary Detection

Computer Vision: AI3064

What Are Edges?

Rapid changes in image intensity within small region



I.1

Edge and Boundary Detection

Convert a 2D Image into a Set of Curves

Finding Object Boundaries from Edge Pixels

Topics:

(1) Theory of Edge Detection

(2) Edge Detection Using Gradients

(3) Edge Detection Using Laplacian

(4) Preprocessing Edge Images

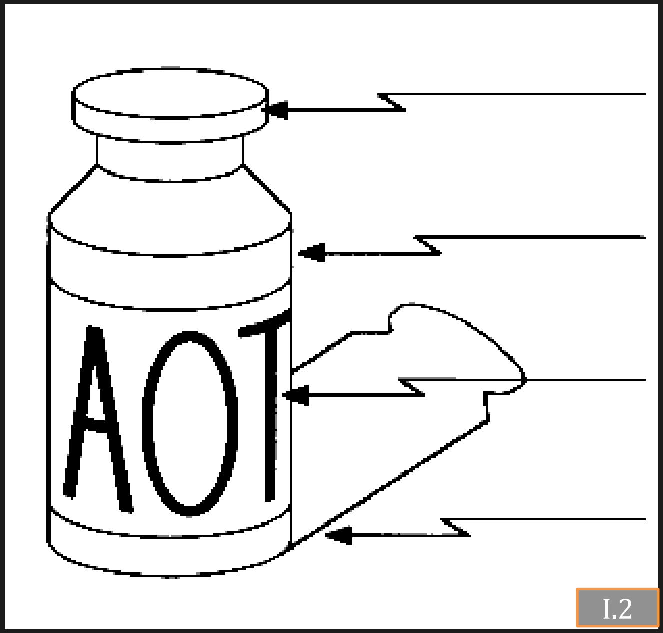
(5) Fitting Lines and Curves to Edges

(6) Active Contours (also called Snakes)

(7) The Hough Transform

Causes of Edges

Edges are caused by a variety of factors



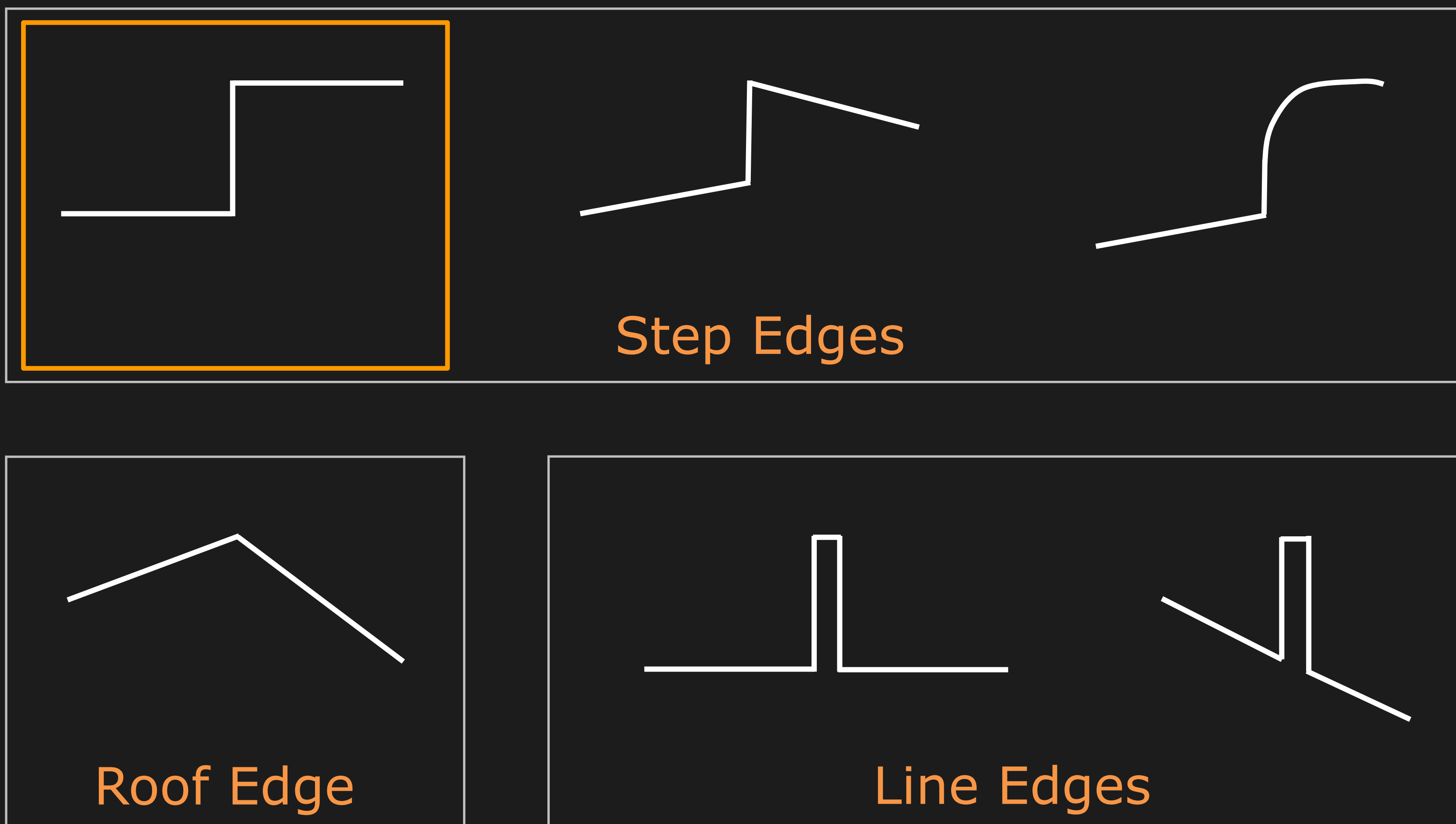
Surface Normal Discontinuity

Depth Discontinuity

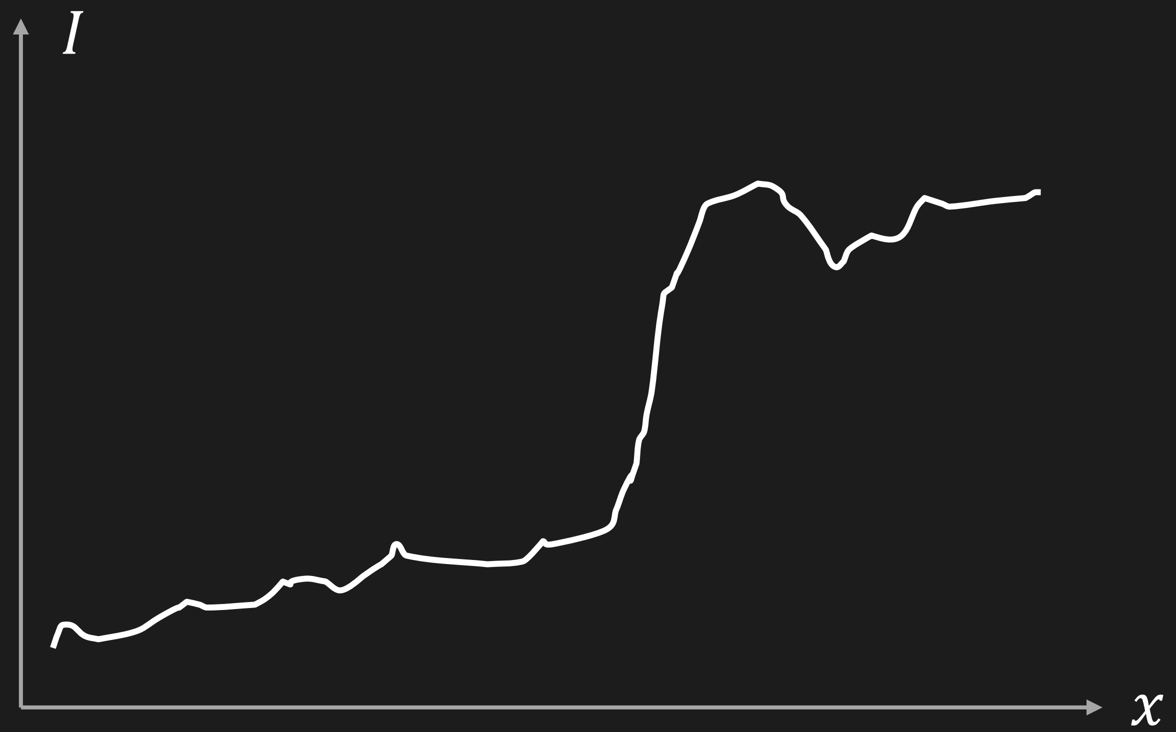
Surface Color Discontinuity

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)

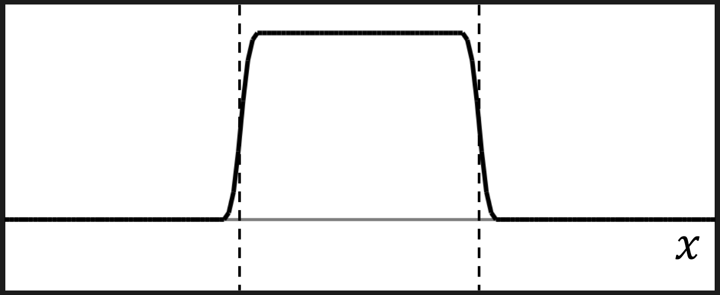
Crucial Requirements:

- High Detection Rate
- Good Localization
- Robust to Noise

1D Edge Detection

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

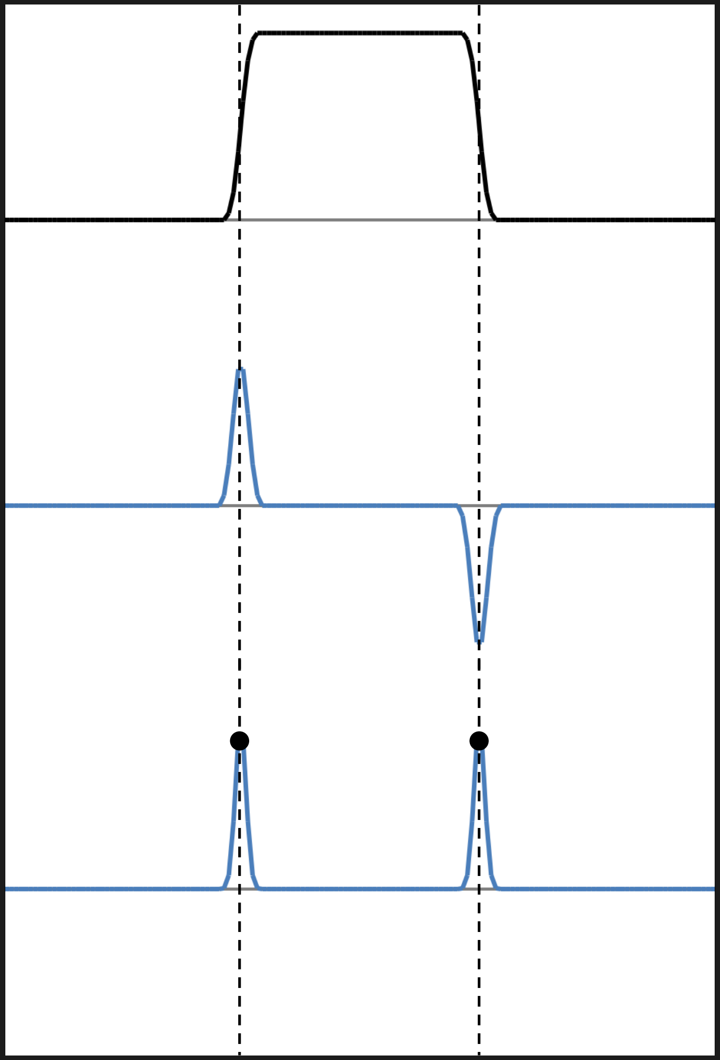
$f(x)$:



Edge Edge

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

Edge Detection Using 1st Derivative



$f(x)$

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

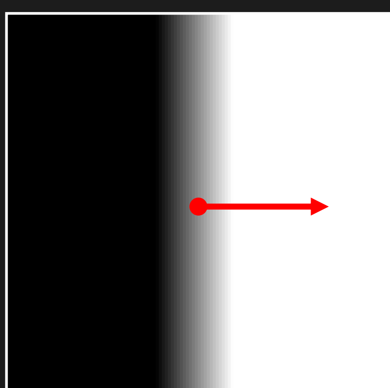
Provides Both Location and Strength of an Edge

Gradient (∇)

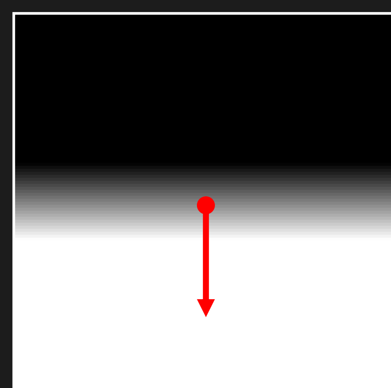
Gradient (Partial Derivative) Represents the Direction of Most Rapid Change in Intensity

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

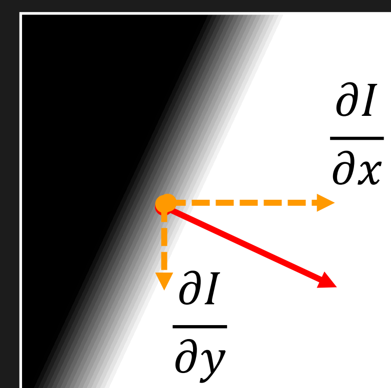
Pronounced as "Del I"



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



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

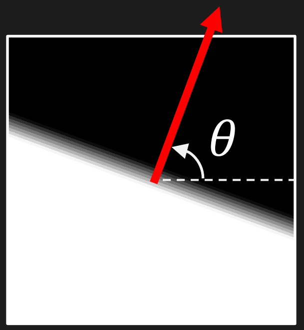


$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

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:

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

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

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

ε

Can be implemented as Convolution!

$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad \frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array}$$

Note: Convolution flips have been applied

Comparing Gradient (∇) Operators

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

Good Localization
Noise Sensitive
Poor Detection

Poor Localization
Less Noise Sensitive
Good Detection

Gradient (∇) Using Sobel Filter

Image (I)

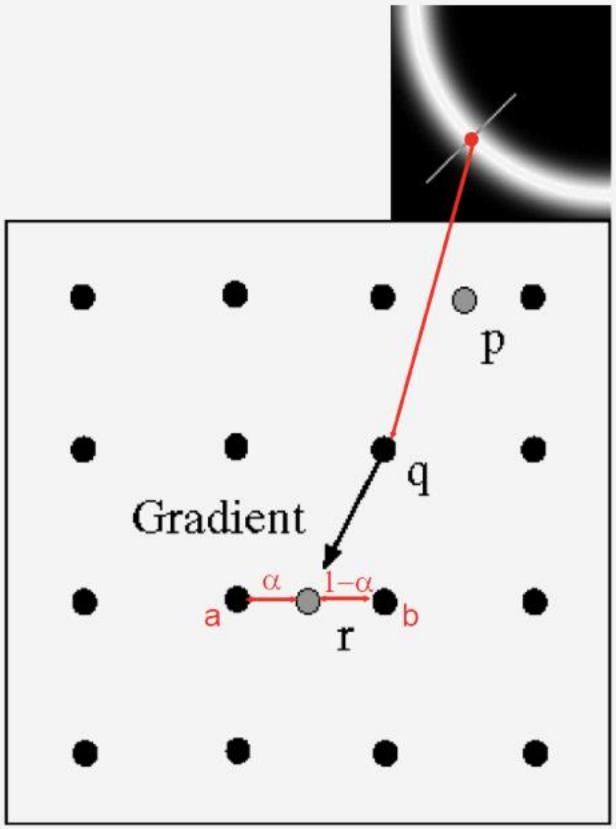
$\partial I / \partial x$

$\partial I / \partial y$

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)

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

Sobel Edge Detector

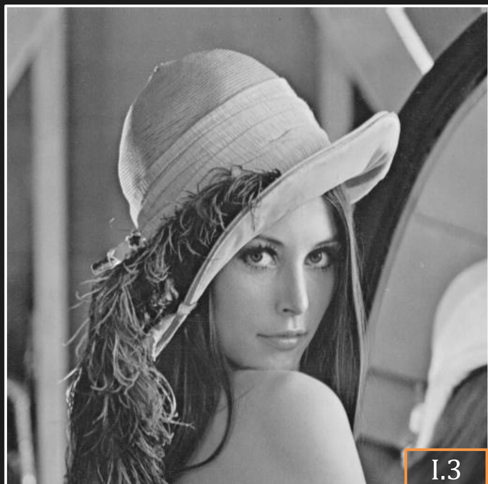


Image (I)

$\partial I / \partial x$

$\partial I / \partial y$

Gradient Magnitude

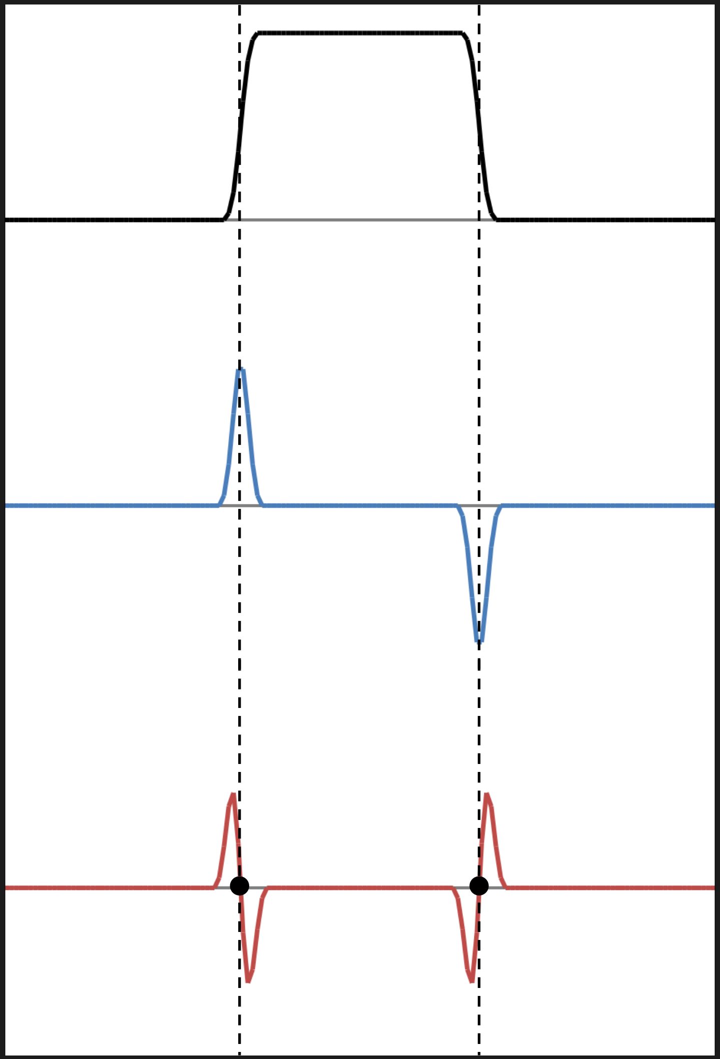
Thresholded Edge

Edge Detection Using 2nd Derivative

$f(x)$

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

Second Derivative: $\frac{\partial^2 f}{\partial x^2}$



Local Extrema Indicate Edges

Zero-Crossings Indicate Edges

Provides Only the Location of an Edge

Laplacian (∇^2) as Edge Detector

Laplacian: Sum of Pure Second Derivatives

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

Pronounced as "Del Square I"

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

[Marr1980]

Discrete Laplacian(∇^2) Operator

Finite difference approximations:

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

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

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

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

Convolution Mask :

$$\nabla^2 \approx \frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array}$$

OR

$$\nabla^2 \approx \frac{1}{6\varepsilon^2}\begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array}$$

(More Accurate)

Laplacian Edge Detector

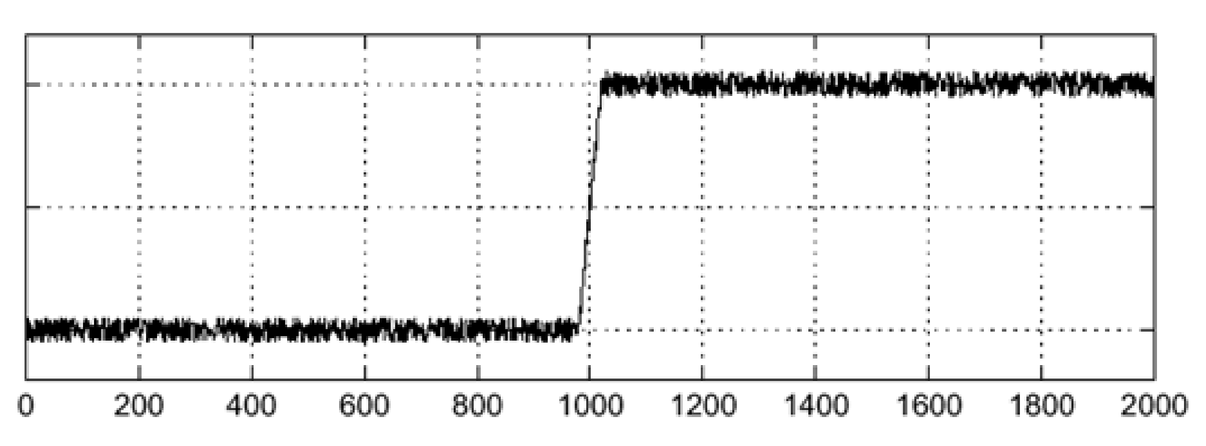
Image (I)

Laplacian
(0 maps to 128)

Laplacian
"Zero Crossings"

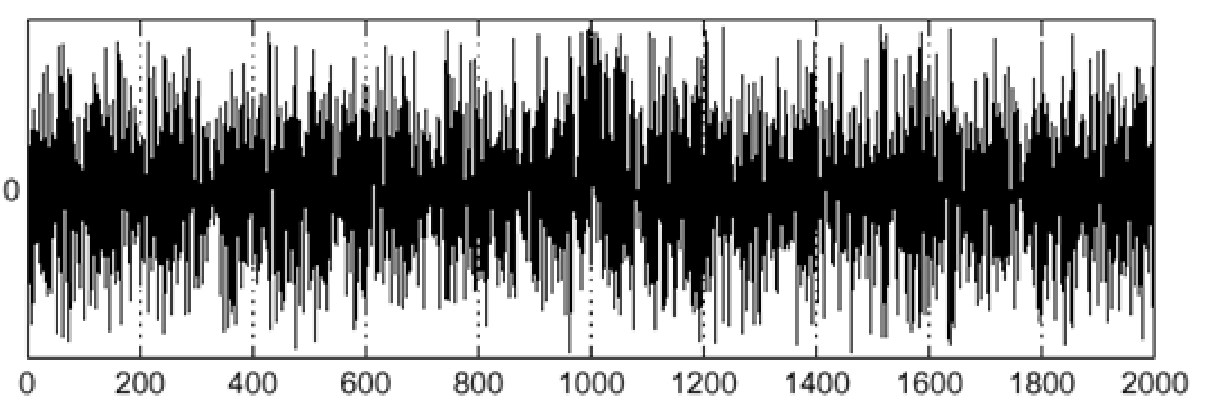
Effects of Noise

$f(x)$



$\nabla f(x)$

(Gradient)



Where is the edge??

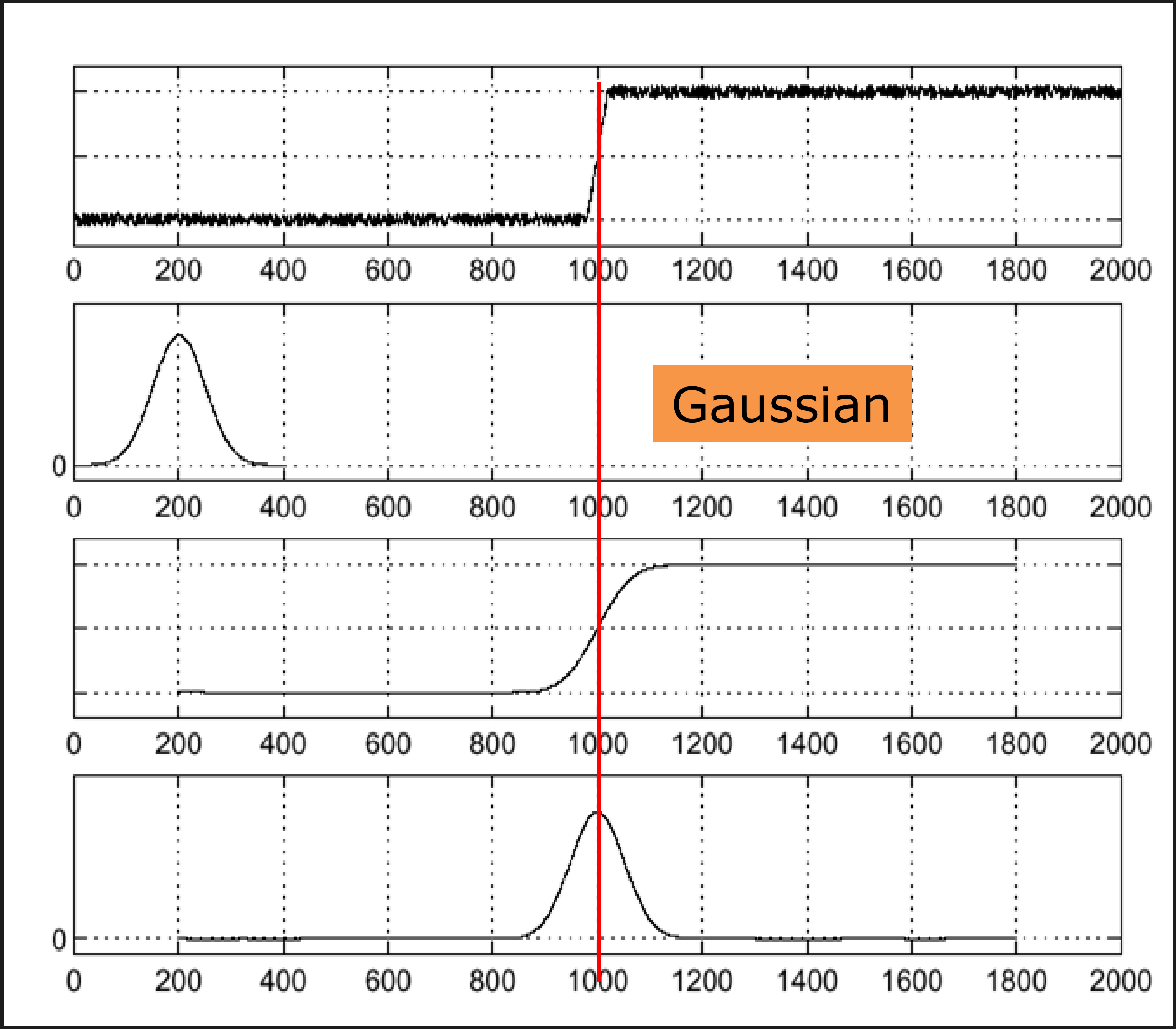
Solution: Gaussian Smooth First

f

n_σ

$n_\sigma * f$

$\nabla(n_\sigma * f)$



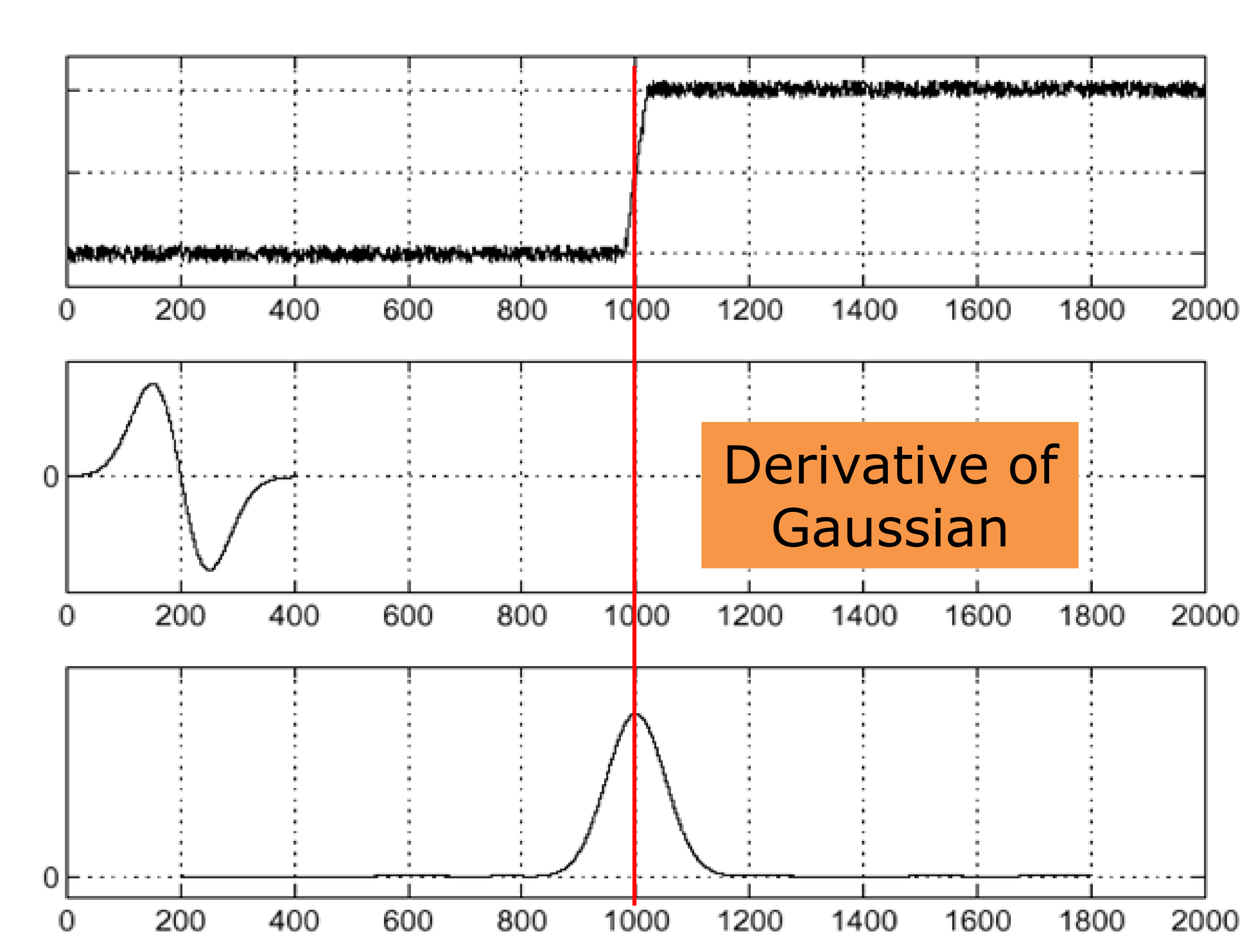
Derivative of Gaussian ($\nabla(n_\sigma)$)

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

f

$\nabla(n_\sigma)$

$\nabla(n_\sigma) * f$



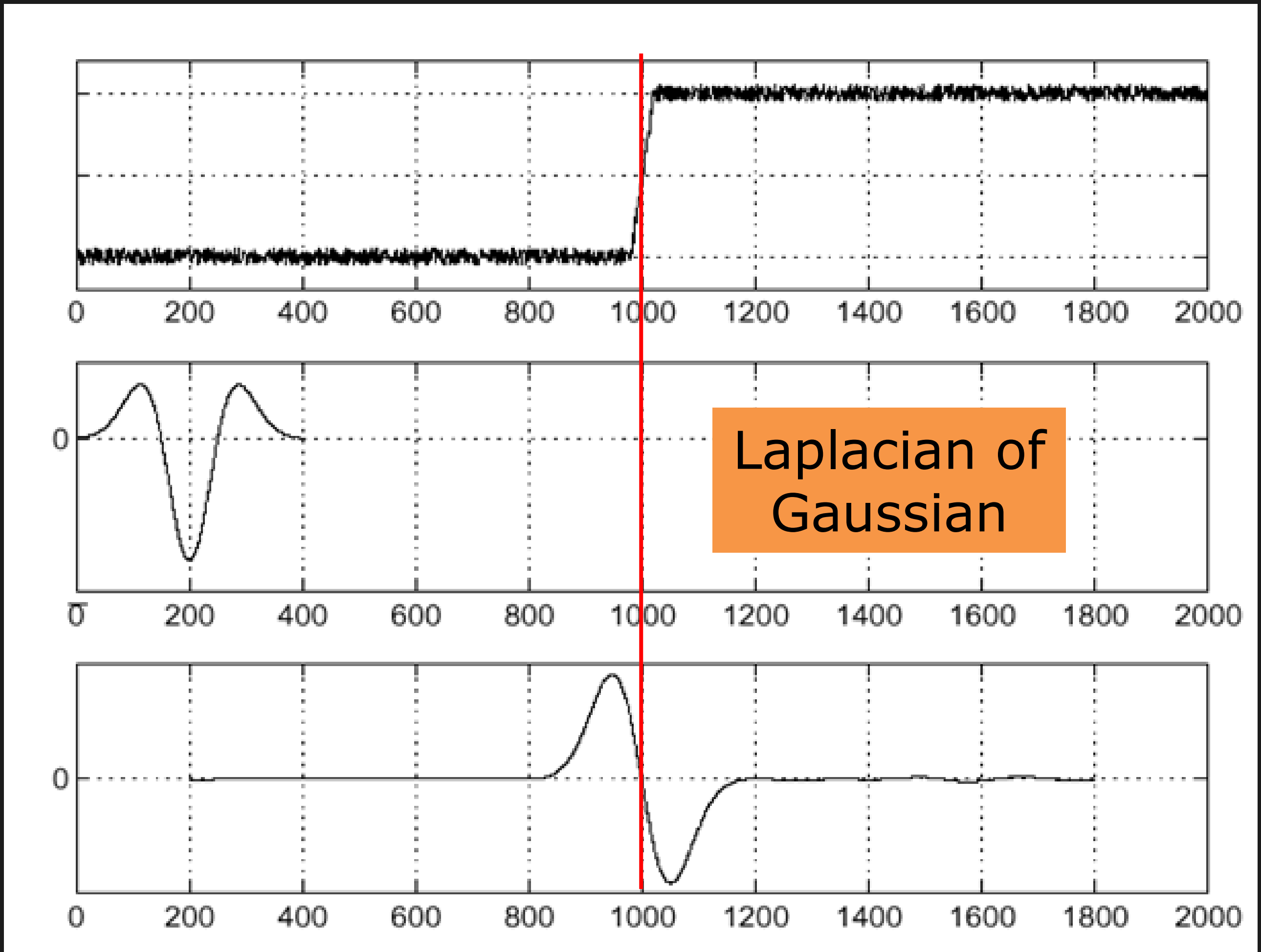
Laplacian of Gaussian ($\nabla^2 n_\sigma$ or $\nabla^2 G$)

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

f

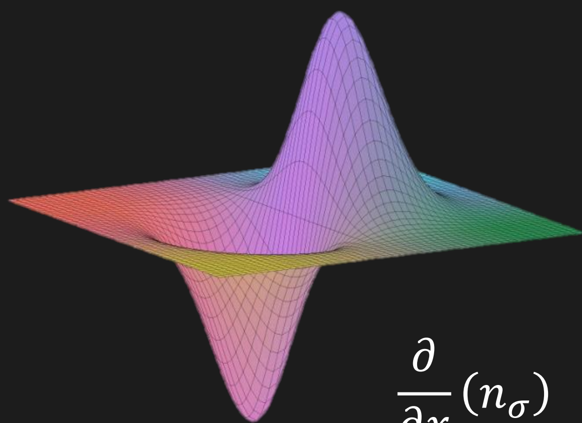
$\nabla^2(n_\sigma)$

$\nabla^2(n_\sigma) * f$

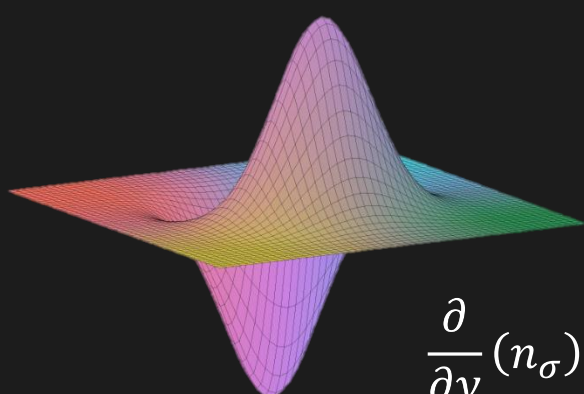


Gradient vs. Laplacian

Derivative of Gaussian (∇G)

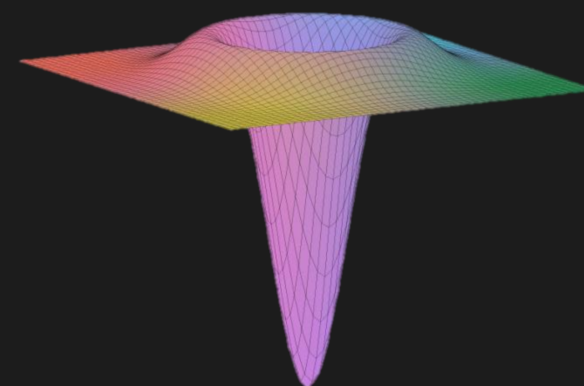


$$\frac{\partial}{\partial x}(n_\sigma)$$



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

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

Gradient	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?

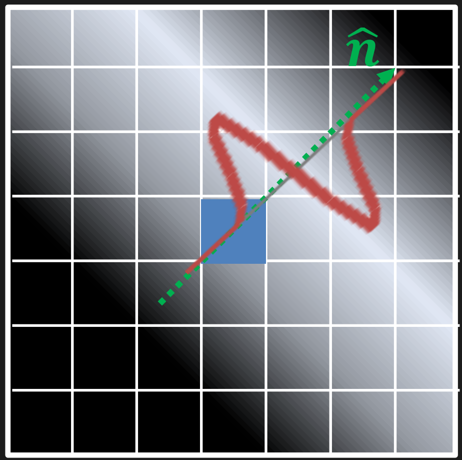
Canny Edge Detector

- 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

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



$|\nabla n_\sigma * I|$

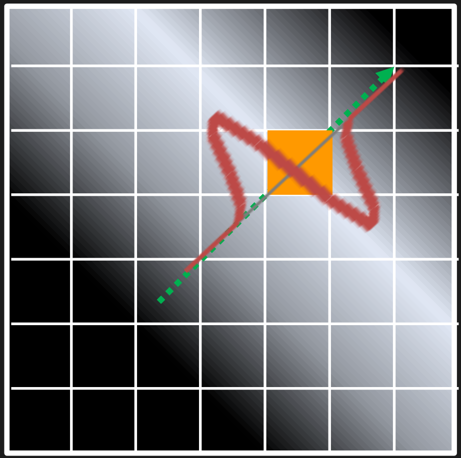
http://justin-liang.com/tutorials/canny

[Canny1986]

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:

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



$|\nabla n_\sigma * I|$

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

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

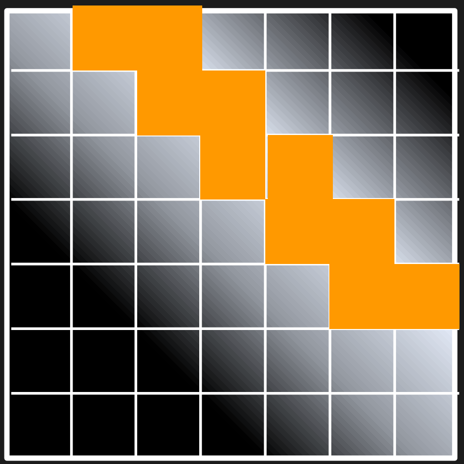
- Find Zero Crossings in Laplacian to find the edge location

[Canny1986]

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:

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



$|\nabla n_\sigma * I|$

- 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

[Canny1986]

Canny Edge Detector Results

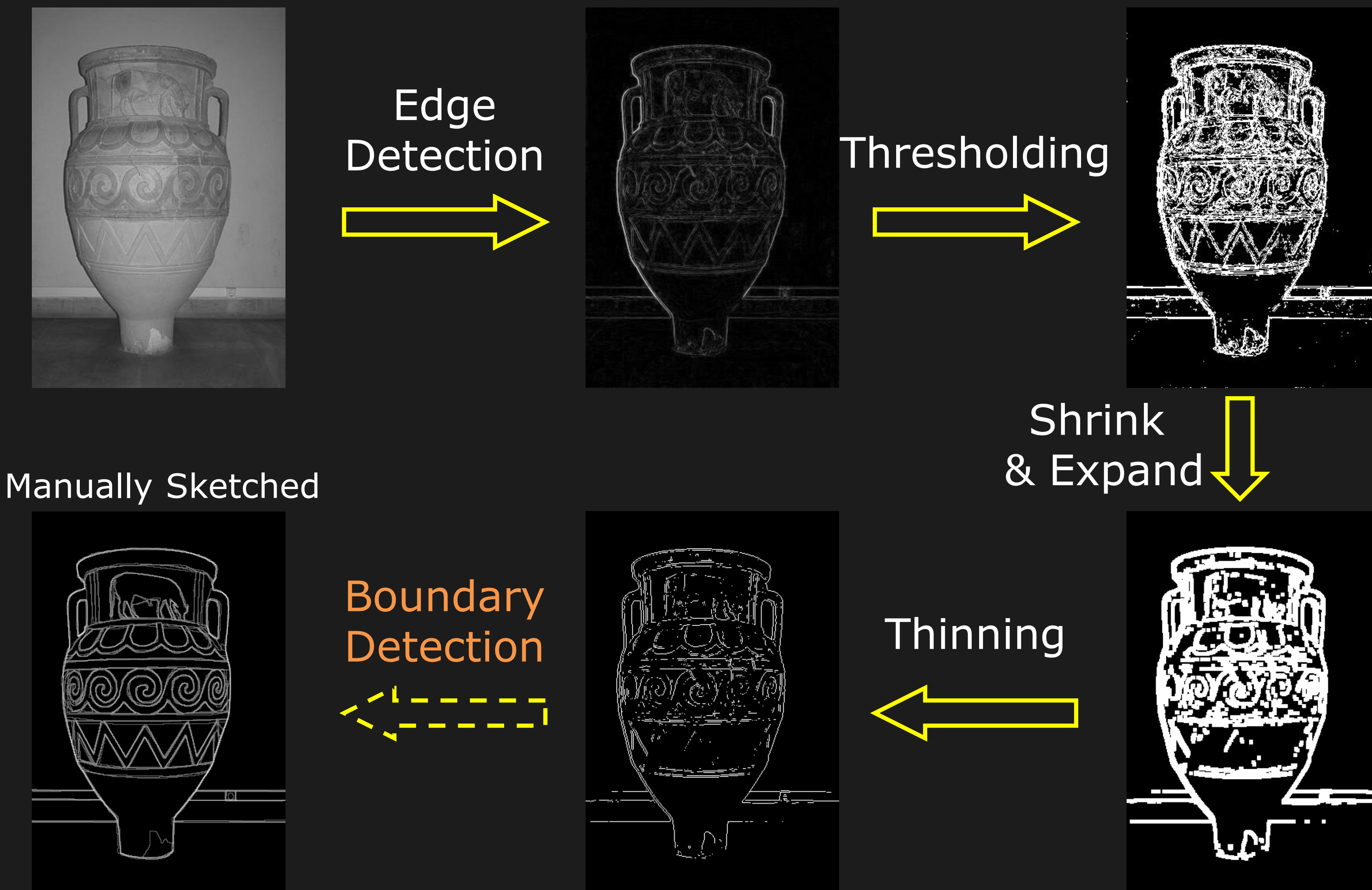
Image

$\sigma = 1$

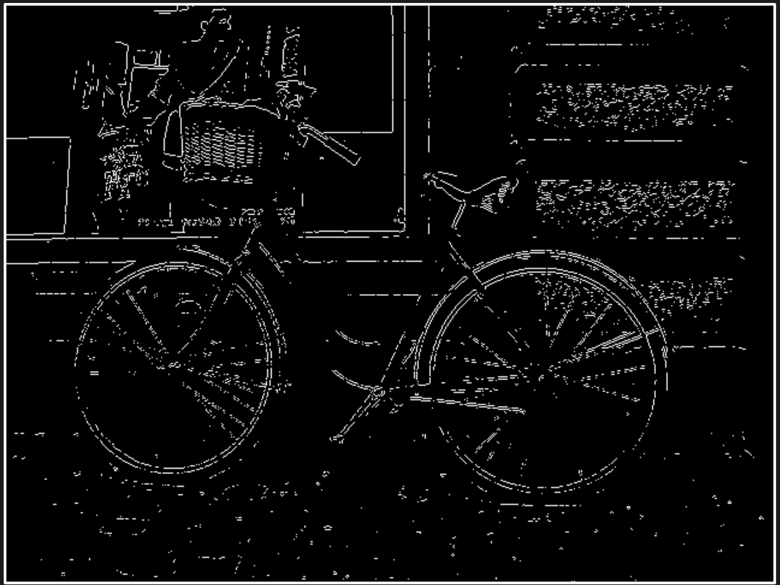
$\sigma = 2$

$\sigma = 4$

Preprocessing Edge Images



Difficulties for the Fitting Approach



- Extraneous Data: What points to fit to?
- Incomplete Data: Only part of the model is visible.
- Noise

Solution: The VOTING approach! (Hough Transform)

The Hough Transform

Elegant method for Direct Object Recognition

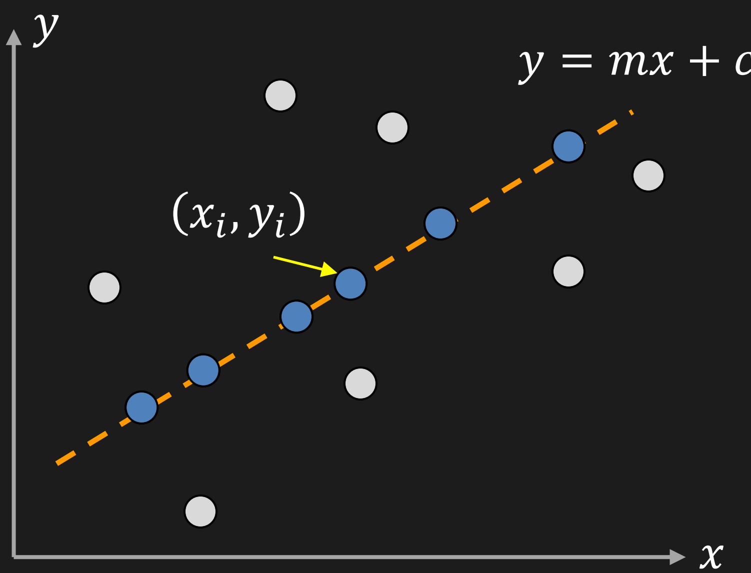
- Robust to disconnected edges
- Complete object need not be visible
- Relatively robust to noise

Hough Transform: Line Detection

Given: Edge Points (x_i, y_i)

Task: Detect line

$$y = mx + c$$



Consider point (x_i, y_i)

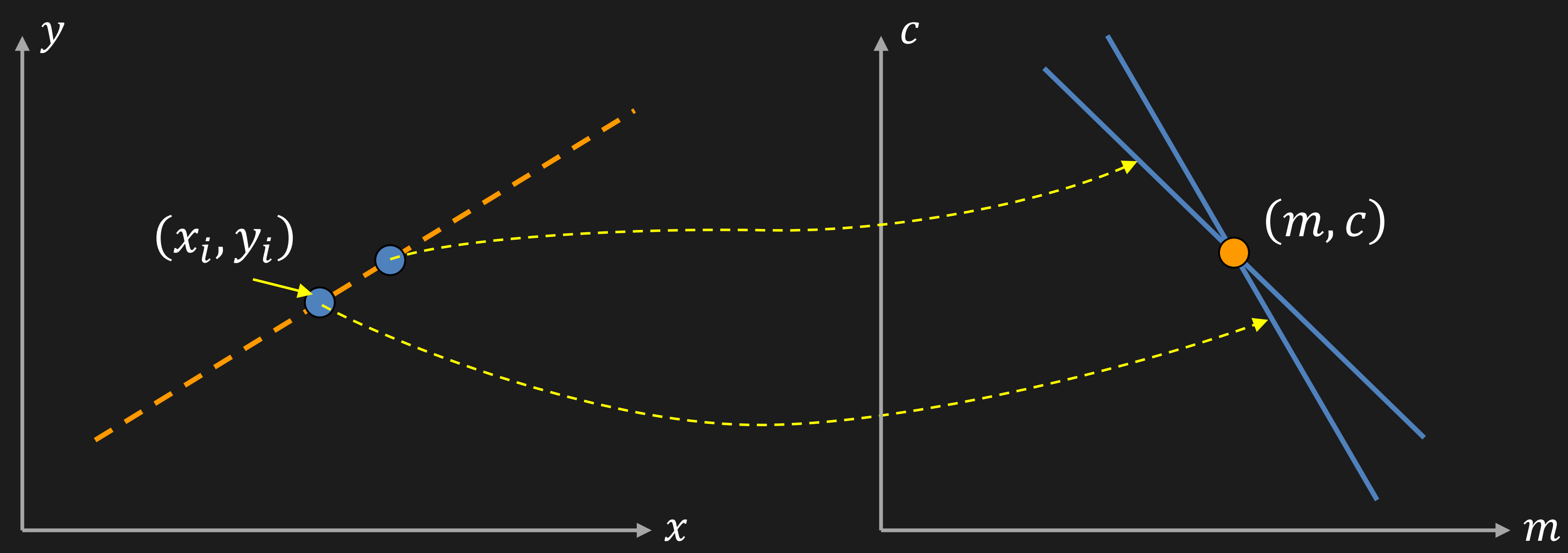
$$y_i = mx_i + c \quad \Longleftrightarrow \quad c = -x_i m + y_i$$

[Hough 1962]

Hough Transform: Concept

Image Space

Parameter Space

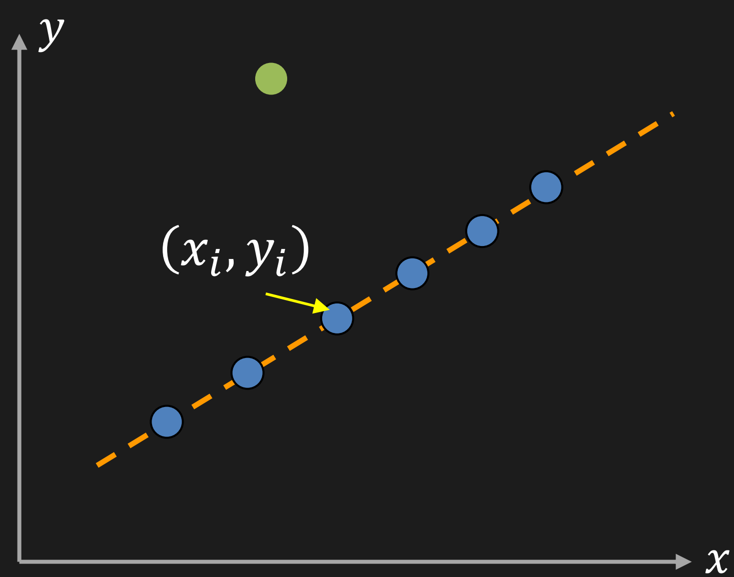


$y_i = m x_i + c$

$c = -x_i m + y_i$

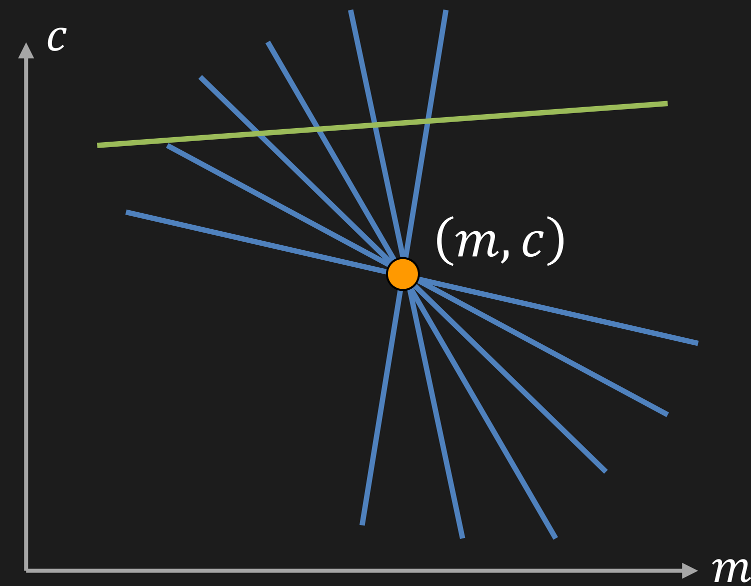
Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

Parameter Space



$$c = -x_i m + y_i$$

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)

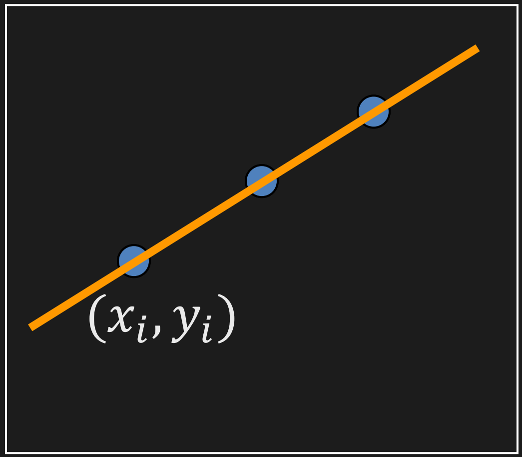
Step 4. For each edge point (x_i, y_i),

$$A(m, c) = A(m, c) + 1$$

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

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

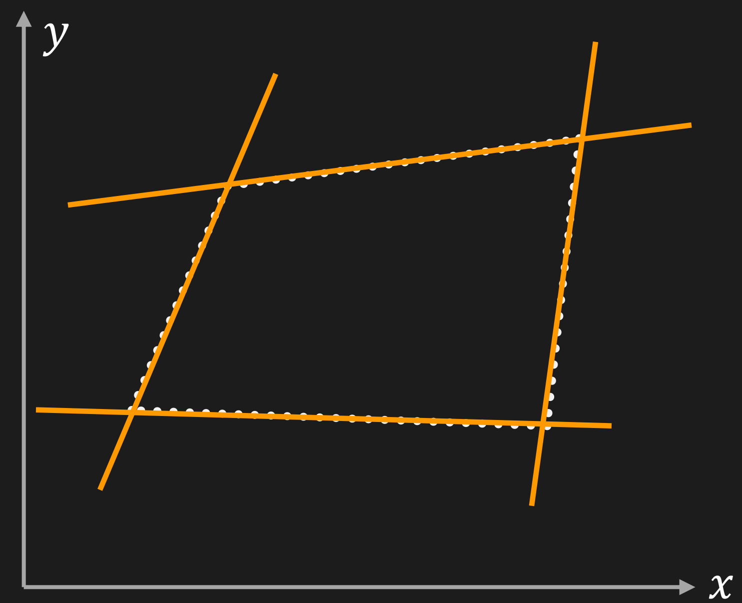
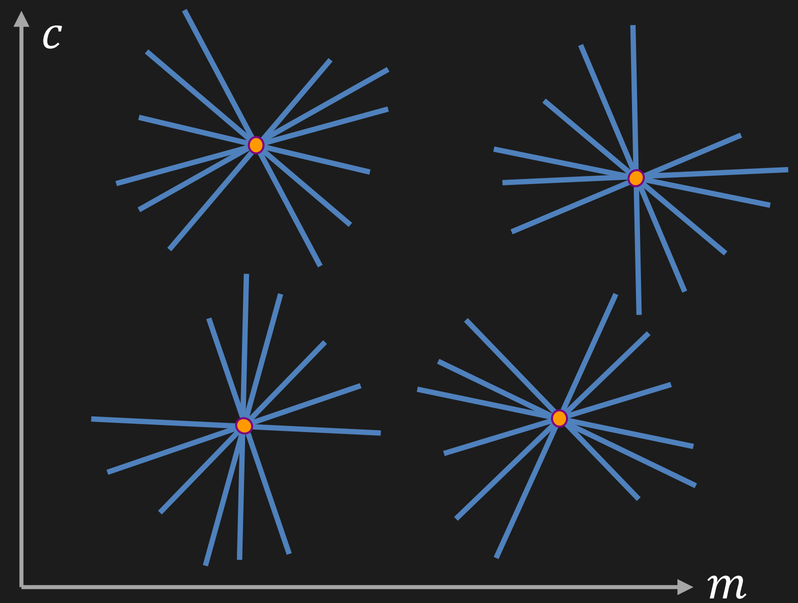


Image Space



Parameter Space

Better Parameterization

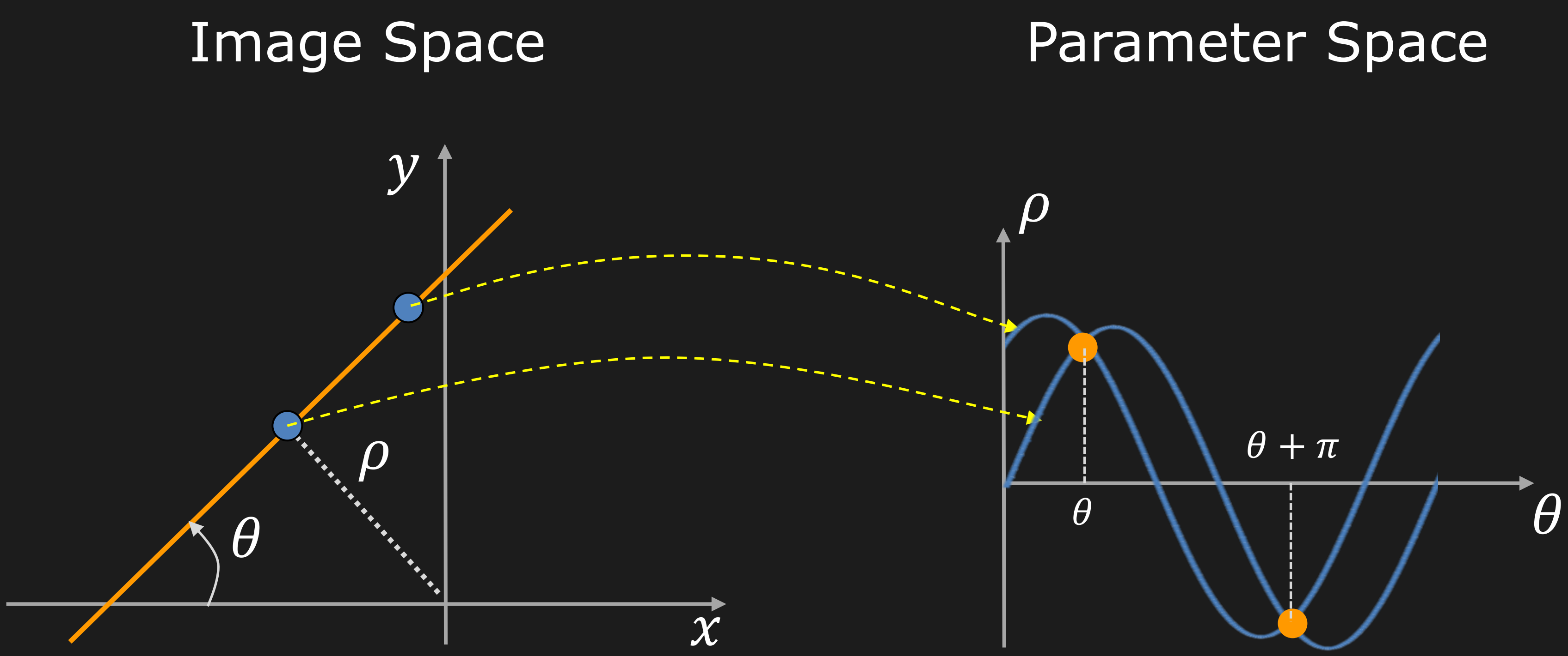
Issue: Slope of the line $-\infty \leq m \leq \infty$

- 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

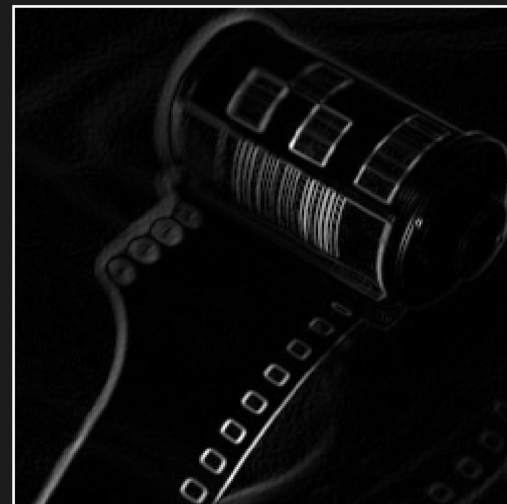
- How big should the accumulator cells be?
  - Too big, and different lines may be merged
  - Too small, and noise causes lines to be missed

- How many lines?
  – Count the peaks in the accumulator array

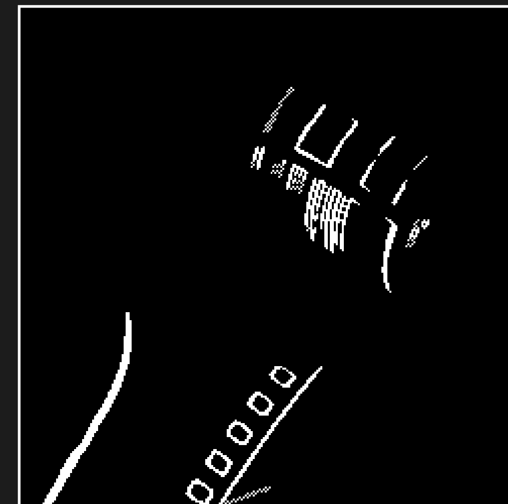
- Handling inaccurate edge locations:
  - Increment patch in accumulator rather than single point

Line Detection Results

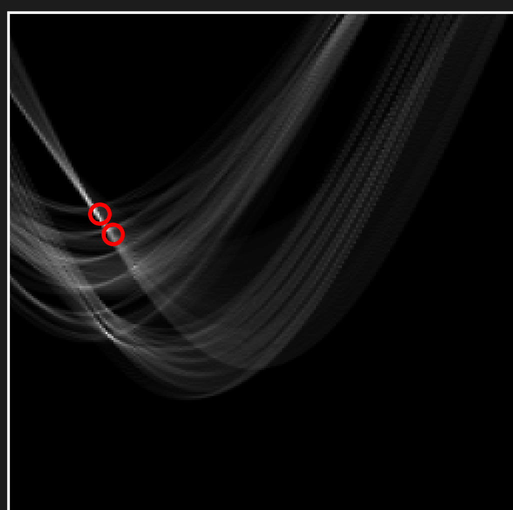
Original Image



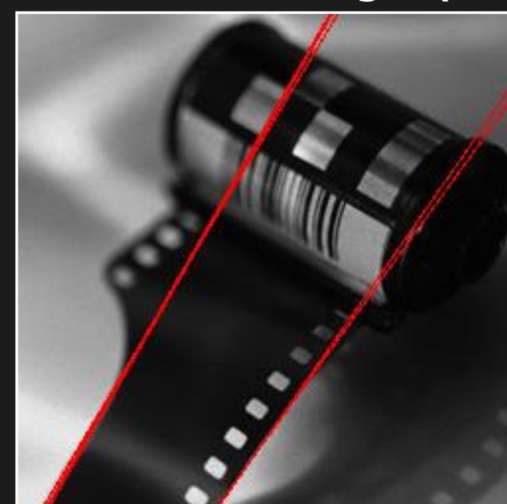
Gradient



Edge (Threshold)



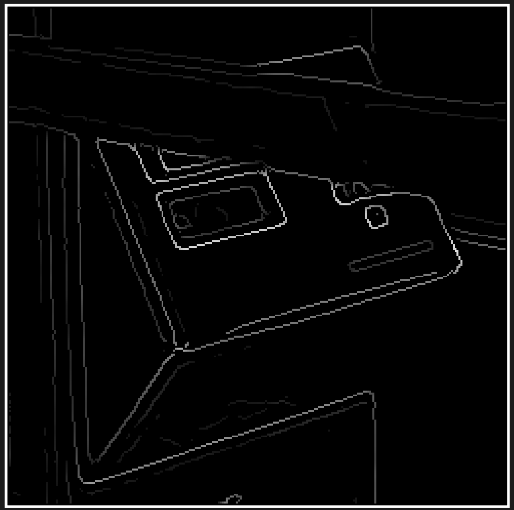
Hough Transform $A(\rho, \theta)$



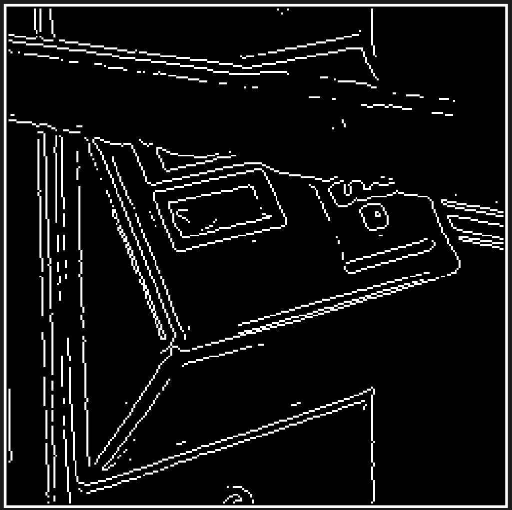
Detected Lines

Line Detection Results

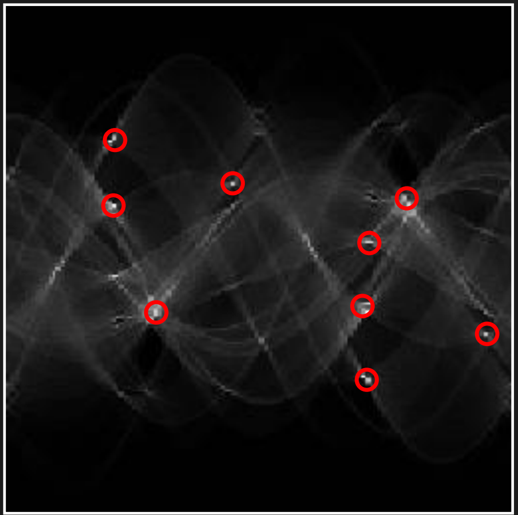
Original Image



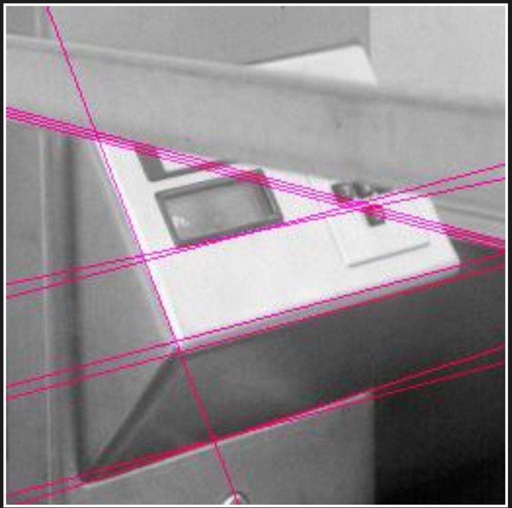
Gradient



Edge (Threshold)

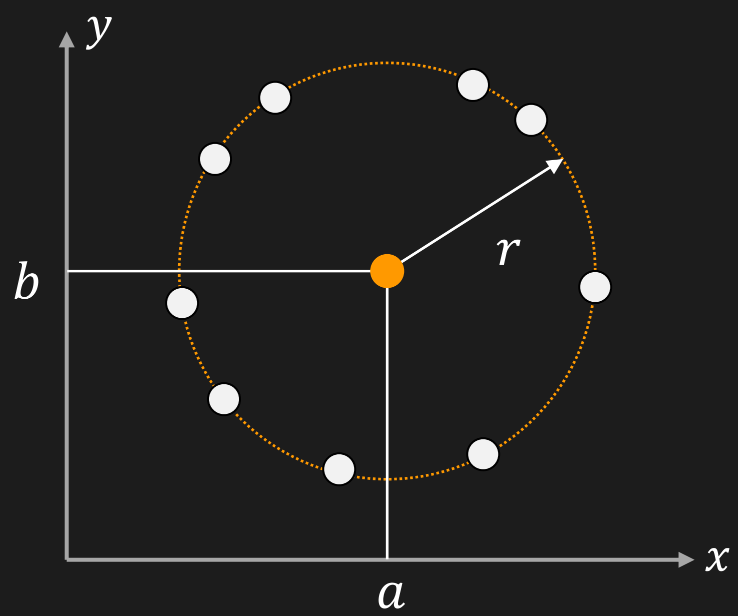


Hough Transform $A(\rho, \theta)$



Detected Lines

Hough Transform: Circle Detection

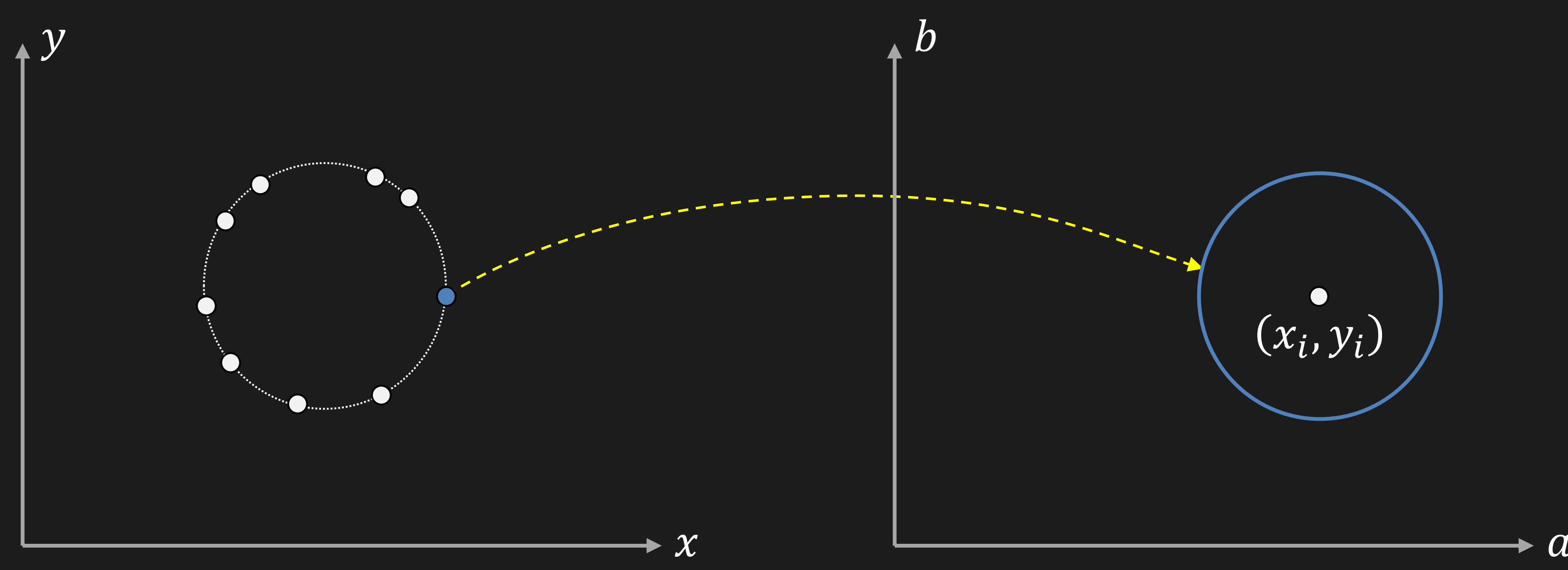


Equation of Circle: $(x_i - a)^2 + (y_i - b)^2 = r^2$

Hough Transform: Circle Detection

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

Image Space | Parameter Space



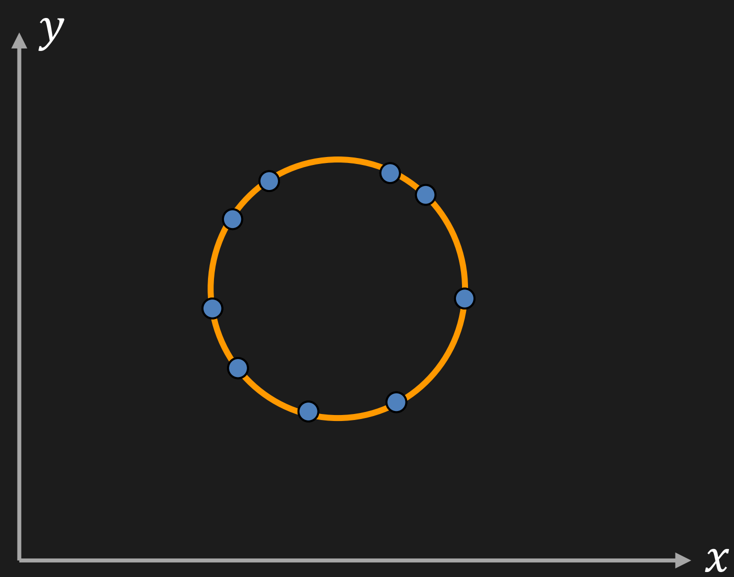
$(x_i - a)^2 + (y_i - b)^2 = r^2$ | $(a - x_i)^2 + (b - y_i)^2 = r^2$

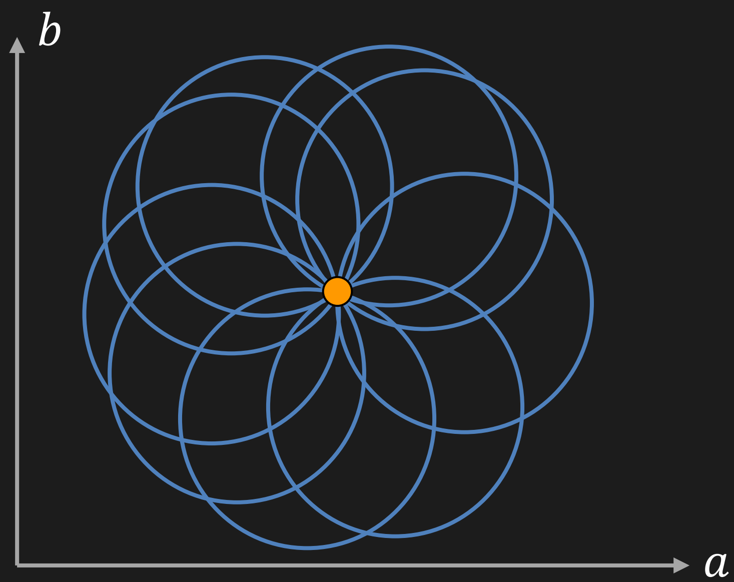
Hough Transform: Circle Detection

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

Image Space

Parameter Space



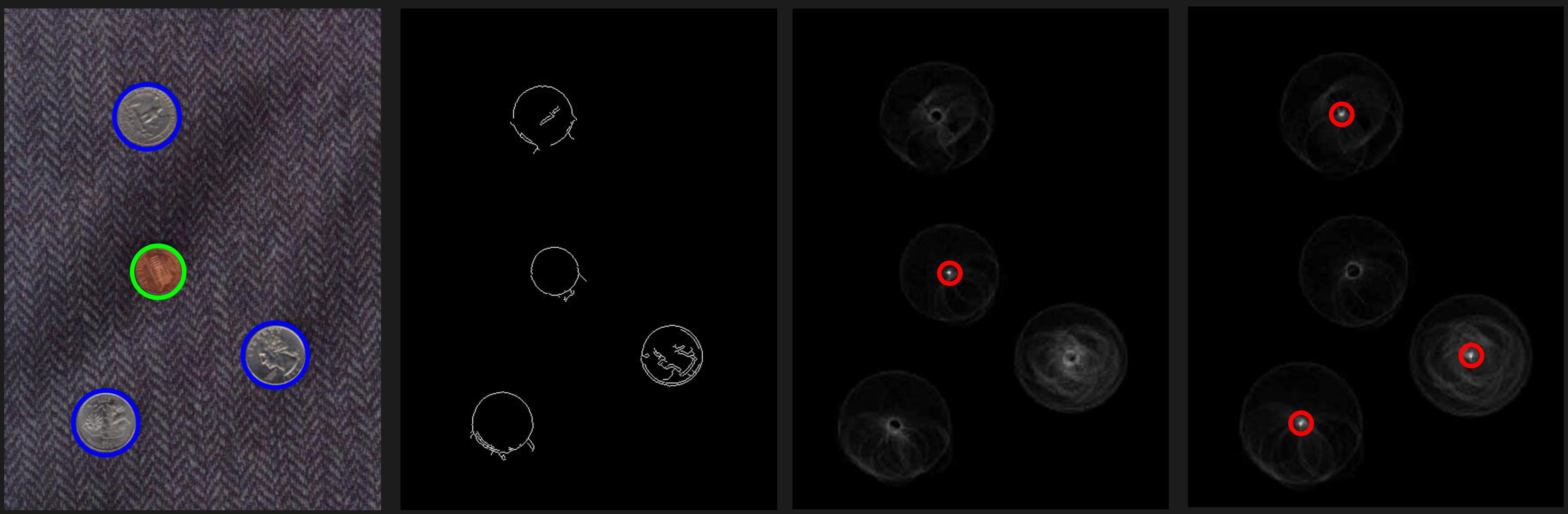


$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

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

Circle Detection Results

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



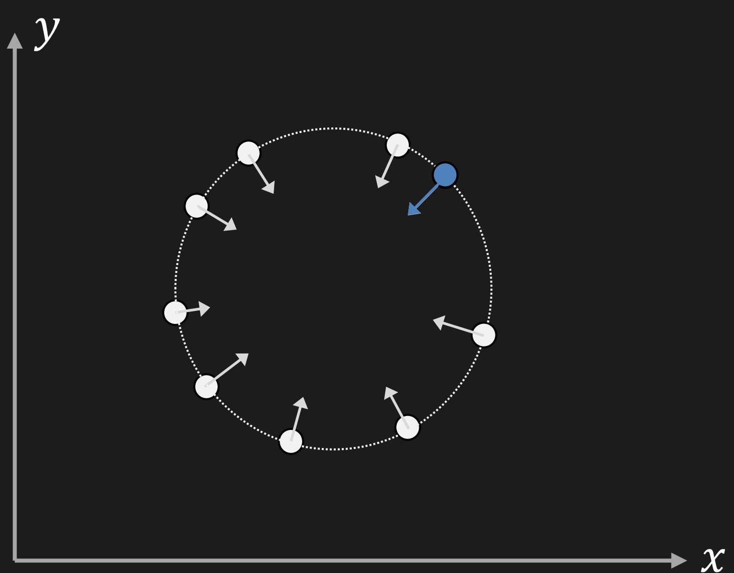
Original Image | Edge (Threshold) | Hough Transform $A_1(a,b)$ | Hough Transform $A_2(a,b)$

I.5

Using Gradient Information

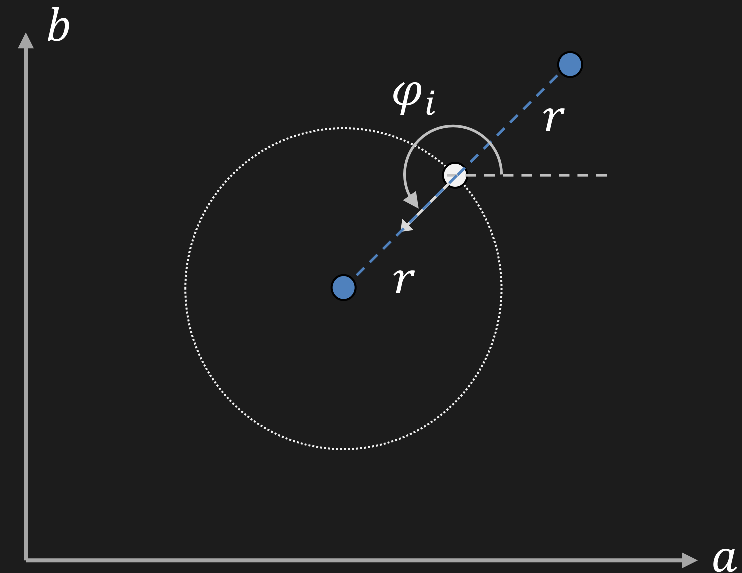
Given: Edge Location (x_i, y_i), Edge Direction φ_i and Radius r

Image Space



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

Parameter Space



$$a = x_i \pm r\cos\varphi_i$$

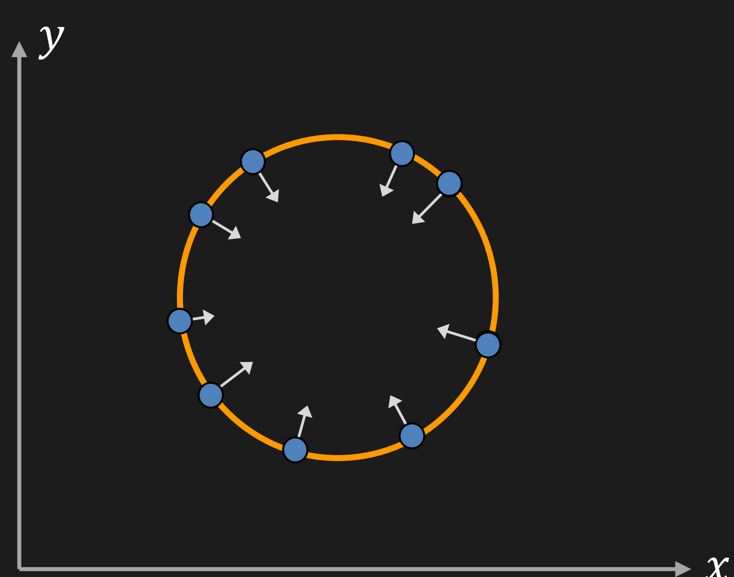
$$b = y_i \pm r\sin\varphi_i$$

Need to increment only TWO points in $A(a, b)$

Using Gradient Information

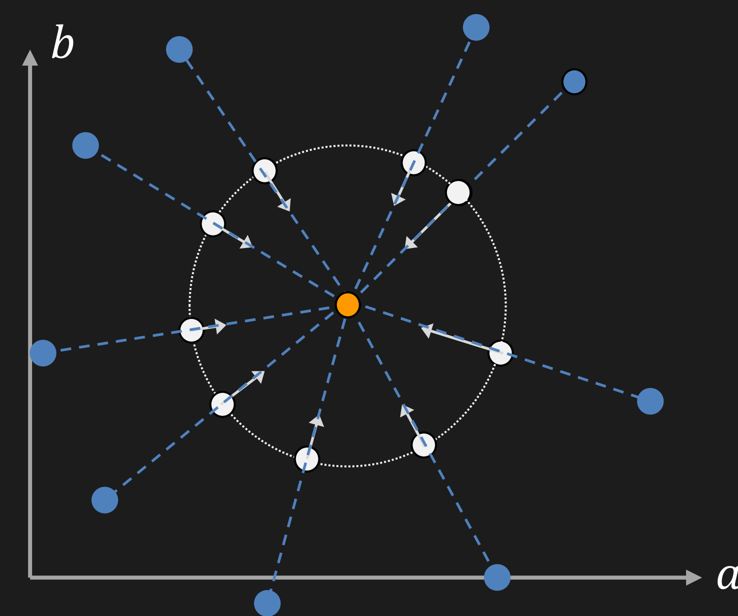
Given: Edge Location (x_i, y_i), Edge Direction φ_i and Radius r

Image Space



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

Parameter Space



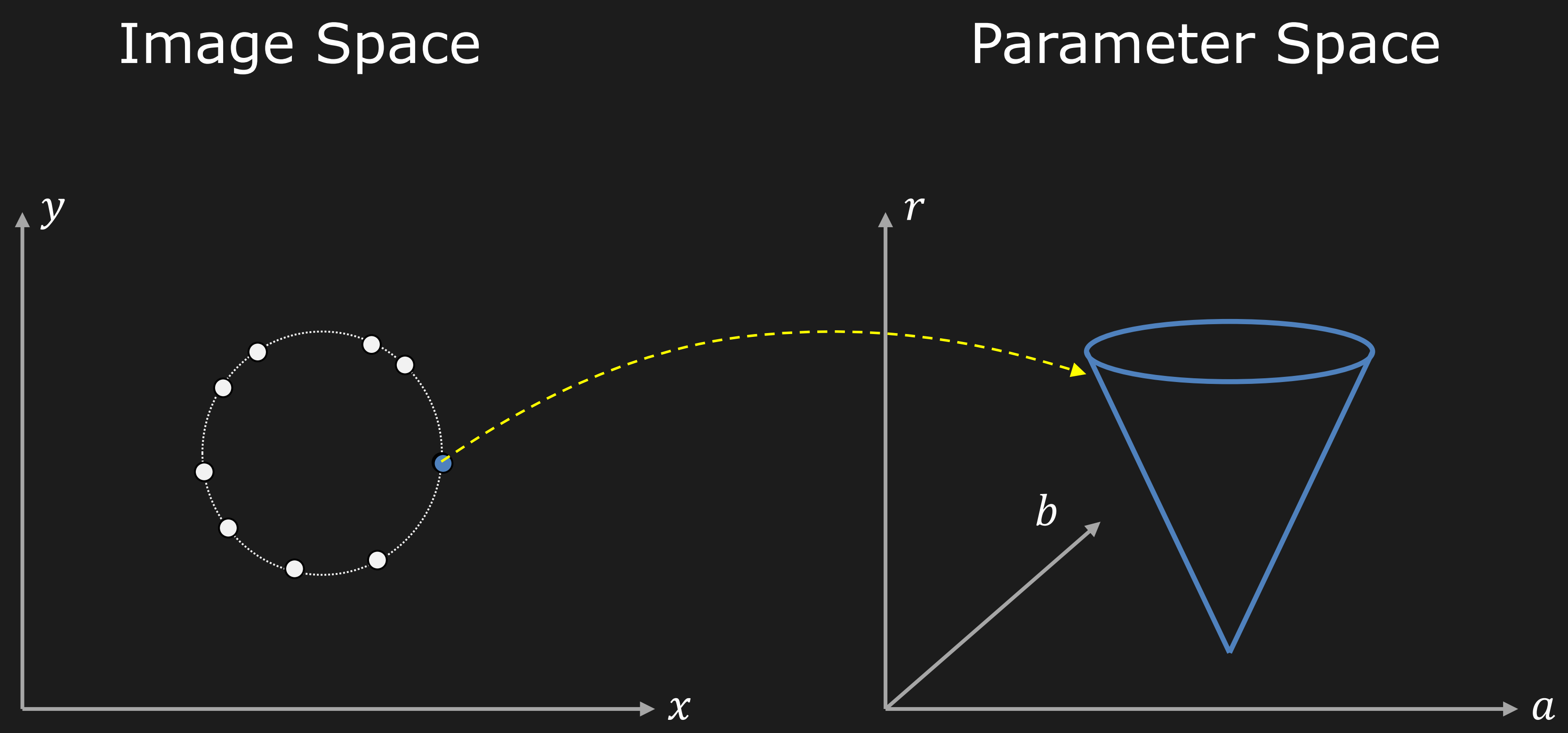
$$a = x_i \pm r\cos\varphi_i$$

$$b = y_i \pm r\sin\varphi_i$$

Need to increment only TWO points in $A(a, b)$

Hough Transform: Circle Detection

If radius r is NOT known: Accumulator Array: $A(a, b, r)$

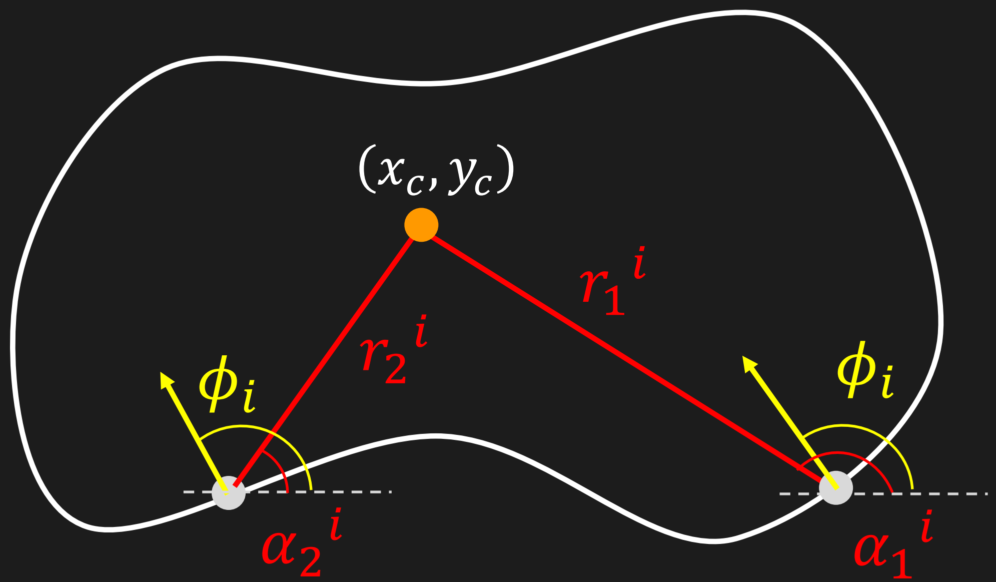


$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

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

Generalized Hough Transform

Find shapes that cannot be described by Equations



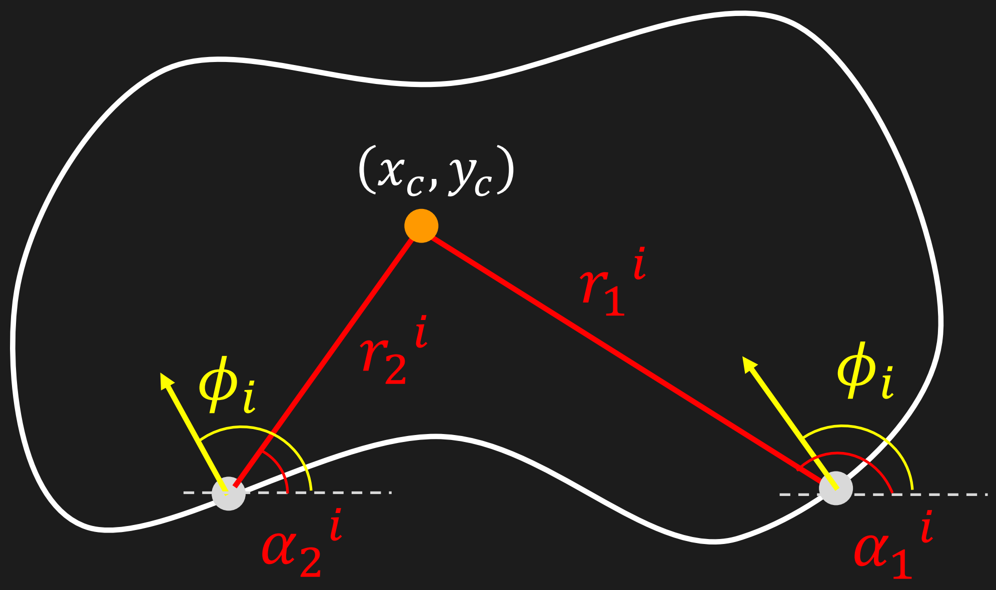
Reference point: (x_c, y_c)

Edge direction: ϕ_i $\quad 0 \leq \phi_i < 2\pi$

Edge location: $\vec{r}_k{}^i = (r_k{}^i, \alpha_k{}^i)$

[Ballard 1981]

Hough Model



ϕ-Table:

Edge Direction	Edge location
ϕ_1	$\vec{r}_1{}^1 = ({r_1}^1, {\alpha_1}^1), \vec{r}_2{}^1 = ({r_2}^1, {\alpha_2}^1)$
ϕ_2	$\vec{r}_1{}^2 = ({r_1}^2, {\alpha_1}^2), \vec{r}_1{}^2 = ({r_2}^2, {\alpha_2}^2)$
…	…
ϕ_i	$\ldots \vec{r}_k{}^i = ({r_k}^i, {\alpha_k}^i) \ldots$

Generalized Hough Algorithm

- Create accumulator array $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i),

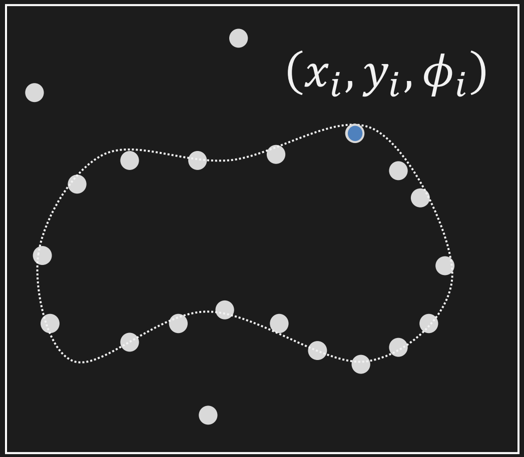
  For each entry $\phi_i \to \vec{r}_k{}^i$ in the $\phi -$ table,

$$x_c = x_i \pm r_k{}^i \cos\left(\alpha_k{}^i\right)$$

$$y_c = y_i \pm r_k{}^i \sin\left(\alpha_k{}^i\right)$$

$$A(x_c, y_c) = A(x_c, y_c) + 1$$

Image



$A(x_c, y_c)$

x_c

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

y_c

Generalized Hough Algorithm

- Create accumulator array $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i),

  For each entry $\phi_i \rightarrow \vec{r}_k{}^i$ in the $\phi -$ table,

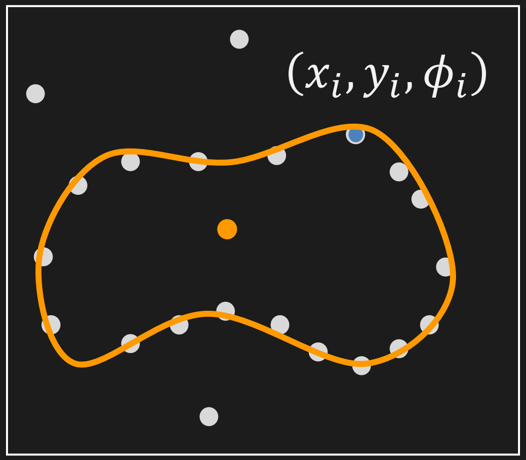
$$x_c = x_i \pm r_k{}^i \cos\left(\alpha_k{}^i\right)$$

$$y_c = y_i \pm r_k{}^i \sin\left(\alpha_k{}^i\right)$$

$$A(x_c, y_c) = A(x_c, y_c) + 1$$

- Find local maxima in $A(x_c, y_c)$

Image



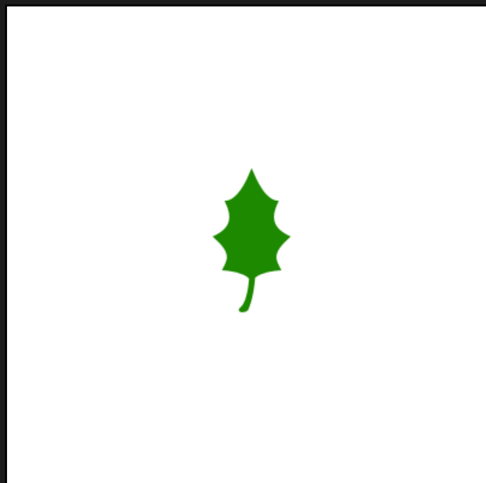
$A(x_c, y_c)$

x_c

0	0	0	0	0
0	2	0	1	0
0	0	$\boxed{4}$	1	0
0	2	0	0	0
0	0	0	1	0

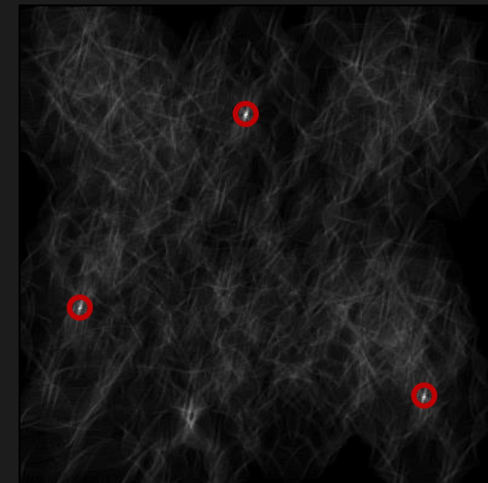
y_c

Results

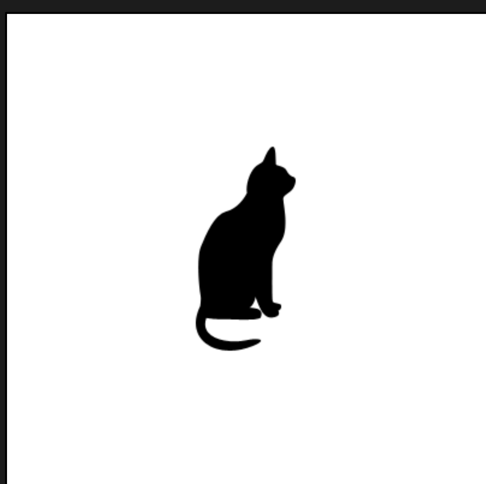


Model

Image

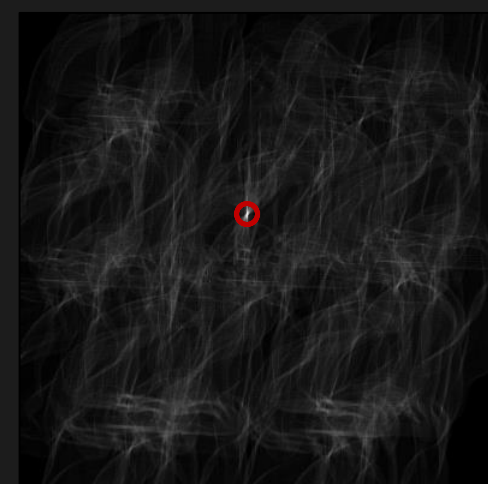


Hough Transform $A(x_c, y_c)$



Model

Image

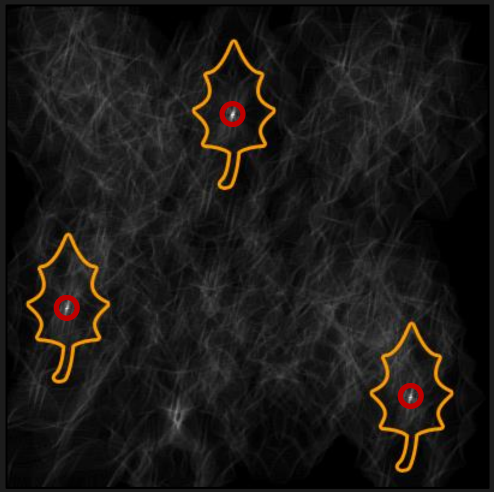


Hough Transform $A(x_c, y_c)$

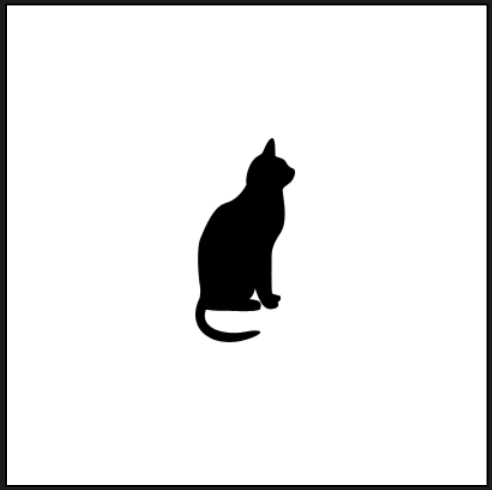
Results

Model

Model Detected

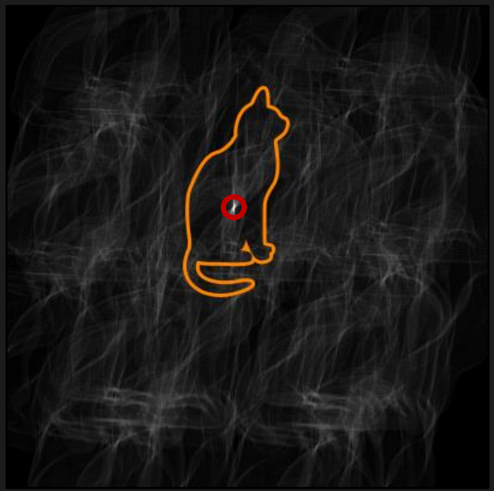


Hough Transform $A(x_c, y_c)$



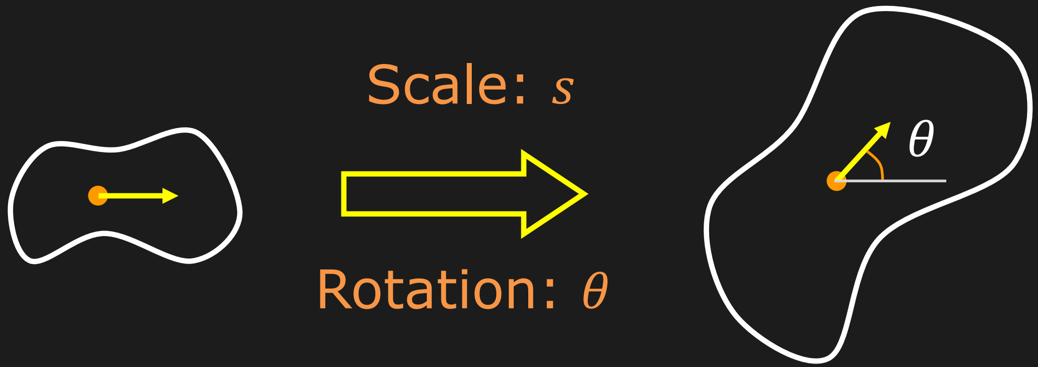
Model

Model Detected



Hough Transform $A(x_c, y_c)$

Handling Scale And Rotation



Use Accumulation Array: $A(x_c, y_c, s, \theta)$

$$x_c = x_i \pm r_k{}^i . s \cos\left(\alpha_k{}^i + \theta\right)$$

$$y_c = y_i \pm r_k{}^i . s \sin\left(\alpha_k{}^i + \theta\right)$$

$$A(x_c, y_c, s, \theta) = A(x_c, y_c, s, \theta) + 1$$

Huge Memory and Computationally Expensive!

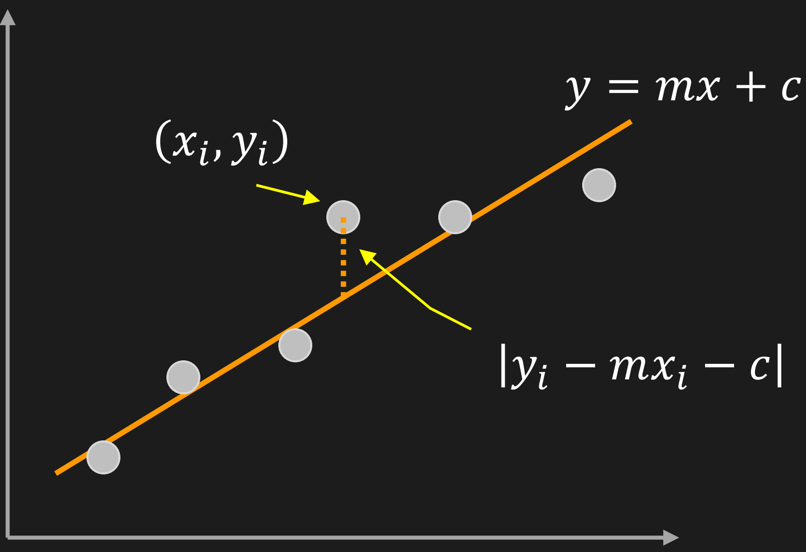
Hough Transform: Comments

- Works on disconnected edges
- Relatively insensitive to occlusion and noise
- Effective for simple shapes (lines, circles, etc.)
- Trade-off between work in image space and parameter space

Fitting Lines to Edges

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



Minimize: Average Squared Vertical Distance

$$E = \frac{1}{N}\sum_i (y_i - mx_i - c)^2$$

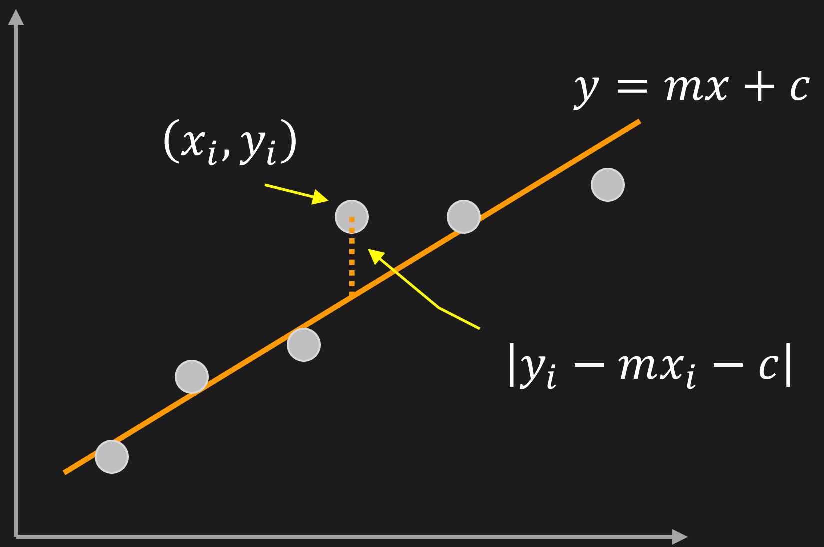
Using: $\frac{\partial E}{\partial m} = 0 \qquad \frac{\partial E}{\partial c} = 0$

Least Squares

Fitting Lines to Edges

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



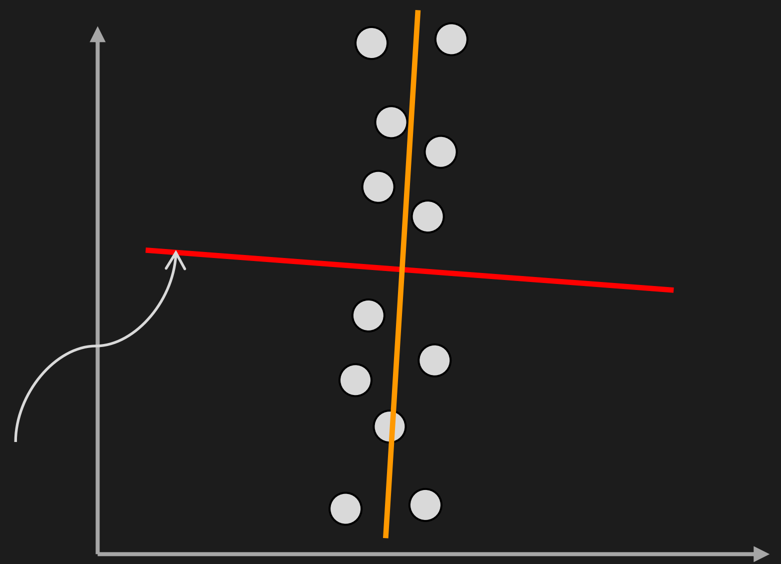
Solution:

$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad c = \bar{y} - m\bar{x}$$

where: $\bar{x} = \frac{1}{N}\sum_i x_i \qquad \bar{y} = \frac{1}{N}\sum_i y_i$

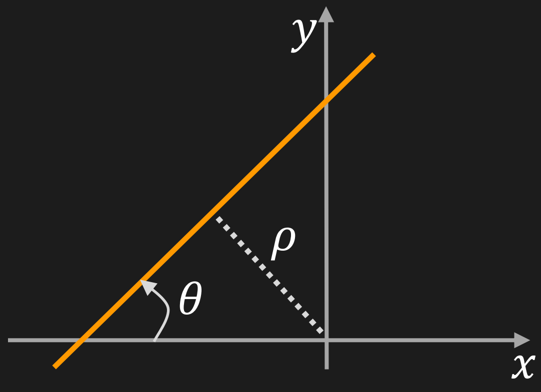
Fitting Lines to Edges

Problem: When the points represent a vertical line.



Line that minimizes E!

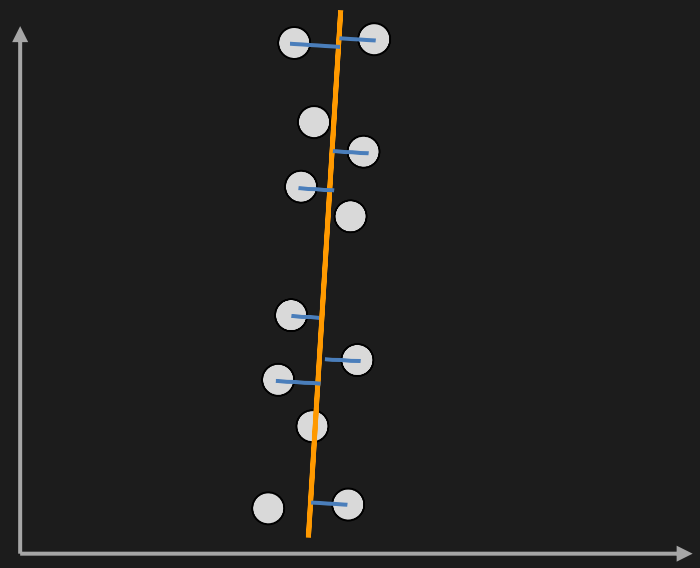
Solution: Use a different line equation



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

Fitting Lines to Edges

Problem: When the points represent a vertical line.



Minimize: Average Squared Perpendicular Distance

$$E = \frac{1}{N}\sum_i (\underbrace{x_i \sin\theta - y_i \cos\theta + \rho}_{\text{(Perpendicular Distance)}})^2$$

References

Textbooks:

Robot Vision (Chapter 8)
Horn, B. K. P., MIT Press

Computer Vision: Algorithms and Applications (Chapter 4.2, 4.3)
Szelinski, 2011 (available online)

Computer Vision: A Modern Approach (Chapter 8)
Forsyth, D and Ponce, J., Prentice Hall

Digital Image Processing (Chapter 3)
González, R and Woods, R., Prentice Hall

Papers:

[Canny1986] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[Harris1988] Harris, C. and Stephens, M., A combined corner and edge detector. Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

[Marr1980] Marr, D. and Hildreth, E., Theory of Edge Detection," Proc. R. Soc. London,B 207, 187-217, 1980.

References: Papers (cont.)

[Ballard 1981] D. H. Ballard."Generalizing the Hough Transform to Detect Arbitrary Shapes". *Pattern Recognition*, vol. 13, no.2, 1981.

[Duda and Hart 1975] R. O. Duda and P. E. Hart. "Use of the Hough Transform to Detect Lines and Curves in Pictures". *Comm. ACM,* vol.15, 1975.

[Hough 1962] P. V. C. Hough. *Method and Means for Recognizing Complex Patterns*. U.S. Patent 3069654, 1962.

[Kass 1987] M. Kass, A. Witkin and D. TerzoPoulos. *"Snakes: Active Contour Models", IJCV,* 1987.

[Xu 1997] C. Xu and J. Prince. "*Gradient Vector Flow: A New external force for Snakes*", CVPR, 1997.