Boundary Detection

Computer Vision: CS 566

Computer Science

University of Wisconsin-Madison

Boundary Detection

We need to find Object Boundaries from Edge Pixels.

Topics:

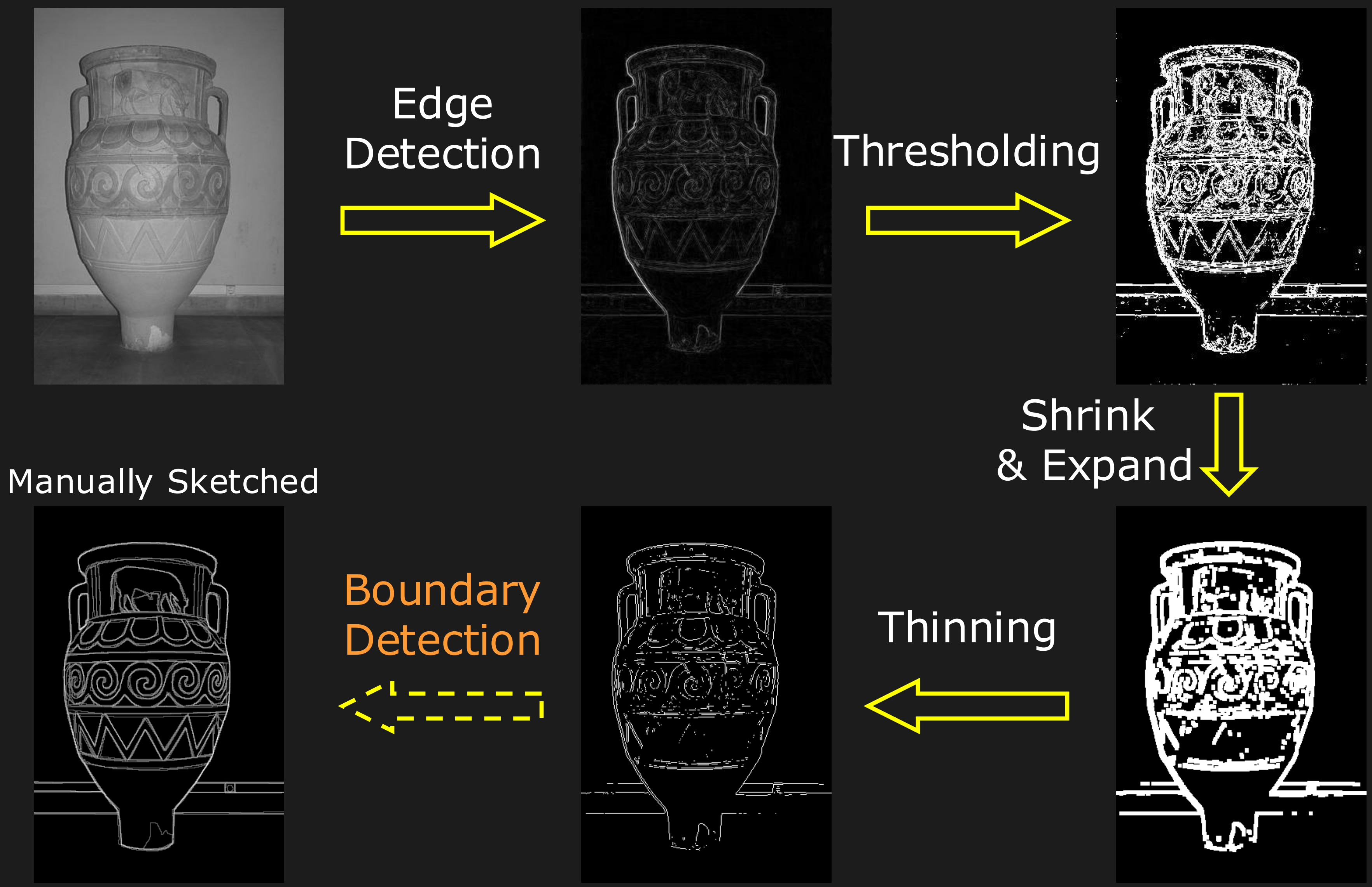
(1) Preprocessing Edge Images

(2) Fitting Lines and Curves to Edges

(3) The Hough Transform

(4) Active Contours (also called Snakes)

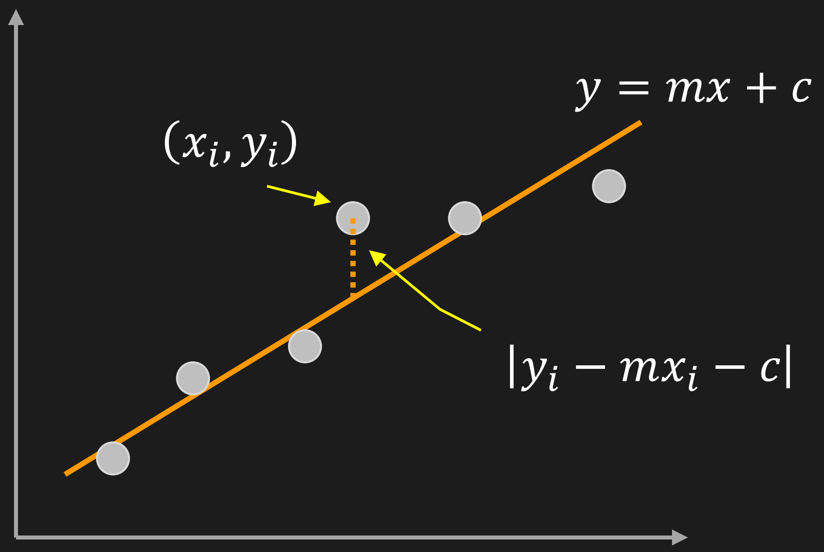
Preprocessing Edge Images



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 Least Squares Solution:

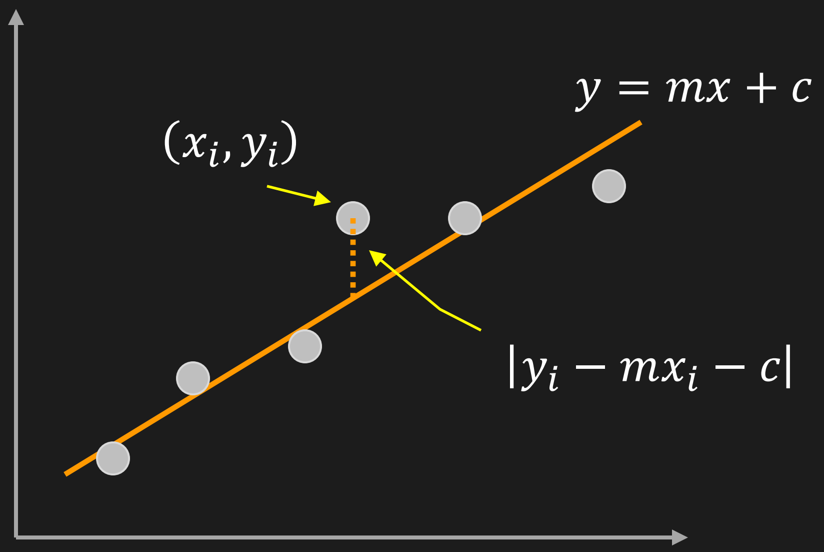
$$\frac{\partial E}{\partial m} = \frac{-2}{N} \sum_i x_i (y_i - mx_i - c) = 0$$

$$\frac{\partial E}{\partial c} = \frac{-2}{N} \sum_i (y_i - mx_i - c) = 0$$

Fitting Lines to Edges

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



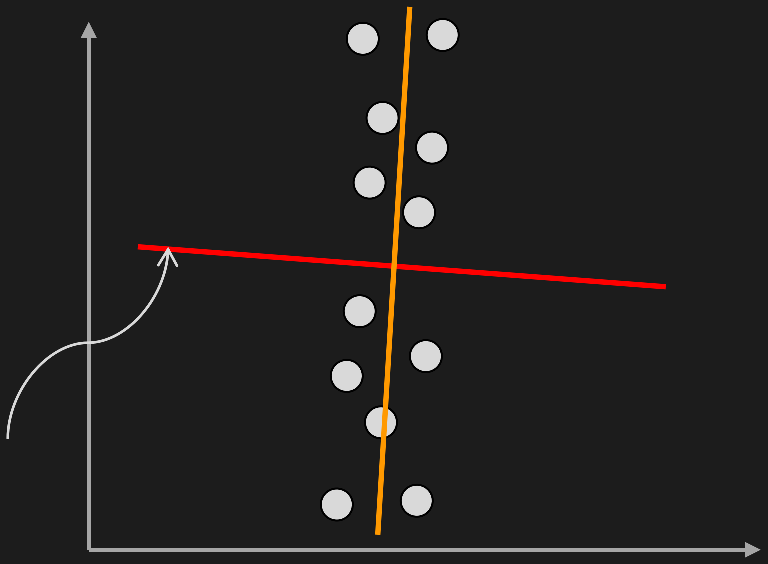
Solution:

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

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

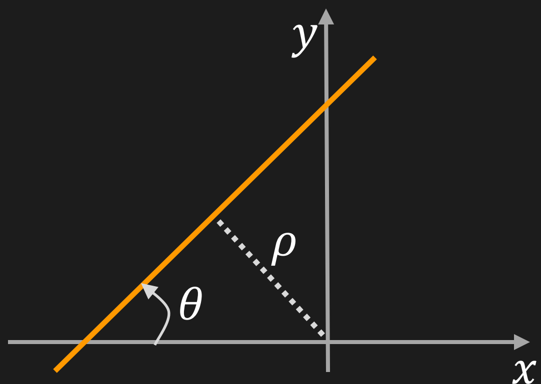
Fitting Lines to Edges

Problem: When the points represent a vertical line.



Line that minimizes E!

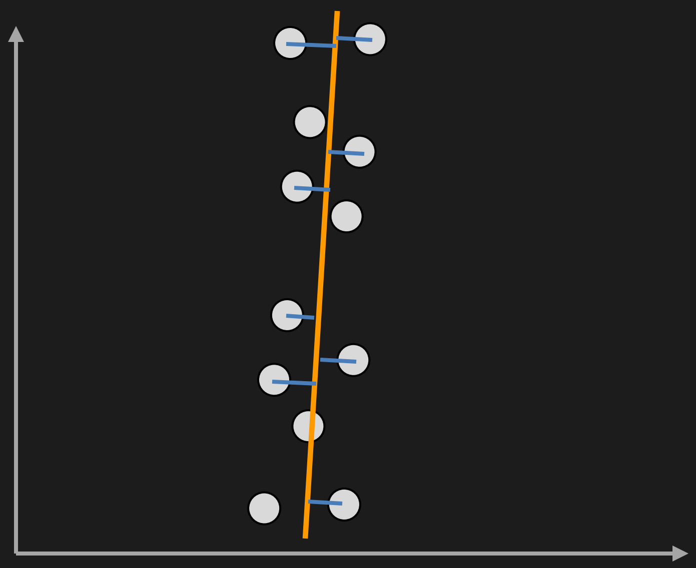
Solution: Use a different line equation



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

Fitting Lines to Edges

Problem: When the points represent a vertical line.



Minimize: Average Squared Perpendicular Distance

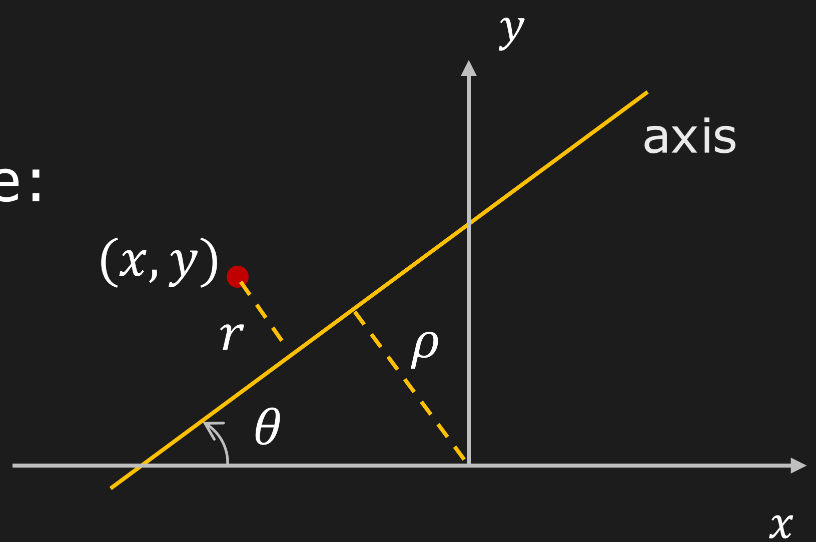
$$E = \frac{1}{N}\sum_{i}\underbrace{(x_i \sin\theta - y_i \cos\theta + \rho)}_{\text{Perpendicular Distance}}{}^2$$

Distance Between Point and Line

Given a line $ax + by + c = 0$

Distance of point (x, y) from line:

$$r = \left| \frac{ax + by + c}{\sqrt{a^2 + b^2}} \right|$$



Similarly, given line $x \sin\theta - y \cos\theta + \rho = 0$

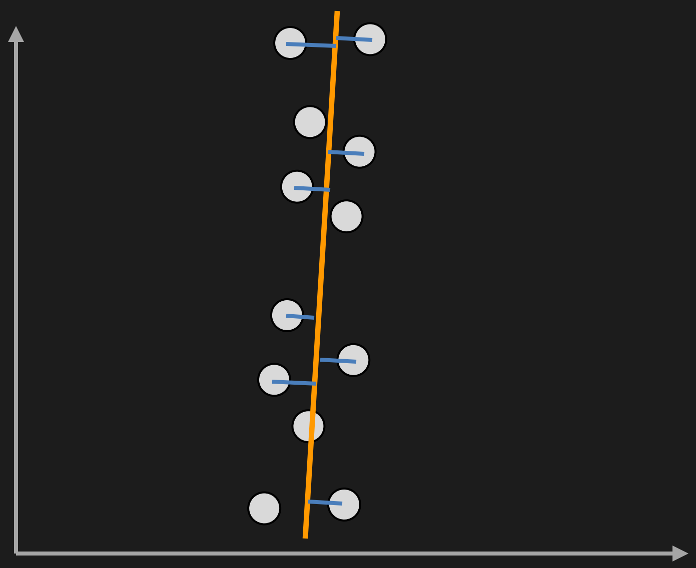
Distance of point (x, y) from line:

$$r = \left| \frac{x \sin\theta - y \cos\theta + \rho}{\sqrt{\sin^2\theta + \cos^2\theta}} \right|$$

$$\boxed{r = |x \sin\theta - y \cos\theta + \rho|}$$

Fitting Lines to Edges

Problem: When the points represent a vertical line.



Minimize: Average Squared Perpendicular Distance

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

Find ρ and θ that minimize E

Fitting Lines to Edges

Minimize: Average Squared Perpendicular Distance

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

Find ρ and θ that minimize E

Using $\frac{\partial E}{\partial \rho} = 0$ we get: $\bar{x} \sin\theta - \bar{y} \cos\theta + \rho = 0$

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

Line passes through centroid $(\bar{x}, \bar{y})$!

Shift the Coordinate System

Change coordinates:

$$x' = x - \bar{x},\ y' = y - \bar{y}$$

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

Therefore, we can rewrite E as:

$$\boxed{E = a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta}$$

where: $\begin{cases} a = \sum_i (x'_i)^2 \\ c = \sum_i (y'_i)^2 \end{cases}$ $\quad b = 2\sum_i x'_i\, y'_i$

(a, b, c are easy to compute)

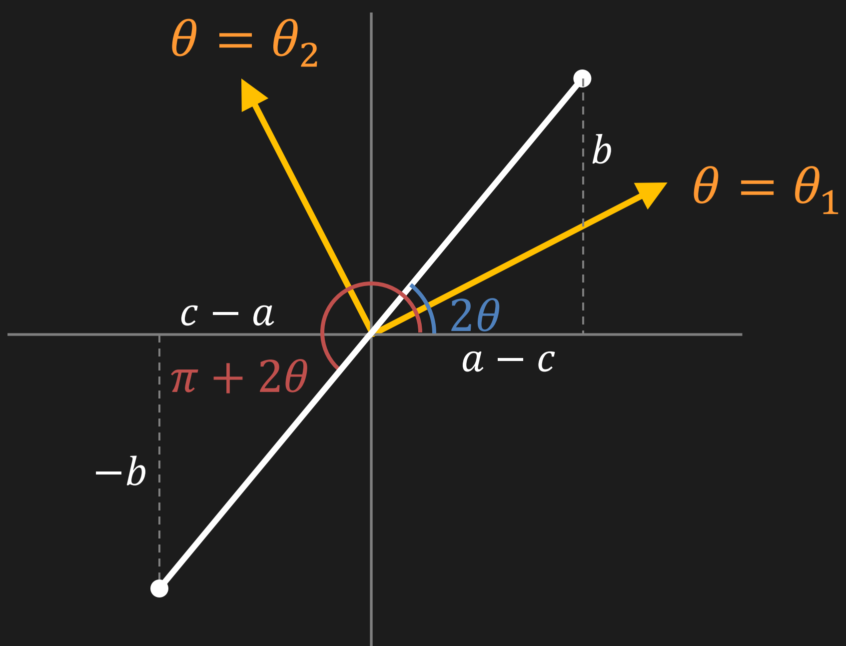
Finally, Minimize E

Using $\frac{dE}{d\theta} = (a - c)\sin 2\theta - b\cos 2\theta = 0$ we get: $\boxed{\tan 2\theta = \frac{b}{a-c}}$

We know that: $\tan 2\theta = \tan(2\theta + \pi) = \frac{b}{a-c}$

θ has two solutions.

1. $\theta = \theta_1$
2. $\theta = \theta_2 = \theta_1 + \frac{\pi}{2}$



One gives Minimum of E
and the other Maximum of E

Which One To Use?

Using second derivative test:

If $\dfrac{d^2E}{d\theta^2} = (a-c)\cos 2\theta + b\sin 2\theta \begin{cases} > 0 \text{ then Minimum} \\ < 0 \text{ then Maximum} \end{cases}$

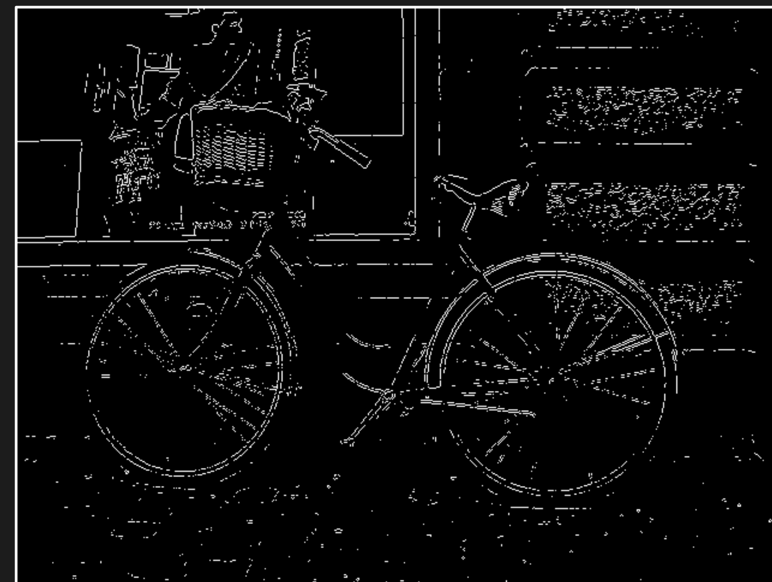
Substituting $\cos 2\theta_1$, $\sin 2\theta_1$, $\cos 2\theta_2$ and $\sin 2\theta_2$:

$$\frac{d^2E}{d\theta^2}(\theta_1) > 0 \quad \text{and} \quad \frac{d^2E}{d\theta^2}(\theta_2) < 0$$

Therefore,

Orientation: $\boxed{\theta = \theta_1 = \dfrac{atan2(b, a-c)}{2}}$

Difficulties for the Fitting Approach



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

Solution: Hough Transform

The Hough Transform

Elegant method for Direct Object Recognition

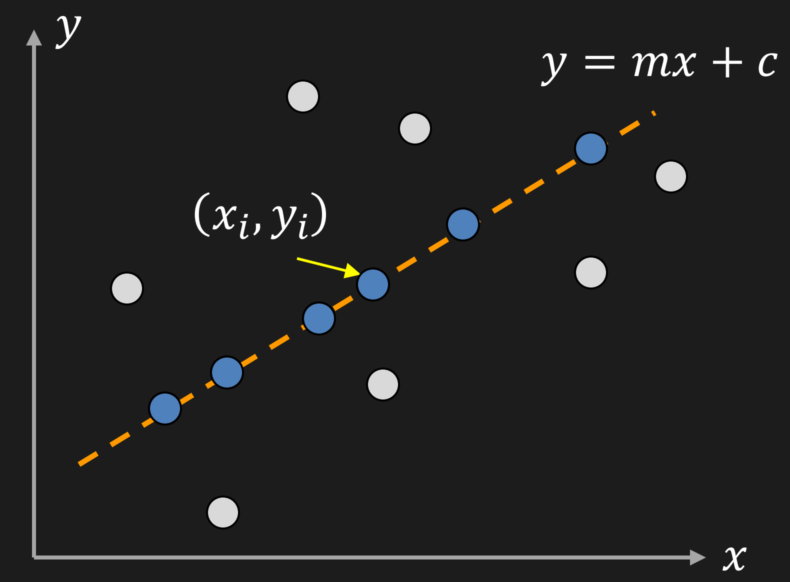
- Robust to disconnected edges
- Complete object need not be visible
- Relatively robust to noise

Hough Transform: Line Detection

Given: Edge Points (x_i, y_i)

Task: Detect line

$$y = mx + c$$

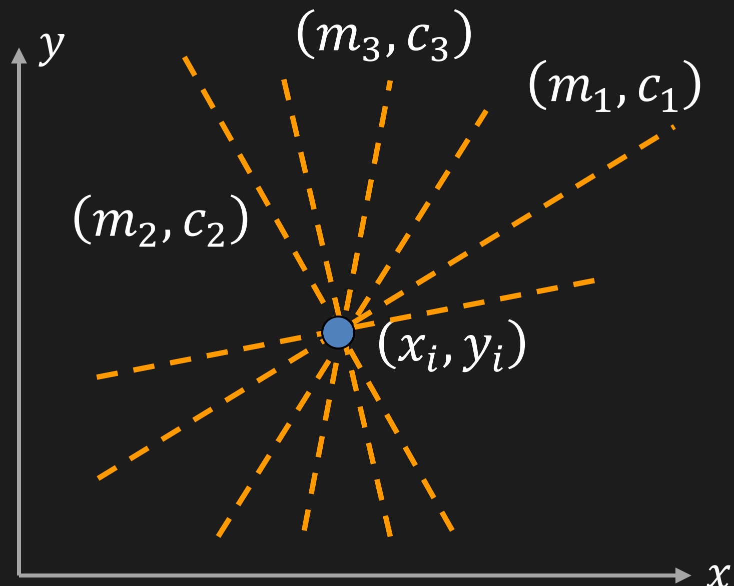


Consider point (x_i, y_i)

[Hough 1962]

Hough Transform: Concept

Image Space



$$y_i = m_1 x_i + c_1$$

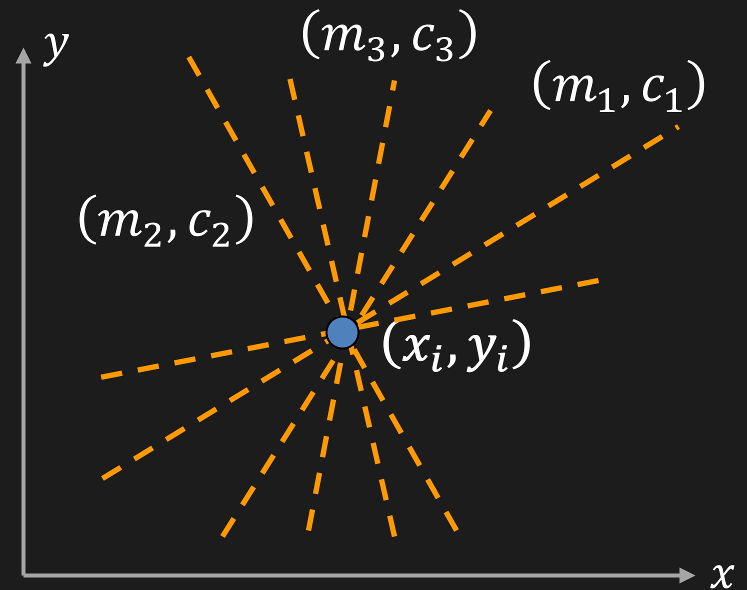
$$y_i = m_2 x_i + c_2$$

$$y_i = m_3 x_i + c_3$$

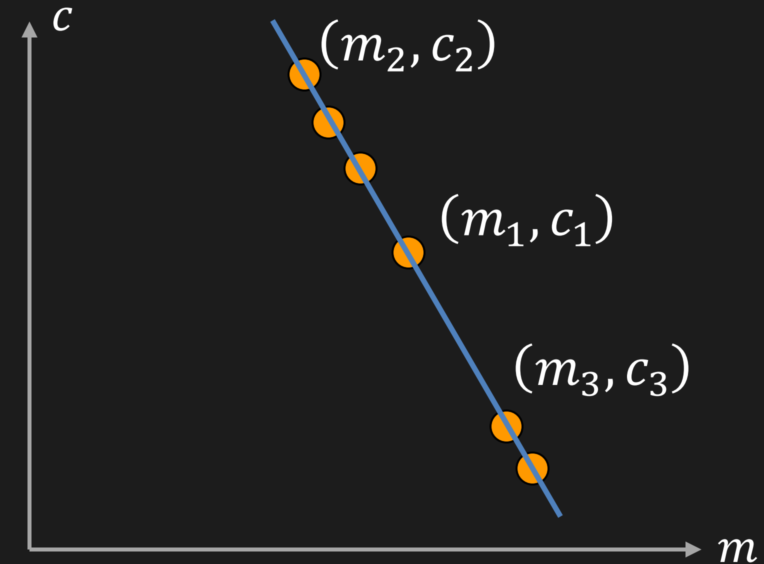
$$\vdots$$

Hough Transform: Concept

Image Space



Parameter Space



$$y_i = m_1 x_i + c_1$$
$$y_i = m_2 x_i + c_2$$
$$y_i = m_3 x_i + c_3$$
$$\vdots$$

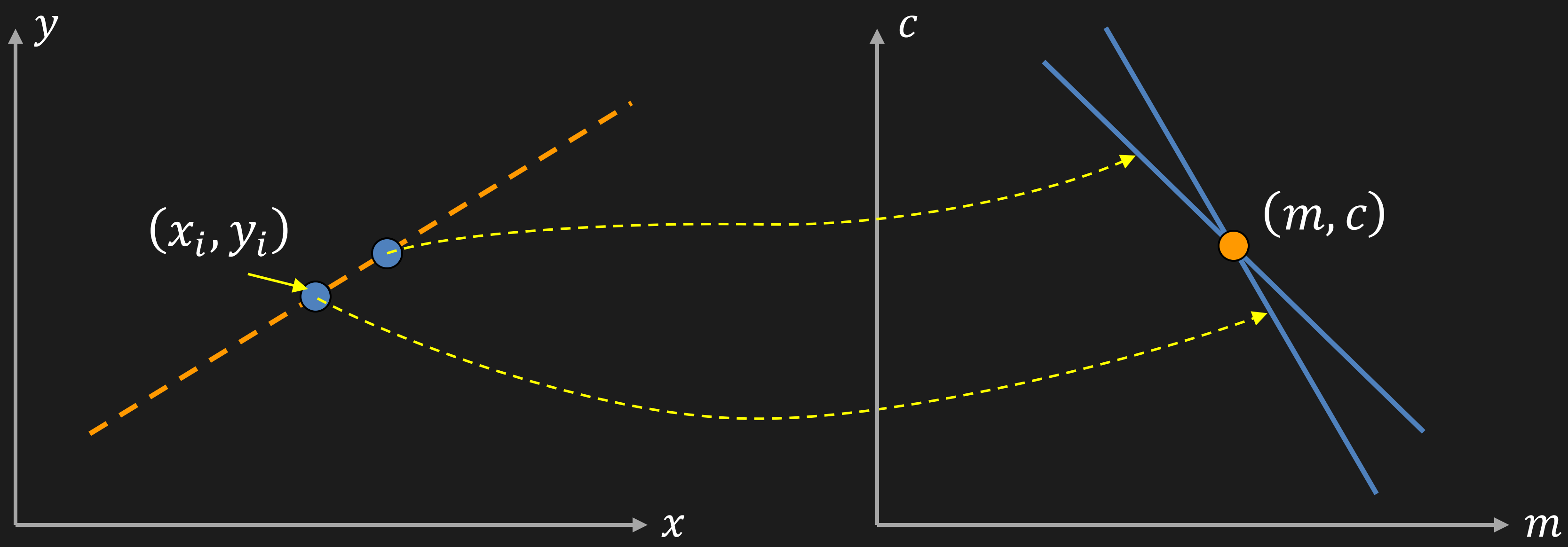
$$\Longleftrightarrow$$

$$c = (-x_i)\, m + y_i$$

Hough Transform: Concept

Image Space

Parameter Space

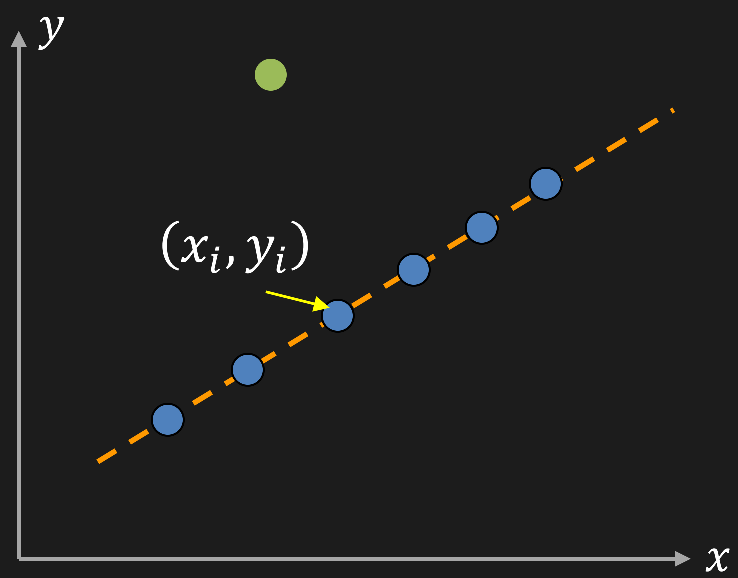


$y_i = mx_i + c$

$c = -mx_i + y_i$

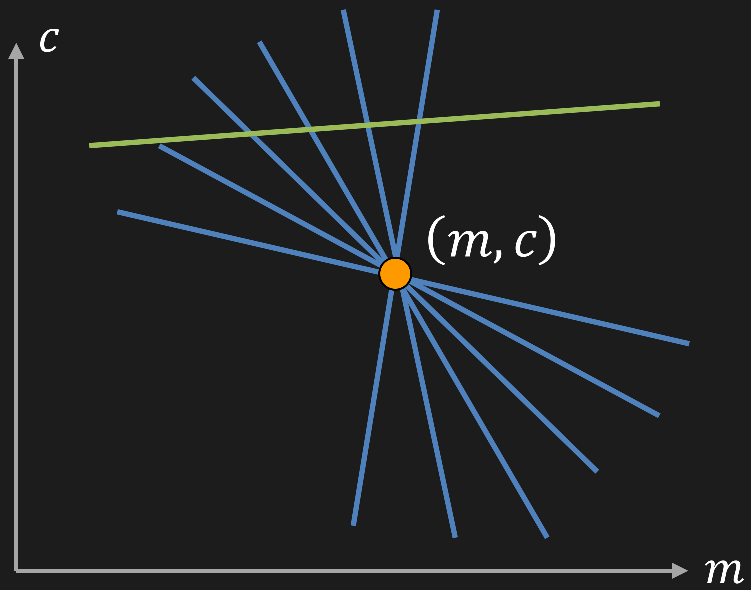
Hough Transform: Concept

Image Space



$y_i = mx_i + c$

Parameter Space



$c = -mx_i + y_i$

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)

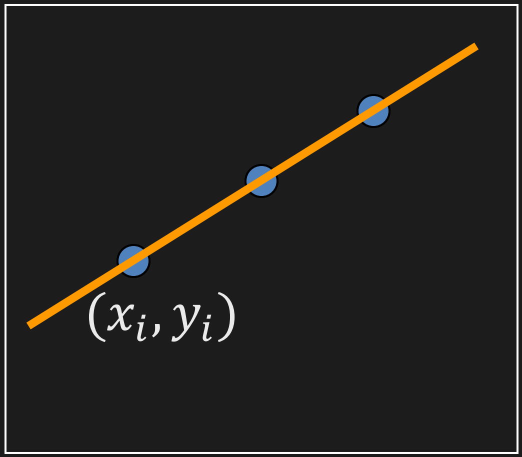
Step 4. For each edge point (x_i, y_i),

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

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

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	$\boxed{3}$	1	1
0	1	0	1	0
1	0	0	0	1

m

Multiple Line Detection

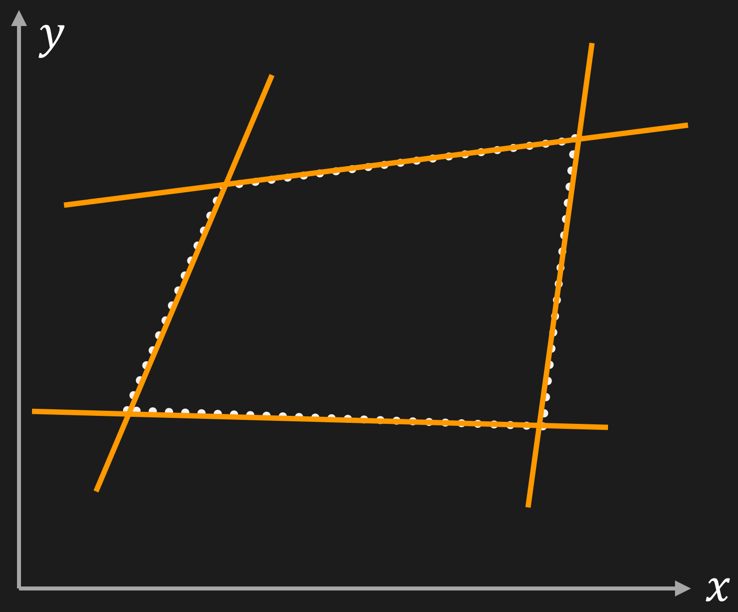
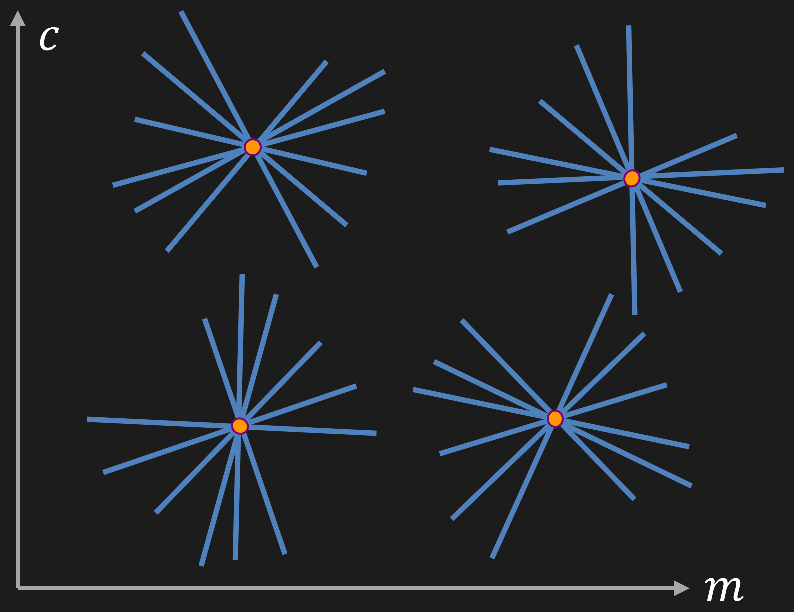


Image Space



Parameter Space

Better Parameterization

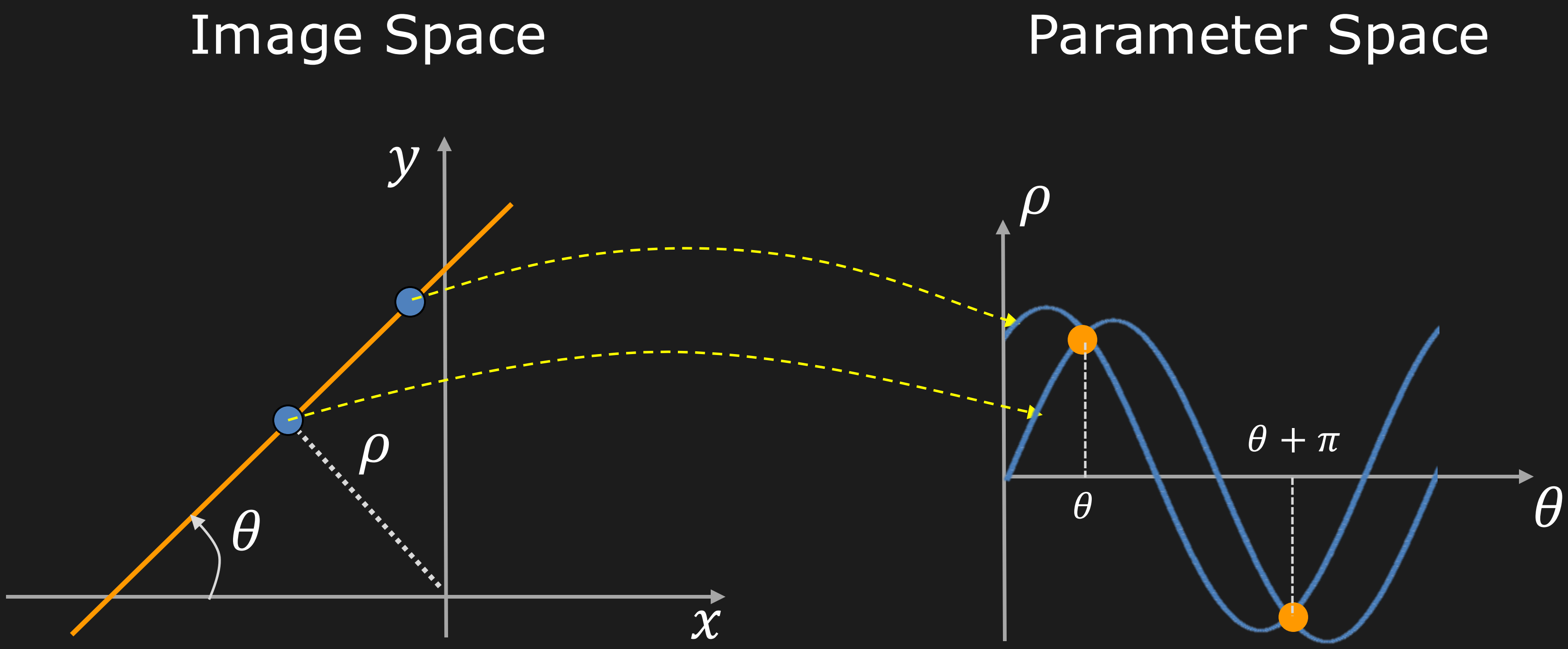
Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation

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

- Orientation θ is finite: $0 \leq \theta < \pi$
- Distance ρ is finite

Better Parameterization



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

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

For images: $0 \leq \theta < \pi$ and $|\rho| \leq$ 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

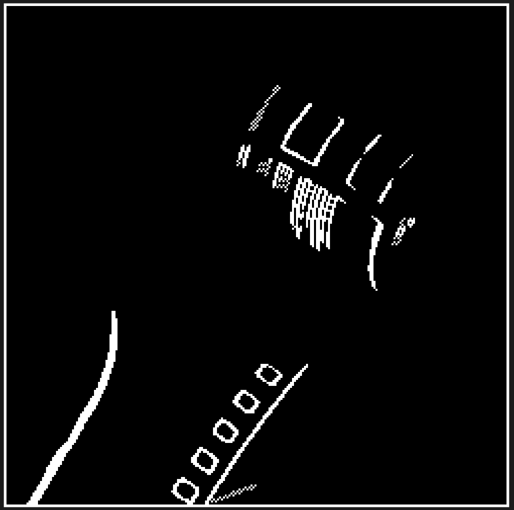
- How many lines?
  – Count the peaks in the accumulator array

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

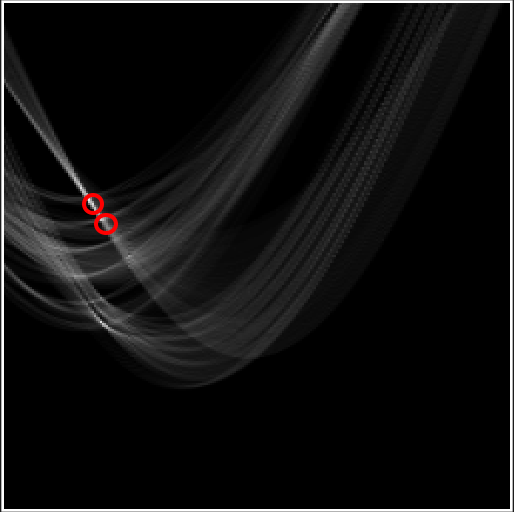
Line Detection Results

Original Image

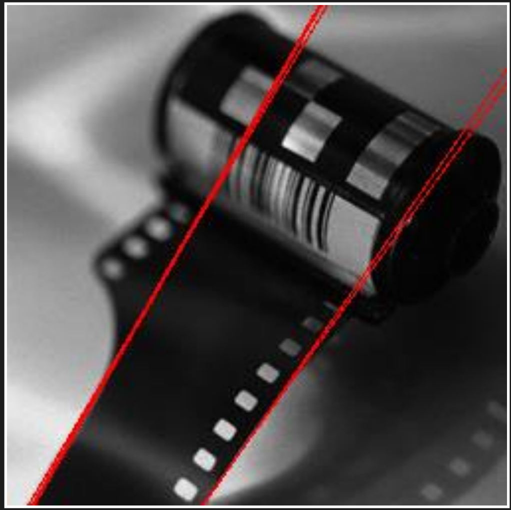
Gradient



Edge (Threshold)



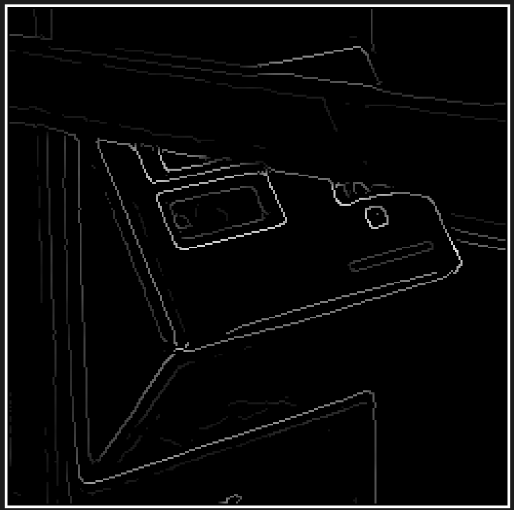
Hough Transform $A(\rho, \theta)$



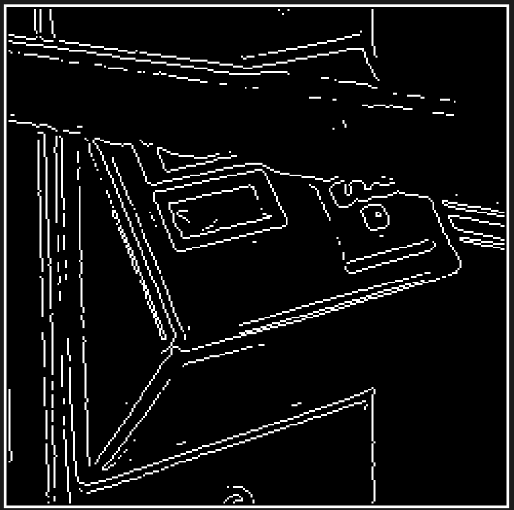
Detected Lines

Line Detection Results

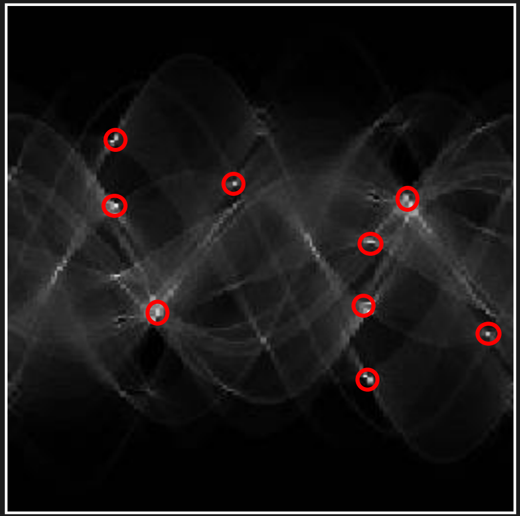
Original Image



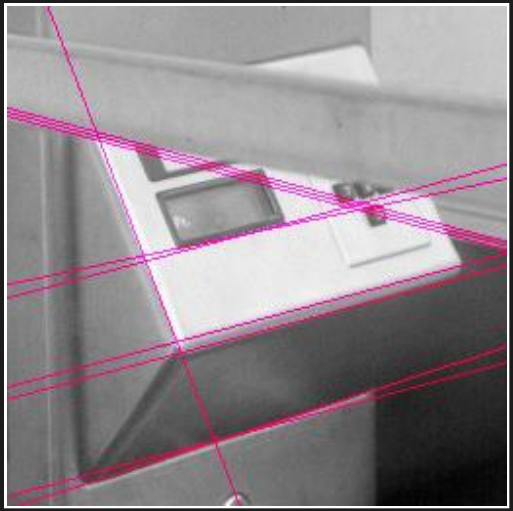
Gradient



Edge (Threshold)

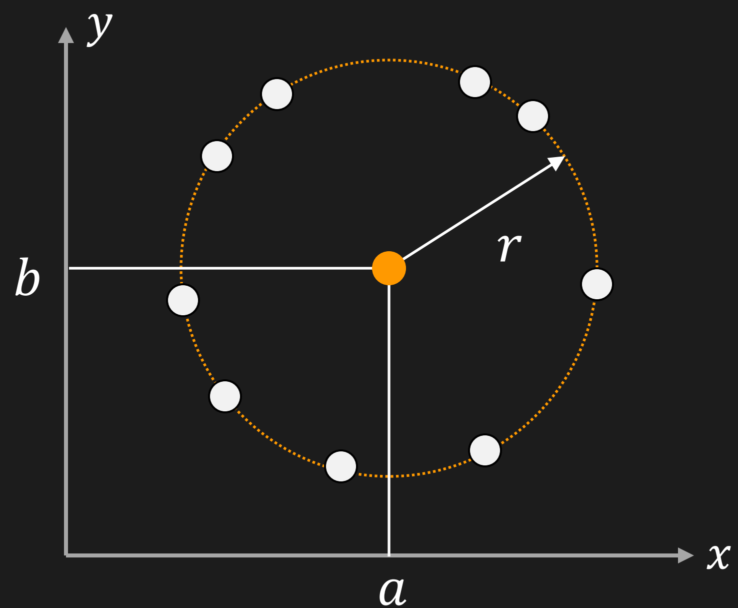


Hough Transform $A(\rho, \theta)$



Detected Lines

Hough Transform: Circle Detection

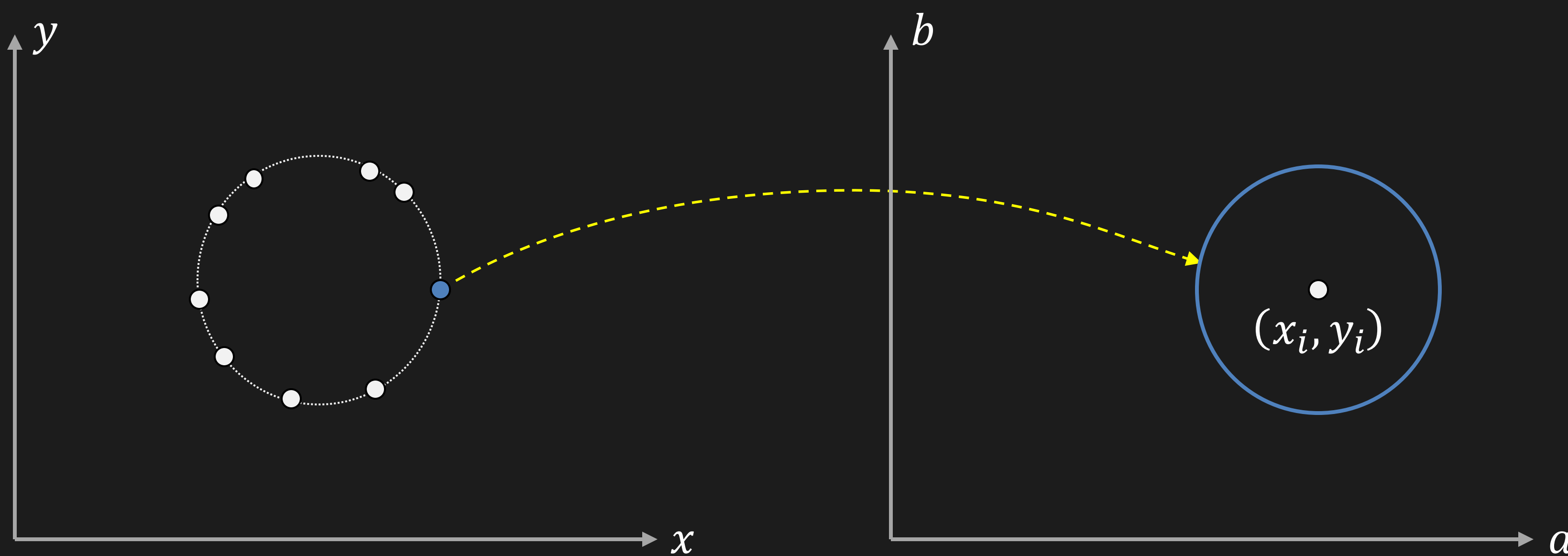


Equation of Circle: $(x_i - a)^2 + (y_i - b)^2 = r^2$

Hough Transform: Circle Detection

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

Image Space | Parameter Space



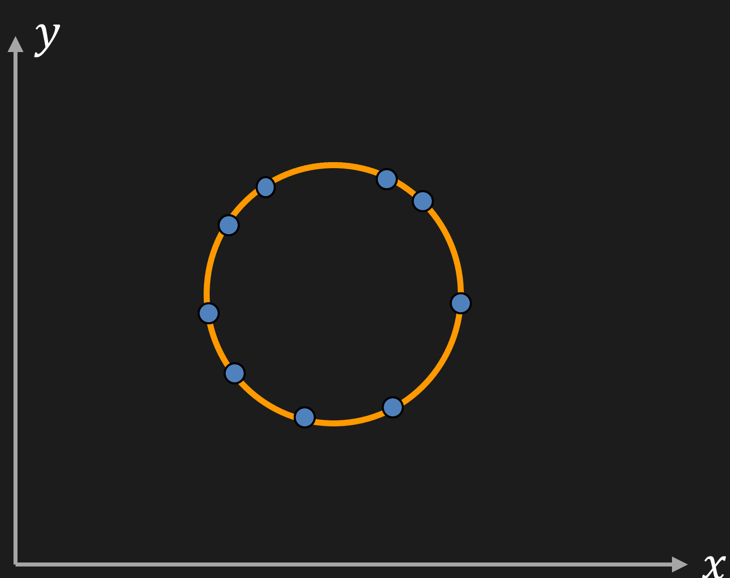
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

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

Hough Transform: Circle Detection

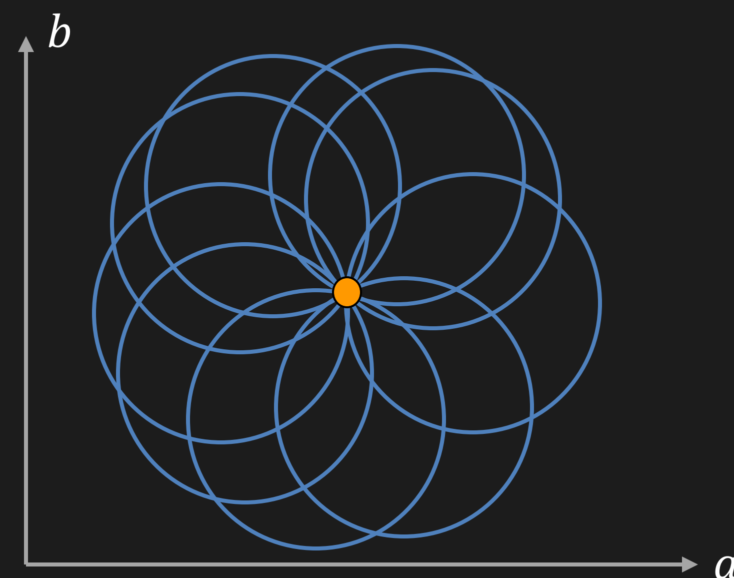
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



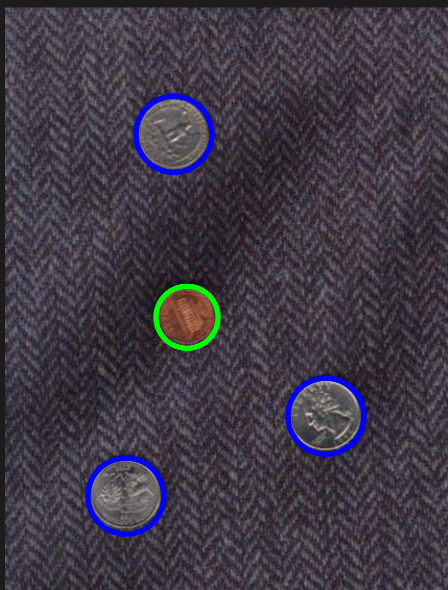
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space

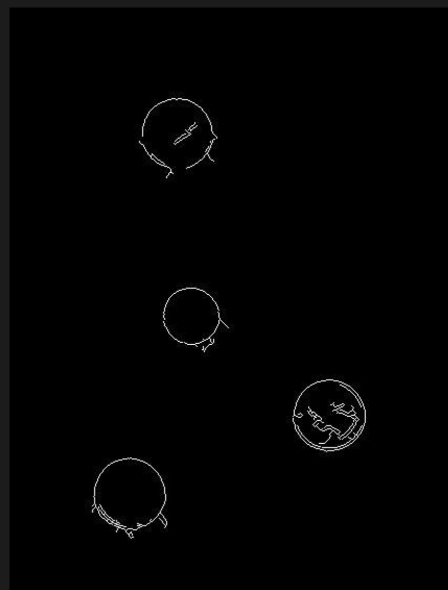


$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Circle Detection Results

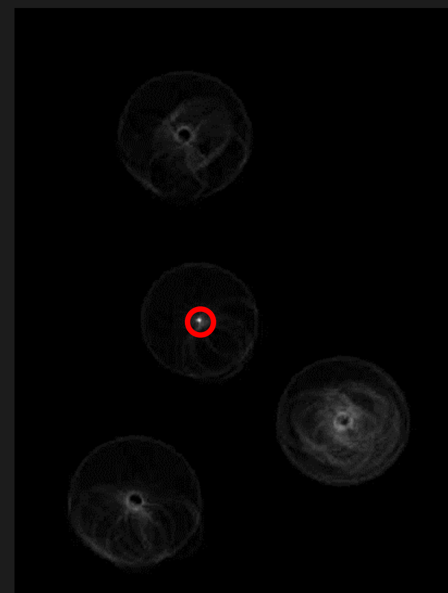


Original Image



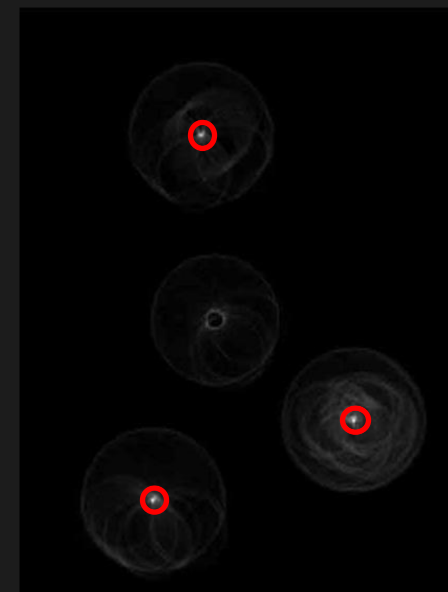
Edge (Threshold)

Penny $(r = r_1)$



Hough Transform $A_1(a, b)$

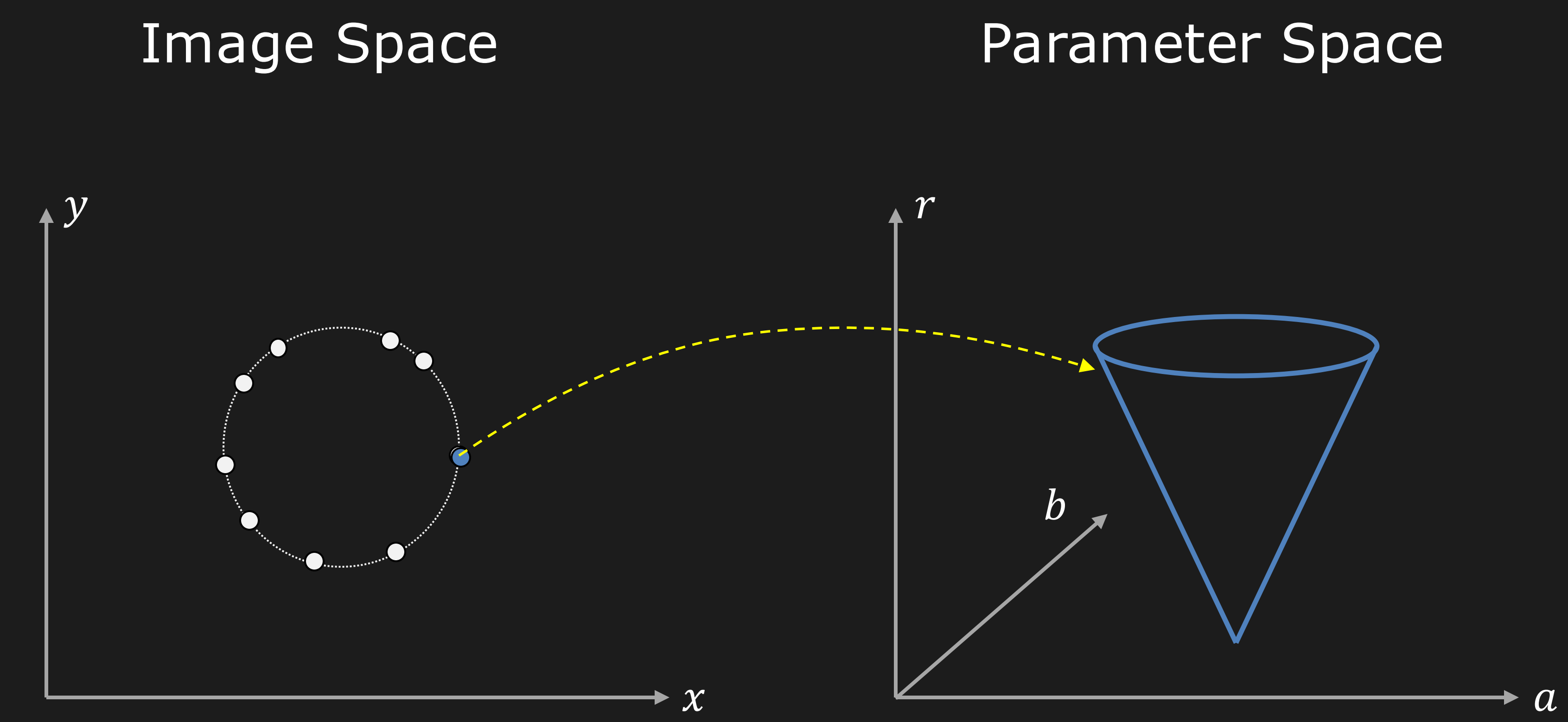
Quarter $(r = r_2)$



Hough Transform $A_2(a, b)$

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

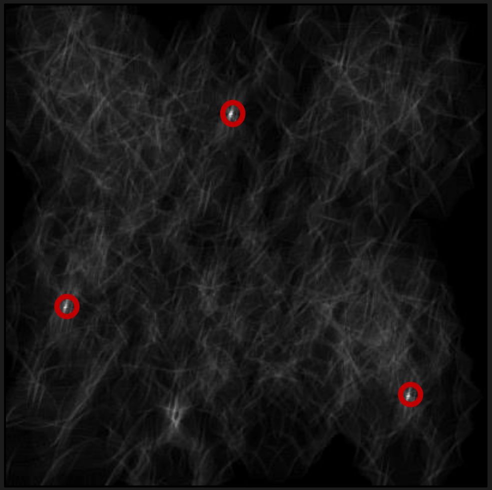
Find shapes that cannot be described by Equations

[Ballard 1981]

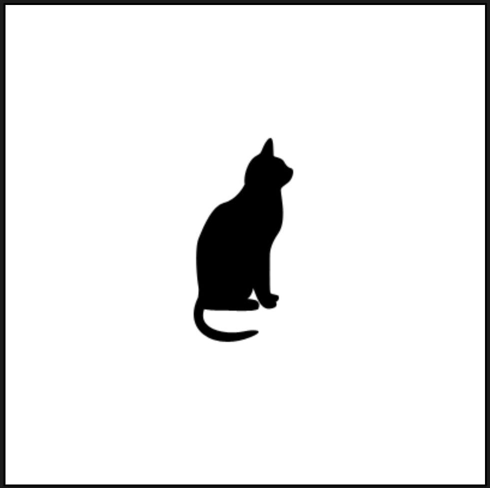
Results

Model

Image

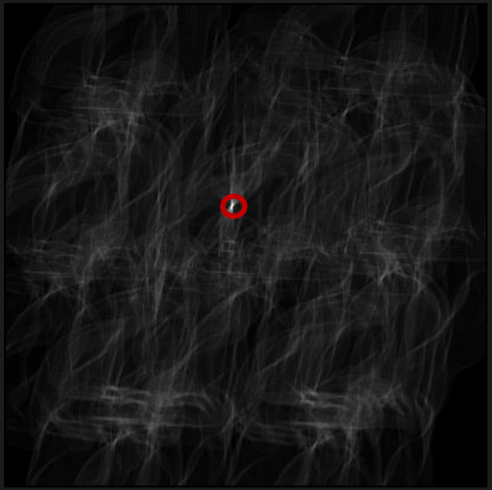


Hough Transform $A(x_c, y_c)$



Model

Image

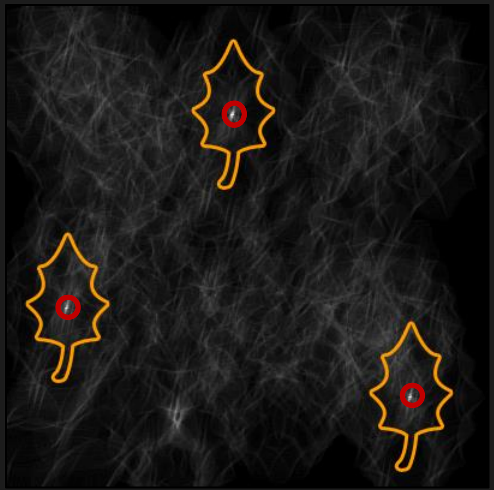


Hough Transform $A(x_c, y_c)$

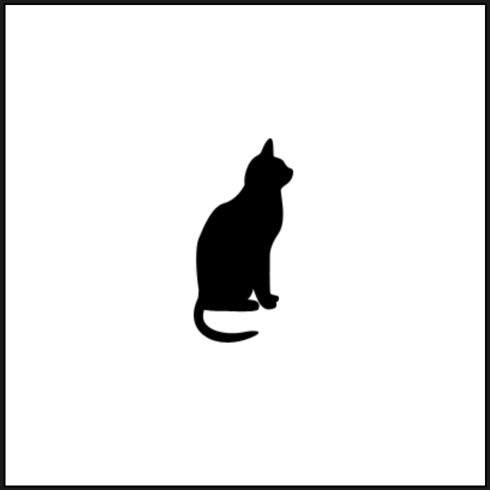
Results

Model

Model Detected

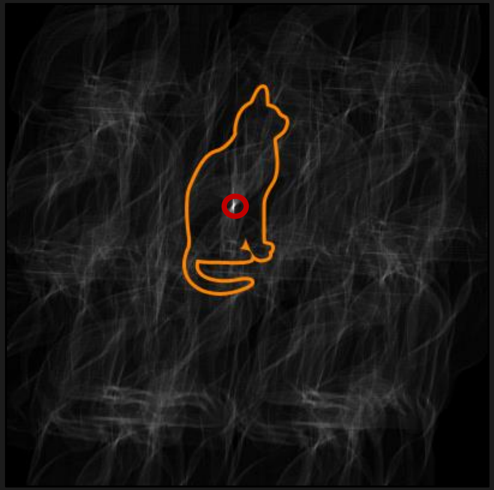


Hough Transform $A(x_c, y_c)$



Model

Model Detected



Hough Transform $A(x_c, y_c)$

Hough Transform: Comments

- Works on disconnected edges
- Relatively insensitive to occlusion and noise
- Effective for simple shapes (lines, circles, etc.)
- Complex Shapes: Generalized Hough Transform

Refining Approximate Boundary

Given: Approximate boundary (contour) around the object

Task: Evolve (move) the contour to fit exact object boundary

Image

Deformable Contours:

Iteratively "deform" the initial contour so that:

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

Also called **Active Contours** or **Snakes**

[Kass 1987]

Why Deformable Contours?

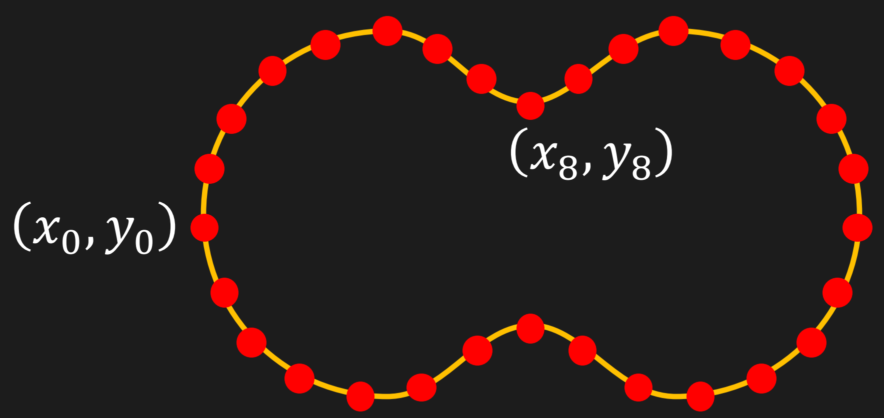
Boundaries could deform over time

Boundaries could deform with viewpoint

Boundary Tracking: Use the boundary from the current image as initial boundary for the next image.

Representing a Contour

Contour $\mathbf{v}$: A ordered list of 2D vertices (control points) connected by straight lines



$$\mathbf{v} = \{v_i = (x_i, y_i) \mid i = 0, 1, 2, \dots, n-1\}$$

Deformable Contours (Snakes)

Deformable Contours: Iteratively move each vertex to a nearby position that provides a “better fit”.

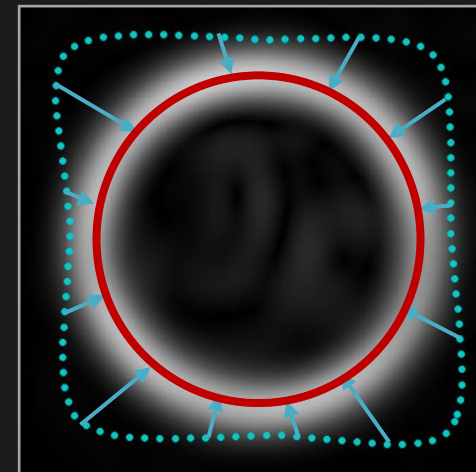
Contract contour to snap on edges

We need some force to attract the contour towards the edges

Attracting Contours to Edges

Image with
Initial Contour

Gradient Magnitude
Squared
$\|\nabla I\|^2$



Blurred Gradient
Magnitude Squared
$\|\nabla n_\sigma * I\|^2$

Maximize Sum of Image Gradient at contour points

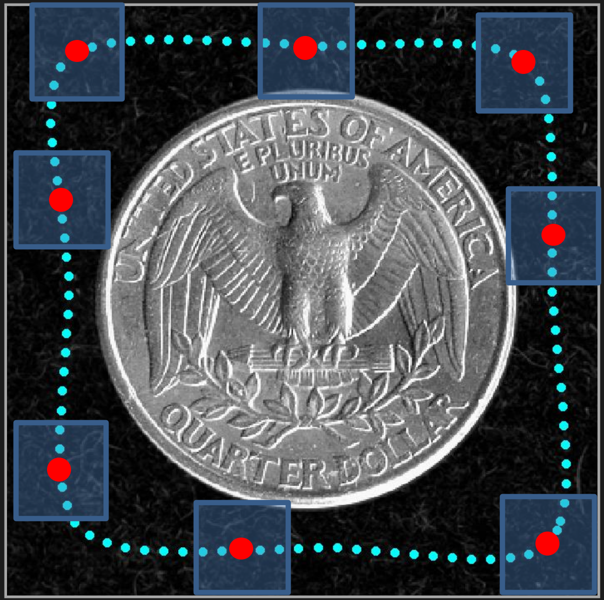
≡ Minimize –ve (Sum of Image Gradient at contour points)

≡ Minimize $\boxed{E_{image} = -\sum_{i=0}^{n-1} \|\nabla n_\sigma * I(v_i)\|^2}$

Contour Deformation: Greedy Algorithm

1. For each contour point v_i $(i = 0, \dots, n-1)$, move v_i to a position within a window W where the energy function E_{image} for the contour is minimum.

2. If the sum of motions of all the contour points is less than a threshold, stop. Else go to Step 1.



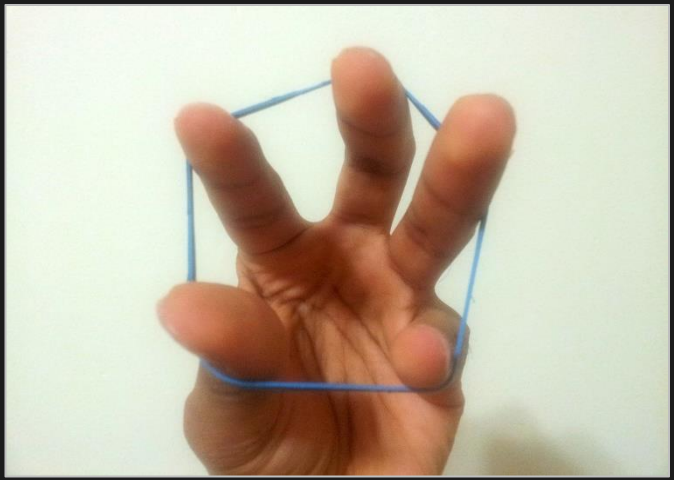
Greedy solution might be suboptimal.

Sensitivity to Noise and Initialization

Contour fitted to
gradient magnitude

Solution: Add constraints that make the contour
contract and remain smooth

Making Contours Elastic and Smooth



Elastic and contracts
like a rubber band

Smooth
like a metal strip

Minimize Internal Bending Energy of the Contour:

$$\boxed{E_{contour} = \alpha E_{elastic} + \beta E_{smooth}}$$

(α, β): Control the influence of elasticity and smoothness

Elasticity and Smoothness

Internal bending energy along the entire contour:

$$\boxed{E_{contour} = \alpha\, E_{elastic} + \beta\, E_{smooth}}$$

where:

$$E_{elastic} = \sum_{i=0}^{n-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2]$$

$$E_{smooth} = \sum_{i=0}^{n-1} [(x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2]$$

Combining the Forces

Image Energy, E_{image}: Measure of how well the contour latches on to edges

Internal Energy, $E_{contour}$: Measure of elasticity and smoothness

Total Energy of the Snake: $\boxed{E_{total} = E_{image} + E_{contour}}$

Minimize the Total Energy

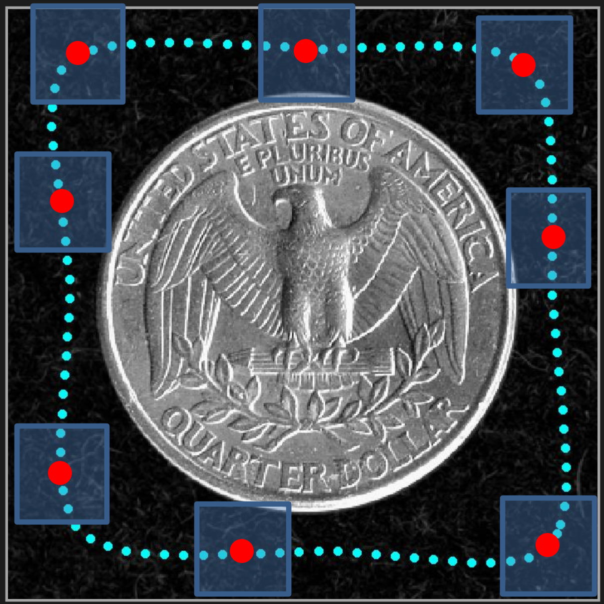
Contour Deformation: Greedy Algorithm

1. Uniformly sample the contour to get n contour points.

2. For each contour point v_i $(i = 0, \ldots, n-1)$, move v_i to a position within a window W where the energy function E_{total} for the entire contour is minimum.

$$\boxed{E_{total} = E_{image} + E_{contour}}$$

3. If the sum of motions of all the contour points is less than a threshold, stop. Else go to Step 1.



Result: Effect of Contour Constraint

Without contour constraint

$$E_{total} = E_{image}$$

With contour constraint

$$E_{total} = E_{image} + E_{contour}$$

Result: Boundary Around Two Objects

Large α, Small β

(More like a rubber band)

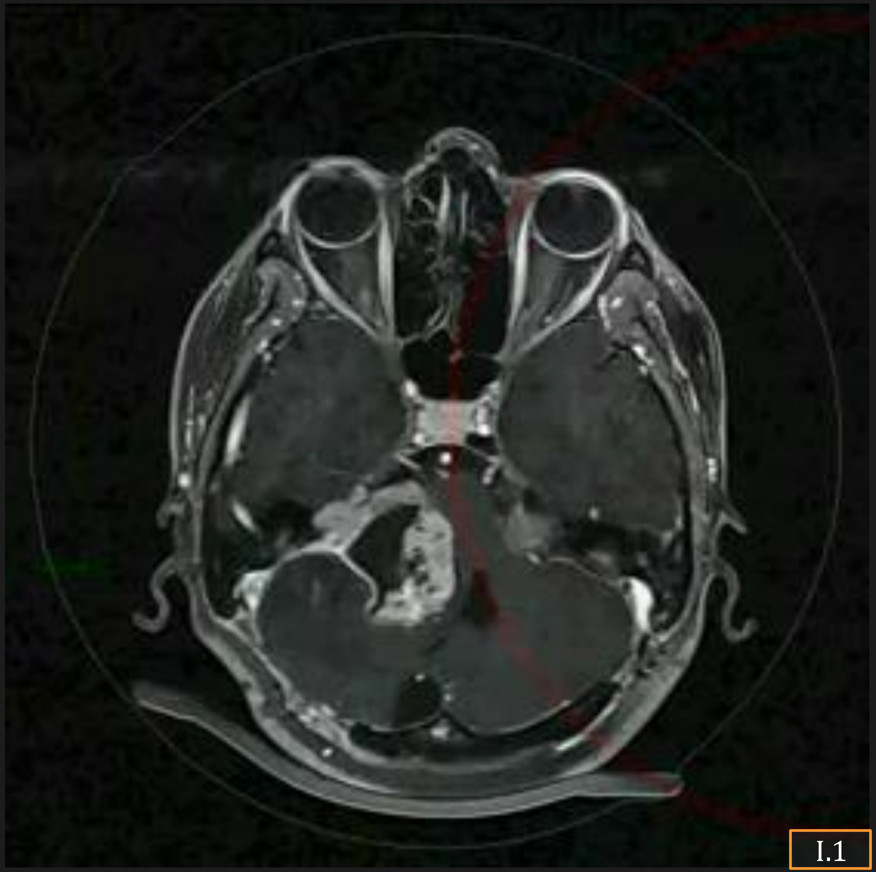
Small α, Large β

(More like a metal strip)

Active Contours: Comments

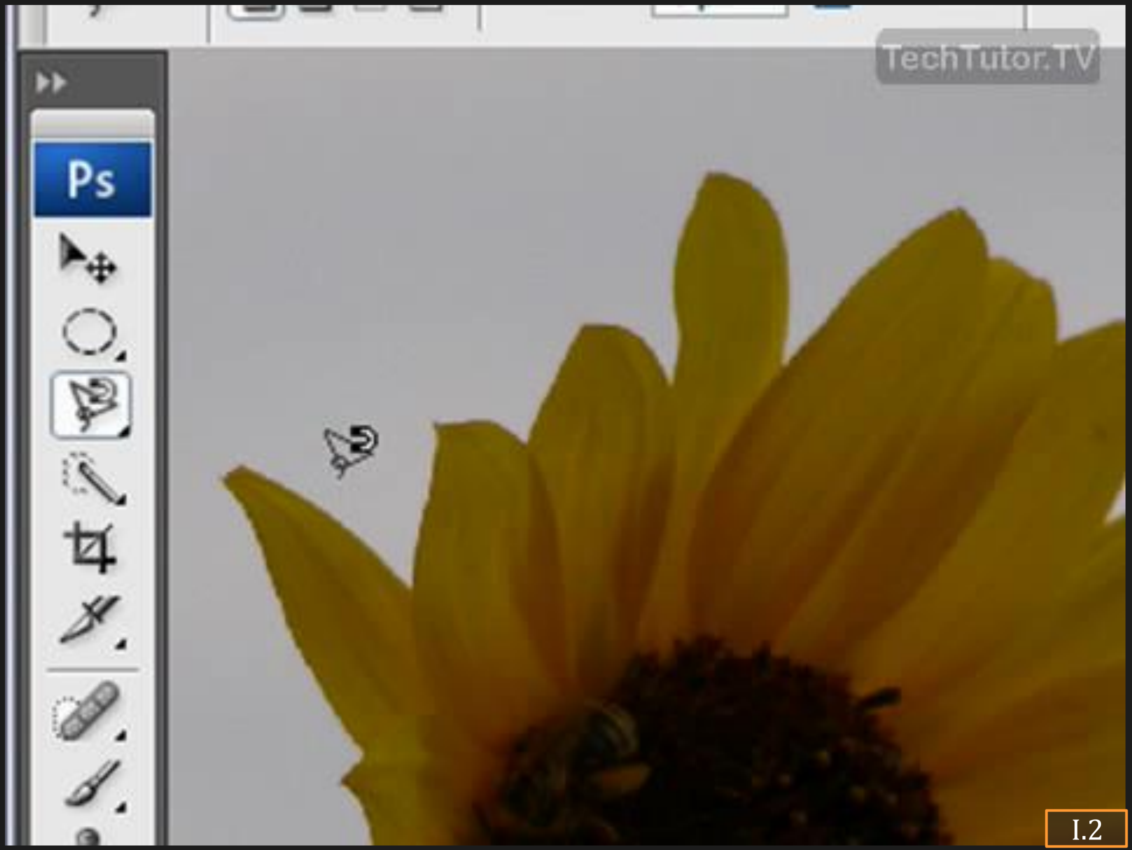
- Additional energy constraints can be added
  - Penalize deviation from prior model of shape
- Requires good initialization
  - Edges cannot attract contours that are far away
- Elasticity makes contour contract
  - Replace contracting force with ballooning force to expand

Active Contours for Segmentation



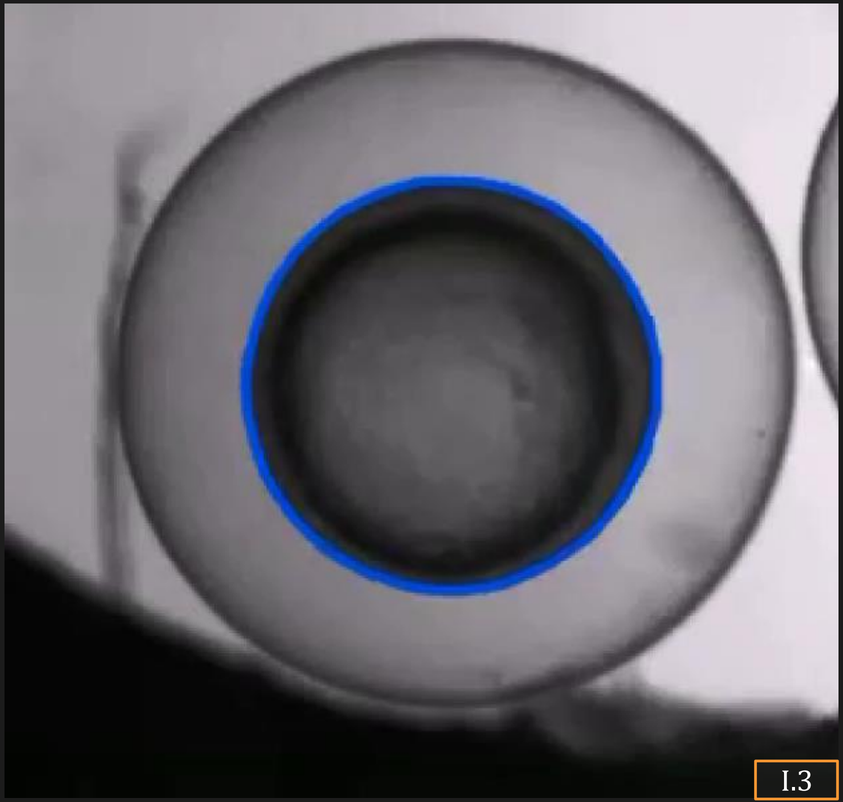
Medical Image Segmentation

Contour Fitting for Interactive Segmentation



Magnetic Lasso Tool in Photoshop

Active Contours for Tracking



Tracking Embryonic Development of Fish

References: Papers

[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.

Image Credits

I.1	http://www.youtube.com/watch?v=CeU_yZjdVqY
I.2	http://www.youtube.com/watch?v=KolDV4AUGko
I.3	http://www.youtube.com/watch?v=n6nHAnc0E2E
I.4	http://www.flickr.com/photos/84598054@N00/74553636/
I.5	http://www.cc.gatech.edu/~kwatra/computer_vision/coins/coins.html