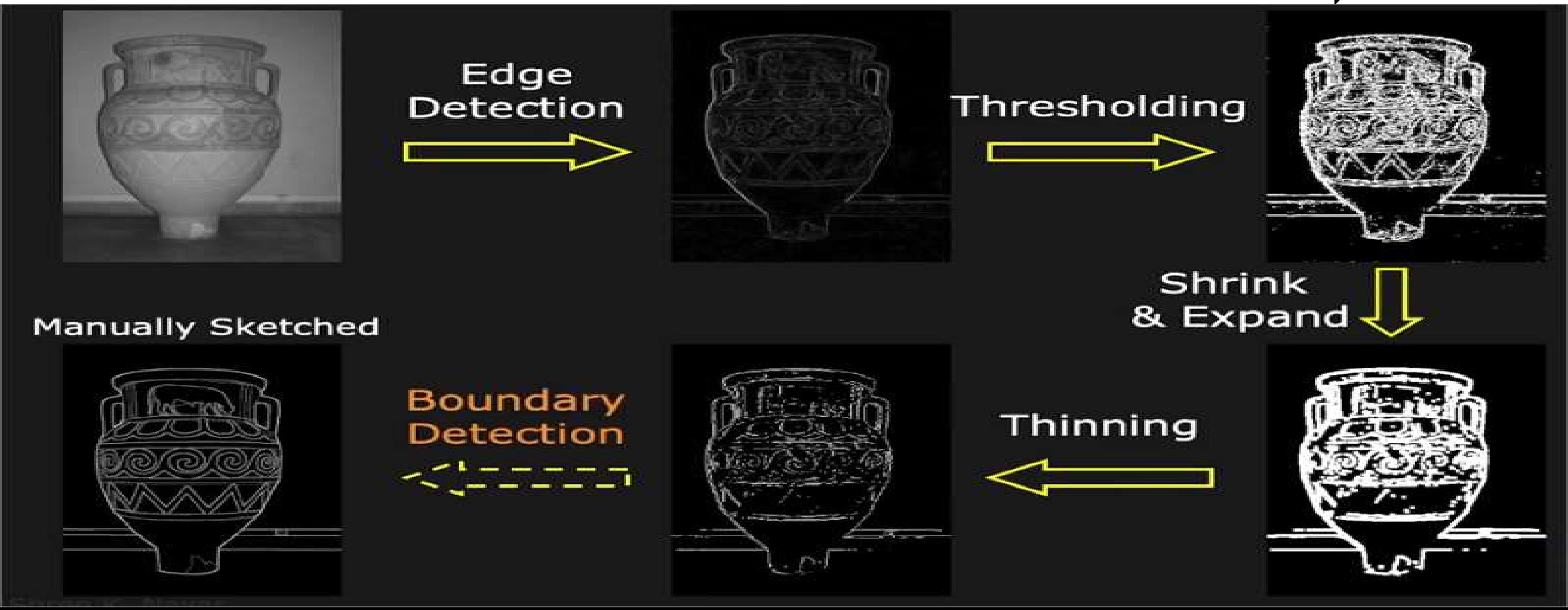
Boundary Detection

- We need to find object boundary from the edge pixels
  - Fitting lines and curves to edges
  - Active contours (Snakes)
  - The Hough Transform
  - The generalized Hough Transform

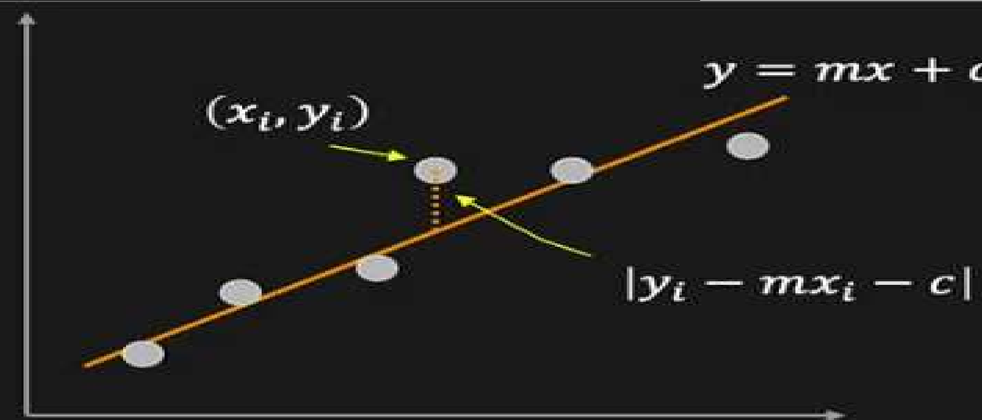
Fitting line and Curves: Preprocessing Edge Images



Line Fitting

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



Minimize: Average Squared Vertical Distance

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

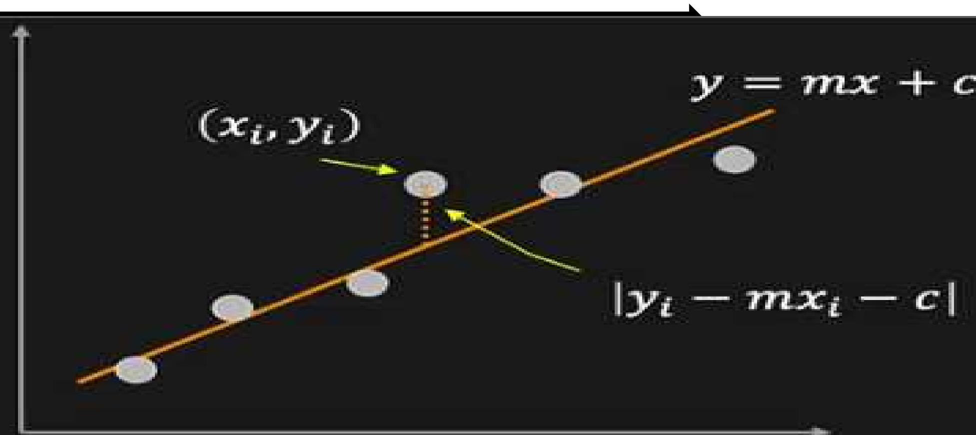
Least Squares Solution:

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

Close form solution

Given: Edge Points (x_i, y_i)

Task: Find (m, c)

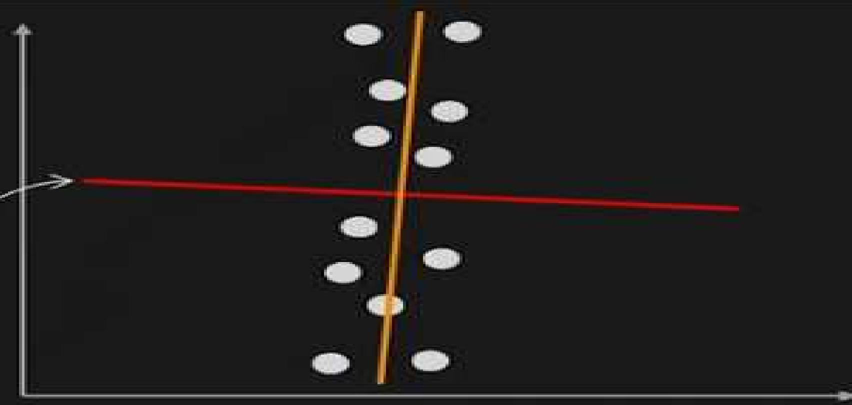


Solution:

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

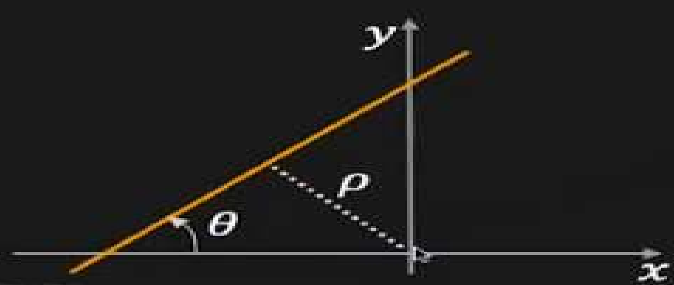
where: $\bar{x} = \frac{1}{N}\sum_i x_i \qquad \bar{y} = \frac{1}{N}\sum_i y_i$

Problem: When the points represent a vertical line.



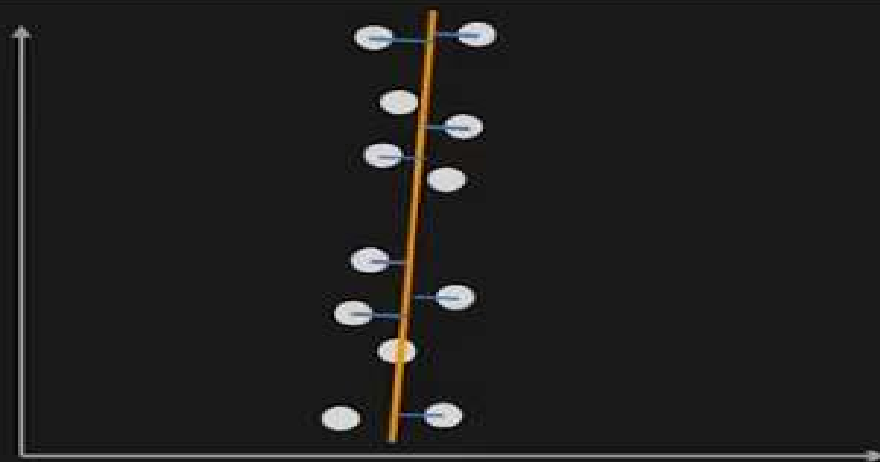
Line that minimizes E!

Solution: Use a different line equation



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

Problem: When the points represent a vertical line.



Minimize: Average Squared **Perpendicular** Distance

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

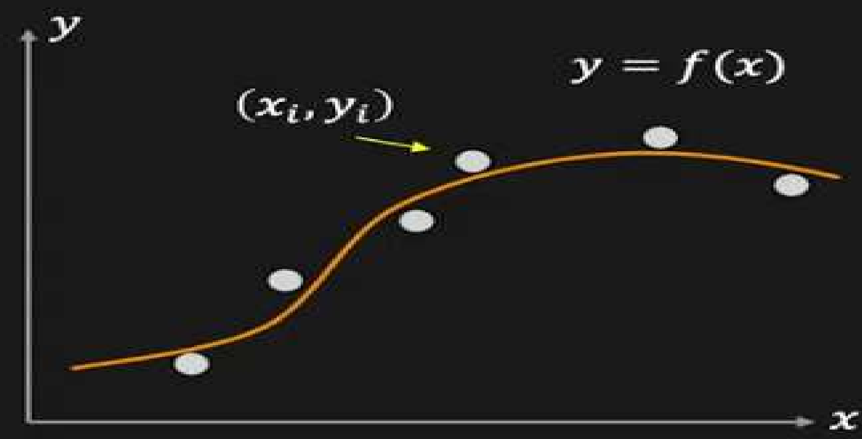
Fitting curves to edges

Given: Edge Points (x_i, y_i)

Task: Find polynomial

$y = f(x) = ax^3 + bx^2 + cx + d$

that best fits the points



Minimize:

$$E = \frac{1}{N}\sum_i (y_i - a{x_i}^3 - b{x_i}^2 - cx_i - d)^2$$

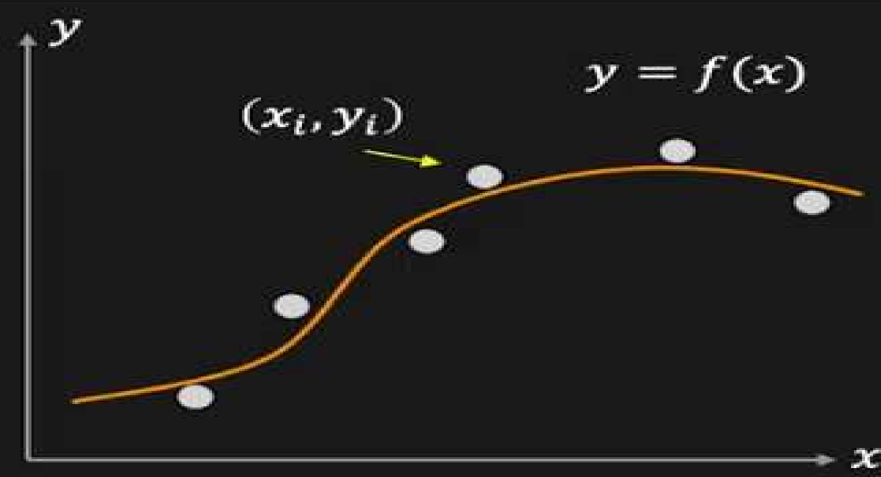
Solve the Linear System Using Least Squares Fit by:

$$\frac{\partial E}{\partial a} = 0 \qquad \frac{\partial E}{\partial b} = 0 \qquad \frac{\partial E}{\partial c} = 0 \qquad \frac{\partial E}{\partial c} = 0$$

Overdetermined problem

Solving as a Linear System:

$$y_0 = a{x_0}^3 + b{x_0}^2 + cx_0 + d$$
$$y_1 = a{x_1}^3 + b{x_1}^2 + cx_1 + d$$
$$\vdots$$
$$y_i = a{x_i}^3 + b{x_i}^2 + cx_i + d$$
$$\vdots$$
$$y_n = a{x_n}^3 + b{x_n}^2 + cx_n + d$$



Given many (x_i, y_i)'s, this is an over-determined linear system with four unknowns (a, b, c, d).

Solving Linear Equations

An over-determined linear system with m unknowns $\{a_j\}$ $(j = 0, \dots, m)$ and n observations $\{(x_{ij}, y_i)\}$ $(i = 0, \dots, n)$ $(n > m)$ can be written in a matrix form.

$$\underbrace{\begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0m} \\ x_{10} & x_{11} & \cdots & x_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nm} \end{bmatrix}}_{\substack{X_{n\times m} \\ \text{Known}}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}}_{\substack{\mathbf{a}_{m\times 1} \\ \text{Unknown}}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\substack{\mathbf{y}_{n\times 1} \\ \text{Known}}}$$

$$X\mathbf{a} = \mathbf{y}$$

$X_{n\times m}$ is not a square matrix and hence not invertible.

Least Squares Solution:

$$X^T X\mathbf{a} = X^T\mathbf{y} \quad \Rightarrow \quad \mathbf{a} = (X^T X)^{-1} X^T \mathbf{y}$$

$$X^+ = (X^T X)^{-1} X^T$$

(Pseudo Inverse)

$$\boxed{\mathbf{a} = X^+\mathbf{y}}$$

What is active contours.

Given: Approximate boundary (contour) around the object

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

Image

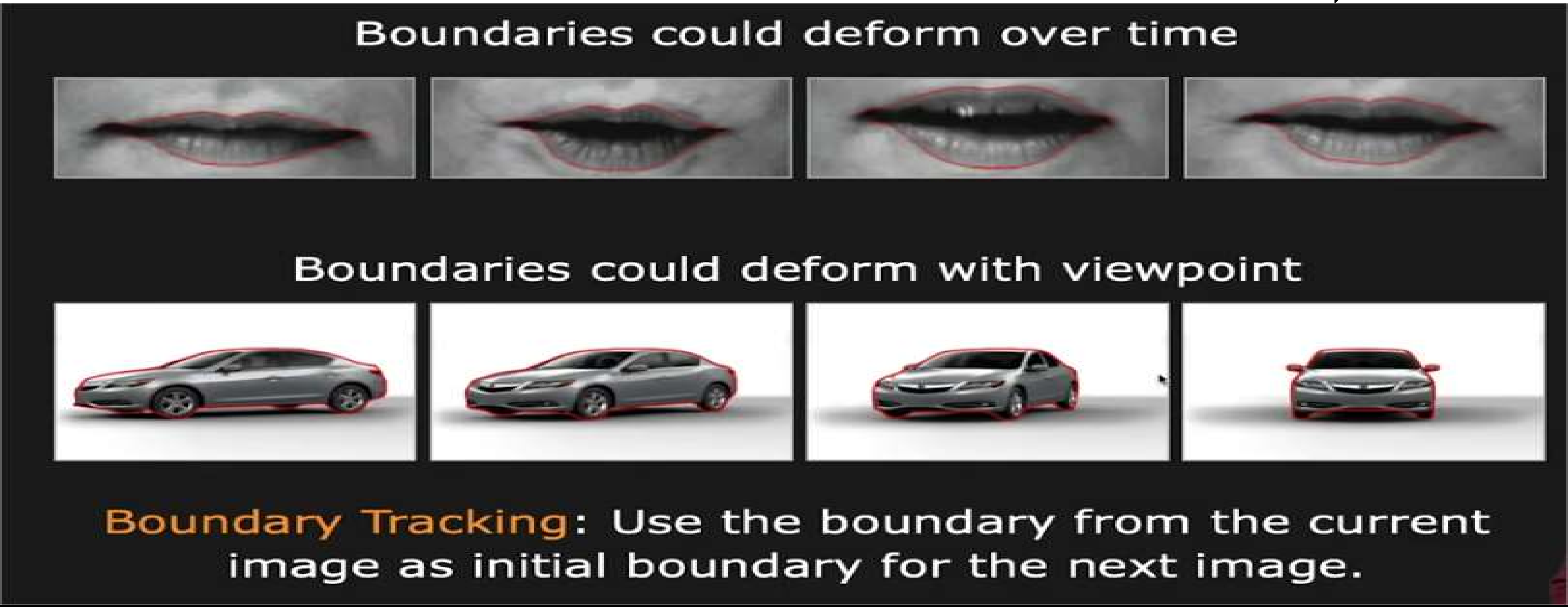
Active Contour:

Iteratively "deform" the initial contour so that:

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

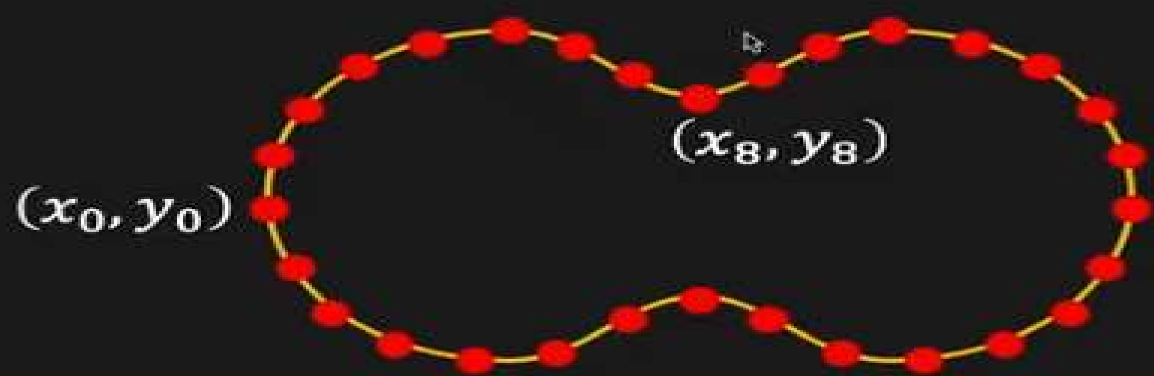
Also called Snakes

Deformable countours



Representing a contours

Contour **v**: An ordered list of 2D vertices (control points) connected by straight lines of fixed length



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

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 Gradient Magnitude Square

≡ Minimize −ve (Sum of Gradient Magnitude Square)

≡ 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 $(i = 0, \ldots, n-1)$, 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 and slow.

Sensitivity to noise and initialization


Contour fitted to gradient magnitude


Contour fitted to gradient magnitude

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

Making contour 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

For point $0 \leq s \leq 1$ on continuous contour $\mathbf{v}(s) = (x(s), y(s))$:

$$\boxed{E_{elastic} = \left\|\frac{d\mathbf{v}}{ds}\right\|^2} \qquad \boxed{E_{smooth} = \left\|\frac{d^2\mathbf{v}}{ds^2}\right\|^2}$$

Discrete approximations at control point $\mathbf{v}_i$:

$$E_{elastic}(\mathbf{v}_i) = \left\|\frac{d\mathbf{v}}{ds}\right\|^2 \approx \|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

$$E_{smooth}(\mathbf{v}_i) = \left\|\frac{d^2\mathbf{v}}{ds^2}\right\|^2 \approx \|(\mathbf{v}_{i+1} - \mathbf{v}_i) - (\mathbf{v}_i - \mathbf{v}_{i-1})\|^2$$

$$= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2$$

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 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 Active Contour:

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

Minimize the Total Energy

Counter Deformation : Greedy algorithm

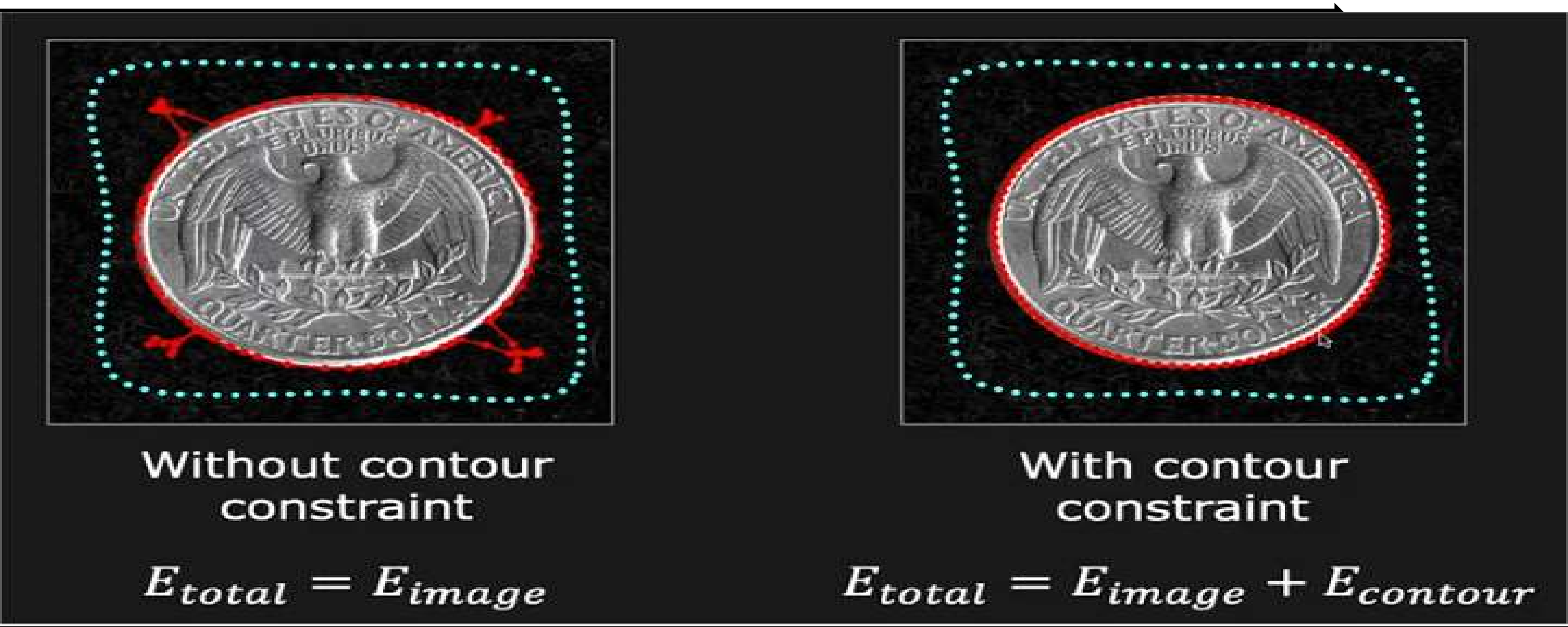
1. Uniformly sample the contour to get n contour points.

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.

Results : Effects of contour constraints

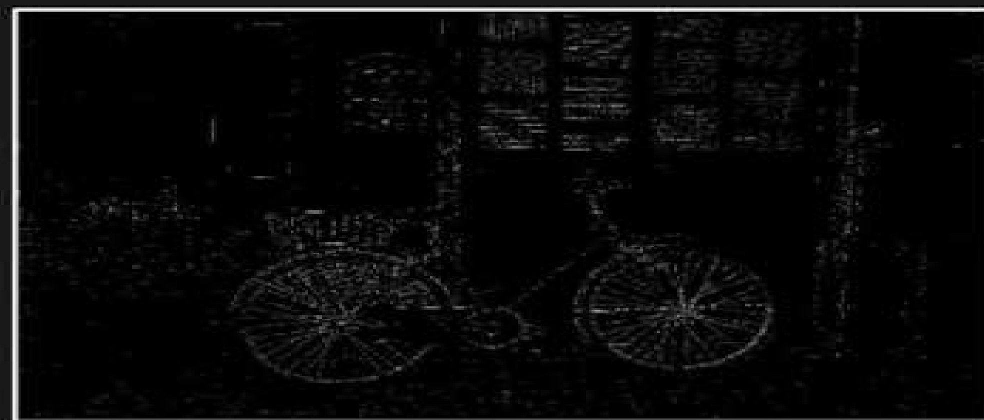


Active Contours: conclusion

- 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

Medical Image Segmentation

Line detectors (Hough Transform)



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

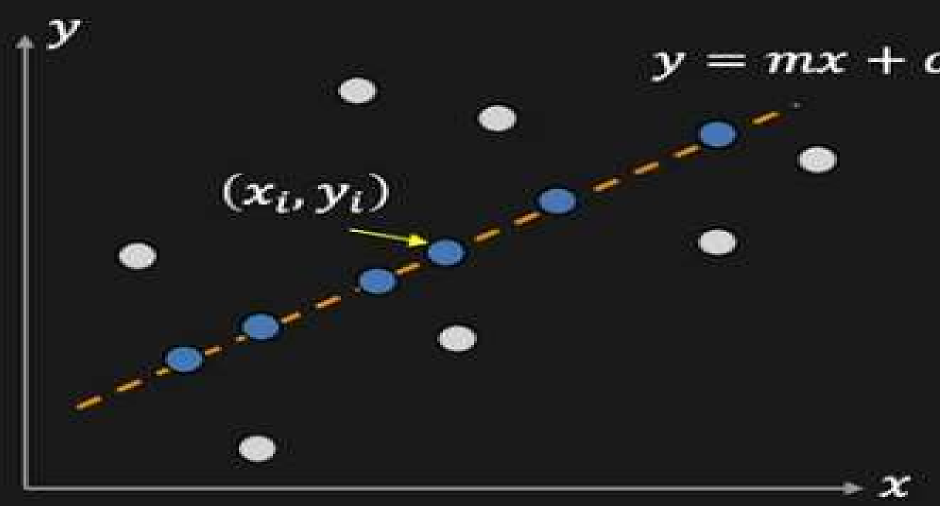
Solution: Hough Transform

Hough Transform: concept

Given: Edge Points (x_i, y_i)

Task: Detect line

$$y = mx + c$$



Consider point (x_i, y_i)

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

Concept

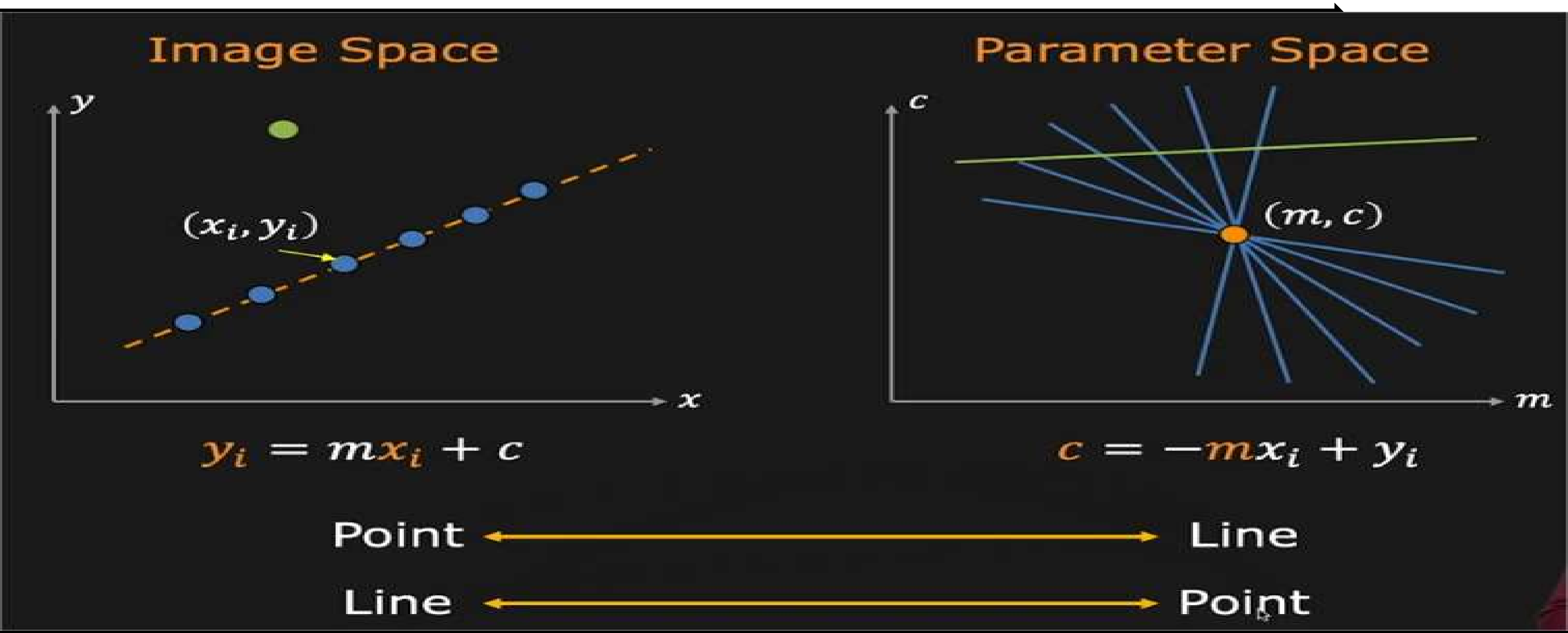


Image Space
Parameter Space
y
x
(x_i, y_i)
c
m
(m, c)
$y_i = mx_i + c$
$c = -mx_i + y_i$
Point
Line
Line
Point

Hough Transform : 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)

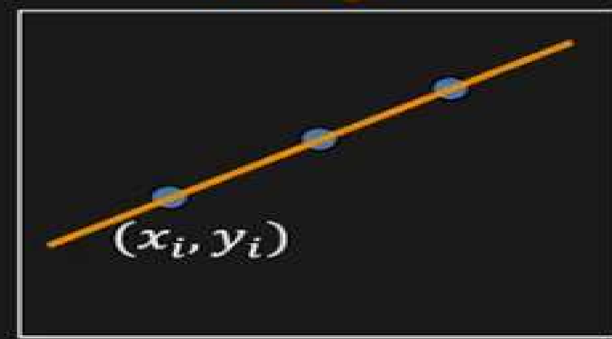
Step 4. For each edge point (x_i, y_i),

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

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

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

Image



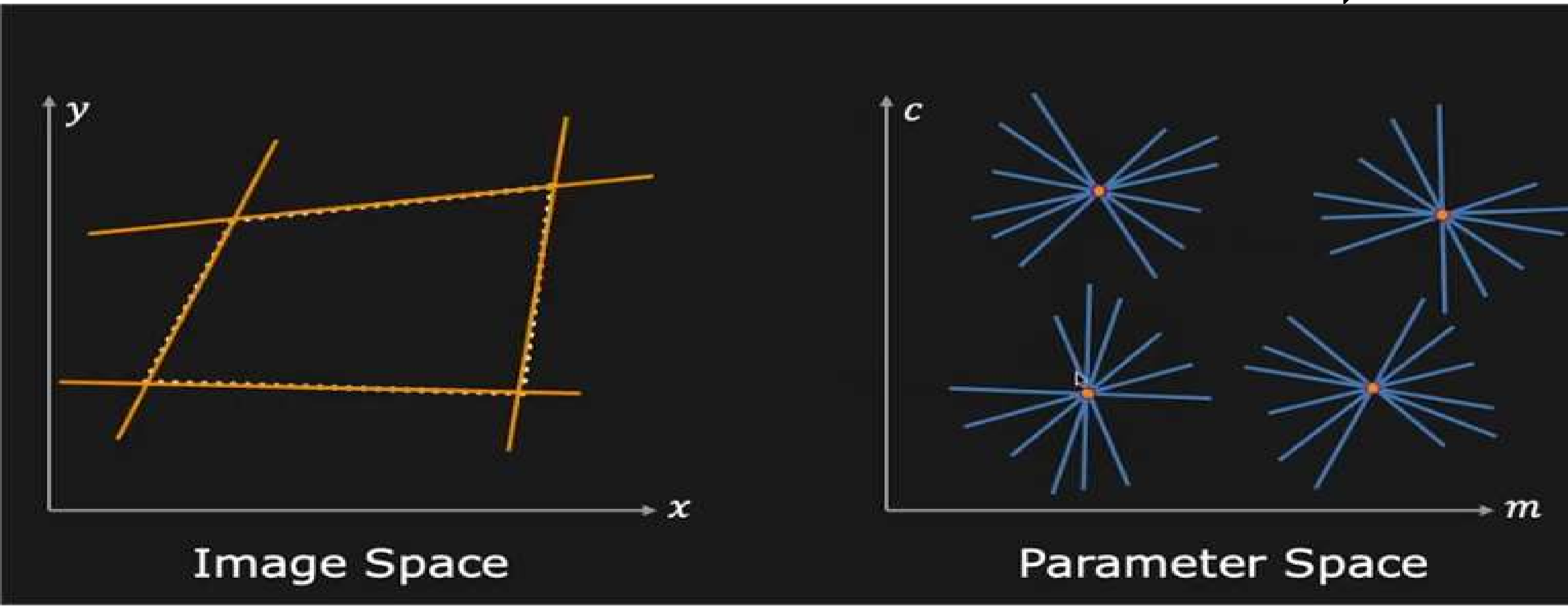
$A(m, c)$

c

1	0	0	0	1
0	1	0	1	0
1	1	3	1	1
0	1	0	1	0
1	0	0	0	1

m

Multiple Line detection



Better parameterization

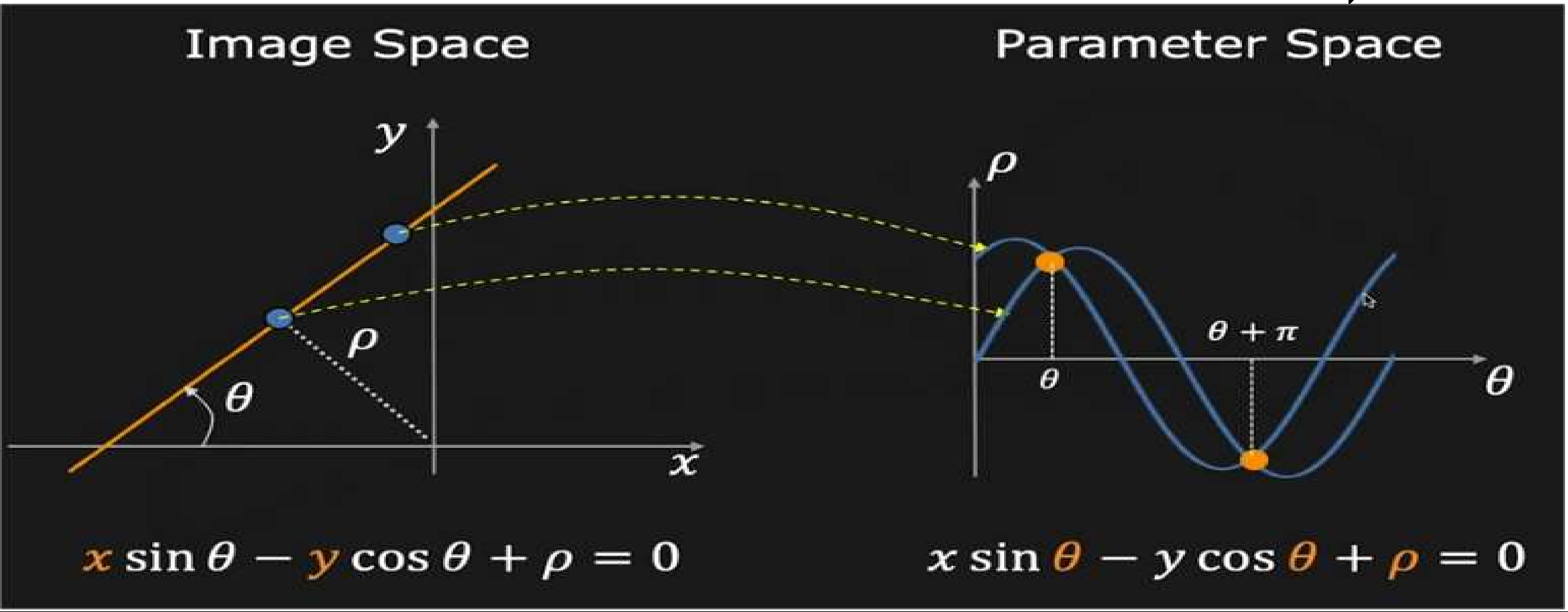
Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation

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

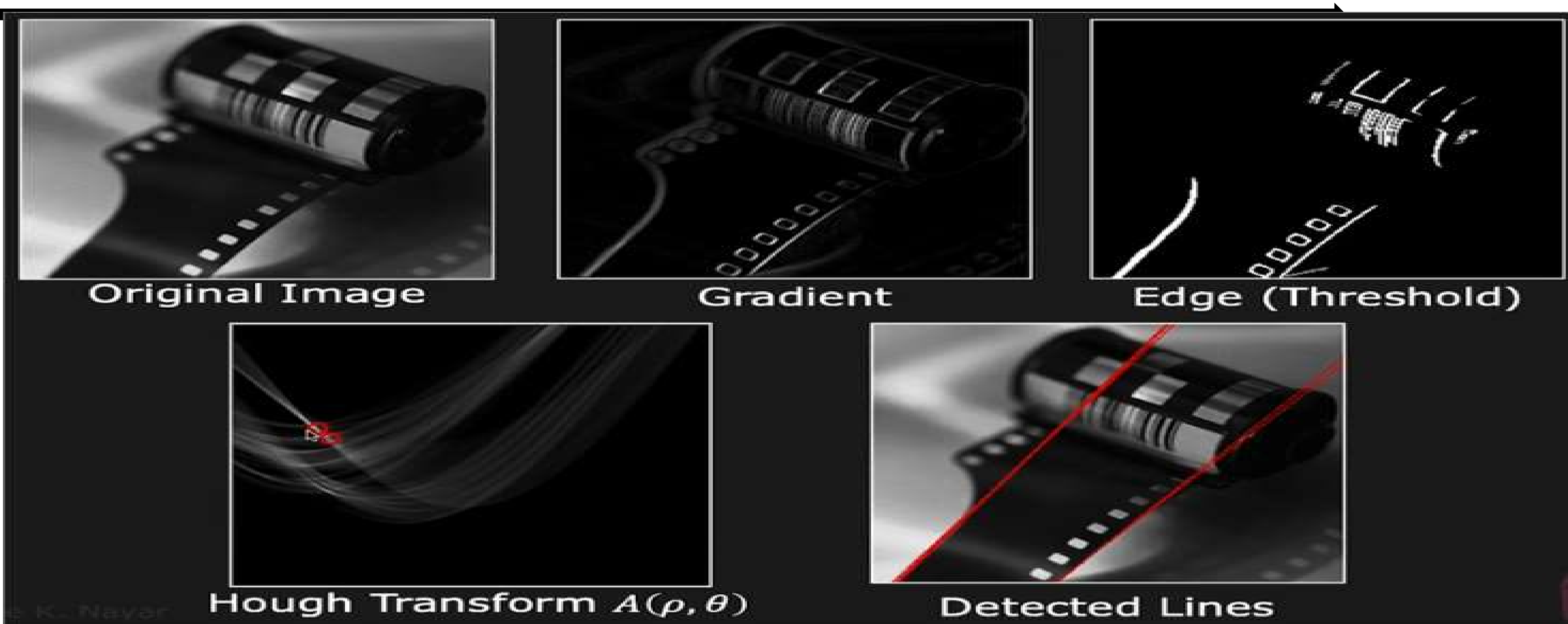
- Orientation θ is finite: $0 \leq \theta < \pi$
- Distance ρ is finite

Better parameterization

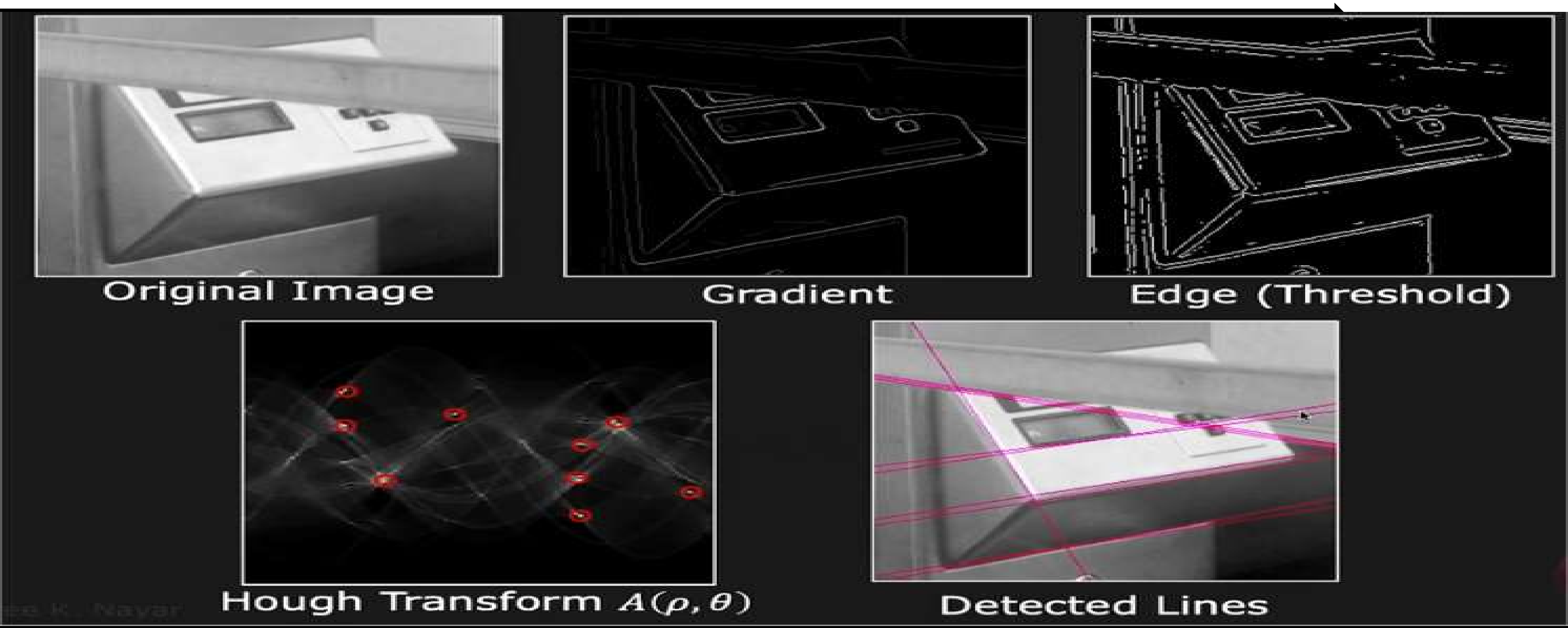


Hough Transform Mechanics

- 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

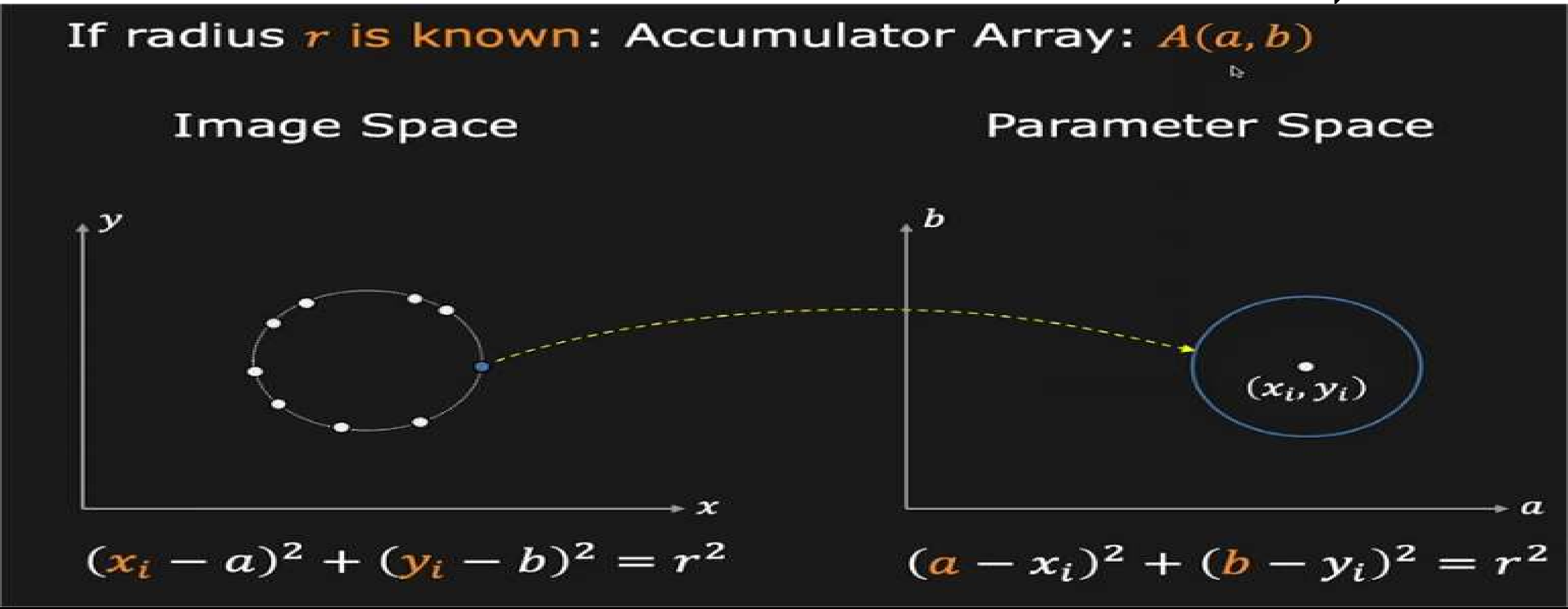

Original Image
Gradient
Edge (Threshold)
Hough Transform $A(\rho, \theta)$
Detected Lines

Results

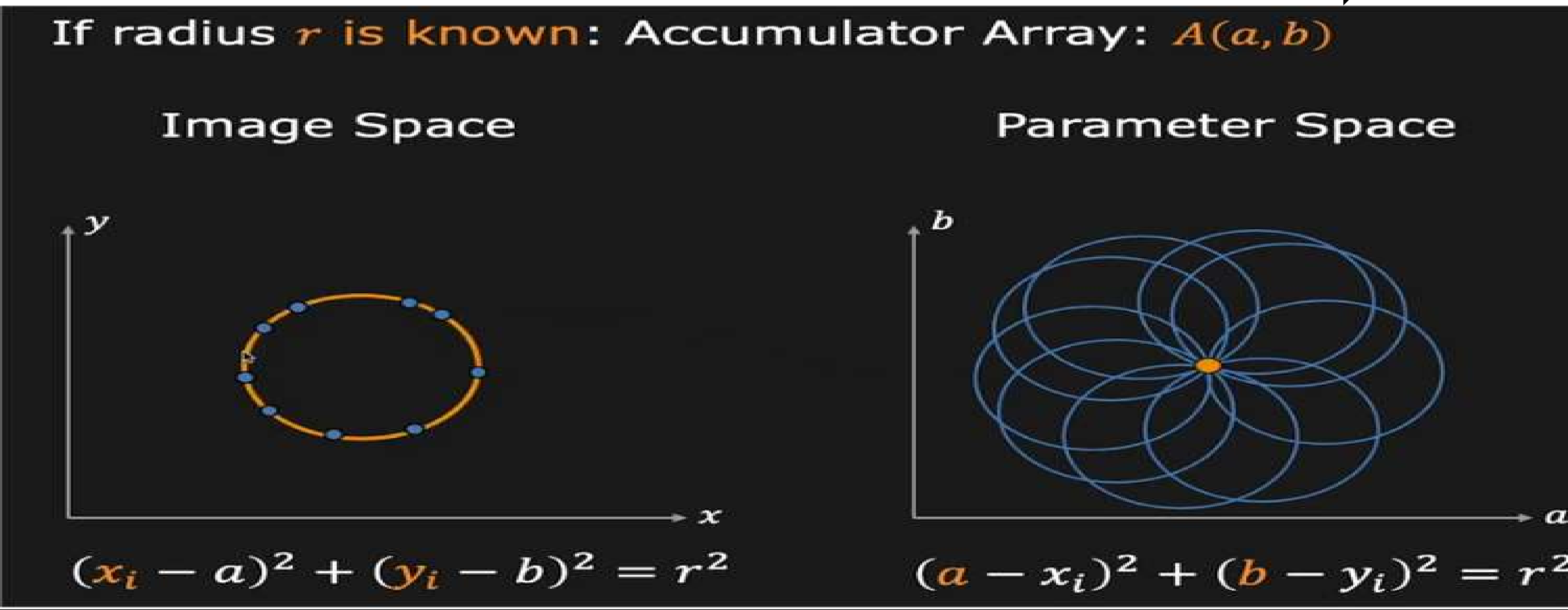


Original Image
Gradient
Edge (Threshold)
Hough Transform $A(\rho, \theta)$
Detected Lines

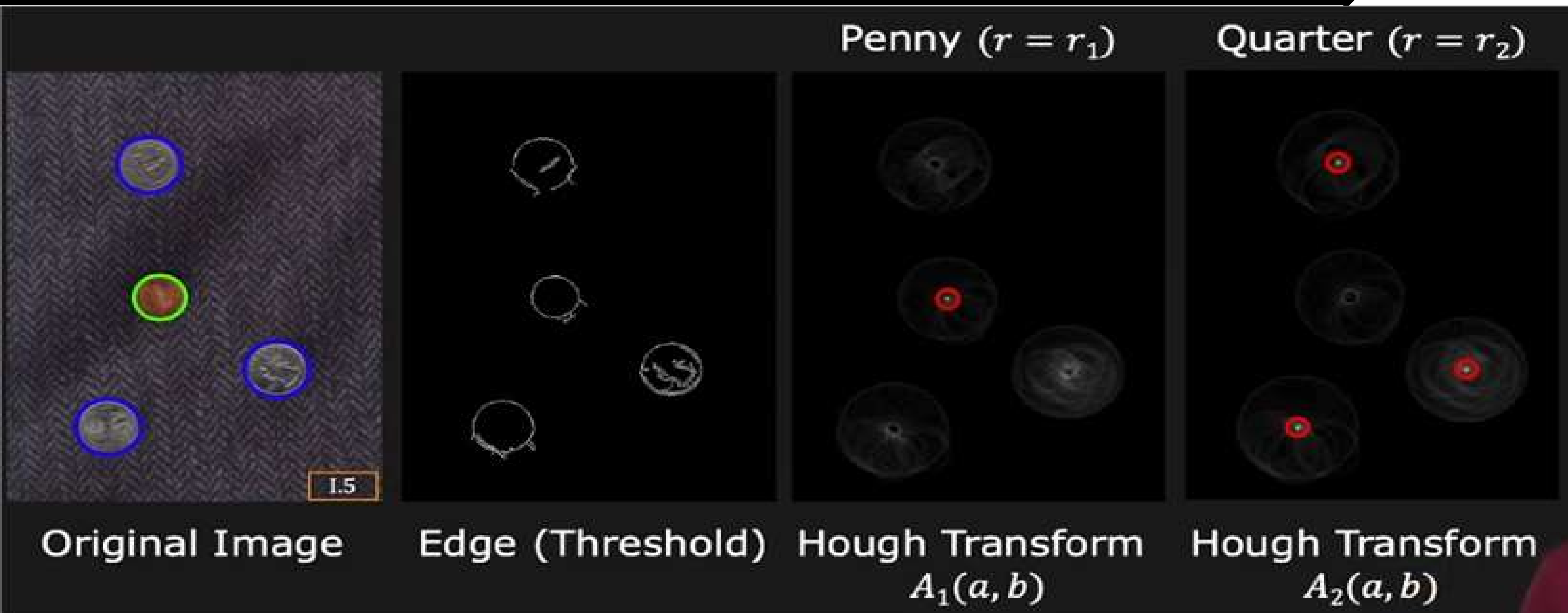
Hough Transform: circle detection



Circle detection

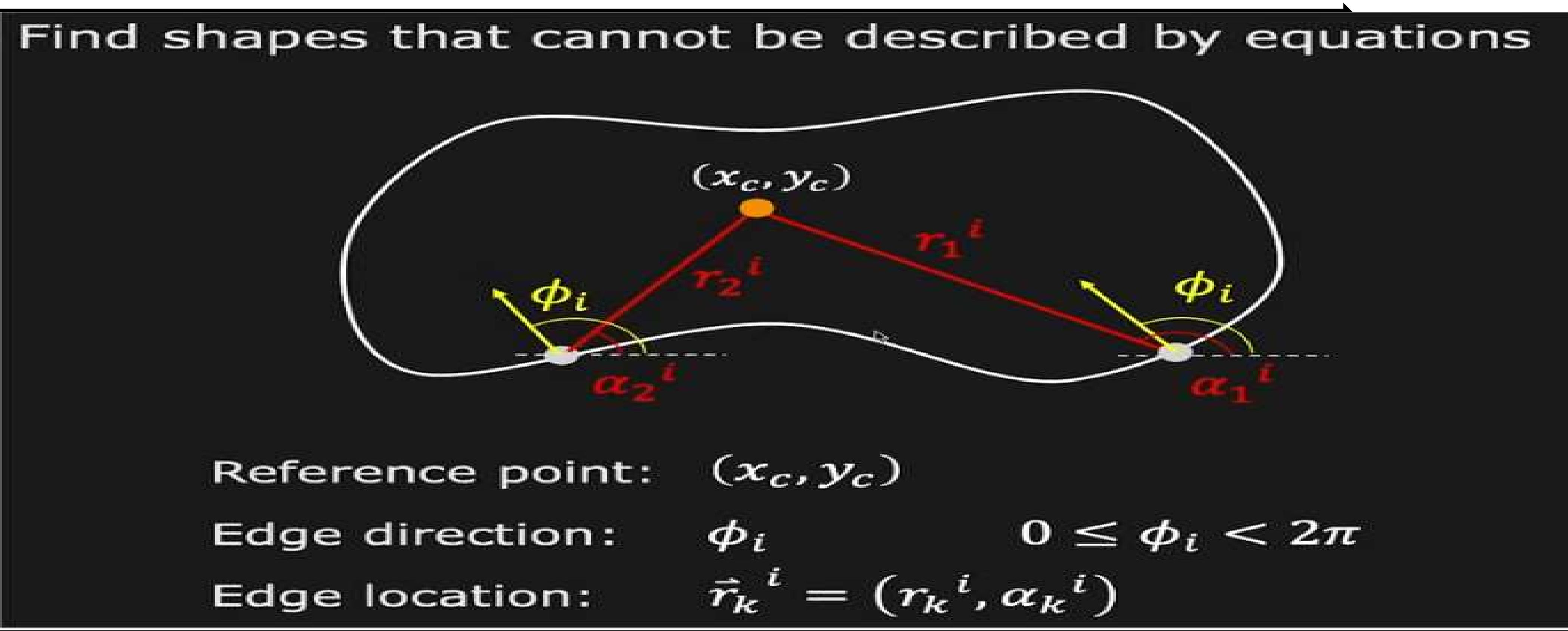


Results

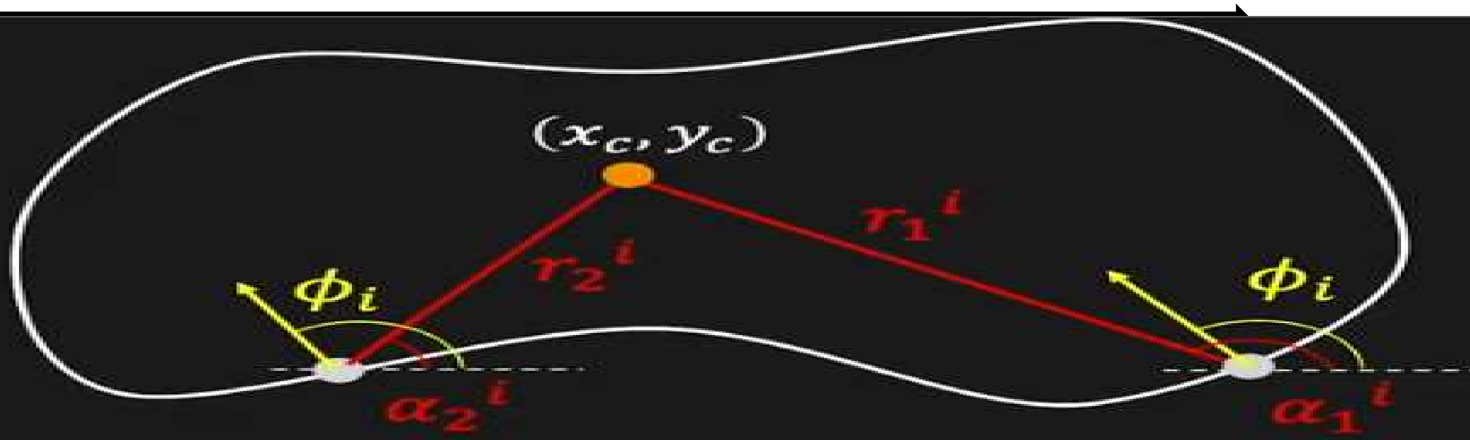


Penny $(r = r_1)$
Quarter $(r = r_2)$
1.5
Original Image
Edge (Threshold)
Hough Transform $A_1(a, b)$
Hough Transform $A_2(a, b)$

Generalized Hough transform



Hough Model



ϕ-Table:

Edge Direction	$\vec{\mathbf{r}} = (\mathbf{r}, \alpha)$
ϕ_1	$\vec{r}_1{}^1, \vec{r}_2{}^1, \vec{r}_3{}^1$
ϕ_2	$\vec{r}_1{}^2, \vec{r}_2{}^2$
$\vdots$	$\vdots$
ϕ_n	$\vec{r}_1{}^n, \vec{r}_2{}^n, \vec{r}_3{}^n, \vec{r}_4{}^n$

- 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),

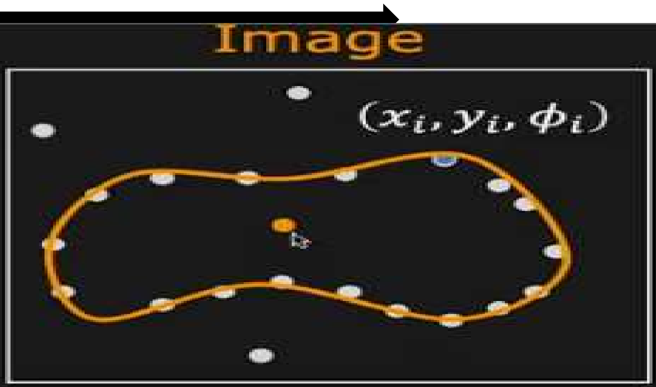
  For each entry $\phi_i \rightarrow \vec{r}_k{}^i$ in $\phi -$ table,

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

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

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

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



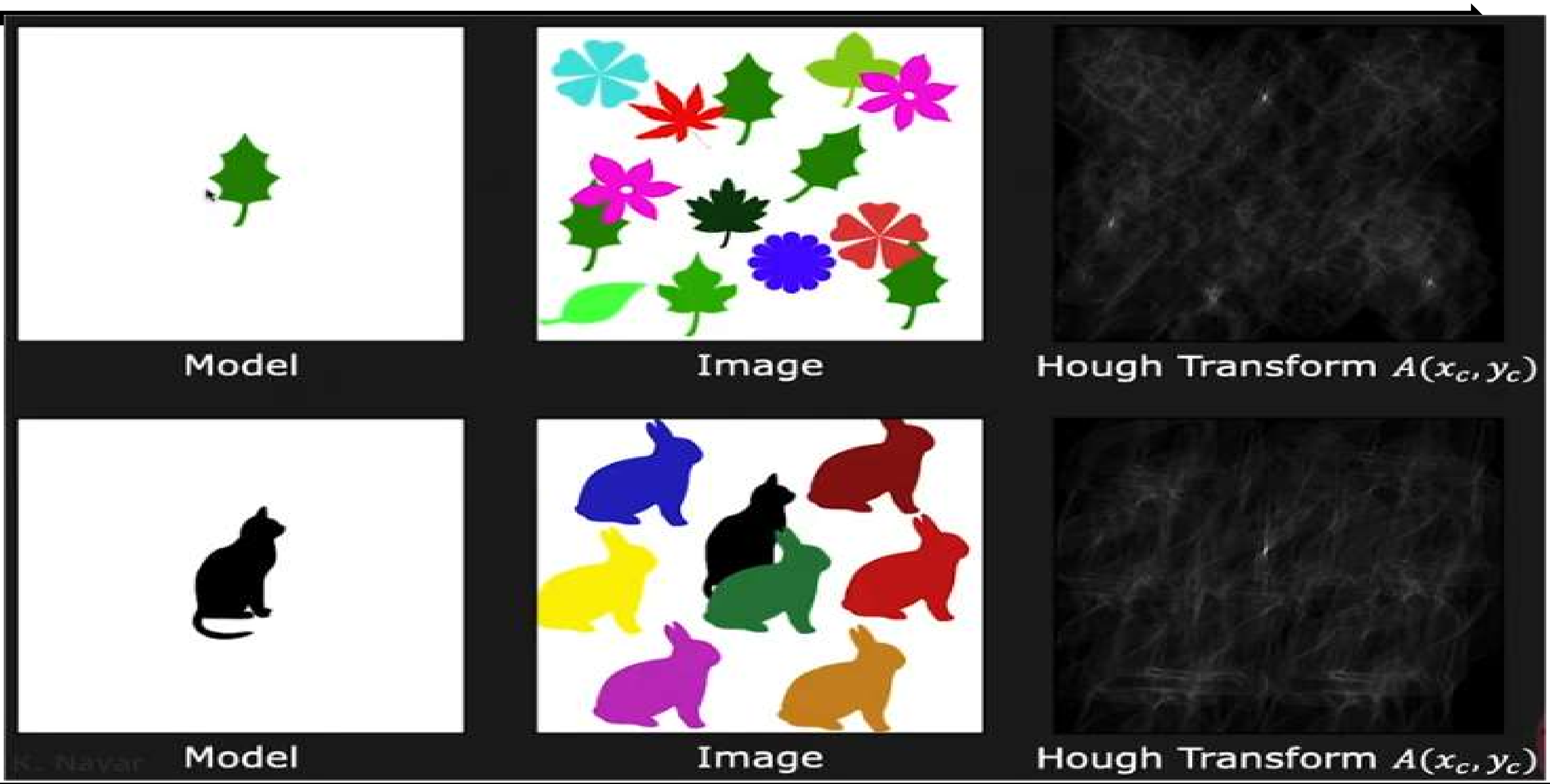
$A(x_c, y_c)$

x_c

0	0	0	0	0
0	2	0	1	0
0	0	4	1	0
0	2	0	0	0
0	0	0	1	0

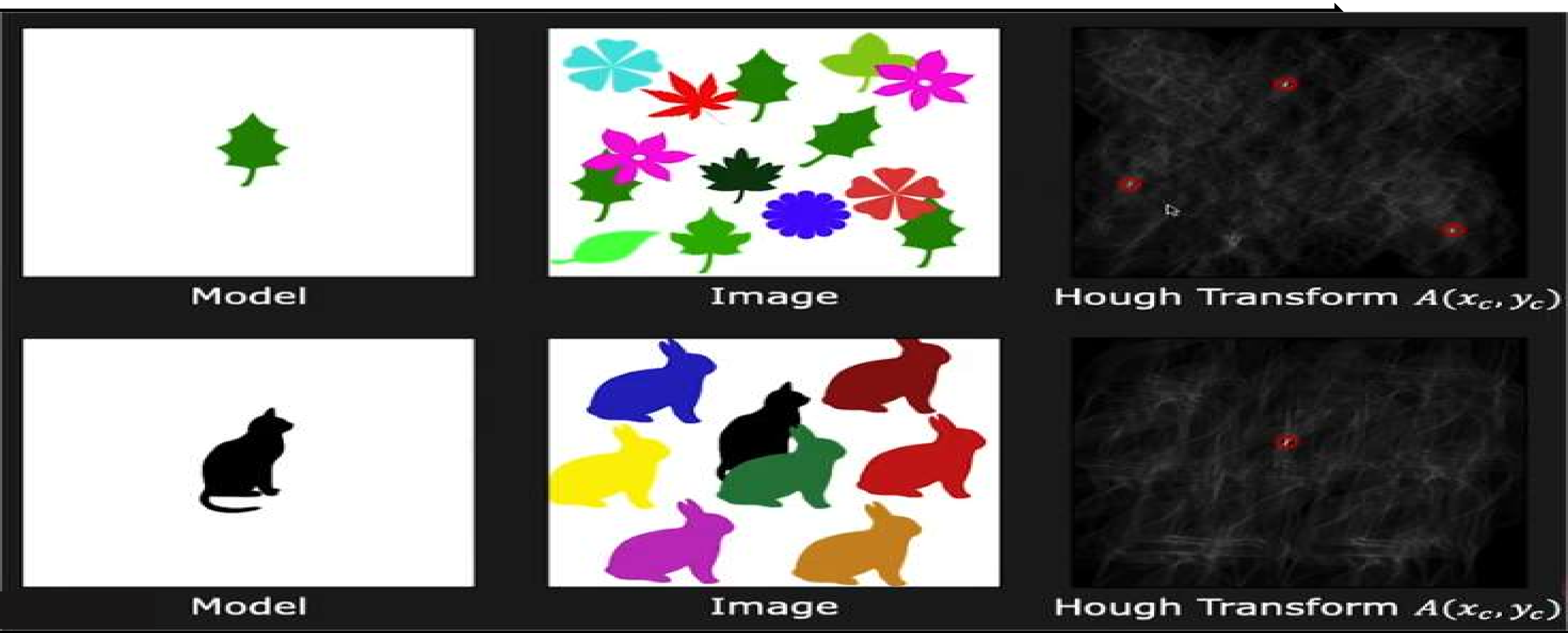
y_c

Results

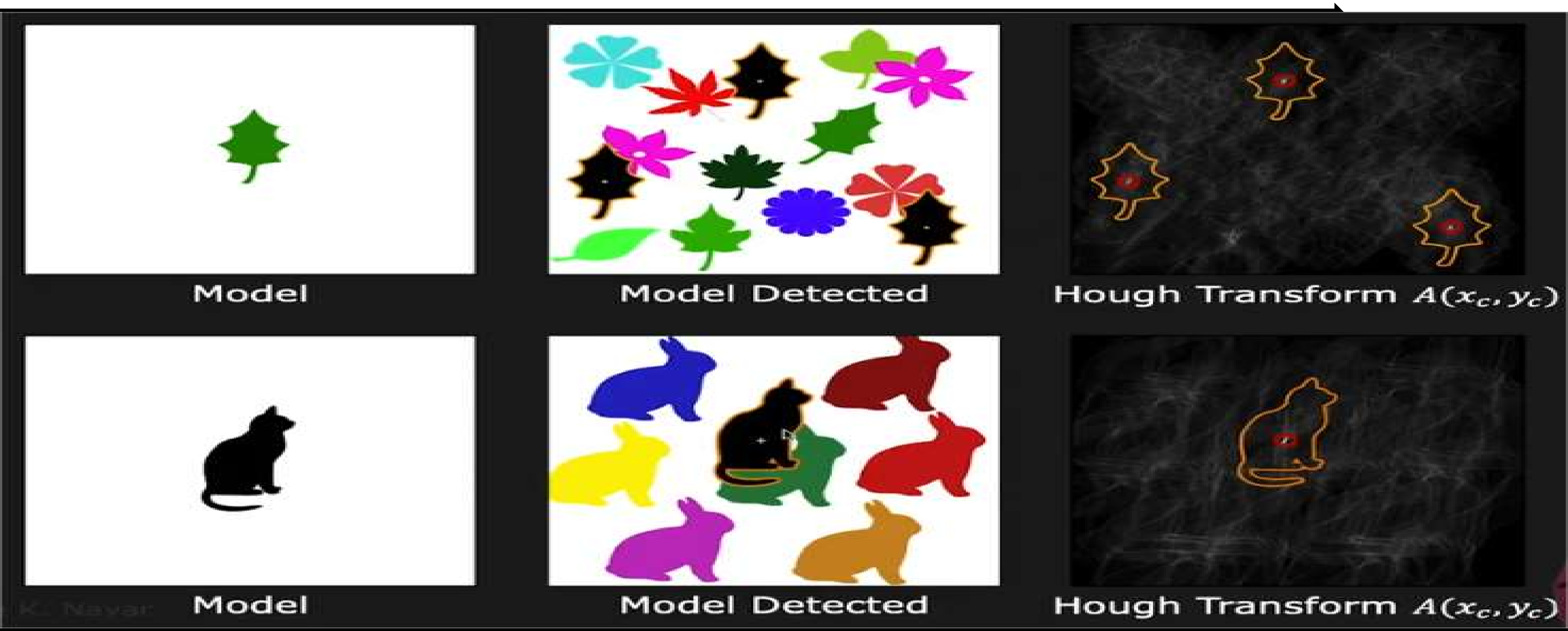


Model
Image
Hough Transform $A(x_c, y_c)$
Model
Image
Hough Transform $A(x_c, y_c)$

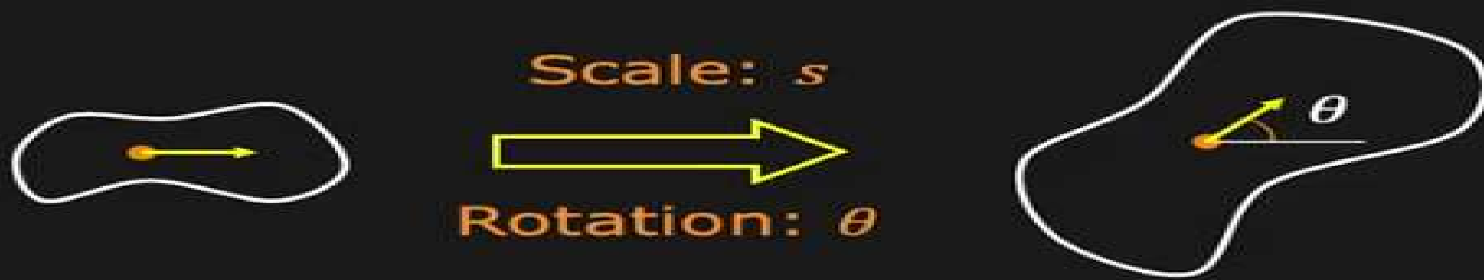
Results



Model
Image
Hough Transform $A(x_c, y_c)$
Model
Image
Hough Transform $A(x_c, y_c)$


Model
Model Detected
Hough Transform $A(x_c, y_c)$
Model
Model Detected
Hough Transform $A(x_c, y_c)$

Scale and Rotation



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

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

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

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

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.)
- Complex Shapes: Generalized Hough Transform
- Trade-off between work in image space and parameter space