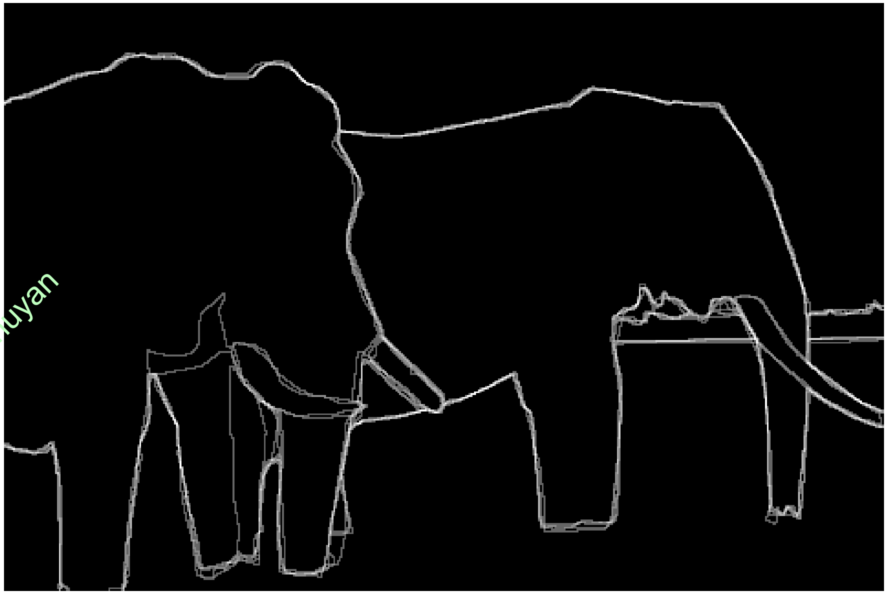
Boundary Detection: Hough Transform

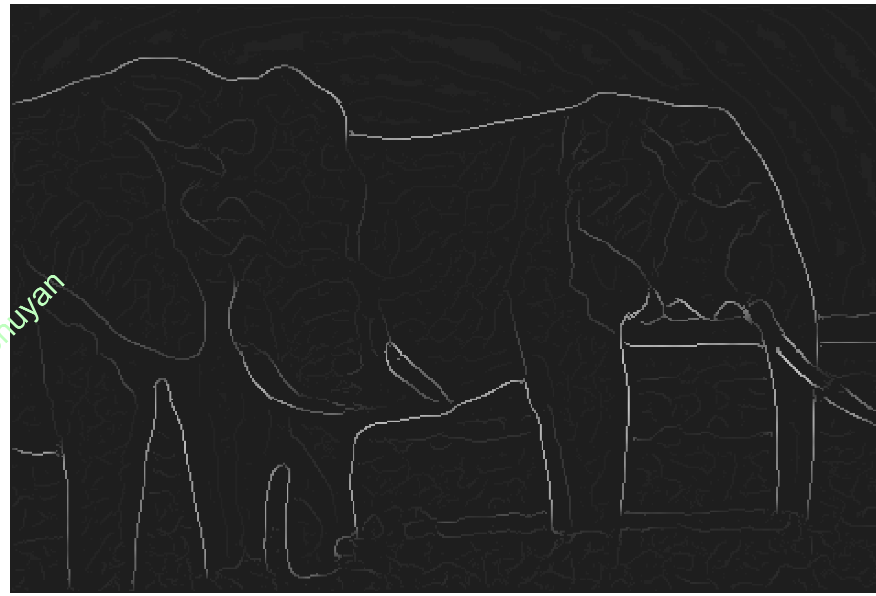
Boundaries of Objects



Marked by many users

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/bench/html/images.html

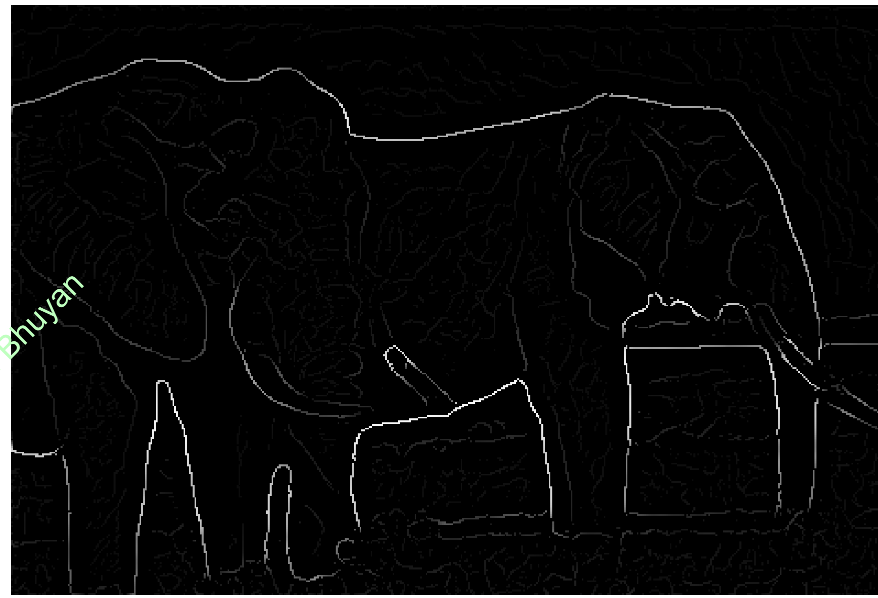
Boundaries of Objects from Edges



Brightness Gradient (Edge detection)

- Missing edge continuity, many spurious edges

Boundaries of Objects from Edges



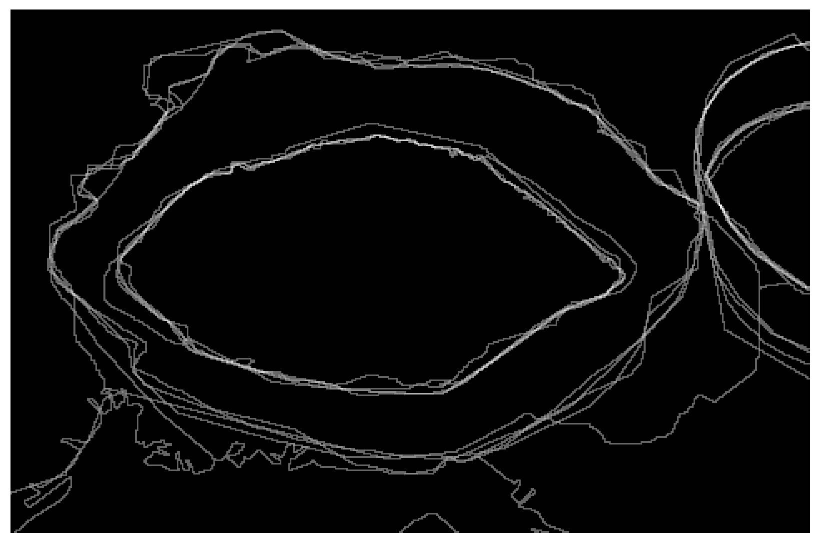
Multi-scale Brightness Gradient

- But, low strength edges may be very important

Boundaries of Objects from Edges

Image

Machine Edge Detection



Human Boundary Marking

Boundaries in Medical Imaging

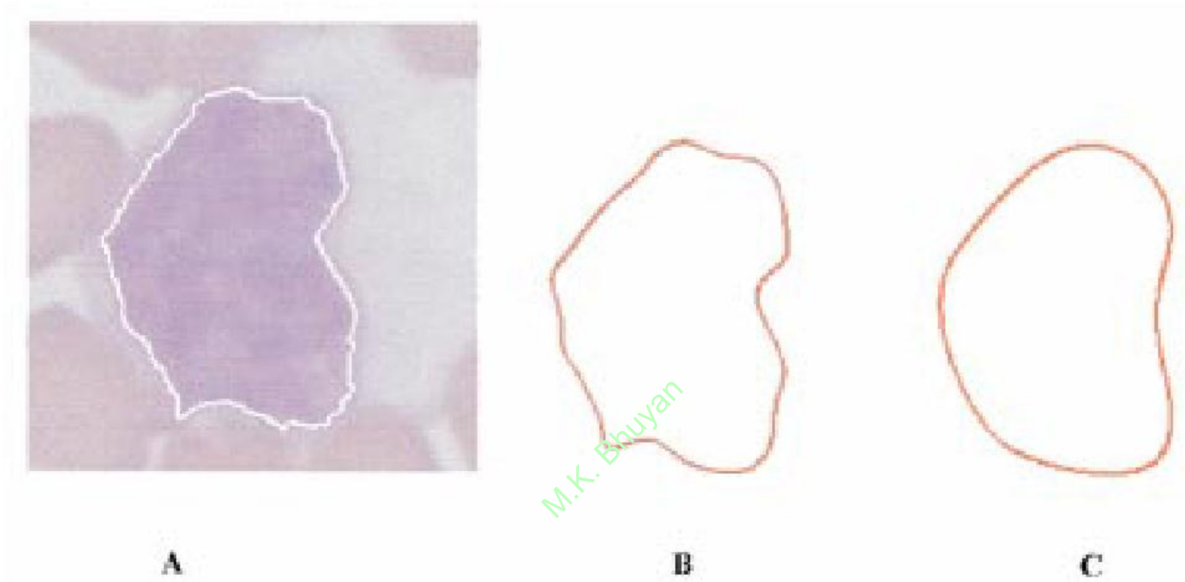
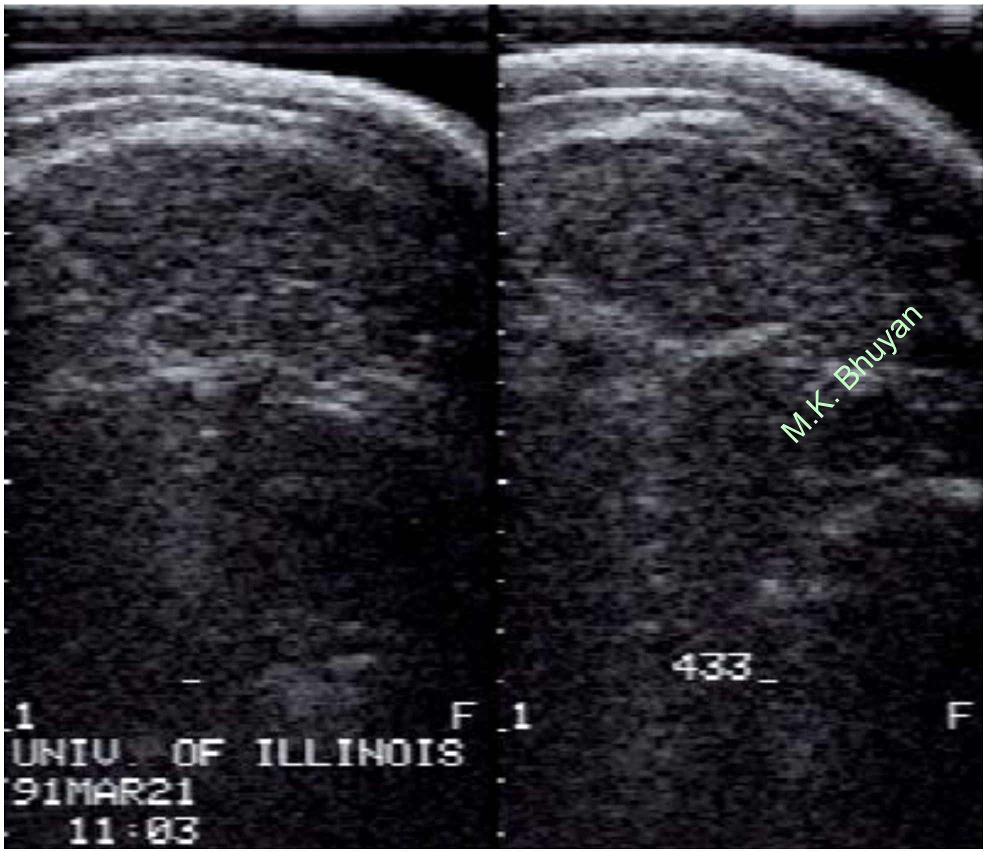
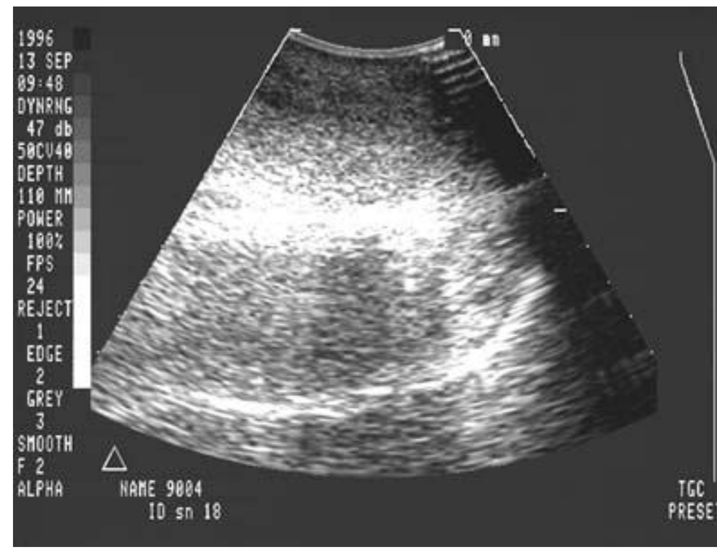


Fig. 2. Representation of a closed contour by elliptic Fourier descriptors. (a) Input. (b) Series truncated at 16 harmonics. (c) Series truncated to four harmonics.

Detection of cancerous regions.

Boundaries in Ultrasound Images



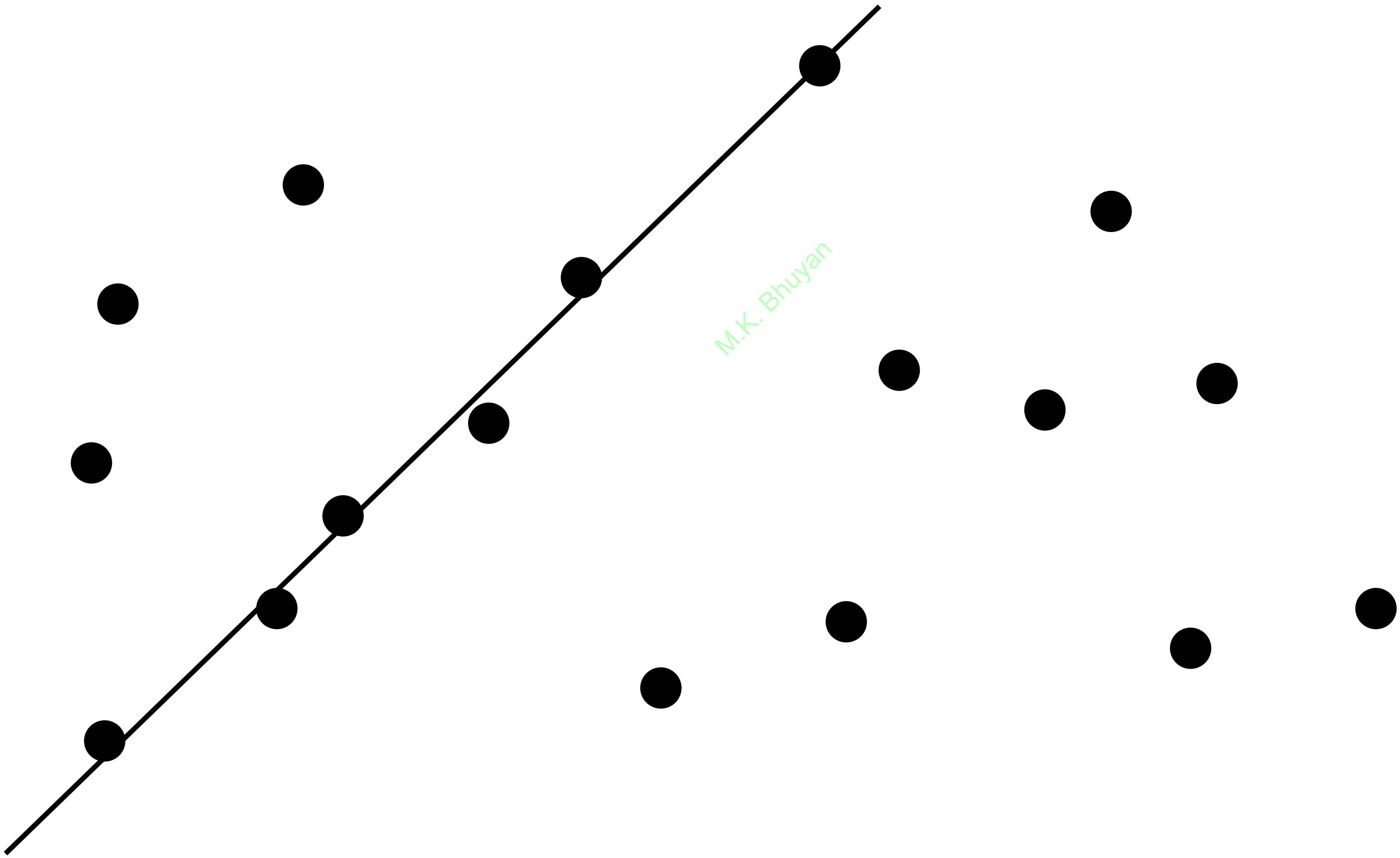


Hard to detect in the presence of large amount of speckle noise

Boundaries of Objects

Sometimes hard even for humans!

Line Grouping Problem



Line Detection and Hough transform

- Many edges can be approximated by straight lines.

- For n edge pixels, there are $\dfrac{n(n-1)}{2}$ possible lines.

- To find whether a point is closer to a line we have to perform $n \times \dfrac{n(n-1)}{2}$ comparisons. Thus, a total of $0(n^3)$ comparisons.

Hough Transform

- Elegant method for direct object recognition
    - Edges need not be connected
    - Complete object need not be visible
    - Key Idea: Edges VOTE for the possible model

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

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

Parameter space also called Hough Space

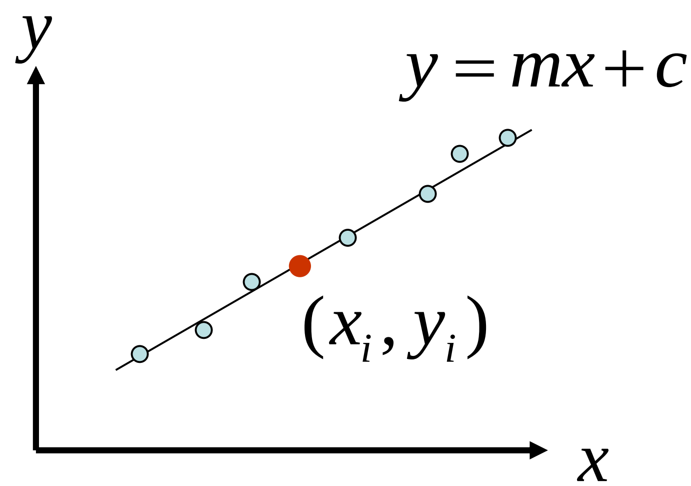
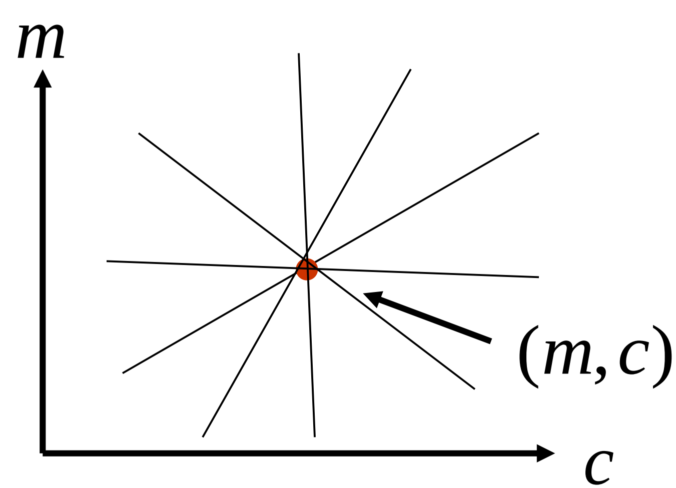
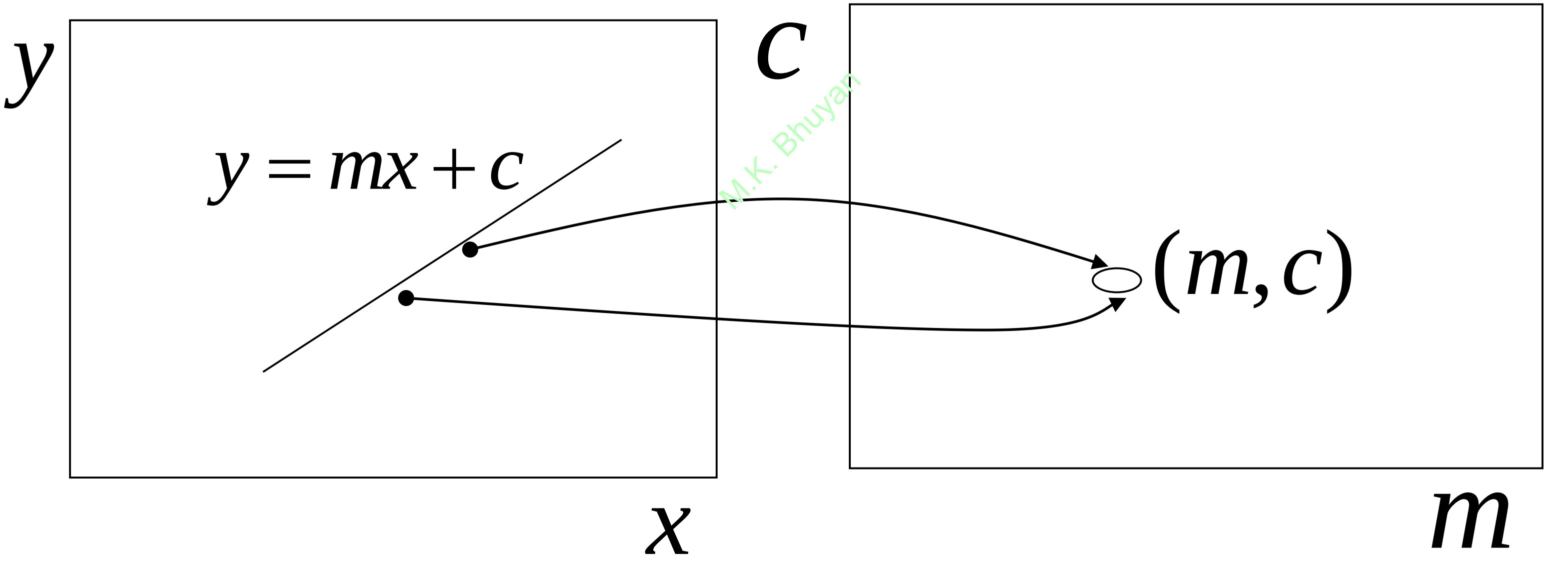


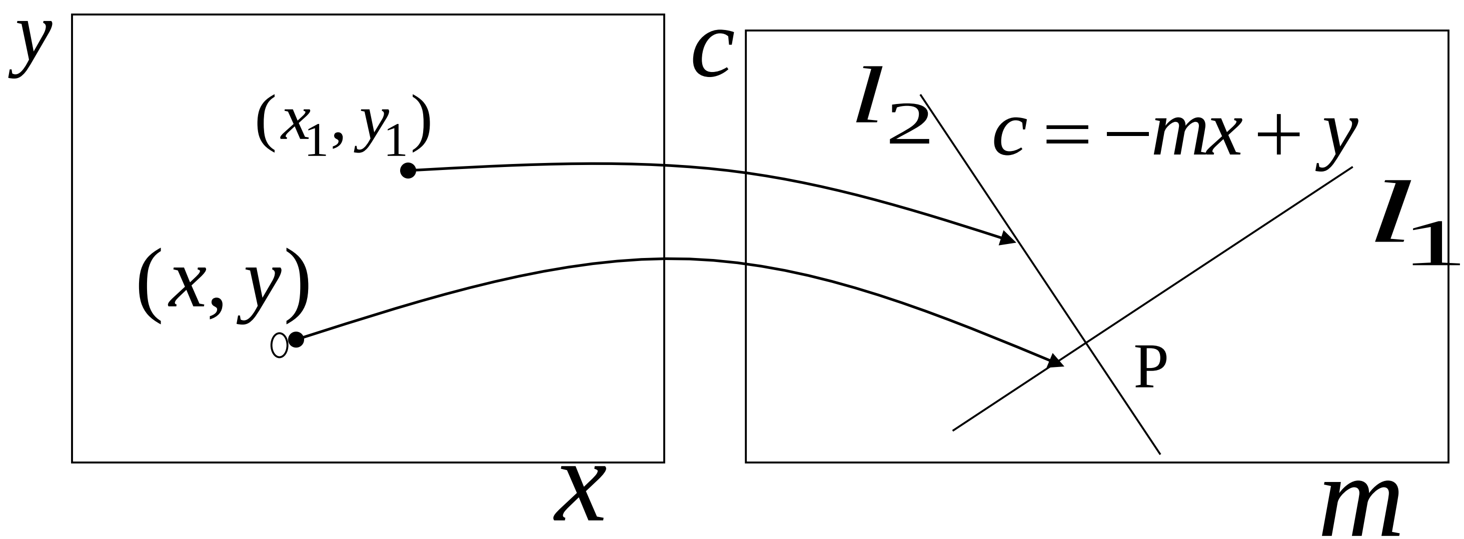
Image Space



Parameter Space

- **Hough transform** uses parametric representation of a straight line for line detection.





- The points (x, y) and (x_1, y_1) are mapped to lines l_1 and l_2 respectively in $m-c$ space
- l_1 and l_2 will intersect at a point P representing the (m, c) values of the line joining (x, y) and (x_1, y_1).
- The straight line map of another point collinear with these two points will also intersect at P. The intersection of multiple lines in the *m*-*c* plane will give the *(m,c)* values of lines in the edge image plane.

Line Detection by Hough Transform

Algorithm:

- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m, c) = 0 \quad \forall m, c$
- For each image edge (x_i, y_i) increment:

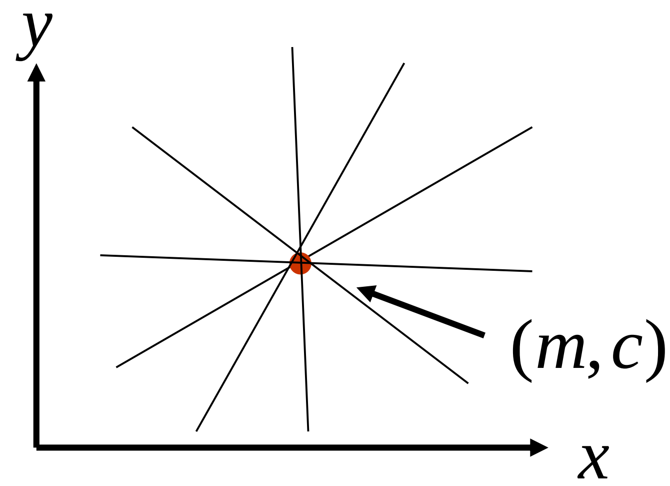
$$A(m, c) = A(m, c) + 1$$

- If (m, c) lies on the line:

$$c = -x_i m + y_i$$

- Find local maxima in $A(m, c)$

Group the edges that belong to each line by traversing each line.



Parameter Space

$A(m, c)$

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

Better Parameterization

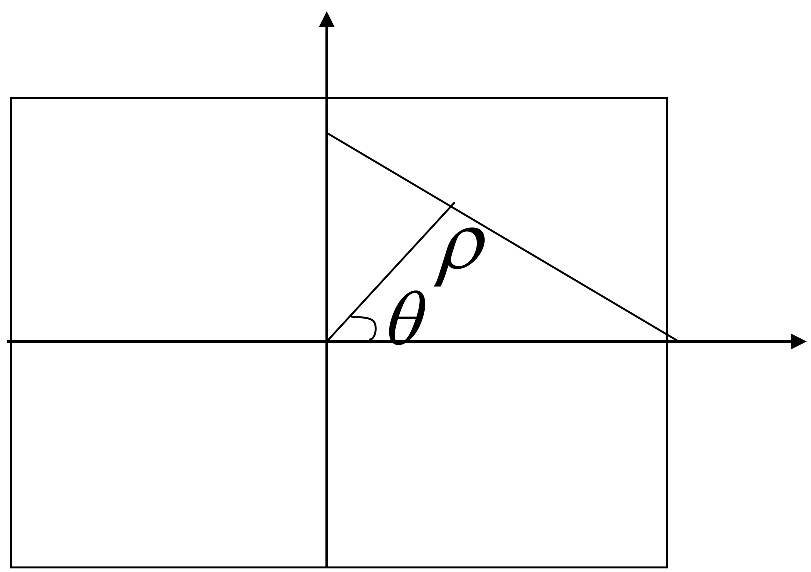
Note: m and c are, in principle, unbounded: cannot handle all situations.

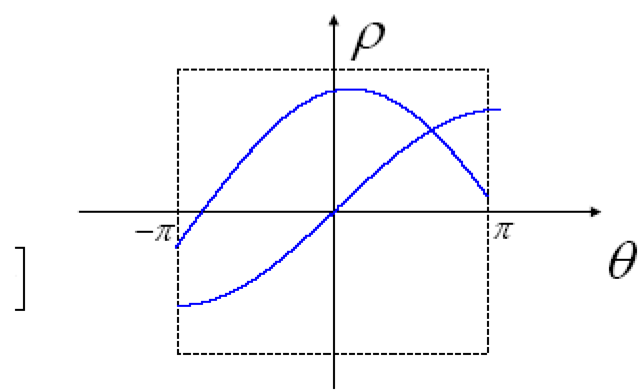
$$-\infty \le m \le \infty$$

Large Accumulator

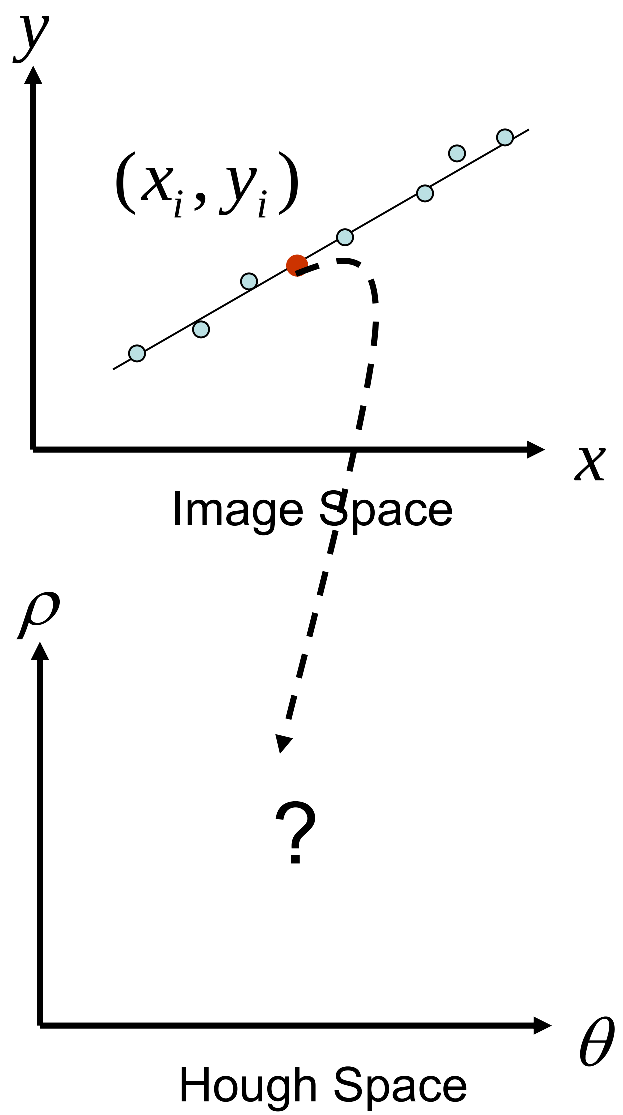
Improvement: (Finite Accumulator Array Size)

- Instead of (m, c), we can consider (ρ, θ) as the parameters with θ varying between -90° and 90° and p varying from 0 to $\sqrt{M^2 + N^2}$ for an $M \times N$ image.





Hough Space is Sinusoid



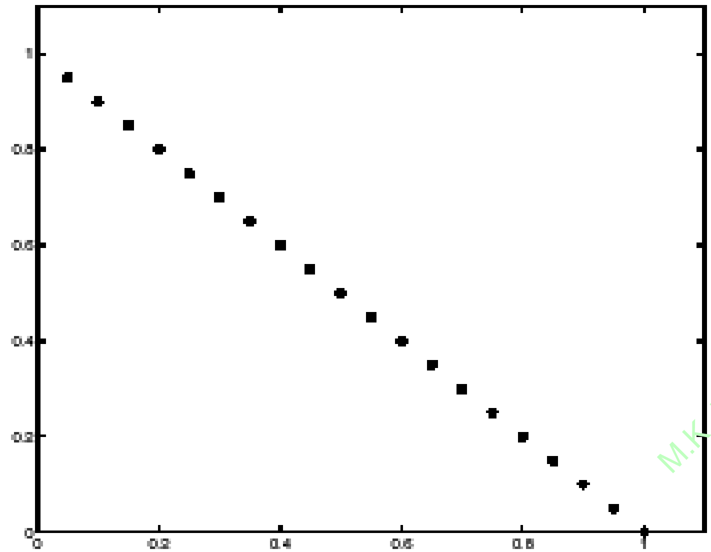
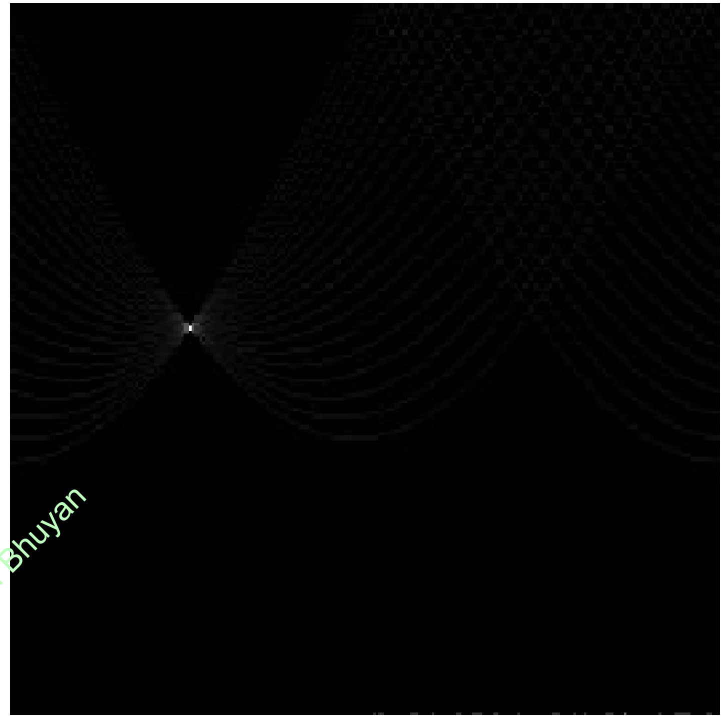


Image space

ρ



Votes θ

Note that most points in the vote array are very dark, because they get only one vote.

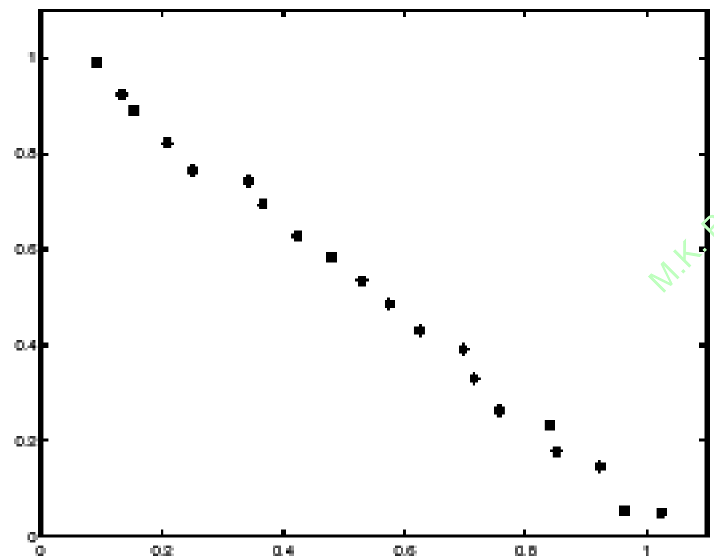
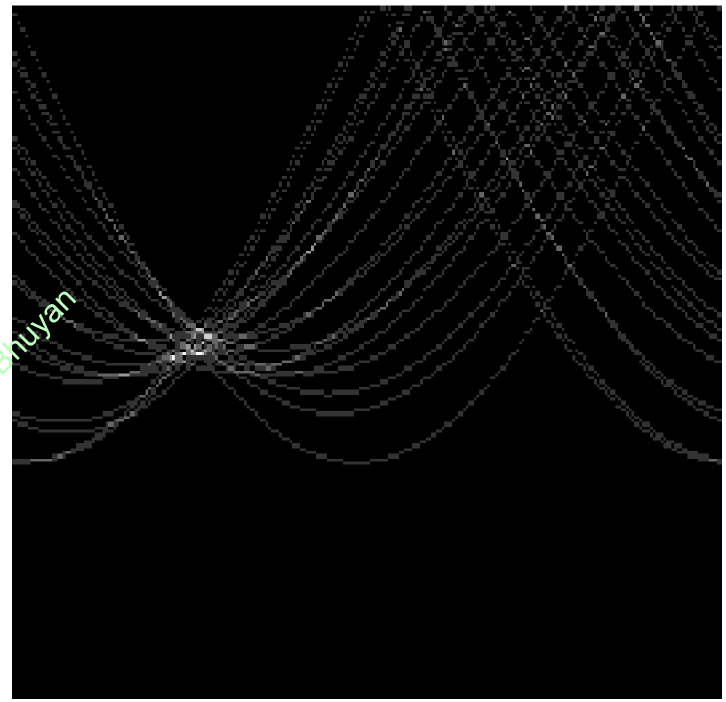
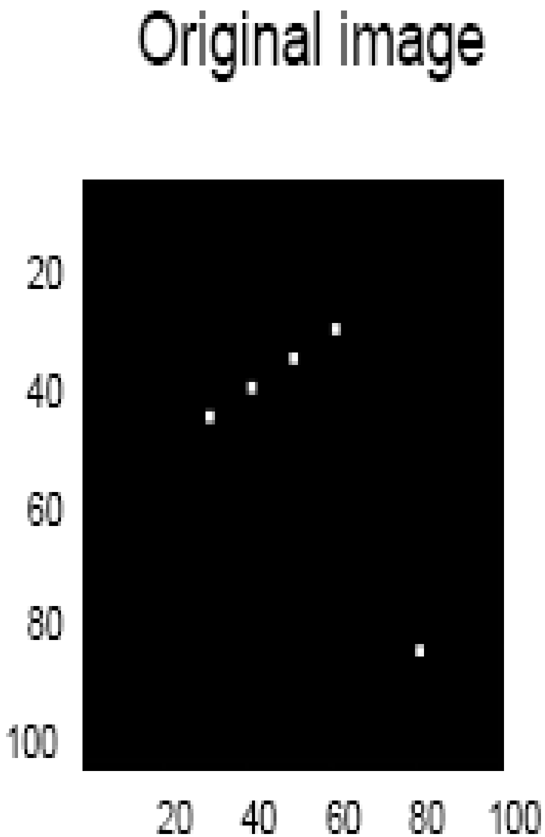
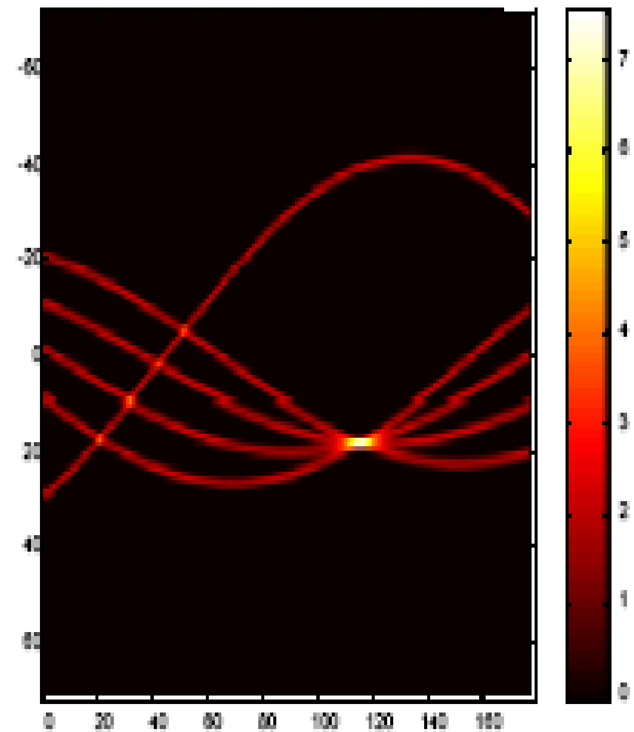


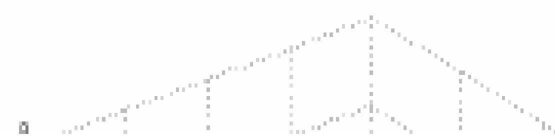
Image space

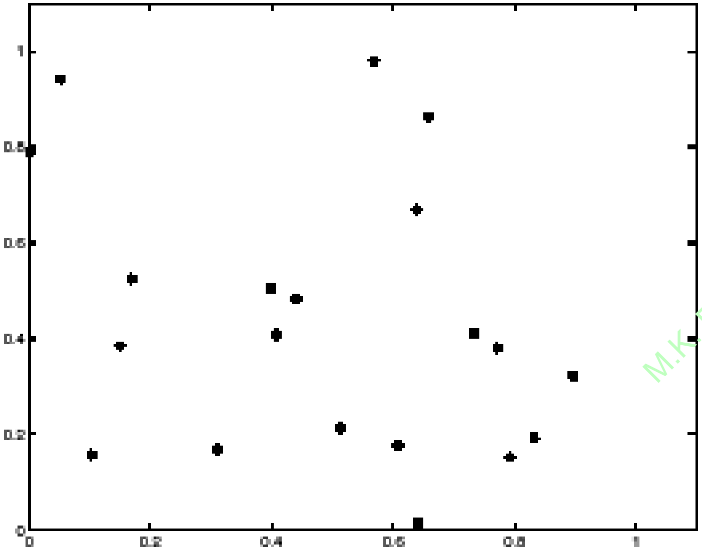


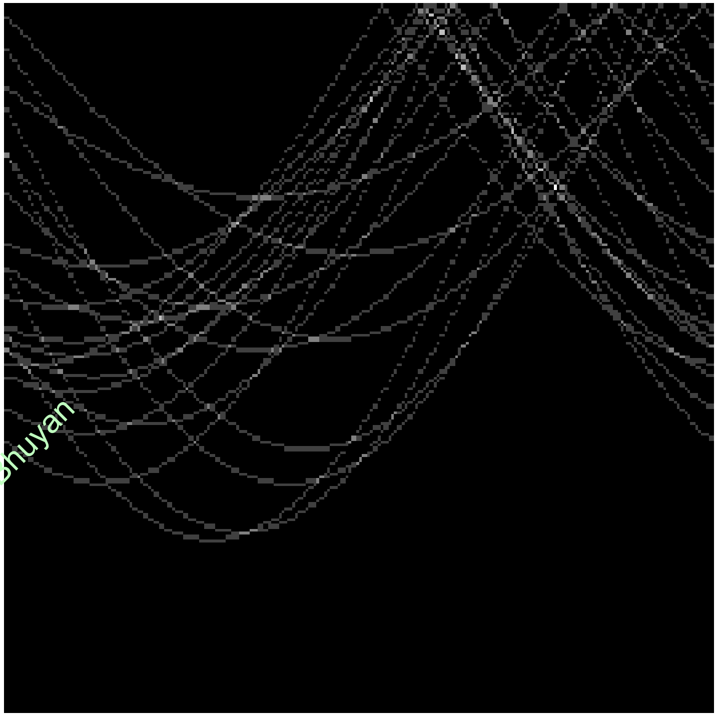
Votes


Original image
20
40
60
80
100
20
40
60
80
100





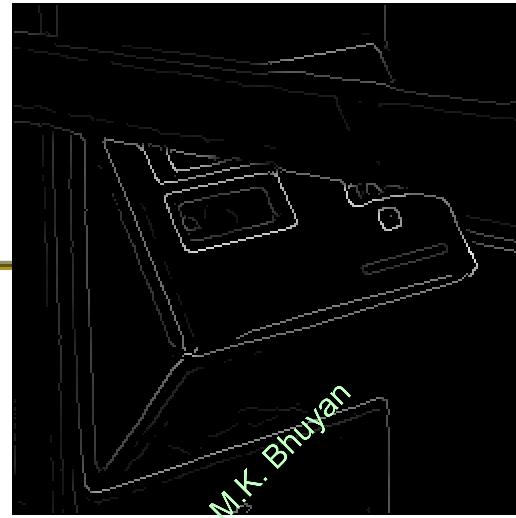




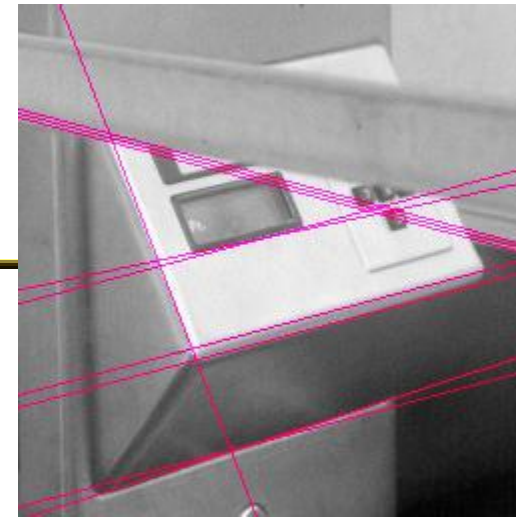
Lots of noise can lead to large peaks in the array

Real World Example

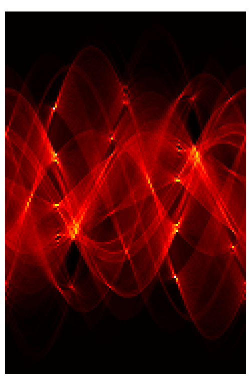
Original



Edge
Detection



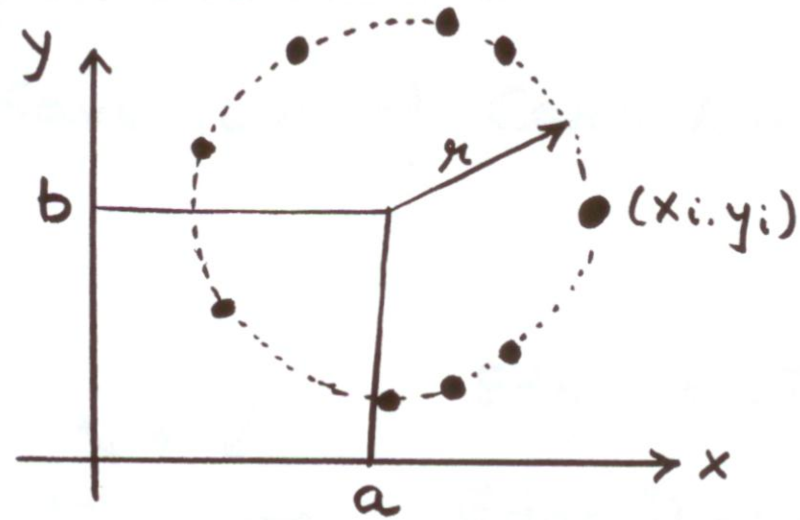
Found Lines



Parameter Space

Finding Circles by Hough Transform

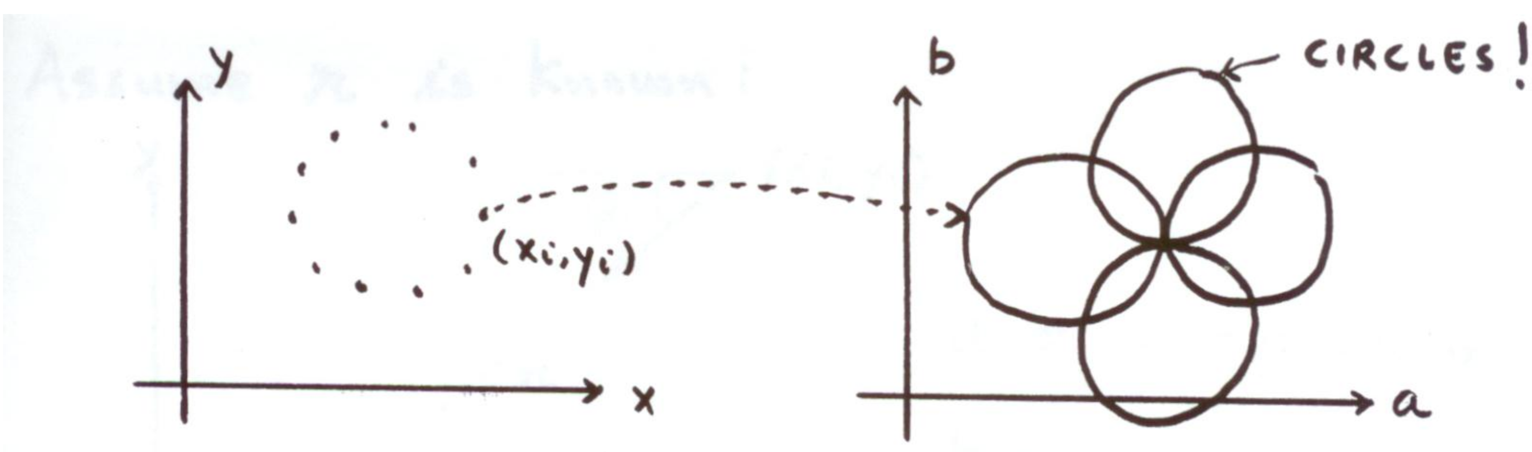
Equation of Circle:



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

If radius is known: (2D Hough Space)

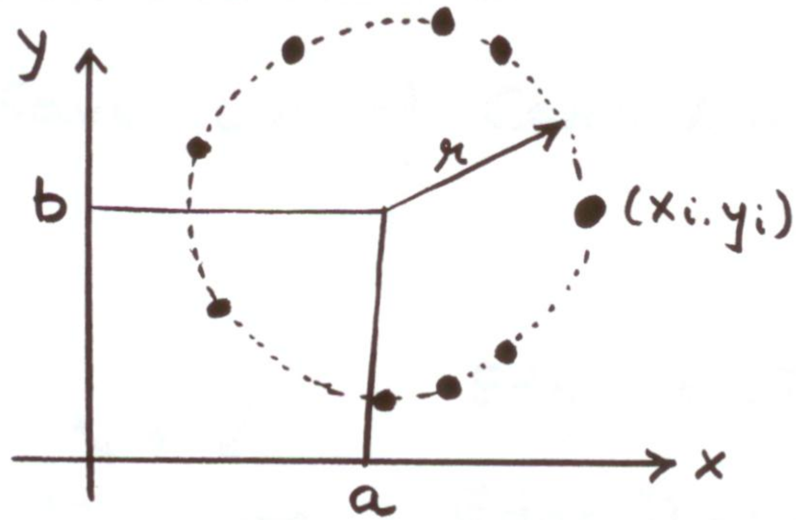
Accumulator Array $A(a, b)$



Finding Circles by Hough Transform

Equation of Circle:

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

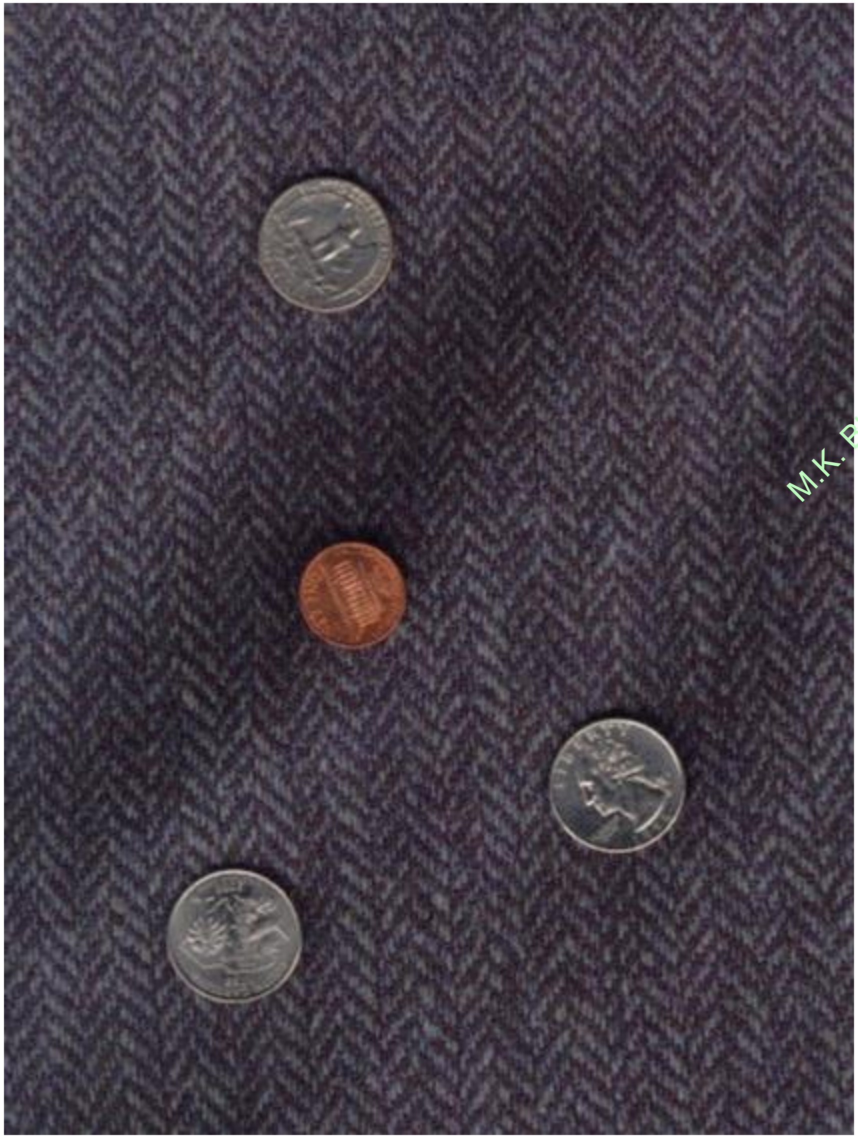


If radius is not known: 3D Hough Space!

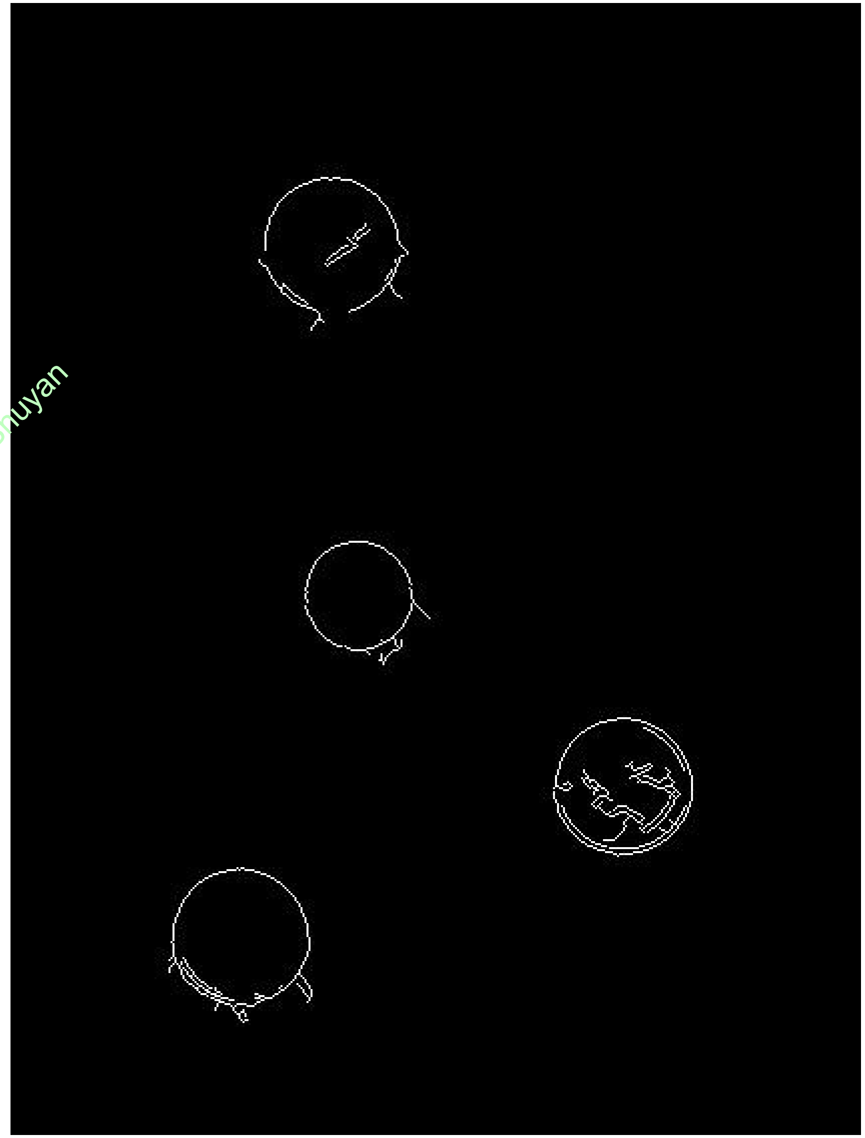
Use Accumulator array $A(a, b, r)$

Finding Coins

Original

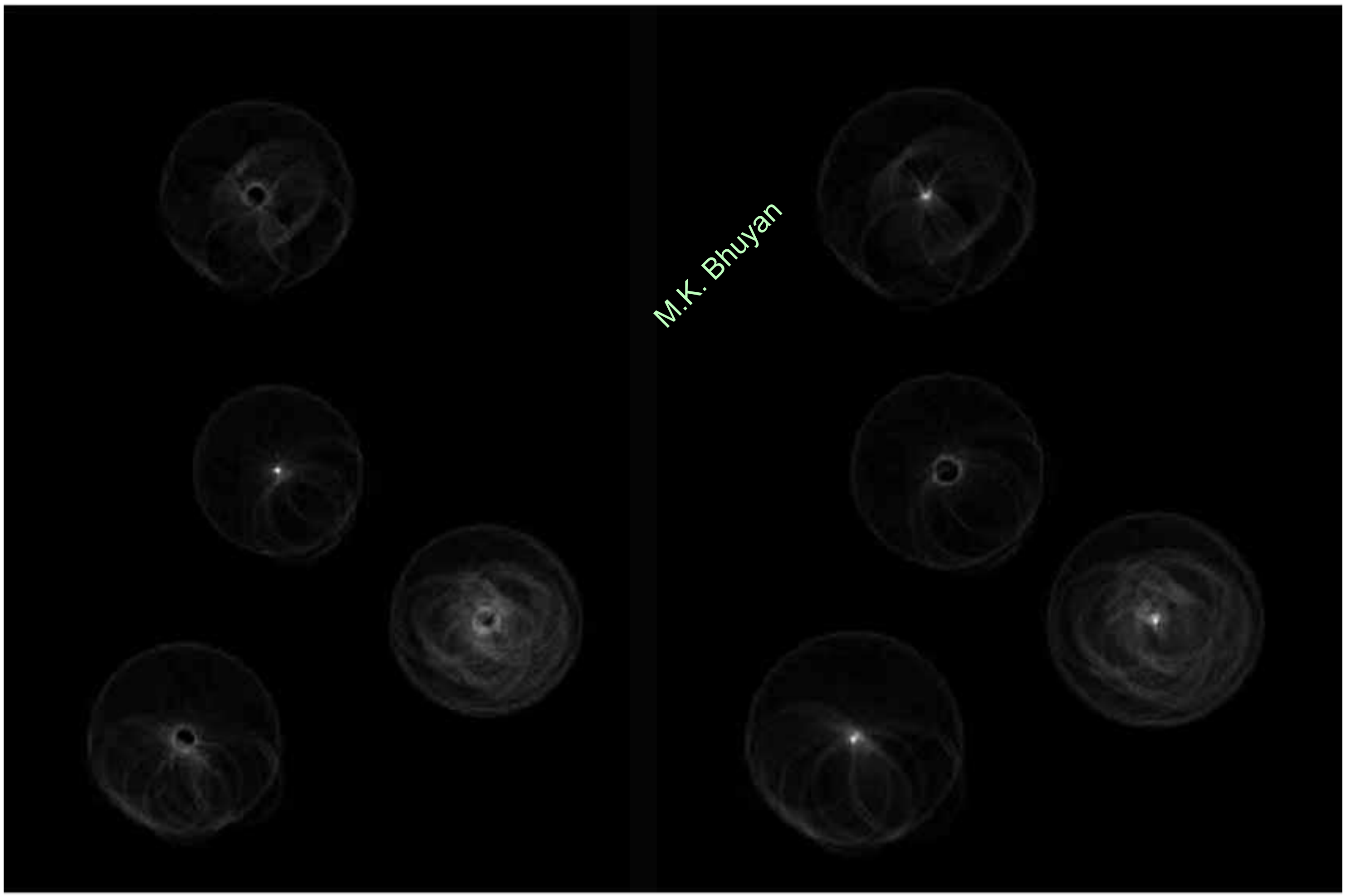


Edges (note noise)



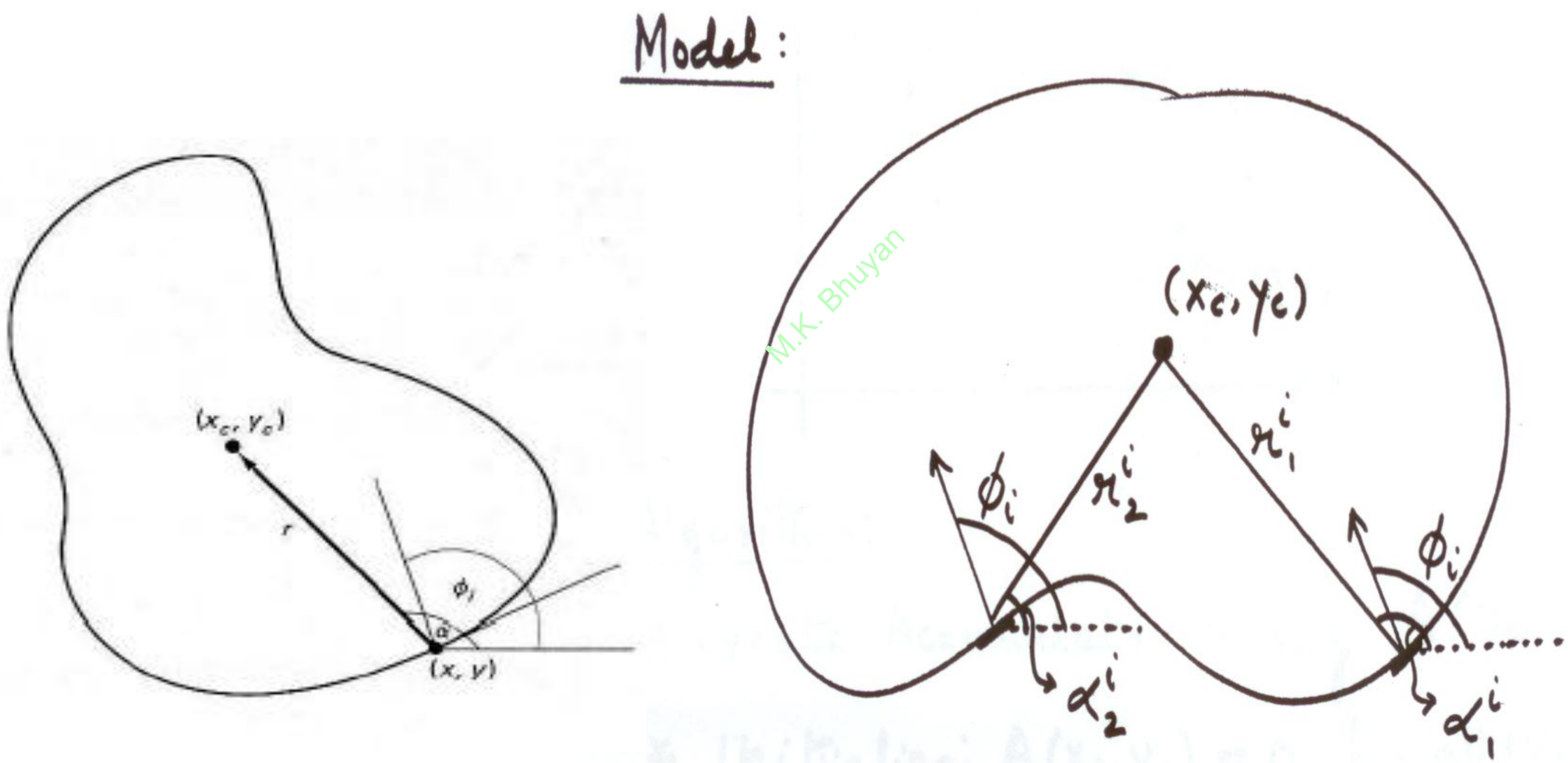
Finding Coins (Continued)

Penn Quarters



Generalized Hough Transform

- Model Shape NOT described by equation



Generalized Hough Transform

- Model Shape NOT described by equation

ϕ - Table

Edge Direction	$\bar{r} = (r, \alpha)$
ϕ_1	$\bar{r}^1_1, \bar{r}^1_2, \bar{r}^1_3$
ϕ_2	$\bar{r}^2_1, \bar{r}^2_2$
$\vdots$	$\vdots$
ϕ_i	$\bar{r}^i_1 \; \vdots \; \bar{r}^i_2$
$\vdots$	$\vdots$
ϕ_n	$\bar{r}^n_1, \bar{r}^n_2$

Generalized Hough Transform

Find Object Center (x_c, y_c) given edges (x_i, y_i, ϕ_i)

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each entry $\overline{r}_k^i$ in table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

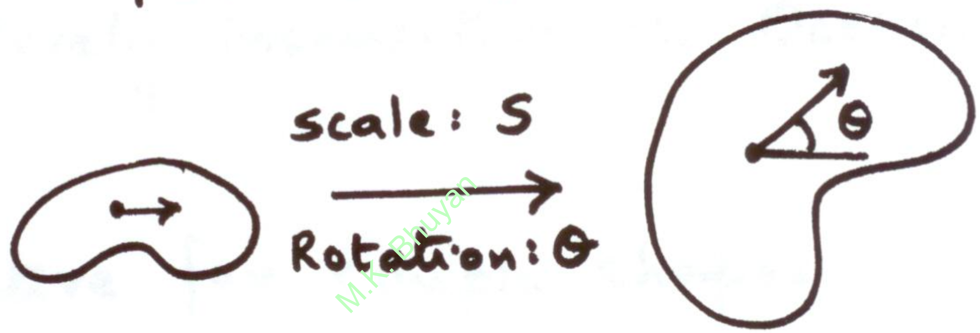
Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$

Scale & Rotation :

Use Accumulator Array :

$$A[x_c, y_c, S, \theta]$$



Use :

$$x_c = x_i + r^i_k S \cos(\alpha^i_k + \theta)$$

$$y_c = y_i + r^i_k S \sin(\alpha^i_k + \theta)$$

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

Hough Transform: Comments

- Works on Disconnected Edges
- Relatively insensitive to occlusion
- Effective for simple shapes (lines, circles, etc)
- Trade-off between work in Image Space and Parameter Space
- Handling inaccurate edge locations:
    - Increment Patch in Accumulator rather than a single point