Machine Vision
Part 2

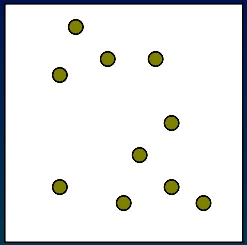
Professor Kok-Meng Lee
Georgia Institute of Technology
George W. Woodruff School of Mechanical Engineering
Atlanta, GA 30332-0405

Webpage: www.me.gatech.edu/me6406

Course Outline

- Introduction and low-level processing
  - Physics of digital images, histogram equalization, segmentation, edge detection, linear filters.
- **Model-based Vision**
  - **Hough transform, pattern representation, matching**
- Geometric methods
  - Camera model, calibration, pose estimation
- Neural network for machine vision
  - Basics, training algorithms, and applications
- Color images and selected topics
  - Physics, perception, processing and applications

Hough Transform



- A technique used to detect specific structural relationships among pixels in a binary digital image.
  - Mathematical equations; lines, circles and ellipses.
- Introduction: Given n points in an image, find a subsets of these points that lie on straight lines.

One possible solution (trivial) ,

- Step 1: find all lines determined by every pairs of points,

$$\frac{n(n-1)}{2} \sim O(n^2) \text{ lines}$$

- Step 2: find all subsets of points that are closed to particular lines.

$$(n-2)\left[\frac{n(n-1)}{2}\right] \sim O(n^3) \text{ operations}$$

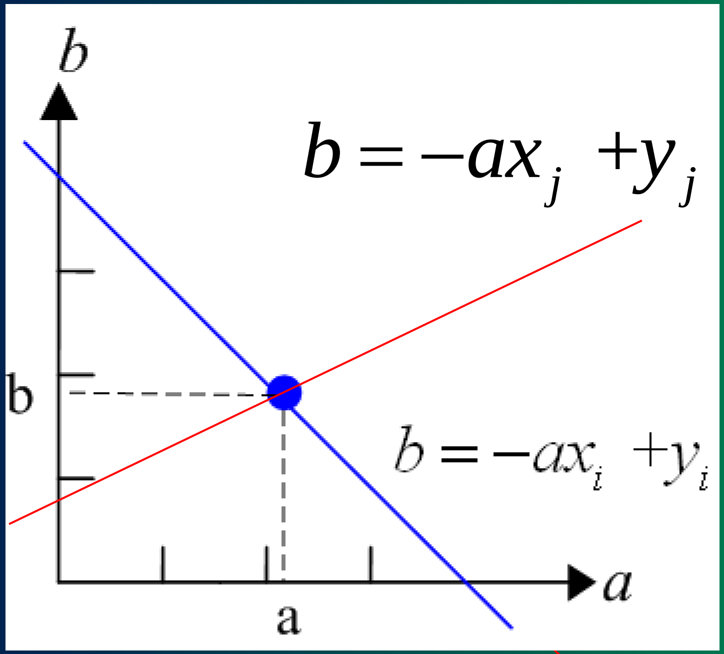
Problems: Computationally prohibitive!

Hough's basic argument

- Alternatively, consider a point (x_i, y_i) and the general equation of a straight line:

$$y_i = ax_i + b \quad (1)$$

  - On the xy plane: Infinite number of lines pass through (x_i, y_i) but all satisfy Eq. (1) for every pair of (a,b).



  - If we rewrite the equation

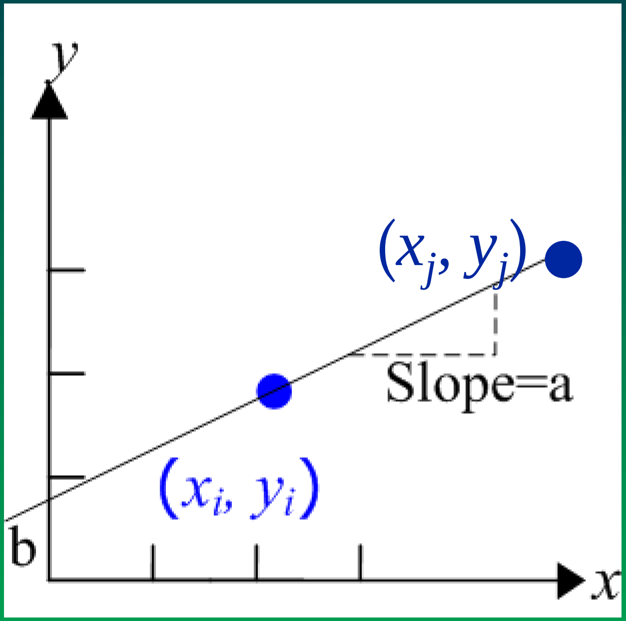
$$b = -ax_i + y_i \quad (2)$$

  then we have a **single line** for a fixed pair of (x_i, y_i).

  - We call the *ab* plane as the parametric plane.
  - Every point on the xy corresponds to a line in ab plane.

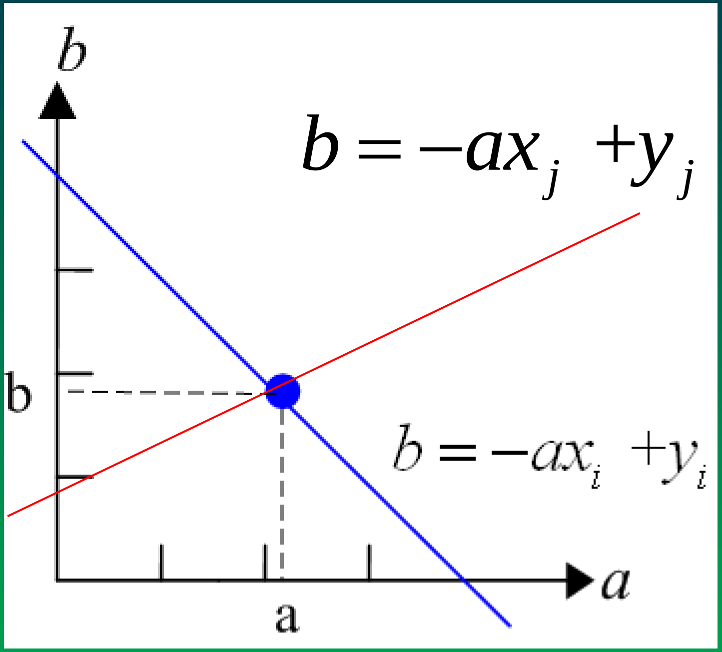
Comparison

Image (xy) plane



$$y_i = ax_i + b \quad (1)$$

Parametric (ab) plane



$$b = -ax_i + y_i \quad (2)$$

Example:

x, y	$y = ax + b$	$b = -ax + y$
1, 3	$3 = a1 + b$	$b = -1a + 3$
2, 3	$2 = a2 + b$	$b = -2a + 2$
4, 3	$3 = a4 + b$	$b = -4a + 3$
4, 0	$0 = a4 + b$	$b = -4a + 0$

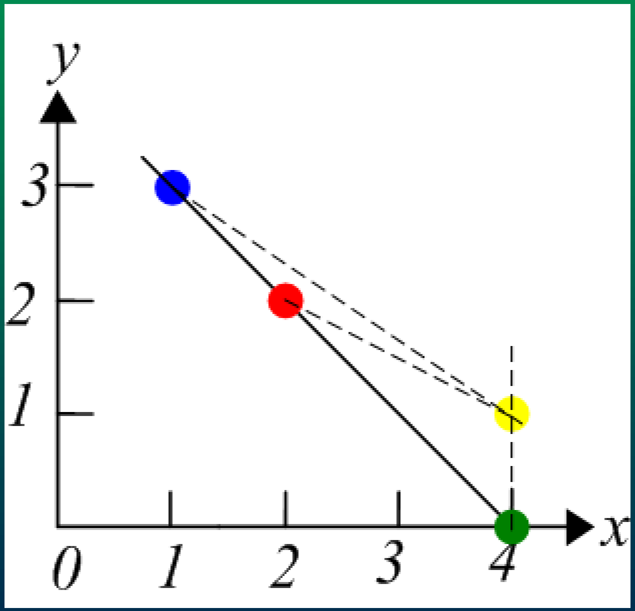
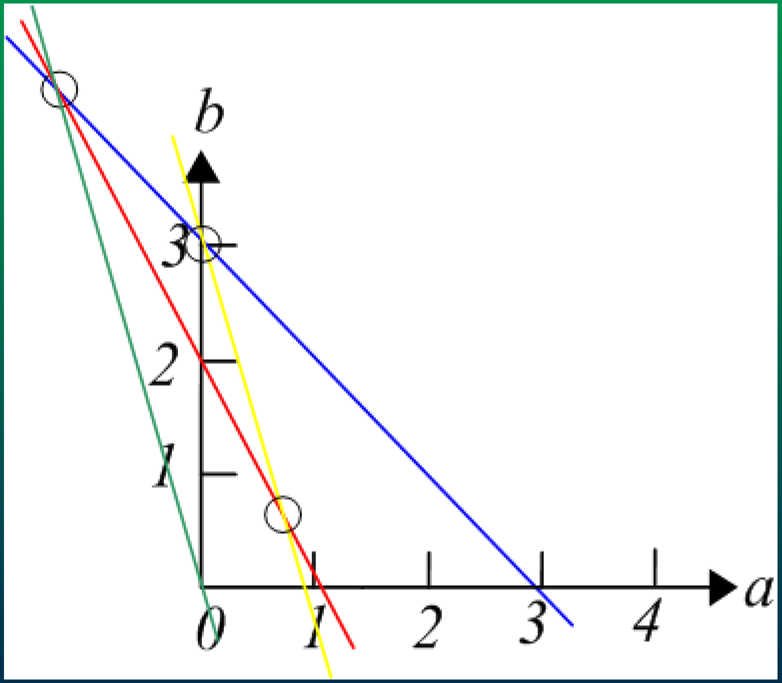


Image (xy) plane



Parametric (ab) plane

Hough's basic argument

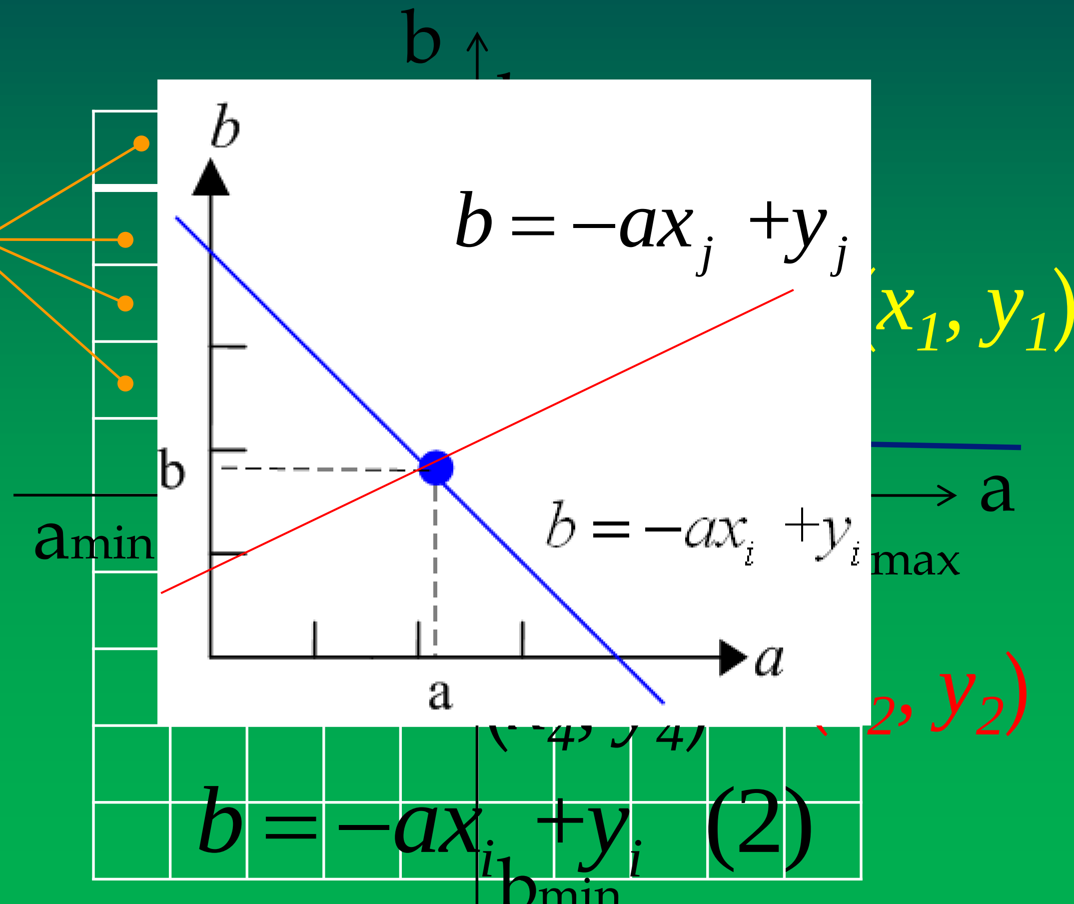
- If n points, (x_i, y_i) … (x_j, y_j)… (x_n, y_n), are on the same line (with slope a and intecept b), then there will be n lines intercepting on the parametric space at (a,b).

$$b = -ax_i + y_i$$

Accumulator cell A(i,j)

$$a_{\min} \le a_k \le a_{\max}$$

$$b_{\min} \le b_k \le b_{\max}$$



Example:
A(2,1) = 4 implying that (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) are on the same line in the image plane.

Basic concept of Hough's algorithm

- Initialize A(I,j)
- For every point (x_k, y_k) in the image plane,
  - Let $a = a_i$ where $a_{\min} \le a \le a_{\max}$

    compute $b = -ax_k + y_k$

    round $b = b_j$

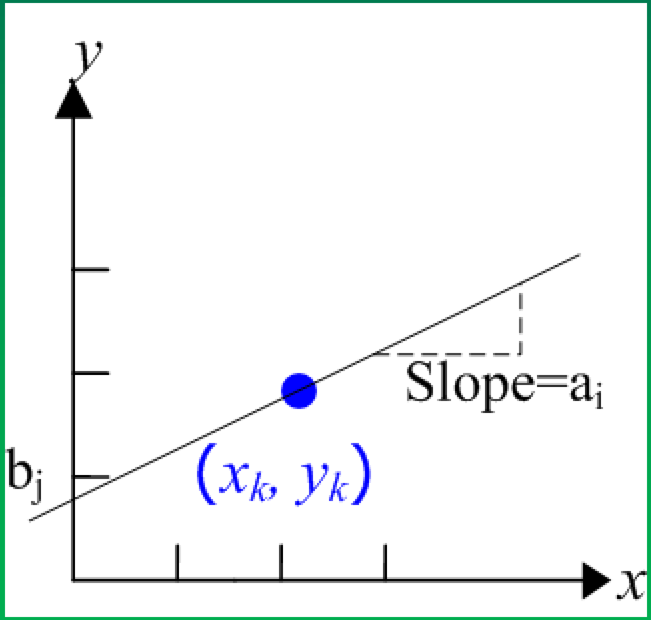
  - If a choice of a_p results in solution b_q

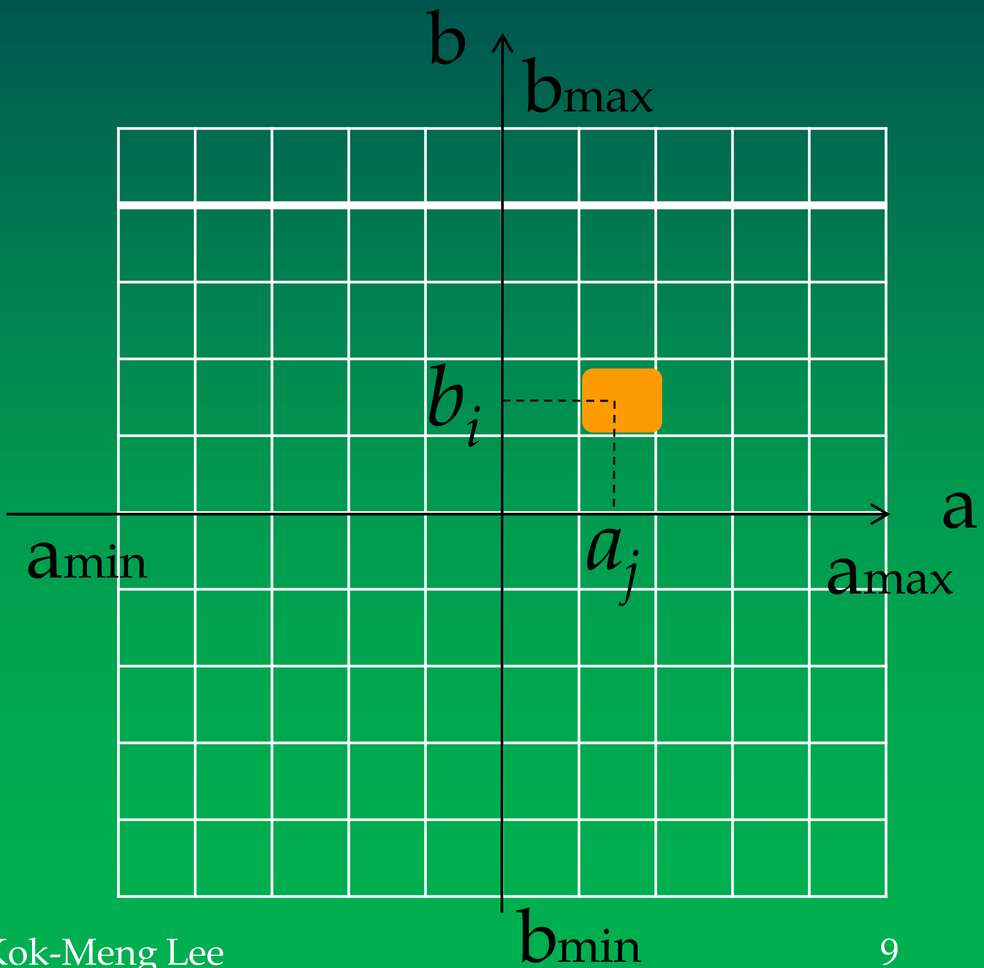
    Increment $A(p,q) = A(p,q) + 1$

Basic concept of Hough's algorithm

- At the end of this procedure, a value of M in $A(i, j)$ corresponds to M points in the image (xy) plane lying on the line:

$$b_j = -a_i x + y$$



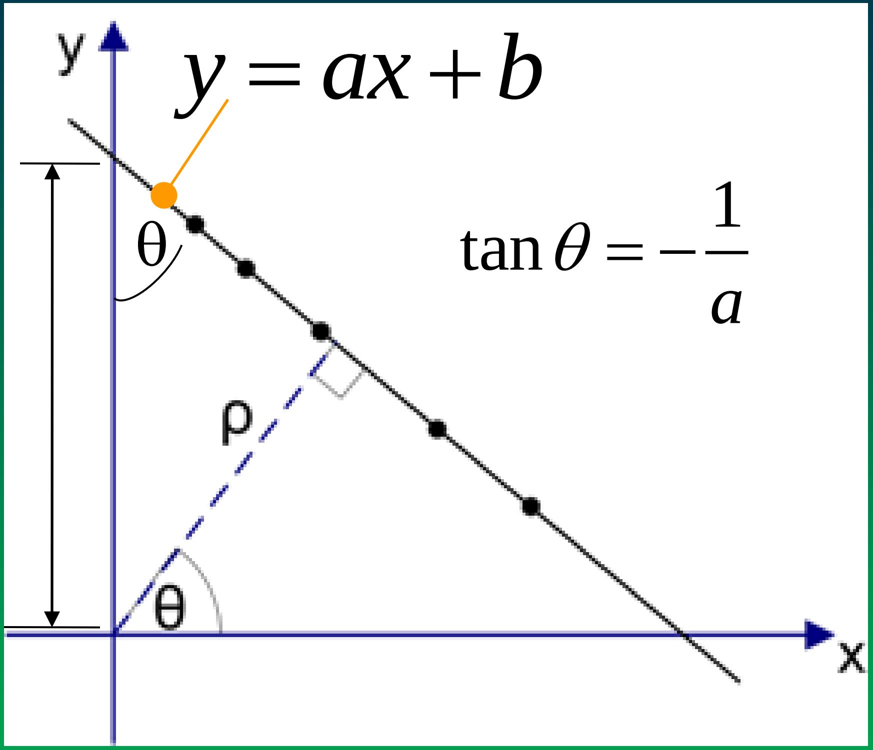


Hough Transform

- Computationally attractive.
  - If a is subdivided to K increment for every point (x_k, y_k), there are K possible values of b corresponding to K possible values of a.
  - For *n* image points, involving *nK* computations; linear in *n.*
- Accuracy of the co-linearity depends on the number of subdivisions in the ab plane.
- The range of the parameters (*a*, *b*) for the equation presents a problem:

$$y = ax + b \qquad -\infty \le a \le +\infty$$ **(Vertical straight line)**

Alternative parameters (ρ, θ)



$$b = \frac{\rho}{\sin\theta}$$

$$y = \underbrace{\left(-\frac{\cos\theta}{\sin\theta}\right)}_{a} x + \underbrace{\left(\frac{\rho}{\sin\theta}\right)}_{b} \quad \Rightarrow \quad \rho = x\cos\theta + y\sin\theta$$

$$0^\circ \le \theta \le 180^\circ$$

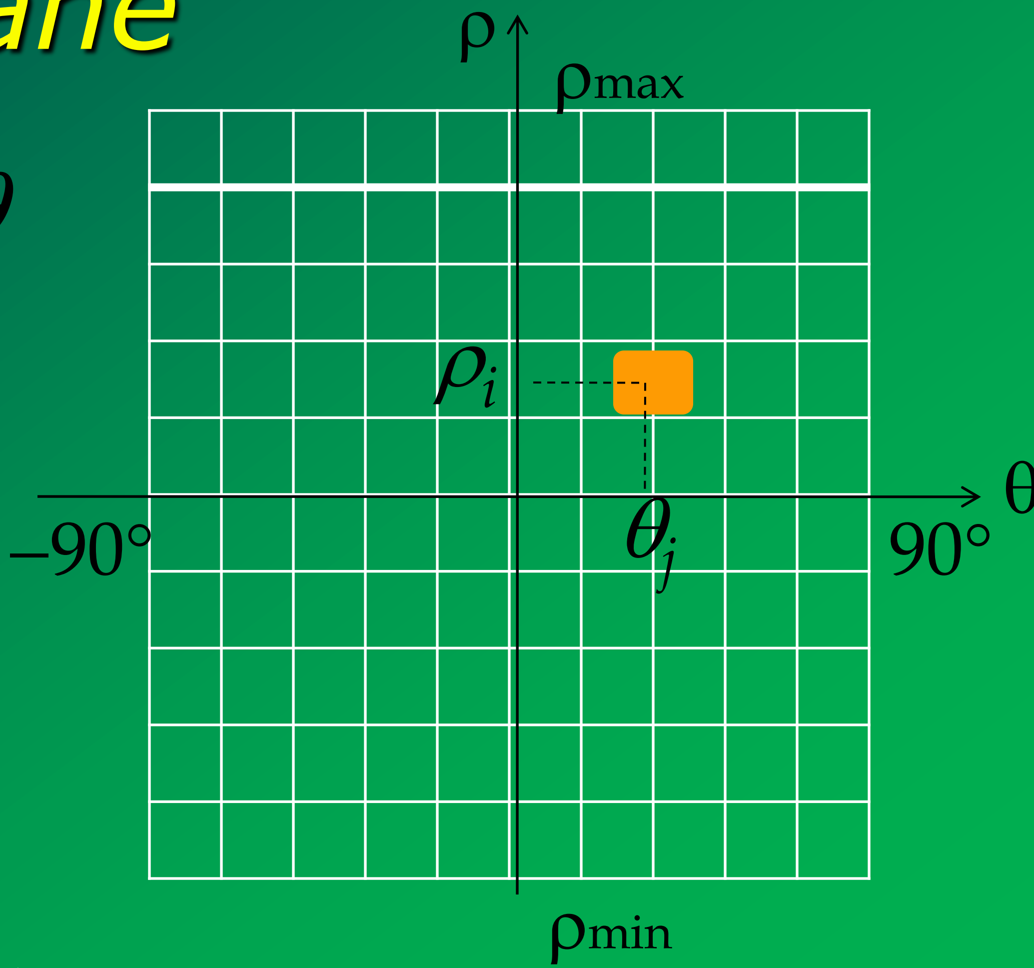
Hough Transform of straight lines on the $\rho\theta$ plane

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

$$0° \le \theta \le 180°$$

$$\rho_{\min} \le \rho \le \rho_{\max}$$

where ρ<D/2 ; and D is the diagonal dimension of the image.



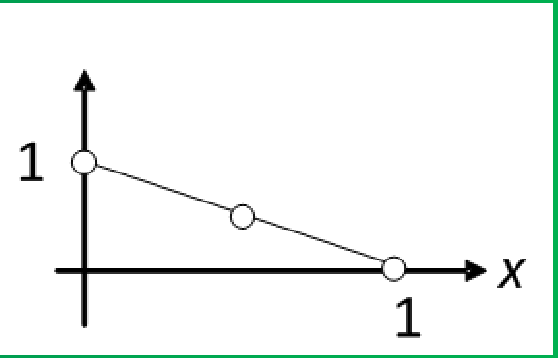
Example

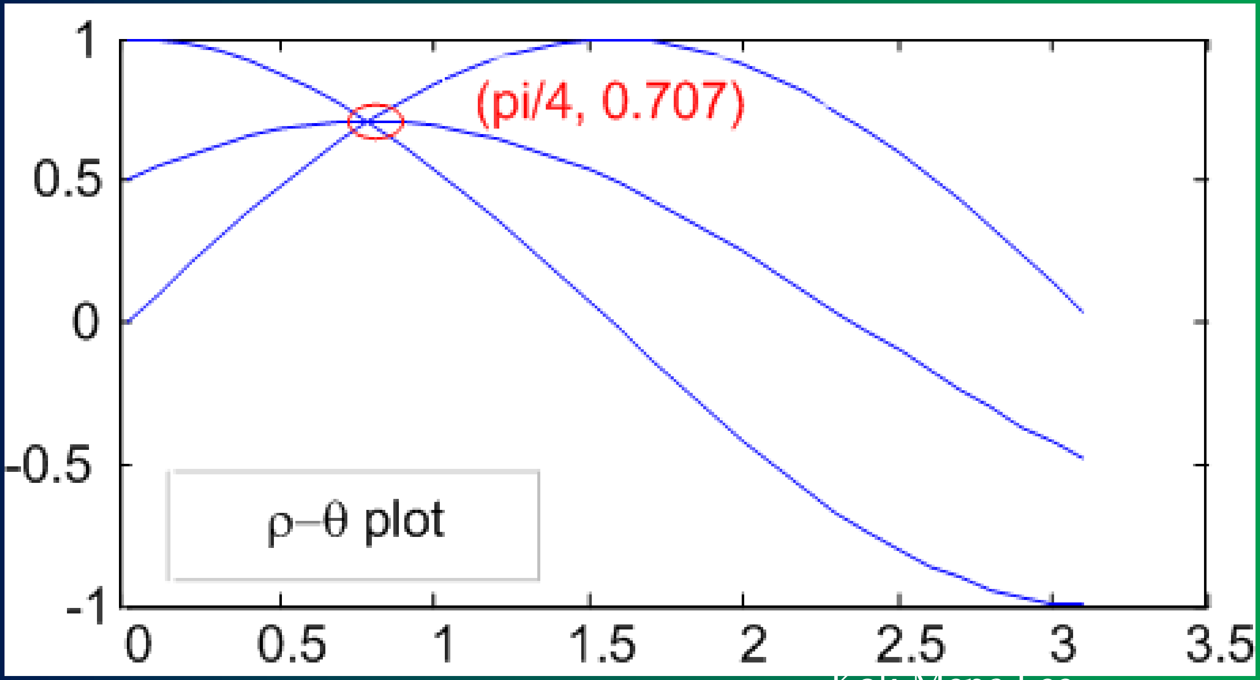
$$\rho = x\cos\theta + y\sin\theta$$

Point (0,1): $\rho = \sin\theta$

Point (0.5, 0.5): $\rho = 0.5(\cos\theta + \sin\theta)$

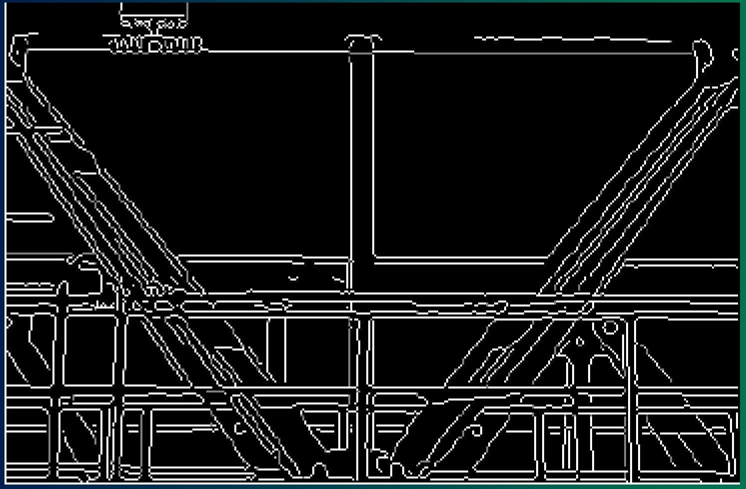
Point (1,0): $\rho = x\cos\theta$





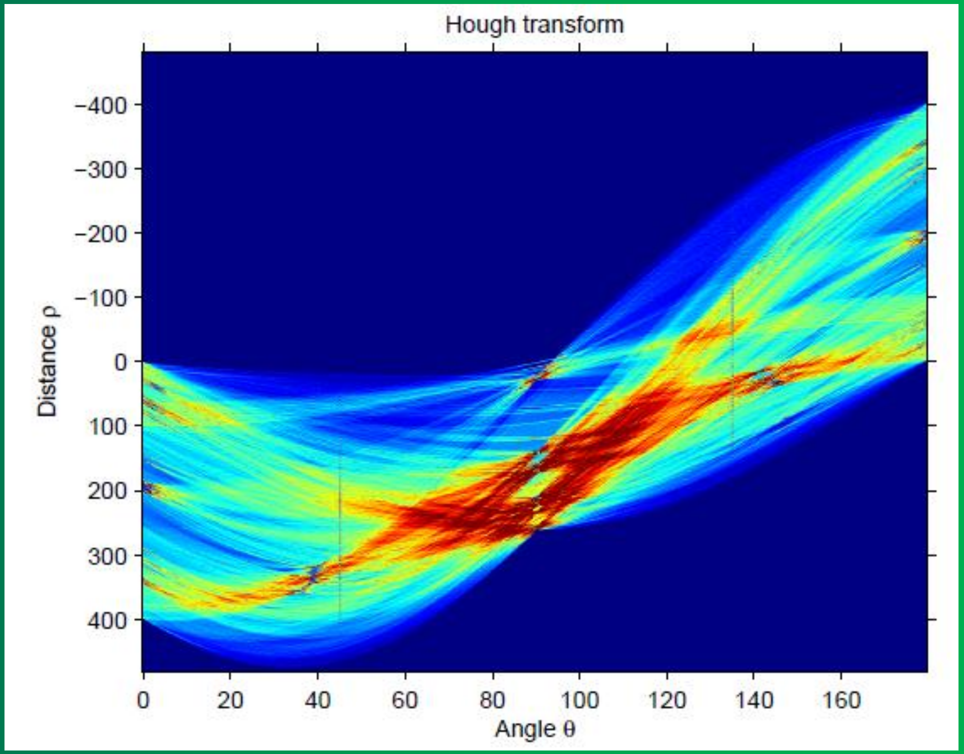
Example

Original Image



Edge Image

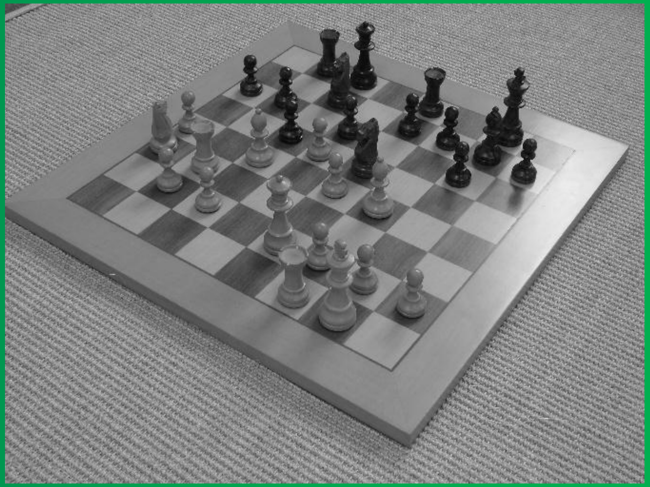
$$\rho = x\cos\theta + y\sin\theta$$



Jeppe Jensen, 2007.

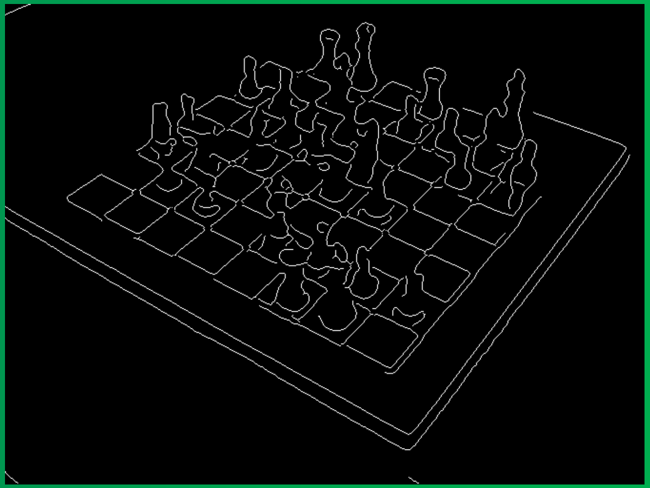
Example (Matlab)

David Young, University of Sussex, 2006



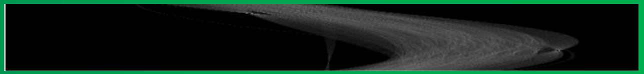
The image is read, converted to gray-level and then displayed in a new figure.

im = teachimage('chess1.bmp'); f1 = figure;
imshow(im);



Find edges (Canny edge detector; thresholds 0.1 and 0.2, and the smoothing constant 2 to remove the high spatial frequency texture in the background).

e = edge(im, 'canny', [0.1 0.2], 2); figure;
imshow(e);



Perform the (ρ,θ) Hough transform

[h, theta, rho] = hough(e); figure;
imshow(sqrt(h'), [ ]);

Find the peaks in the transform

The maximum number of peaks to find is 21, and peaks must not be closer together than 27 bins in rho and 11 bins in theta.

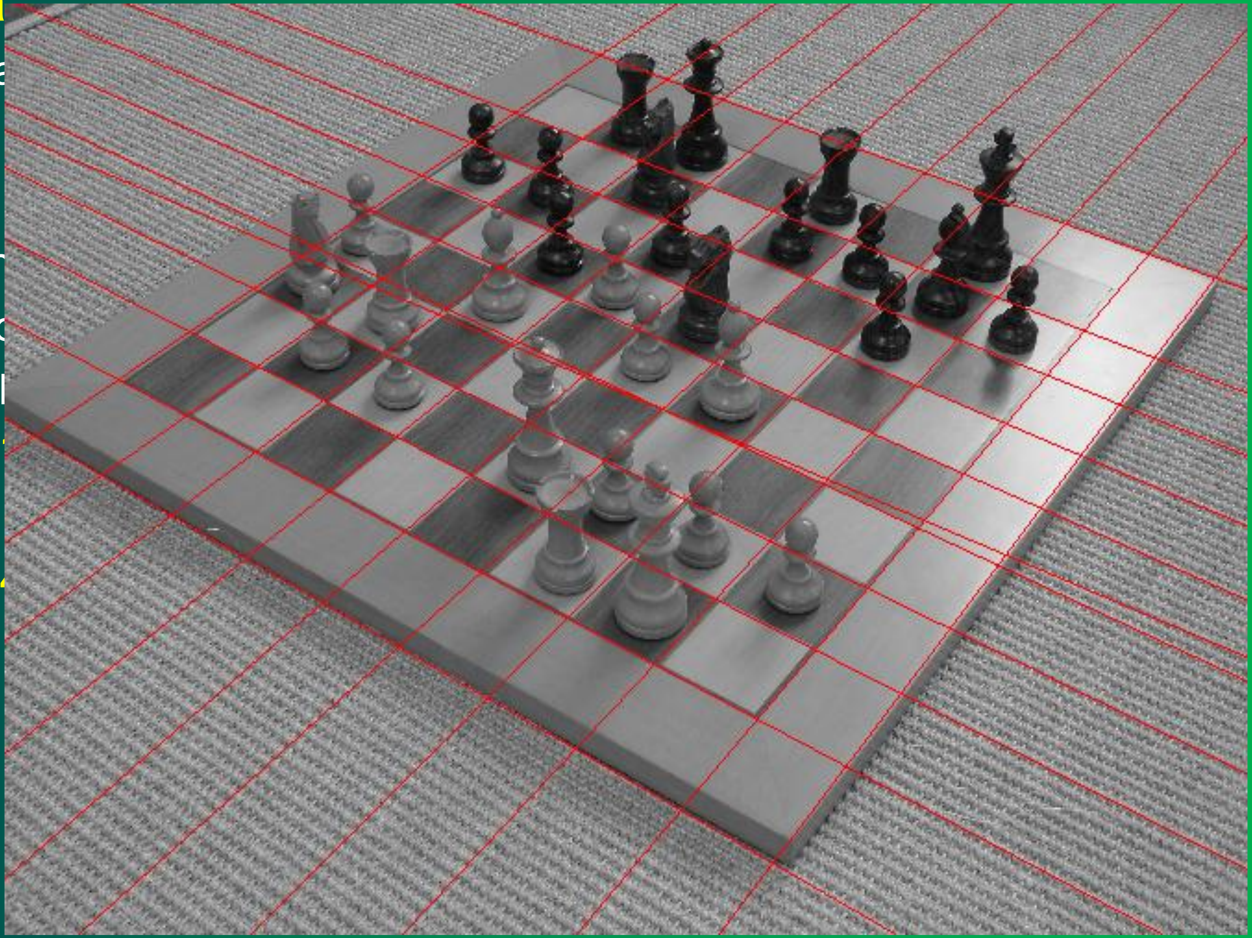
p = houghpeaks(h, 21, 'Threshold', 0, 'NhoodSize', [27 11]);

Example (Matlab) cont.

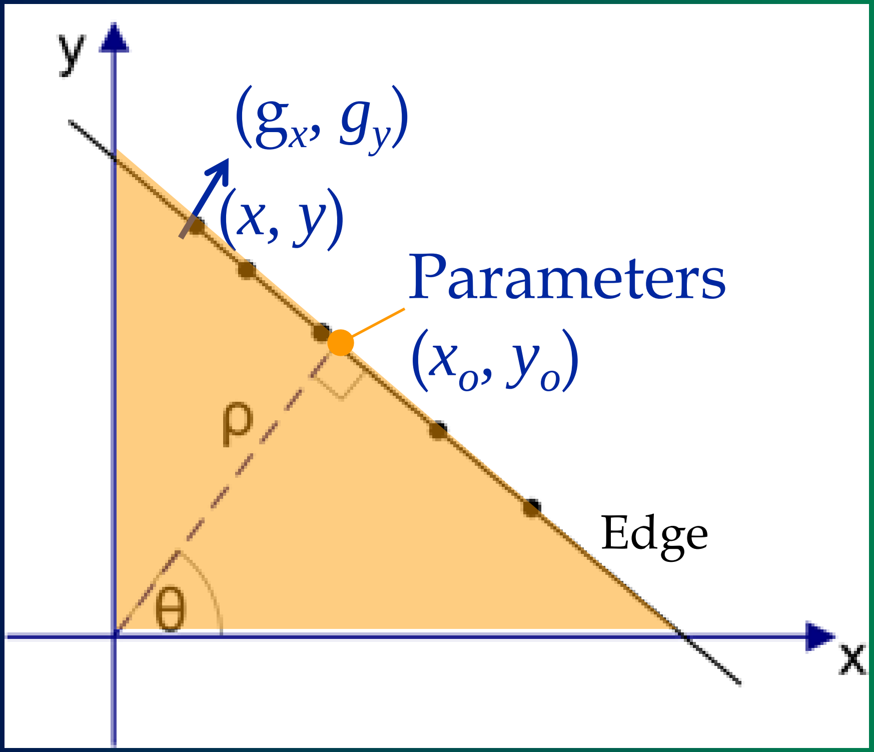
Plot the lines on the image

%convert the angles used from degrees to radians (because normal Matlab functions use radians – hough returns in degrees).

```
theta = (pi/180)*theta;
figure(f1); % original ima
[nr, nc] = size(im);
hold on % plot on top of
for pr = p' % pr is assign
    r = rho(pr(1)); % loc
    t = theta(pr(2)); % 
    l = line box(1, nc, 
    if ~isempty(l)
      plot(l([1 3]), l([2,
    end
end
hold off
```



Another alternative (foot of normal)



$$y = ax + b$$

$$a = \frac{y - y_o}{x - x_o} = -\frac{g_x}{g_y} = -\frac{x_o}{y_o}$$

$$\Rightarrow x_o\left(x - x_o\right) + y_o\left(y - y_o\right) = 0$$

$$\Rightarrow x_o\left(x - x_o\right) + \frac{g_y x_o}{g_x}\left(y - \frac{g_y x_o}{g_x}\right) = 0$$

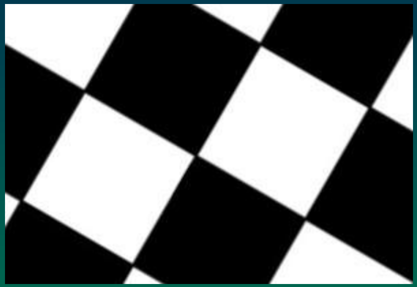
$$\boxed{\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \nu \begin{bmatrix} g_x \\ g_y \end{bmatrix} \quad \text{where } \nu = \frac{x g_x + y g_y}{g_x^2 + g_y^2}}$$

Note: $\rho = \nu\sqrt{x_o^2 + y_o^2}$

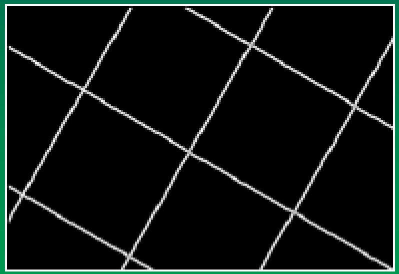
Advantages:

- Eliminate trigonometry functions
- Use of gradient information
- Compute both parameters simultaneously

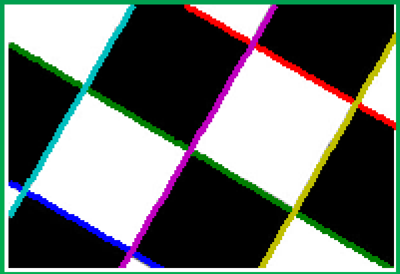
Example
(foot of normal)



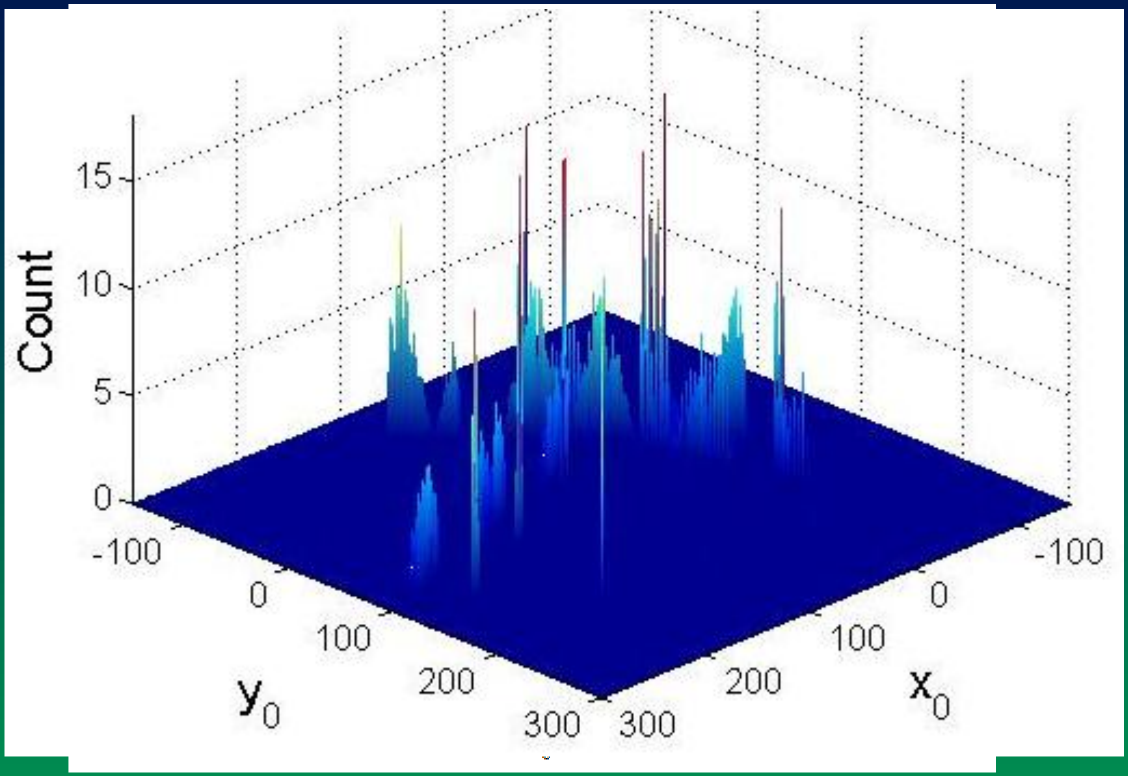
Rotated checker



Gradient



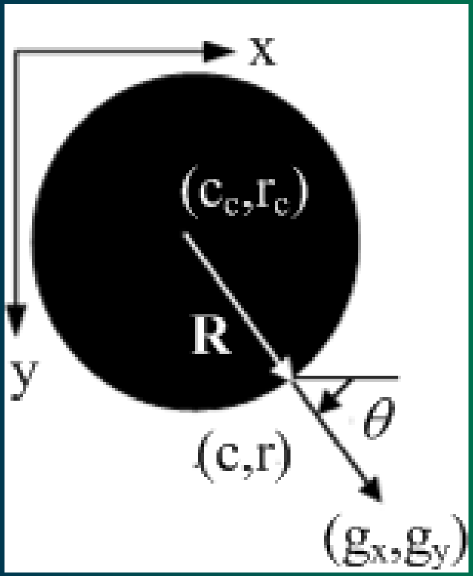
Plot limes on image



	x_0	y_0	a	b	ρ	$\theta°$
Line 1	-58	116	0.500	145	130	26.6
Line 2	-19	40	0.48	49.0	44	25.4
Line 3	25	-50	0.50	-62.5	56	25.6
Line 4	71	32	-2.22	189.5	78	-65.7
Line 5	159	80	-1.99	396.0	178	-63.3
Line 6	246	120	-2.05	624.3	274	-64.0

Hough Transform for finding circles

Three parameters: c_c, r_c; R



$$\boxed{\begin{aligned}&(c-c_c)^2+(r-r_c)^2=R^2\\&c_c=c-R\cos\theta\\&r_c=r-R\sin\theta\end{aligned}}$$

$$\cos\theta=\frac{g_x}{\sqrt{g_x^{\ 2}+g_y^{\ 2}}}$$

From gradient: $\tan\theta=\dfrac{g_y}{g_x}$

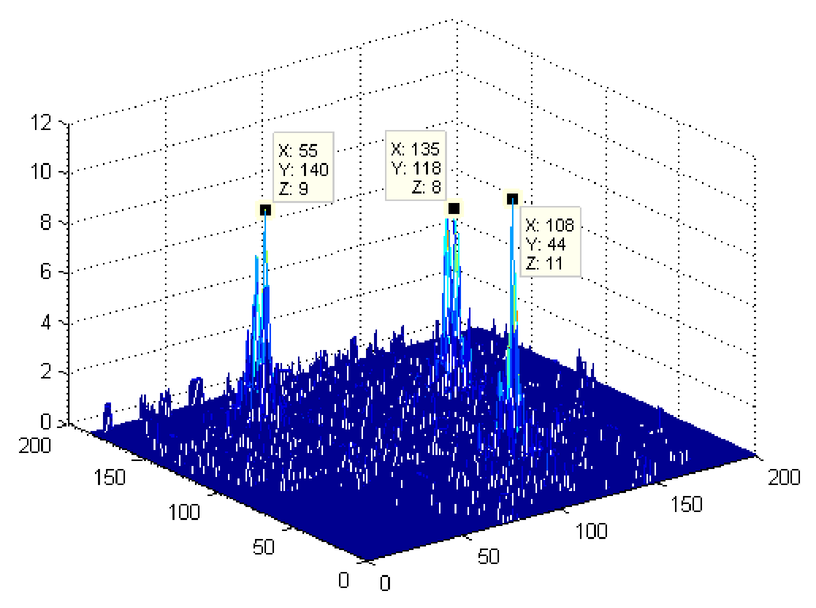
$$\sin\theta=\frac{g_y}{\sqrt{g_x^{\ 2}+g_y^{\ 2}}}$$

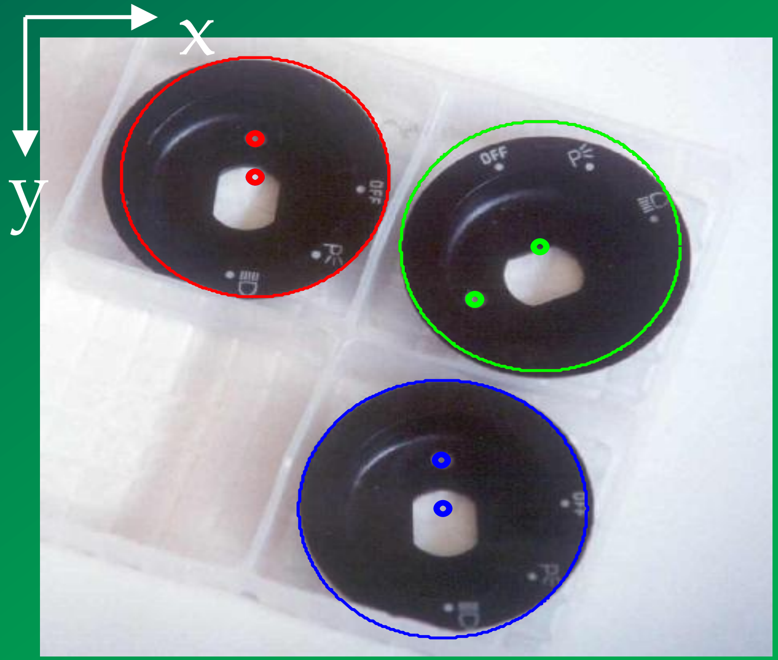
Example on HT for circles

Find the radius location of the circles using Hough transform.

Matlab Implementation

- Import grayscale image
- Find gradient (Sobel)
- Initialize Accumulator (x_c, y_c, r)
- Inspect each pixel,
    1) If gradient exist, calculate G_x and G_y
    2) For each r, calculate x_c and y_c.
    3) Update accumulator cell

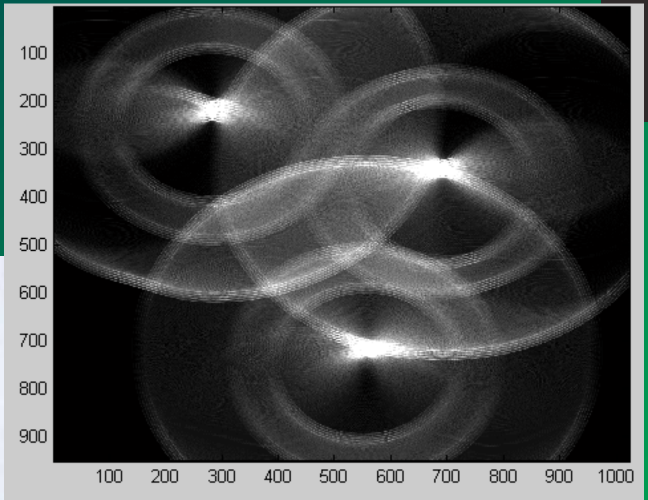


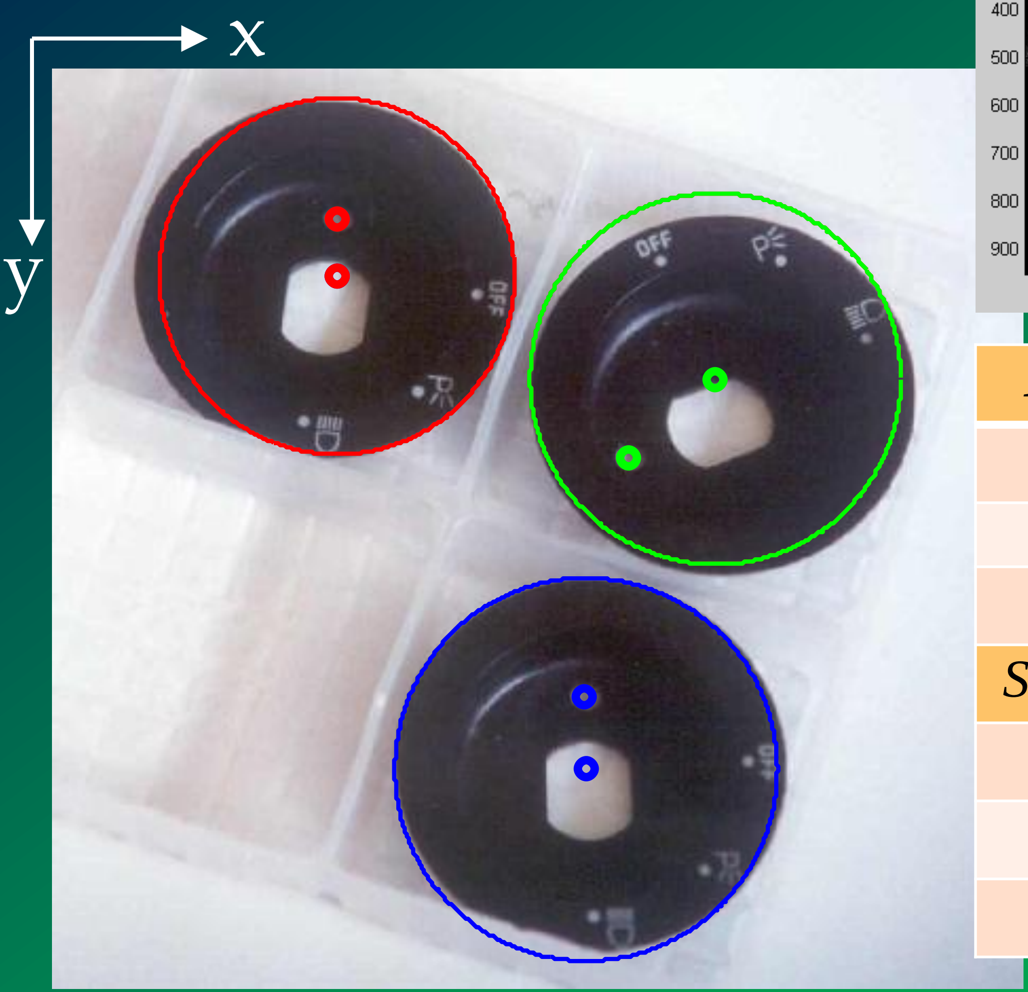


Kok-Meng Lee

Example on HT for circles

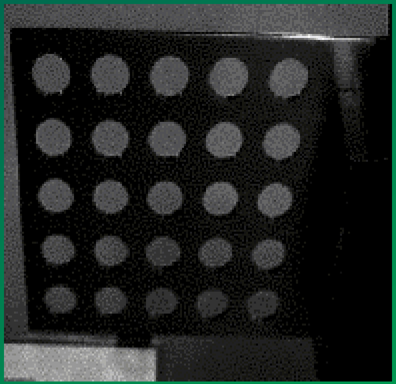
Find the radius location of the circles using Hough transform.

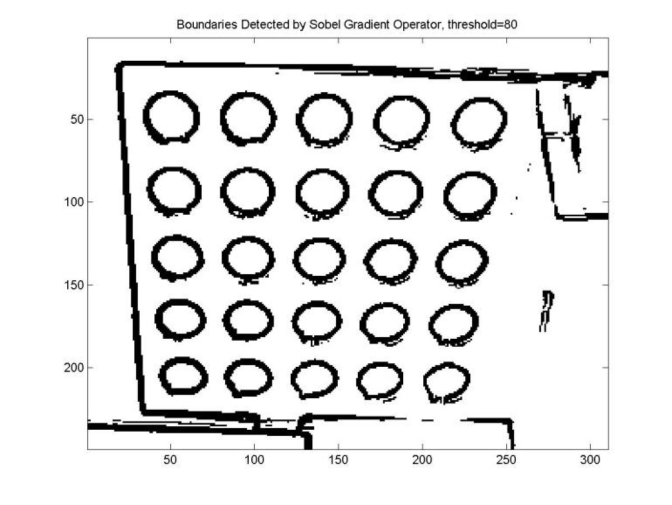


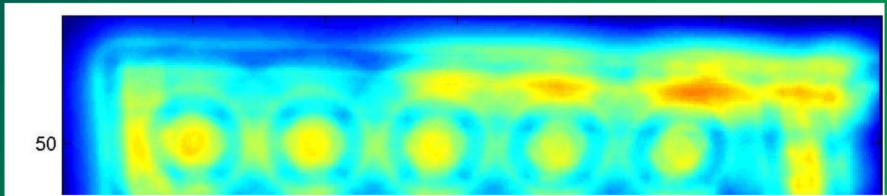


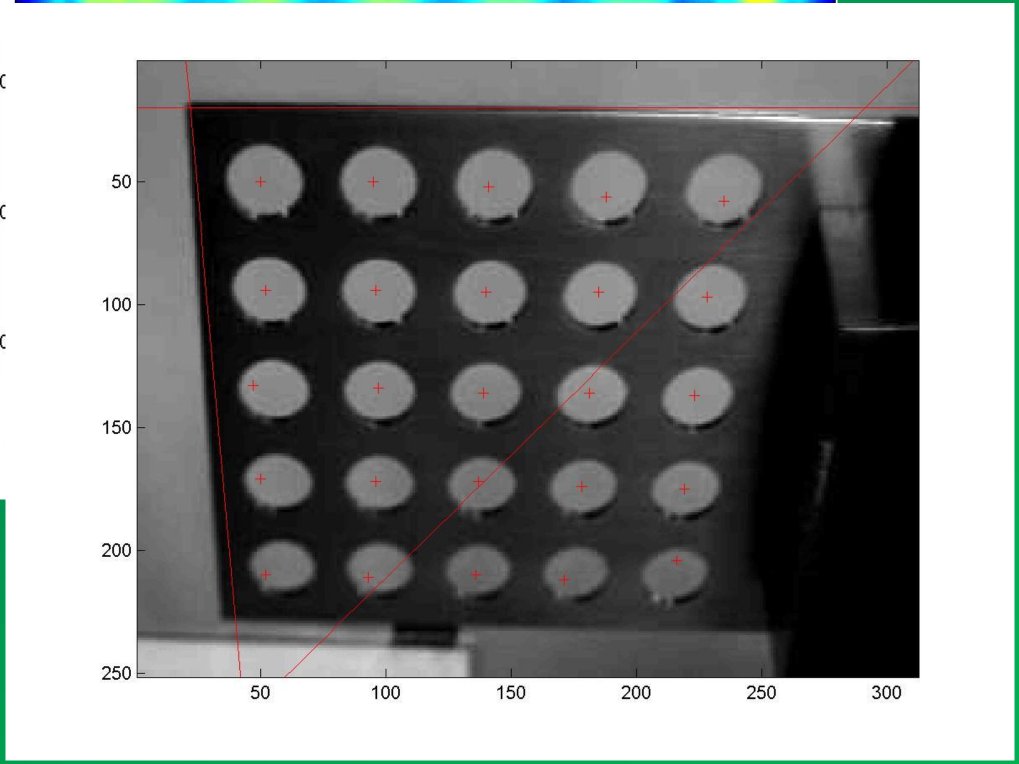
Big Circle	x_c	y_c	r
1	296	216	184
2	688	322	192
3	554	726	198
Small Circle	x_c	y_c	
1	296	156	
2	597	403	
3	551	651	

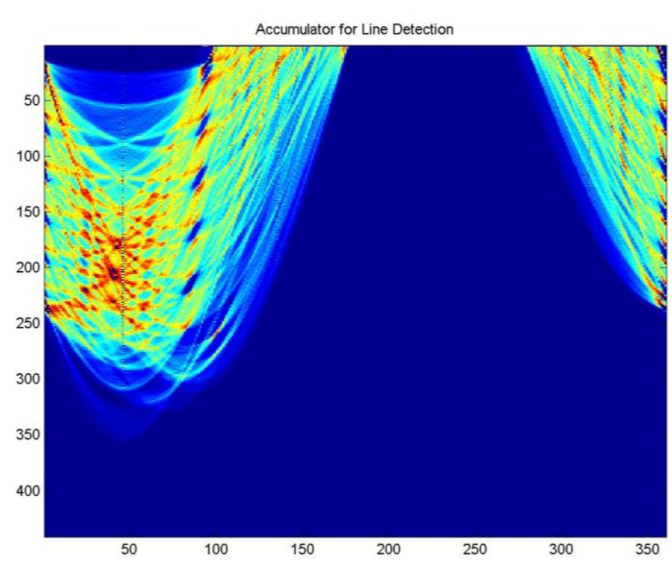
Example on HT for circles











Kok-Meng Lee

Hough Transform for finding ellipses

For illustration, assume that the principal axis of the ellipse is parallel to the x axis

Four parameters (a, b, x_c, y_c):

$$\frac{(x-x_c)^2}{a^2}+\frac{(y-y_c)^2}{b^2}=1$$

Take total derivative w. r. t. x:

$$\frac{x-x_c}{a^2}+\frac{y-y_c}{b^2}\frac{dy}{dx}=0$$

Using edge operator (g_x/g_y).

$$x-x_c=-\left(a/b\right)^2\left(y-y_c\right)\tan\phi$$

$$\boxed{x_c=x\pm\frac{a}{\sqrt{1+\left(b/a\right)^2/\tan^2\phi}} \qquad y_c=y\pm\frac{b}{\sqrt{1+\left(a/b\right)^2\tan^2\phi^2}}}$$

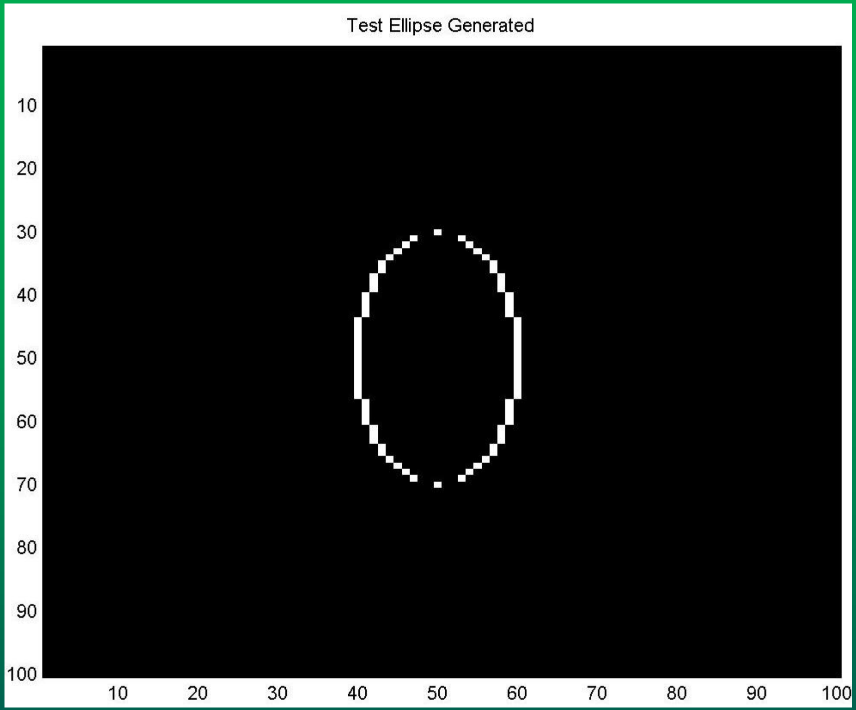
Hough Transform for finding ellipses

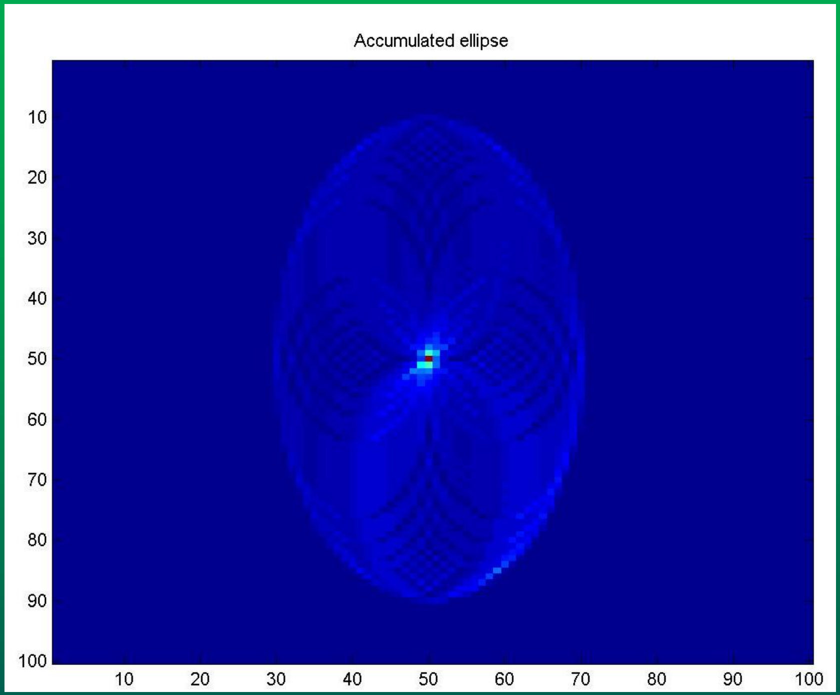
$$x_c = x \pm \frac{a}{\sqrt{1+(b/a)^2/\tan^2\phi}} \qquad y_c = y \pm \frac{b}{\sqrt{1+(a/b)^2\tan^2\phi^2}}$$

Algorithm for applying Hough technique to detect an ellipse from an image:

1. Quantize parameter space between appropriate maximum and minimum values for parameters (a, b, x_c, y_c).
2. Form an accumulator array whose elements are initially zero.
3. For every two discrete points of x and y, solve for .
4. Increment the point in parameter space .
5. Local maxima in the accumulator array now correspond to collinear points in the image array. The values of the accumulator array provide a measure of points on the ellipse.

Hough Transform for finding ellipses

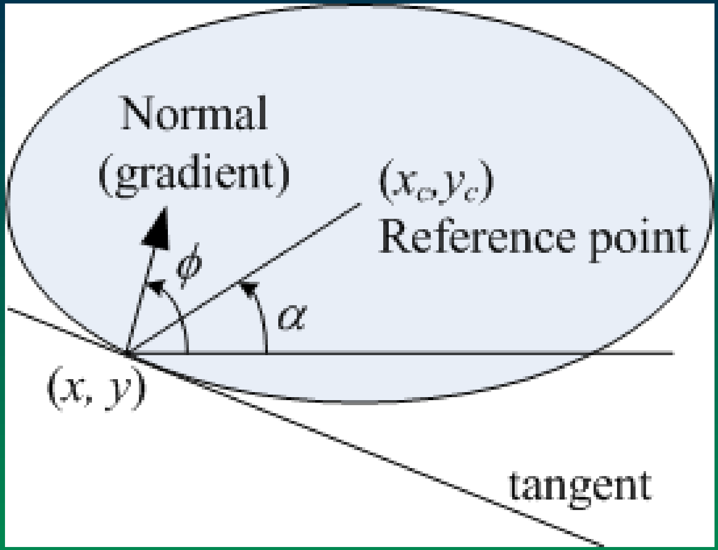




Generalized Hough Transform

(Object has no analytical form)

For illustration, assume that scaling and rotation have been fixed.



Form a R-Table (pre-stored template):

ϕ_i	$\vec{r}_i(R_i, \alpha_i)$ where $R_i(\phi_i) = \lvert \vec{r}_i \rvert$; and $\alpha_i(\phi_i) = \angle \vec{r}_i$
ϕ_1	$\vec{r}_1^{\,1}\vec{r}_2^{\,1}\cdots$
ϕ_2	$\vec{r}_1^{\,2}\vec{r}_2^{\,2}\cdots$

Basic strategy:

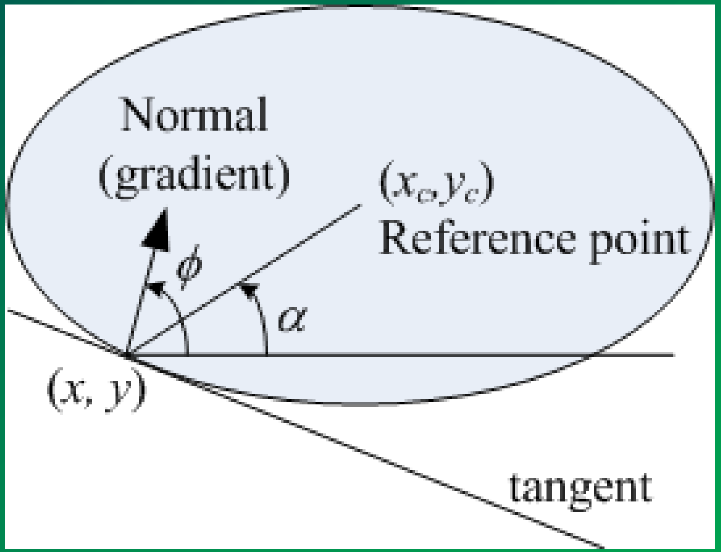
1. To compute the possible loci of reference point (x_c, y_c) in parameter space for edge point data,

$$x_c = x + r(\phi)\cos\left[\alpha(\phi)\right] \qquad y_c = y + r(\phi)\sin\left[\alpha(\phi)\right]$$

2. Then increment the parameter points in an accumulator array.

Generalized Hough Transform

(Object has no analytical form)



Algorithm:

1. Form a R-Table
2. Initialize Accumulator (x, y).
3. From each pixel point, do the following:
   a) Compute ϕ
   b) Calculate possible centers
   c) Increment Accumulator.
4. Increment the point in parameter space .
5. Local maxima in the accumulator array now correspond to collinear points in the object shape.

To include Scaling and rotation:

1. Accumulator (x, y, S, θ)
2. From each pixel point, do the following:

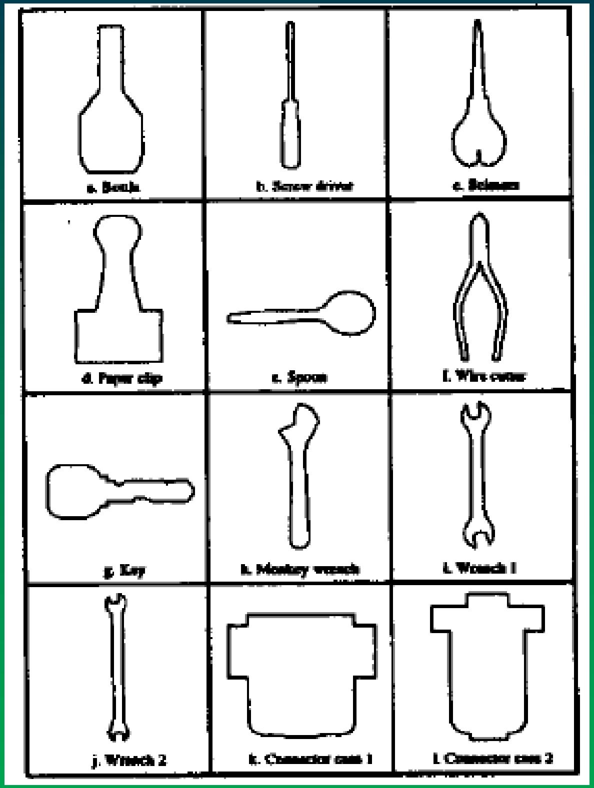
$$x_c = x + r(\phi) S \cos\left[\alpha\left(\phi + \theta\right)\right] \qquad y_c = y + r(\phi) S \sin\left[\alpha\left(\phi + \theta\right)\right]$$

Template matching an introduction

Typical tasks:

- Model objects
- Identify objects
- Locate objects

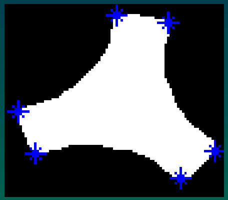
Pattern Recognition

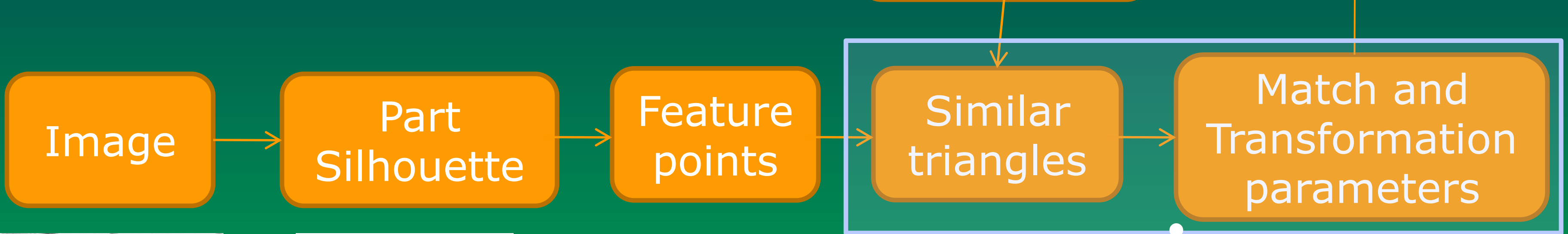


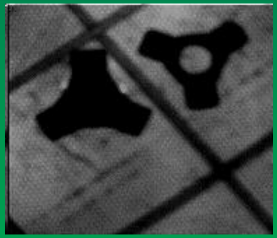
Face Recognition

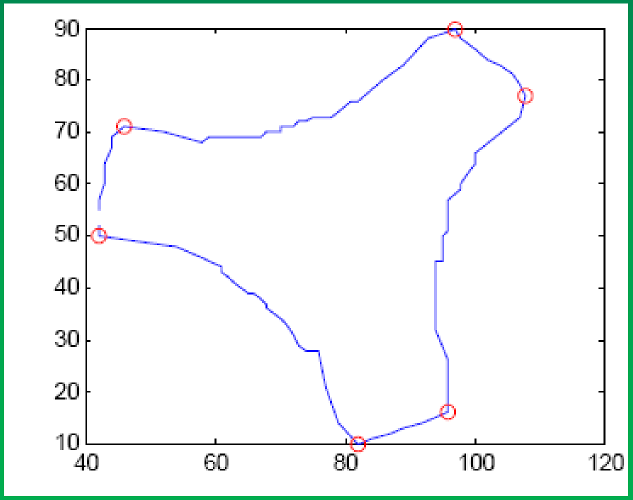
Template matching

Feature point position and orientation w. r. t. local coordinates are pre-stored in data base



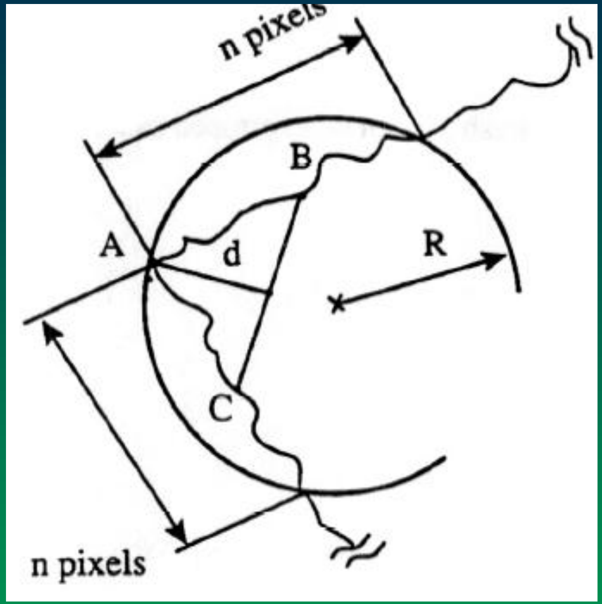






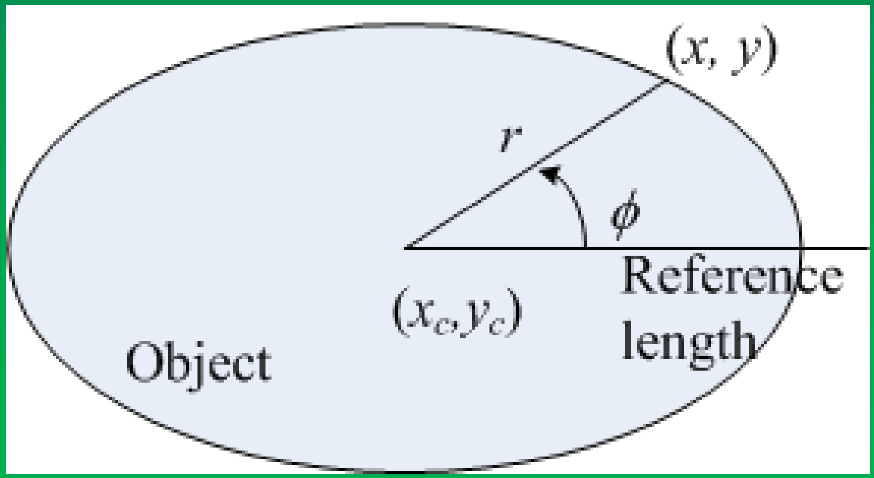
Template matching algorithm

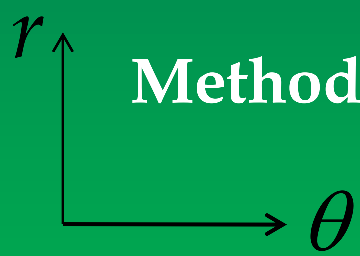
Feature representation *(many methods)*



Method of median length

B = Average location of n pixels on one side
C = Average location of n pixels on the other side
d = length of the median
A is a critical point if d > threshold distance, and
d is maximum in a neighborhood of '2n' pixels.

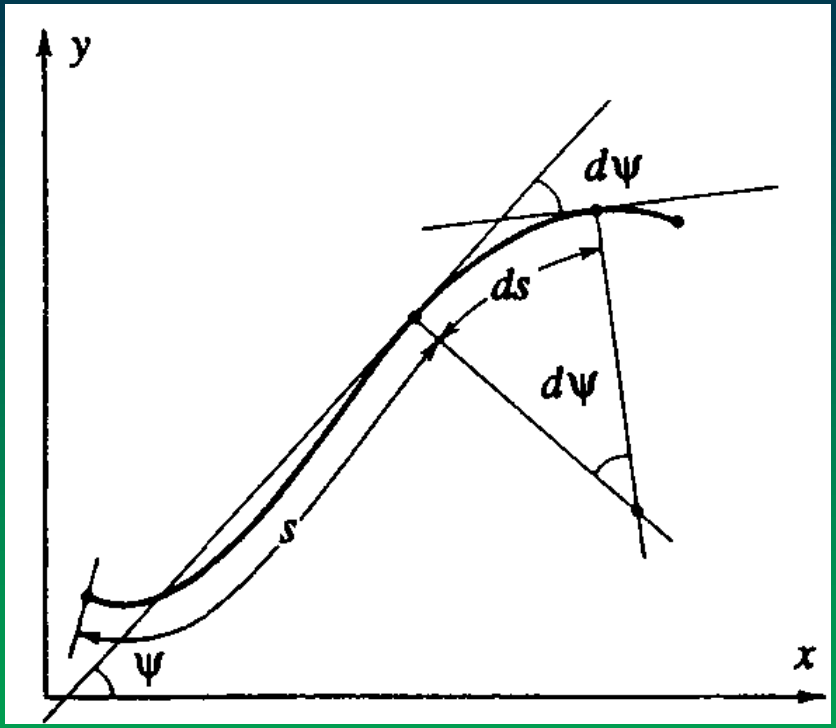




Method of r-θ plot

Feature representation (many methods)

Method of Curvature



$$g_\sigma(s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-s^2}{2\sigma^2}\right)$$

$$X(s,\sigma) = x * g_\sigma = \int_{-\infty}^{\infty} x(s-u) g_\sigma(u) du$$

$$Y(s,\sigma) = y * g_\sigma = \int_{-\infty}^{\infty} y(s-u) g_\sigma(u) du$$

$$\boxed{K_\sigma(X,Y) = \dot{X}\ddot{Y} - \dot{Y}\ddot{X}}$$

$$\boxed{\text{Curvature, } K(x,y) = \frac{\partial \psi}{\partial s} = \dot{x}\ddot{y} - \dot{y}\ddot{x}}$$

$$\dot{x} = \frac{\partial x}{\partial s} \quad \dot{y} = \frac{\partial y}{\partial s} \quad \ddot{x} = \frac{\partial^2 x}{\partial s^2} \quad \ddot{y} = \frac{\partial^2 y}{\partial s^2}$$

Scale-space curvature optimization

$$\partial K(X,Y) / \partial s = 0$$

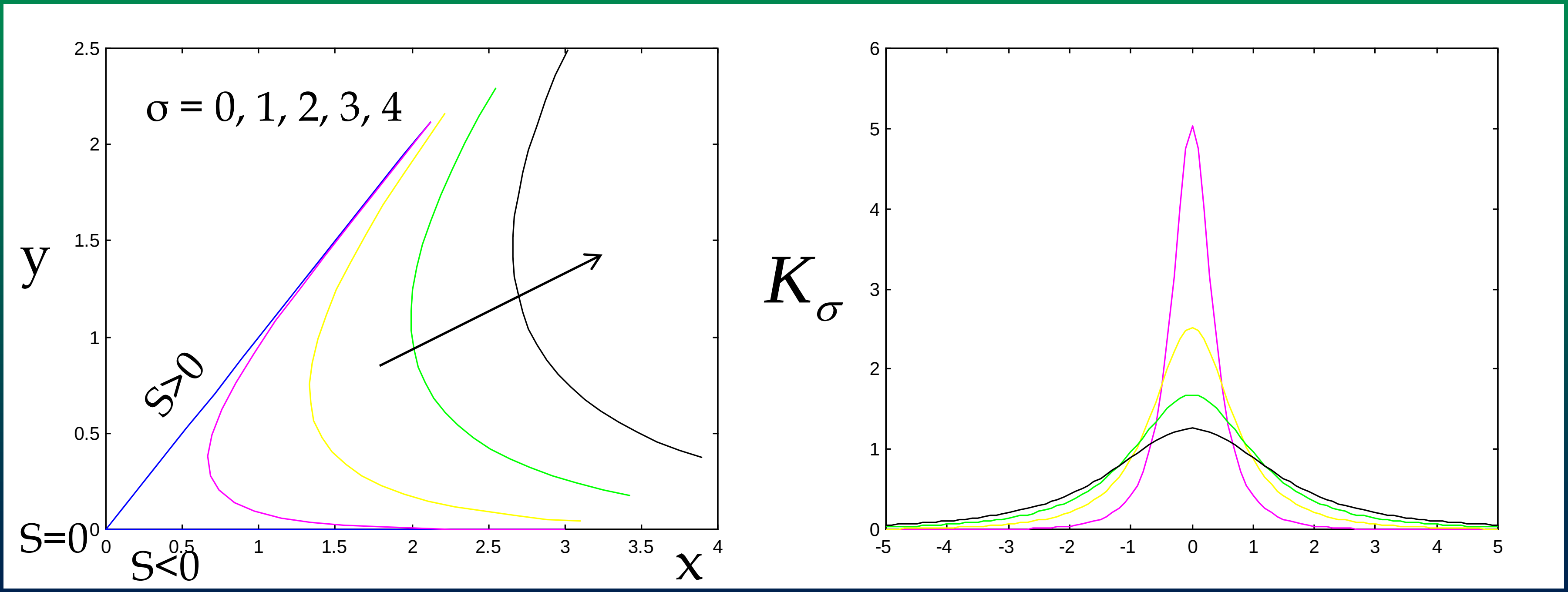
$$\dot{X}(s,\sigma)\dddot{Y}(s,\sigma) - \dot{Y}(s,\sigma)\dddot{X}(s,\sigma) = 0$$

Feature representation *(an example)*

$$x(s)=\begin{cases} -s & s<0 \\ s\cos\theta & s\ge 0 \end{cases}$$

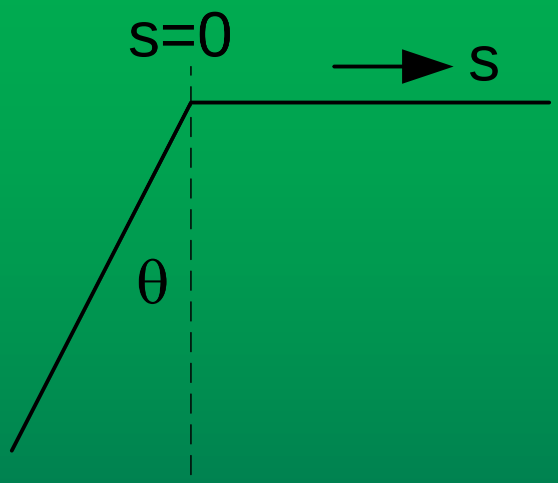
$$K_\sigma(s)=\dot{X}\ddot{Y}-\dot{Y}\ddot{X}=-g_\sigma\cos\theta$$

$$y(s)=\begin{cases} 0 & s<0 \\ s.\sin\theta & s\ge 0 \end{cases}$$



Feature representation

$$\boxed{K_{\sigma}(X,Y)=\dot{X}\ddot{Y}-\dot{Y}\ddot{X}}$$



Scale-space curvature optimization

$$\partial K(X,Y)/\partial s=0$$

$$\dot{X}(s,\sigma)\dddot{Y}(s,\sigma)-\dot{Y}(s,\sigma)\dddot{X}(s,\sigma)=0$$

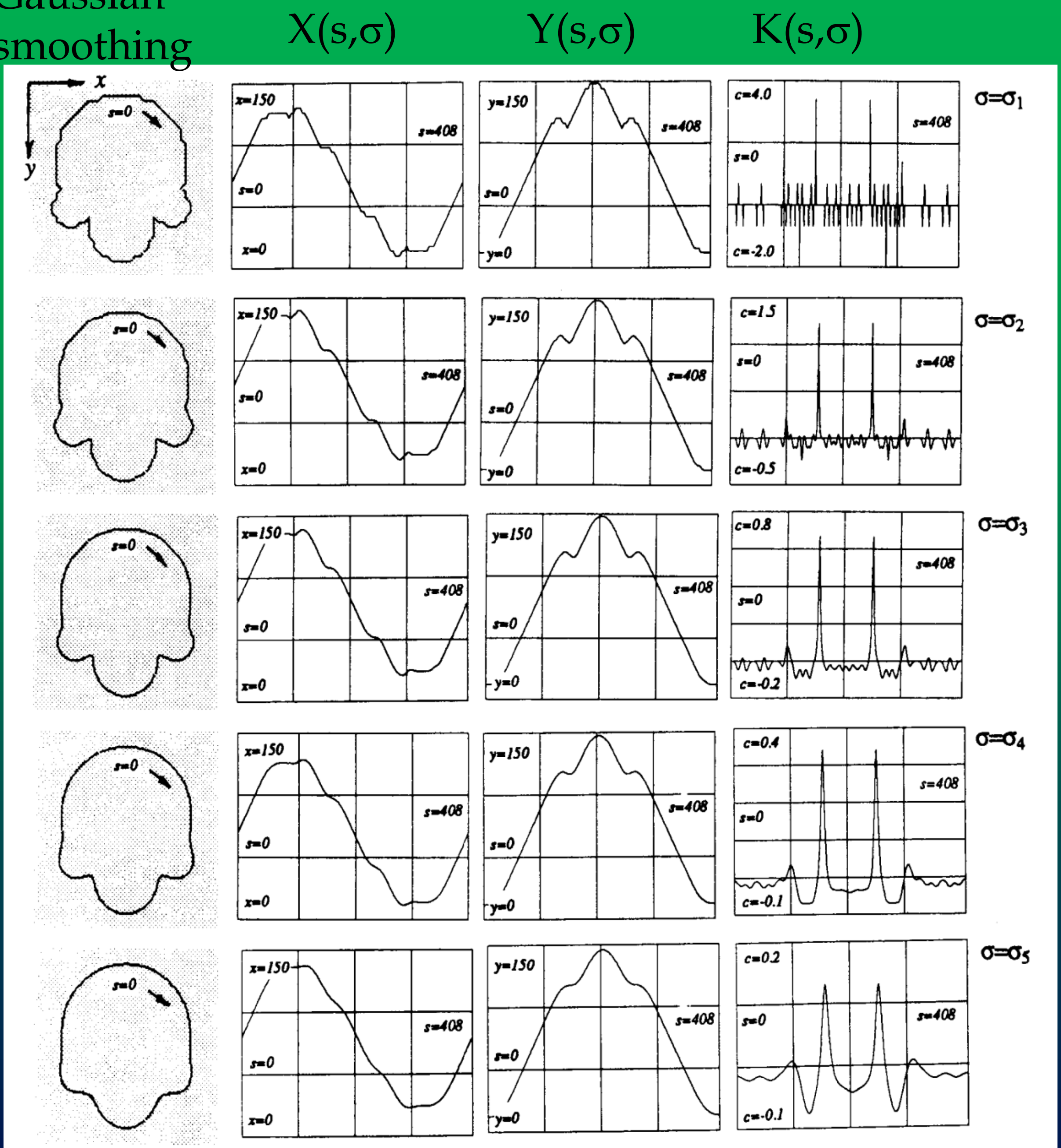
$$-\dot{g}_{\sigma}\cos\theta=0$$

At **0 = 90°, it becomes a straight line.**

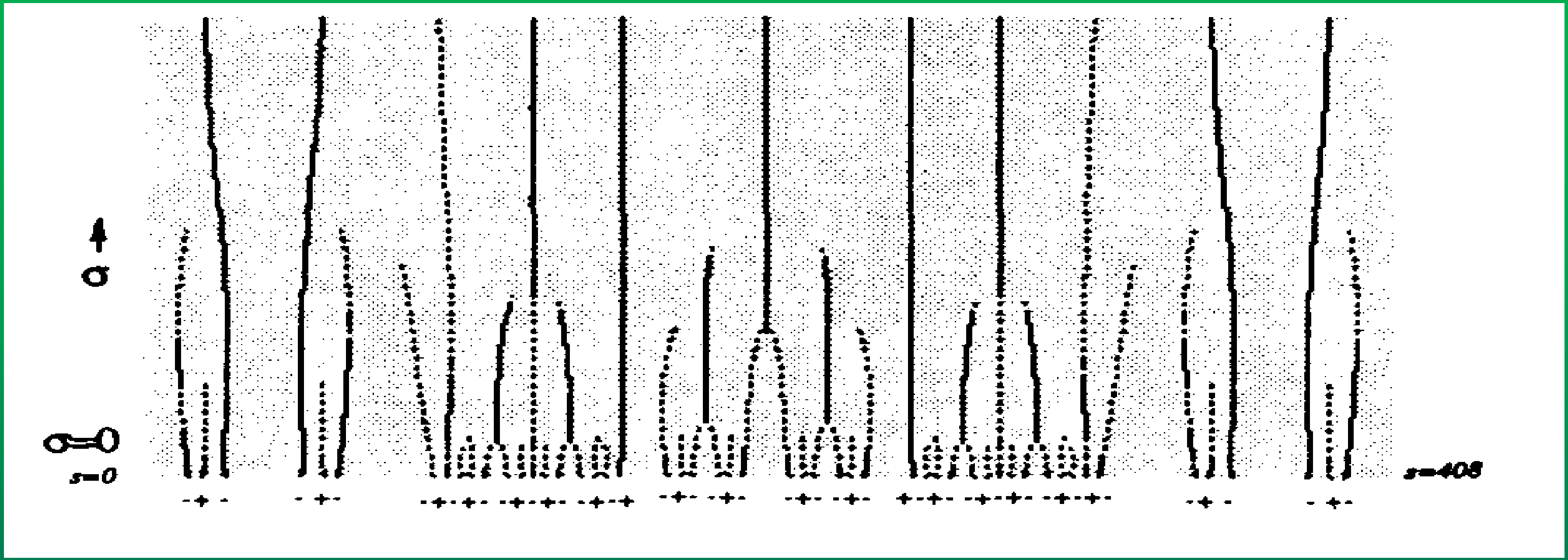
In the case of **0 > -*90°*** but not equal to ***90°,*** $\dot{g}_{\sigma}=0$

Therefore, the only solution is at s = 0 independent of the corner angle **0, and the scale parameter σ.** ***This produces a*** vertical line in scale space, that is, the absolute maxima occur at the same contour location independent of smoothing.

Result sof the Gaussian smoothing



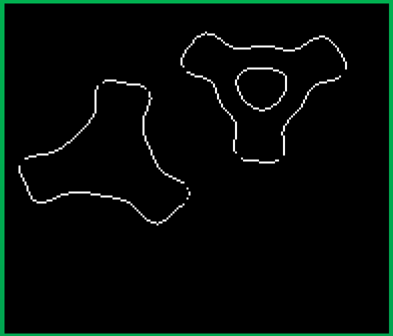
Example



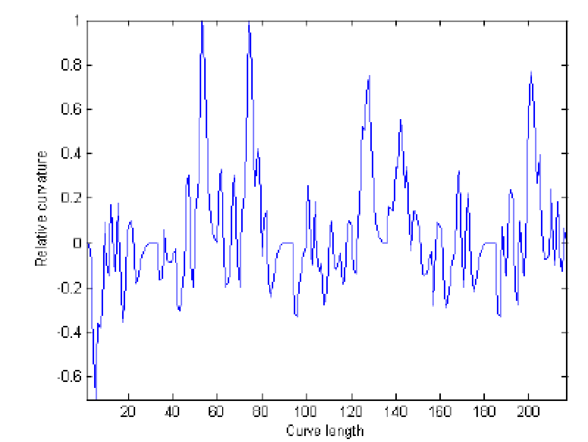
Example

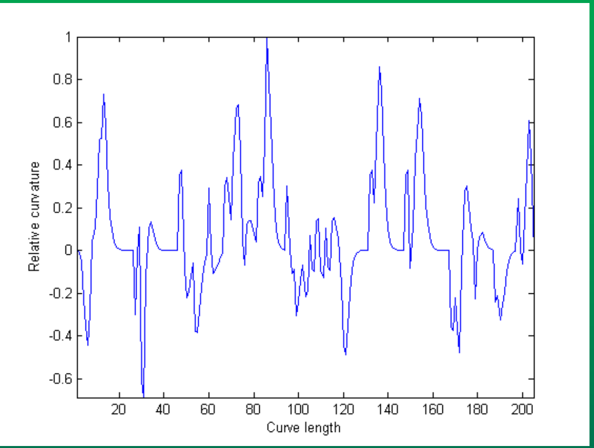
Scale space map of maxima of absolute curvature. The horizontal axis is the arc length of the curve at $\sigma = 0$. *The vertical axis is the Gaussian* function parameter *D determining the degree of smoothing. The line pattern* in the map represents the locations of the local maxima of the curvature. A +sign indicates downward concavity, and a - sign indicates upward concavity.

Example

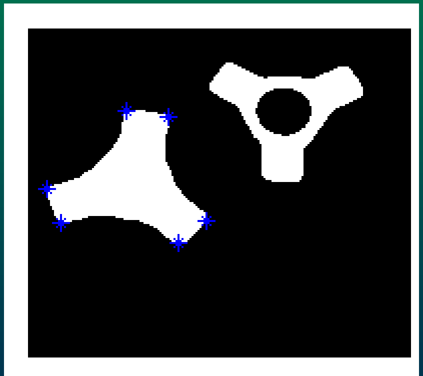


Left object

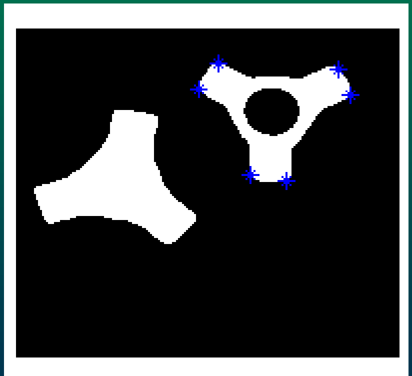




Right Object

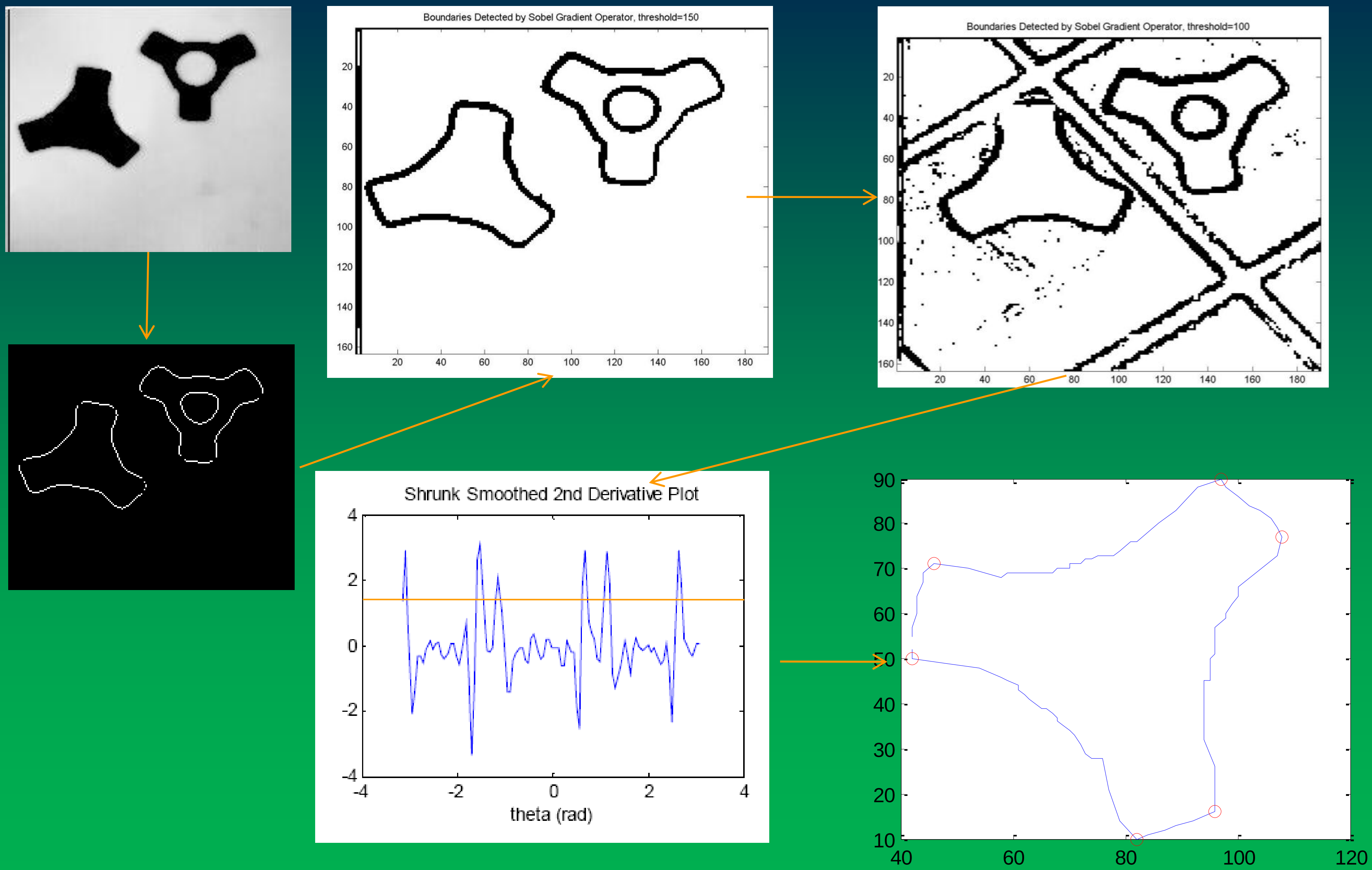


Detected corners in left object

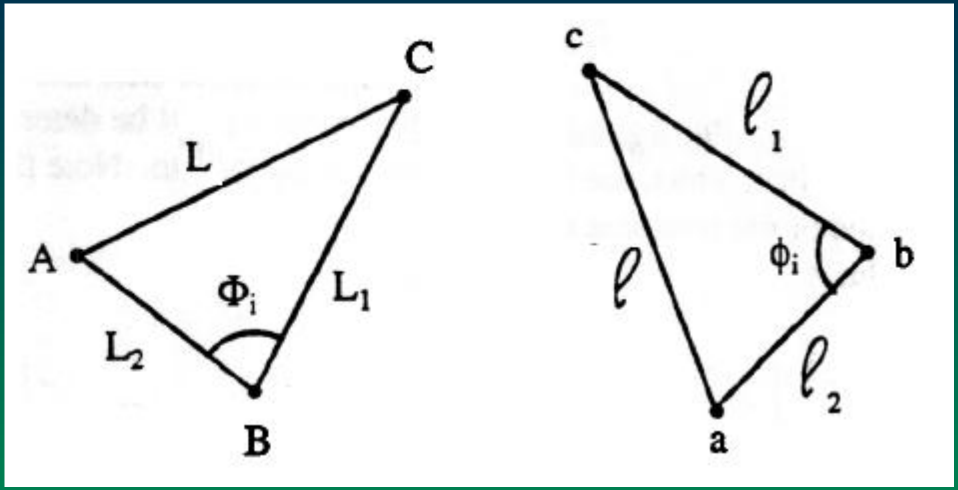


Detected corners in right object

Feature representation *(an example)*



Templating matching *(identification)*



Similar triangles:
Two constraints must be made.

$$\left|\frac{L_i}{L} - \frac{\ell_i}{l}\right| < \varepsilon_i \quad i = 1,2$$

$$\left|\Phi_i - \phi_i\right| < \varepsilon_\phi$$

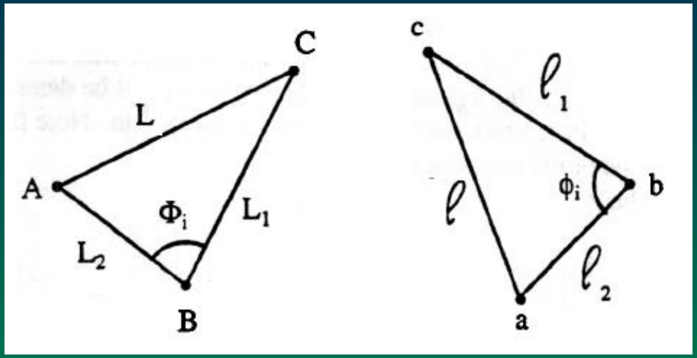
$$\frac{l}{L} = \frac{\ell_i}{L_i} = k = \text{scaling factor}$$

$$\rho(>0) = \frac{|\delta|}{\ell} = \frac{|\delta_i|}{\ell_i} = \text{Error bound (tolerance)}$$

Given the dimensional tolerance, find ε in terms of ρ.

Templating matching *(identification)*

Given the dimensional tolerance, find ε in terms of ρ.



$$\left|\frac{L_i}{L} - \frac{\ell_i}{l}\right| < \varepsilon_i \qquad (1)$$

$$\frac{l}{L} = \frac{\ell_i}{L_i} = k \qquad (2)$$

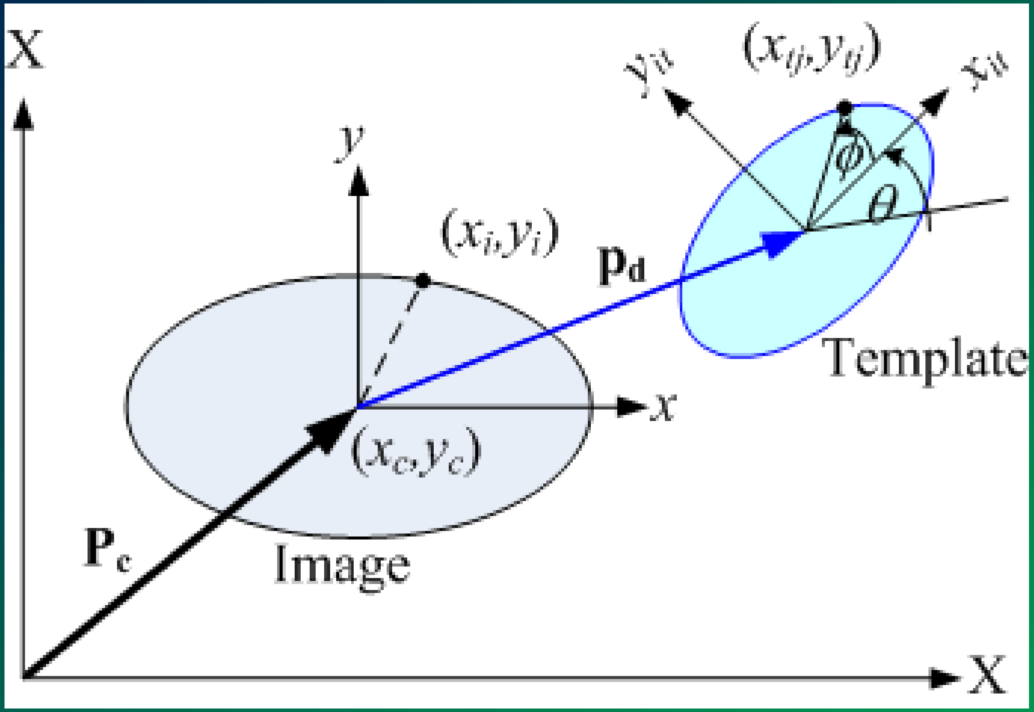
$$\rho = \frac{|\delta|}{\ell} = \frac{|\delta_i|}{\ell_i} \qquad (3)$$

$$\left|\frac{L_i}{L} - \frac{kL_i \pm \delta_i}{kL \pm \delta}\right| < \varepsilon_i$$

$$\frac{2\rho}{1 \pm \rho}\left(\frac{\ell_i}{\ell}\right) = \frac{2\rho}{1 \pm \rho}\left(\frac{L_i}{L}\right) < \varepsilon_i$$

$$\boxed{\left(\varepsilon_i\right)_{\max} = \frac{2\rho}{1-\rho}\left(\frac{\ell_i}{\ell}\right)}$$

Template matching



$$\mathbf{P}_c + \mathbf{p}_d = \begin{bmatrix} X_c + x_d \\ Y_c + y_d \end{bmatrix}$$

$$\begin{bmatrix} x_{tj} \\ y_{tj} \end{bmatrix} = r \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} + r \begin{bmatrix} \cos(\phi+\theta) \\ \sin(\phi+\theta) \end{bmatrix}$$

$$\begin{bmatrix} X_{tj} \\ Y_{tj} \end{bmatrix} = k \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{tj} \\ y_{tj} \end{bmatrix} + \begin{bmatrix} X_c + x_d \\ Y_c + y_d \end{bmatrix}$$

Template matching

If the scaled template identically matches the object, and (x_{tj}, y_{tj}) corresponds to (x_i, y_i)

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} = \begin{bmatrix} X_{tj} \\ Y_{tj} \end{bmatrix} = k \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{tj} \\ y_{tj} \end{bmatrix} + \begin{bmatrix} X_c + x_d \\ Y_c + y_d \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} k\cos\theta \\ k\sin\theta \\ X_c + x_d \\ Y_c + y_d \end{bmatrix}$$

$$\boxed{\underbrace{\begin{bmatrix} x_{tj} & -y_{tj} & 1 & 0 \\ y_{tj} & x_{tj} & 0 & 1 \end{bmatrix}}_{[\mathbf{A}]} \underbrace{\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}}_{\mathbf{Q}} = \underbrace{\begin{bmatrix} X_i \\ Y_i \end{bmatrix}}_{\mathbf{R}}}$$

$$[\mathbf{A}]\mathbf{Q} = \mathbf{R}$$

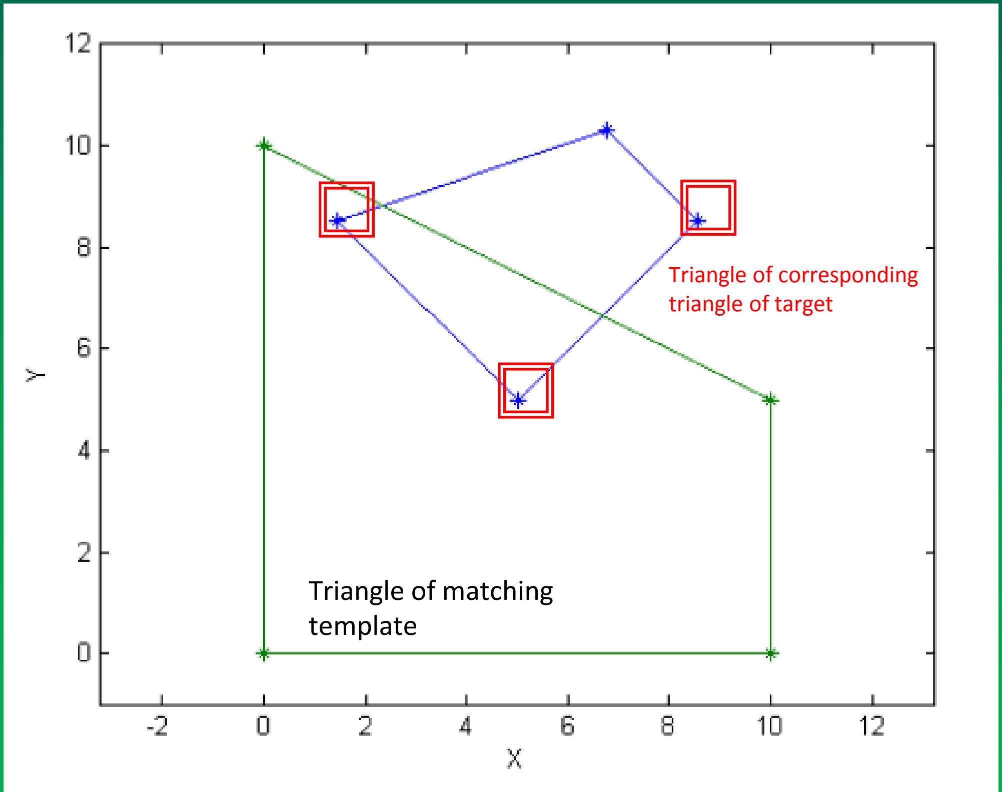
$$\mathbf{Q} = \left[\mathbf{A}^{\mathrm{T}}\mathbf{A}\right]^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{R}$$

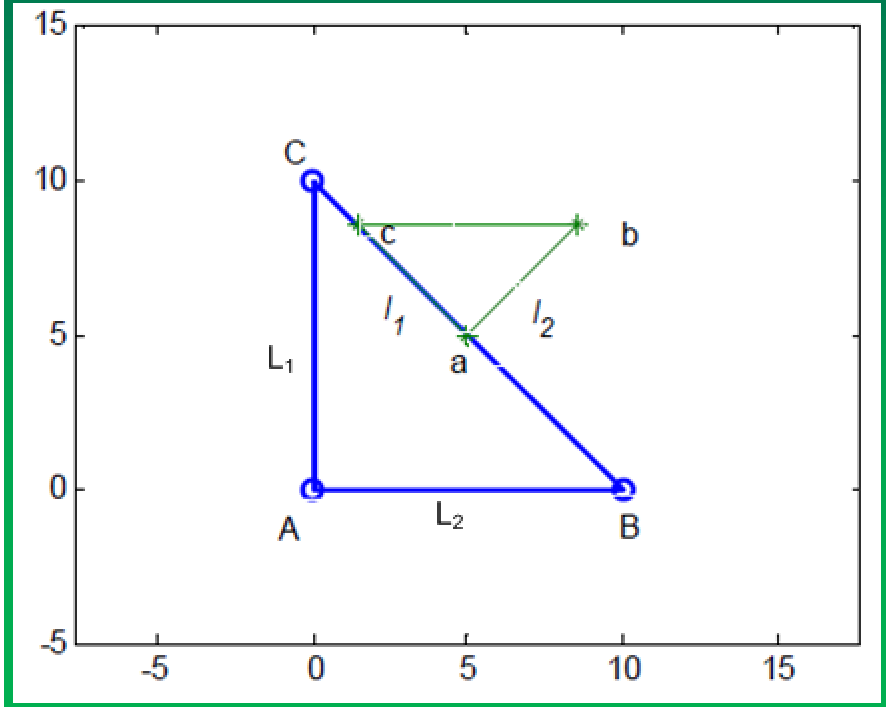
$$k = \sqrt{q_1^{\,2} + q_2^{\,2}},\ \ \theta = \tan^{-1}\left(\frac{q_2}{q_1}\right),\ \ x_d = q_3 - X_c, y_d = q_4 - Y_c$$

Template matching example

	Template		Target Object	
	x-coord	y-coord	x-coord	y-coord
Feature 1	0	0	8.5355	8.5355
Feature 2	10	0	5	5
Feature 3	10	5	1.4645	8.5355
Feature 4	0	10		

k=0.5, θ=45°, x_d=y_d=5



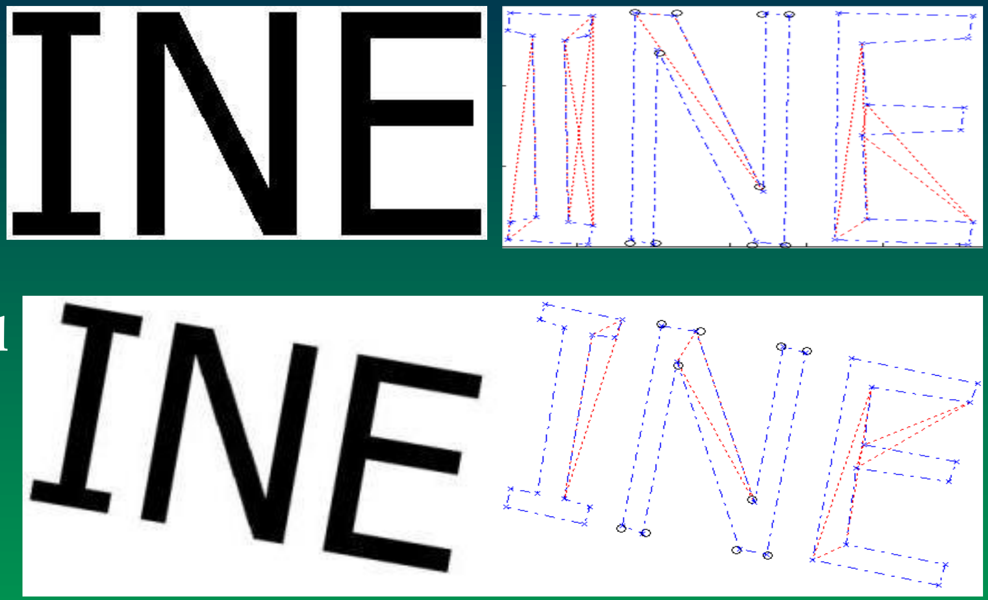


Template matching example

Template Image **Pattern Result**

Original

Scaled and Rotated



	x_d	y_d	Scale, k	Θ	E_M
	87	4	0.9925	0.5443^o	21
	85	15	15	0.7404	10.7^o

$$E_M = \sum_{j=1}^{n} \sqrt{\left(x_{tj} - x_i\right)^2 + \left(y_{tj} - y_i\right)^2}$$