

CS443: Digital Imaging and Multimedia  
Perceptual Grouping  
Detecting Lines and Simple Curves

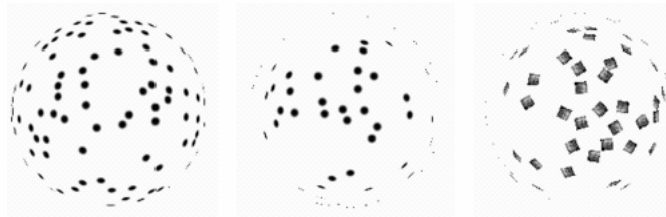
Spring 2008  
Ahmed Elgammal  
Dept. of Computer Science  
Rutgers University

## Outlines

- Perceptual Grouping and Segmentation
- Detecting Lines with Hough Transform
- Detecting Circles and Ellipses.
  
- Sources:
  - Burger and Burge “Digital Image Processing” Chapter 9
  - Fosyth and Ponce “Computer Vision a Modern approach”

## Mid-level vision

- Vision as an inference problem:
  - Some observation/measurements (images)
  - A model
  - Objective: what caused this measurement ?
- What distinguishes vision from other inference problems ?
  - A lot of data.
  - We don't know which of these data may be useful to solve the inference problem and which may not.
    - Which pixels are useful and which are not ?
    - Which edges are useful and which are not ?



Why do these tokens belong together?

It is difficult to tell whether a pixel (token) lies on a surface by simply looking at the pixel

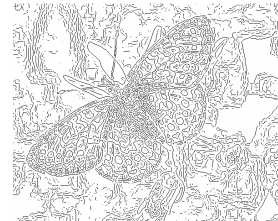
## Segmentation

- Can we achieve a compact and suggestive representation of the interesting image data that emphasizes the properties that make it interesting
  - Segmentation
  - Grouping
  - Perceptual organization
  - Fitting
- What is interesting and what is not depends on the application



## General ideas

- tokens
  - whatever we need to group (pixels, points, surface elements, etc., etc.)
- top down segmentation
  - tokens belong together because they lie on the same object
- bottom up segmentation
  - tokens belong together because they are locally coherent
- Grouping (or clustering)
  - collect together tokens that “belong together”
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?



## Segmentation

Different problems – same problem: segmentation

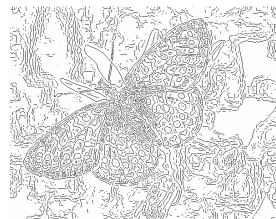
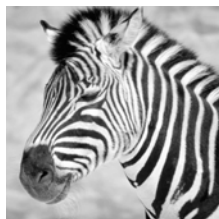
- Summarizing a video: segment a video into shots, find coherent segments in the video, find key frames...
- Finding machine parts: finding lines, circles,...
- Finding people: find body segments, find human motion patterns
- Finding buildings from aerial imagery: find polygonal regions, line segments...
- Searching a collection of images: find coherent color, texture regions, shape...
- ...

## Segmentation

- Segmentation is a big topic

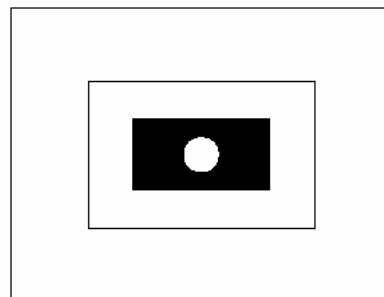
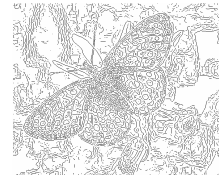
We will look into:

- Segmentation by clustering: Forming image segments:
  - How to decompose the image into “superpixels” image regions that are *coherent* in color and texture
  - Shape of the region is not that important while segmenting
- Segmentation by model fitting:
  - Fitting lines and curves to edge points:
  - Which points belong to which line, how many lines ?
  - What about more complicated models, e.g. fitting a deformable contour!



## Segmentation as Clustering

- Objective: Which components of a data set naturally belong together
- Partitioning – Decomposition:
  - Starting from a large data set how to partition it into pieces given some notion of association between data items
    - Decompose an image into regions that have coherent color and texture
    - Decompose a video sequence into shots
- Grouping
  - Collect sets of data item that make sense together given our notion of association
    - Collect together edge segments that seems to belong to a line
- Question: what is our notion of association ?

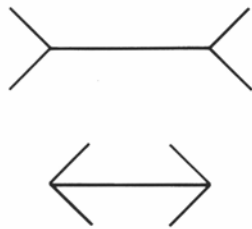


- One view of segmentation is that it determines which component of the image form the figure and which form the ground.
- What is the figure and the background in this image?

## Grouping and Gestalt

- Gestalt: German for form, whole, group
- Laws of Organization in Perceptual Forms (Gestalt school of psychology) Max Wertheimer 1912-1923

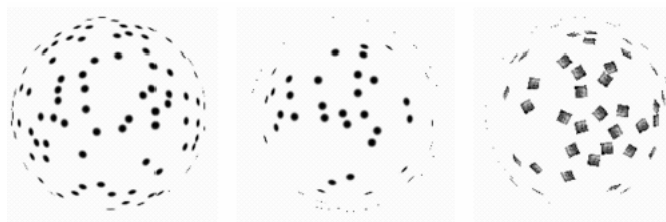
*“there are contexts in which what is happening in the whole cannot be deduced from the characteristics of the separate pieces, but conversely; what happens to a part of the whole is, in clearcut cases, determined by the laws of the inner structure of its whole”*











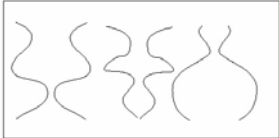

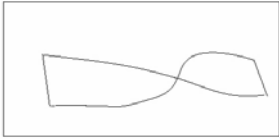
Muller-Layer effect:  
This effect arises from some property of the relationships that form the whole rather than from the properties of each separate segment.

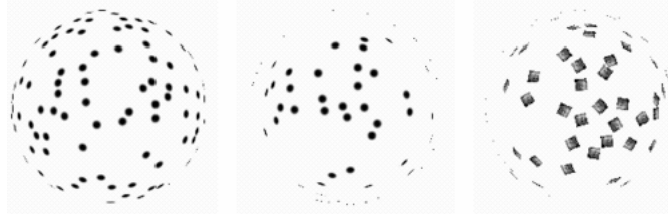
## Grouping and Gestalt

- Can we write down a series of rules by which image elements would be associated together and interpreted as a group ?
- What are the factors that makes a set of elements to be grouped
- Human vision uses these factors in some way

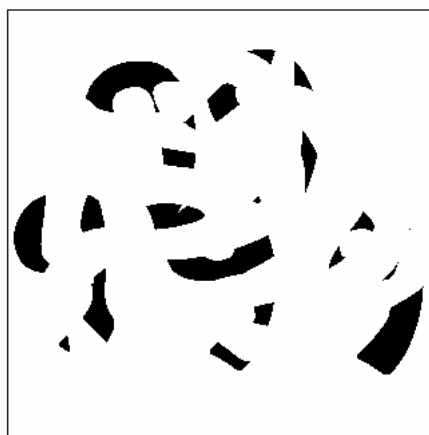


	Not Grouped
	<b>Proximity:</b> Tokens that are nearby tend to be grouped together.
	<b>Similarity:</b> Similar tokens tend to be grouped together.
	Similarity
	<b>Common Fate:</b> Tokens that have coherent motion tend to be grouped together
	<b>Common Region:</b> Tokens that lie inside the same closed region tend to be grouped together
	

	<b>Parallelism:</b> Parallel curves or tokens tend to be grouped together
	<b>Symmetry:</b> Curves that lead to symmetric groups are grouped together
	<b>Continuity:</b> Tokens that lead to continuous curves tend to be grouped
	<b>Closure:</b> Tokens or curves that tend to lead to closed curves tend to be grouped together.



Familiar configuration: tokens that, when grouped, lead to a familiar object tend to be grouped



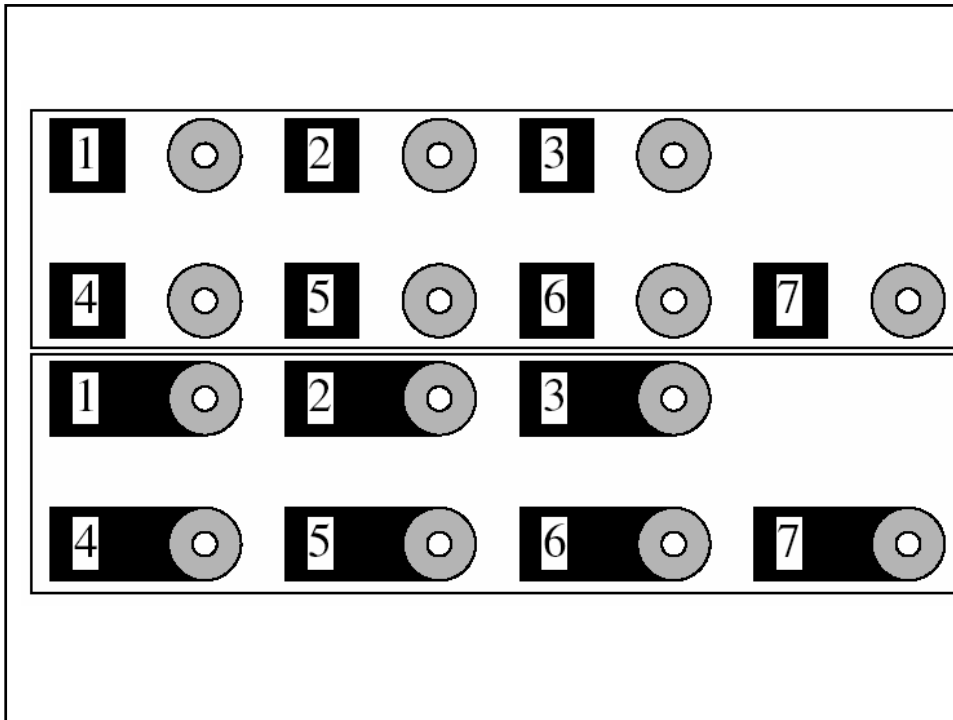




Occlusion appears to be a very important cue in grouping

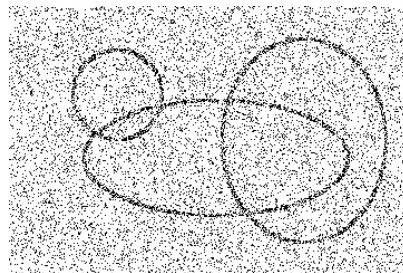
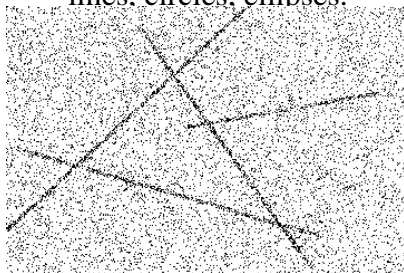


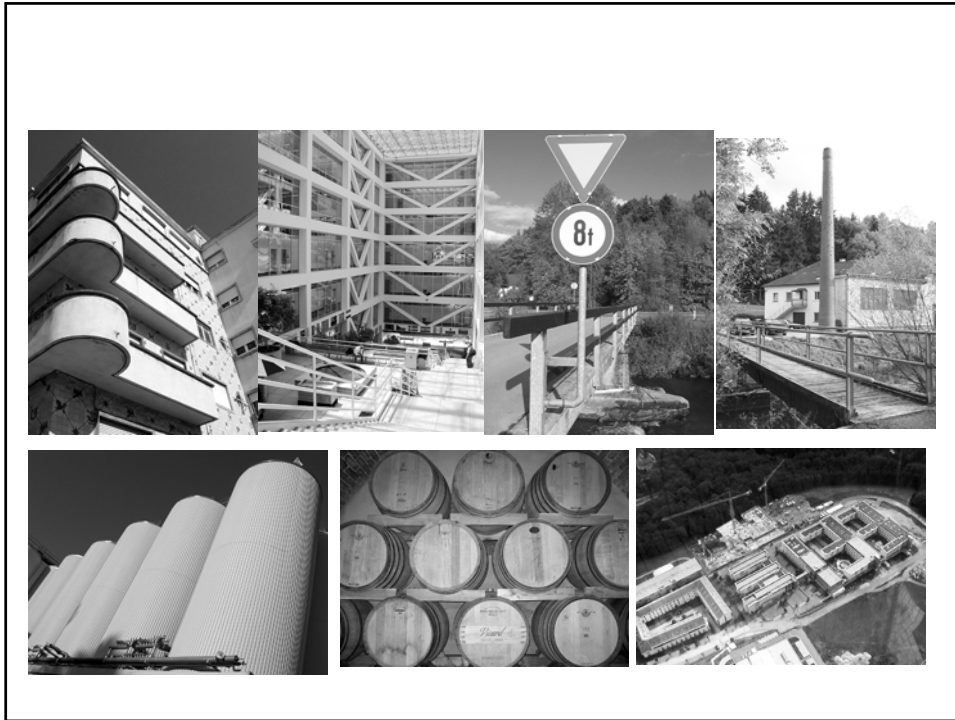
Illusory contours: tokens are grouped together because they provide a cue to the presence of an occluding object



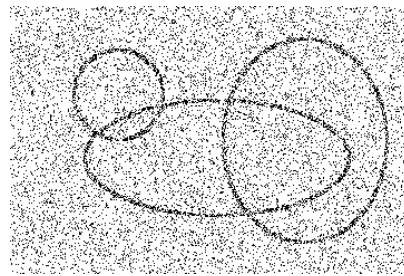
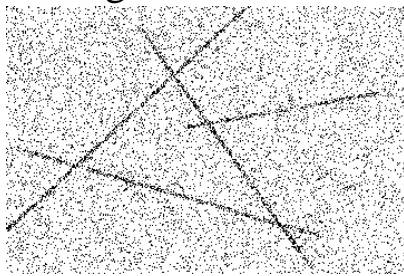
## Detecting lines and simple curves

- Finding salient structures
- How to group collections of edge points (or corners) belonging to the same line or curve?
- Search for globally apparent structures that inherently express certain common shape features
- Many man-made objects exhibits simple geometric forms: lines, circles, ellipses.

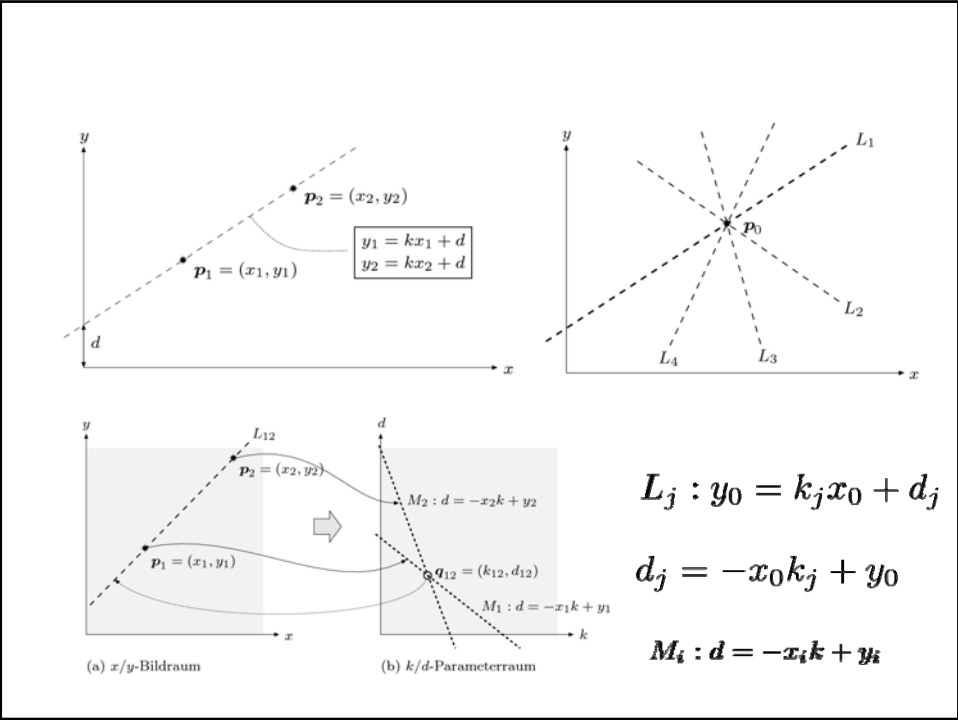
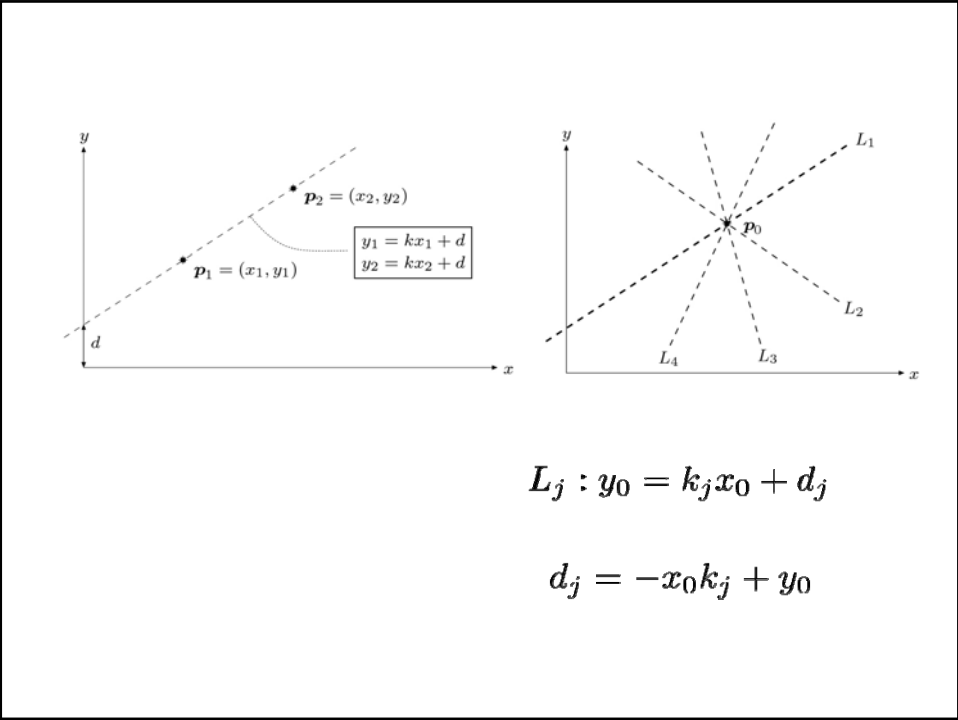




## Hough Transform

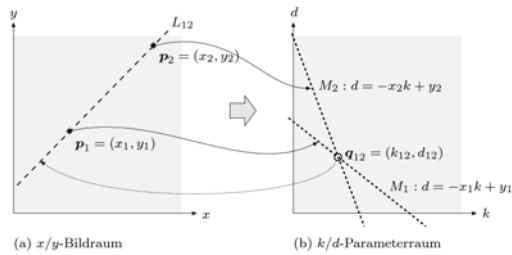


- Goal: find which tokens belongs to which objects
- Hough transform: localizing any shape that can be defined parametrically within a distribution of points (Paul Hough)
- Example: lines, circles, ellipses.

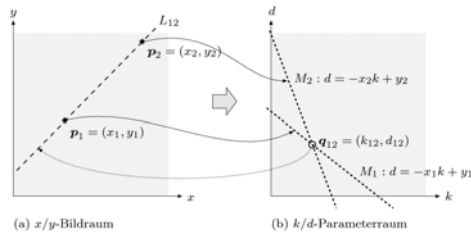


- All lines passing through an image point  $(x_i, y_i)$  are characterized by a line in the parameter space with equation:

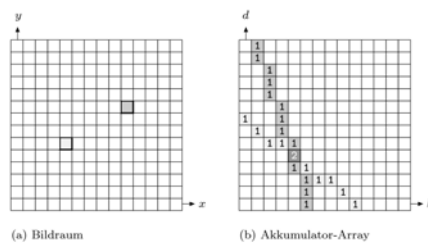
$$M_i : d = -x_i k + y_i$$



<i>Image Space (x, y)</i>		<i>Parameter Space (k, d)</i>	
Point	$p_i = (x_i, y_i)$	$M_i : d = -x_i k + y_i$	Line
Line	$L_j : y = k_j x + d_j$	$q_j = (k_j, d_j)$	Point

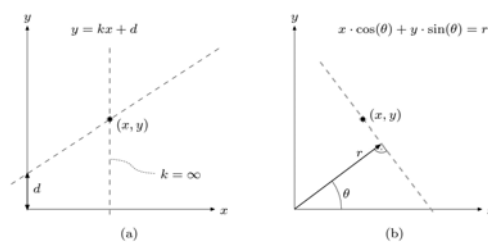


- Accumulator array: a discrete representation of the parameter space as a 2D array
- Given an image point, we increment all the points on it's corresponding line in the parameter space
- A line in the image will be the intersection of multiple lines in the parameter space.



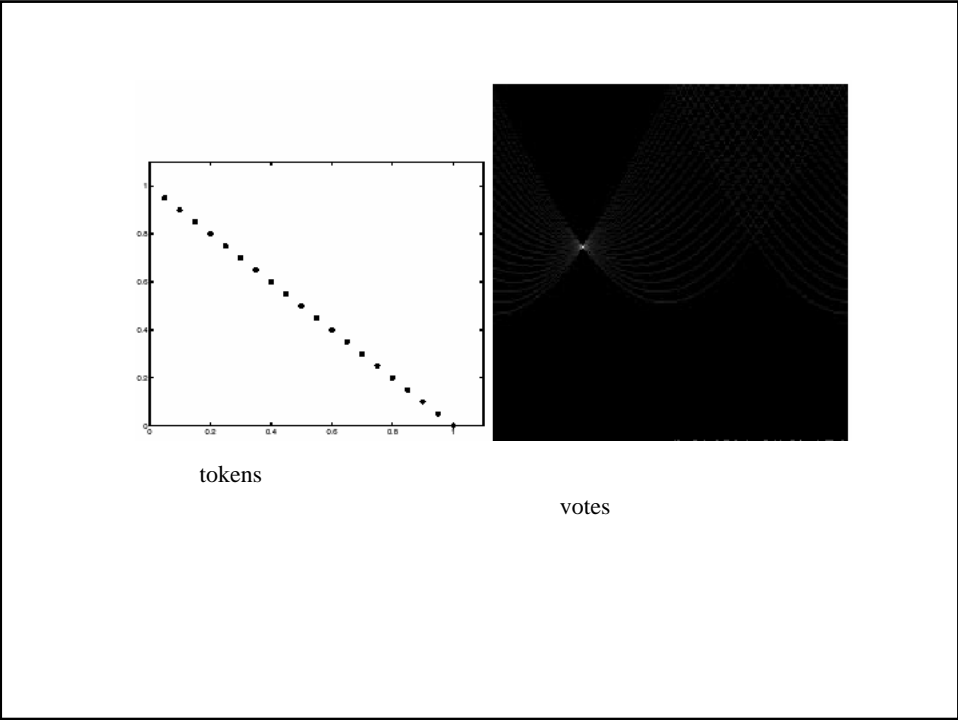
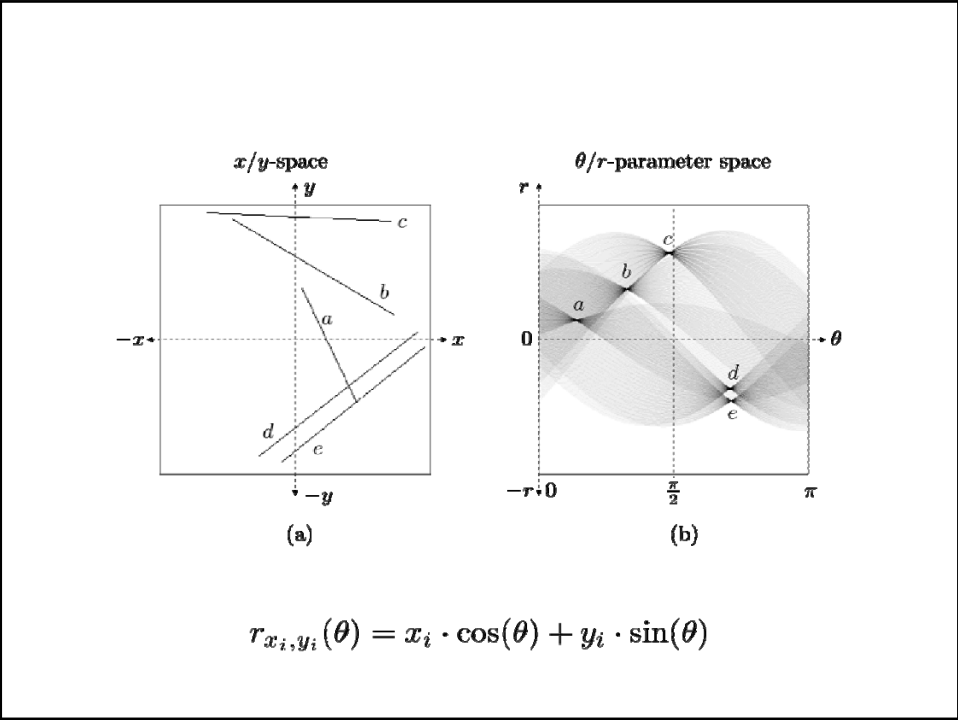
$$M_i : d = -x_i k + y_i$$

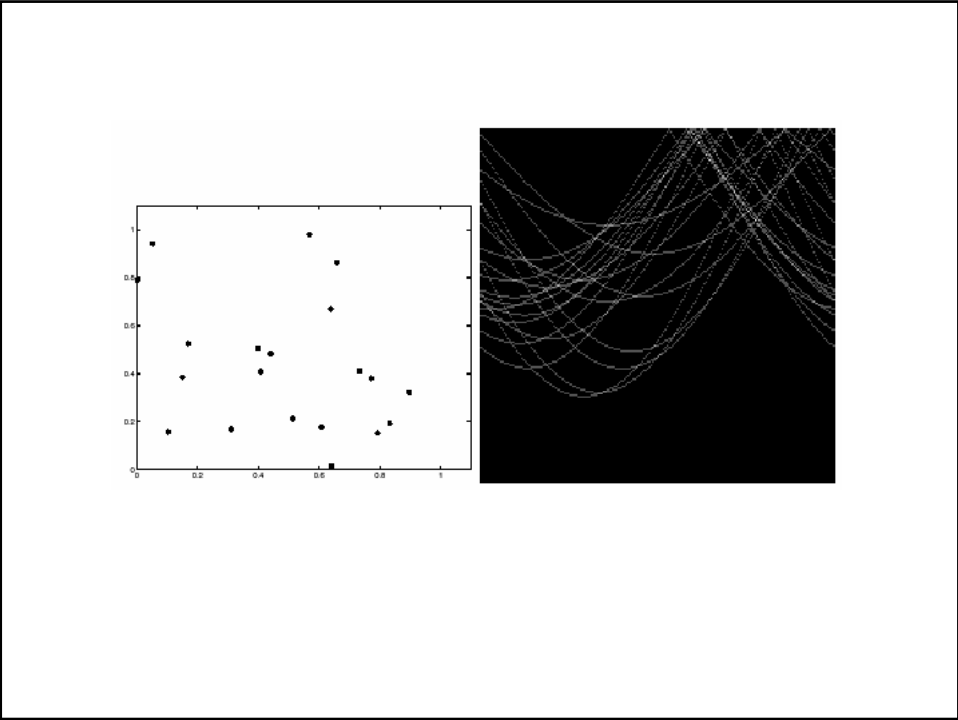
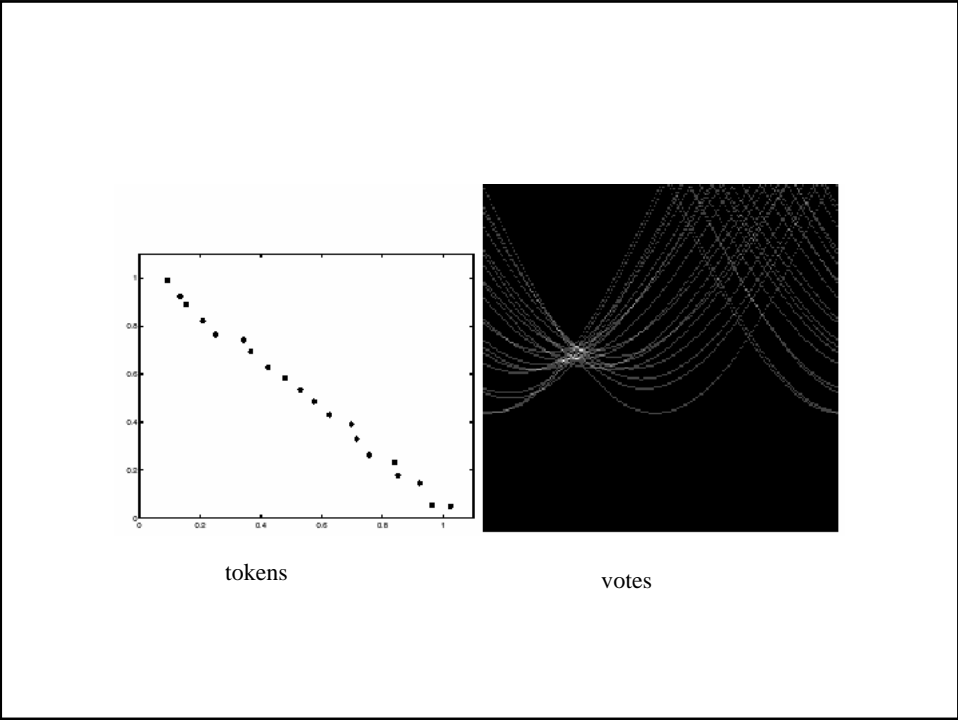
## A better line representation



- The Hessian Normal Form (HNF) of a line gives a better representation. Why?
- HNF:  $x \cdot \cos(\theta) + y \cdot \sin(\theta) = r$
- Parameter space is defined by r and theta

$$r_{x_i, y_i}(\theta) = x_i \cdot \cos(\theta) + y_i \cdot \sin(\theta)$$







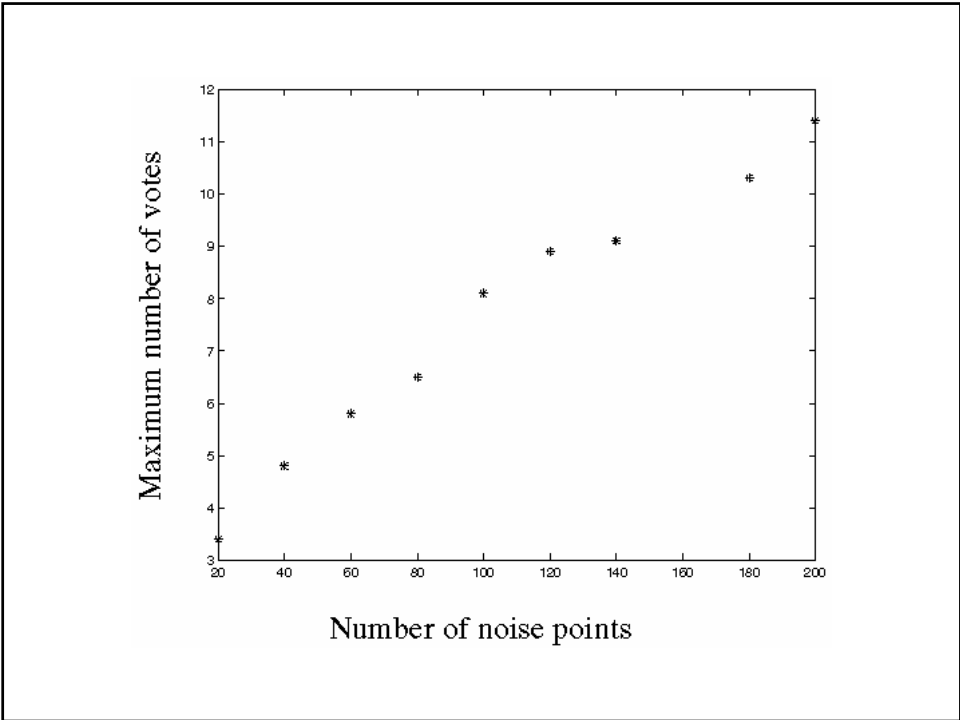
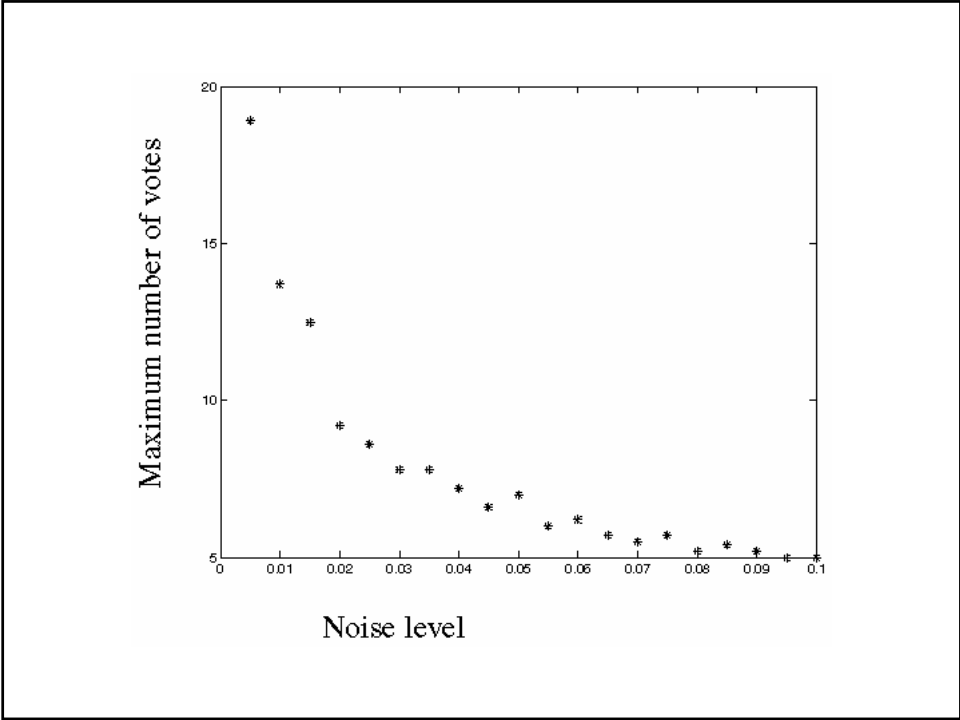
```

1: HOUGHLINES(I)
   Returns the list of parameters  $\langle \theta_i, r_i \rangle$  corresponding to the strongest
   lines found in the binary image I.
2: Set up a two-dimensional array  $Acc[\theta, r]$  of counters, initialize to 0.
3: Let  $(u_c, v_c)$  be the center coordinates of the image I
4: for all image coordinates  $(u, v)$  do
5:   if I(u, v) is an edge point then
       Get coordinate relative to the image center  $(u_c, v_c)$ :
6:      $(x, y) \leftarrow (u - u_c, v - v_c)$ 
7:     for  $\theta_i = 0 \dots \pi$  do
8:        $r_i = x \cdot \cos(\theta_i) + y \cdot \sin(\theta_i)$ 
9:       Increment  $Acc[\theta_i, r_i]$ 
   Return the list of parameter pairs  $\langle \theta_j, r_j \rangle$  for K strongest lines:
10: MaxLines  $\leftarrow$  FINDMAXLINES(Acc, K)
11: return MaxLines.

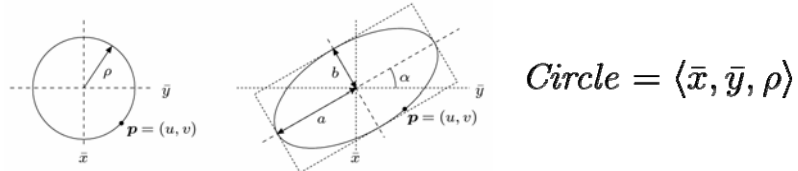
```

## Mechanics of the Hough transform

- Construct an array representing  $\theta, r$
- For each point, render the curve  $(\theta, r)$  into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
  - How to find peaks?
    - Threshold
    - Non-maxima suppression
  - How many lines?
    - count the peaks in the Hough array
  - Who belongs to which line?
    - tag the votes
  - Hardly ever satisfactory in practice, because problems with noise and cell size defeat it



## Circles and ellipses



$$\text{Circle} = \langle \bar{x}, \bar{y}, \rho \rangle$$

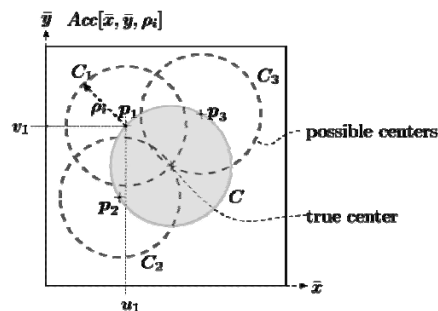
$$(u - \bar{x})^2 + (v - \bar{y})^2 = \rho^2$$

```

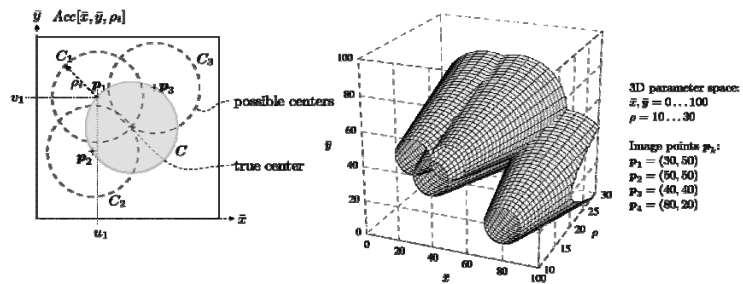
1: HOUGH_CIRCLES(I)
   Returns the list of parameters  $(\bar{x}_i, \bar{y}_i, \rho_i)$  corresponding to the
   strongest circles found in the binary image I.
2: Set up a three-dimensional array  $Acc[\bar{x}, \bar{y}, \rho]$  and initialize to 0
3: for all image coordinates  $(u, v)$  do
4:   if  $I(u, v)$  is an edge point then
5:     for all  $(\bar{x}_i, \bar{y}_i, \rho_i)$  in the accumulator space do
6:       if  $(u - \bar{x}_i)^2 + (v - \bar{y}_i)^2 = \rho_i^2$  then
7:         Increment  $Acc[\bar{x}_i, \bar{y}_i, \rho_i]$ 
8:    $MaxCircles \leftarrow \text{FINDMAXCIRCLES}(Acc)$  ▷ a list of tuples  $(\bar{x}_j, \bar{y}_j, \rho_j)$ 
9: return  $MaxCircles$ .
    
```

Better Idea:

- If we know the radius, the locations of all possible centers lie on a circle



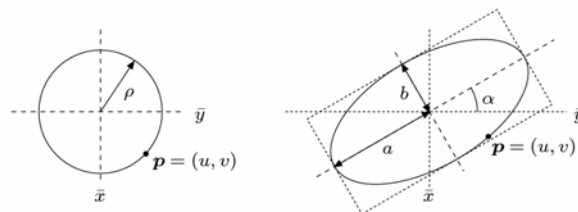
- For a given point  $(u, v)$ , at each plane  $\rho_i$  along the  $\rho$  axis, a circle centered at  $u, v$  with radius  $\rho_i$  is rendered



- Detecting Ellipses?
- An ellipse has five dimensional parameter space!

$$Ellipse = \langle \bar{x}, \bar{y}, r_a, r_b, \alpha \rangle$$

- There are better ways! Generalized Hough transform



# Circles and ellipses

