Gebze Institute of Technology
Department of Computer Engineering

Computer Vision

Segmentation and Grouping

- Obtain a compact representation from an image/motion sequence/set of tokens
- Should support application
- Broad theory is absent at present

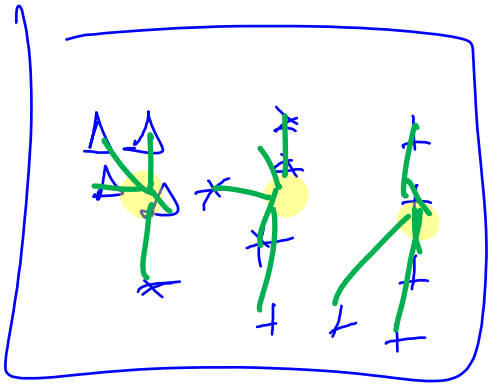
- Grouping (or clustering)
  - collect together tokens that "belong together"
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?

Segmentation as clustering

- Cluster together (pixels, tokens, etc.) that belong together
- Agglomerative clustering
  - attach closest to cluster it is closest to
  - repeat
- Divisive clustering
  - split cluster along best boundary
  - repeat
- Point-Cluster distance
  - single-link clustering
  - complete-link clustering
  - group-average clustering

K-ortalamalar

K-Means



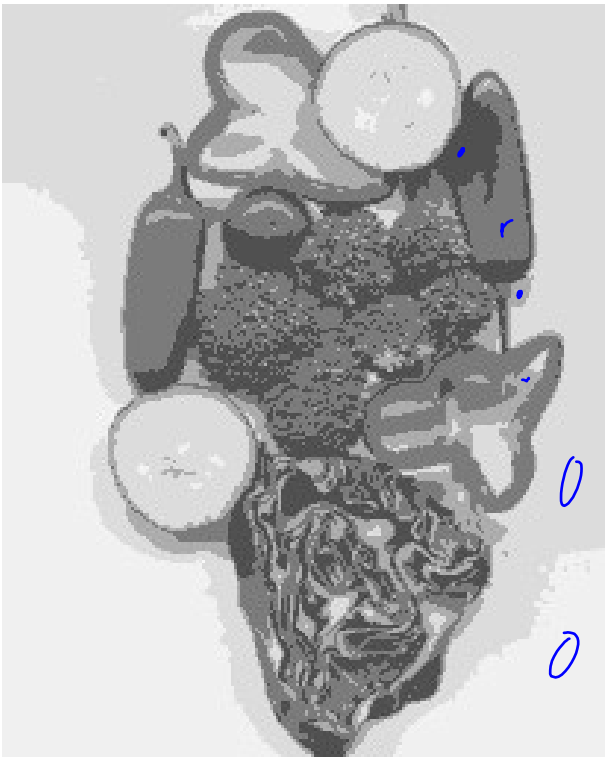
- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error
- can't do this by search, because there are too many possible allocations.

- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

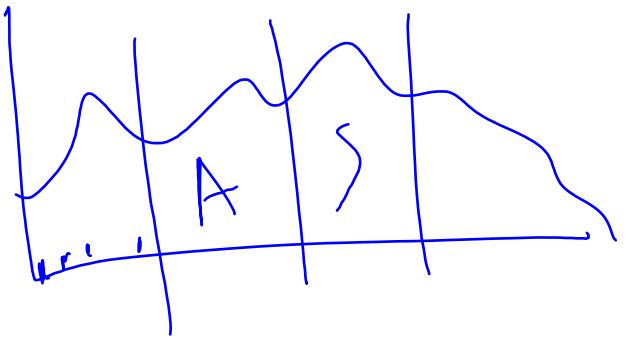
Image

Clusters on intensity

Clusters on color



K-means clustering using intensity alone and color alone (5 clusters)



Image

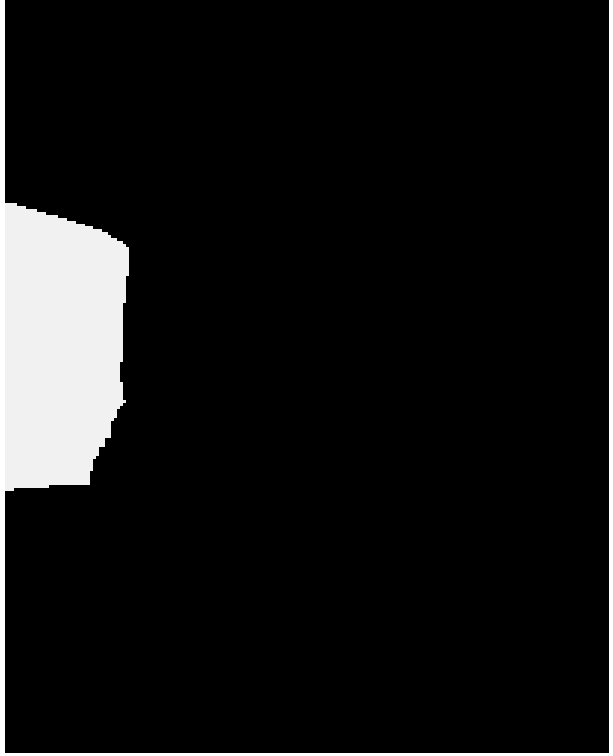
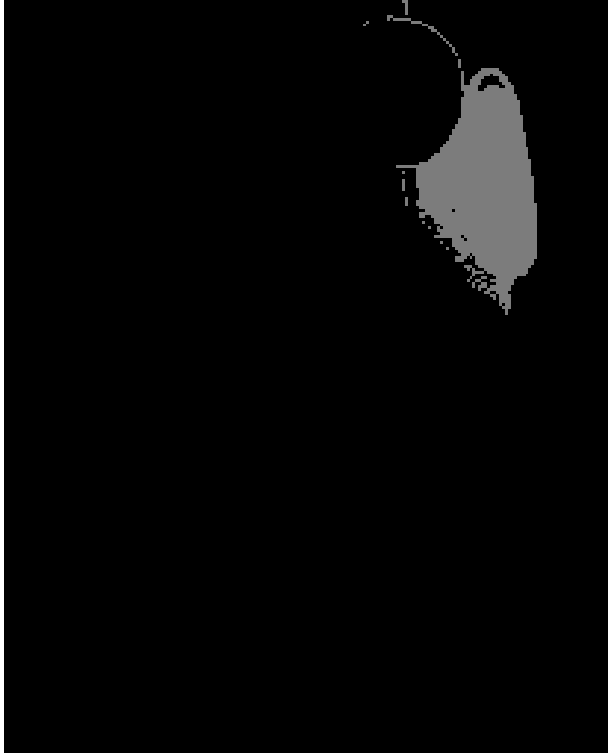
Clusters on color

K-means using color alone, 11 segments

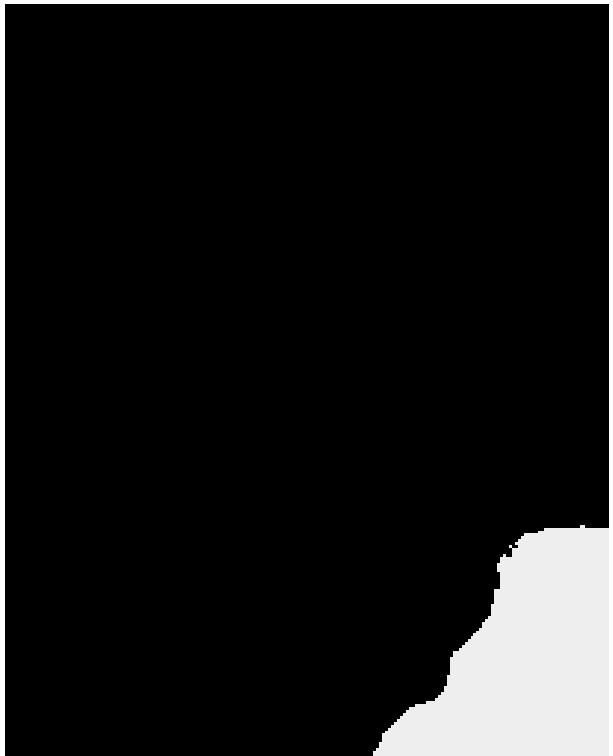
x $\quad (x_1 - y_1)^2 + (x_5 - y_5)^2 + (x_6 - y_6)^2$

y $\quad (x_x - y_4)^2 + (x_y - y_y)^2$

K-means using
color alone,
11 segments.

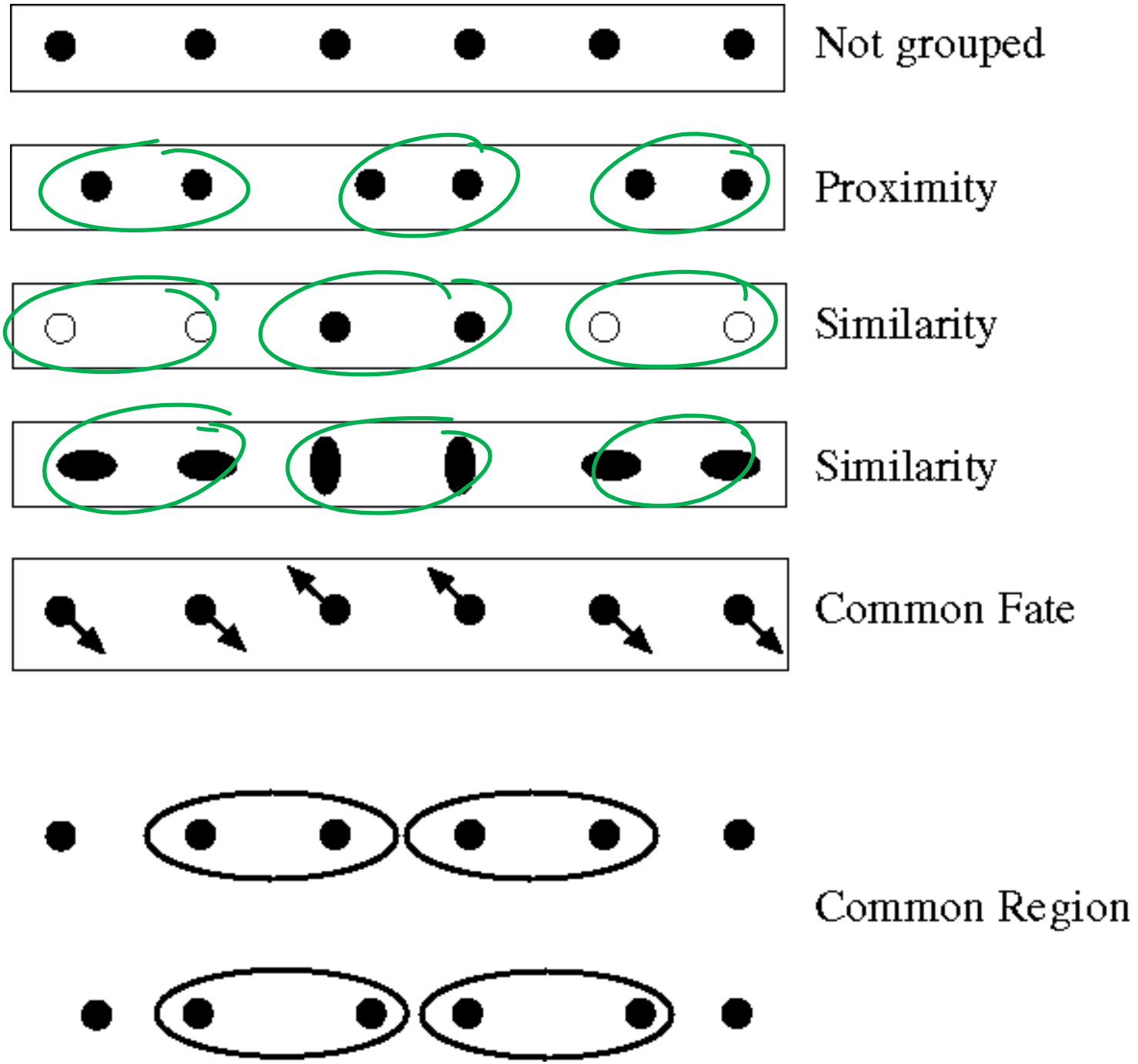




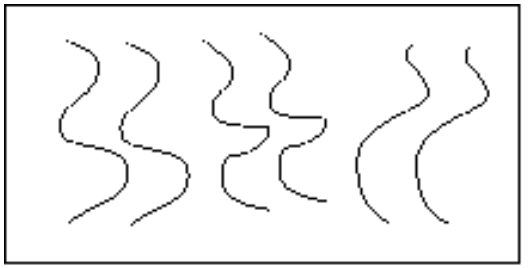
K-means using colour and position, 20 segments



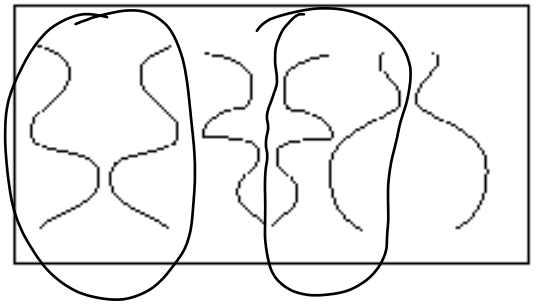

Basic ideas of grouping in humans

- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground
- Gestalt properties
  - elements in a collection of elements can have properties that result from relationships (Muller-Lyer effect)
    - gestaltqualitat
  - A series of factors affect whether elements should be grouped together
    - Gestalt factors

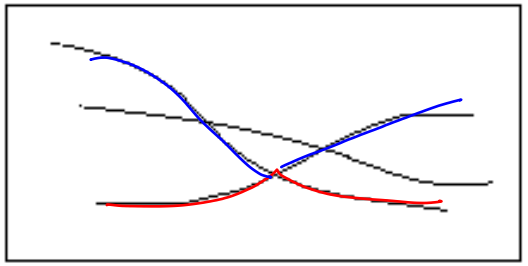

Not grouped
Proximity
Similarity
Similarity
Common Fate
Common Region



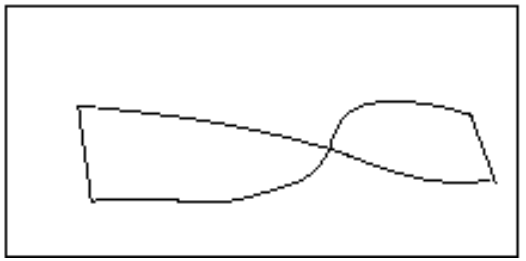
Parallelism



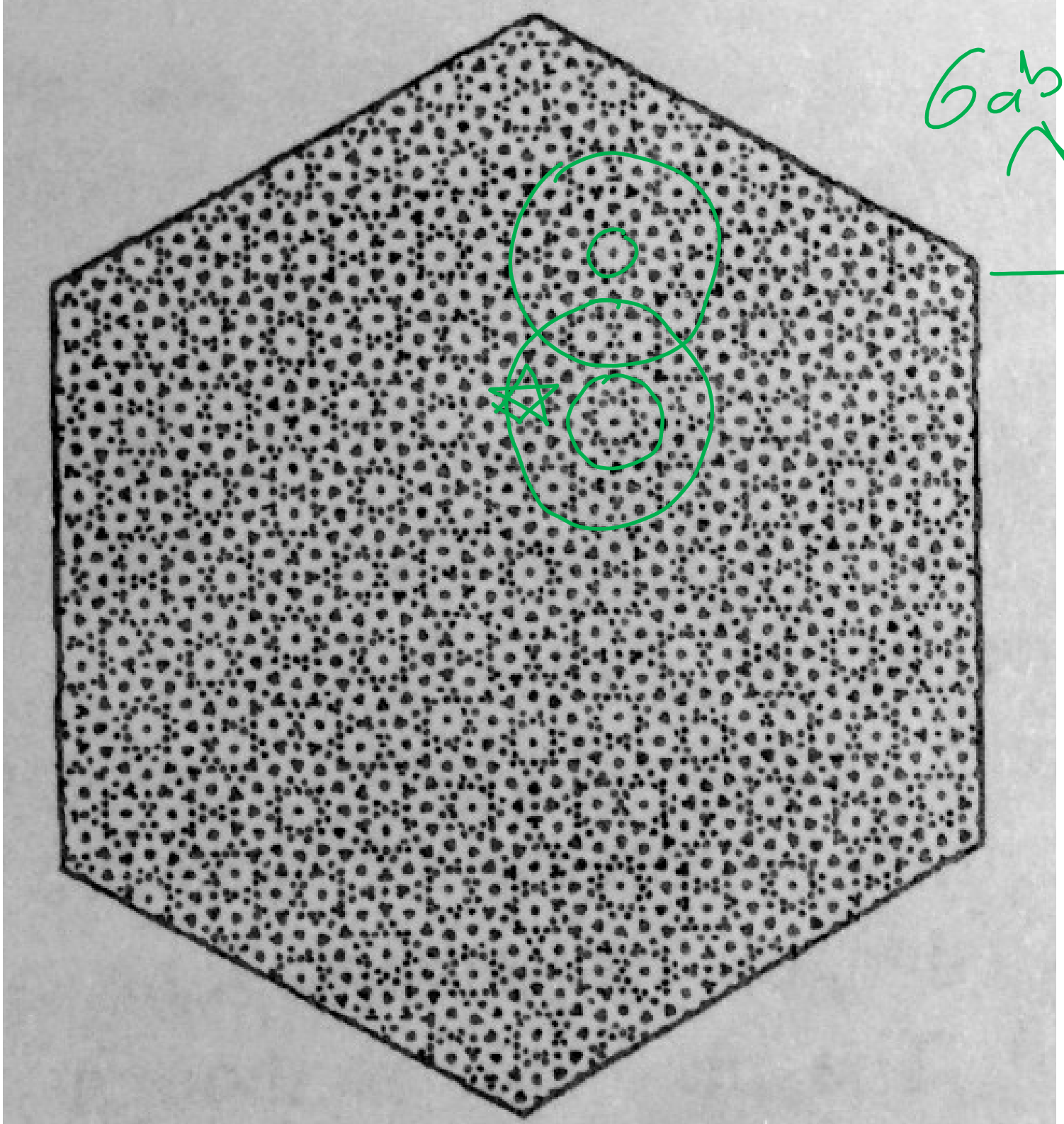
Symmetry



Continuity

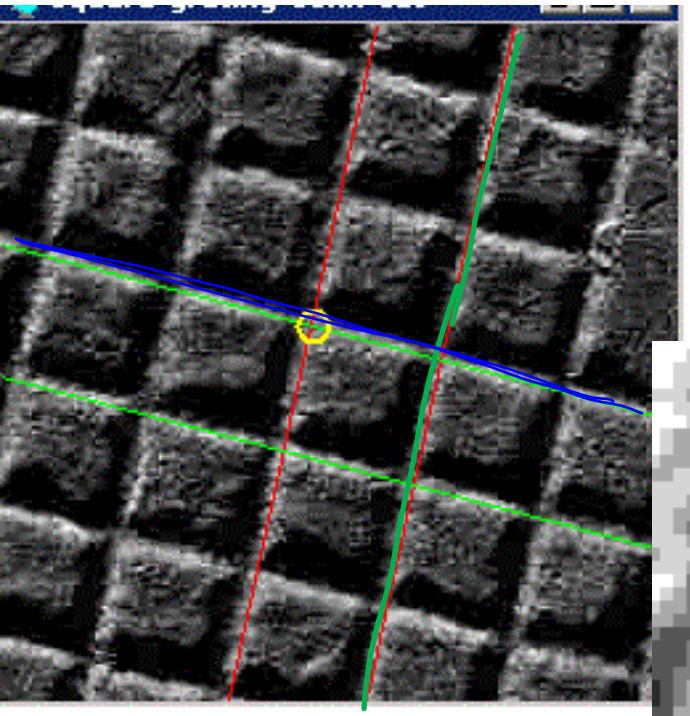


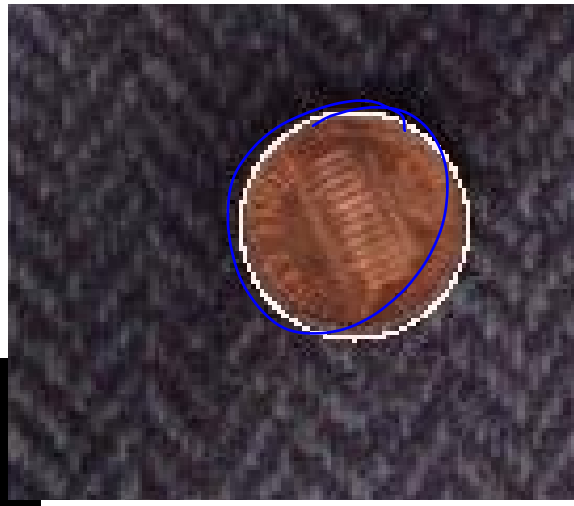
Closure

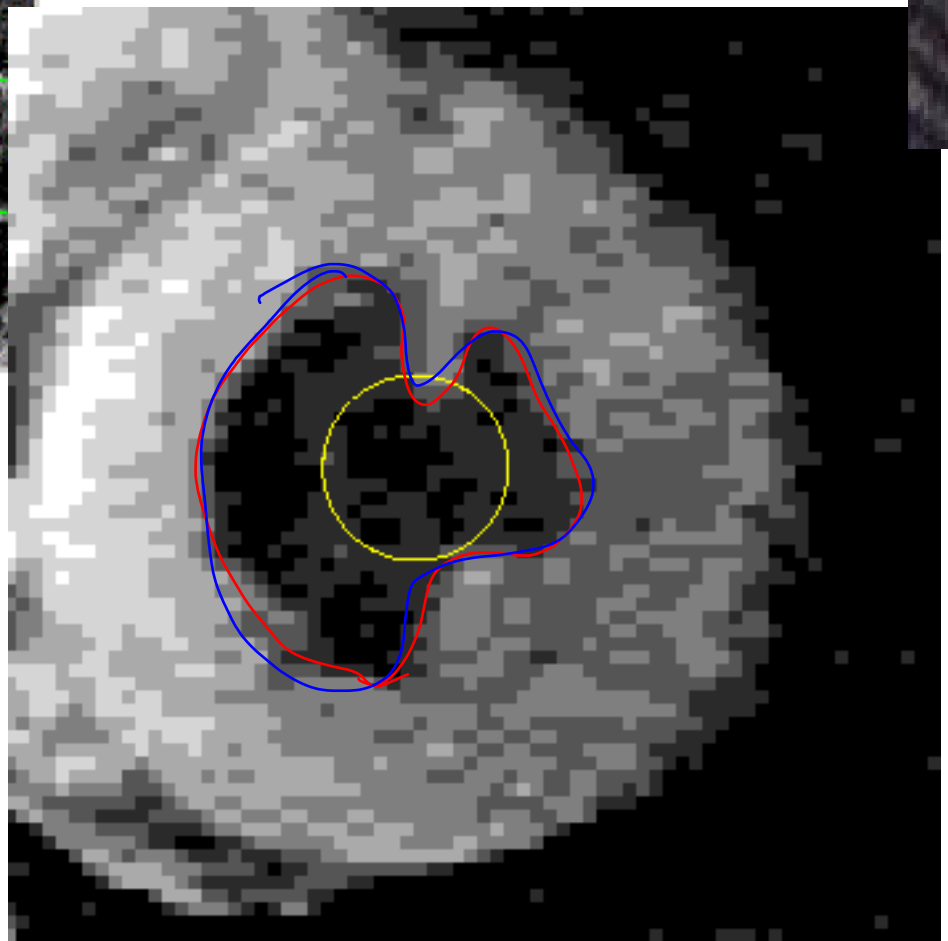

Gabs

Model Fitting (Line and Curve Detection)

Hough Line







Grouping and Model Fitting

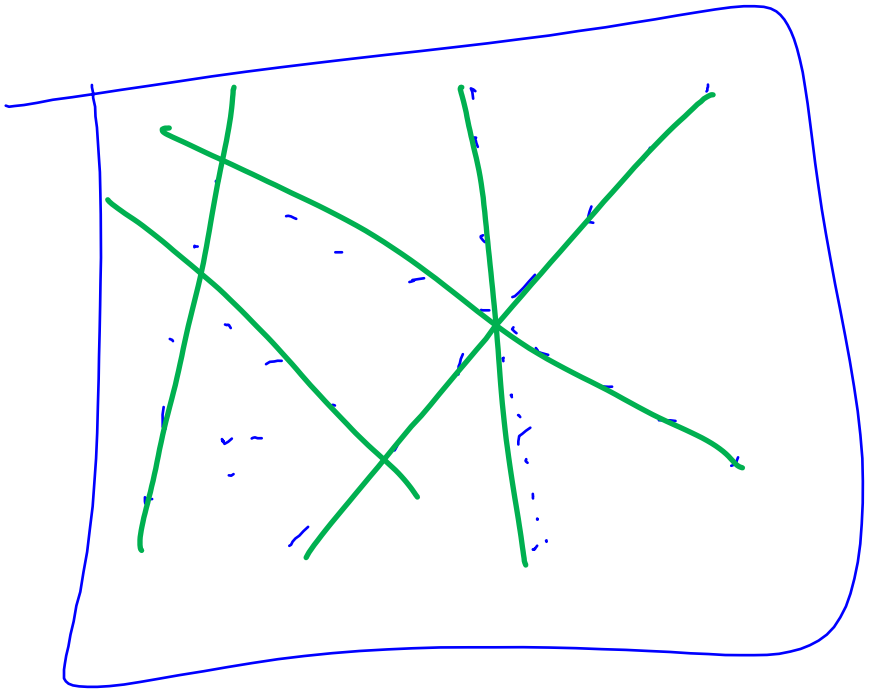
What is involved in line or curve detection?

- Grouping is deciding which image points belong to a target line or curve in the image.
- Model fitting: given a set of points that belong to a curve or line, find the best fit for the line or curve.

Line Detection

How do we detect lines on images?

Template matching…



Line Detection by Hough Transforms (HT)

- Instead of going directly for the solution (as in template matching), we can use a voting scheme.
- Each possible point (edge) votes in the parameter space of a line. A line can be specified uniquely by two parameters m and n.

$$y = m.x + n$$

Parameter space of 2 dimensions (m and n).

Hough Transforms

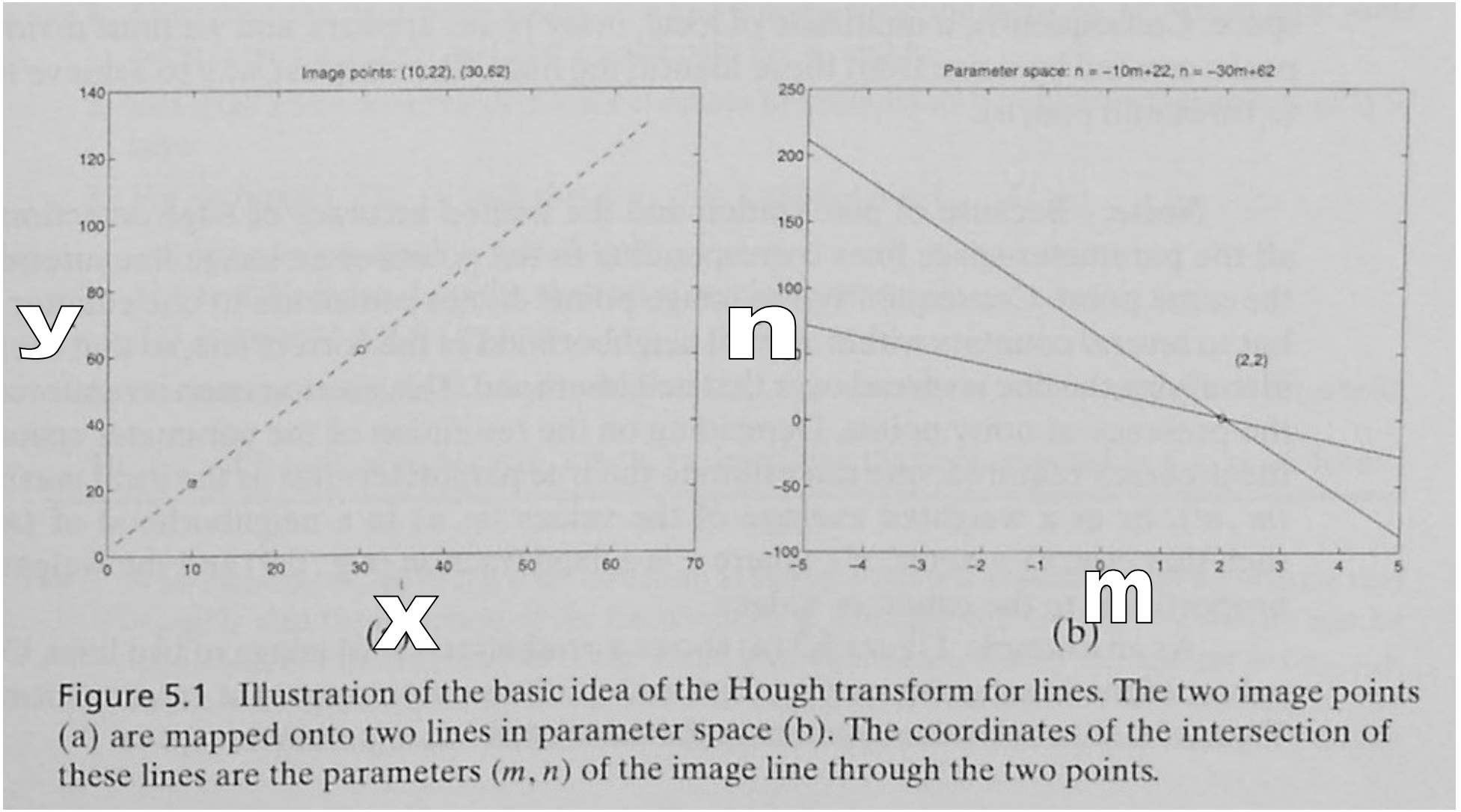
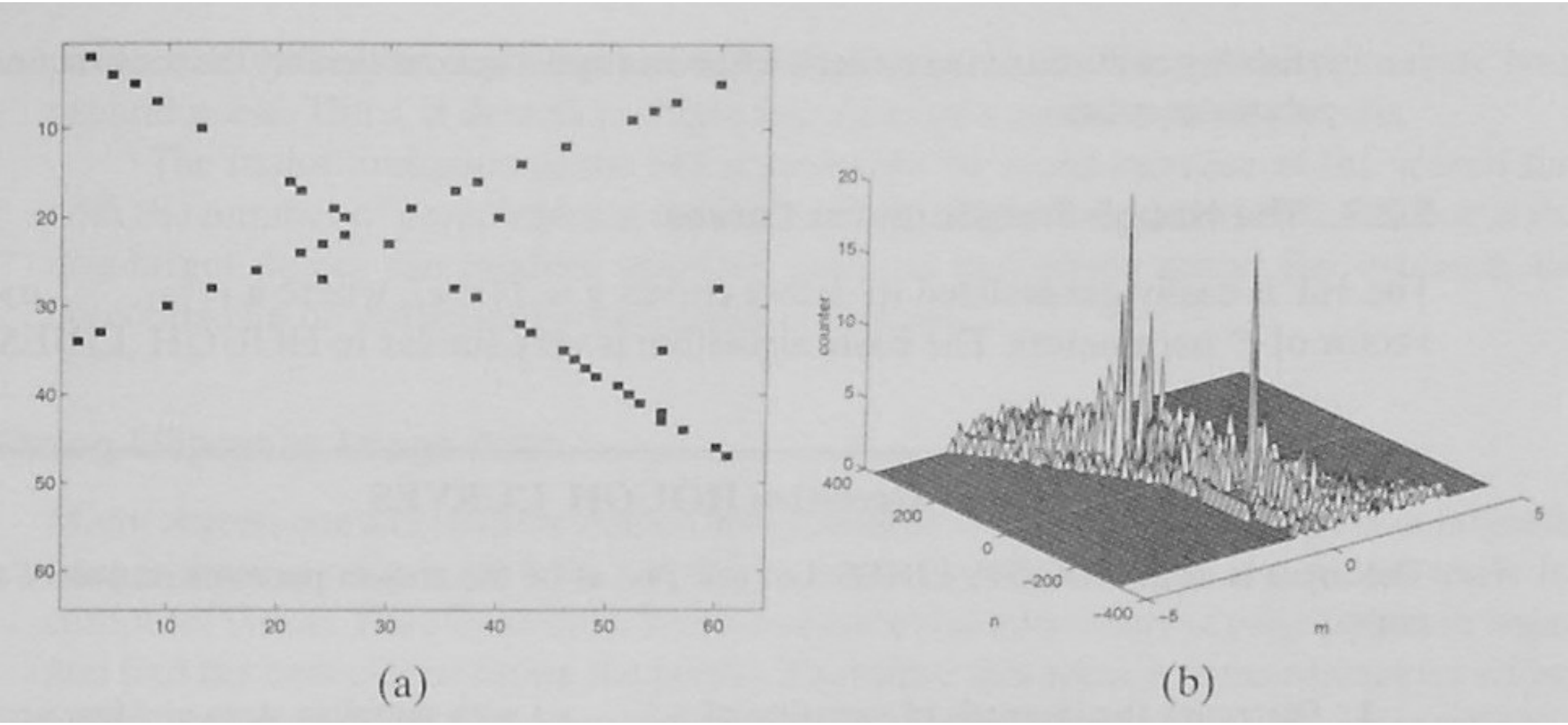


Figure 5.1 Illustration of the basic idea of the Hough transform for lines. The two image points (a) are mapped onto two lines in parameter space (b). The coordinates of the intersection of these lines are the parameters (m, n) of the image line through the two points.

$$y = m.x + n$$

Line Detection by HT



What problems did we solve with HT?

Line Detection by HT

Problem with m-n parameter space.

- Both m and n can get very large [-∞, ∞].
- What happens to m when x is constant?

How do we address this problem?

Line detection with HT

An alternative parameter space is polar coordinates. We use (ρ,θ) space to represent a line.

$\rho = x.\cos(\theta) + y.\sin(\theta)$

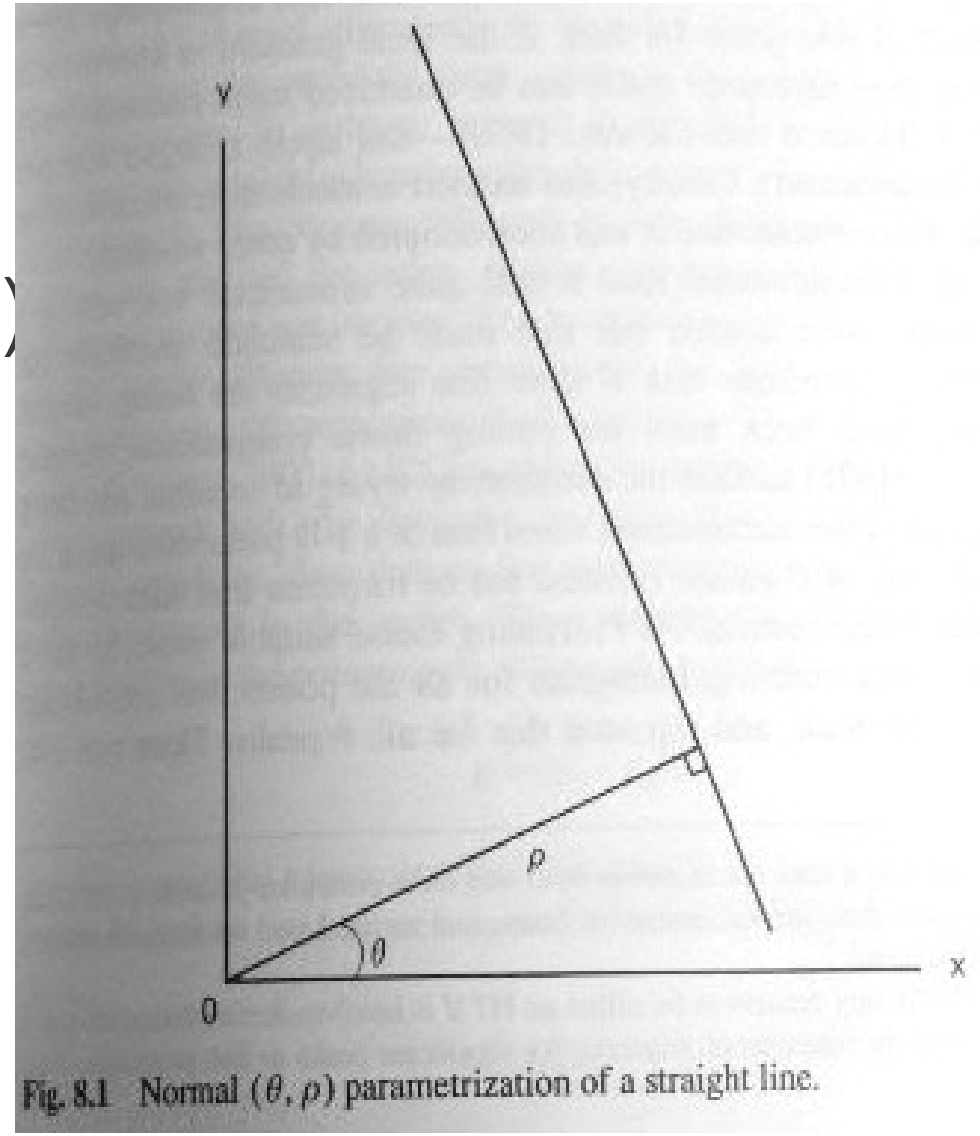


Fig. 8.1 Normal (θ, ρ) parametrization of a straight line.

Hough Line Detection Algo

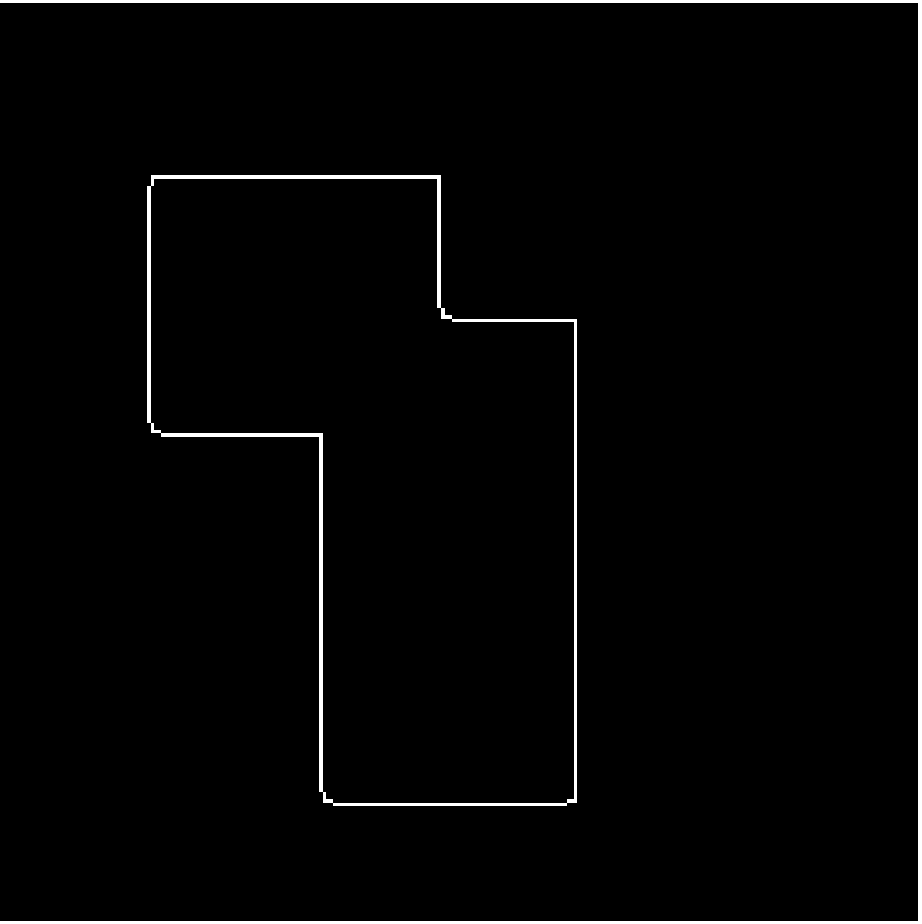
Algorithm HOUGH_LINES

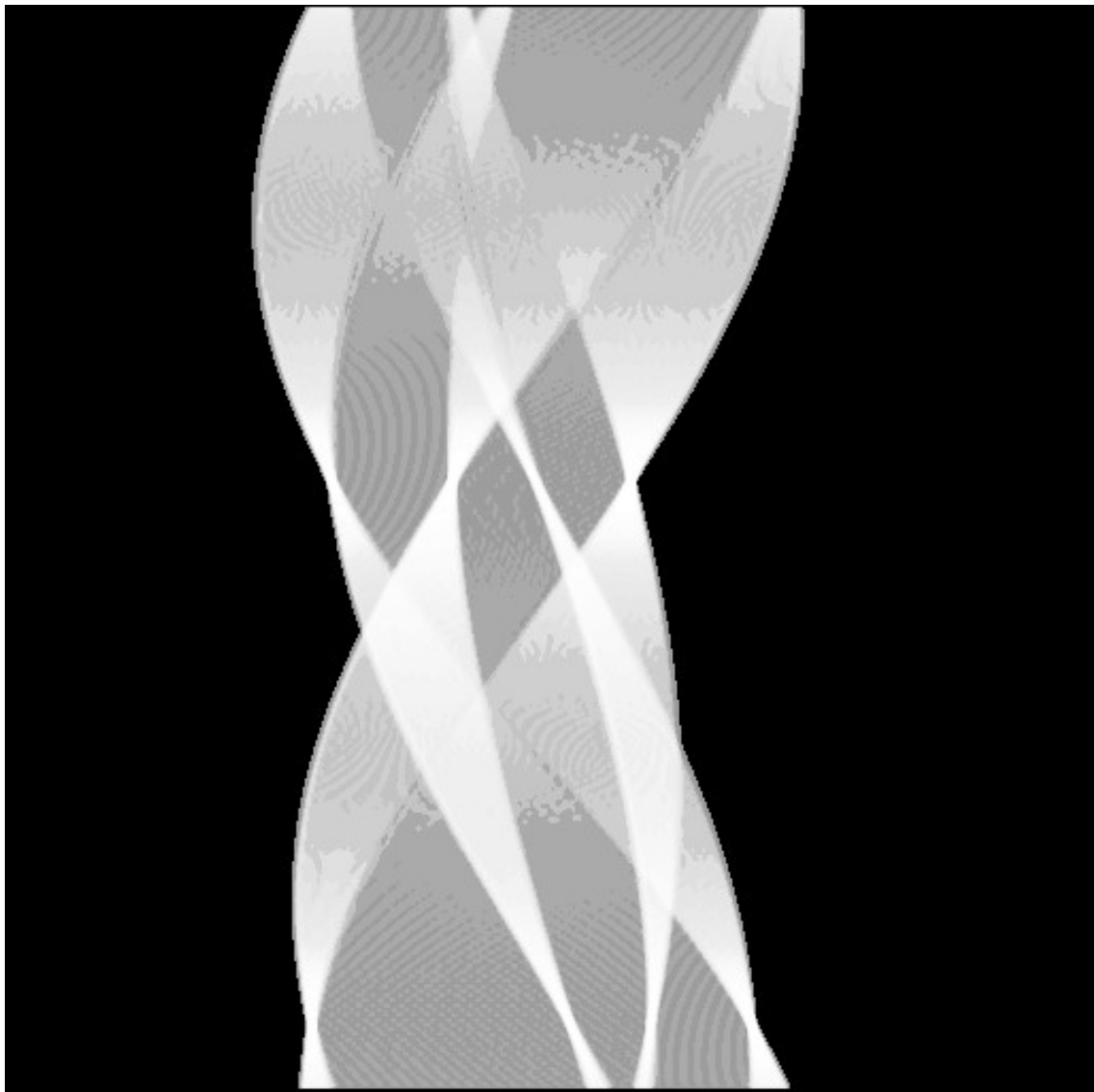
The input is E, a $M \times N$ binary image in which each pixel $E(i, j)$ is 1 if an edge pixel, 0 otherwise. Let ρ_d, θ_d be the arrays containing the discretized intervals of the ρ, θ parameter spaces ($\rho \in [0, \sqrt{M^2 + N^2}], \theta \in [0, \pi]$), and R, T, respectively, their number of elements.

1. Discretize the parameter spaces of ρ and θ using sampling steps $\delta\rho$, $\delta\theta$, which must yield acceptable resolution and manageable size for ρ_d and θ_d.
2. Let $A(R, T)$ be an array of integer counters (accumulators); initialize all elements of A to zero.
3. For each pixel, $E(i, j)$, such that $E(i, j) = 1$ and for $h = 1 \ldots T$:
   (a) let $\rho = i \cos \theta_d(h) + j \sin \theta_d(h)$;
   (b) find the index, k, of the element of ρ_d closest to ρ;
   (c) increment $A(k, h)$ by one.
4. Find all local maxima (k_p, h_p) such that $A(k_p, h_p) > \tau$, where τ is a user-defined threshold.

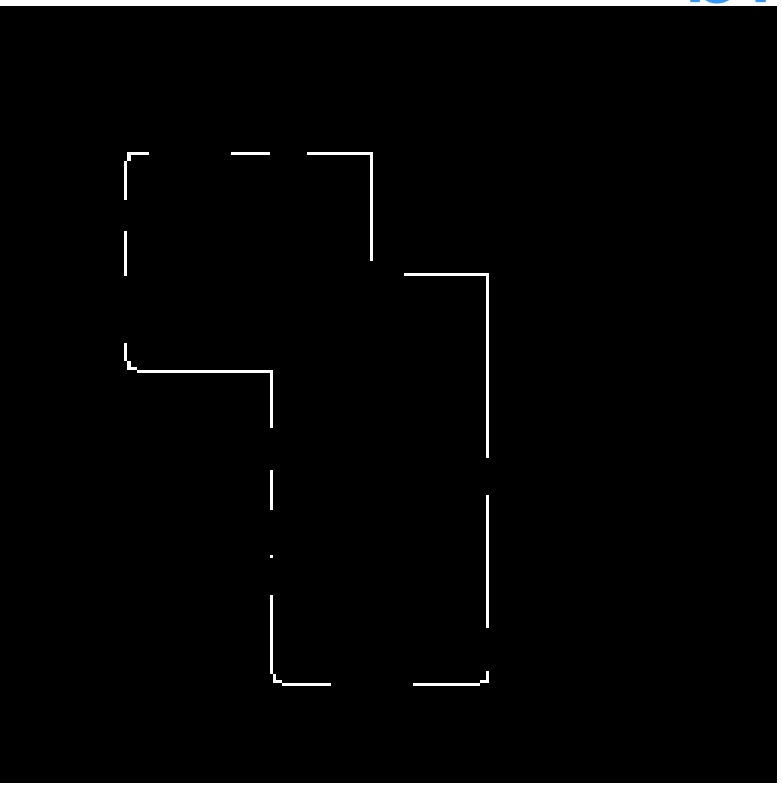
The output is a set of pairs $(\rho_d(k_p), \theta_d(h_p))$, describing the lines detected in E in polar form.

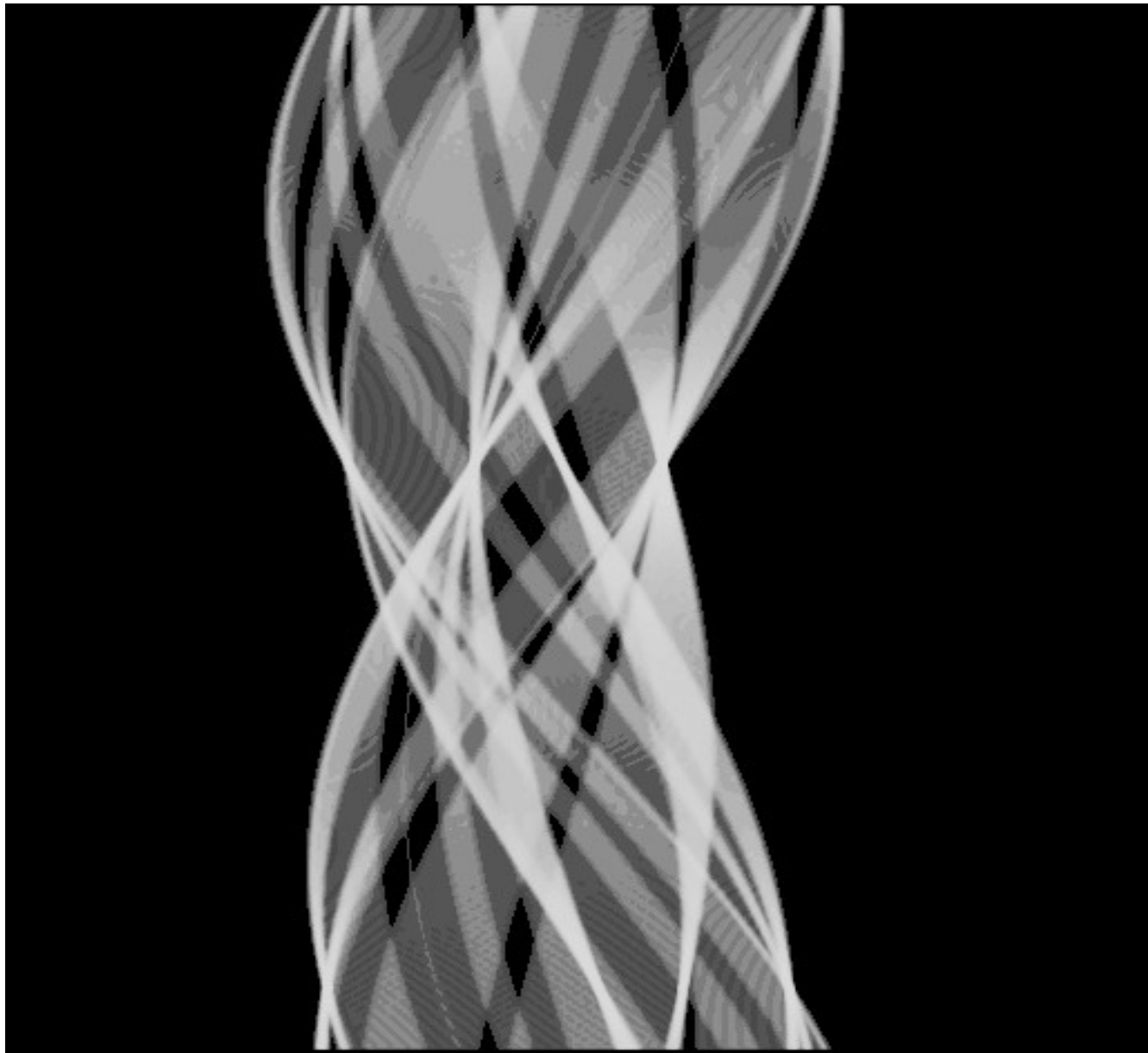
Hough Transform Example



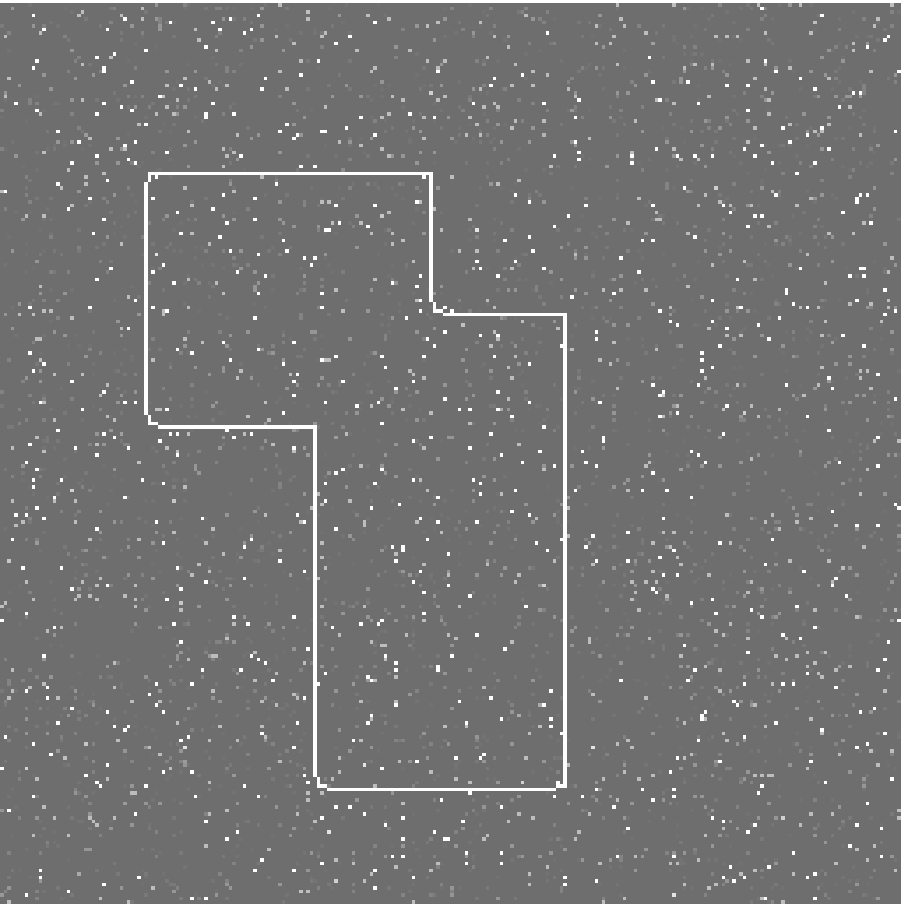


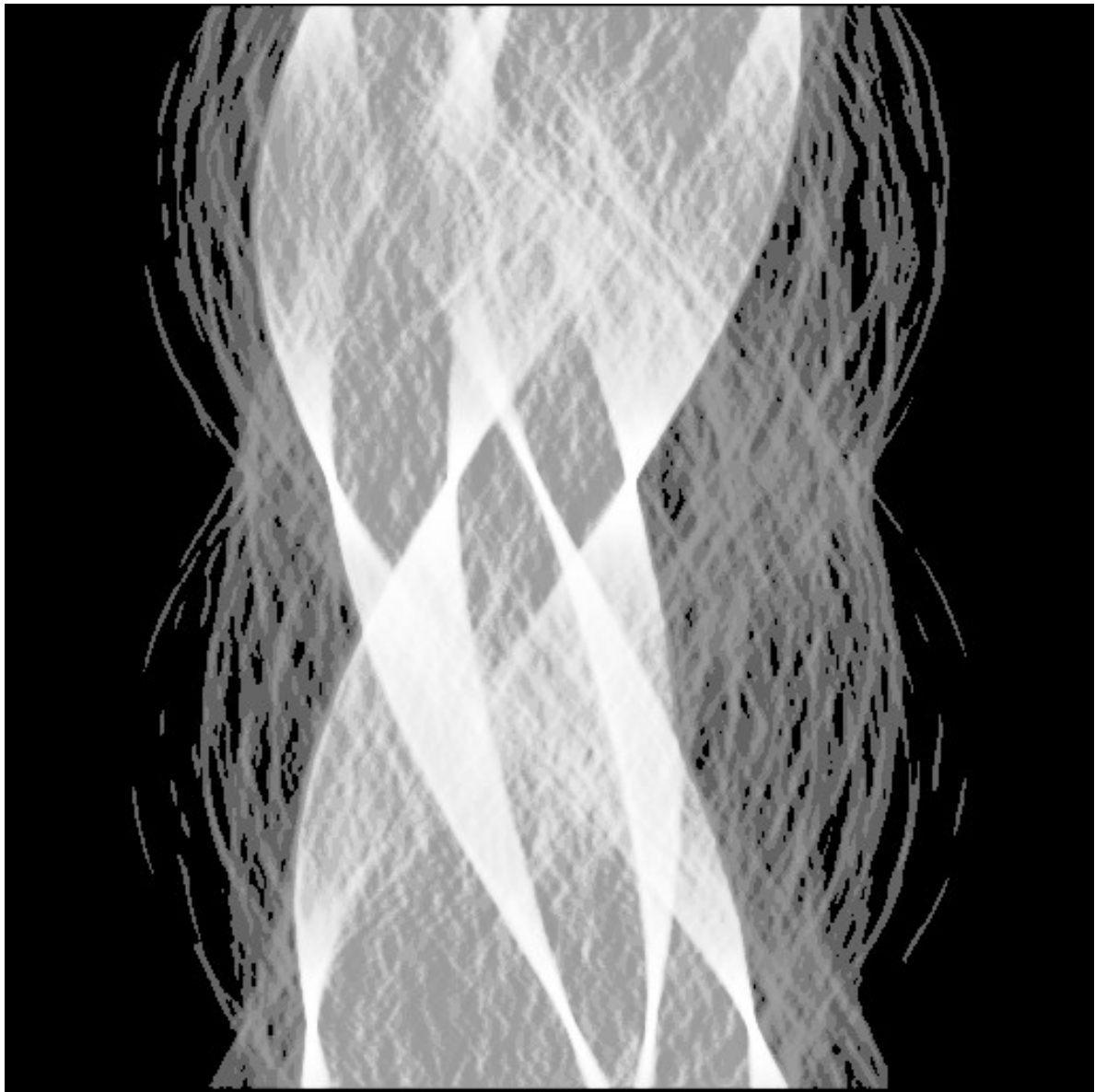
Hough Transform Example with breaks in lines





Hough Transform Example with 1% salt and pepper noise





Hough Line Detection

Problem with above algorithm?

It doesn’t use the angle information produced by the edge detector.

Gradient angle information can be used to make the space much less crowded.

Circle Detection With Hough Transforms

How do we use HT for circle detection?

Circle Detection With Hough Transforms

$$(x-a)^2 + (y-b)^2 = r^2$$

is the formula for a circle.

We can use the parametric version

$$x = a + r \cos \theta$$

$$y = b + r \sin \theta$$

When we eliminate r from the above formula, we would have

$$b = a \tan \theta - x \tan \theta + y$$

Circle Detection With Hough Transforms

$$b = a \tan \theta - x \tan \theta + y$$

1. Quantize the parameter space for parameters a and b
2. Zero the accumulator array M(a,b)
3. Compute the edges in the image and their angles.
4. For each edge point, increment all points in the accumulator array M(a,b) along the line

$$b = a \tan \theta - x \tan \theta + y$$

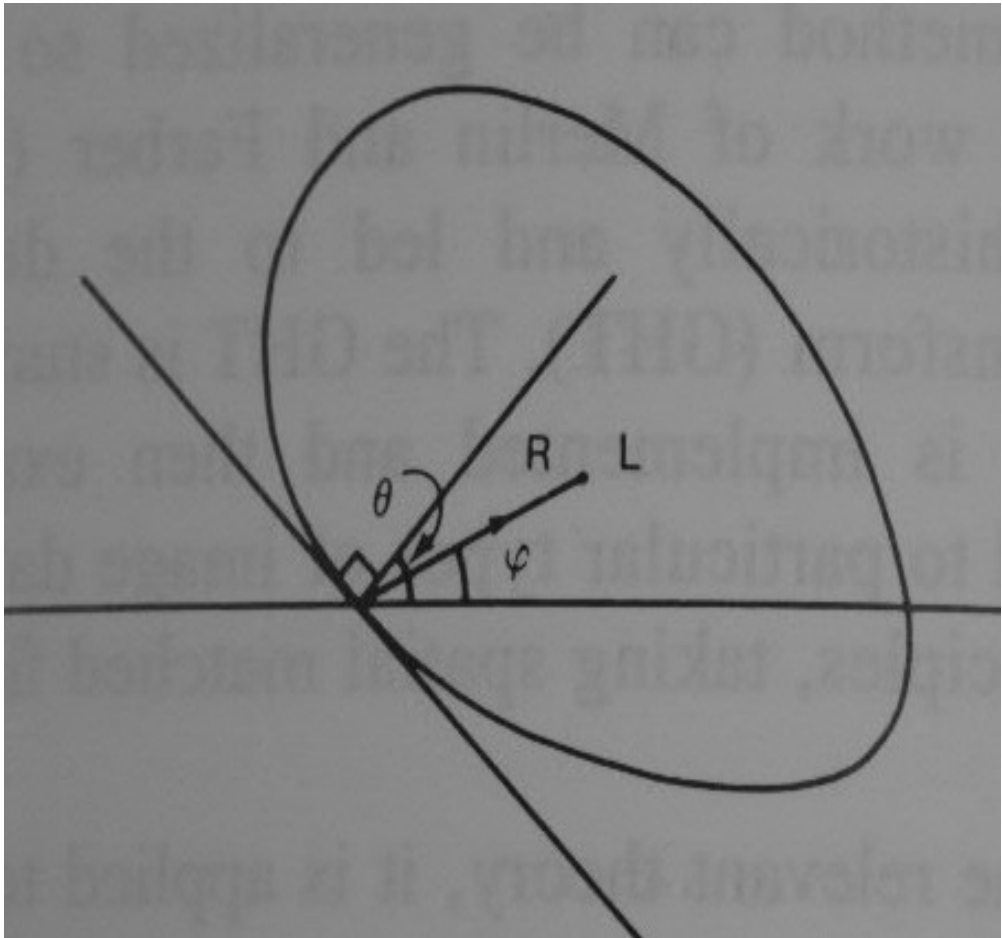
1. Local maxima in the accumulator array will give you the centers of the circles.

Generalized Hough Transform

We can use HT to localize arbitrary shaped objects.

On the object to be localized:

- Pick a reference point on the pattern object
- Compute the angles θ_i along the object.
- For each θ_i, store the distance R_i, angle φ_i from the reference point.



Generalized Hough Transform

In the search image

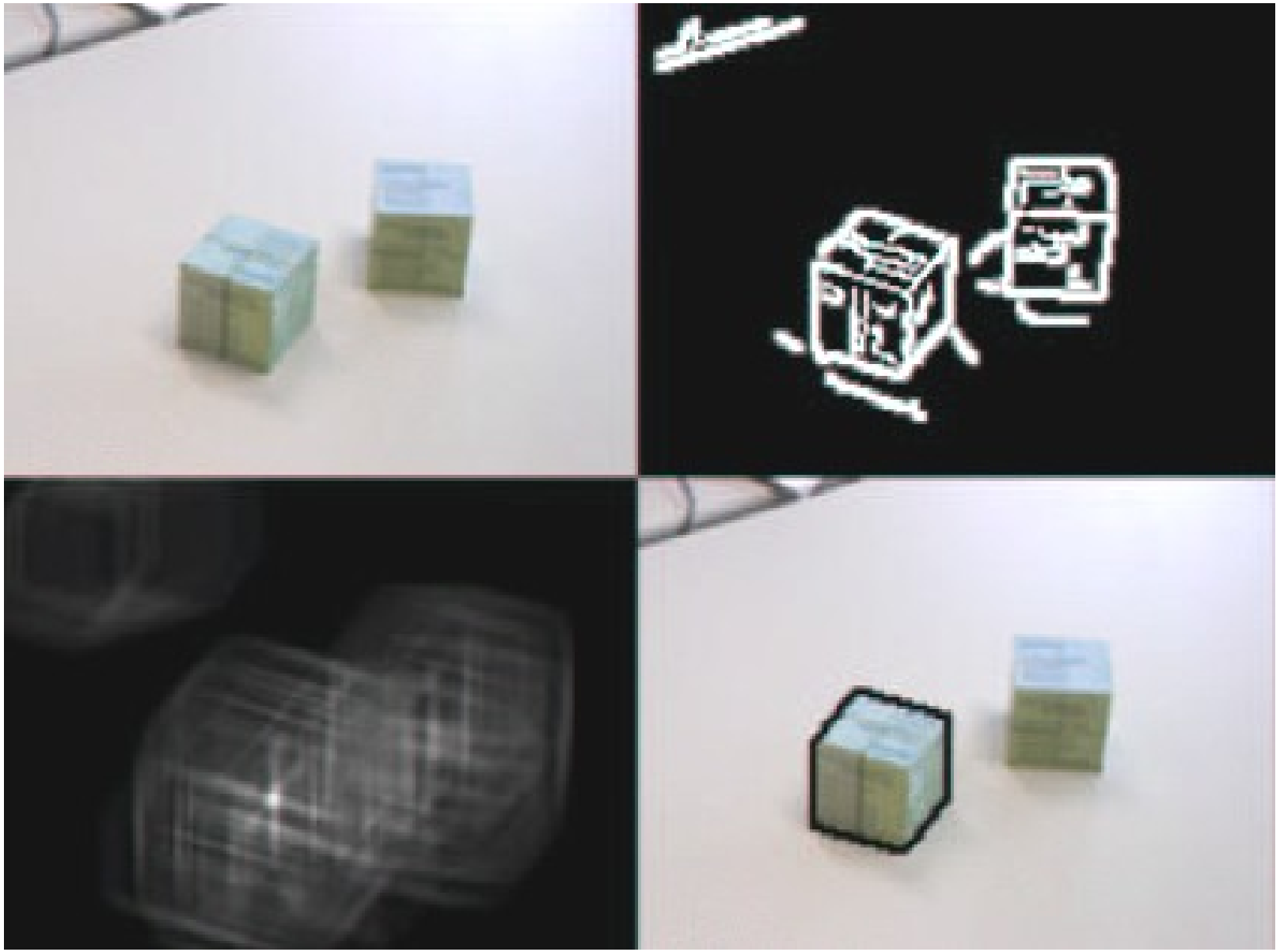
- For each edge point (x,y) with gradient angle θ, the possible locations of the reference point are given by

$$a = x - R(\theta)\cos(\varphi(\theta))$$

$$b = y - R(\theta)\sin(\varphi(\theta))$$

- The peaks on the voting space is the location of the reference point.

Generalized Hough Transform Example



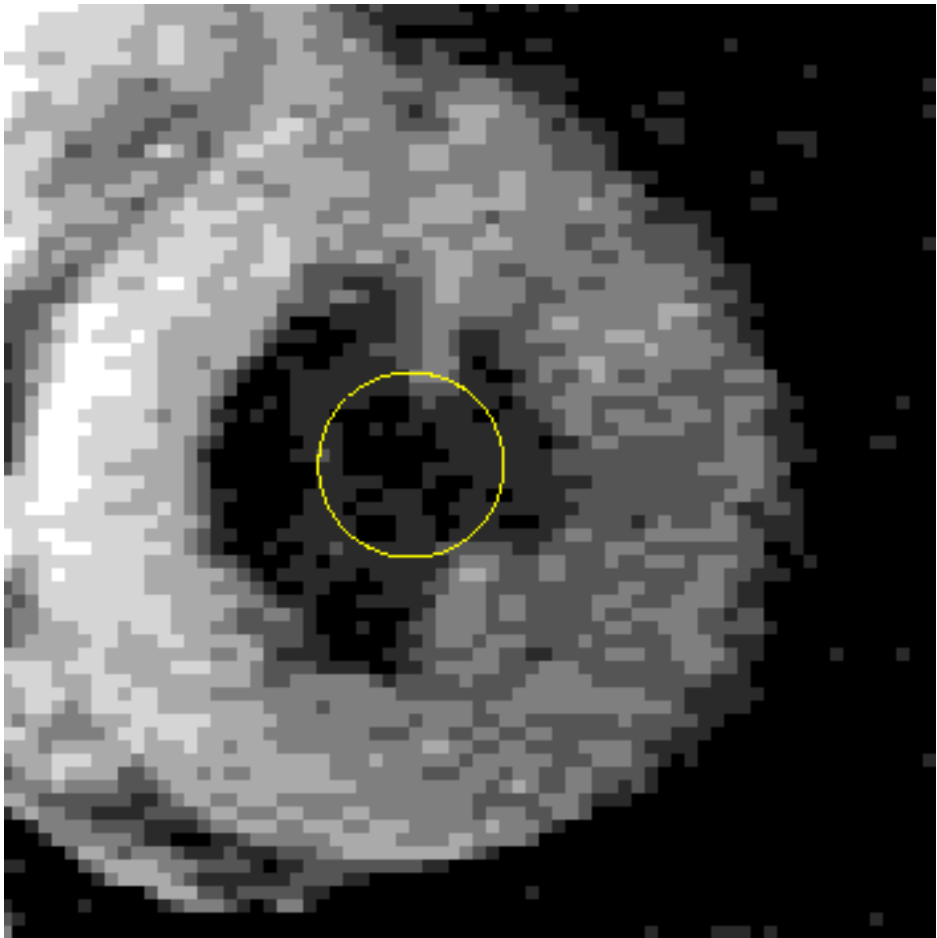
Problems with HT

- Exponential time and memory increase with the increase of the parameter numbers.
- Sampling of parameter space is difficult.
- Similar shapes might produce peaks in the Hough Space (wide ellipses might interfere with lines)
- Generalized HT is orientation and scale dependent.

Deformable Contours

A deformable contour (snake or active contour) is like an elastic band that moves on an image to localize an arbitrary object.

The snake starts from an initial position and deforms into the shape of the object that we like to localize.



Deformable Contours

Snakes are modeled by an energy functional.
Smaller values of energy functional moves the snake into more comfortable position and shape.

Comfortable position for a snake is

- A smooth and continuous snake shape
- The snake position is close to high gradient areas.

Deformable Contours

A snake is a ordered set of equality spaced points p1,.., pn on an image. Associated with a snake is the energy functional.

$$\sum_{i=1}^{N} \left(\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image}\right),$$

E*cont* is for continuity term

E*curv* is for curvature (smoothness) term

E*image* is for the image term.

Deformable Contours

Each energy term is evaluated at each snake element position p_i.

The snake element positions that produce the smallest snake energy value is the localized shape.

Alpha, beta, and gamma are weighting constants.

Deformable Contours

Econt (continuity term) makes the snake continuous like a rubber band.

$$E_{cont} = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^2.$$

If we like to keep a distance, d, between snake elements, then

$$E_{cont} = \left(\bar{d} - \|\mathbf{p}_i - \mathbf{p}_{i-1}\|\right)^2,$$

Deformable Contours

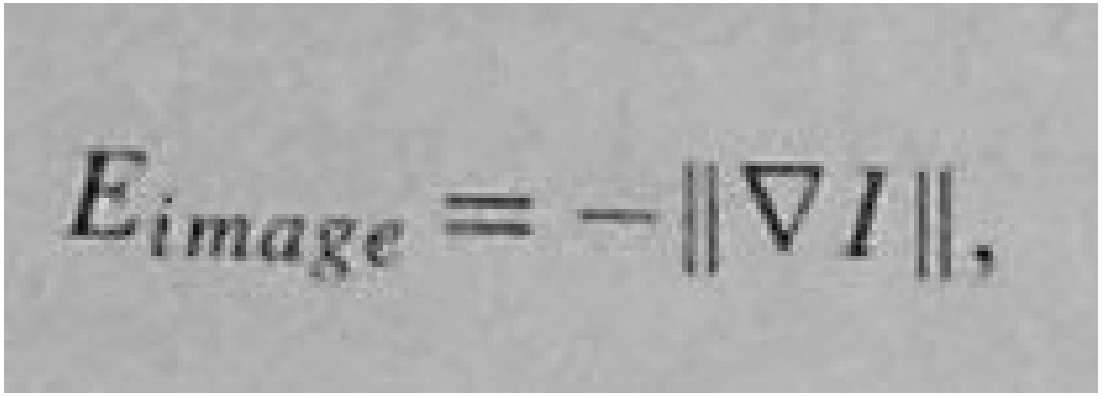
E*curv* (curvature or smoothness term) makes the snake straight or less curvy.

It is approximated by the second derivative of the contour.

$$E_{curv} = \|\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}\|^2.$$

Deformable Contours

E*image* (image terms) ties the snake to the image properties. We want our snake to pass through high gradient areas of the image where there are possible edges.



Deformable Contours

We know how to measure continuity, smoothness, and image term constraints on the snake.

How do we impose all the constraints at the same time?

Deformable Contours

There are many functional minimization methods.

One of the easiest is the ***greedy*** algorithm.

Basic idea:

1. For each snake element, p, find the position that produces smallest energy in a small neighborhood (5x5 or 7x7) around p.
2. Continue this process until all elements are done.

Deformable Contours

The greedy algorithm from the book

Step1 greedy minimization

For each snake element, p, move p to the position that minimizes the snake terms within a small region

Step 2 Corner elimination

Find snake element positions with very high curvature and set the beta value of that element to zero (why do we do this?)

Deformable Contours

The input is formed by an intensity image, I, which contains a closed contour of interest, and by a chain of image locations, $\mathbf{p}_1, \ldots, \mathbf{p}_N$, defining the initial position and shape of the snake.

Let f be the minimum fraction of snake points that must move in each iteration before convergence, and $U(\mathbf{p})$ a small neighborhood of point $\mathbf{p}$. In the beginning, $\mathbf{p}_i = \bar{\mathbf{p}}_i$ and $d = \bar{d}$ (used in E_{cont}).

While a fraction greater than f of the snake points move in an iteration:

1. for each $i = 1, \ldots, N$, find the location of $U(\mathbf{p}_i)$ for which the functional $\mathcal{E}$ defined in (5.10) is minimum, and move the snake point $\mathbf{p}_i$ to that location;
2. for each $i = 1, \ldots, N$, estimate the curvature k of the snake at $\mathbf{p}_i$ as

$$k = |\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}|,$$

   and look for local maxima. Set $\beta_j = 0$ for all $\mathbf{p}_j$ at which the curvature has a local maximum and exceeds a user-defined minimum value;
3. update the value of the average distance, $\bar{d}$.

On output this algorithm returns a chain of points $\mathbf{p}_i$ that represent a deformable contour.

Deformable Contours

What are the problems with snakes?

- How do we choose alpha, beta and gamma?
- When do we stop moving the snake?
- What if there are more than one contour in the image?

$$\sum_{i=1}^{N} \left(\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image} \right) .$$