Edge Segmentation

Goal: Connect edges to produce object contours

Approaches:

- thresholding with hysteresis
- edge relaxation
- edge linking
- graph searching
- fitting

Hough Transform

Basic Idea (line finding version):

- Find all features of interest (usually edge points)
- Let each one "vote" for each of the lines through it
- Use an *accumulator* to count the votes from all feature points
- Most votes wins

Requires:

- known shape
- parametric description shape
- edges

Advantages:

- can handle disconnected edges
- does not require all of the shape
- very robust to noise

Parametric Shape Representation

Suppose that what you're looking for is a straight line in the image.

Parametric representation of a line:

$$y = mx + b$$

(your book uses $kx + q$)

Any combination of m and b values defines a line through (x, y).

OR

The set of all lines through (x, y) is the space of all possible (m, b)

Parametric spaces

A point in (x, y) defines a line in (m, b) space.

A point in (m, b) defines a line in (x, y) space.

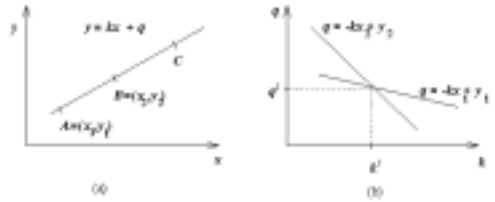


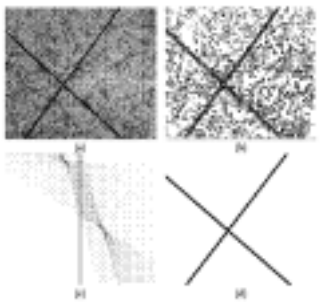
Figure 5.20 *Hough transform principles: (a) Image space, (b) k, q parameter space.*

The Hough Algorithm

The algorithm for the Hough transform can be expressed as follows:

1. Find all of the desired feature points in the image.
2. For each feature point $\overline{x}_i$
3. For each possibility $\overline{p}_j$ in the accumulator that passes through the feature point
4. Increment that position in the accumulator
5. Find local maxima in the accumulator.
6. If desired, map each maximum in the accumulator back to image space.

Example: Lines



(a) image, (b) feature points, (c) accumulator, (d) result

A Better Way of Expressing Lines

The slope-intercept form of a line has a problem with vertical lines.

Another way of expressing a line is in (ρ, θ) form:

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

Each feature point maps to a sinusoid in the parameter space.

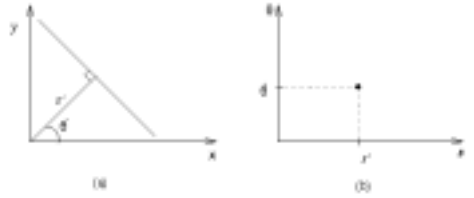


Figure 5.52 Hough transform in x, θ space: (a) Straight line in image space, (b) x, θ parameter space.

Circles

Parametric representation of a circle:

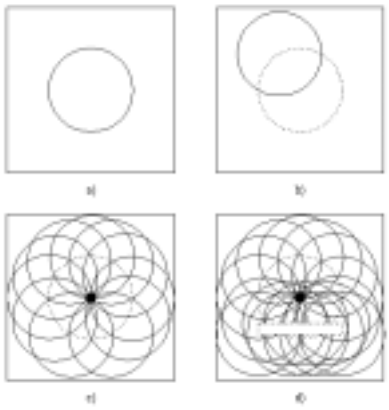
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

Three parameters: center (x_c, y_c) and radius r

Only two parameters if size r is known

Same idea: each feature point votes for each circle it could be on

Example: Circles



More Complicated Shapes

The Hough Transform can be extended to any parametrically-defined shape.

Problem: as the number of parameters increases, so does the dimensionality of the parameter space.

Computational complexity goes up exponentially with the number of parameters.

Implementation Details and Variations

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Using gradient orientation
- Weighting by gradient magnitude

Comparison to Template Matching

Template Matching (Correlation):
For an $N \times N$ image and an $M \times M$ template, complexity is $O(N^2 M^2)$.

Hough Transform:
Uses only edge points, which are far fewer than the number of pixels in the image.
Uses only points on the contour of the target, not all points in the shape.
Edge points increase linearly with N, not N^2.
Target contour points increase linearly with M, not M^2.
Basically $O(NM)$.

Generalized Hough Transform

Can get around problem of complex shapes (large number of parameters) by representing the shape as a table of relative edge positions.

Choose a reference point r for the *known* shape and store relative offsets for each point on the shape.

1. Find all of the desired feature points in the image.
2. For each feature point
3. For each pixel i on the target's boundary
4. Get the relative position of the reference point from i
5. Add this offset to the position of i
6. Increment that position in the accumulator
7. Find local maxima in the accumulator.
8. If desired, map each maxima in the accumulator back to image space using the target boundary table

R-Tables

For efficiency, don't store Cartesian offsets (x, y)—store them in polar form with angle relative to the tangent.

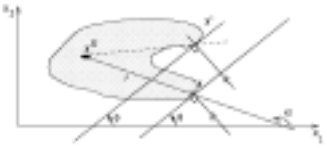


Figure 5.33 *Principles of the generalized Hough transform. Geometry of R-table construction.*

$$\theta_1 : (r_1^1, \alpha_1^1), (r_1^2, \alpha_1^2), \ldots$$
$$\theta_2 : (r_2^1, \alpha_2^1), (r_2^2, \alpha_2^2), \ldots$$
$$\theta_3 : (r_3^1, \alpha_3^1), (r_3^2, \alpha_3^2), \ldots$$
$$\vdots$$

Refining the Accumulator

Problem:
feature points vote for *all* possible shapes they could be on—lots of "crosstalk" (spurious local maxima)

Solution (Gerig 1987):

- Vote once using normal Hough transform
- For each feature point, look at all the accumulator positions that it voted for and find the maximum
- Perform a second round of voting with each feature point *casting only one vote* (for the maximum vote-getter in the first round).