The Hough Transform: Lines, Circles, and Other Primitives

CS 450: Introduction to Digital Signal and Image Processing

Detecting Primitives

- How do you find entire shapes instead of just points? (Lines, circles, etc.)
- Two basic approaches:
  - Bottom up — link and identify
  - Top down — fitting methods
    - Hough transform
    - RANSAC (come back later)

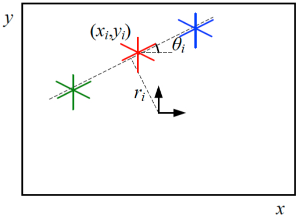
Detecting Primitives

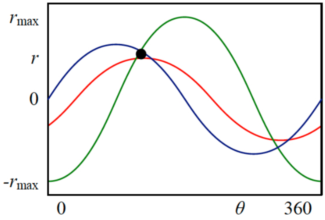
- How do you find entire shapes instead of just points? (Lines, circles, etc.)
- Two basic approaches:
  - Bottom up — link and identify
  - Top down — fitting methods
    - Hough transform
    - RANSAC (come back later)

Hough Transform - A Fitting Approach

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 features
- Most votes wins





Hough Transform - A Fitting Approach

Requires:

- known shape
- parametric description of the shape
- edges or other feature points to fit

Advantages:

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

Hough Transform - A Fitting Approach

Requires:

- known shape
- parametric description of the shape
- edges or other feature points to fit

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 (The basic idea can be extended to other shapes as well)
- Parametric representation of a line:

$$y = ax + b$$

- Key idea:

For a fixed (a, b) the set of satisfying points (x, y) define a line
OR
The set of lines through (x, y) is the space of all satisfying (a, b)

Parametric Shape Representation

- Suppose that what you're looking for is a straight line (The basic idea can be extended to other shapes as well)
- Parametric representation of a line:

$$y = ax + b$$

- Key idea:

For a fixed (a, b) the set of satisfying points (x, y) define a line
OR
The set of lines through (x, y) is the space of all satisfying (a, b)

Parametric Shape Representation

- Suppose that what you're looking for is a straight line (The basic idea can be extended to other shapes as well)
- Parametric representation of a line:

$$y = ax + b$$

- Key idea:

For a fixed (a, b) the set of satisfying points (x, y) define a line

OR

The set of lines through (x, y) is the space of all satisfying (a, b)

Parametric Shape Representation

- Suppose that what you're looking for is a straight line (The basic idea can be extended to other shapes as well)
- Parametric representation of a line:

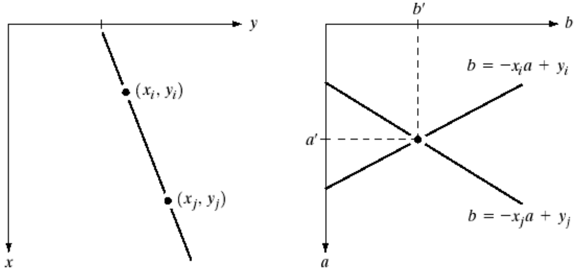
$$y = ax + b$$

- Key idea:

For a fixed (a, b) **the set of satisfying points** (x, y) **define a line**
OR
The set of lines through (x, y) **is the space of all satisfying** (a, b)

Parametric Spaces

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



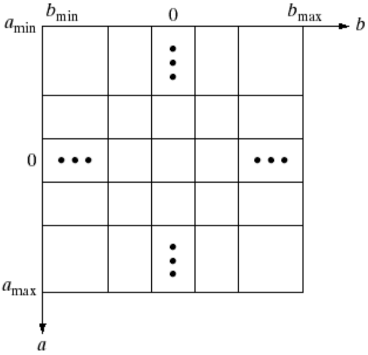
a b

FIGURE 10.17 (a) xy-plane. (b) Parameter space.

Parametric Spaces - Discretizing

- Obviously can't vote for *all* possible lines, so discretize the space

FIGURE 10.18 Subdivision of the parameter plane for use in the Hough transform.



The Hough Algorithm

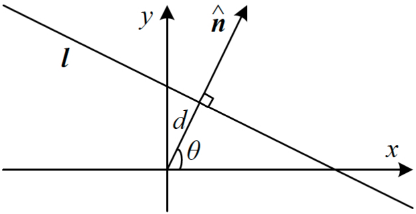
The algorithm for the Hough transform can be expressed as follows:

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

A Better Way of Expressing Lines

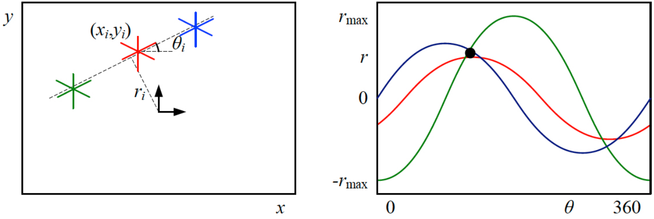
- The slope-intercept form has a problem with vertical lines
- Another way of expressing a line is in polar (d, θ) form:

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



A Better Way of Expressing Lines

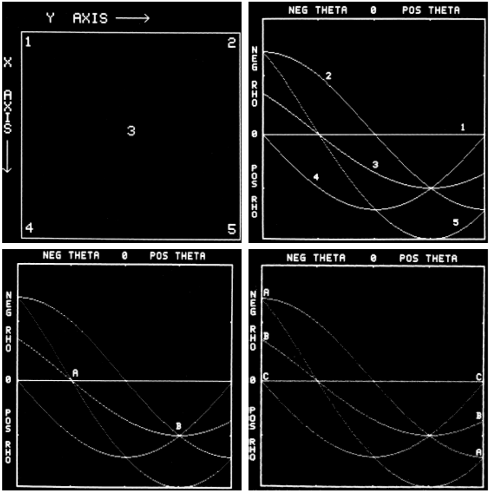
- Each feature point maps to a *sinusoid* in the parameter space



Parameter Space - Alternate Parameterization of Lines

a b
c d

FIGURE 10.20 Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)



Example: More Lines

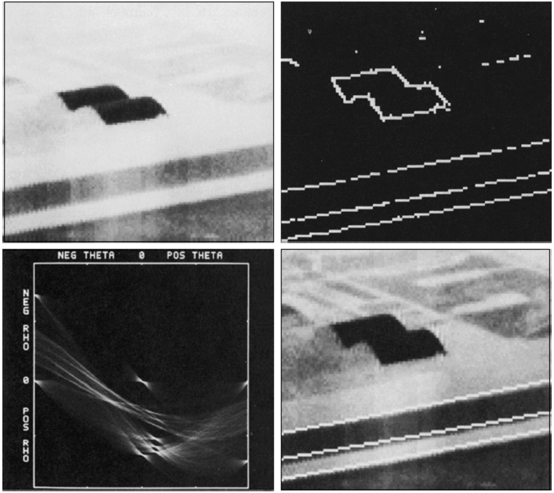


FIGURE 10.21 (a) Infrared image. (b) Thresholded gradient image. (c) Hough transform. (d) Linked pixels. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

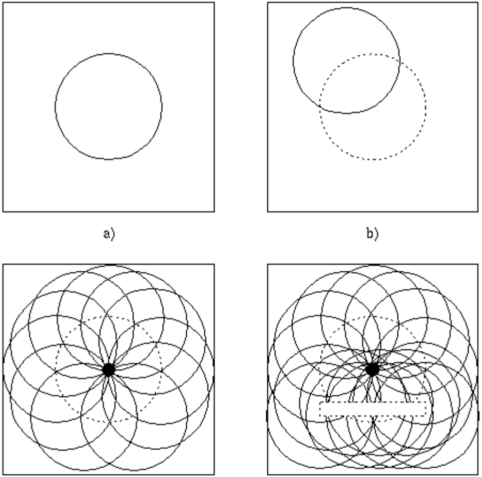
Circles

- 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



a) b) c) d)

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*

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*

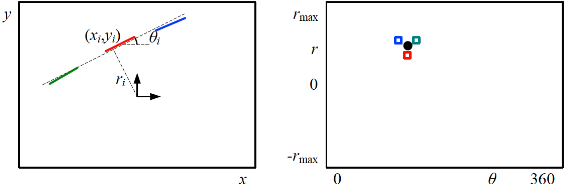
More Complicated Shapes

- 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

Variations between implementations:

- Using gradient orientation
- Weighting by gradient magnitude



Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Implementation Details and Variations

Other variations between implementations:

- Orientation - parameter or known?
- Discrete accumulator
- Smoothing the accumulator
- Finding maxima
- Other postprocessing

Generalized Hough Transform

- Can get around problem of complex shapes 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.
- Algorithm:
  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.
- Use the gradient perpendicular to estimate the tangent direction, thus limiting which template points the edge point could be.
- May be multiple points with same tangent direction, so store a list for each tangent angle.

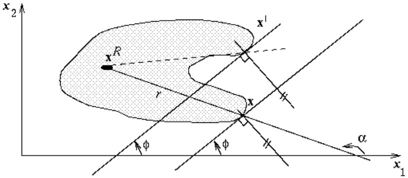


Figure 5.35 *Principles of the generalized Hough transform: Geometry of R-table construction.*

$$\theta_1 : (r_1^1, \alpha_1^1), (r_1^2, \alpha_1^2), \ldots$$

$$\vdots$$

R-Tables (variation)

- Can use the same R-table to handle rotations
- Don't restrict voting only to points with matching tangent angles–vote as if all points
- *But vote relative to the tangent angle (and maybe a little around it just in case)*

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)
- Variation (Morse 1998):
  - Like Gerig’s method, vote once using normal Hough transform
  - For each feature point, likewise look at all the accumulator positions that it voted for *and add them up*
  - On the next pass of voting (into a new accumulator), weight votes by value of old accumulator divided by sum of accumulator positions voted for
  - Repeat until convergence or sufficient solution

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)
- Variation (Morse 1998):
  - Like Gerig's method, vote once using normal Hough transform
  - For each feature point, likewise look at all the accumulator positions that it voted for *and add them up*
  - On the next pass of voting (into a new accumulator), weight votes by value of old accumulator divided by sum of accumulator positions voted for
  - Repeat until convergence or sufficient solution

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)
- Variation (Morse 1998):
  - Like Gerig's method, vote once using normal Hough transform
  - For each feature point, likewise look at all the accumulator positions that it voted for *and add them up*
  - On the next pass of voting (into a new accumulator), weight votes by value of old accumulator divided by sum of accumulator positions voted for
  - Repeat until convergence or sufficient solution

Multiresolution Approaches

- Can greatly reduce the computational complexity by
  - reducing the size of the image
  - reducing the size of the accumulator
- Idea:
  - First find potential candidate shapes using low-res image and low-res accumulator
  - Repeat using high-res accumulator and high-res image *but only use points/lines near the candidates identified from the initial low-res pass*

Multiresolution Approaches

- Can greatly reduce the computational complexity by
  - reducing the size of the image
  - reducing the size of the accumulator
- Idea:
  - First find potential candidate shapes using low-res image and low-res accumulator
  - Repeat using high-res accumulator and high-res image *but only use points/lines near the candidates identified from the initial low-res pass*