Lecture 9: Object Description and Location – Hough Transform

Harvey Rhody
Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology
rhody@cis.rit.edu

October 7, 2004

Abstract

Geometric objects are points, lines, squares, triangles, circles, cubes, spheres and so on. They have mathematical descriptions and properties such as length, area and volume. The Hough transform is a technique that enables such structures to be built by parametric search among lower-level information primitives.

Description of Objects

Suppose you have to write a program to "find the rectangles" in an image. What issues does this bring up?

- How do you describe a rectangle in a computer program?
- How do you relate the description to the image data structure?
- How do you carry out a search?
- What are the efficiency considerations?
- How do you measure performance?
- ...

Geometric Objects

Geometric objects are points, lines, squares, triangles, circles, cubes, spheres and so on. They have mathematical descriptions and properties such as length, area and volume.

We are interested in "finding" geometric objects in images because they can be useful in understanding the location, orientation, shape and other properties of real objects.

Edge detection identifies pixels that are located on edges. A collection of edge pixels is not an object. A method to find an object in a set of pixels is needed.

Geometric Descriptions

A number of common geometric objects are useful in describing components of visual objects.

- Straight lines
- Triangles
- Rectangles
- Polygons
- Circles
- Ellipses
- ...

Each geometric object has a mathematical description.

A line is equivalent to a pair of points $\{(x_0, y_0), (x_1, y_1)\}$.

A triangle can be described by its three corners. A circle, by its center and radius. Etc.

Two Problems

Two kinds of problems arise in relating geometric objects and image data structures.

1. Render an object with a given mathematical description into an image.

2. Find objects of certain mathematical types in the data that represents a real scene.

These two tasks are related but distinct. We are concerned with the second problem.

Pixels, Patterns from Points

There is a natural relationship between mathematical points and pixels, since a pixel with coordinates (u, v) can be regarded as the mathematical point (u, v).

Sets of pixels in an image that have certain desired properties can be selected by operations like filtering that look for desired patterns in the image pixels.

The sets of pixels can be searched for lines that have certain desired properties (length, orientation, grouping,...)

Lines can be grouped to form higher-level objects.

Line Search Strategies

Let $\mathcal{P} = \{p_0, p_1, \ldots, p_{n-1}\}$ be a set of n points.

Strategy 1: Find all of the lines that join pairs of dots. Select those lines that have the desired properties.

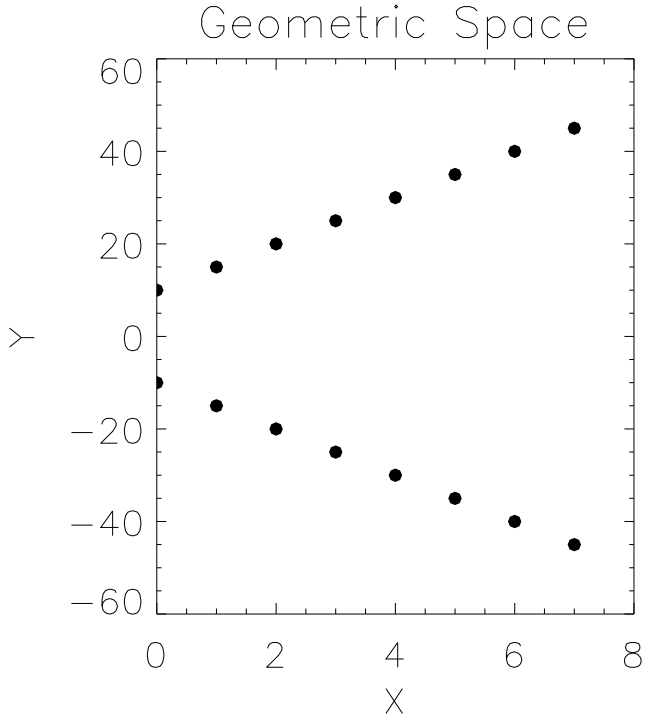
Strategy 2: Search in *parameter space* for lines that fit subsets of the points with certain specified properties.

Strategy 1 produces $n(n-1)/2 \sim n^2$ lines and then $n\,(n(n-1)/2) \sim n^3$ comparisons of every point to all the lines. This is impractical for all but the simplest problems.

The parameter-space approach is the basis of the *Hough Transform*

Lines and Edges

Shown below is an idealized representation of a set of pixels that have been found by edge detection. The task is to find straight line segments to fit the data.



Lines and Edges

A line is a geometric object that can be described by an equation. It is the collection of all points (x_i, y_i) that satisfy an equation such as $y_i = ax_i + b$.

The task is to find a pair of parameters (a, b) for each line that is "in" the data.

If we were given a set of points that fall on a (single) line then it would be a simple task to find the parameters.

How do we write an automatic algorithm that can search for an unknown number of lines in a real data set?

Hough Transform

The Hough Transform was been invented by Paul Hough in 1962 and patented by IBM. It has become a standard tool in the domain of computer vision for the recognition of straight lines, circles and ellipses. The Hough Transform is particularly robust to missing and contaminated data.

It can conduct a search in parameter space for any number of lines (or other geometric objects that have parametric descriptions).

Hough Strategy

The Hough strategy is to find parameter combinations that fit the data. The slope-intercept form of a straight line is

$$y_i = ax_i + b$$

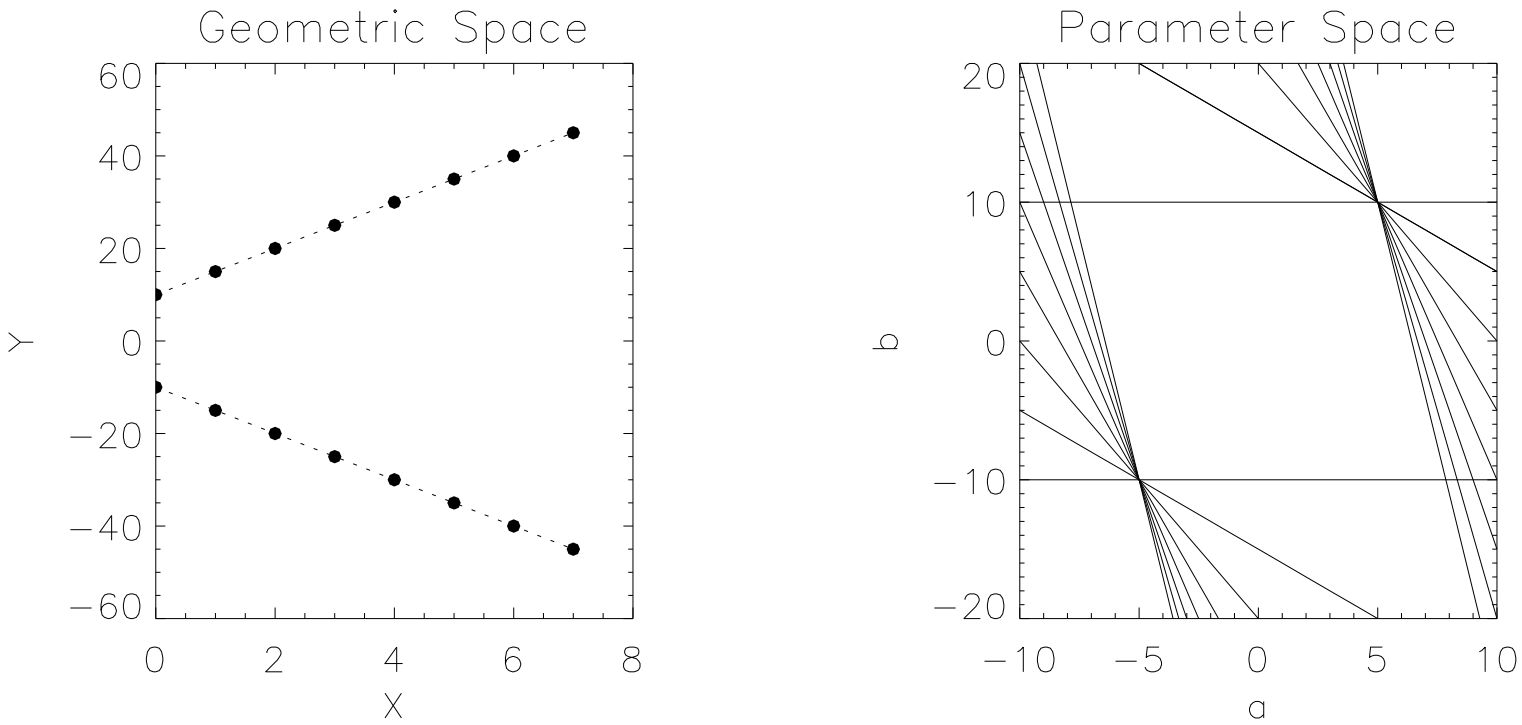
This can be inverted to express the parameter b in terms of a at each data point.

$$b = -ax_i + y_i$$

This represents a straight line in parameter space for each (x_i, y_i) pair.

Only one parameter combination satisfies all points on a given line. The parameter-space lines will cross at that (a, b) point.

Hough Strategy (continued)



Each point in geometric space produces a line in parameter space. Parameter pairs $(-5, -10)$ and $(5, 10)$ at the crossing points in parameter space define the geometric lines (shown dotted).

Example

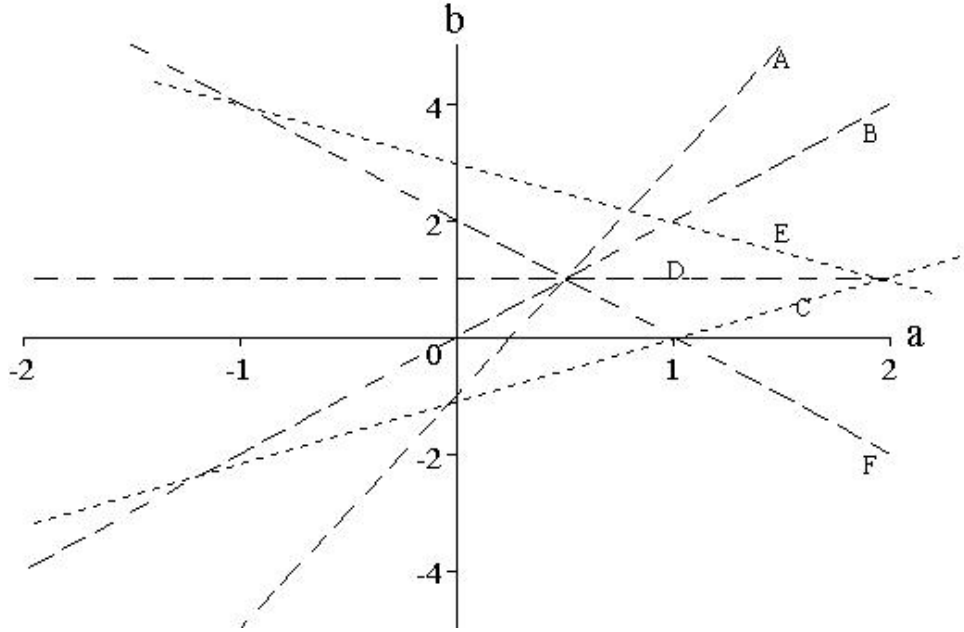
Find the straight line that passes through the maximum number of points from the set

$$\{(-4,-1),(-2,0),(-1,-1),(0,1),(1,3),(2,2)\}$$

We can use the parametric form $y = ax + b$ for a line. The six points provide six lines in parametric space.

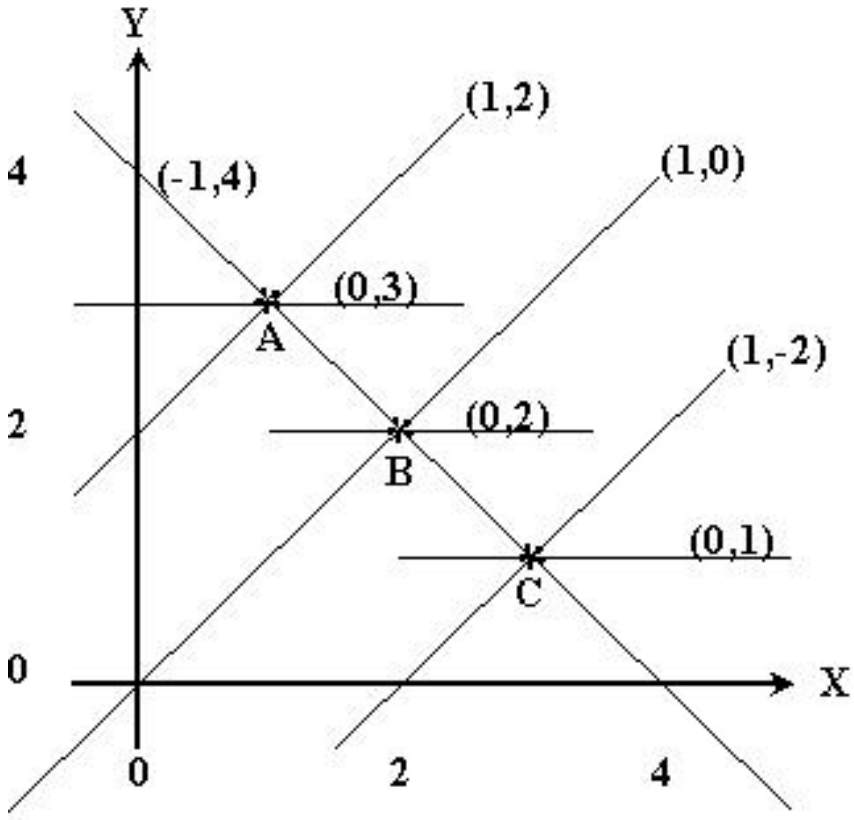
Example

A	$(-4,-1)$	$b = 4a - 1$
B	$(-2,0)$	$b = 2a$
C	$(-1,-1)$	$b = a - 1$
D	$(0,1)$	$b = 1$
E	$(1,3)$	$b = -a + 3$
F	$(2,2)$	$b = -2a + 2$



Example

- Each line in parameter space corresponds to a point in geometric space
- Each point in parameter space corresponds to a line in geometric space
- The point where k parameter space lines cross corresponds to a line through k points in geometric space
- A few examples are shown at the right



Algorithm Idea

The lines in parameter space can be drawn automatically.

Given a set of points from an edge finding algorithm, plot all parameter space lines and find the major intersections.

This will work with other equations. Let a line or curve be described by an equation of the form

$$f(a, b, x, y) = 0$$

If a set of points (x_n, y_n), $0 \leq n \leq N - 1$ is given then one can draw N curves in parameter space. Each intersection yields a parameter pair for a curve that passes through a subset of the points. The more lines in the intersection, the more points on the curve.

Implementation Approach

Divide parameter space into a set of discrete points, (a_i, b_j), These should reasonably cover the space.

Create a set of counters, one for each parameter pair, initialized to zero.

For each point (x_n, y_n)

For each a_i compute $b = -a_i x_n + y_n$

Find the point (a_i, b_j) closest to (a_i, b)

Increment counter for (a_i, b_j)

The counters with the most points correspond to the parameter pairs for desirable lines in geometric space.

Practical Issue

The primary practical issue is determining the location of M representative points in parameter space. If the range of a and b is bounded by a region of size $A \times B$ then it can be covered by a grid of points spaced by (Δ_a, Δ_b). The number of points is $AB/\Delta_a\Delta_b$.

A problem with the form $y = ax + b$ is that the slope is unbounded. Lines that are nearly vertical have very large values of a.

Solution: Use the *normal form* to represent the lines in geometric space.

Normal Form

The normal form of a straight line is

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

ρ is the shortest distance from the origin to the line and θ is the angle between the x-axis and the perpendicular from the origin to the line.

Both ρ and θ are bounded: $0 \leq \rho \leq R$, where R is the maximum distance of a pixel from the origin of the image and $0 \leq \theta \leq 2\pi$.

All of the straight lines that pass through the point (x_n, y_n) have parameters (ρ, θ) that satisfy

$$\rho = x_n\cos\theta + y_n\sin\theta$$

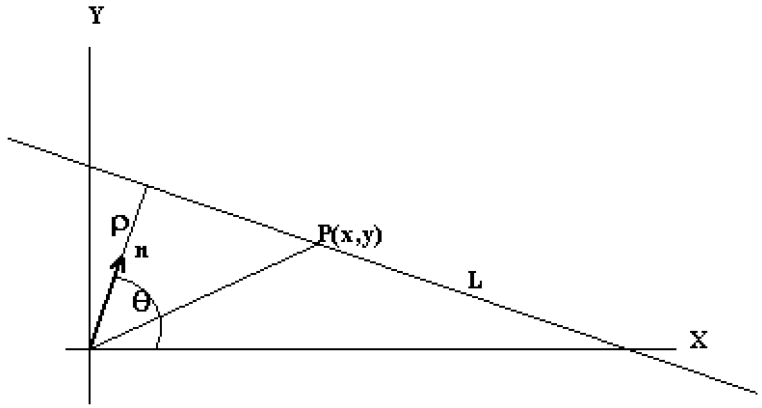
This is a curve in parameter space.

Normal Form

Normal representation of a line:

$$\mathbf{n} \cdot \mathbf{p} = \rho = x \cos\theta + y \sin\theta$$

Let L be a given line and let $\mathbf{n}$ be a unit vector along a line that is perpendicular to L and that passes through the origin. Let $\mathbf{n}$ have direction cosines $[\cos\theta, \sin\theta]$.
Any point $P(x, y)$ on the line can be described by a vector $\mathbf{p} = [x, y]$. This vector can be resolved into a component along $\mathbf{n}$ and another along L.
The component along $\mathbf{n}$ is



$$\mathbf{n} \cdot \mathbf{p} = \rho = x \cos\theta + y \sin\theta$$

Normal Form

The diagram on the next page (left) shows a set of points in geometric space. On the right a solid line is extended through the points and the perpendicular bisector through the origin is constructed (dashed). The heavy dark line extending from the origin to the line has length $\rho \approx 9.2$ and angle $\theta = 45°$.

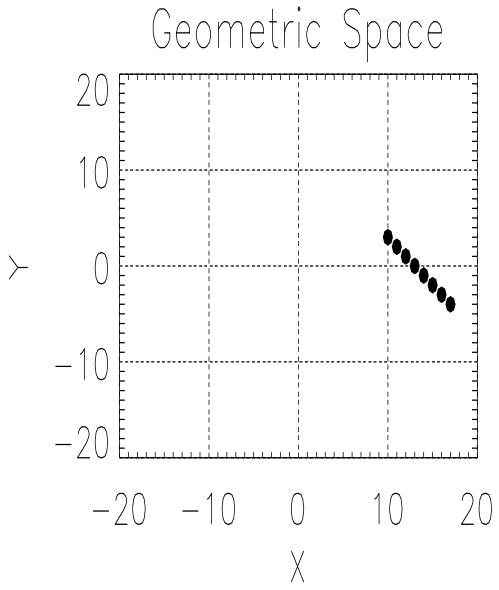
All of the points are on the line that is perpendicular to the vector

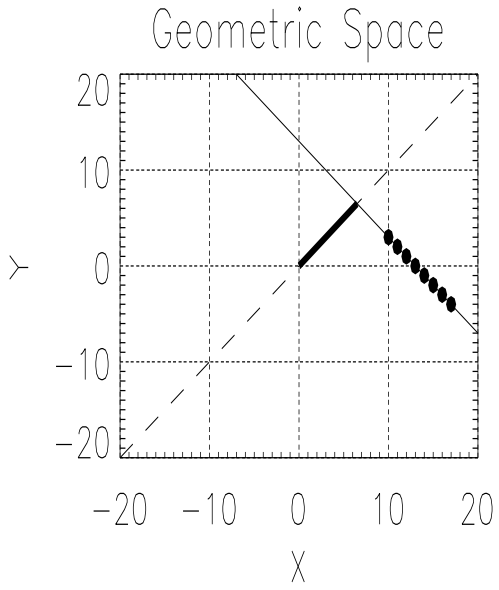
$$\mathbf{r} = \begin{cases} \rho\angle\theta & \text{polar form} \\ (\rho\cos\theta, \rho\sin\theta) & \text{rectangular form} \end{cases}$$

The normal vector $\mathbf{r}$ completely describes the line. The vectors $\mathbf{r}$ and $\mathbf{n}$ are related by $\mathbf{r} = |\mathbf{r}|\mathbf{n}$

The task is to discover all of the normal vectors from the edge point data.

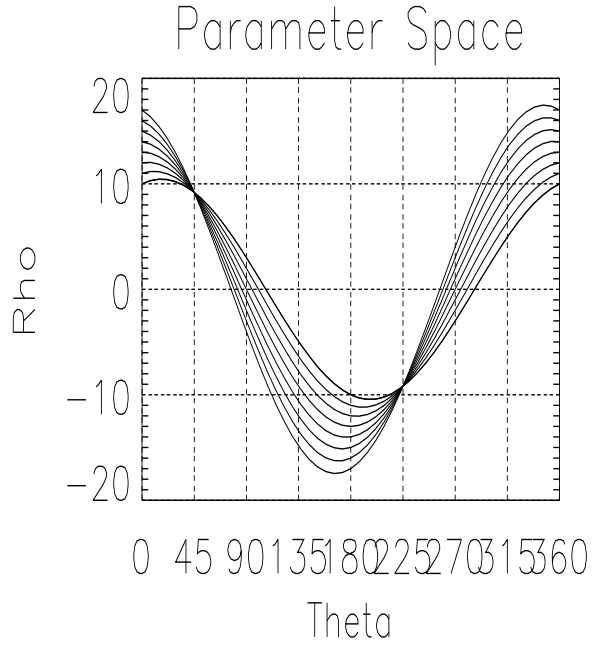
Normal Form


Geometric Space
20
10
0
−10
−20
Y
−20 −10 0 10 20
X


Geometric Space
20
10
0
−10
−20
Y
−20 −10 0 10 20
X

Normal Form

Each point on the preceding figure can be used to draw a curve in parameter space. One parametric curve for each geometric point.
Curves cross at $\mathbf{r} = \rho\angle\theta$ for the line through the corresponding points.



Normal Form

The crossings occur at angle θ and $\theta \pm 180°$, and ρ has opposite sign at the two angles. Note that $\mathbf{r} = [\rho\cos\theta, \rho\sin\theta] = [-\rho\cos(\theta \pm 180°), -\rho\sin(\theta \pm 180°)]$ so that either point can be used.

The search for the parameters of $\mathbf{r}$ $\mathbf{r}$ can be conducted in parameter space. The range of θ can be any $180°$ sector, and the range of ρ is $\pm R$, which is (at most) the length of the image diagonal.

The search for the normal line parameters can be carried out by dividing parameter space into discrete regions of size $(\Delta_\rho, \Delta_\theta)$, each represented by a (ρ, θ) parameter pair, and following a counting process.

Normal Parameter Search

Divide parameter space into a set of discrete points, (ρ_i, θ_j). These should reasonably cover the space.

Create a set of counters, one for each parameter pair, initialized to zero.

For each point (x_n, y_n)

For each θ_j compute $\rho = x_n \cos\theta_j + y_n \sin\theta_j$

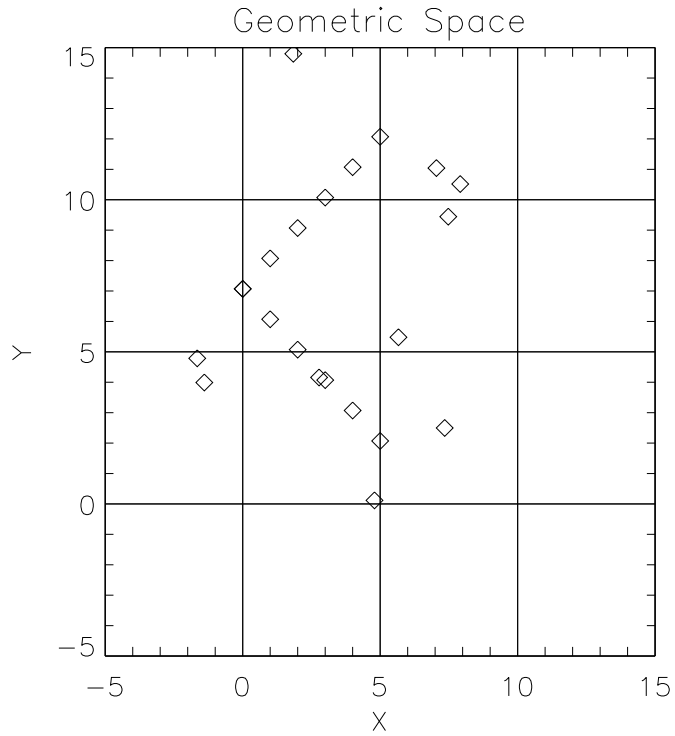
Find the point (ρ_i, θ_j) closest to (ρ, θ_j)

Increment counter for (ρ_i, θ_j)

The counters with the most points correspond to the parameter pairs for desirable lines in geometric space.

Example

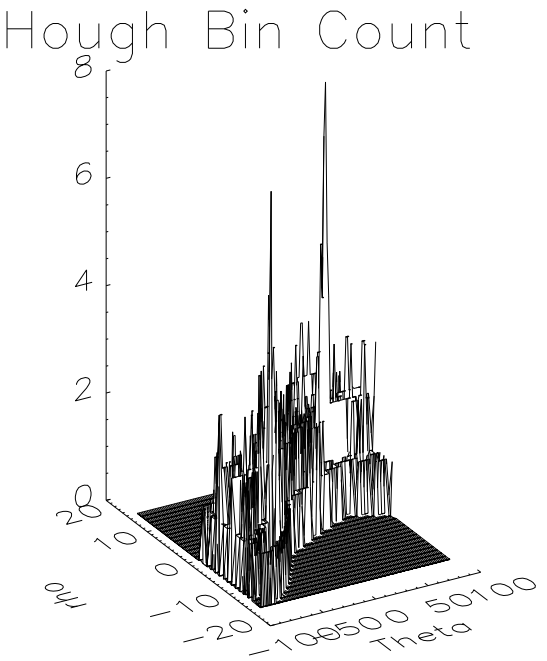
This is a geometric space which contains a number of points. The task is to find the major lines. There are a number of "noise" points that may cause confusion.



Example

The parameter count that is accumulated for each (ρ, θ) cell is illustrated by the surface plot at the right. Note that two cells have a count of 5 or 6 and the rest have much lower counts.

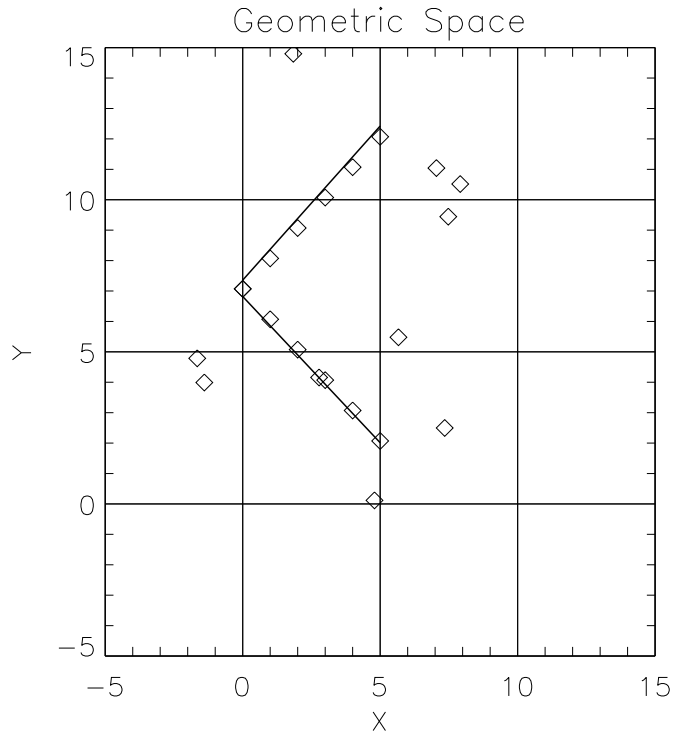
The lines corresponding to the two peaks are shown on the next page.



Example

The result of the search for lines produces the results shown at the right.

The search program finds the endpoints for each line segment.



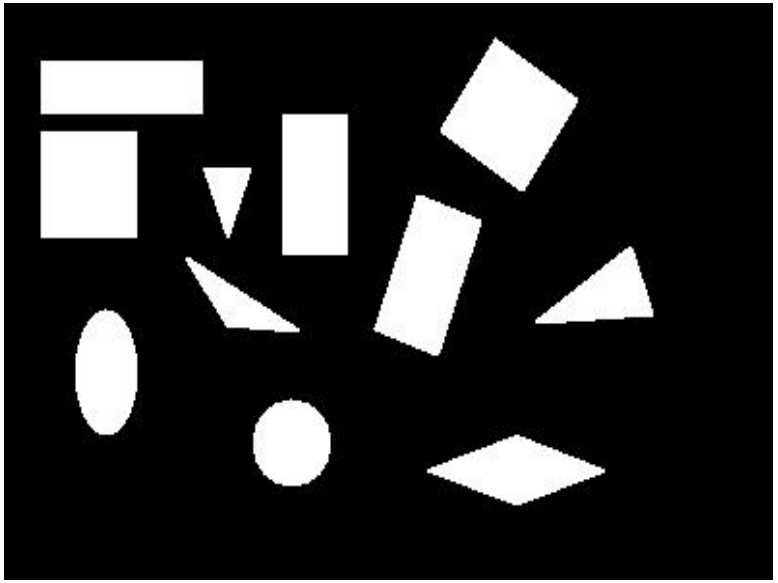
Robustness of Hough Transform

The Hough transform is able to find lines in the presence of noise. It is better than regression algorithms in dealing with outliers and multiple lines.

An example based on edges detected in a real image shows why this is necessary. Multiple lines and random points caused by noise in the imaging and detection process create a difficult environment for the search algorithm.

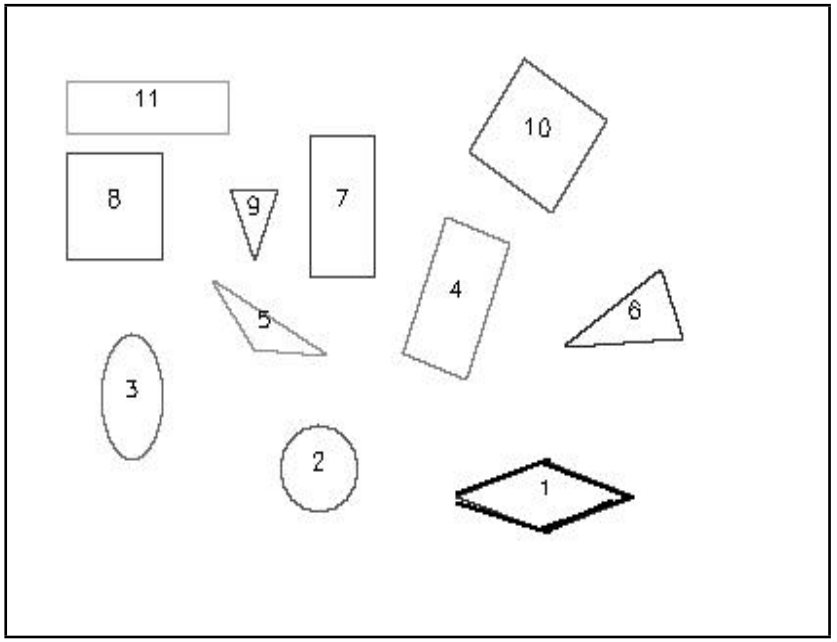
Multiple Object Detection

We can use the Hough transform to find lines that correspond to the edges of objects. Consider the image of several geometric objects shown at the right. This image will have a much larger number of lines. The program `LinearHoughLink` has several options that can be used to tune the search for lines to help eliminate those with undesired characteristics. More about that below.



Multiple Object Detection

An edge detection program can be used to find the points on the edges of the objects. The objects were located and numbered with a labelling program. The points that correspond to object 1



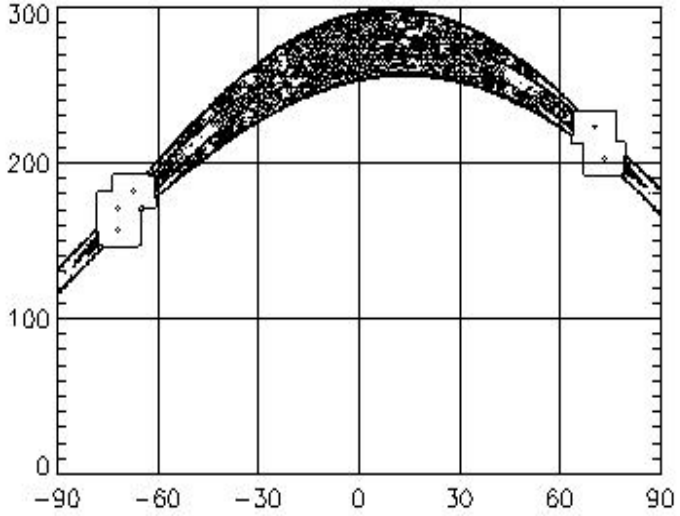
were selected and submitted to a the LinearHoughLink program. The lines that were returned were drawn on the image. We see that the lines correspond to the edges of the object.

Only the points on the boundary of object 1 were submitted on this trial.

Multiple Object Detection

A graph in (ρ, θ) space is shown at the right. The little rectangles with a dot in the middle indicate the locations of the (ρ, θ) parameters for each line.

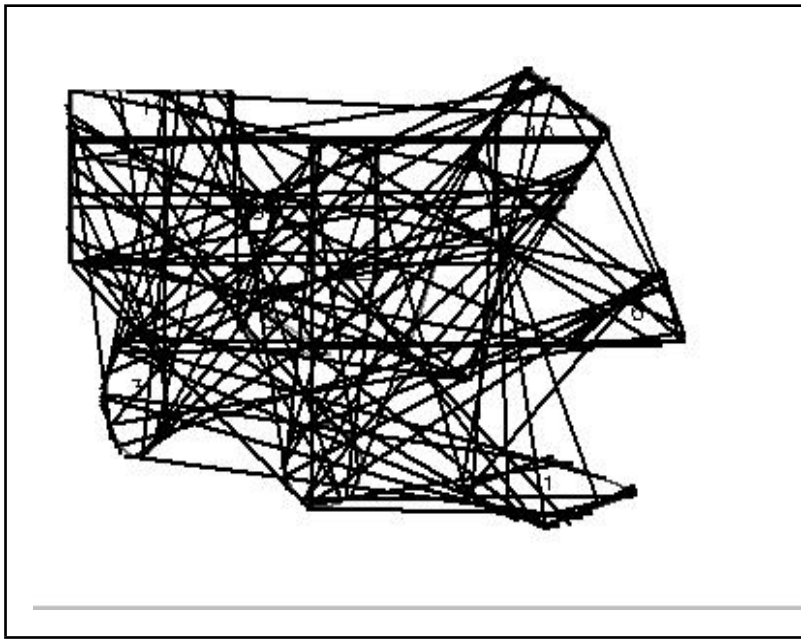
Program parameters were set to find up to five lines and to eliminate a neighborhood of 10×10 bins around each selected point in parameter space.



Multiple Object Detection

Running the program on all of the points in the image finds lots of lines–mostly ones that are unwanted. Why does this happen? The program parameters can be set to control this kind of situation.
For this run the control parameters were set very wide.

nlines	20	Max num lines to find
rhobins	300	Number of ρ bins
thetabins	300	Number of θ bins
thresh	30	Min number of pts per line
nbhd	10	Neighborhood radius
minLen	20	Min line length
maxLen	300	Max line length
maxStep	1000	Max line gap length

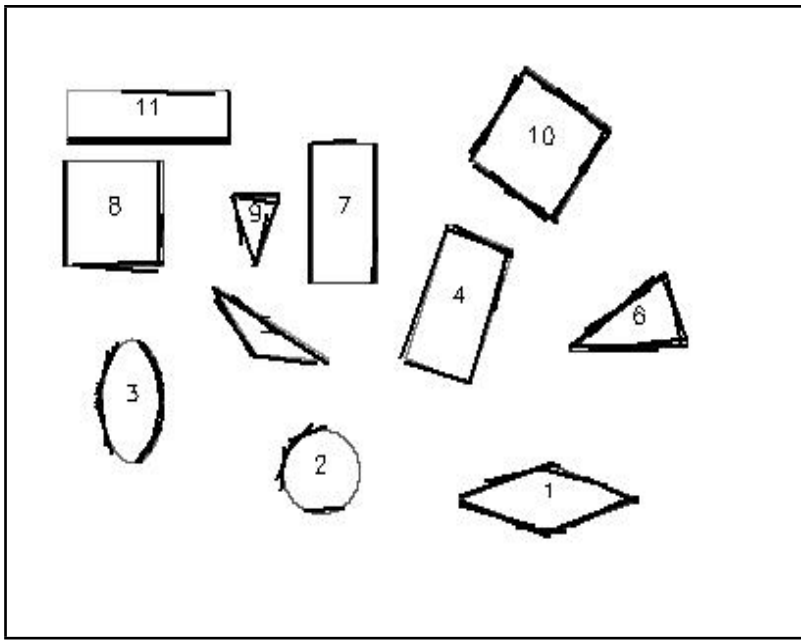


Multiple Object Detection

By changing the parameters for maximum and minimum line length and maximum gap width the results at the right are produced.

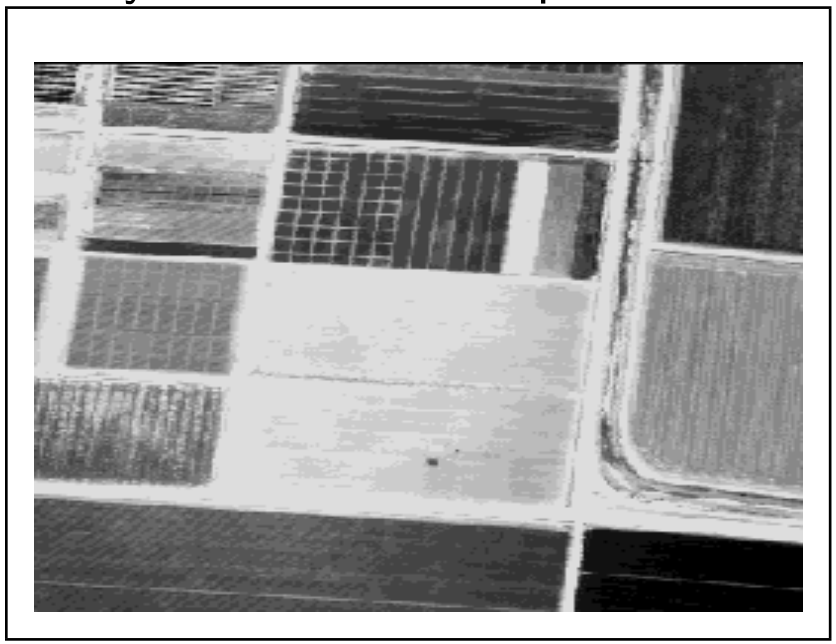
Lines at the end of 11, top of 8, bottom of 7 were probably masked by other lines that have similar parameter descriptions. Removing all the points on found lines and searching again can address this problem.

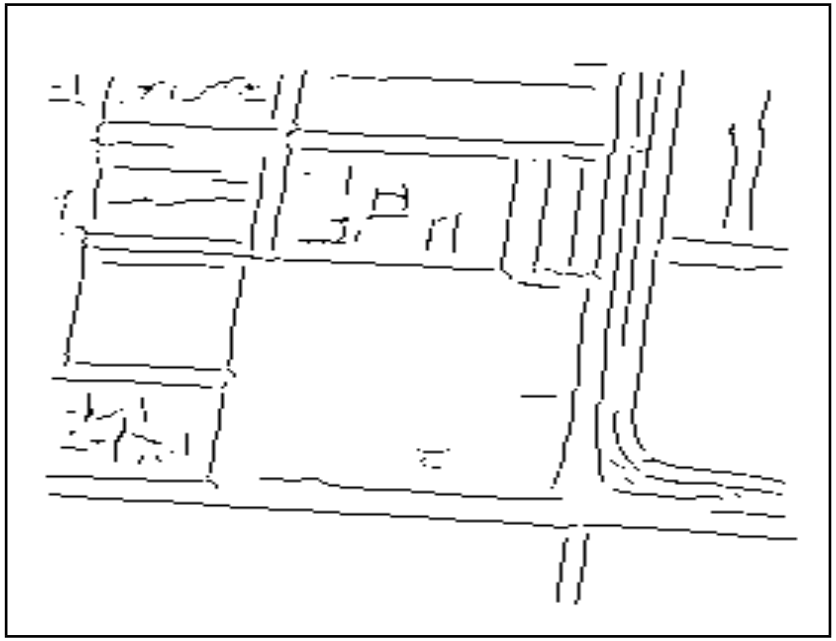
nlines	200	Max num lines to find
rhobins	300	Number of ρ bins
thetabins	300	Number of θ bins
thresh	30	Min number of pts per line
nbhd	10	Neighborhood radius
minLen	20	Min line length
maxLen	100	Max line length
maxStep	5	Max line gap length



Real Image Example

An aerial image of an agricultural region is shown on the left. The result of Canny edge detection is shown on the right. The Canny detector scale parameters have been chosen to hide fine detail.





Real Image Example – Continued

The edge image is shown on the left and lines detected by LinearHoughLink are highlighted in red on the right. Note that the longest lines are detected. The selection of detected lines is controlled by parameter settings in LinearHoughLink.