ECE5554 – Computer Vision
Lecture 4c – Hough Transform

Creed Jones, PhD

Today's Objectives

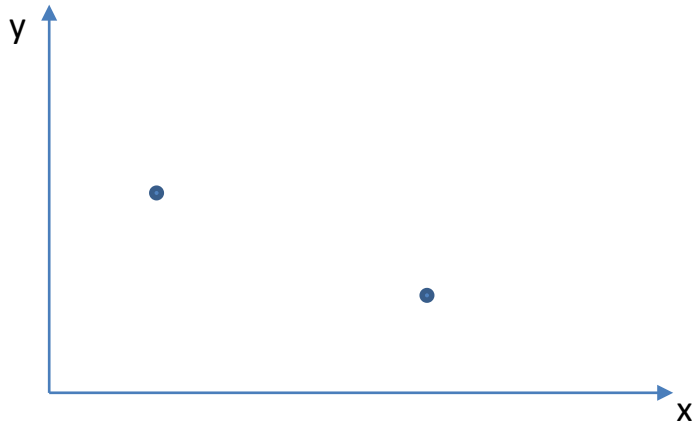
The Hough Transform
- General concept
- Parameter space
- Steps in the process
- ImageJ Hough implementation
- Some examples

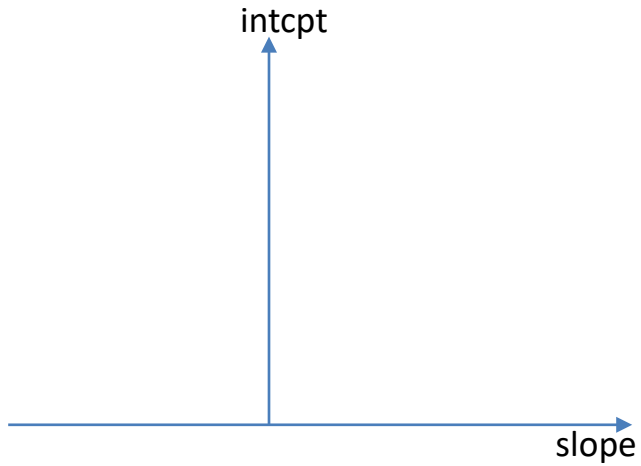
Circular Hough Transform
- concept
- implementation

Generalized Hough Transform

The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

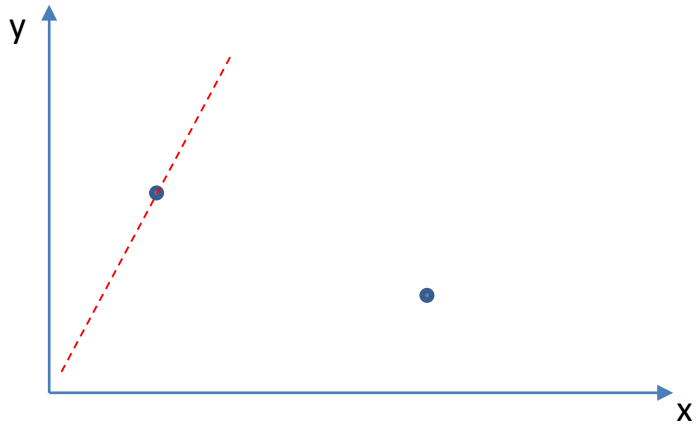
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

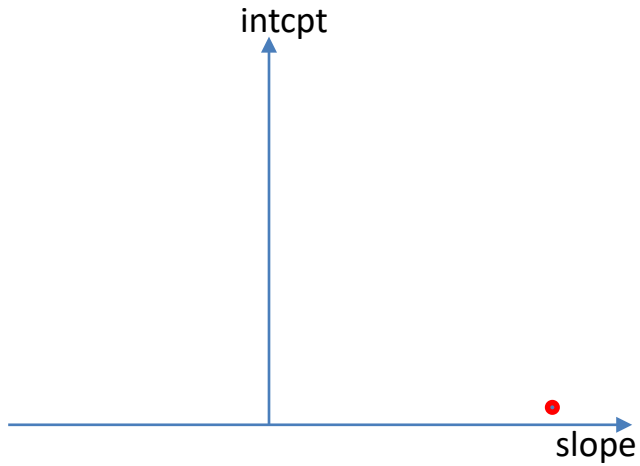




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

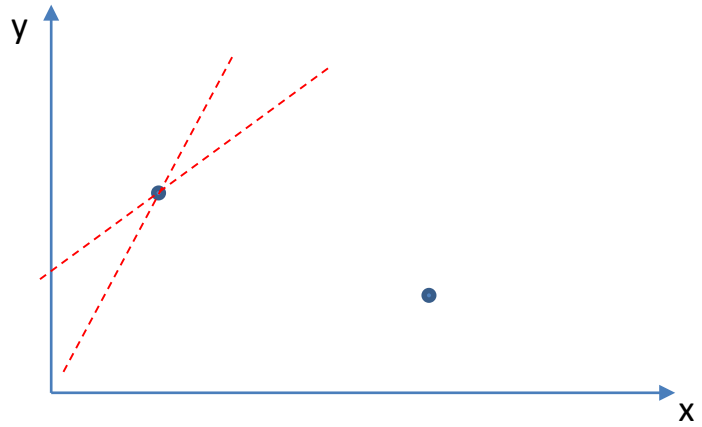
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

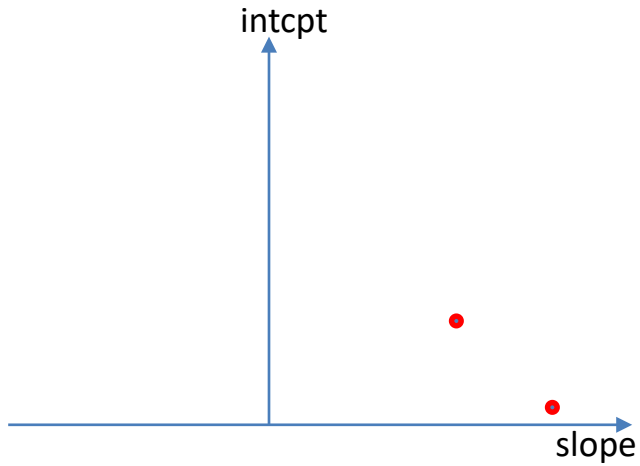




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

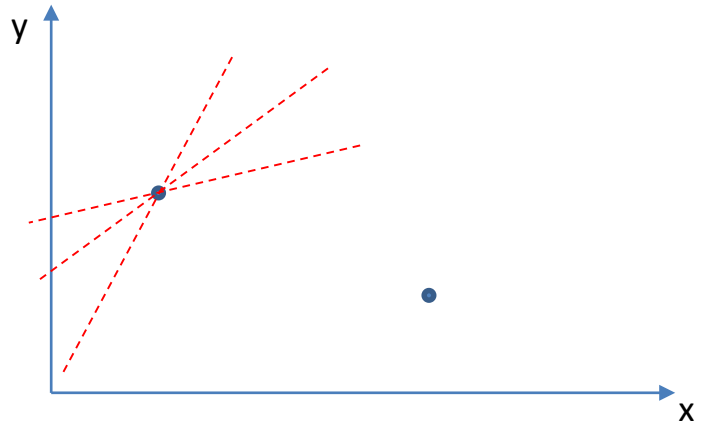
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

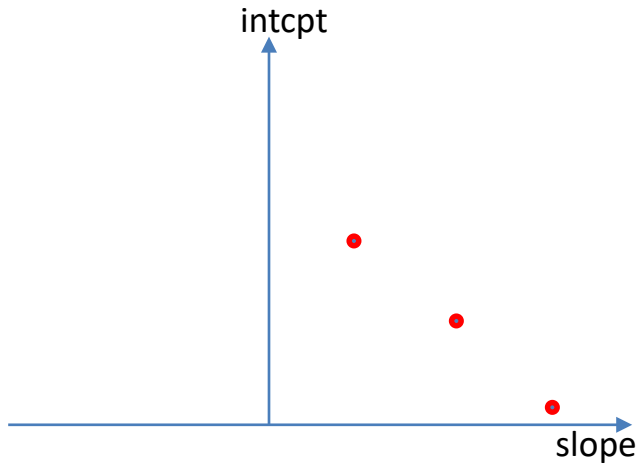




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

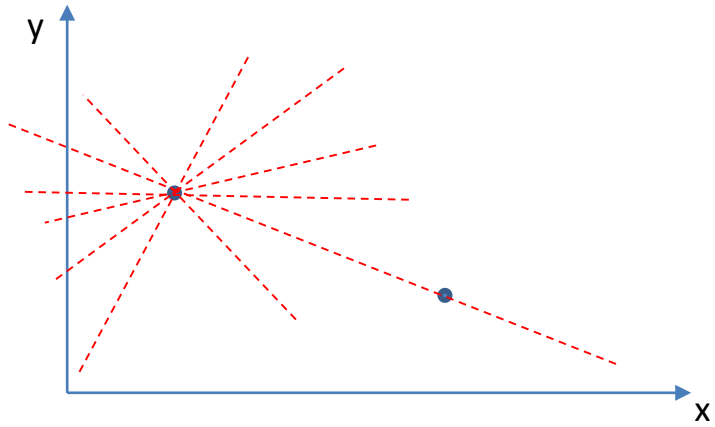
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

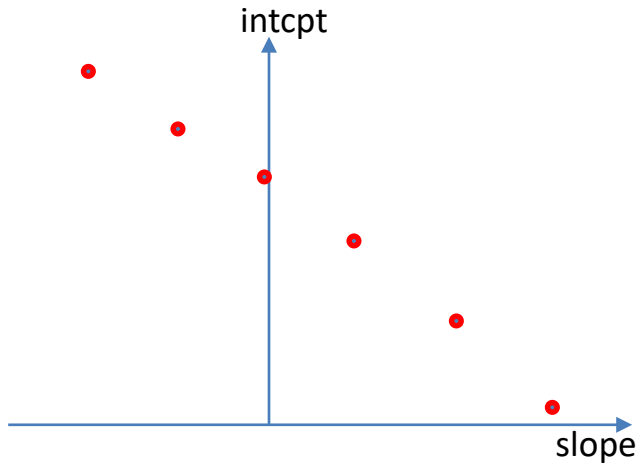




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

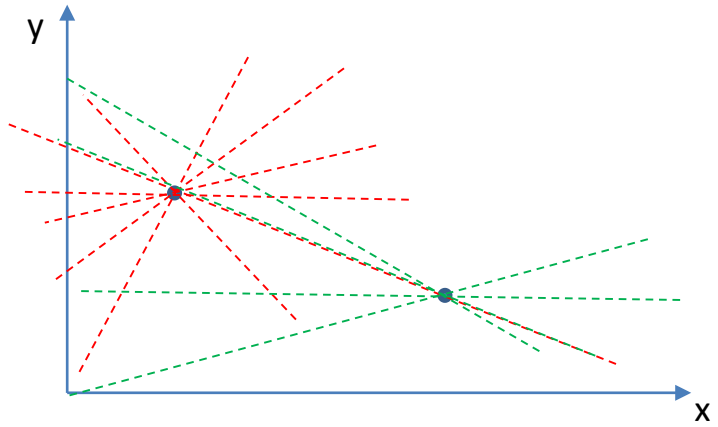
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

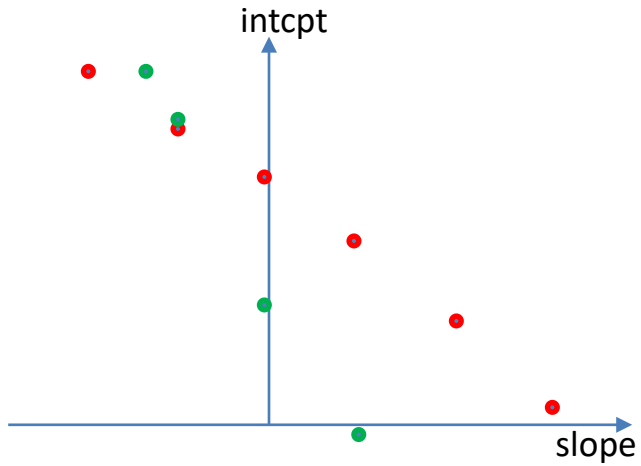




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

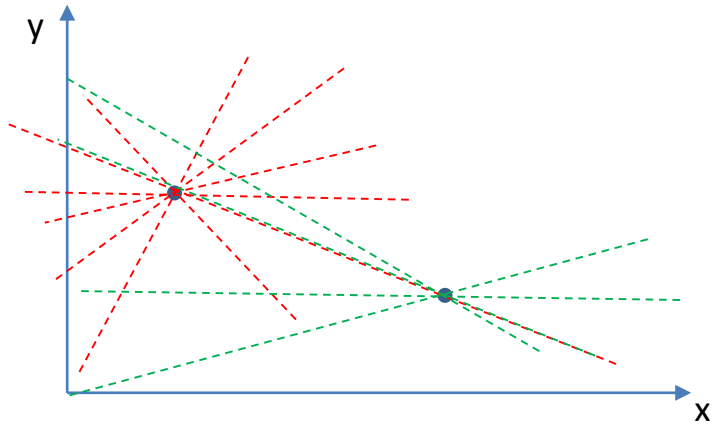
- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

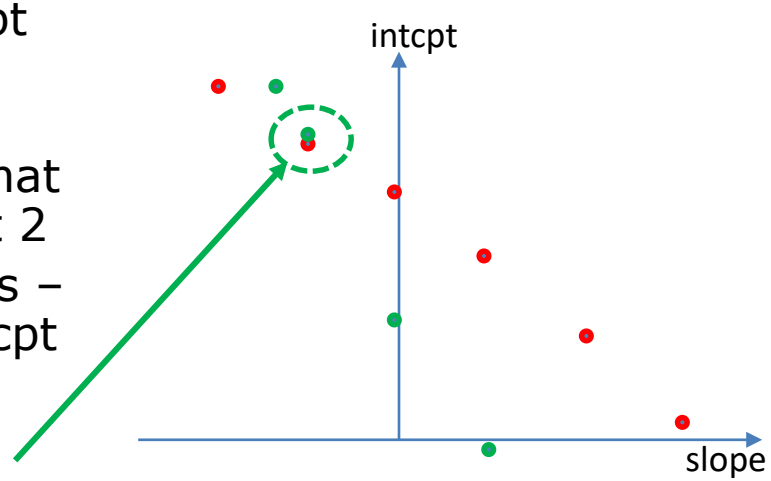




The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

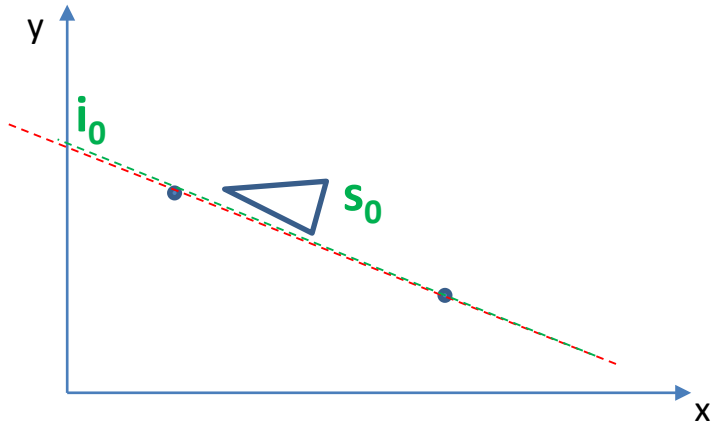


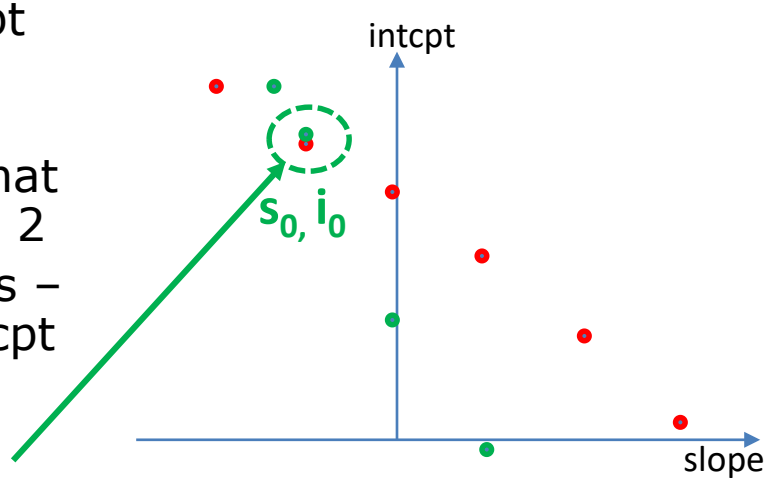


The area where two points overlap indicates the slope-intcpt of the shared line

The *Hough Transform* converts an image representation into a parameter representation in which one or more common shapes (such as lines) can be easily discerned

- Consider an image with only two foreground points in it
- Each point can be on an infinite set of lines, geometrically
- Define a new set of axes – slope and intercept
- In these axes, plot the set of slope-intcpt points that define lines that go through image point 1
- Also plot the set of slope-intcpt points that define lines that go through image point 2
- Only one line passes through both points – it's defined by the place in the slope-intcpt plane where the two sets of points intersect

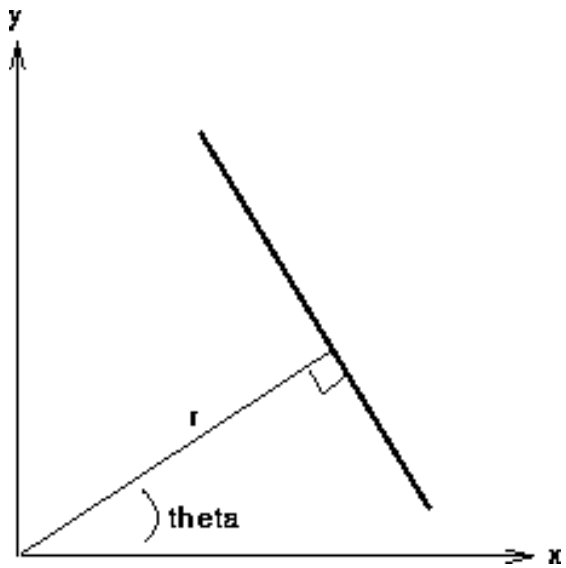




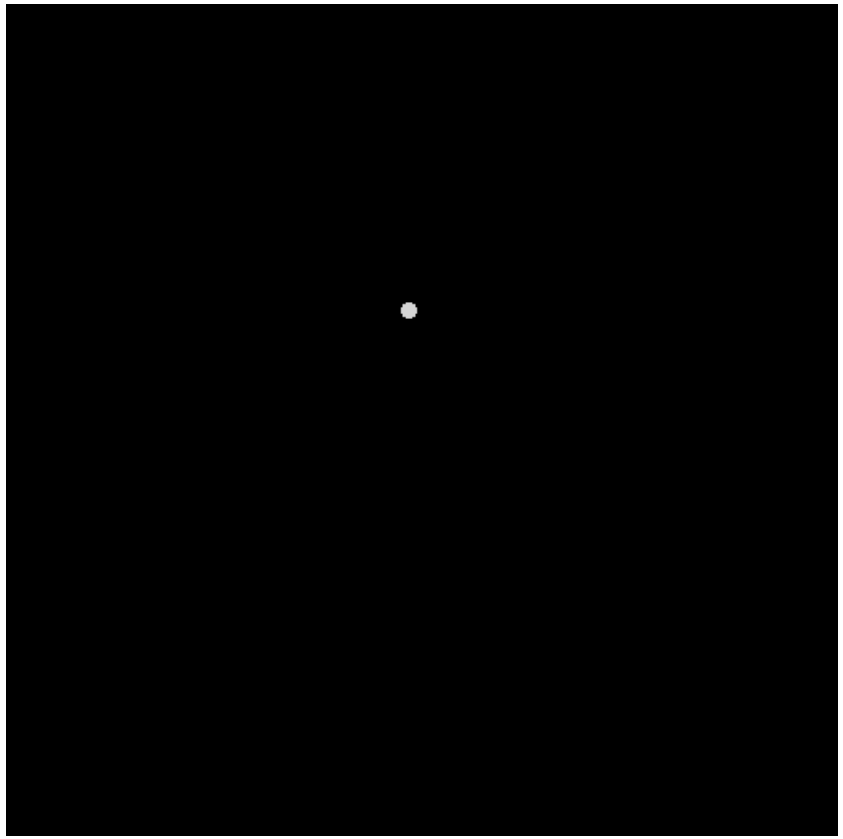
The area where two points overlap indicates the slope-intcpt of the shared line

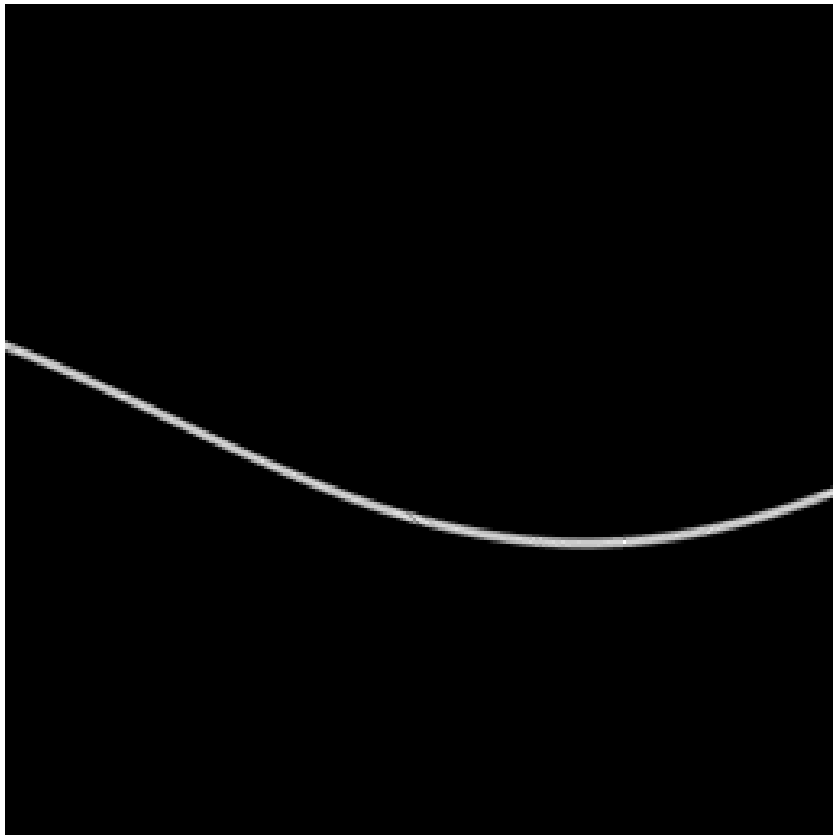
The *parameter space* is an alternate representation of image points that will allow common shapes to be recognized

- In this simple example, the parameter space was slope-intercept
  - This has severe problems with vertical lines: slope $\rightarrow \infty$
- For line-determination, a more common parameter space is formed by the angle and length of the vector normal to a given line, and passing through the origin…

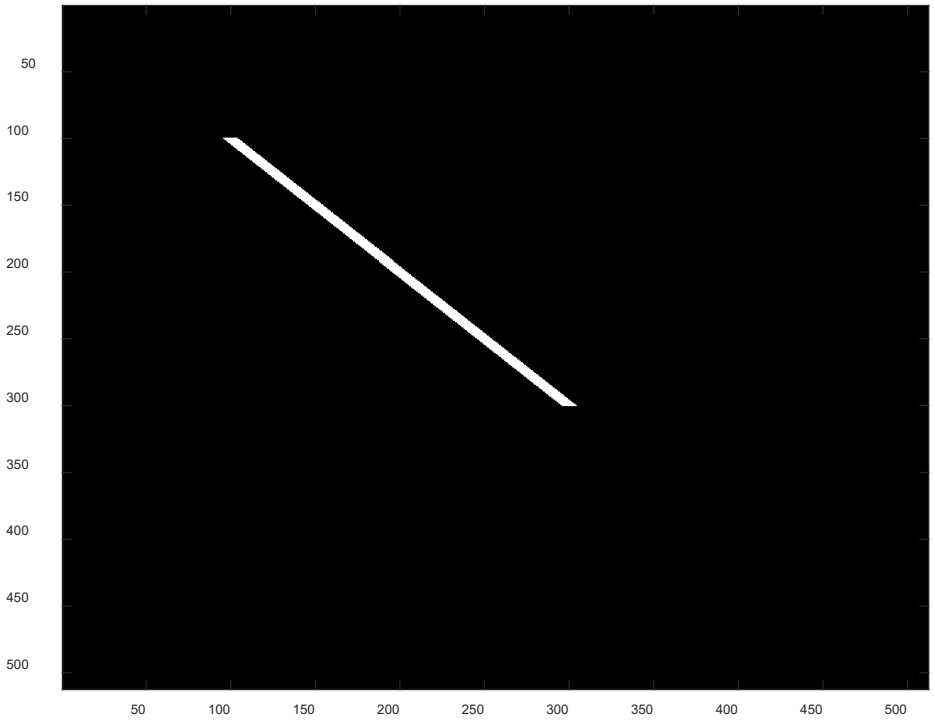


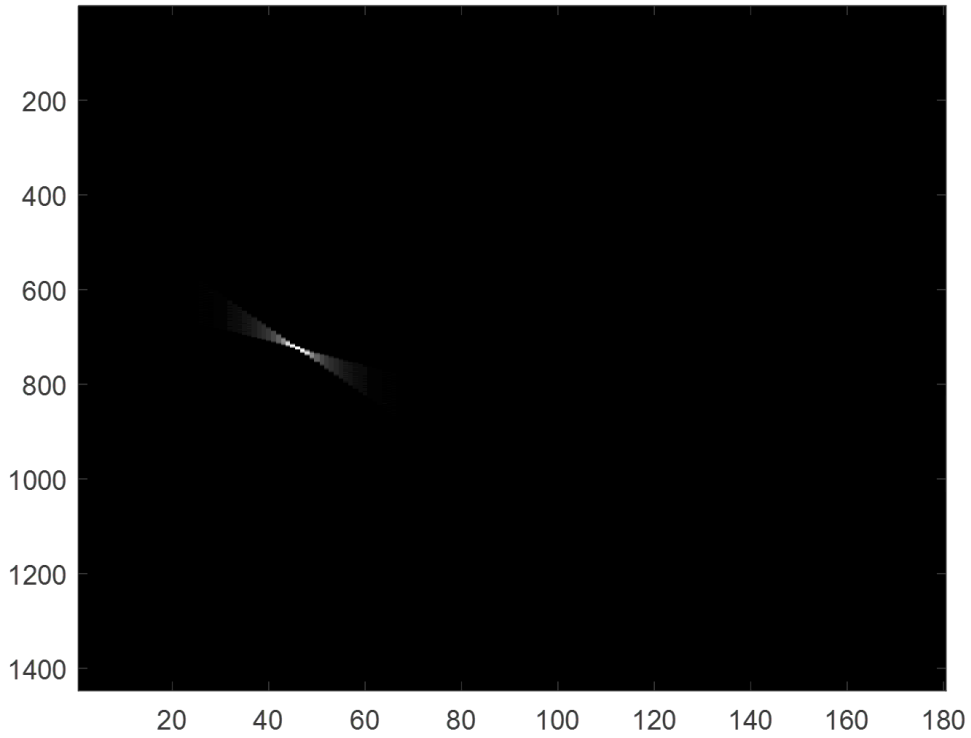
A point in the image will show up in the (r, θ) parameter space as a sinusoidal curve



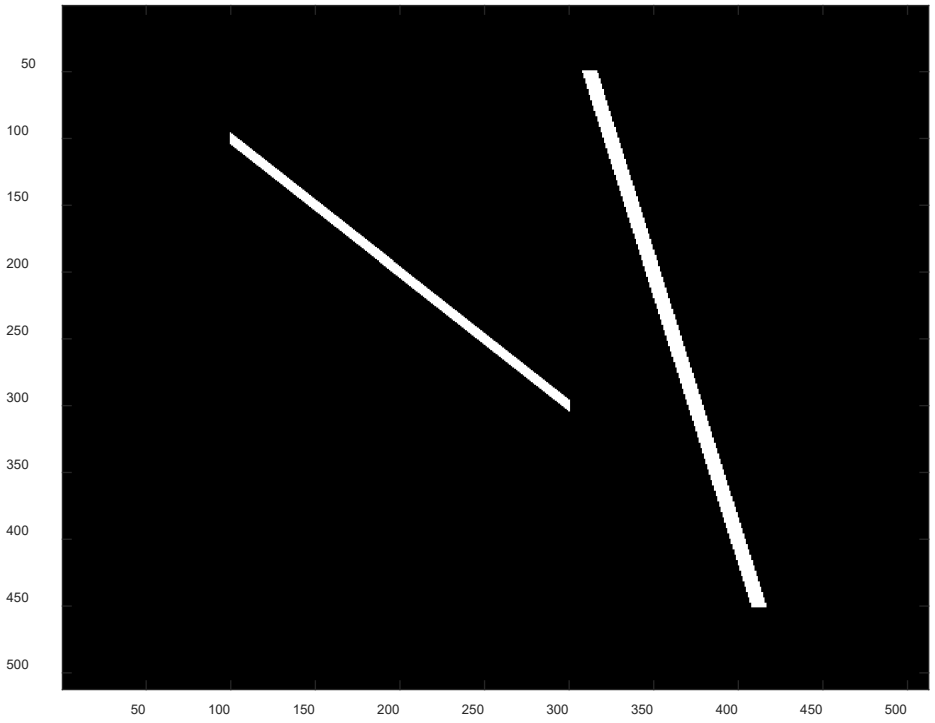


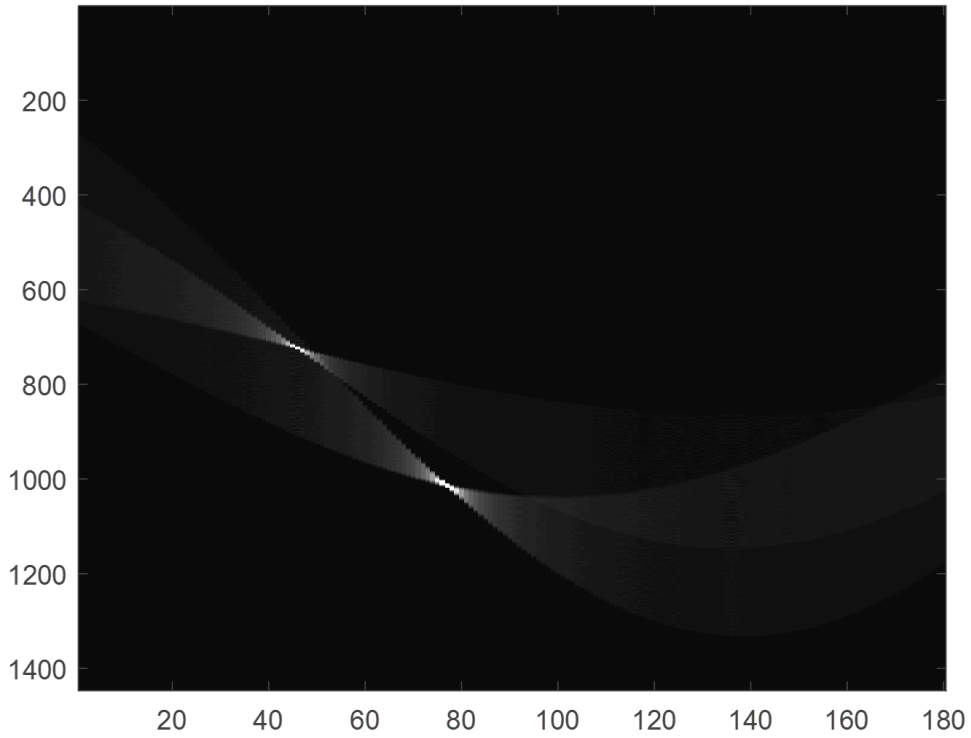
A straight line in the image will show up in the (r, θ) parameter space as a small portion of a set of sinusoidal curves – approaching a point



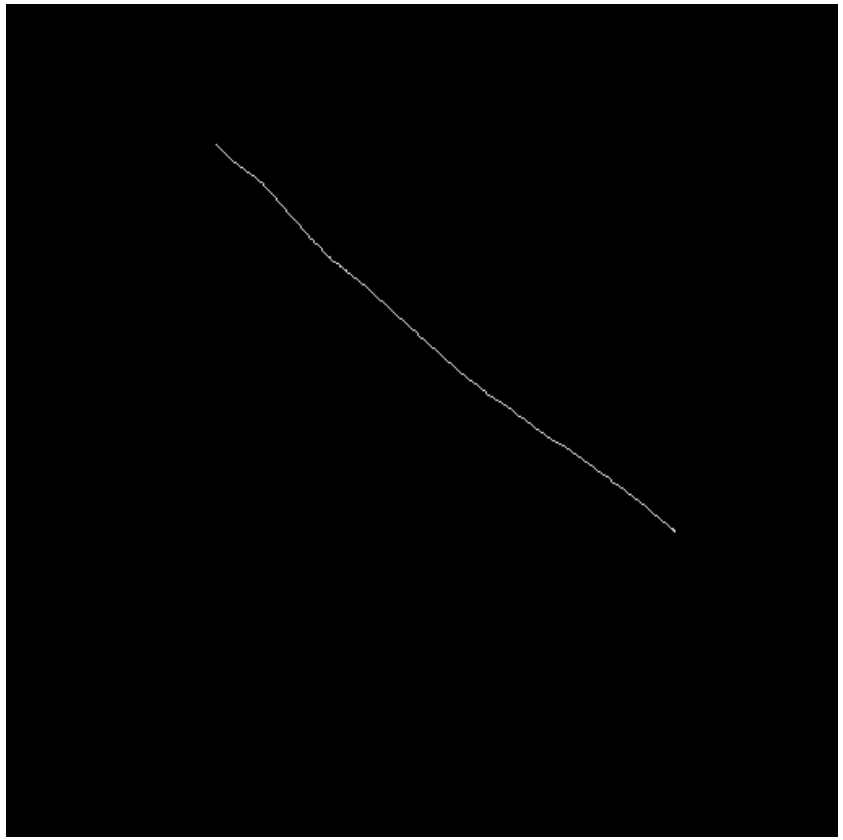


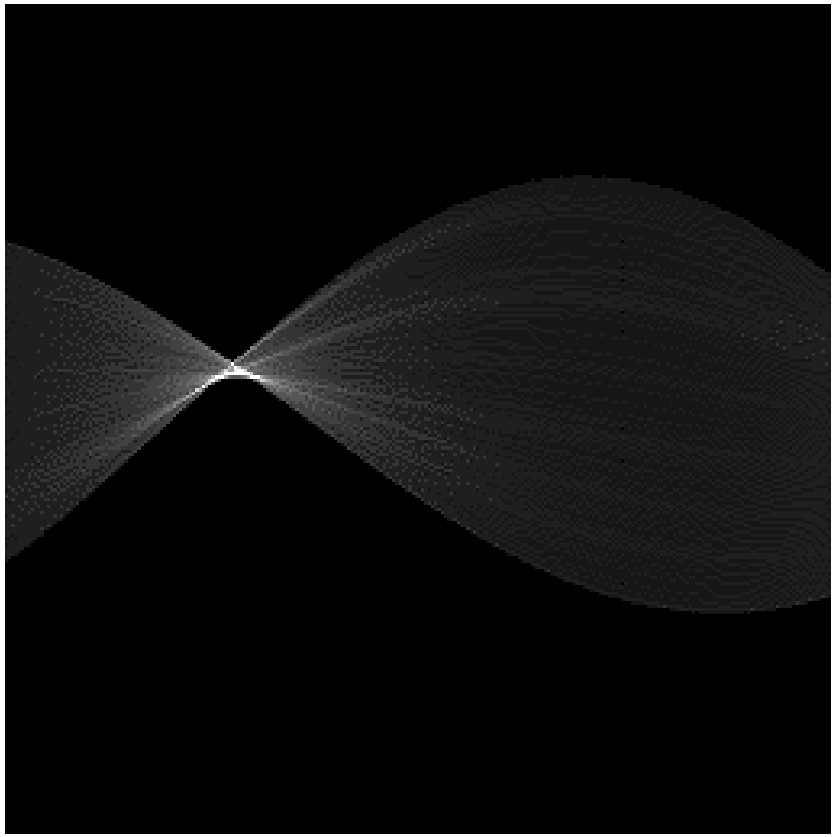
A straight line in the image will show up in the (r, θ) parameter space as a small portion of a set of sinusoidal curves – approaching a point





A (mostly) straight line in the image will show up in the (r, θ) parameter space as a small portion of a set of sinusoidal curves – approaching a point

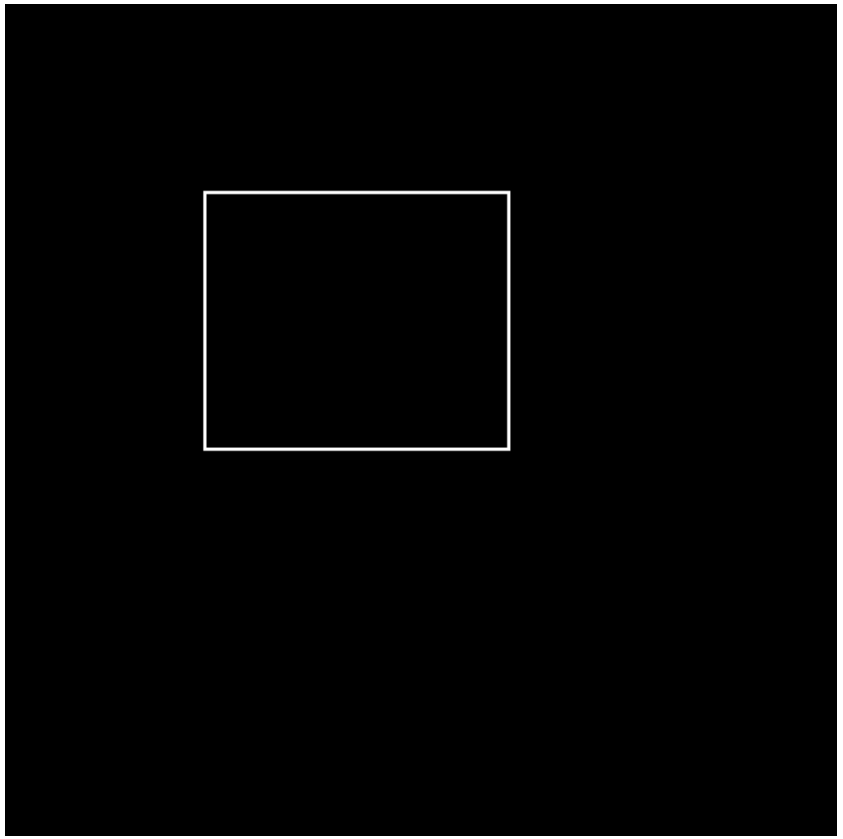


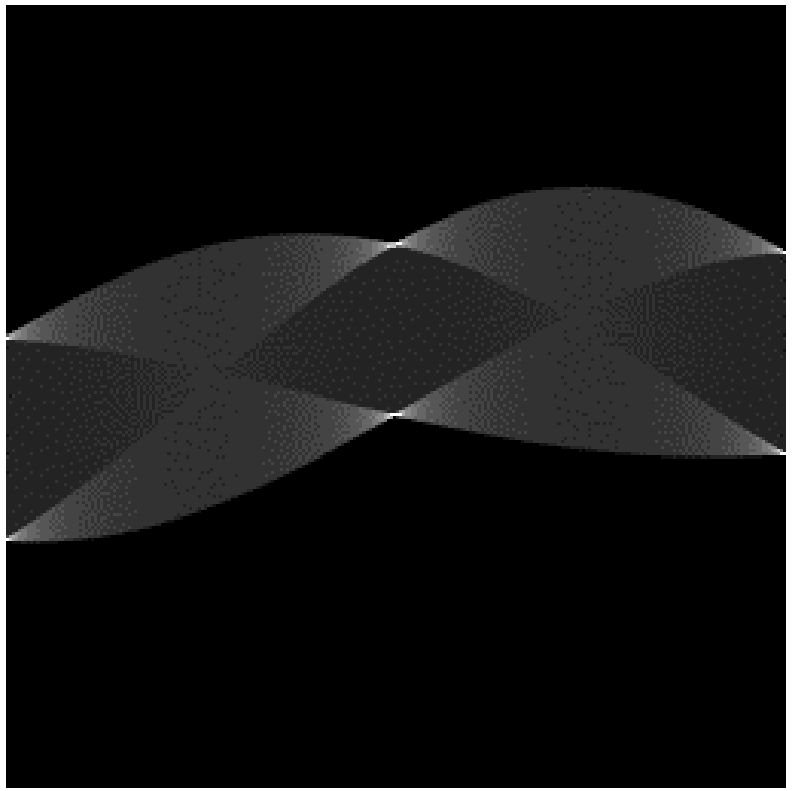


Here are the steps in applying the Hough process to an image, to locate linear features

1. Form the edge image! Linear features are identified by their straight-line edges. (Canny is best because the one-pixel wide edges make for cleaner parameter space representations)
2. Apply the Hough transform to find the parameter space representation of the image
3. Examine the parameter space to identify "peaks" or local maxima in the parameter space
4. Read out the (r, θ) of the maxima and identify these as strong lines in the original image

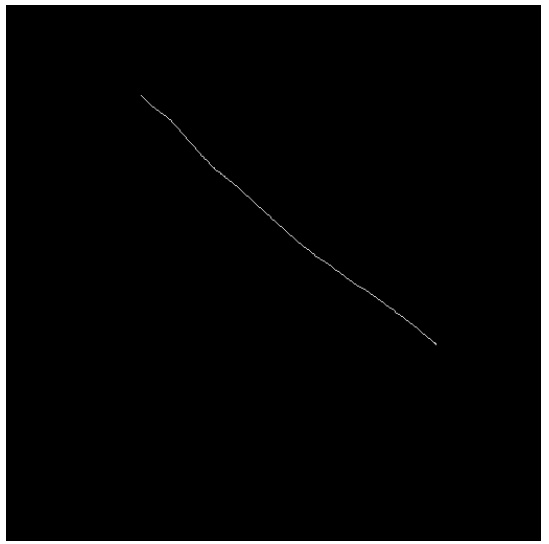
The four lines that compose a square lead to four clearly identifiable maxima in the parameter space

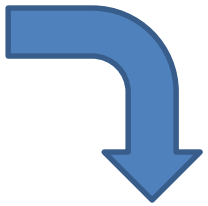


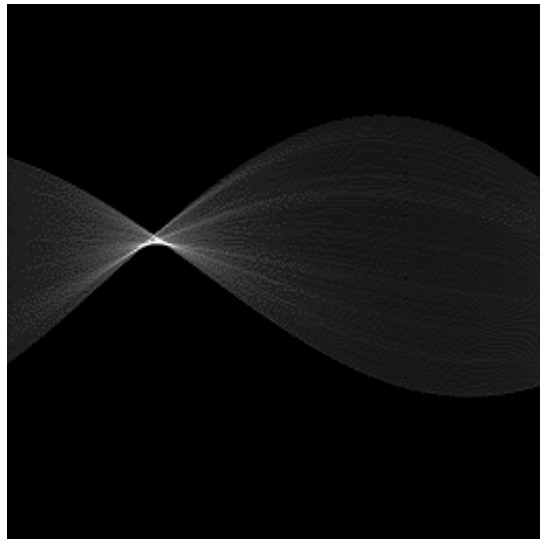


Does the Hough Transform satisfy our three desirable properties of a transform representation?

- Is it reversible?
-
- Is the transform representation useful?
-
- Does the transform representation have a useful interpretation?



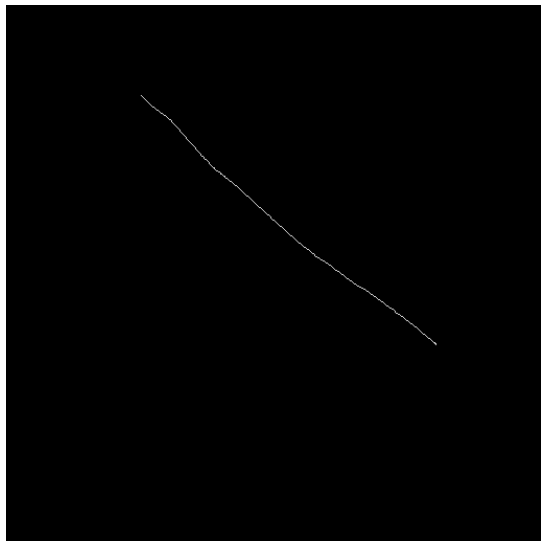


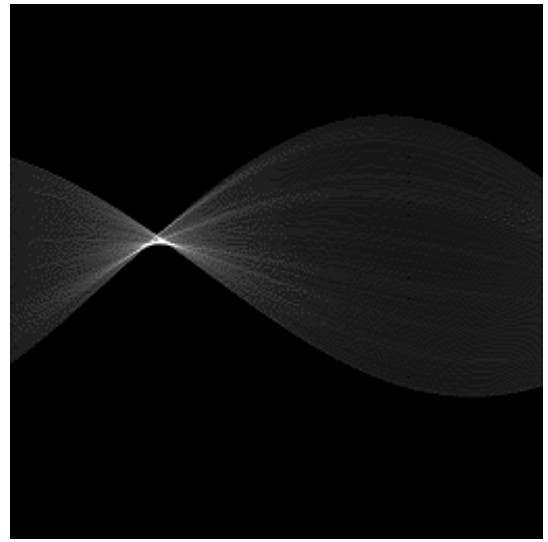


Does the Hough Transform satisfy our three desirable properties of a transform representation?

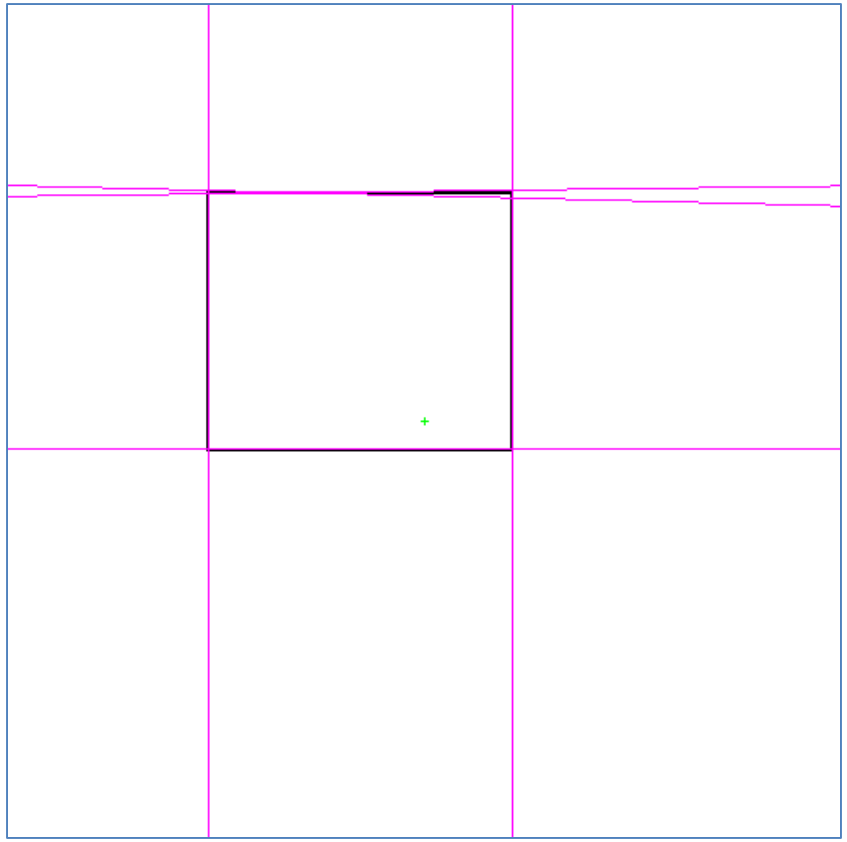
- Is it reversible?
- Yes, but we won't prove it

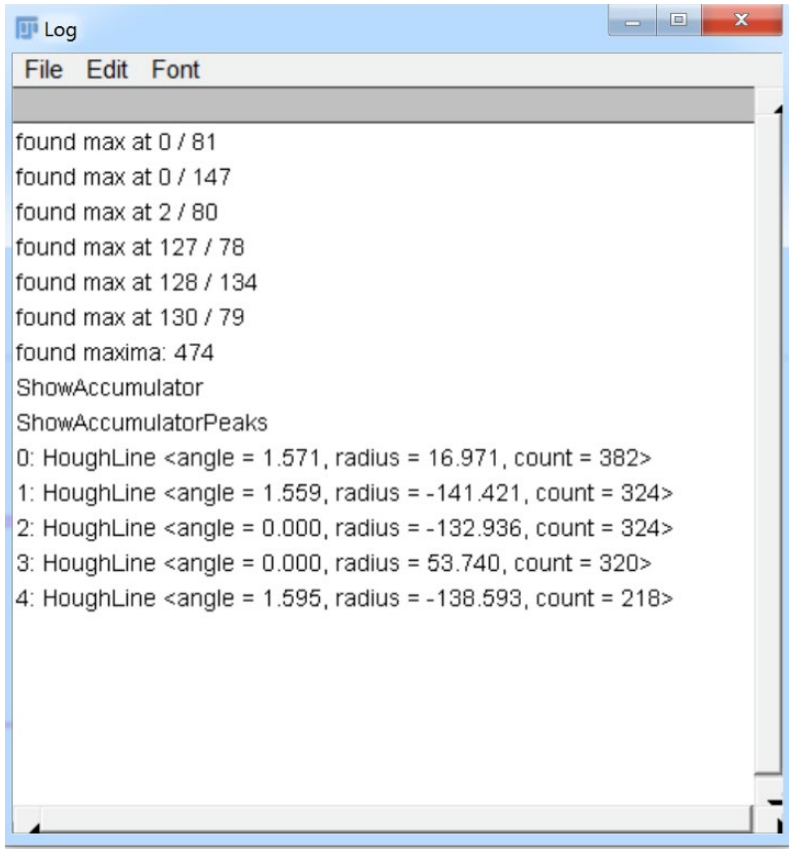
- Is the transform representation useful?
- Yes – points that fit a line appear together
- Does the transform representation have a useful interpretation?
- Yes – distance from origin and angle – or slope/intercept



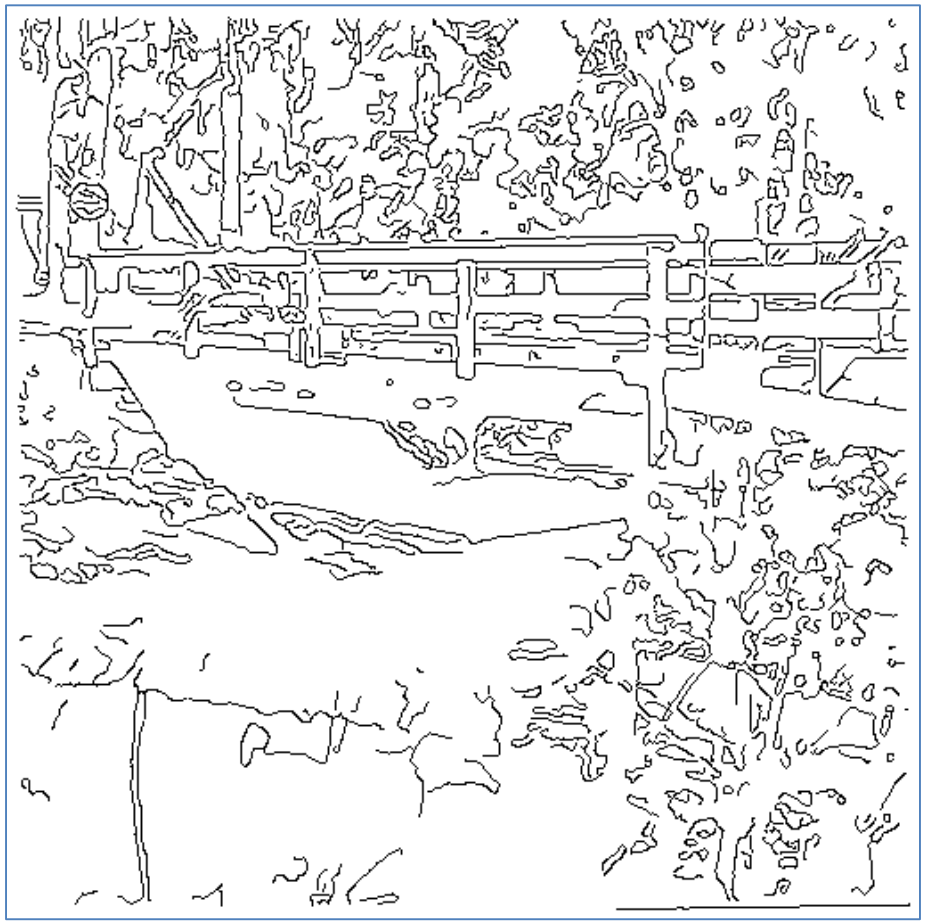


From identifying these maxima, we can determine the four sides of the square and plot the resulting lines (note the fifth line; the Hough plugin was set to identify the best five lines)

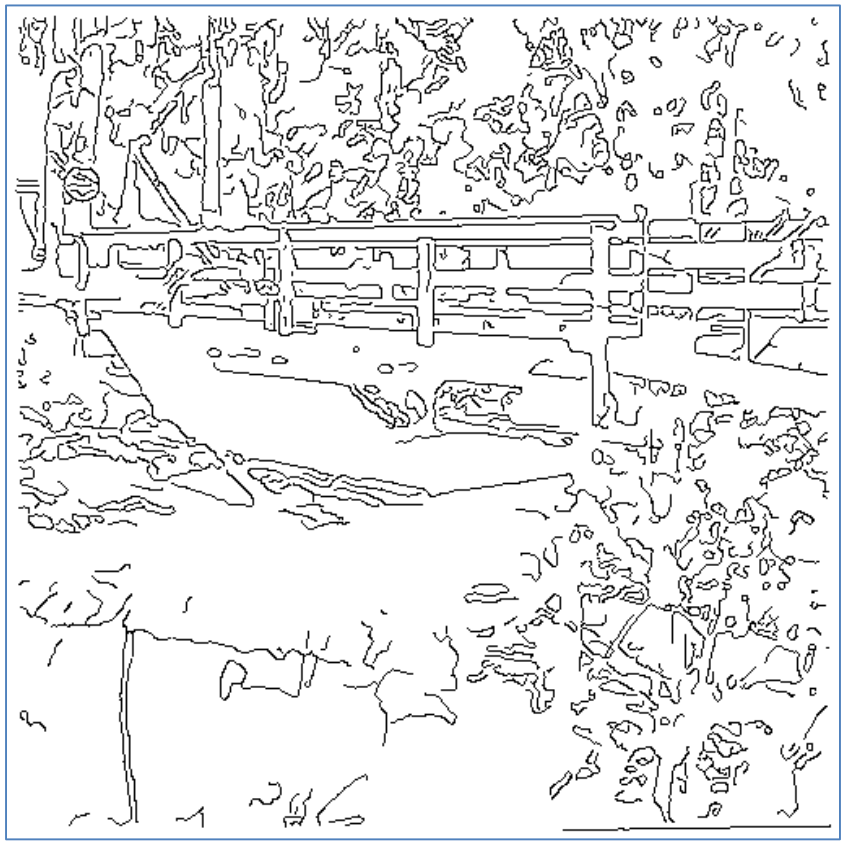


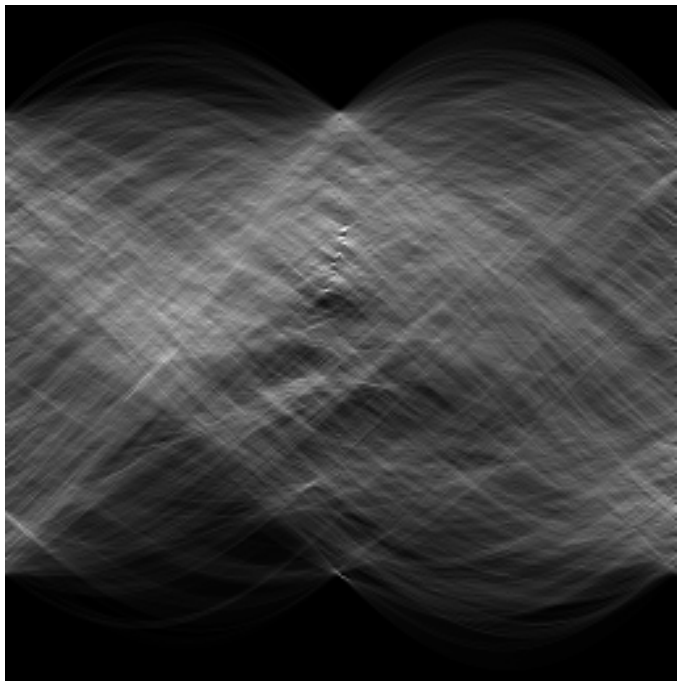


From the bridge image, form the Canny edge picture; edges are all one pixel wide

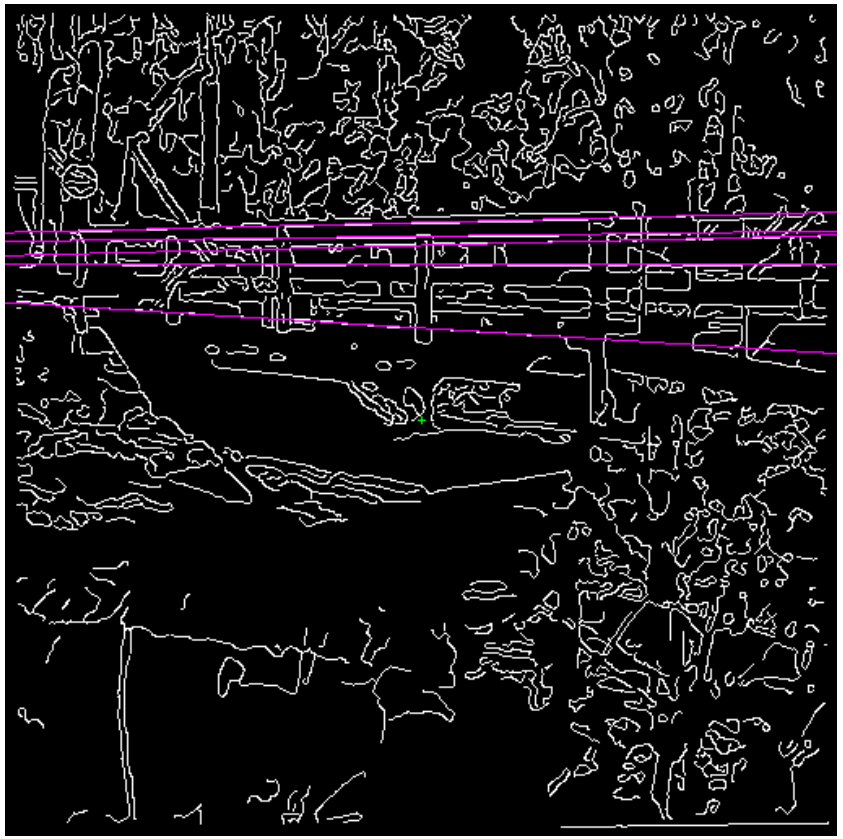


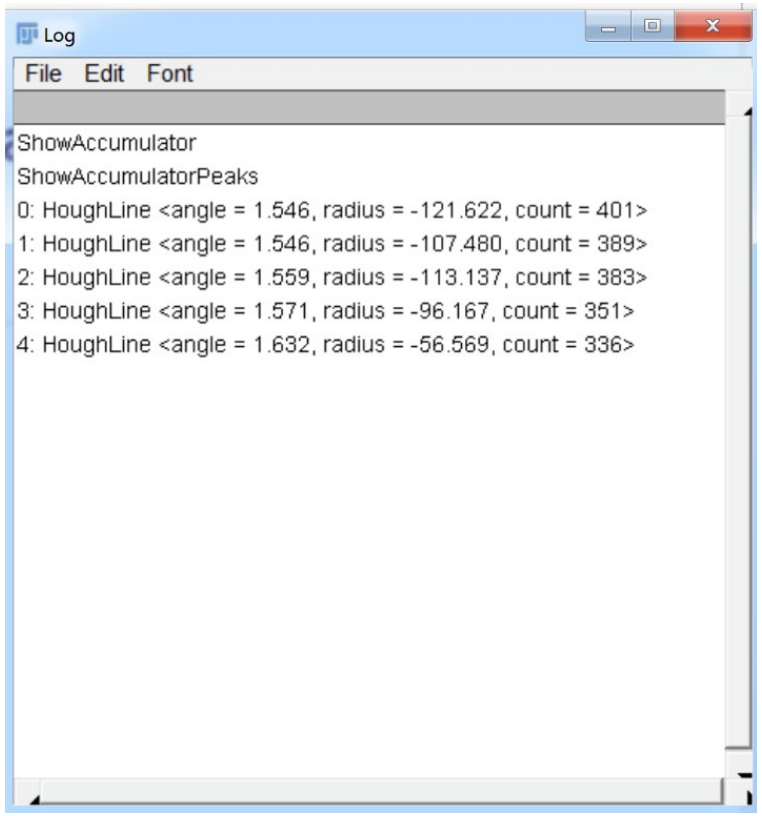
The edge image is transformed to the parameter space – there is a lot of activity but a number of peaks near the middle are clearly seen



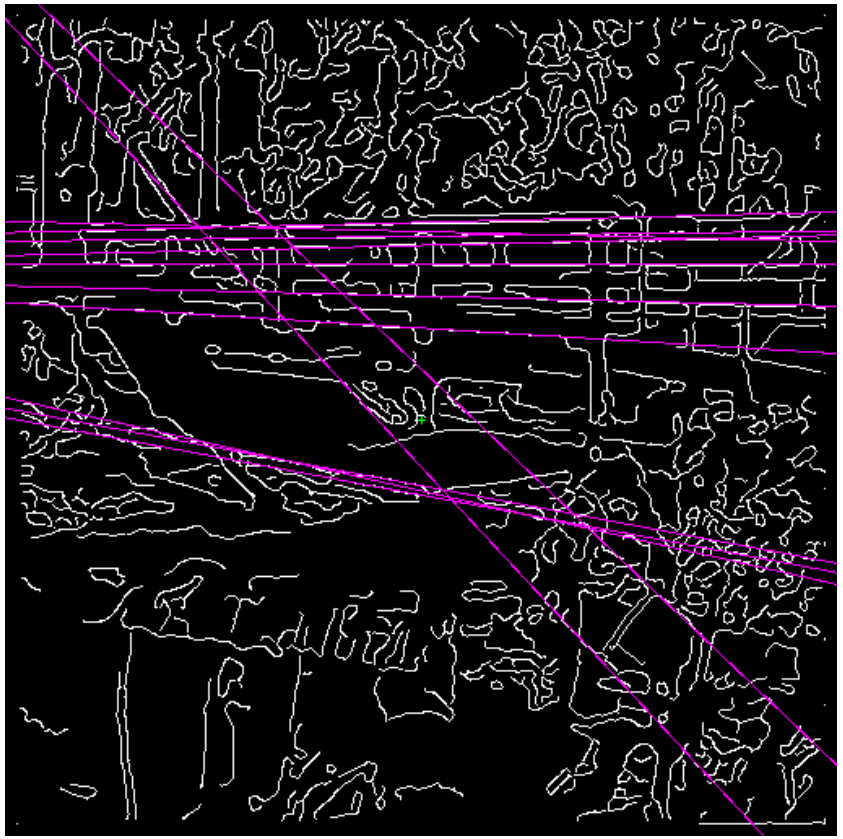


The five strongest peaks are generated by the lines that make up the bridge





If we allow the Hough plugin to find the best 12 lines, we find coincidences between small linear features in the foliage



Log

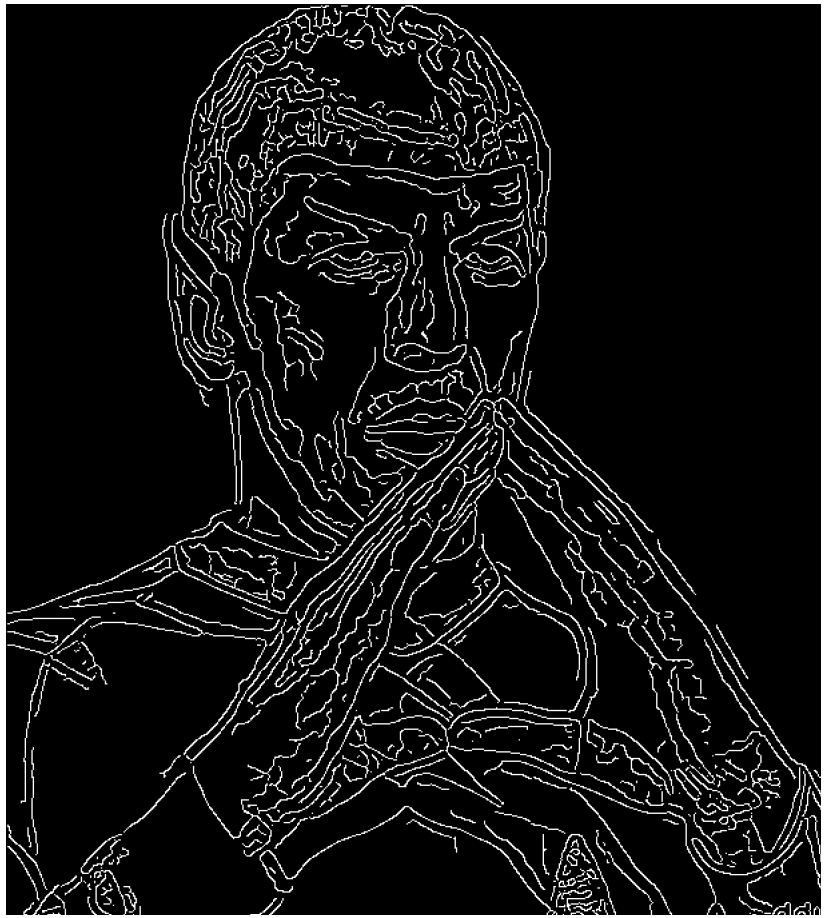
File Edit Font

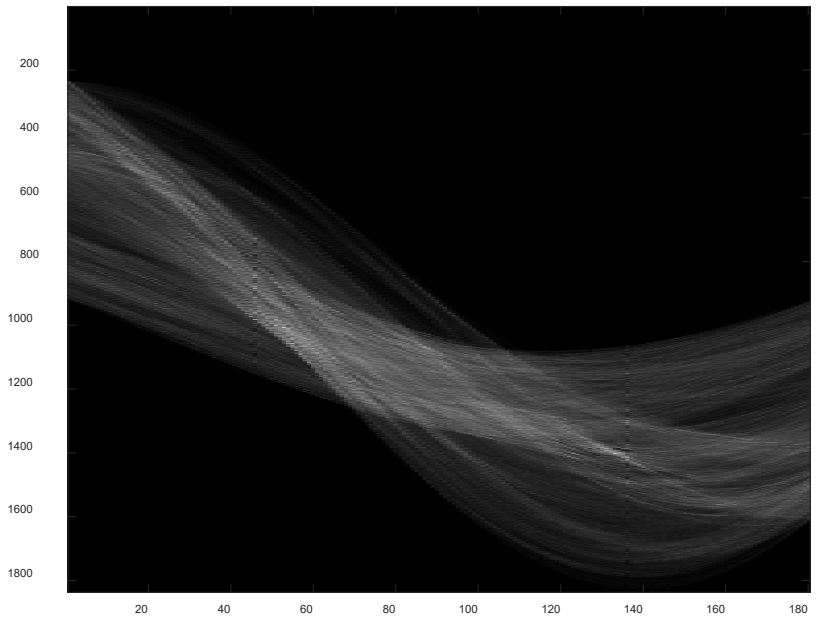
ShowAccumulator
ShowAccumulatorPeaks
0: HoughLine <angle = 1.546, radius = -121.622, count = 370>
1: HoughLine <angle = 1.571, radius = -96.167, count = 343>
2: HoughLine <angle = 1.546, radius = -107.480, count = 332>
3: HoughLine <angle = 1.559, radius = -113.137, count = 329>
4: HoughLine <angle = 1.632, radius = -56.569, count = 285>
5: HoughLine <angle = 2.332, radius = -22.627, count = 284>
6: HoughLine <angle = 1.595, radius = -76.368, count = 265>
7: HoughLine <angle = 1.595, radius = -115.966, count = 260>
8: HoughLine <angle = 1.755, radius = 39.598, count = 258>
9: HoughLine <angle = 1.792, radius = 42.426, count = 258>
10: HoughLine <angle = 1.755, radius = 45.255, count = 249>
11: HoughLine <angle = 2.393, radius = 19.799, count = 248>

```
clear all; close all;

% Now try a real image
rawImg = imread('C:\\Data\\spock.png');
marg = 20;
numLines = 16;
inputIM = rawImg(marg:size(rawImg,1)-marg, marg:size(rawImg,2)-marg, :);
inputIM2 = rgb2gray(inputIM);   % Convert to gray-level and run Canny
BW2 = edge(inputIM2, 'canny');
figure; imshow(BW2);
[H,theta,rho] = hough(BW2);
P = houghpeaks(H, numLines, 'threshold',ceil(0.8*max(H(:))));
lines = houghlines(BW2, theta, rho, P);
figure; imshow(inputIM2); hold on;
max_len = 0;
for k = 1:min(numLines, length(lines))
   xy = [lines(k).point1; lines(k).point2];
   sprintf("Line #%d:, pt1 = [%d, %d], pt2 = [%d, %d]",\
         k,xy(1,1),xy(1,2),xy(2,1),xy(2,2))
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

   % Plot beginnings and ends of lines
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
end
```

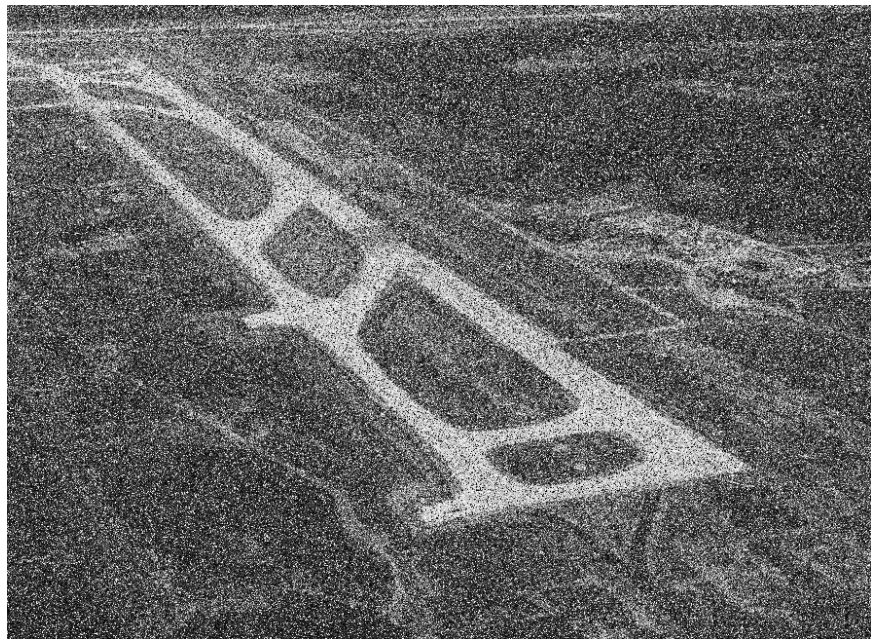
```
Line #1:, pt1 = [391, 289], pt2 = [277, 417]
Line #2:, pt1 = [199, 503], pt2 = [99, 614]
Line #3:, pt1 = [156, 56], pt2 = [181, 88]
Line #4:, pt1 = [197, 109], pt2 = [334, 285]
Line #5:, pt1 = [348, 302], pt2 = [385, 349]
Line #6:, pt1 = [445, 427], pt2 = [501, 498]
Line #7:, pt1 = [545, 554], pt2 = [611, 639]
Line #8:, pt1 = [175, 33], pt2 = [224, 113]
Line #9:, pt1 = [248, 154], pt2 = [310, 257]
Line #10:, pt1 = [321, 275], pt2 = [424, 447]
Line #11:, pt1 = [151, 64], pt2 = [217, 129]
Line #12:, pt1 = [266, 179], pt2 = [321, 234]
Line #13:, pt1 = [336, 249], pt2 = [489, 402]
Line #14:, pt1 = [122, 202], pt2 = [170, 251]
Line #15:, pt1 = [248, 328], pt2 = [336, 416]
Line #16:, pt1 = [469, 550], pt2 = [540, 620]
```

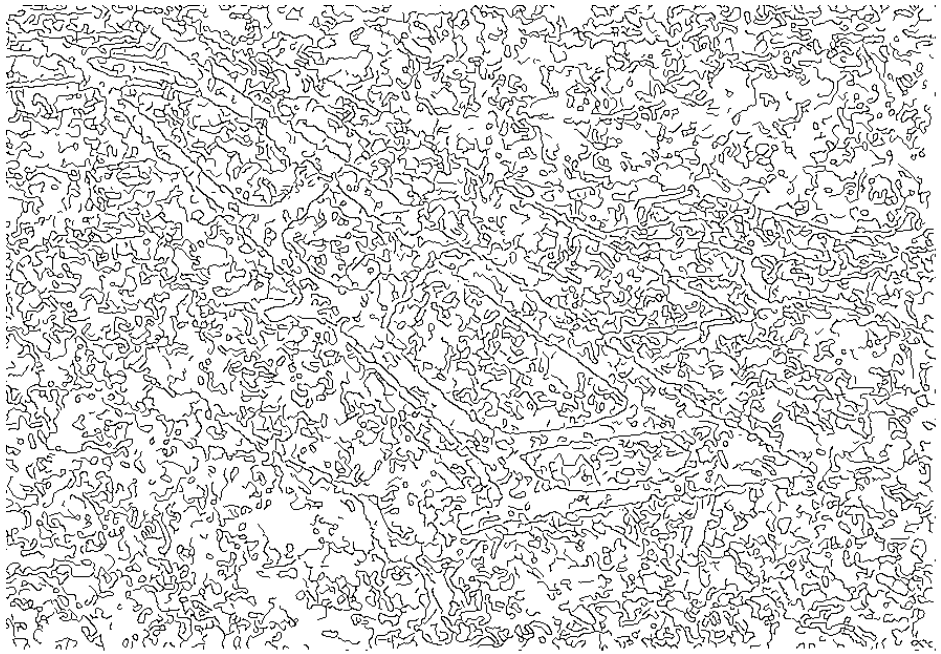
Hough lines on a real image



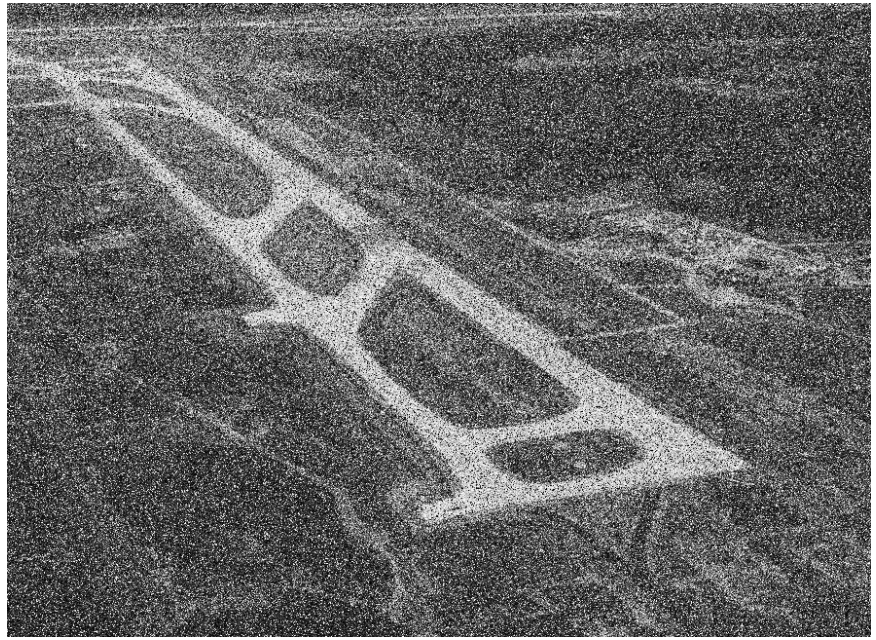


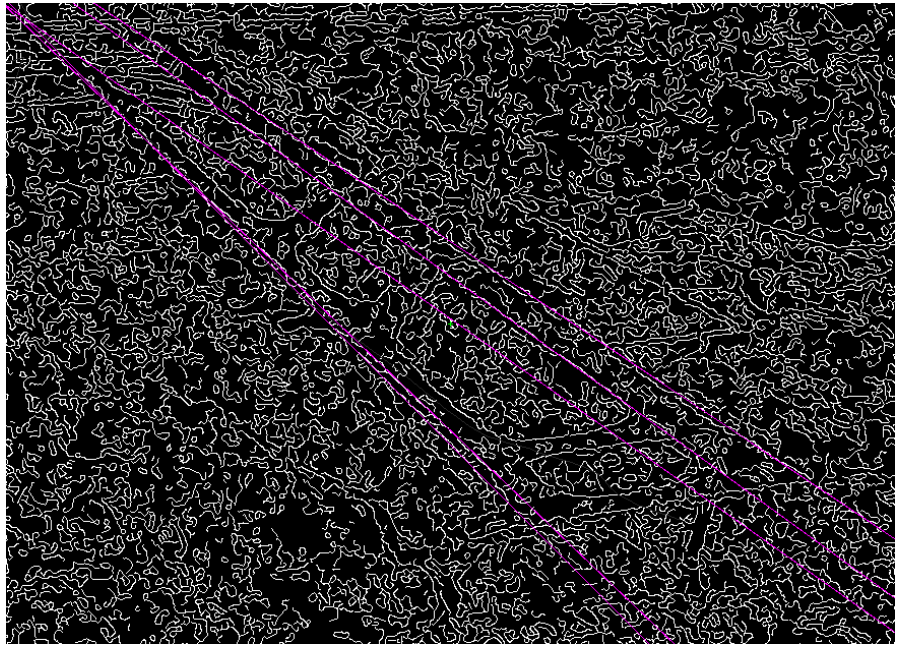
Consider a very noisy image of a runway – the noise is both Gaussian and extremal (salt and pepper); the Canny edge image is on the right





The Hough plugin finds the right runway reliably and finds one side of the left runway – noise is still an issue but the recognition is better than one might expect

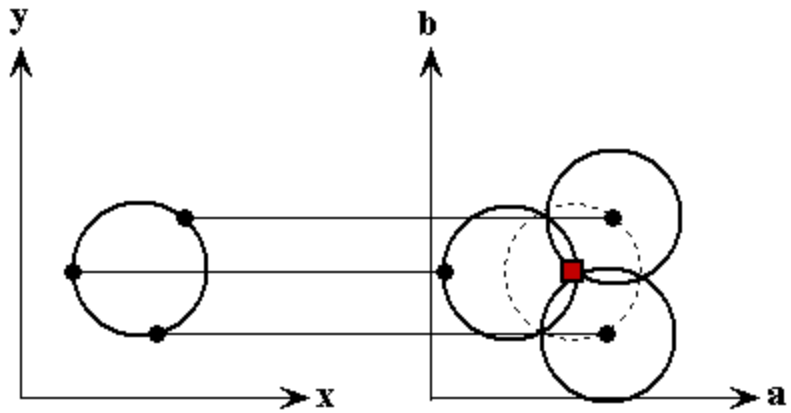




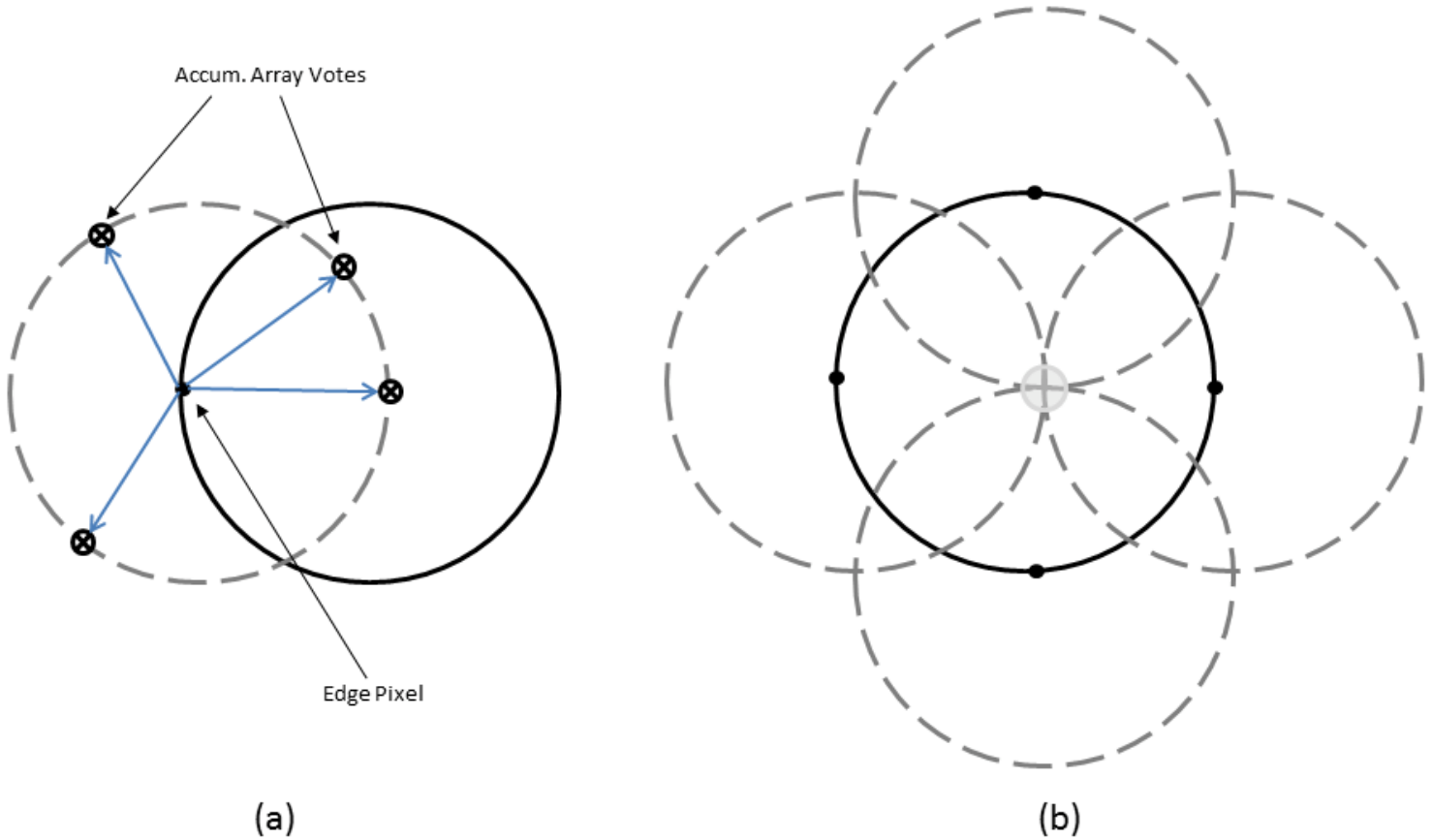
```
0: HoughLine <angle = 2.332, radius = 75.811, count = 914>
1: HoughLine <angle = 2.356, radius = 84.235, count = 880>
2: HoughLine <angle = 2.197, radius = -37.906, count = 867>
3: HoughLine <angle = 2.160, radius = -67.388, count = 855>
4: HoughLine <angle = 2.184, radius = -4.212, count = 815>
```

Consider the case where we are trying to find circles of radius *r* in a given image

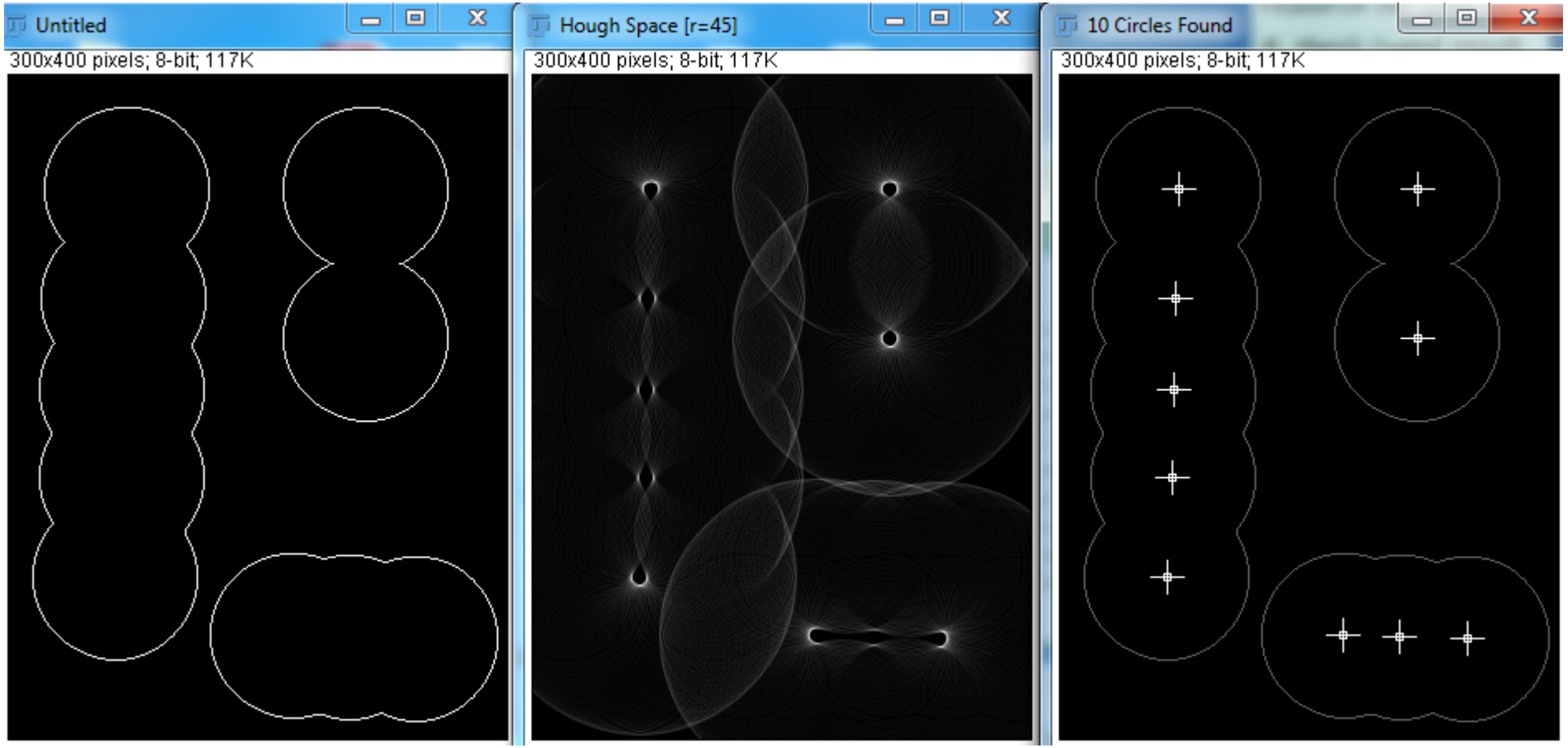
- By using a Hough approach, we can construct a parameter space containing the accumulations of all circles generated by the bright pixels in the image
- A circle is determined by a center point and a radius
- Since the radius is known, the only parameters needed are the x and y coordinates of the center of the circle



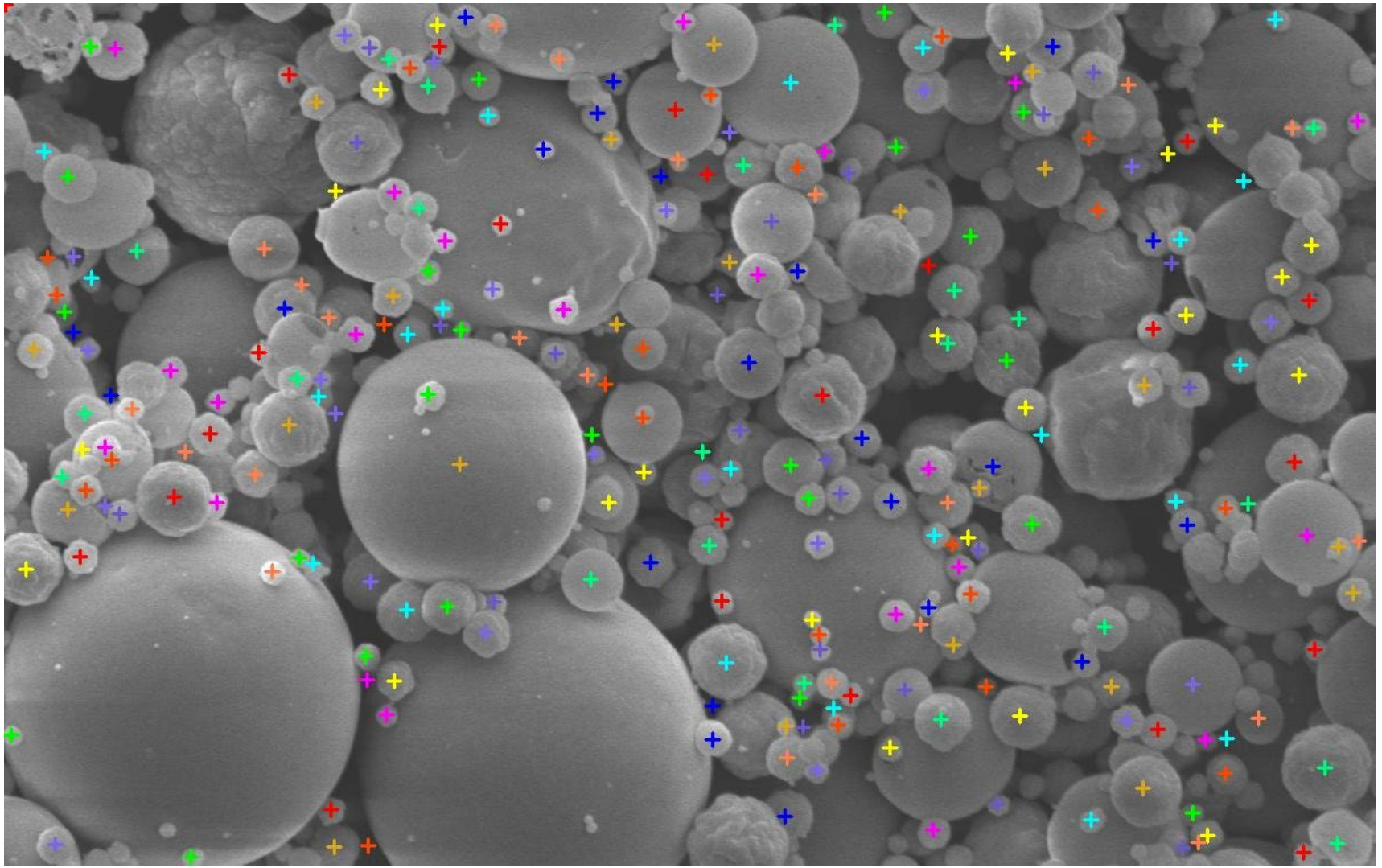
The centers of all the circles of radius *r* which could include a given point, when plotted together, *themselves* form a circle of radius r, centered on the point of interest



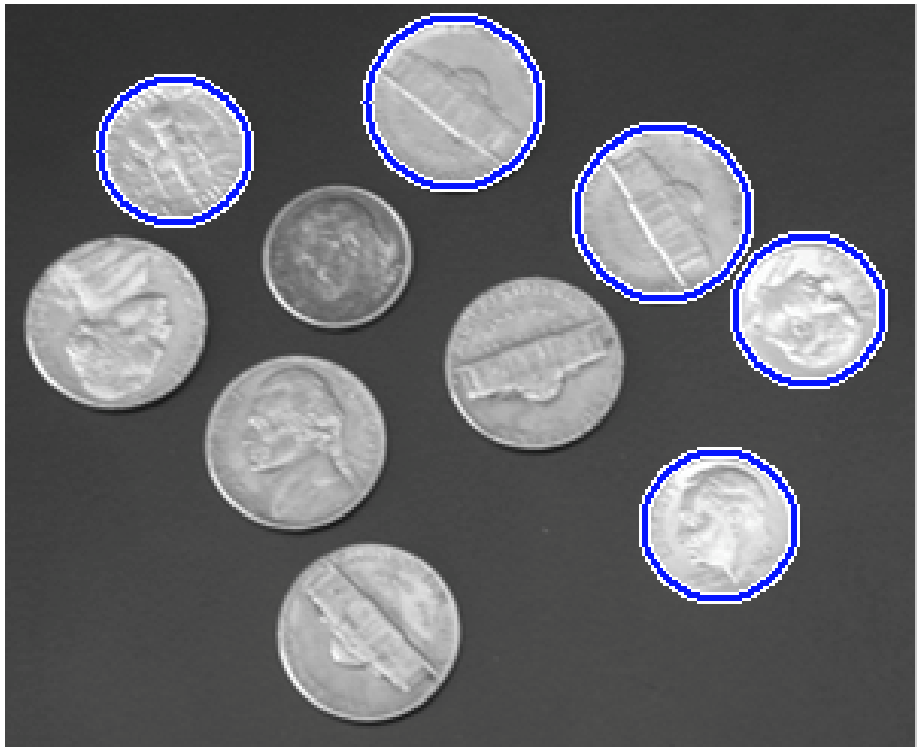
So, for circles of a given radius, the Hough Circle Transform will identify the center points of the "strongest" circles in the input image (typically a Canny edge image, as before)



By searching the 3D parameter space we identify the strongest (r, x, y) combinations



For applications where circles of various known radii must be found, several iterations of the 2D Hough circle transform can work well



There are extensions of the Hough transform to other parametrized shapes, and to more general shapes

- Ellipses are defined by five parameters:
  - x and y of the center
  - radii in the major and minor direction
  - orientation angle of the major direction

- The *Generalized Hough transform* doesn't rely on parametric expression of the shape
  - Assuming a single image shape, a reference point is chosen
  - An *R-table* is built from the edge points and the vectors to the reference
  - By looking at the maxima, we can find the likely center point of the shape in the image
  - An *accumulator table* is built from the edge points in a test image and the R-table entries
  - A location with high count in the accumulator is likely the location of the reference point of an identified object

The Generalized Hough transform

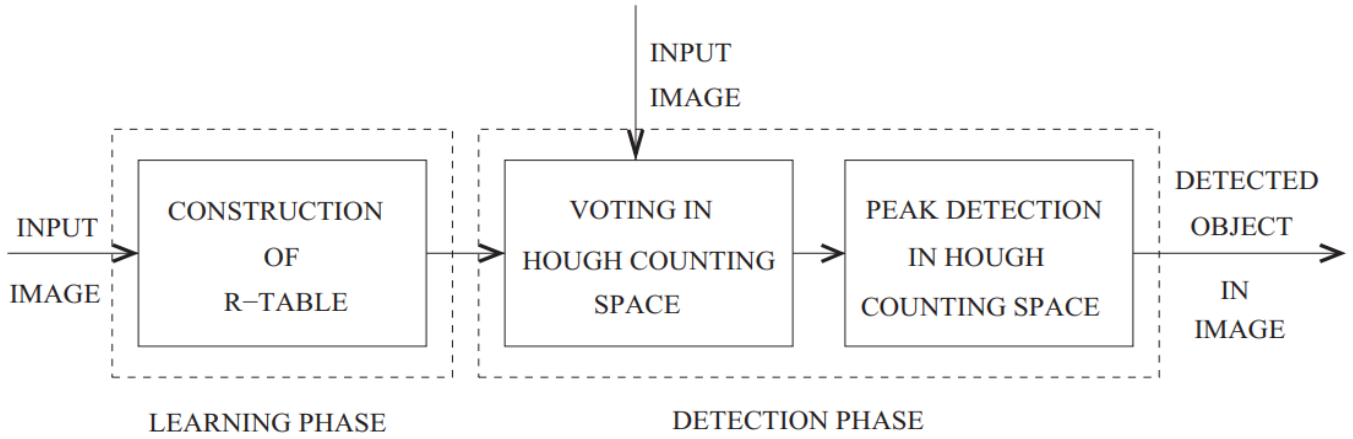


Fig. 5. Block diagram of GHT.

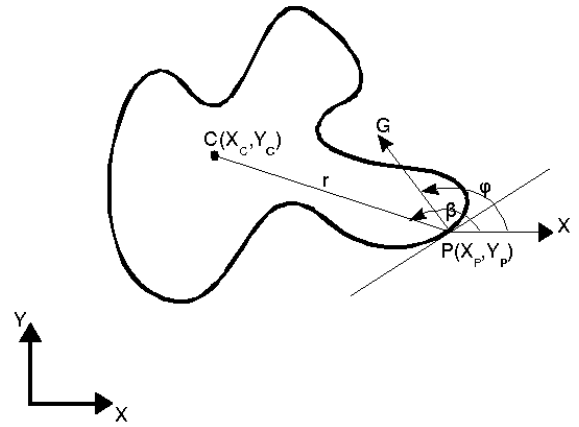


Table 2. R_Table corresponding to character ث

Orientation in degree (θ)	List of offsets pairs
0	(13,-6) (11,-6) (8,-9) (8,-10) (7,-10).....
18	(0,-11) (-1,-10)
20	(-6,-9)
21	(-7,-8)
33	(0,-10) (-4,-9) (-6,-8)
45	(-5,-8) (-7,-7) (-8,-5) (-8,-6) (-8,-8)...
342	(10,-9) (9,-10) (6,-10) (5,-11)

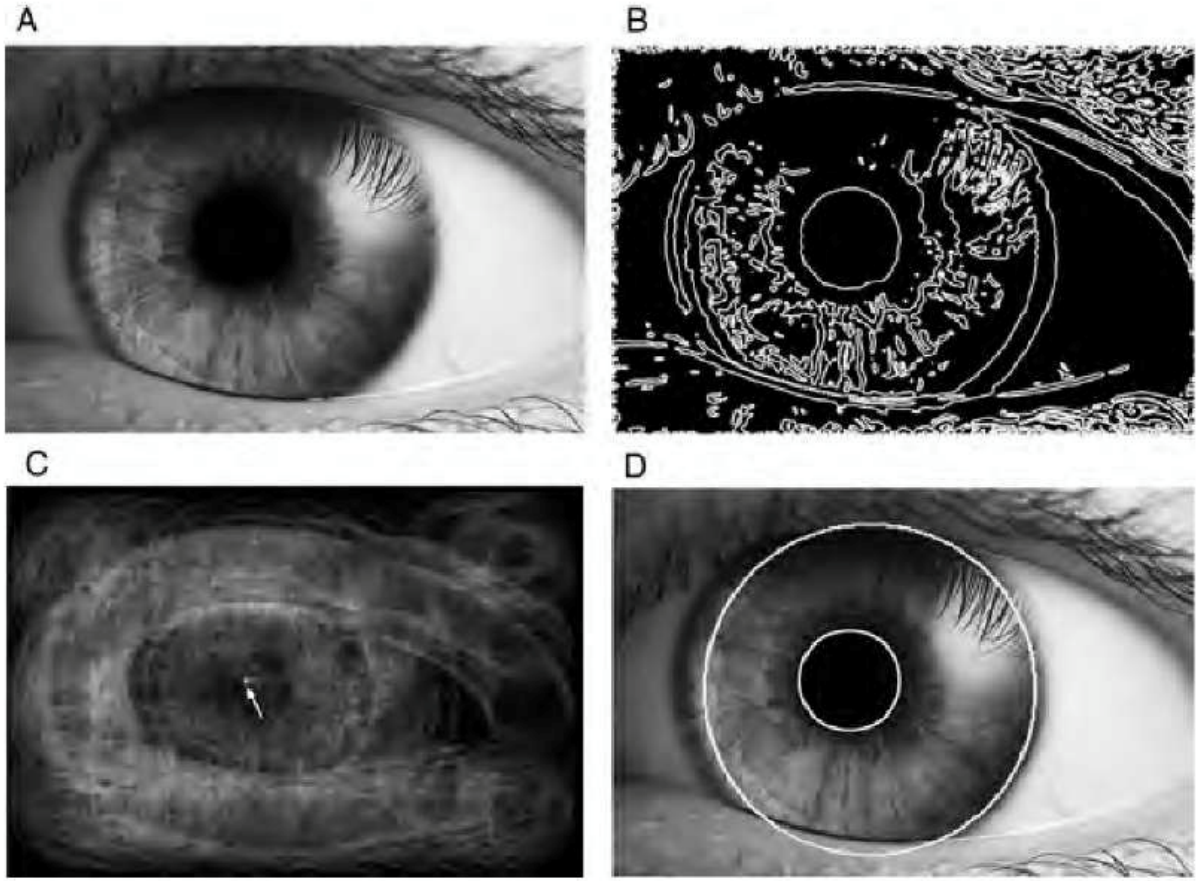
- Form an Accumulator array to hold the candidate locations of the reference point
- For each point on the edge
  - Compute the gradient direction and determine the row of the R-Table it corresponds to
  - For each entry on the row calculate the candidate location of the reference point

$$x_c = x_i + r\cos\theta$$

$$y_c = y_i + r\sin\theta$$

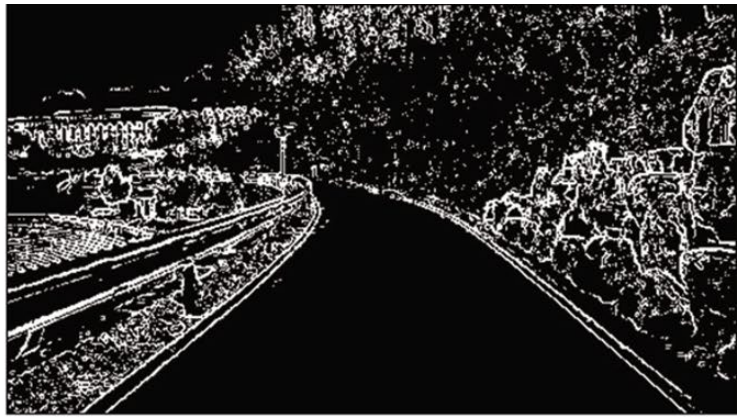
  - Increase the Accumulator value for that point
- The reference point location is given by the highest value in the accumulator array

Applications of the Hough transform



Applications of the Hough transform

(a) Input image (480×270)



(b) Edge of input image

(c) Line detection (480×480 HS)

(d) Line detection (120×120 HS)

Applications of the Hough transform

Today's Objectives

The Hough Transform
- General concept
- Parameter space
- Steps in the process
- ImageJ Hough implementation
- Some examples

Circular Hough Transform
- concept
- implementation

Generalized Hough Transform