HOUGH TRANSFORM

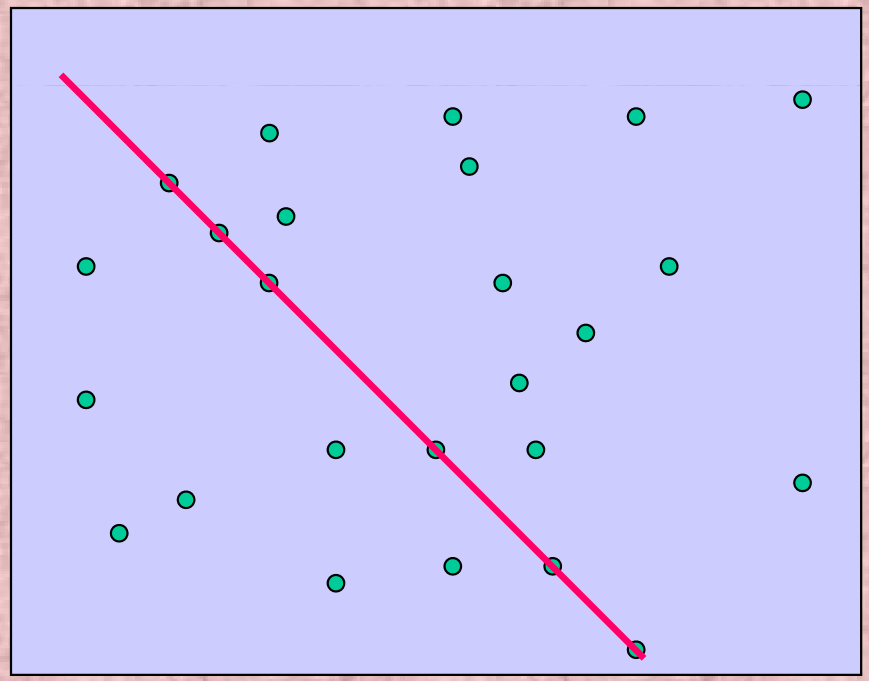
CS 6350

C V

HOUGH TRANSFORM

The problem:

Given a set of points in 2-D, find if a sub-set of these points, fall on a LINE.



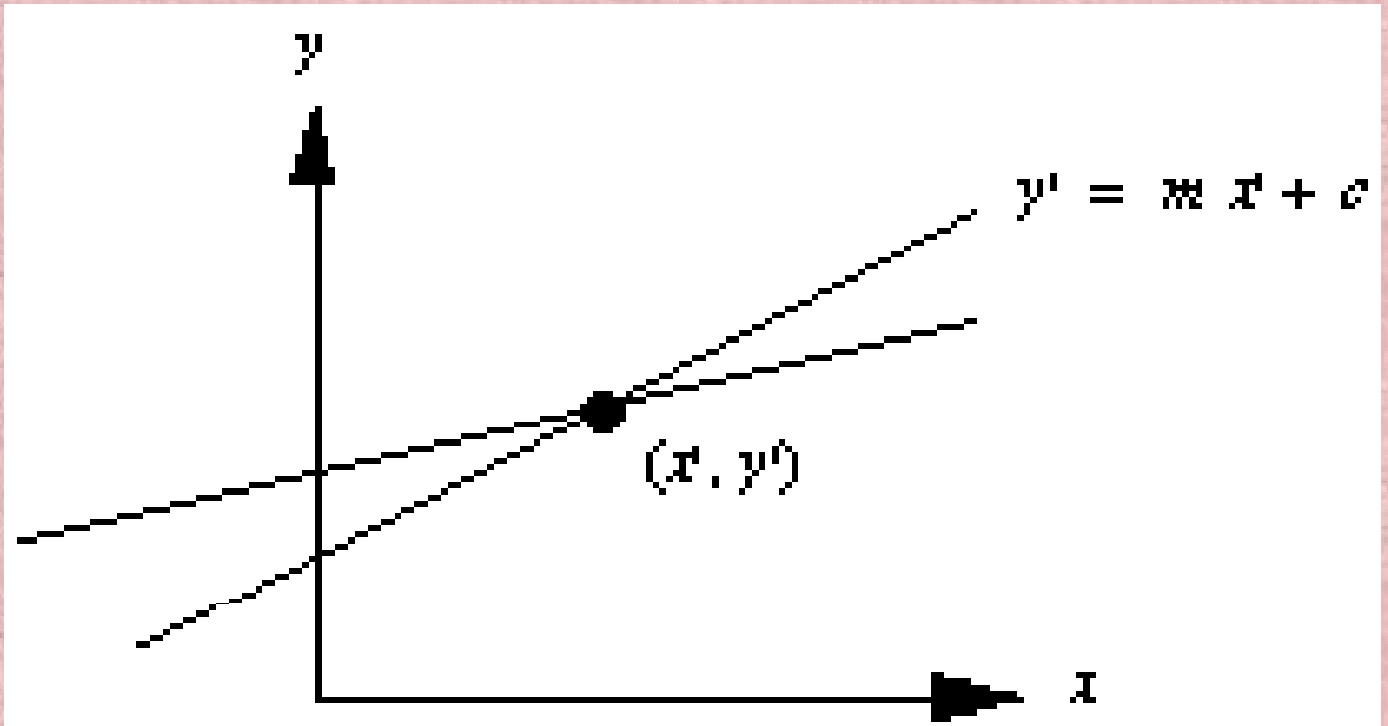
Hough Transform

One powerful global method for detecting edges (lines and parametric curves) is called the *Hough transform*.

Let us suppose that we are looking for straight lines in an image. If we take a point (*x*',*y*') in the image, *all* lines which pass through that pixel have the form

$$y' = m\,x' + c$$

for varying values of *m* and *c*. See Figure below:

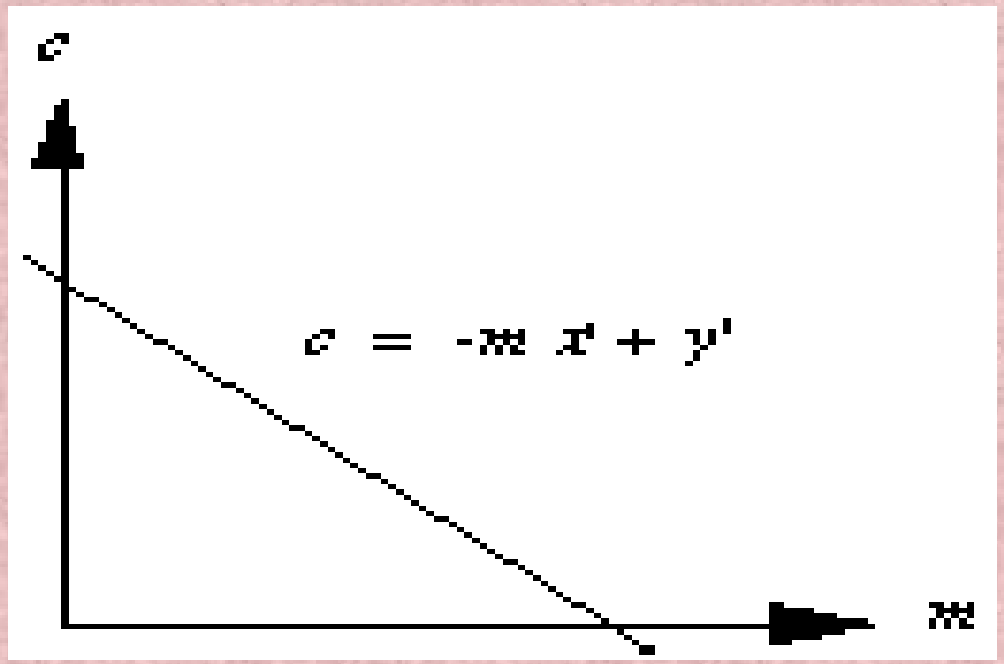


However, this equation can also be written as

$$c = -x' m + y'$$

where, we now consider x' and y' to be constants, and m and c as varying.

This is a straight line on a graph of c against m as shown in Figure below:

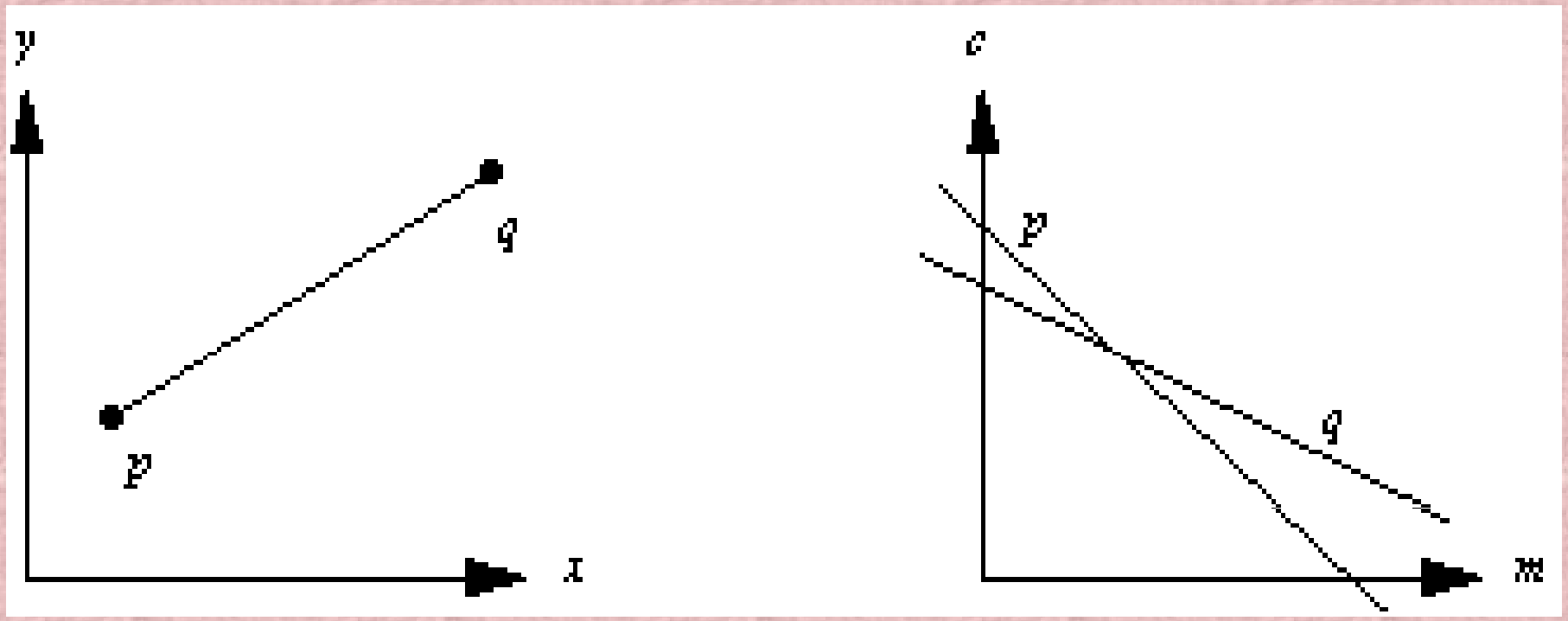


Each different *line* through the point (x', y') corresponds to one of the *points* on the line in (m, c) space (termed as *Hough space*).

Now consider two pixels *p* and *q* in (*x*, *y*) space which lie on the same line.

- For each pixel, all of the possible lines through it are represented by a single line in (*m*, *c*) space.

- Thus the single line in (*x*, *y*) space which goes through both pixels lies on the intersection of the two lines representing *p* and *q* in (*m*, *c*) space, as illustrated by Figure



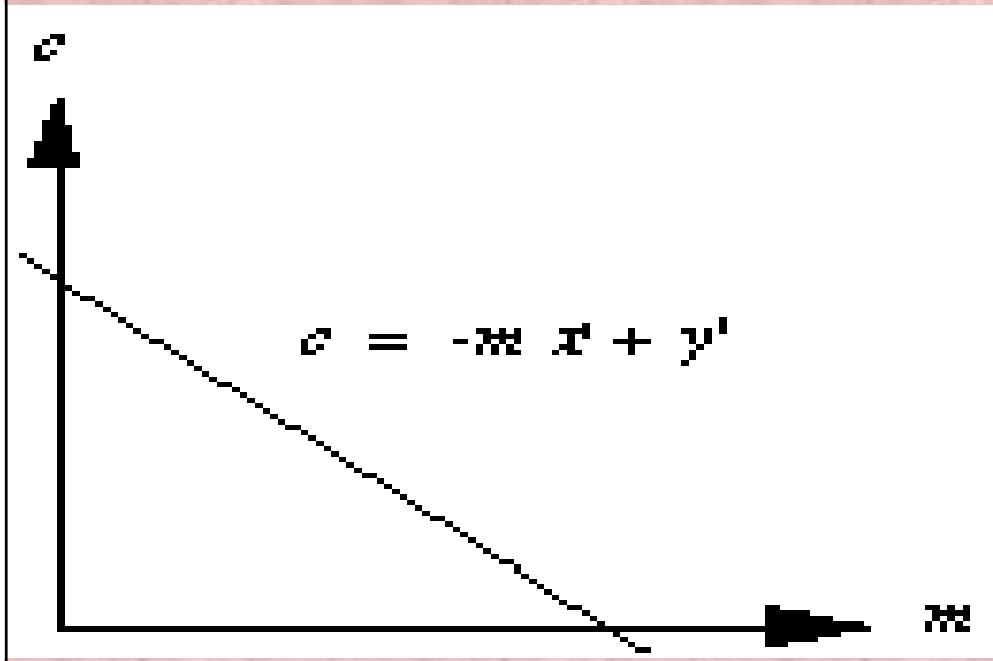
Identify the point <--> line correspondences interchangeably between these two spaces: Spatial coordinate and parametric (Hough).

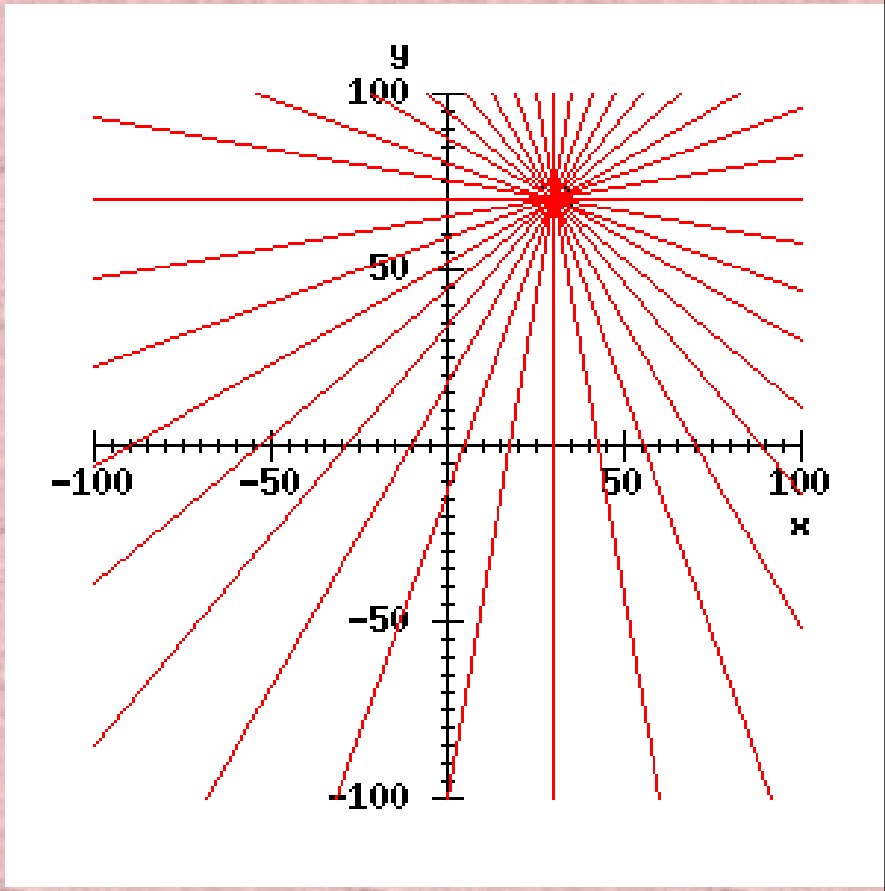
However, this equation can also be written as

$$c = -x' m + y'$$

where, we now consider x' and y' to be constants, and m and c as varying.

This is a straight line on a graph of c against m as shown in Figure below:





Each different *line* through the point (x', y') corresponds to one of the *points* on the line in (m, c) space (termed as *Hough space*).

Taking this one step further,

- ***All* pixels which lie on the same line in (*x*, *y*) space are represented by lines which all pass through a single point in (*m*, *c*) space.**

- **The single point through which they all pass gives the values of *m* and *c* in the equation of the line: $y = m x + c$.**

The steps to detect straight lines in an image:

1. **Quantize (*m*, *c*) space into a two-dimensional array *A* for appropriate steps of *m* and *c*.**

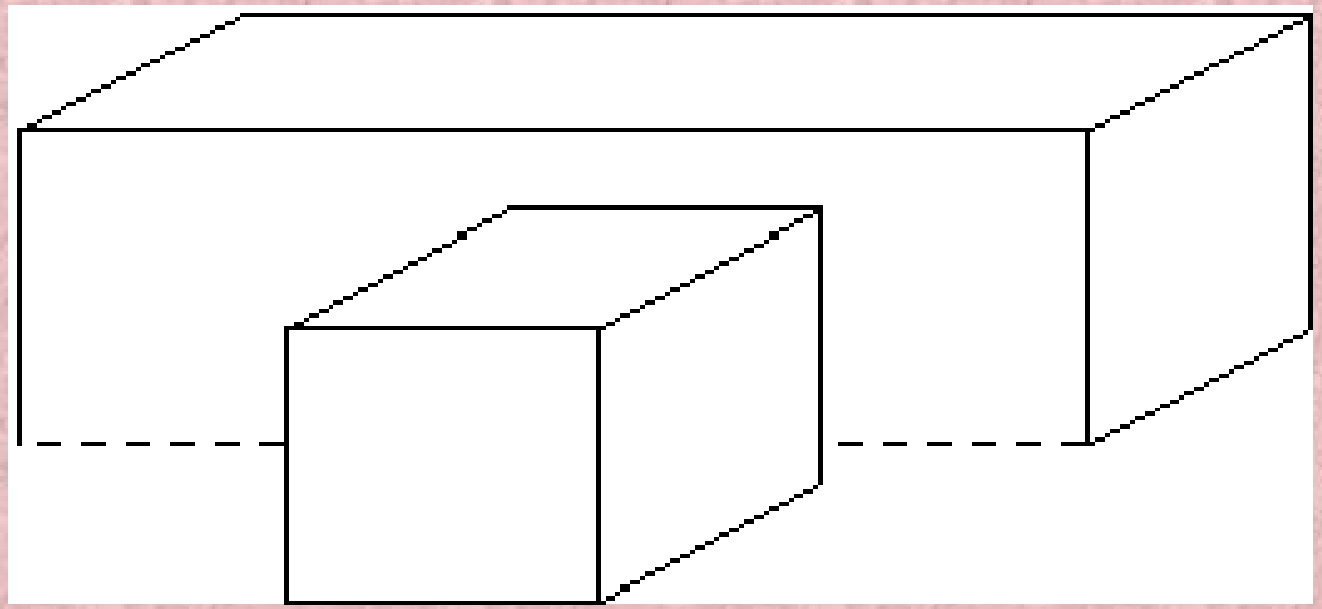
2. **Initialize all elements of *A*(*m*, *c*) to zero.**

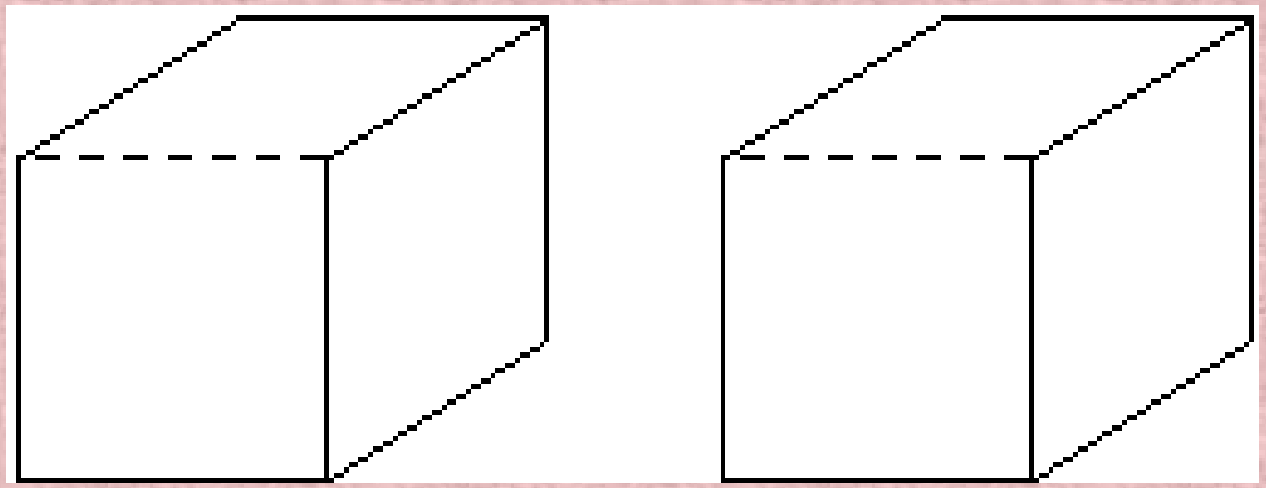
3. **For each pixel (*x*', *y*') which lies on some edge in the image, we add 1 to all elements of *A*(*m*, *c*) whose indices *m* and *c* satisfy $y' = m x' + c$.**

Search for elements of *A*(*m*, *c*) which have large values -- Each one found corresponds to a line in the original image.

One useful property of the Hough transform is that the pixels, which lie on the line, need not all be contiguous.

For example, all of the pixels lying on the two dotted lines in Figure below, will be recognized as lying on the same straight line. This can be very useful when trying to detect lines with short breaks in them due to noise, or when objects are partially occluded as shown below:





Drawback: **The figure above clearly demonstrates one disadvantage of the Hough transform method. It gives an *infinite line* as expressed by the pair of *m* and *c* values, rather than a finite *line segment* with two well-defined endpoints.**

One practical difficulty is that, the $y = m\,x + c$ form for representing a straight line breaks down for vertical lines, when *m* becomes infinite.

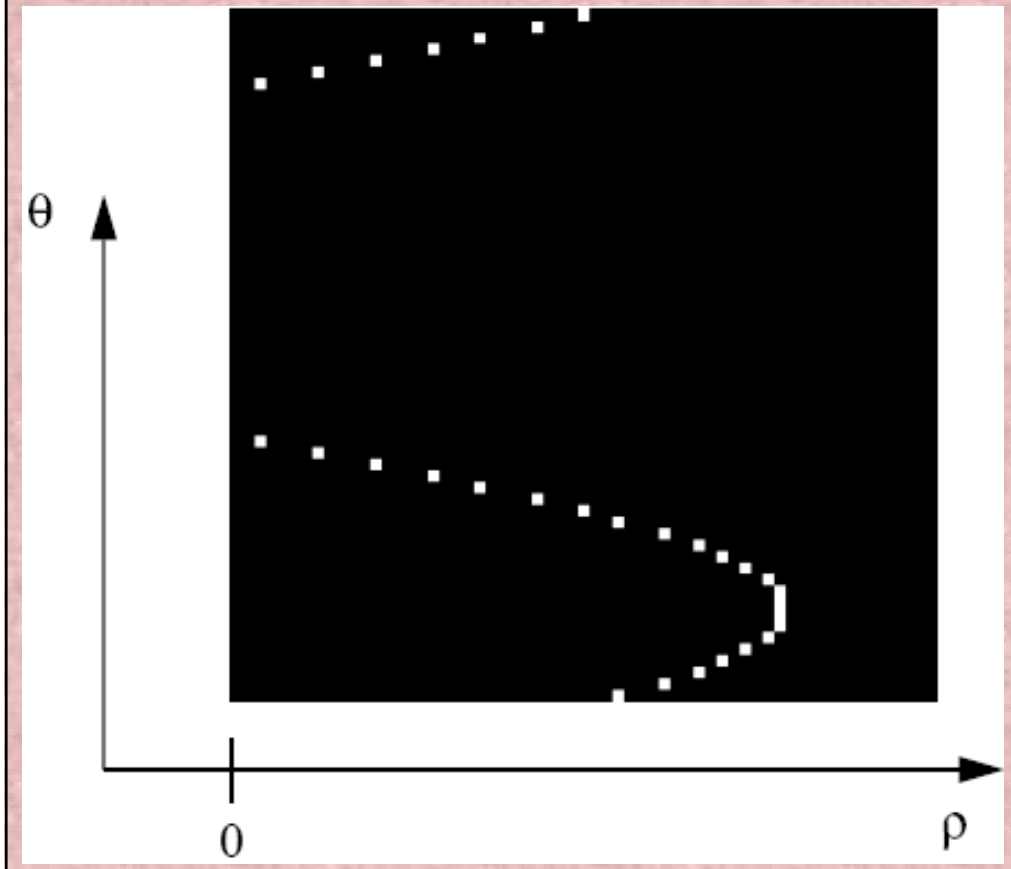
To avoid this problem, it is better to use the alternative formulation as given below:

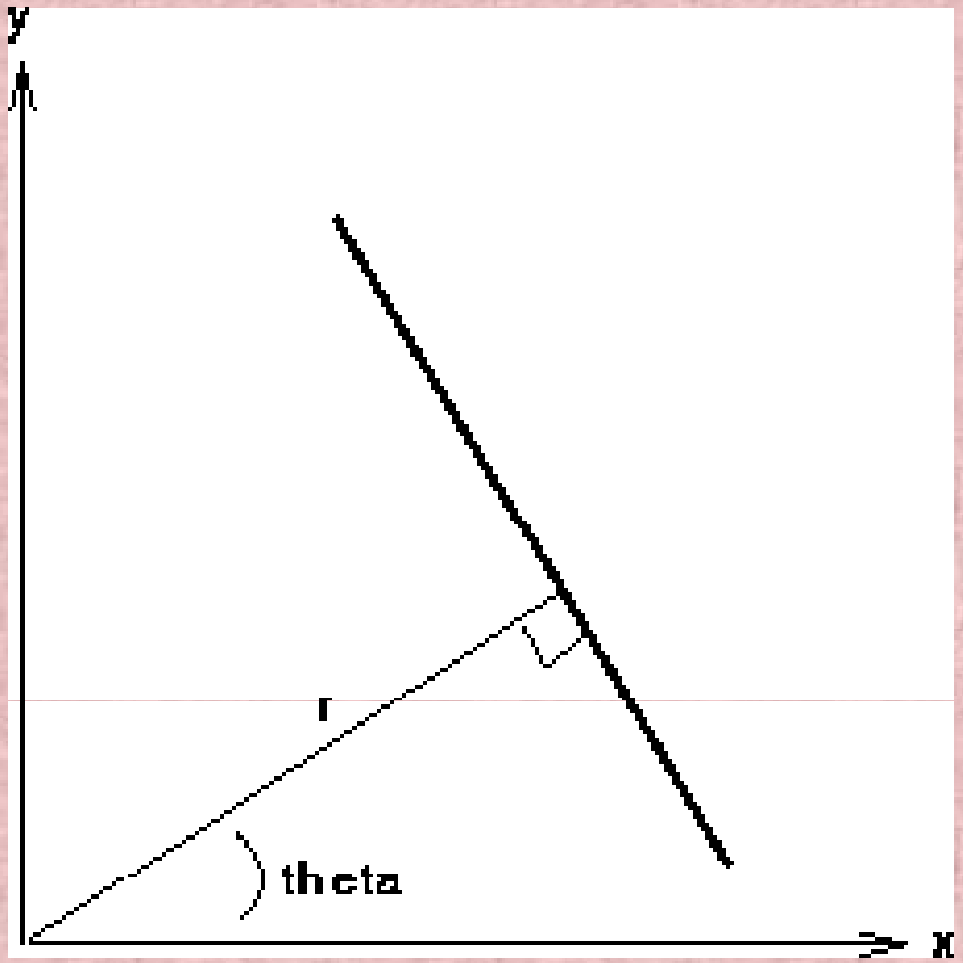
$$x \cos \theta + y \sin \theta = \rho$$

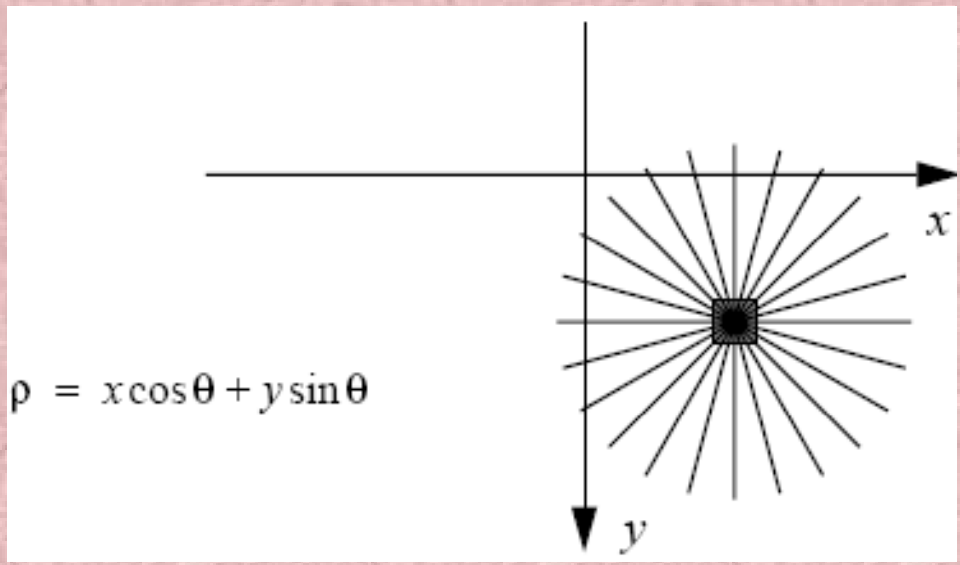
To avoid this problem, it is better to use the alternative formulation as given below:

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

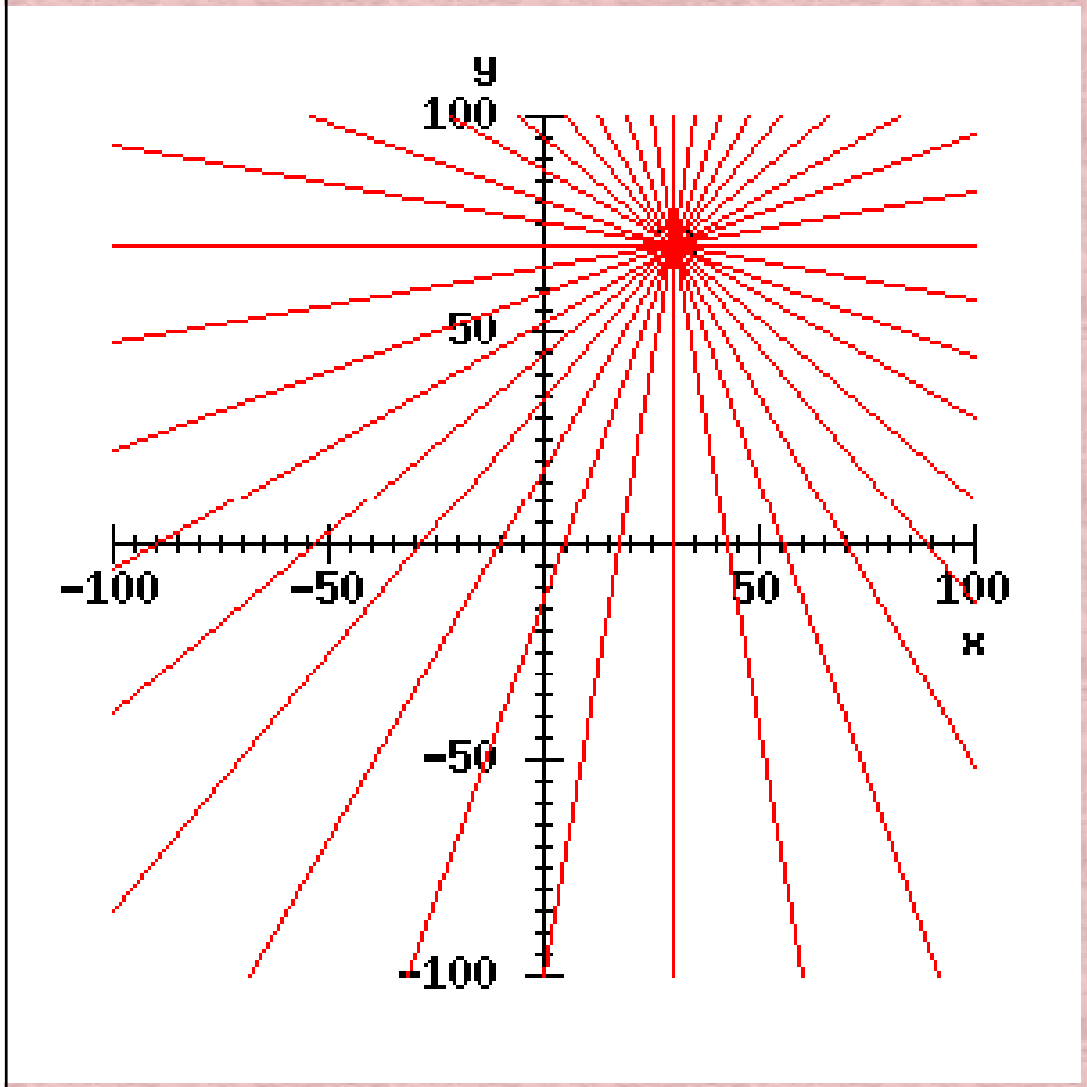
What is the output in the parametric $(\rho–\theta)$ space ??

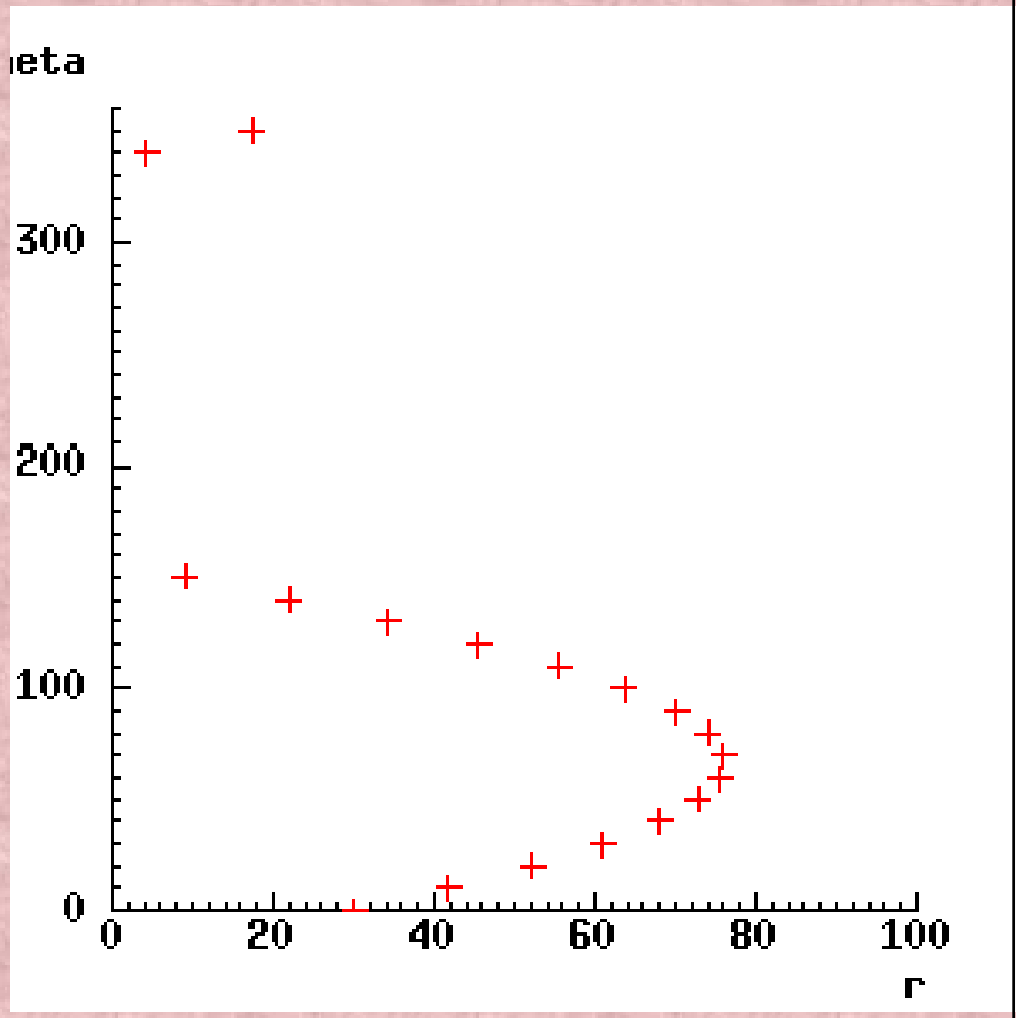


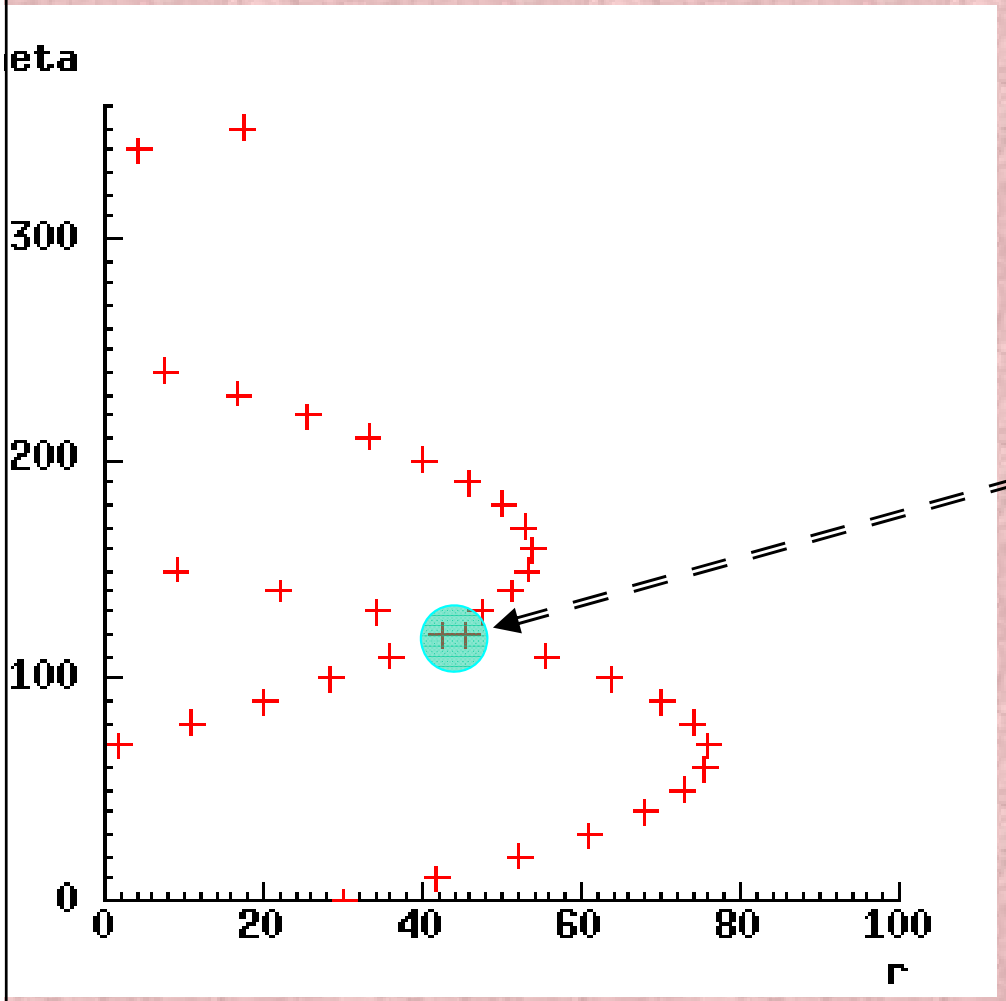


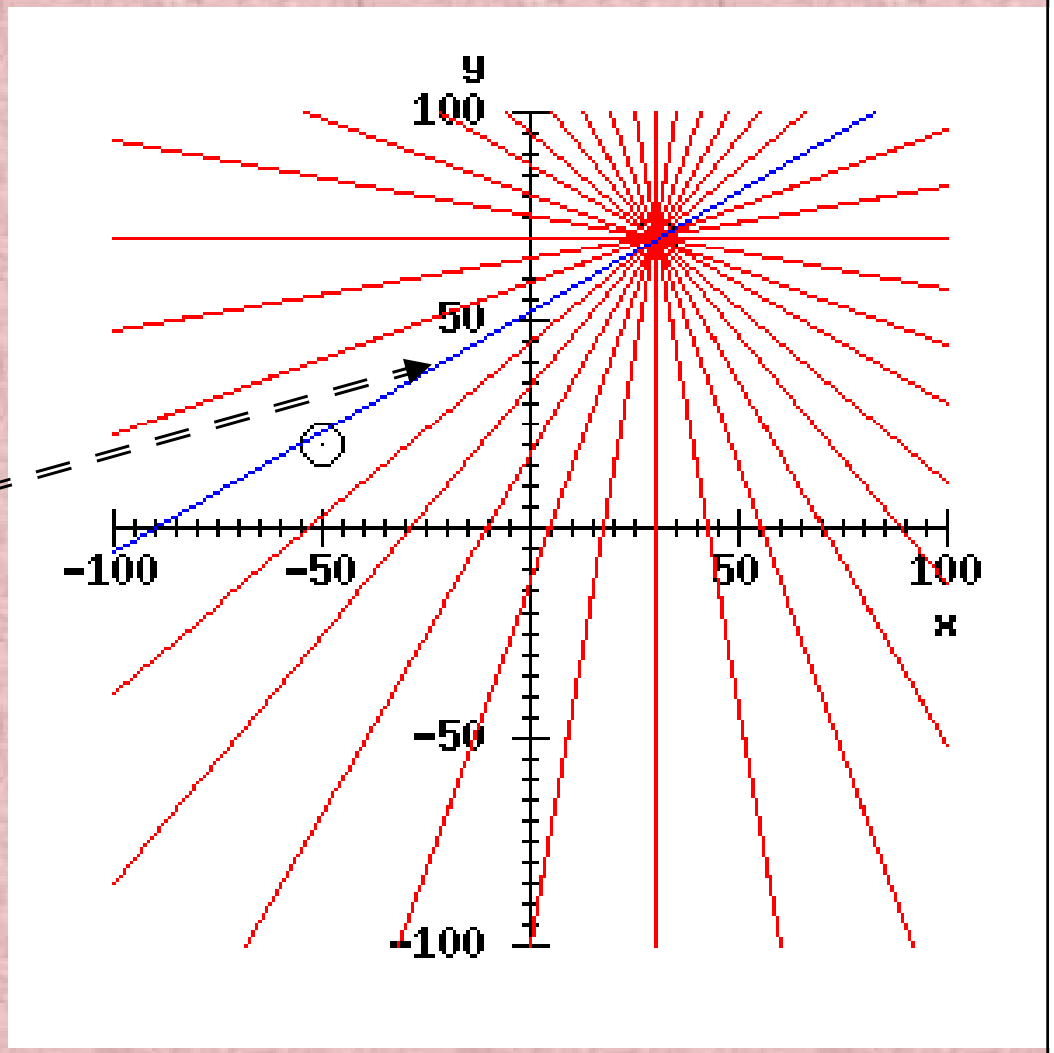


Note, however, that a point in (*x*, *y*) space is now represented by a *curve* in (ρ, θ) space rather than a straight line. Otherwise, the method is unchanged.

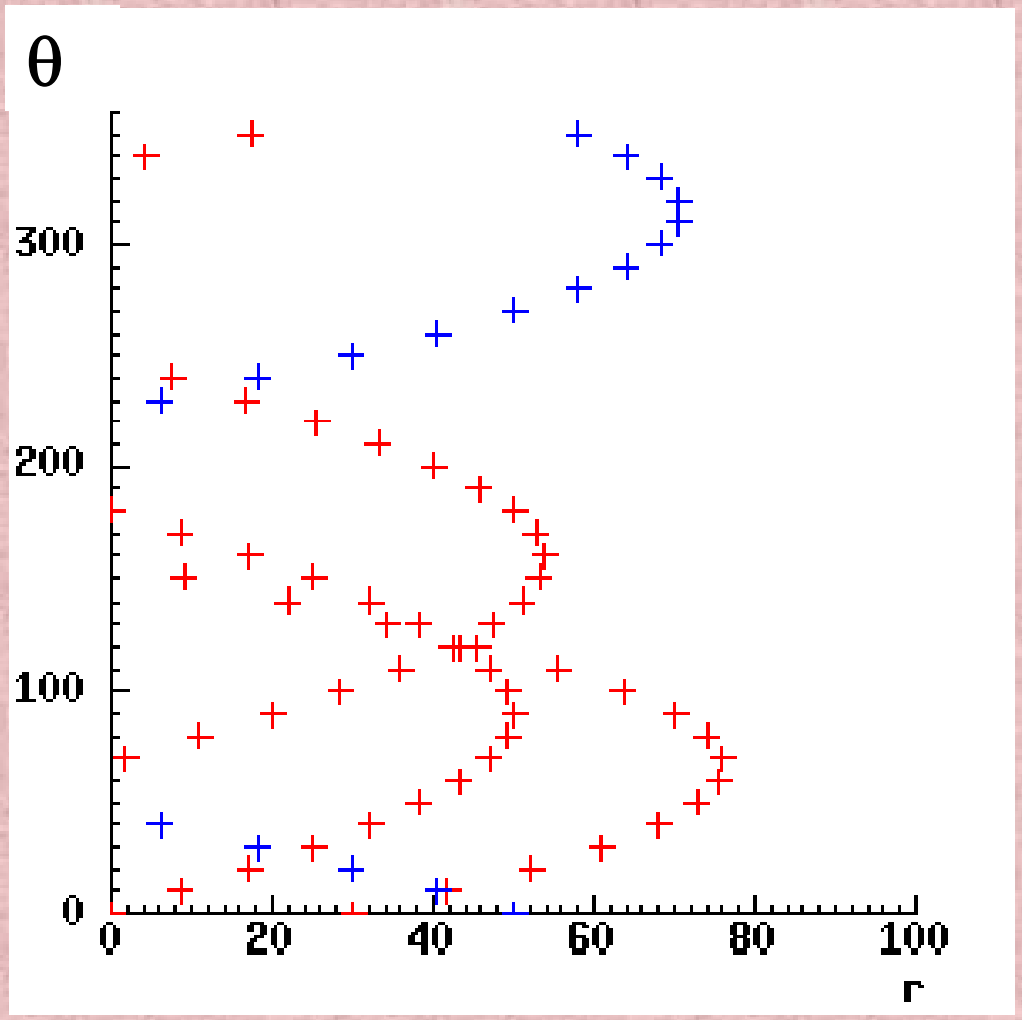




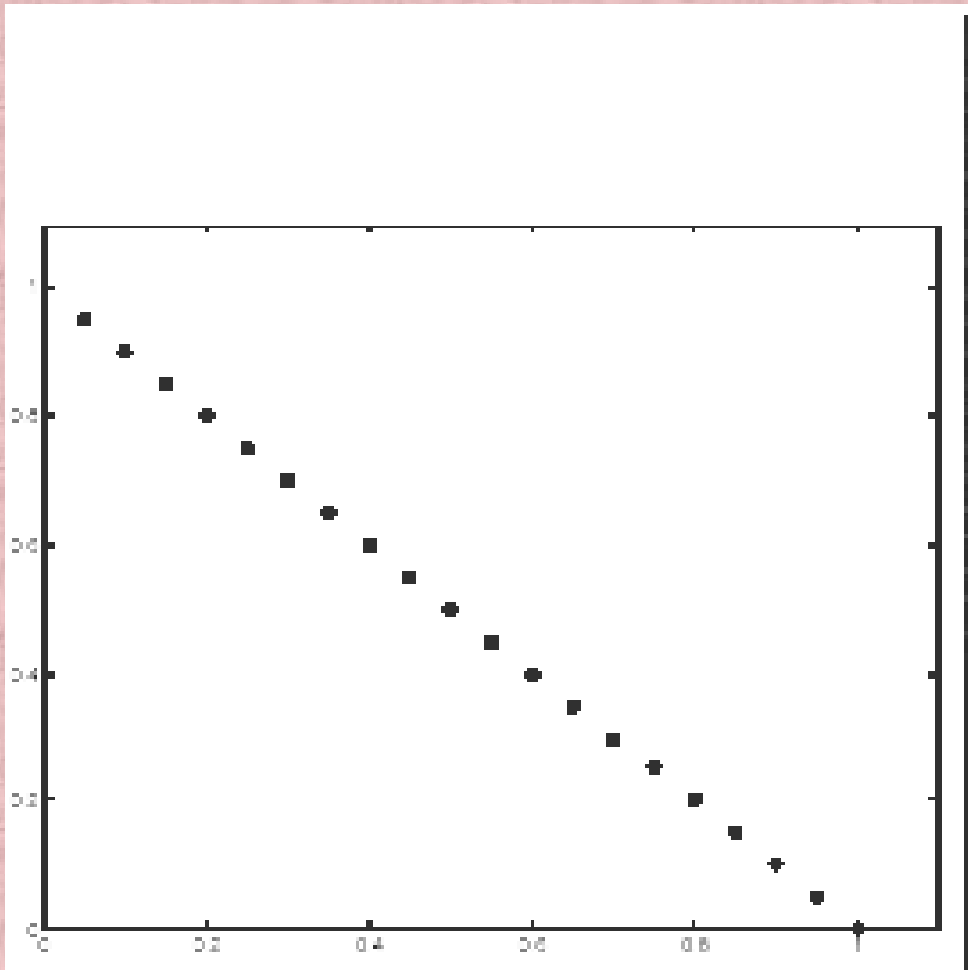




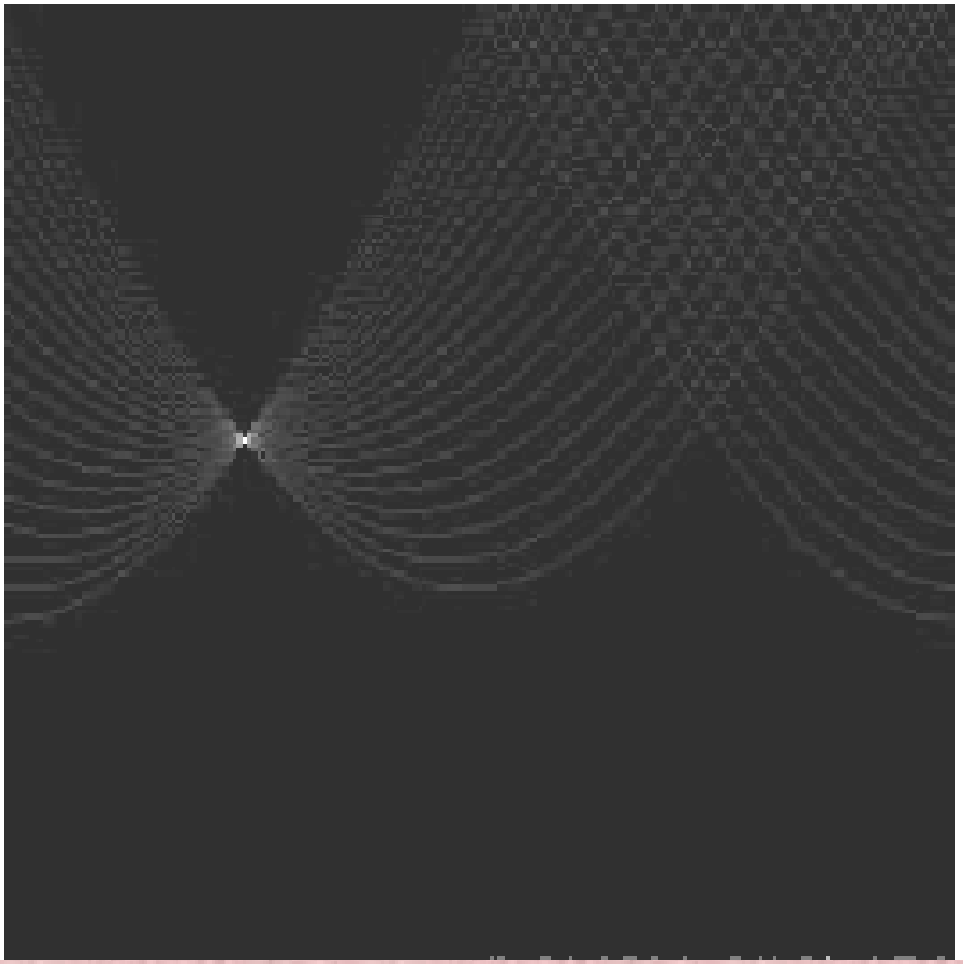
Two points in (*x*, *y*) space will now be represented by two different curves in (ρ, θ) space rather than straight lines.
The intersection corresponds to the parameters of the line in (x, y) space, formed by joining those two points.



Find similarity with [Radon Transform]



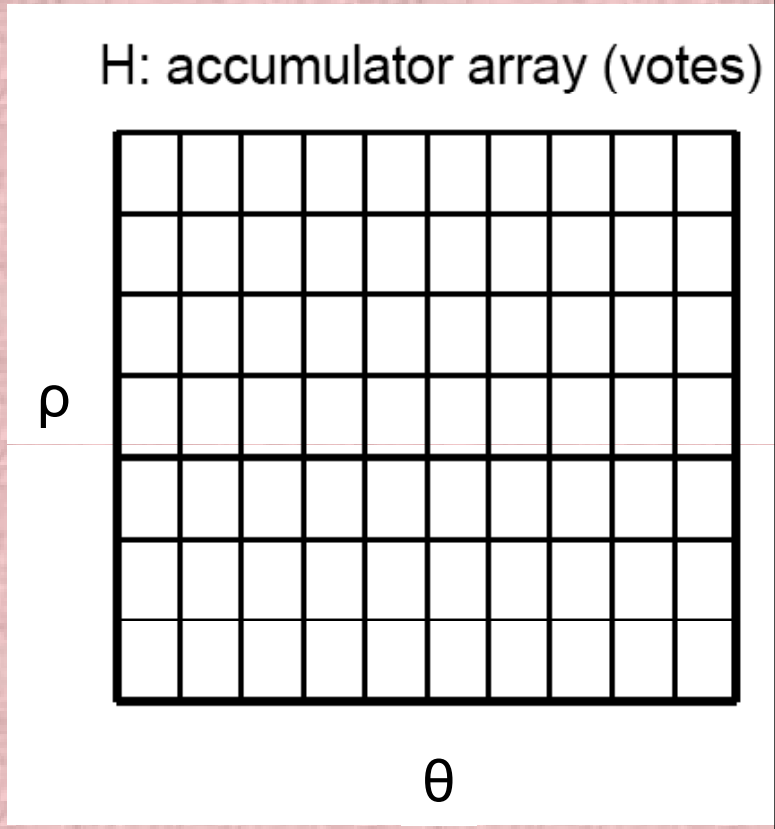
Point Features



Votes

Algorithm for HT

- Initialize all accumulator cells in H to Zeros
- For each edge pixel (x,y) in image
  For **θ = -90 to 90**
  $\rho = x \cos\theta + y \sin\theta$
  ++ $H(i_\theta, j_\rho)$;
  end
  end
- Find the value(s) of (θ, ρ)s, where $H(i_\theta, j_\rho)$s are local maxima.
  - **Detected line(s) in the image are obtained using:**
    $\rho_k = x \cos\theta_k + y \sin\theta_k$



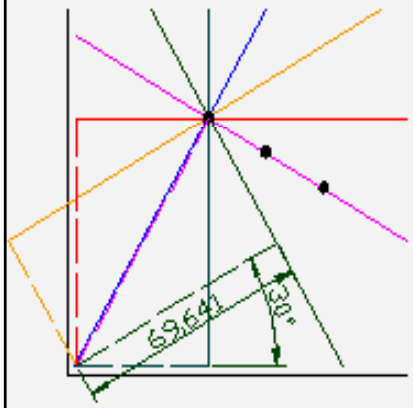
For each data point, a number of lines are plotted going through it, all at different angles. These are shown as solid lines.

For each solid line draw a line perpendicular to it from the origin. These are shown as dashed lines.

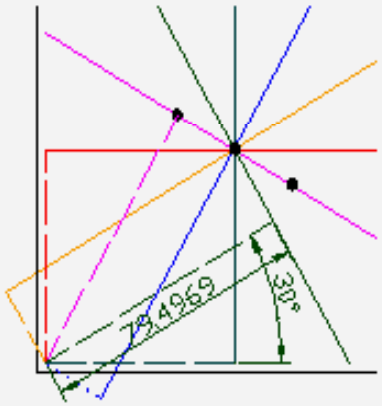
The length and angle of each dashed line is measured. The results are shown in tables. This is repeated for each data point.

A graph of length (R or Dist.) against angle, known as a Hough space graph, is then created.

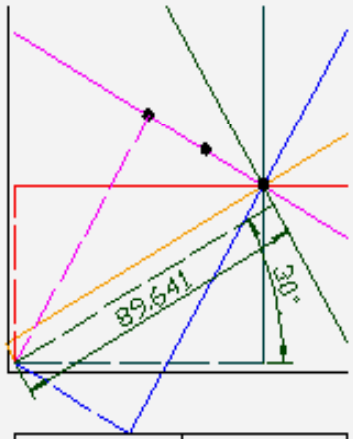
A case of implementation with three points:



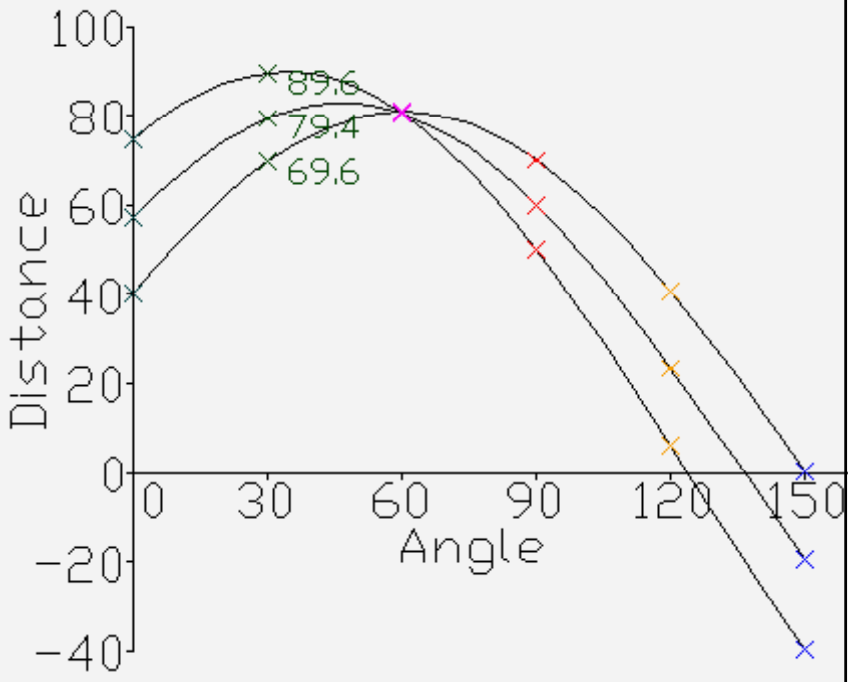
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

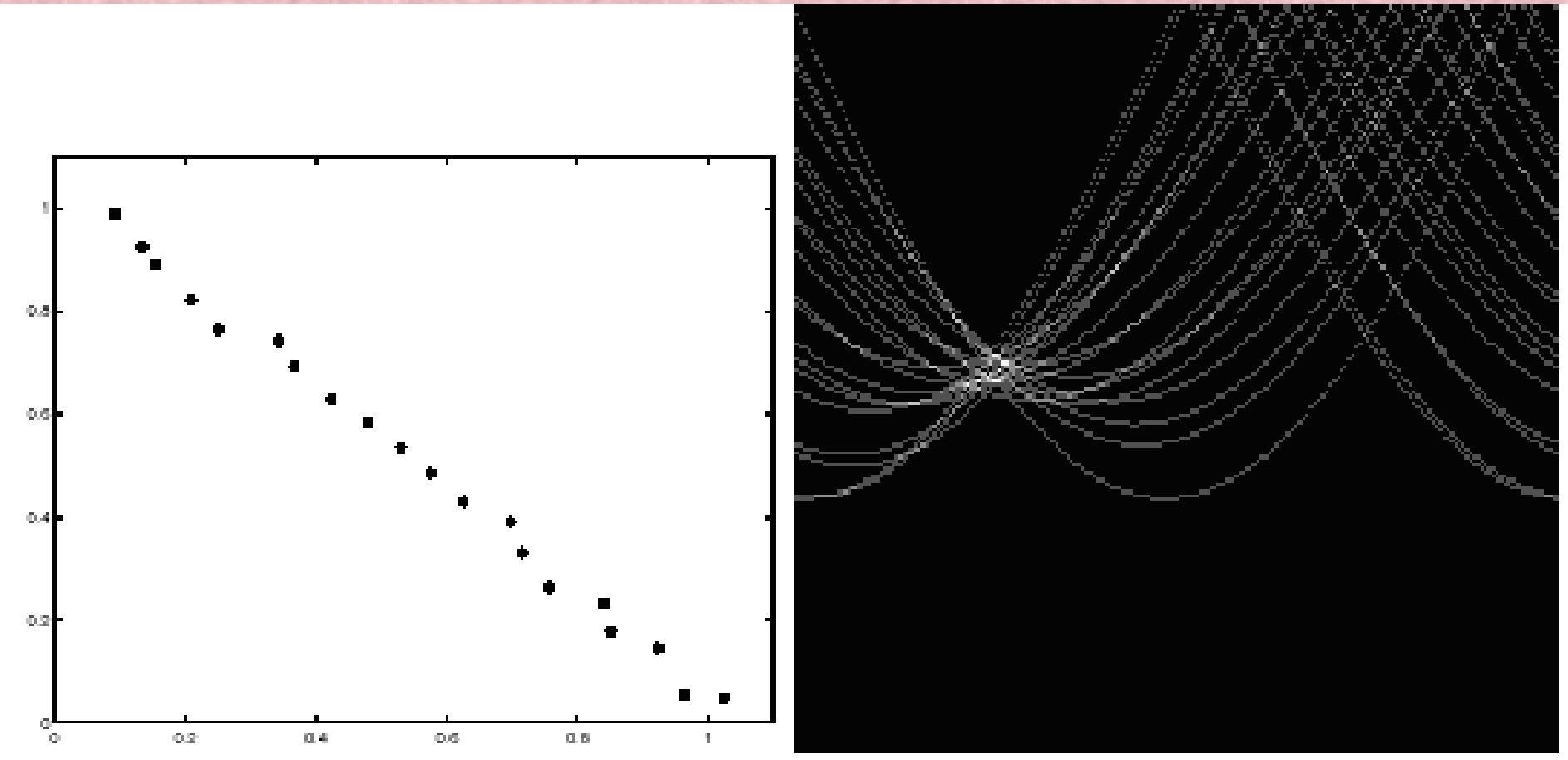


Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6



< -- Construct an **R-table** from here

Presence of noise

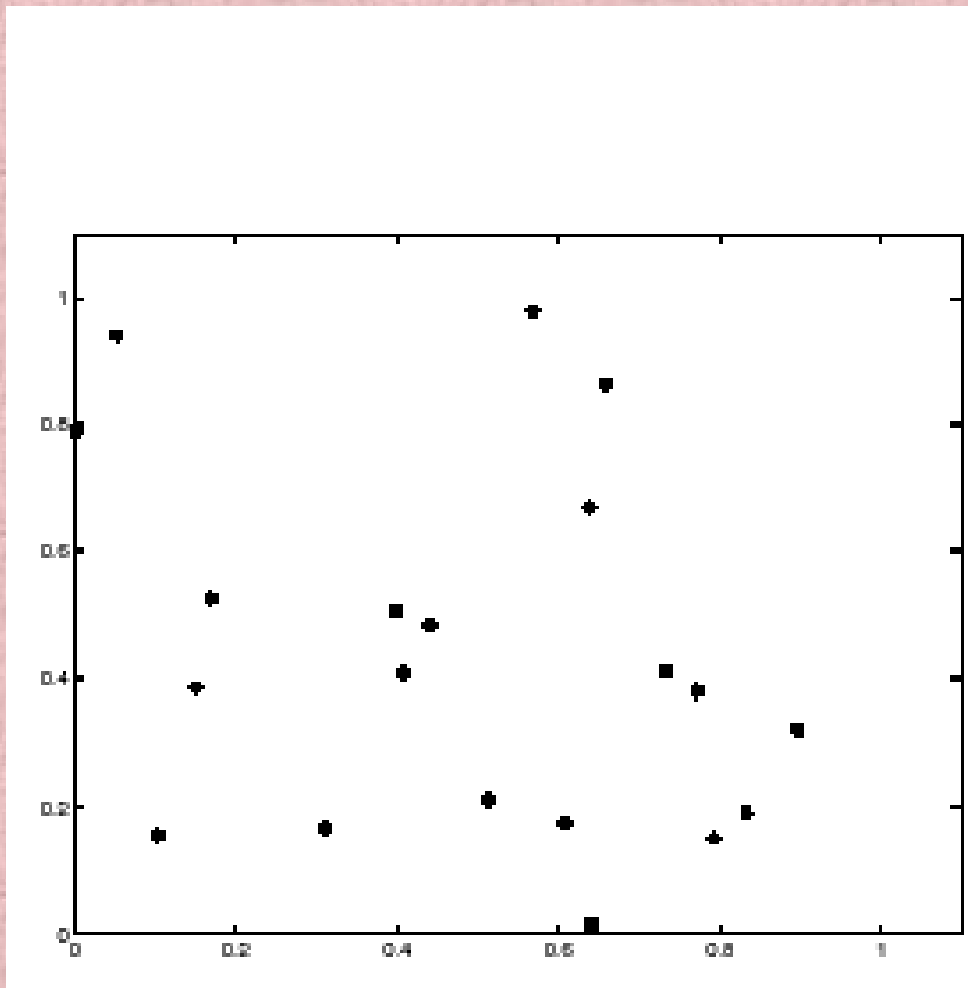


Features

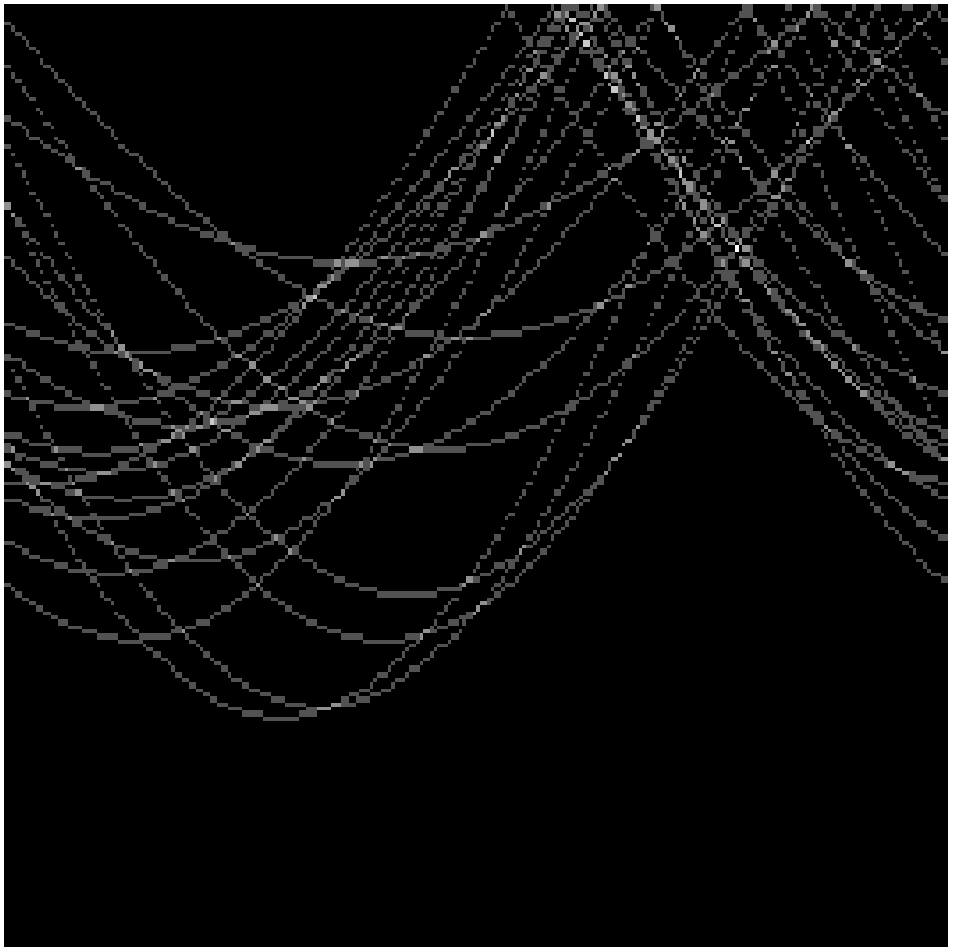
Votes

- Peak gets fuzzy and hard to locate

Random Noise



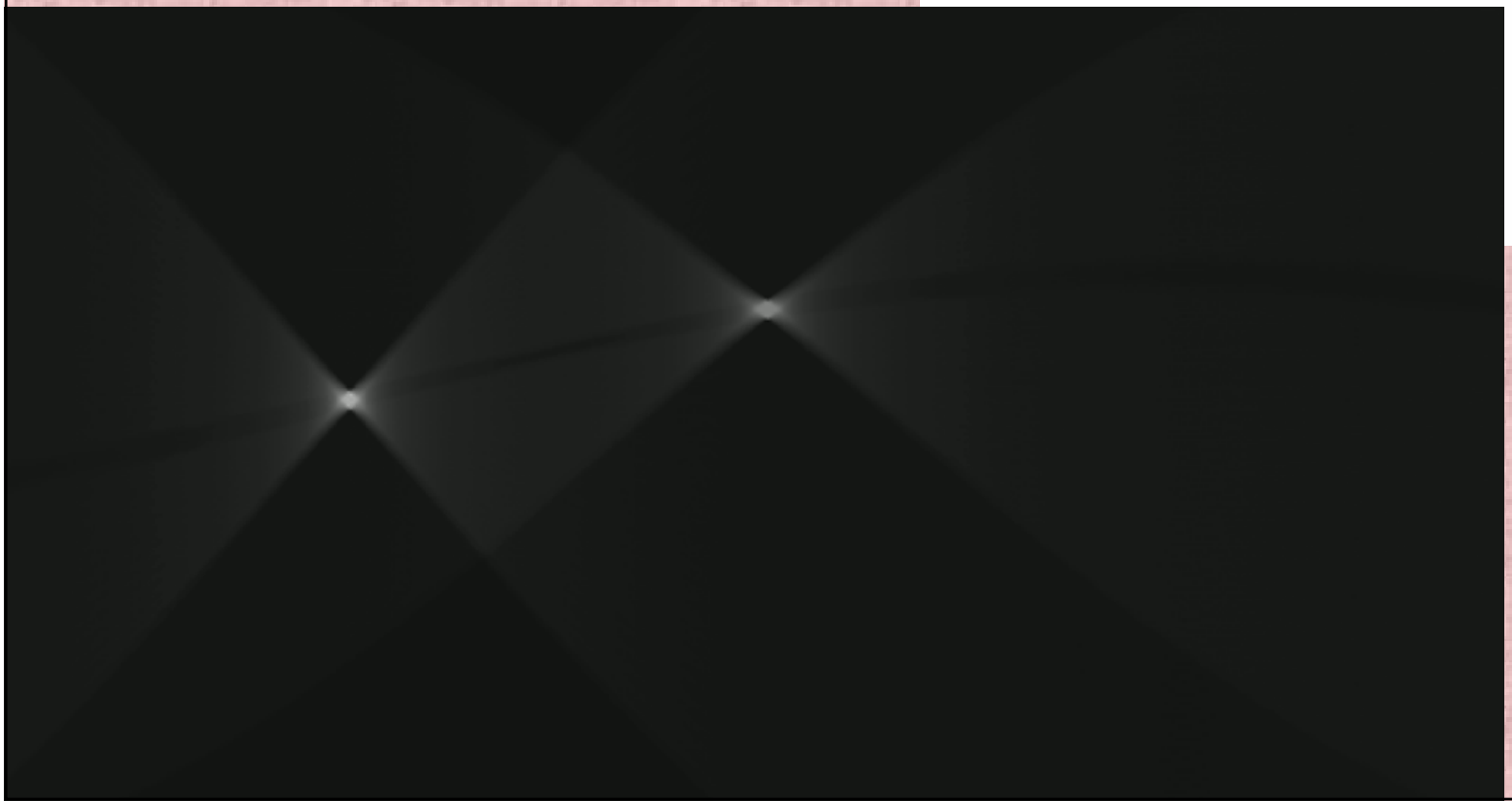
Features



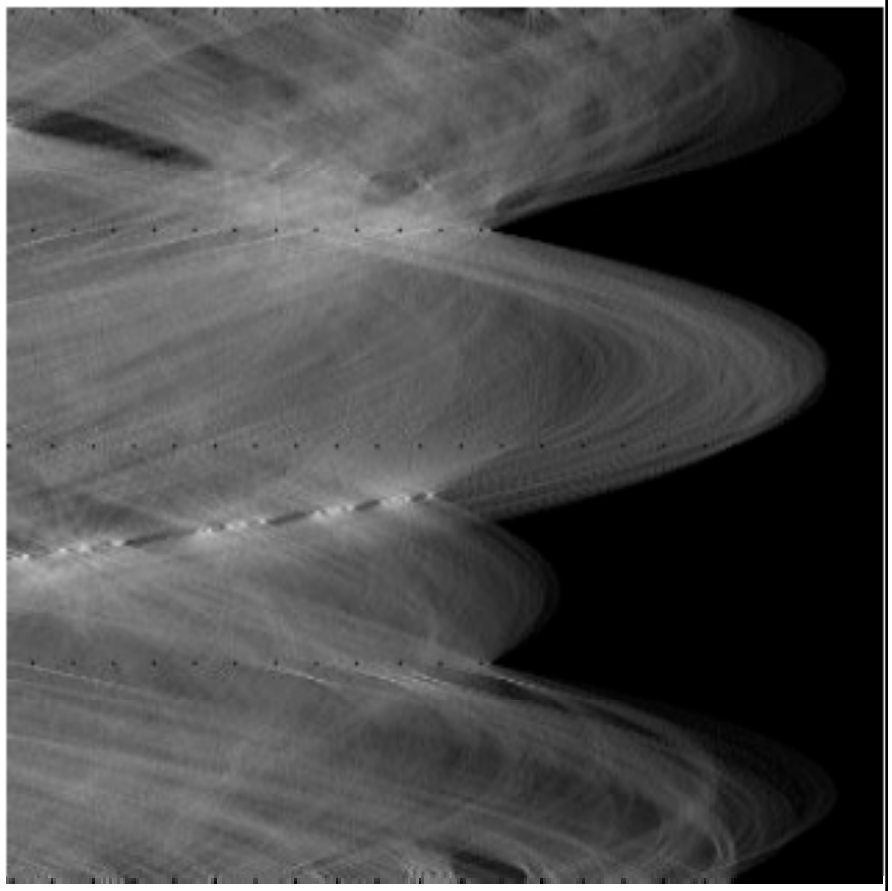
Votes

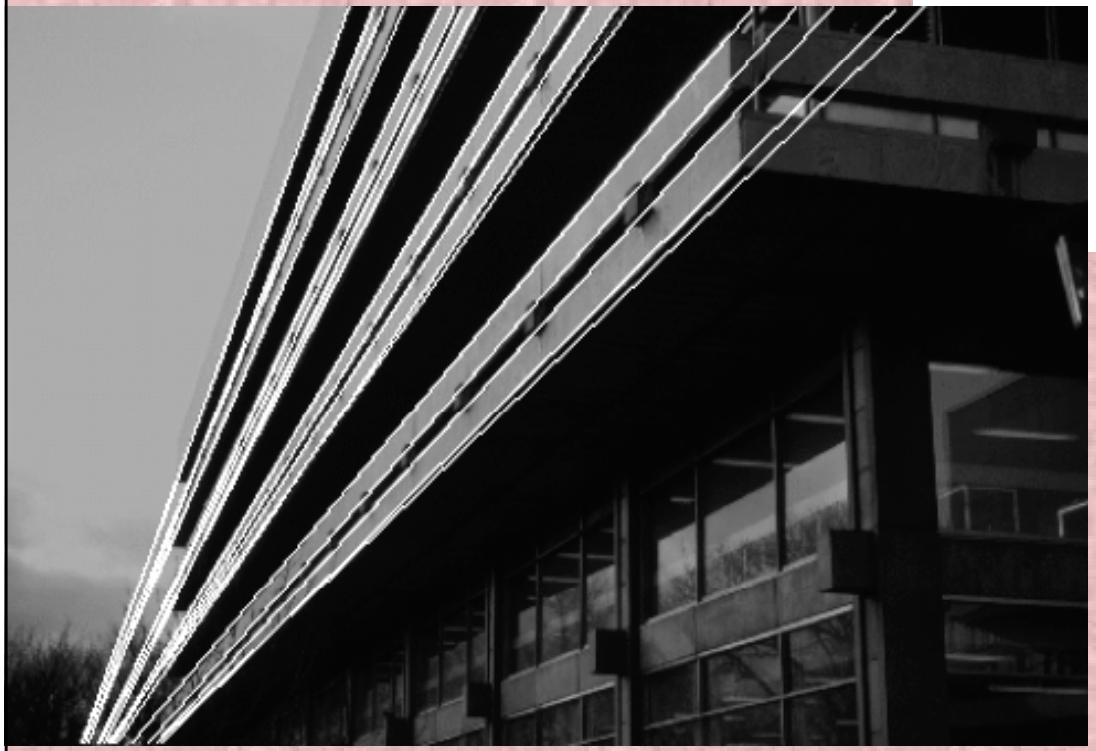
- **Noise produces spurious peaks in the accumulator array / R-table**

Case of two thick lines

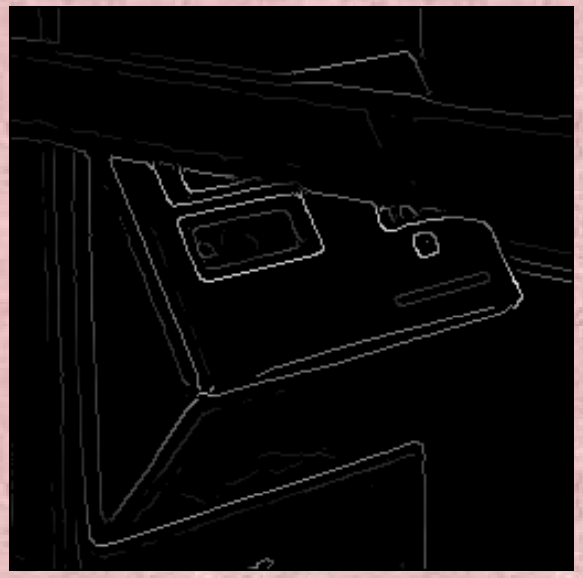


peaks

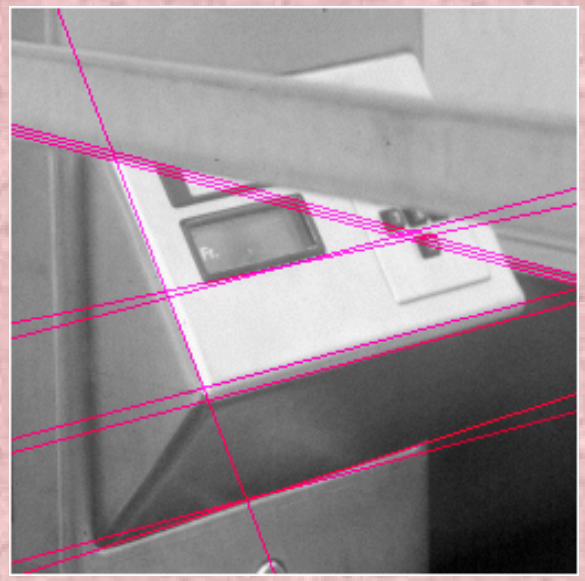




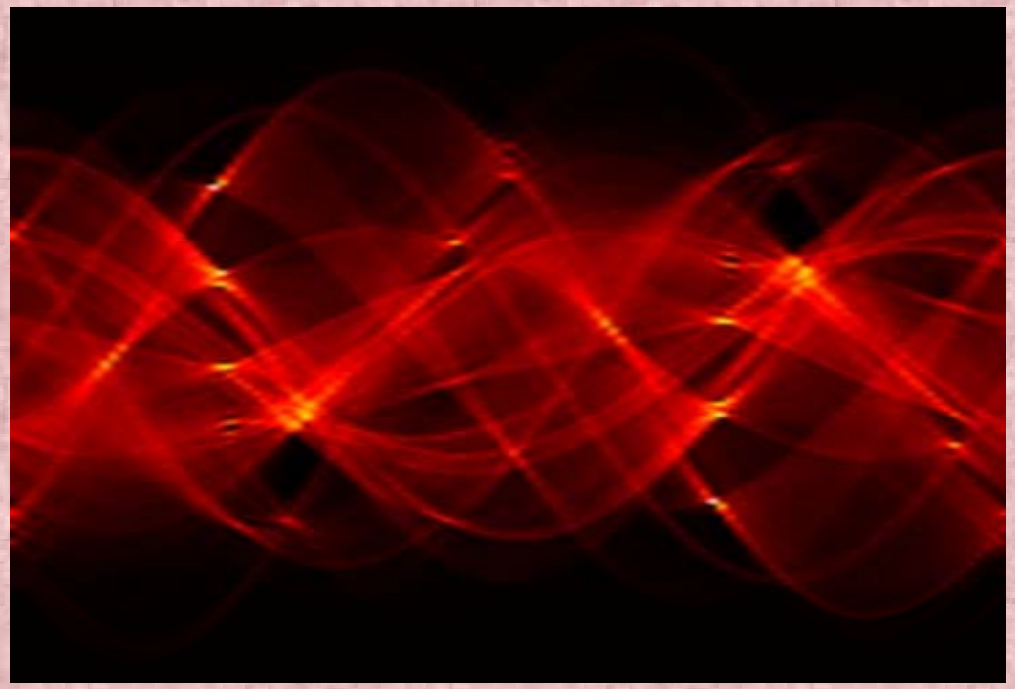
Original Image



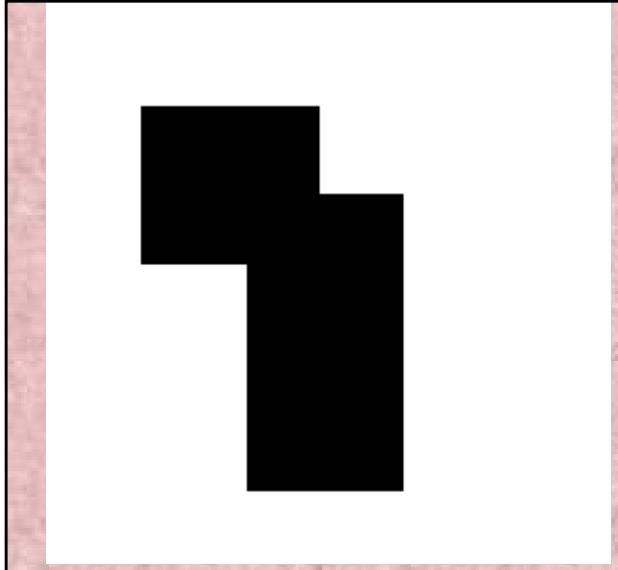
Edge Detection

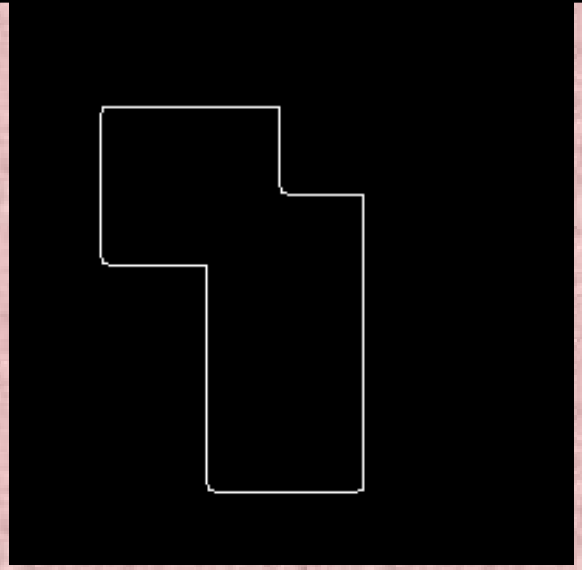


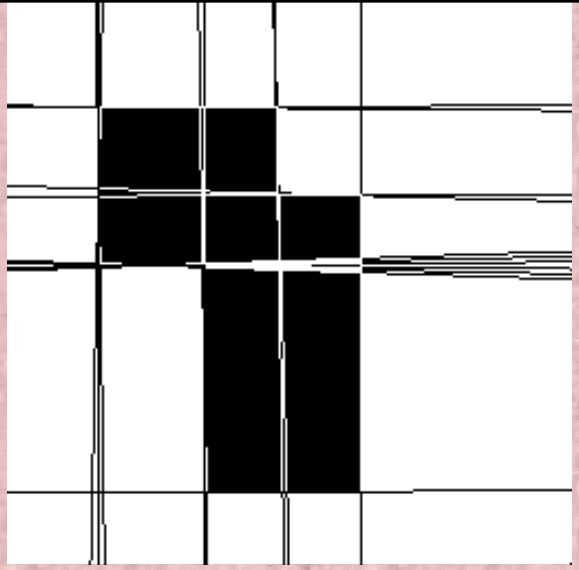
Detected Lines

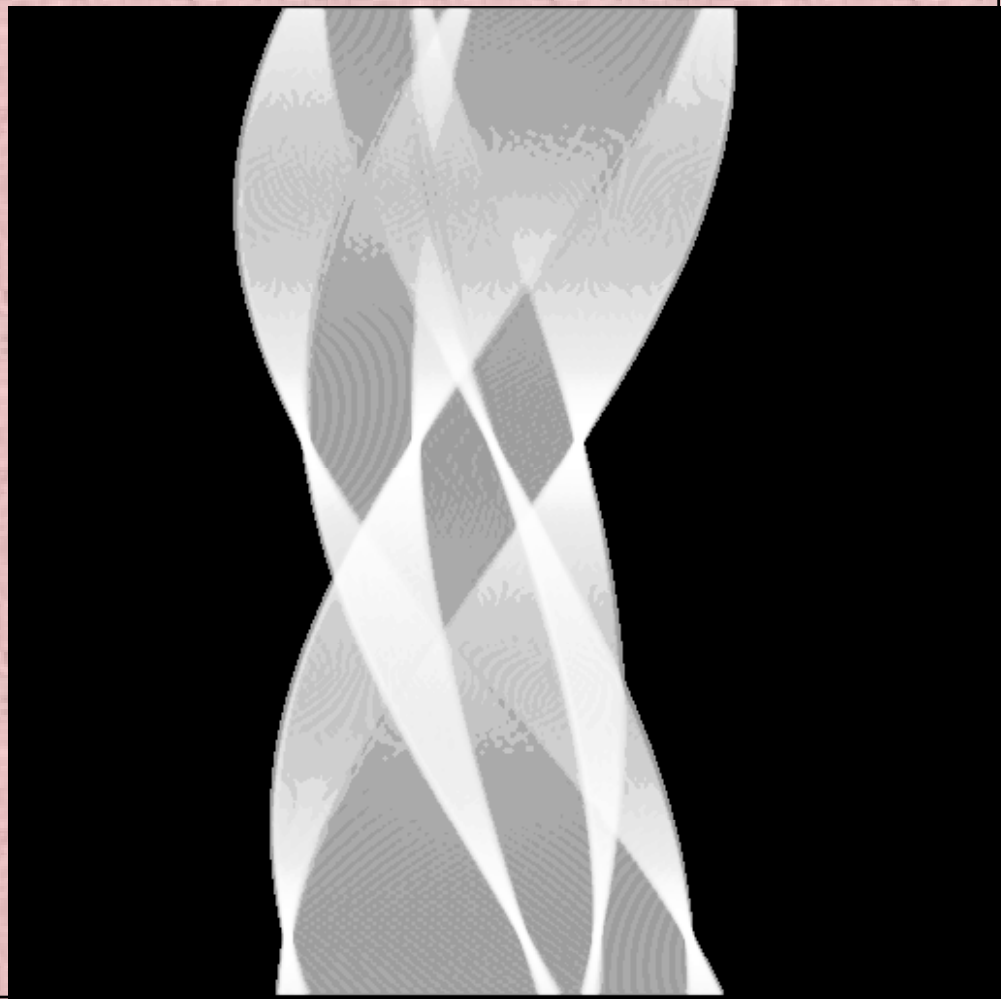


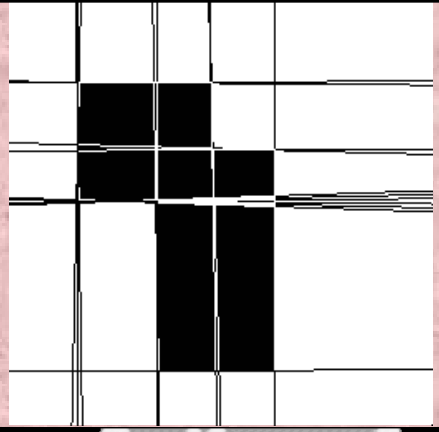
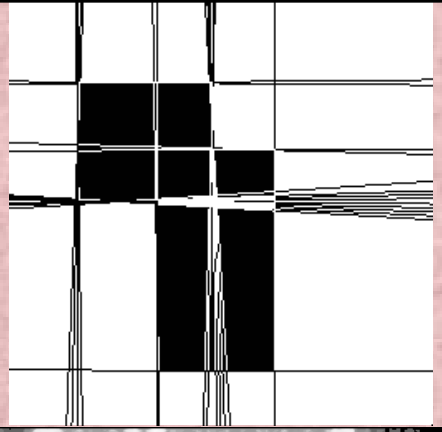
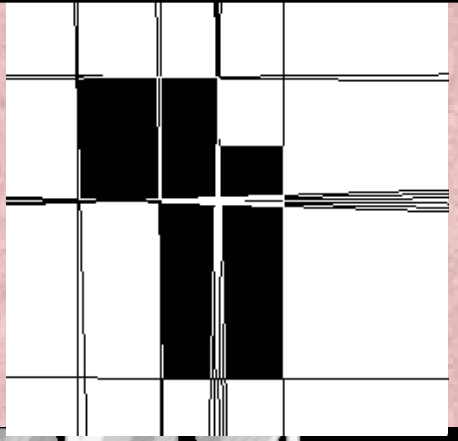
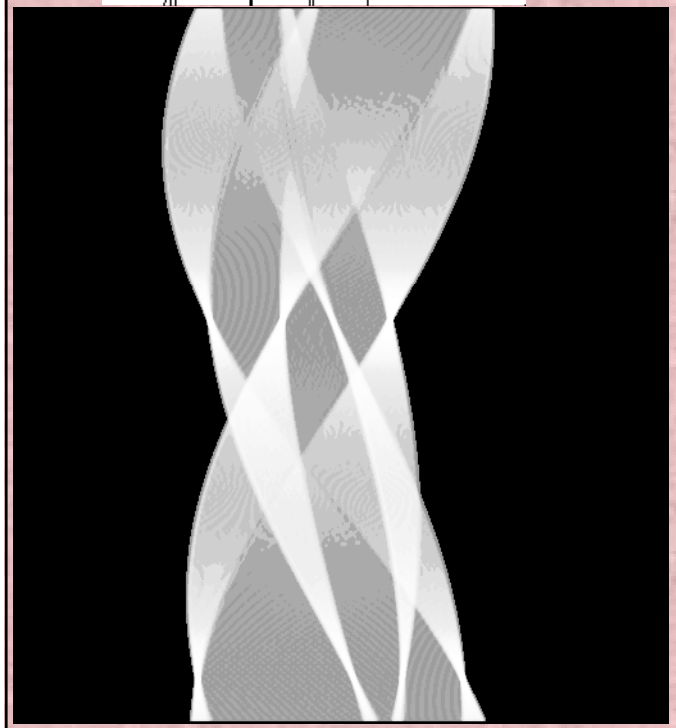
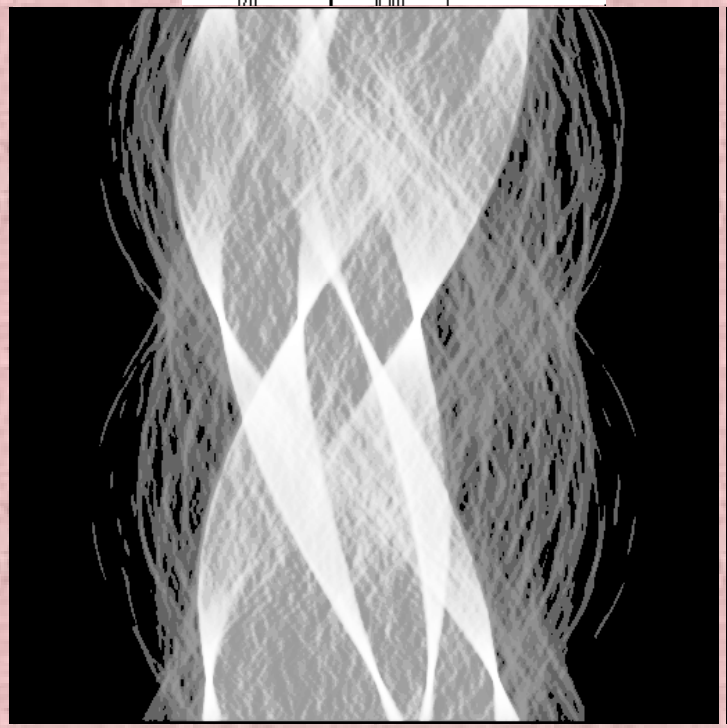
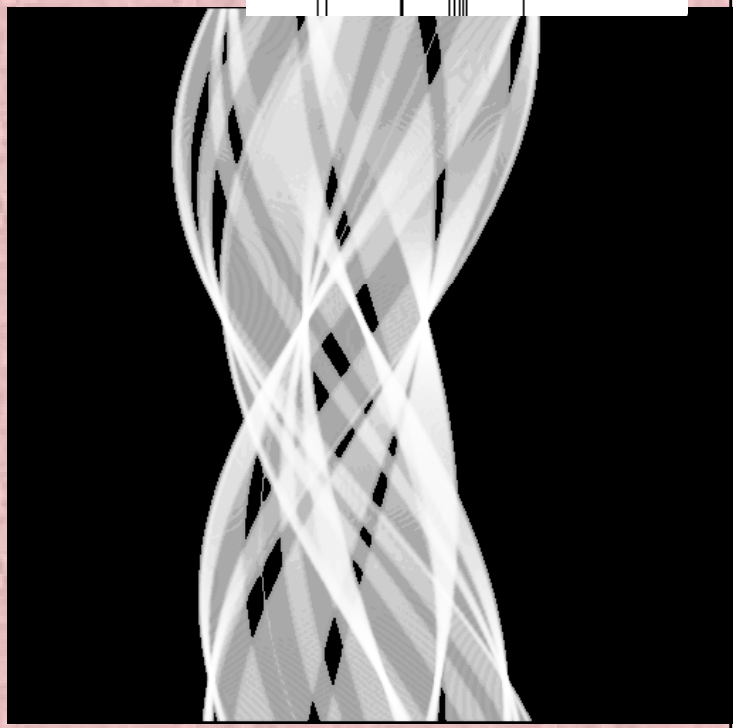
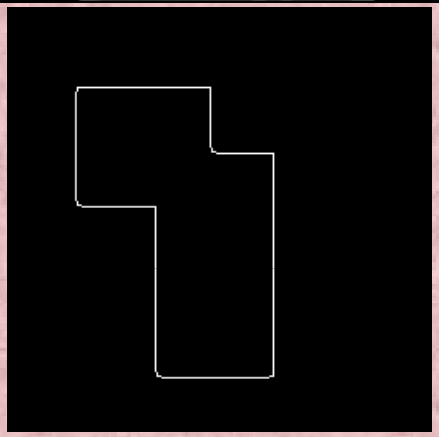
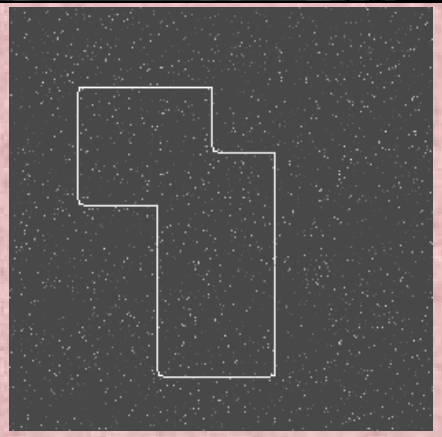
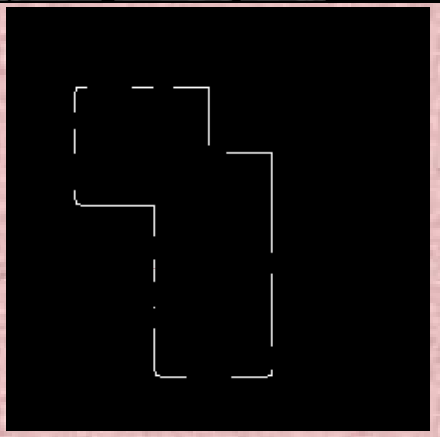
Parameter Space







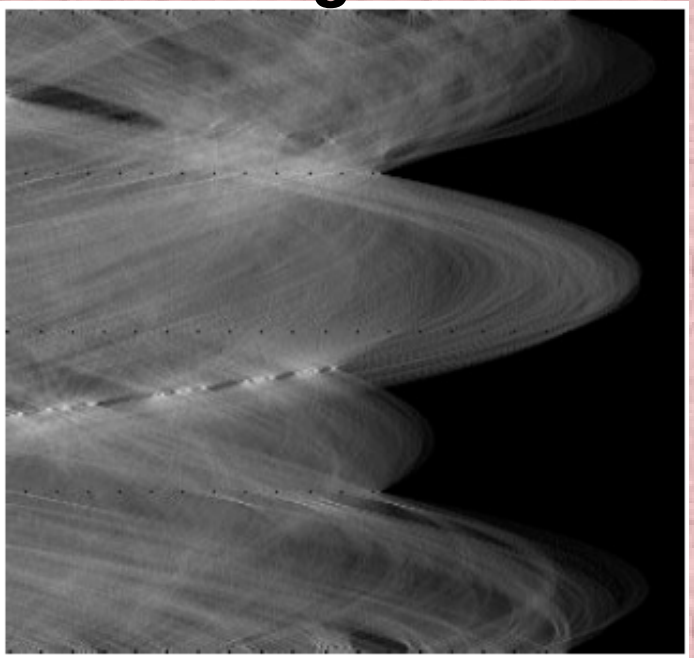


In the Hough transform, feature points vote for all possibilities through that point. This can cause unwanted clutter in the accumulator and produce false matches/alarms (take large gradient edges only).

Suppose that after we find the global maximum we remove all votes cast by feature points on or near the corresponding match in the image space.

We can then continue the process by looking for the next largest global maximum, remembering it, and removing the votes from its points.



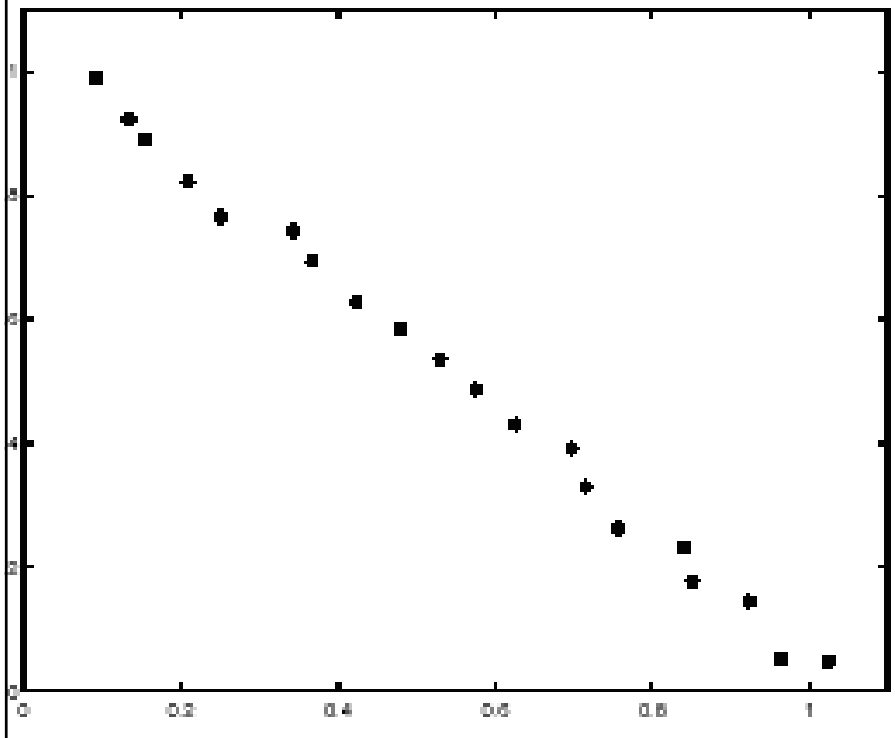
Operational tradeoffs for the Hough transform

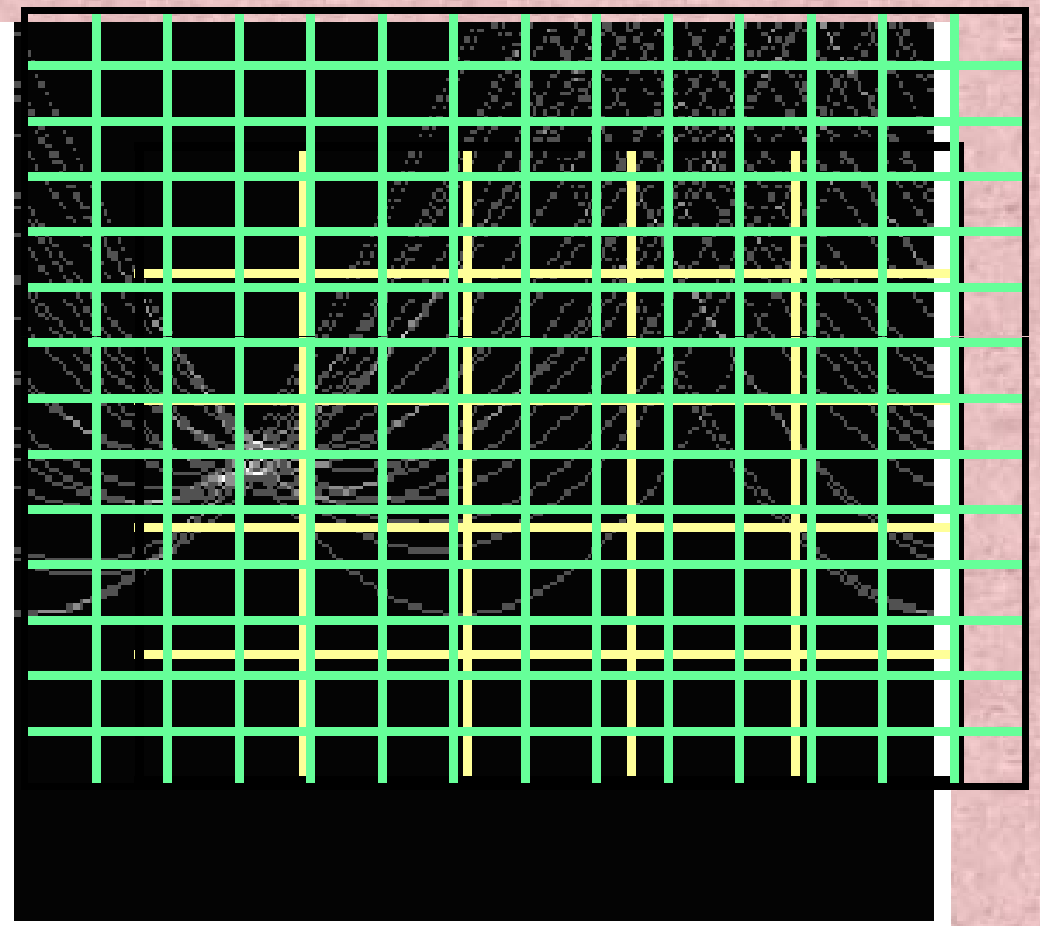
Issue of Resolution / Quantization of Accumulator cells:

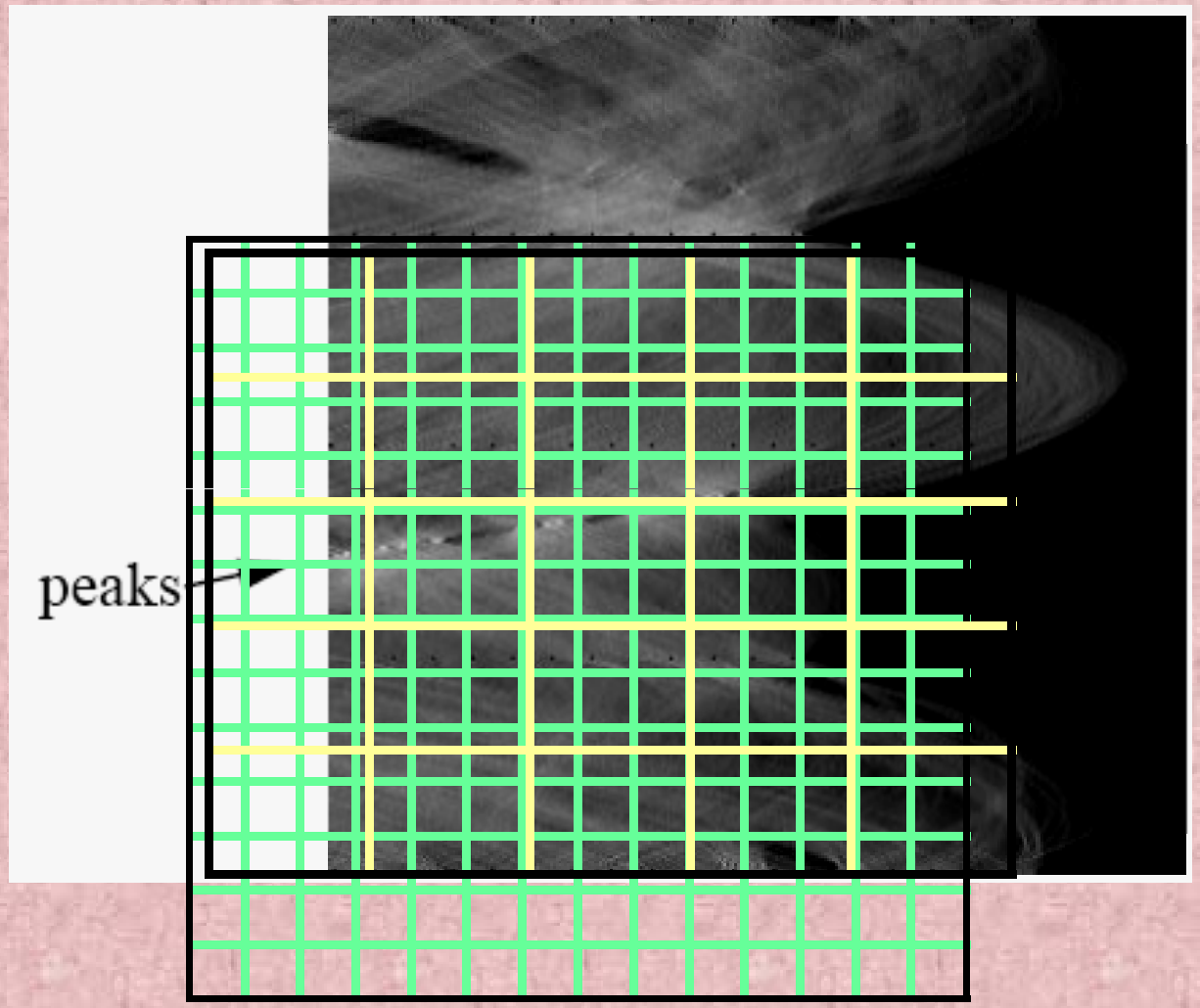
How big should the CELLS be?

- **Too big - large votes obtained when too many different lines correspond to a single bucket and we merge quite different lines;**

- **Too small - miss lines, because some points that are not exactly collinear cast votes for different buckets; problem becomes adverse with noise.**


0
0.2
0.4
0.6
0.8
1




peaks

Additional processing to create robustness in Hough transform

Post-processing steps:

A. How many lines?

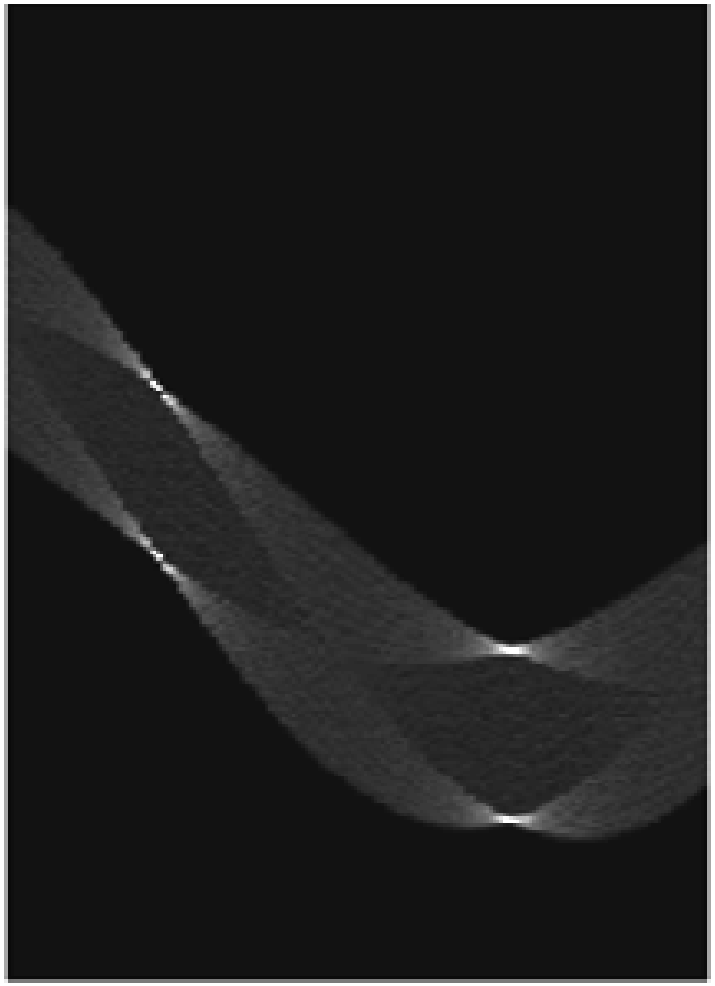
- Count the peaks in the Hough array
- Treat adjacent peaks as a single peak
- Smooth array, threshold and thin isolated clusters of bright spots

B. Which points belong to each line?

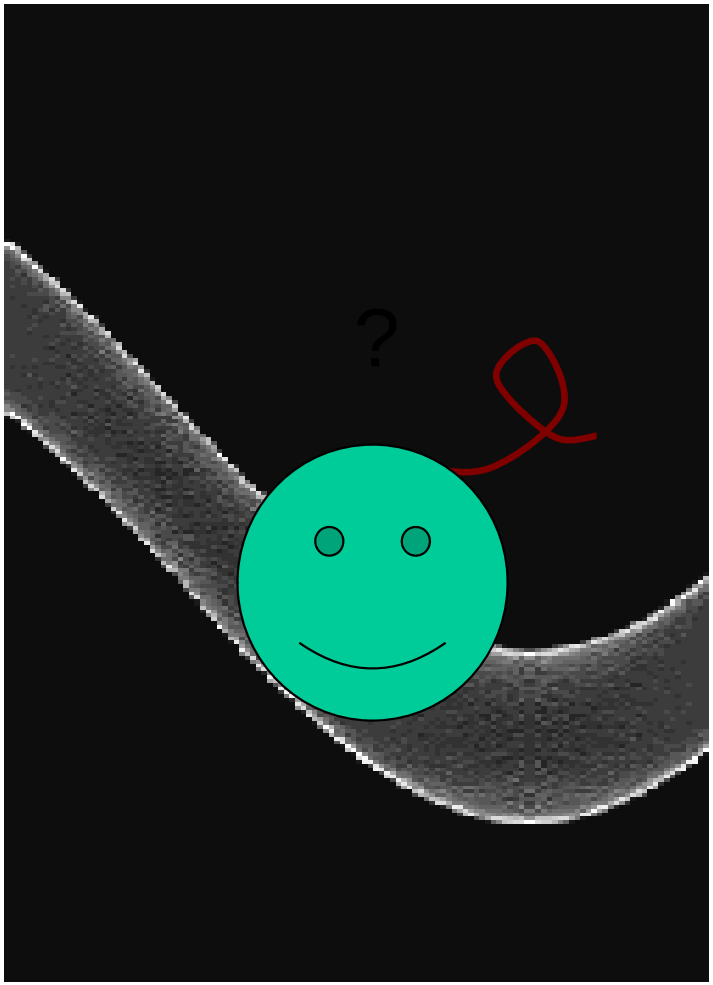
- Search for points close to the line
- Solve again for line (use edge orientation) and iterate
- Use a coarse to fine strategy

Other shapes

Square



Circular



Hough Transform for Parametric Curves

The Hough transform is not restricted to detecting straight lines, though that is a common use. The Hough transform can be used to detect other type of curves in an image as well as straight lines. Other geometrical shapes that can be described with a few parameters are also well suited to it.

The Hough transform can be used to detect other type of curves in an image as well as straight lines.

For example, if we wish to find circles, with equation:

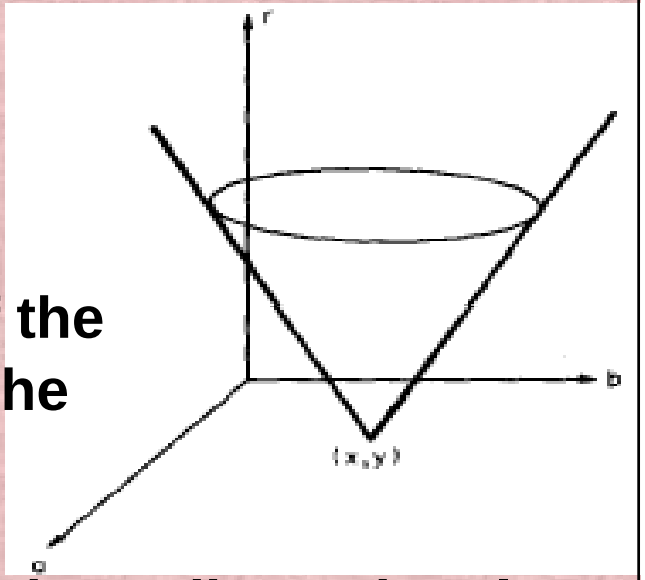
$$(x - a)^2 + (y - b)^2 = R^2$$

Then,

A circle is specified with three parameters: the X and Y coordinates of its centre (a & b), and R, its radius.

To find circles using a Hough transform, you need a three-dimensional accumulator array. ***Each Point in 2-D space yields (in 3-D Hough space) a*****:** *CONE*

Each edge element votes for all the circles that it could lie on, and the 3-D array is searched for peaks that give circle positions and radii. If you happen to know the radius in advance, you only need a 2-D accumulator.



Then,

- **Every point in (*x*, *y*) space corresponds to a *surface* in (*a*, *b*, *r*) space (we can vary any two of the parameters *a*, *b* and *r*, the third is determined by the equation of the circle).**

- **The basic method is, thus, modified to use a three-dimensional Hough Space (3-D array) *A*(*a*, *b*, *r*).**

- **All points in it, which satisfy the equation for a circle, are incremented.**

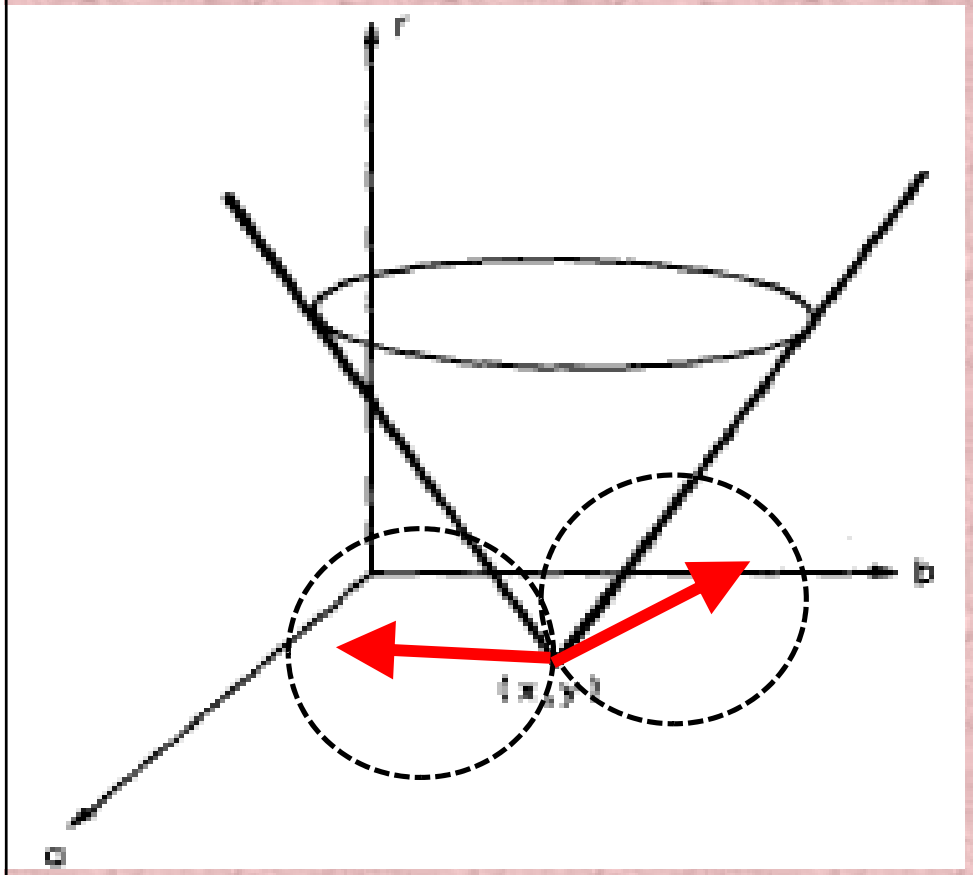
Use of Gradient information at each pixel (edge), converts the problem to a 2-D problem (1-D search).

The Conical surface in Hough space gets transformed to a:

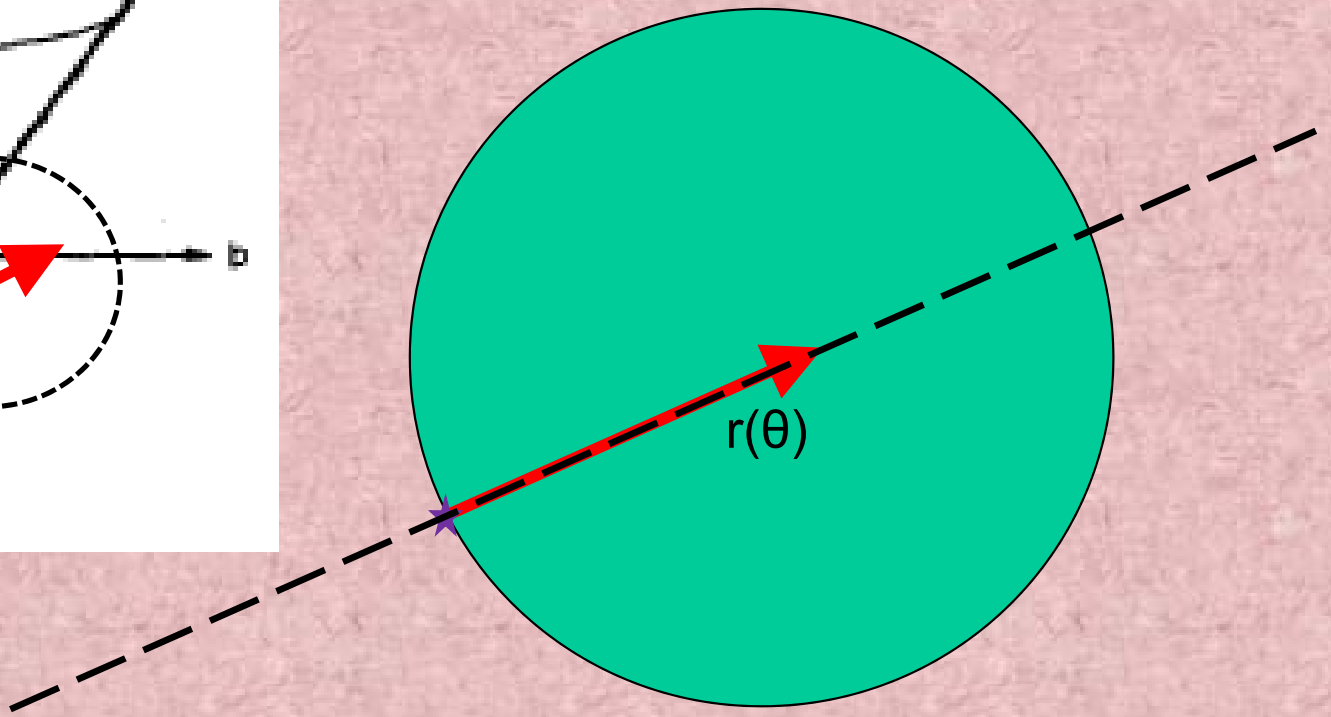
LINE

Use of Gradient information at each pixel (edge), converts the problem to a 2-D problem (1-D search).

The Conical surface in Hough space gets transformed to a:



LINE



More complicated shapes can be found - a general ellipse, for example, needs a 5-D parameter space.

So the method is really useful for simple and parametric curves. The concept of Hough space has been used in many other applications, wherever there is a scope of a **search in multi-dimensional space.**

$$G(X, C) = 0$$

In practice, the technique takes rapidly increasing amounts of time for more complicated curves as the number of variables (and hence the number of dimensions of the array *A*) increases. The two other factors on which the *time complexity* depends are:

- the level of discretization (higher value necessary for more accuracy)
- number of feature points present in the input

The **Generalized Hough Transform, introduced by D.H. Ballard in 1981, was a modification of the** Hough Transform using the principle of template matching.

This modification enables the Hough Transform to be used not only to detect an object described with an analytic equation (e.g. line, circle, etc), but also to detect an arbitrary object described with its model.

The problem of finding the object (described with a model) in the image can be solved by finding the model's position in the image.

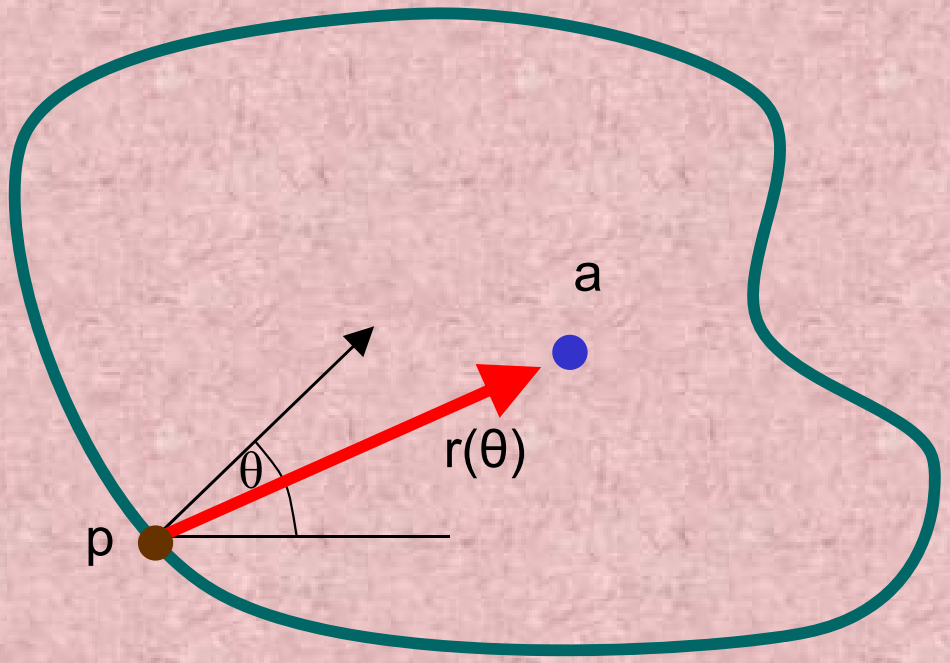
In the Generalized Hough Transform, the problem of finding the model's position is transformed into a problem of finding the transformation parameter that maps the model onto the image.

^ **[1] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, Vol.13, No.2, p.111-122, 1981.**

Generalized Hough transform

- We want to find a shape defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector:

r = a – p as a function of gradient orientation θ

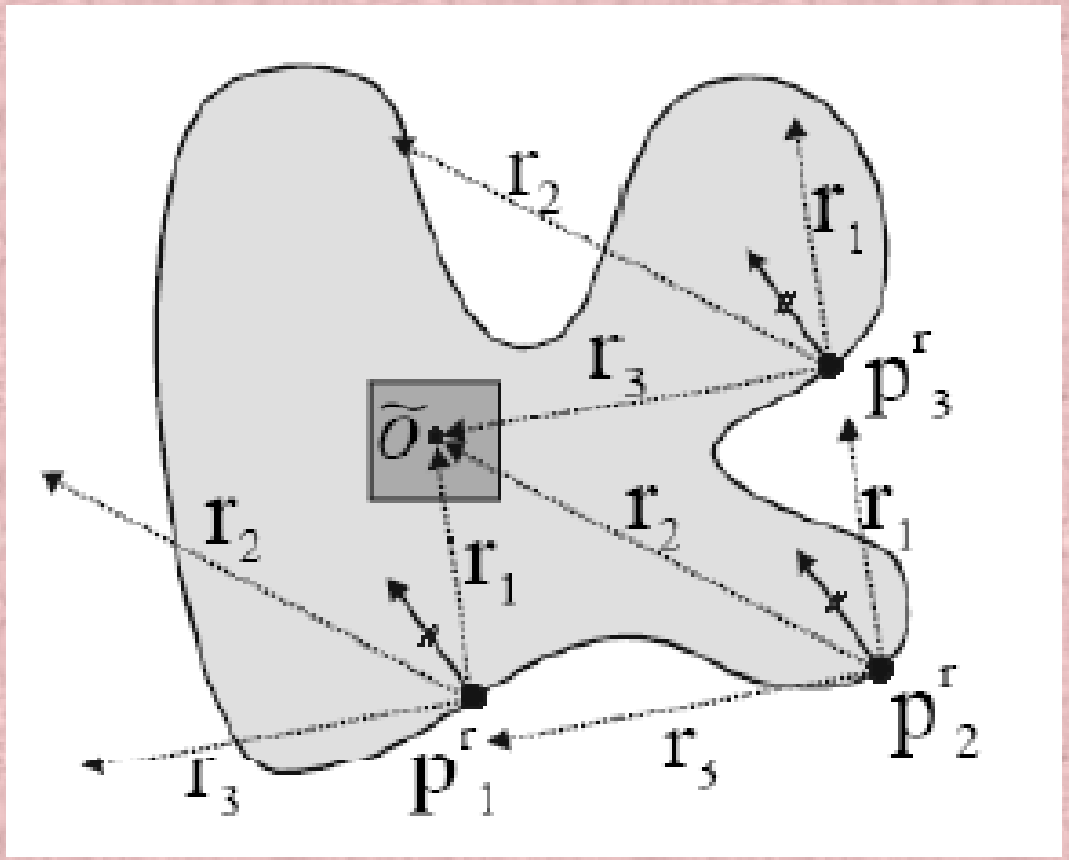


R-table may have multiple entries for each θ.

- For model shape: construct a **R-table** storing displacement vectors r as function of gradient direction θ.
- Detection: For each edge point ***p*** with gradient orientation ***θ***:
  - Retrieve all ***r*** indexed with ***θ***
  - For each ***r(θ)***, put a vote in the Hough space at ***p + r(θ)***
- Peak in this Hough space is the reference point with most supporting edges (only translation invariant in 2-D).
- For scale and orientation invariance, first construct additional tables for all plausible discrete values of **ϕ and s**; as **T_s [R(θ)] and T_ϕ [R(θ)]**. Then vote for :

$$p + r(\theta, s)$$

4-D Hough space for GHT: $S = \{a, \phi, s\}$



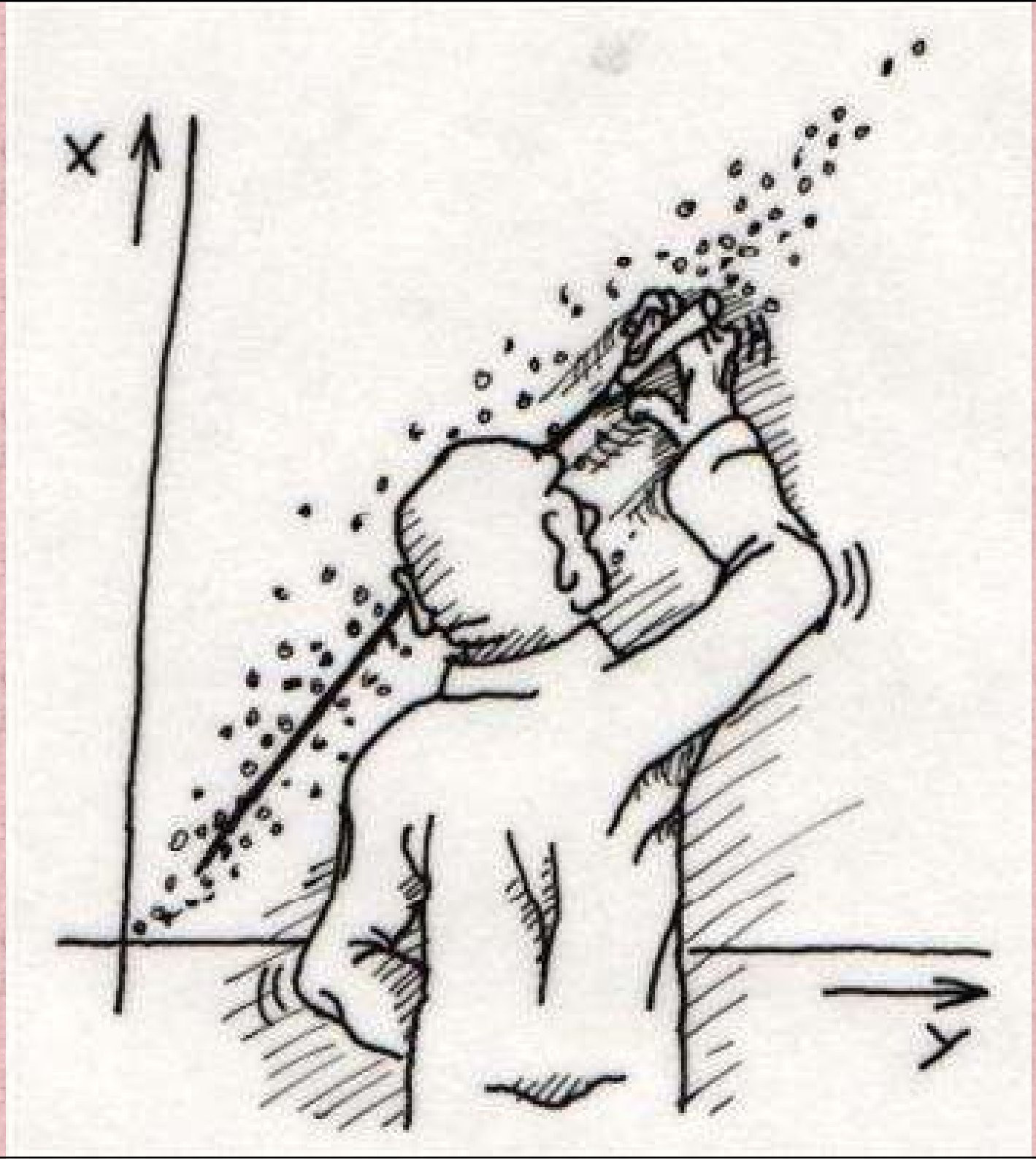
where $\mathbf{a} = (x_a, y_a)$ denotes a reference origin for the shape, ϕ the shape's orientation, and "s" is a scalar scaling factor.

Hough transform: +ves

- **All points are processed independently, so can cope with occlusion**
- **Some robustness to noise: noise points unlikely to contribute consistently to any single bin**
- **Can detect multiple instances of a model in a single pass**

Hough transform: -ves

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- It’s hard to pick a good grid size


x
y