Computational Vision

Handout 4.1: Hough Transform Basics

Hamid Dehghani

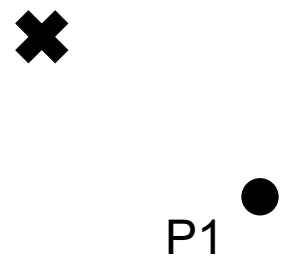
Office: 241

Point in space

- Consider a point, P1 in space
  - How do we define its location?

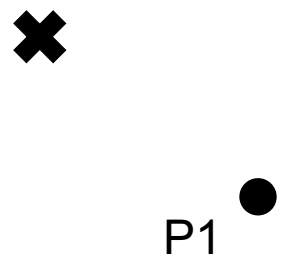
Point in space

- Establish an origin (or reference point)
  - Define the x and y location of P1 with respect to origin
  - P1 = [2, -3]



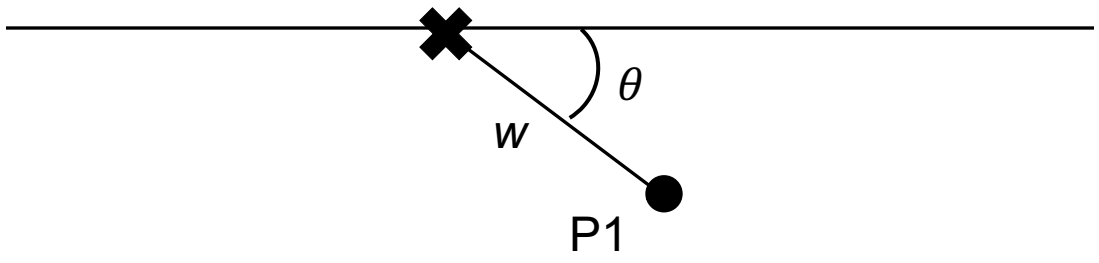
Point in space

- Establish an origin (or reference point)
  - Define the x and y location of P1 with respect to origin
  - P1 = [2, -3]
- How else can we do this?



Point in space

- The first solution was in Cartesian space
- We can also define this in Polar space
  - w is the distance and θ is angle from the origin

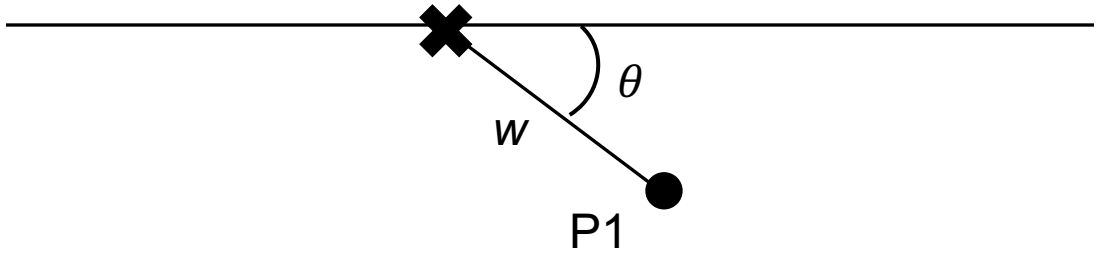


Point in space

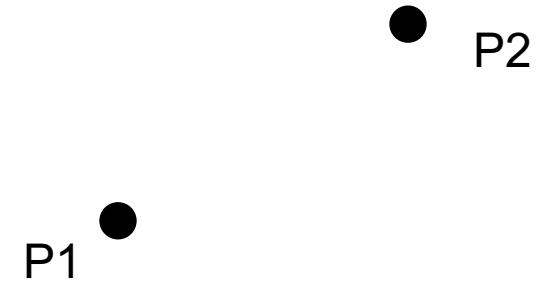
- Mapping from Cartesian to Polar coordinates:

$$w = \sqrt{x^2 + y^2}$$
$$w = \sqrt{2^2 + (-3)^2}$$
$$w = \sqrt{4 + 9}$$
$$w = \sqrt{13} \approx 3.9$$

$$\theta = tan^{-1}\left(\frac{y}{x}\right)$$
$$\theta = tan^{-1}\left(\frac{-3}{2}\right)$$
$$\theta \approx -56.3$$

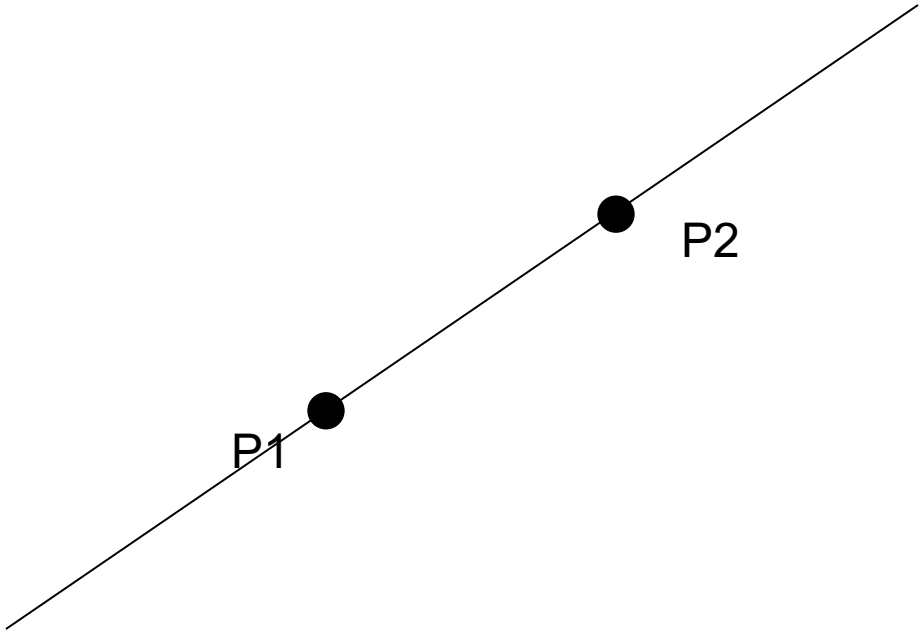


How about 2 points in space?



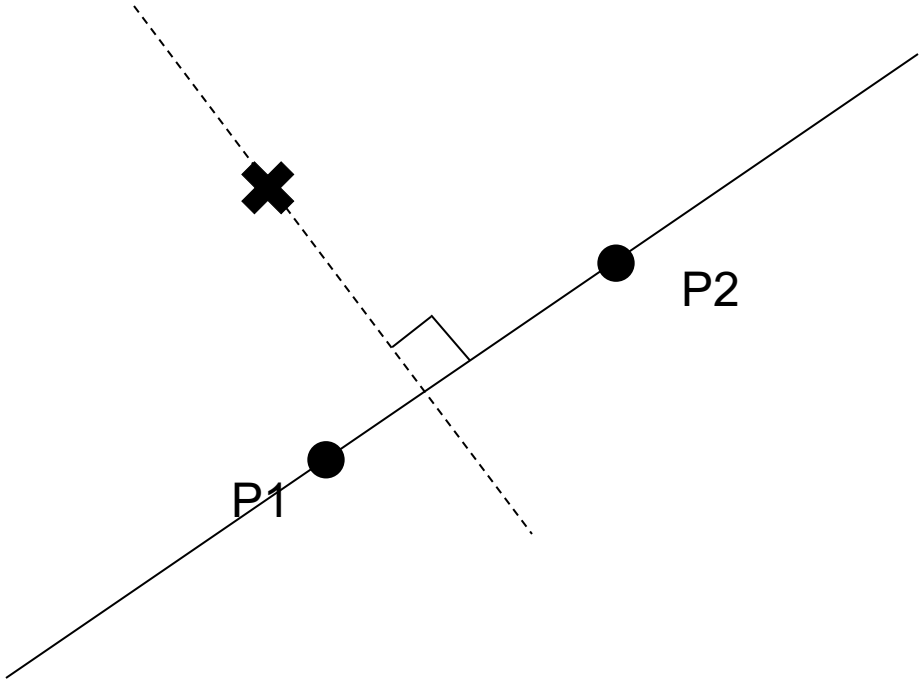
How about 2 points in space?

- It can be seen that a straight line can join the two points together.
- How do you define this line?



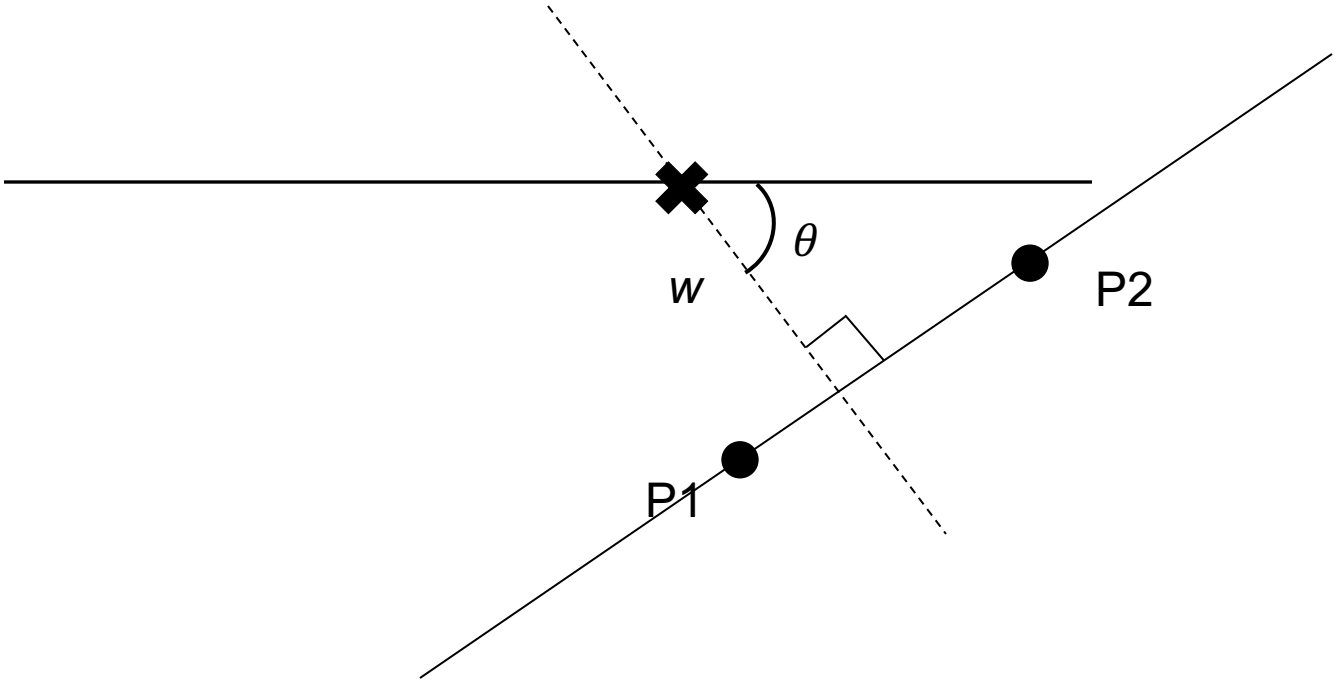
How about 2 points in space?

- As before, define a reference point
- Draw a line from reference point, perpendicular to this line: i.e. makes a right angle



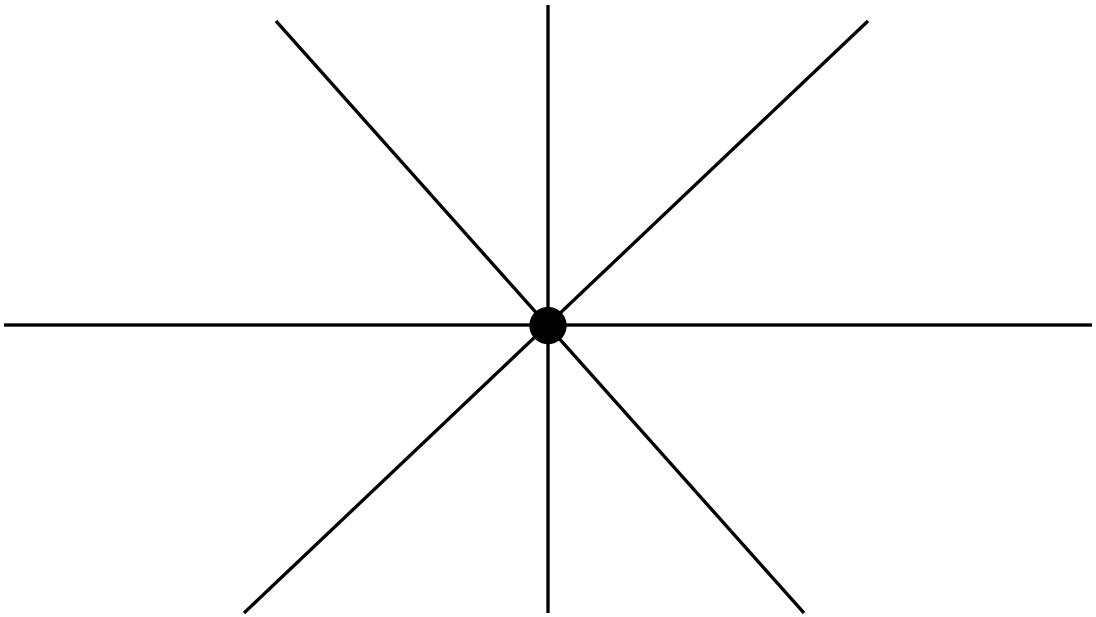
How about 2 points in space?

- Calculate distance and angle of this line with respect to origin



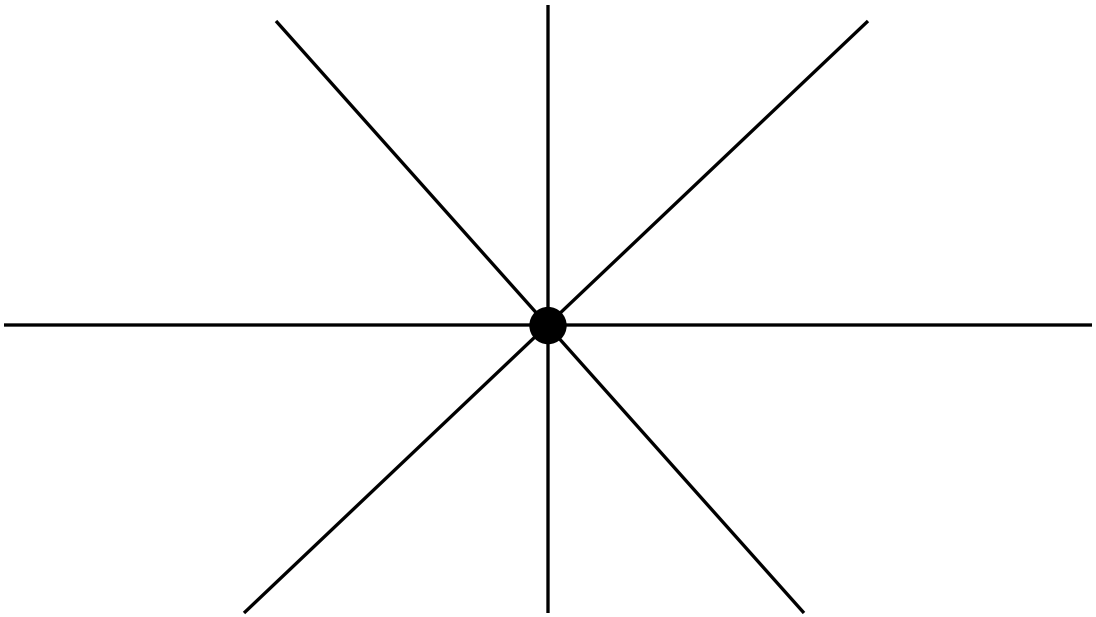
Point in space

- A point in space can have ‘many’ lines going through it



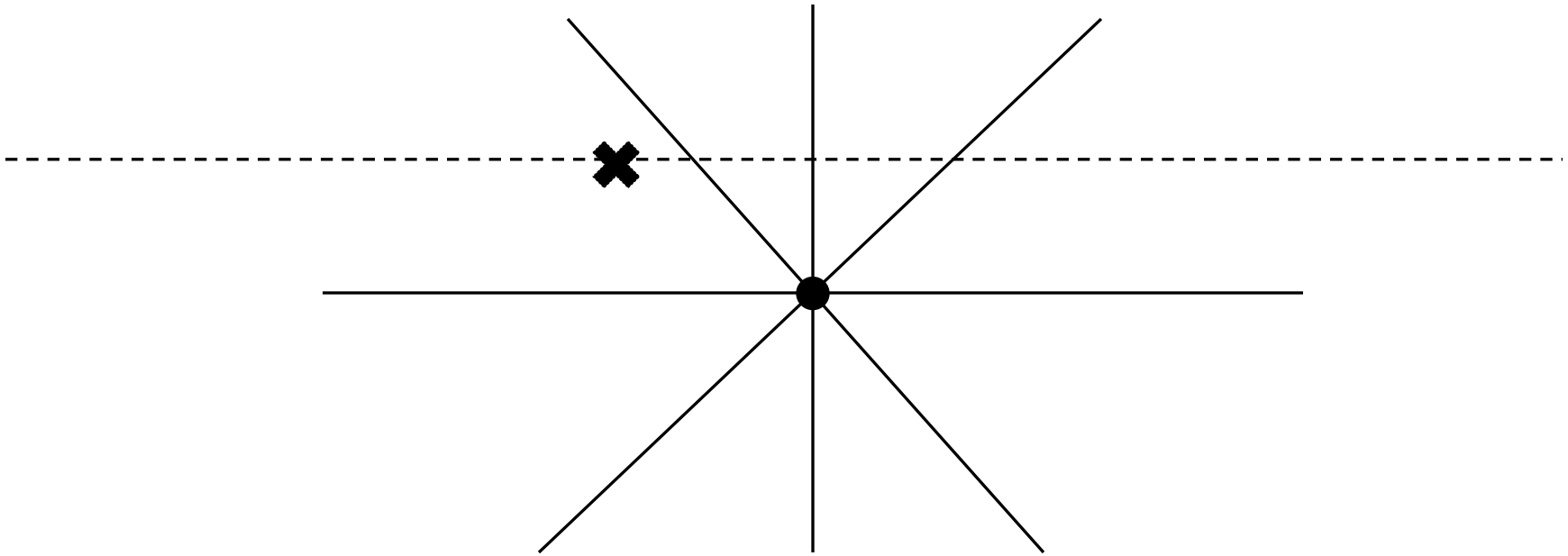
Point in space

- How many lines can go through a point?



Point in space

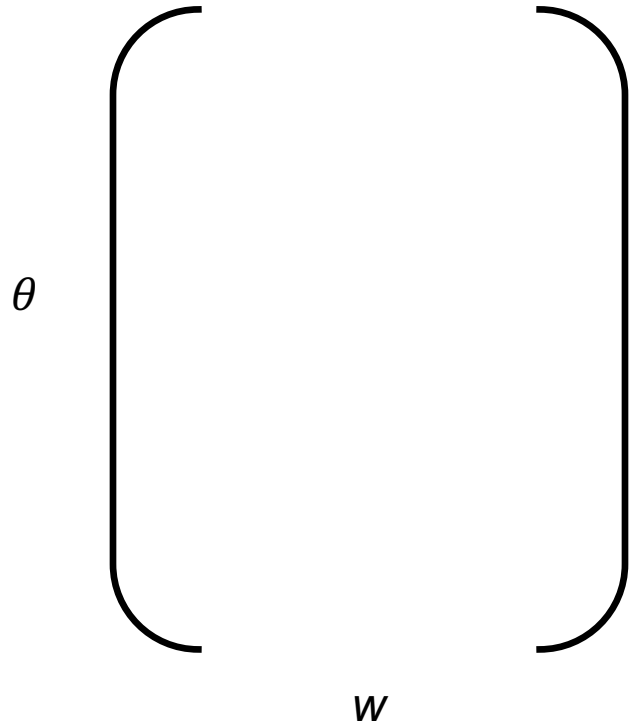
- For every angle from origin, and given [x,y], we can calculate distance, *w*



$$w = \mathrm{x}\cos(\theta) + y\,sin(\theta)$$

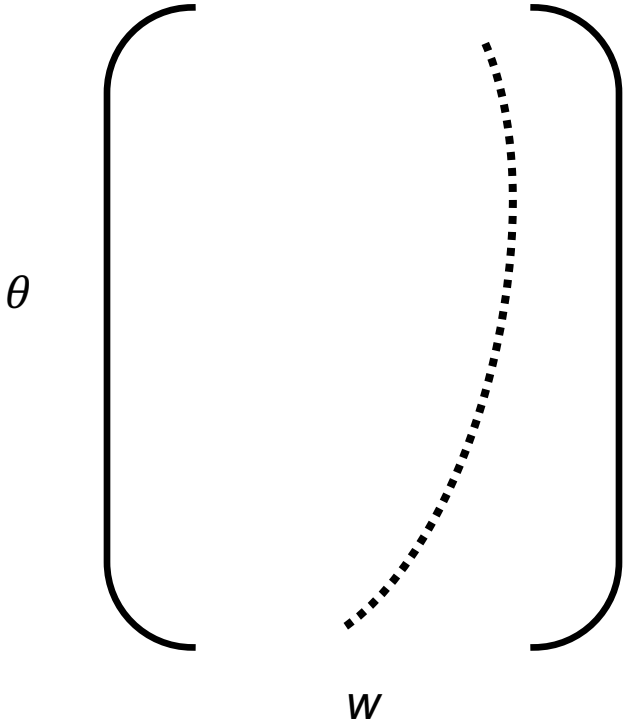
Point in space

- Create an array of $[\theta, w]$



Point in space

- Create an array of $[\theta, w]$
- For a given point [x,y], calculate all lines going through it.



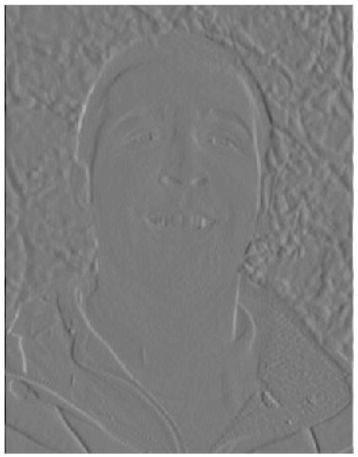
$$w = \mathrm{x}\cos(\theta) + y\,sin(\theta)$$

The story so far

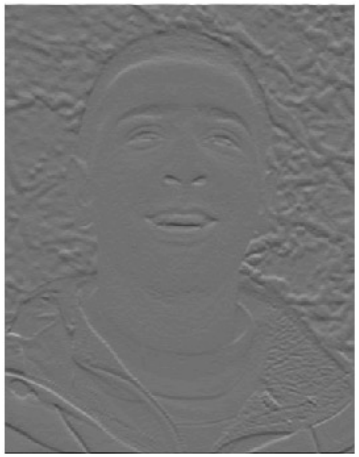
- We know how to find edges by convolving with the derivative of a Gaussian filter in two directions
- Steps:
  - Take image
  - Convolve mask with image for each direction
    - Calculate derivatives Gx and Gy
  - Calculate magnitude = $M(\vec{G}) = \sqrt{G_x^2 + G_y^2}$

Original

Gx



Gy



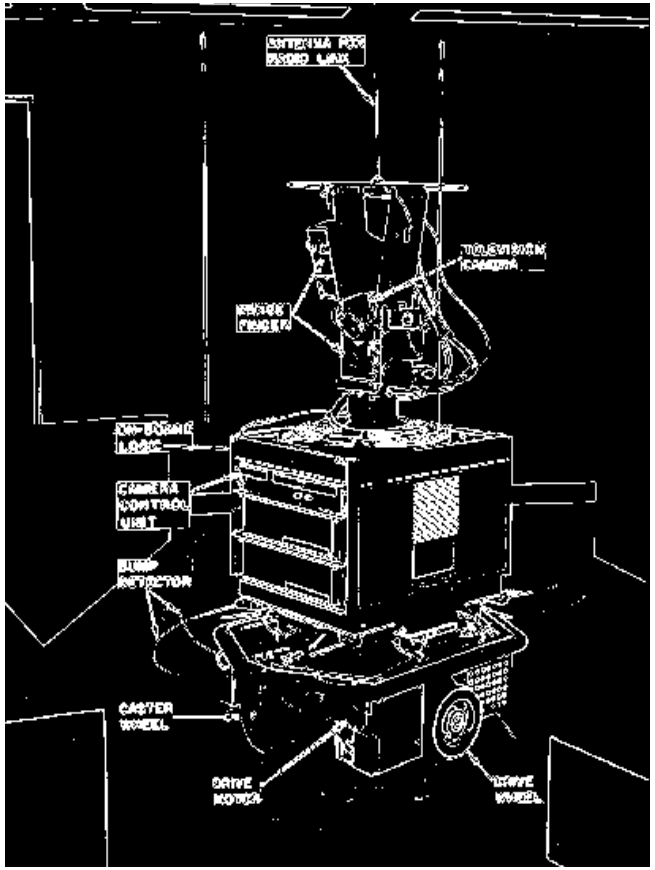
M

Finding edge features

But we haven't found edge segments, only edge points

How can we find and describe more complex features?

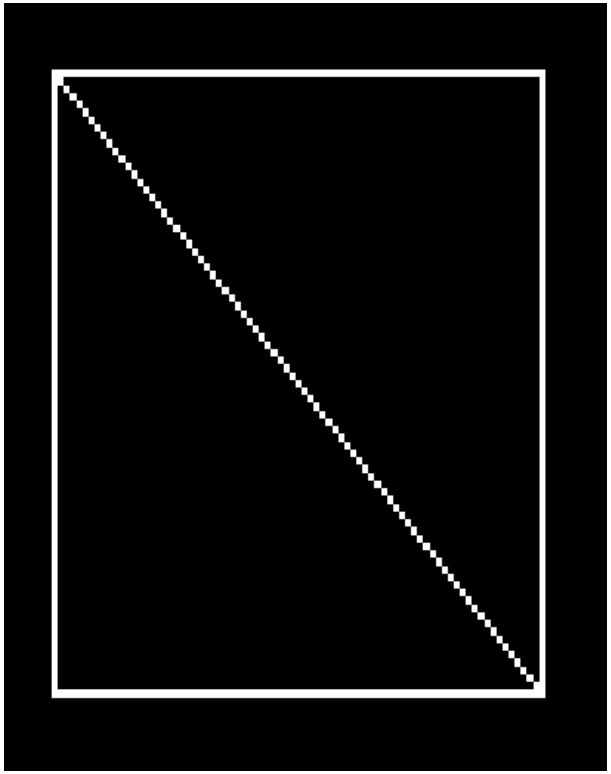
The Hough transform is a common approach to finding parameterised line segments (here straight lines)



The basic idea

Each straight line in this image can be described by an equation

Each white point if considered in isolation could lie on an infinite number of straight lines



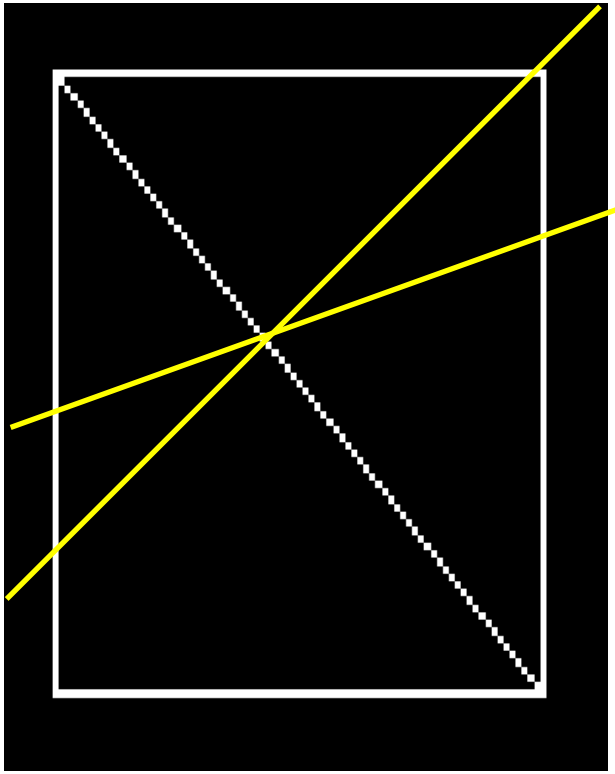
The basic idea

Each straight line in this image can be described by an equation

Each white point if considered in isolation could lie on an infinite number of straight lines

In the Hough transform each point votes for every line it could be on

The lines with the most votes win



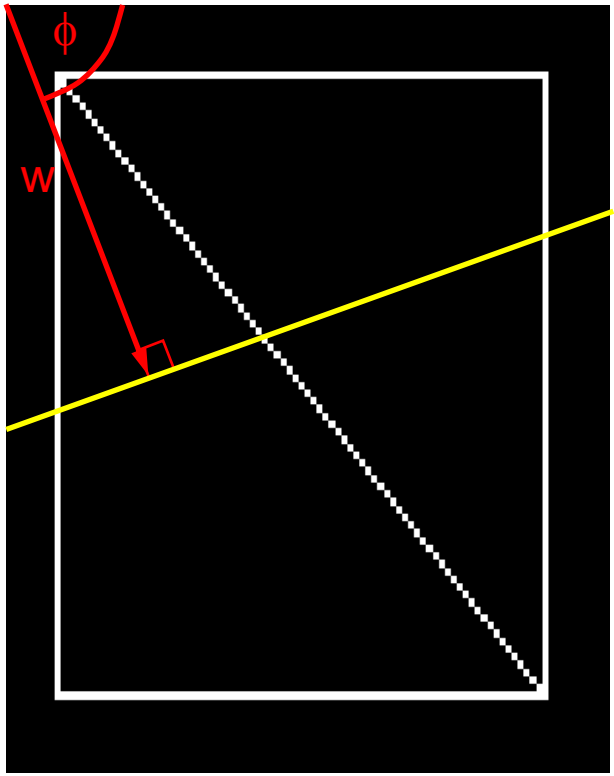
How do we represent lines?

Any line can be represented by two numbers

Here we will represent the yellow line by (w,ϕ)

In other words we define it using

- a line from an agreed origin
- of length w
- at angle ϕ to the horizontal

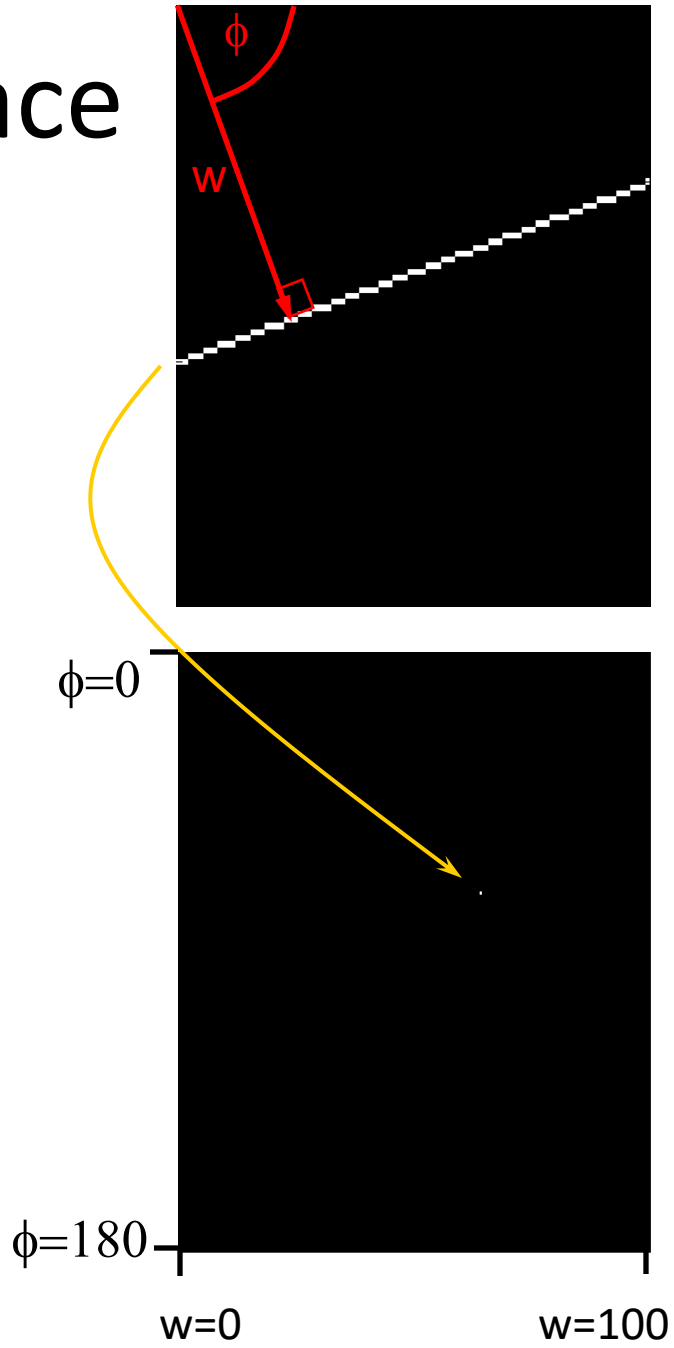


Hough space

Since we can use (w,ϕ) to represent any line in the image space

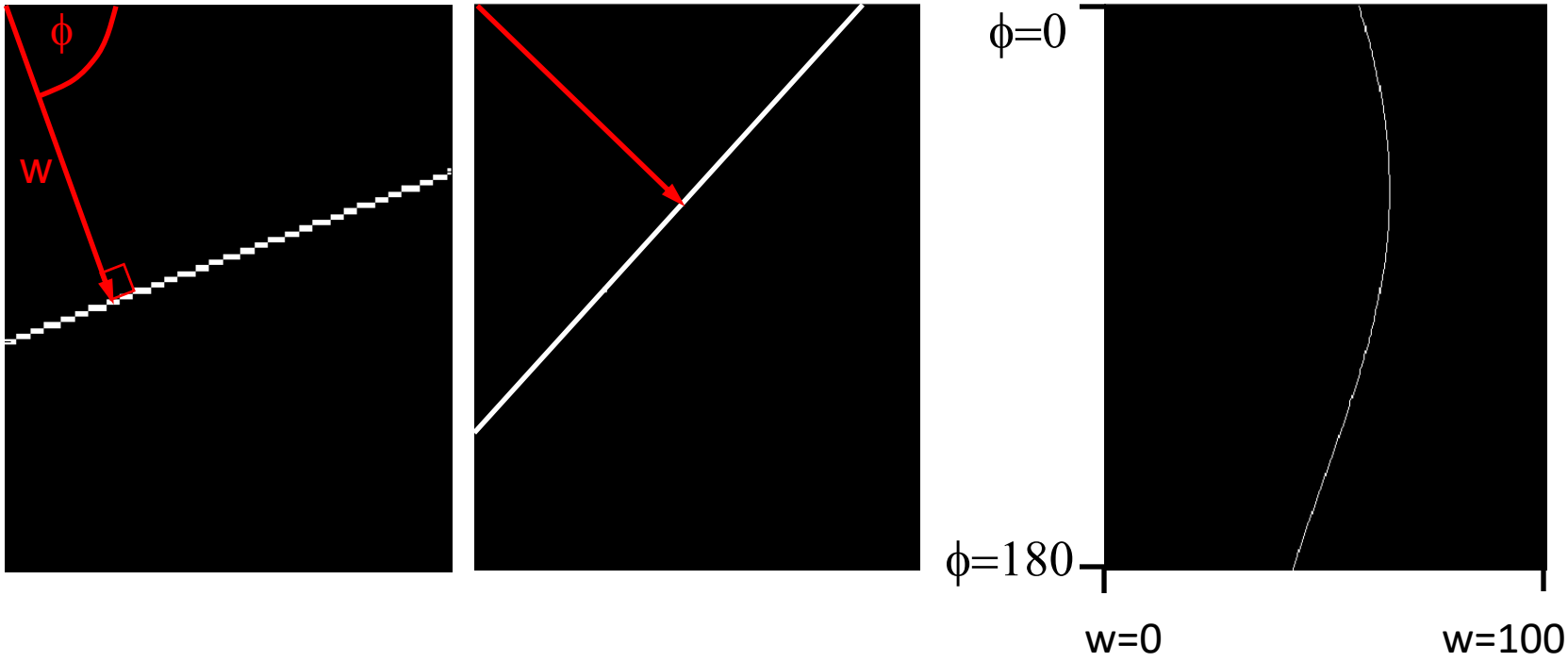
We can represent any line in the image space as a point in the plane defined by (w,ϕ)

This is called Hough space



How does a point in image space vote?

$$w = x\cos(\phi) + y\sin(\phi)$$



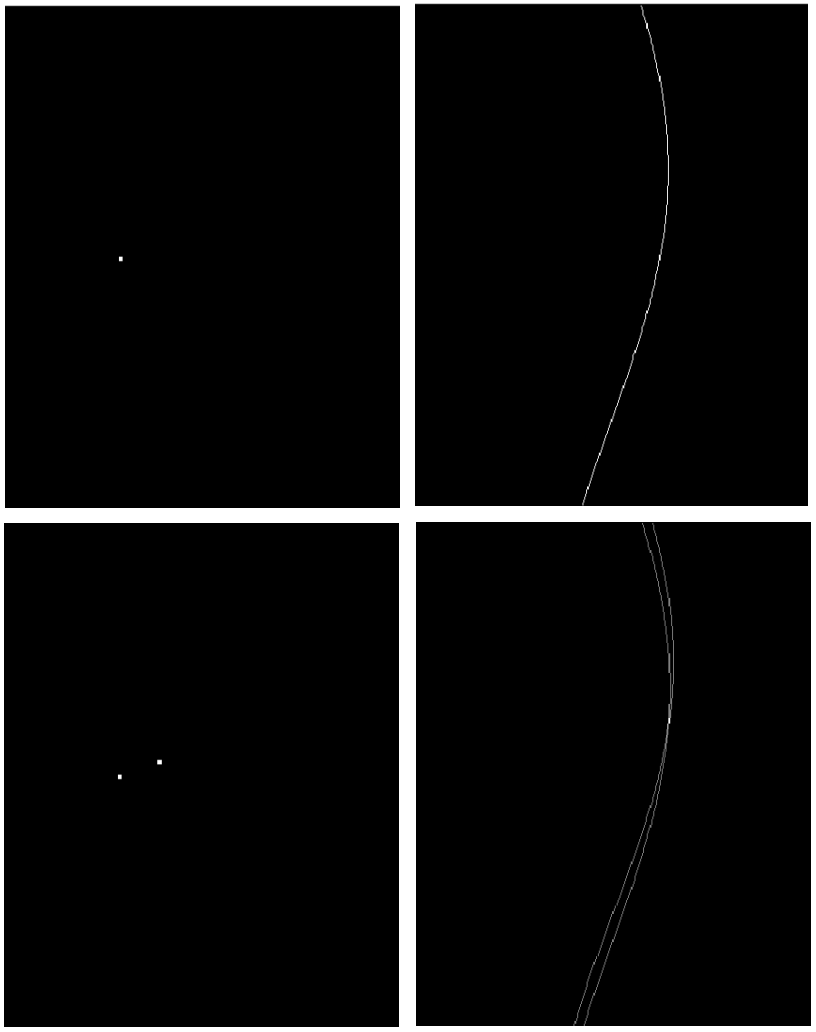
How do multiple points prefer one line?

One point in image space corresponds to a sinusoidal curve in houghspace

Two points correspond to two curves in Hough space

The intersection of those two curves has “two votes”.

This intersection represents the straight line in image space that passes through both points

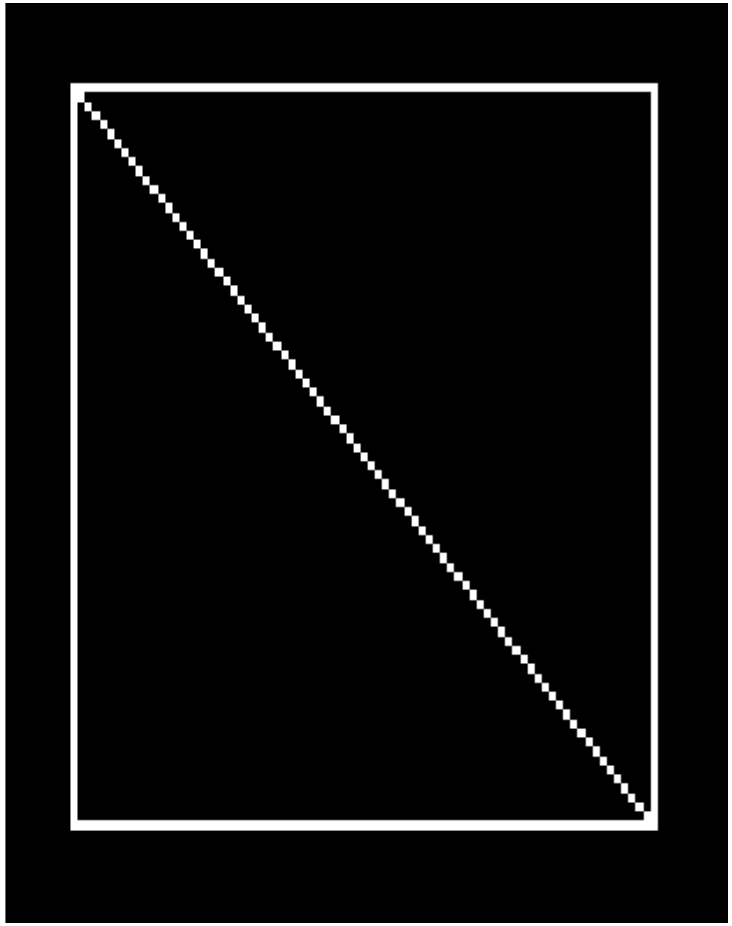


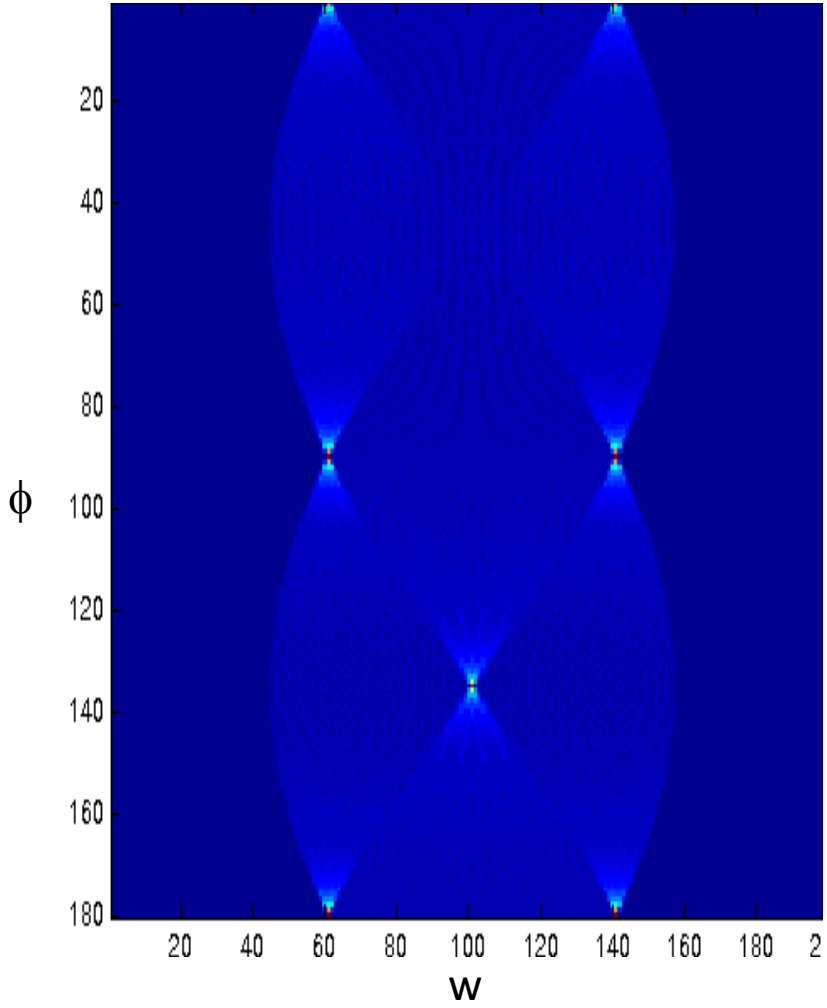
Hough Transform

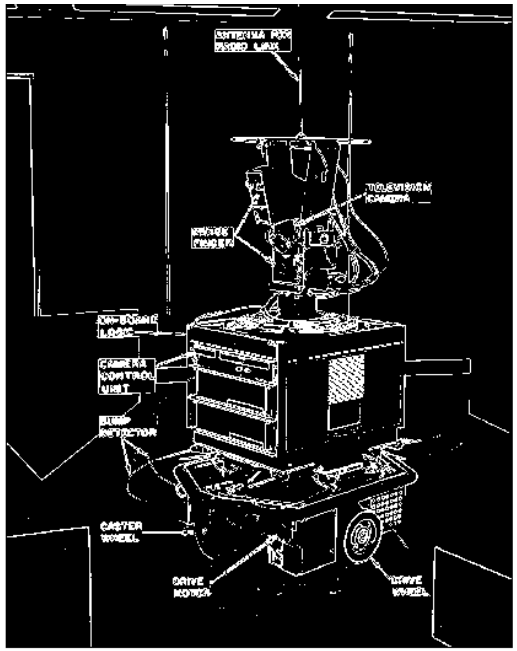
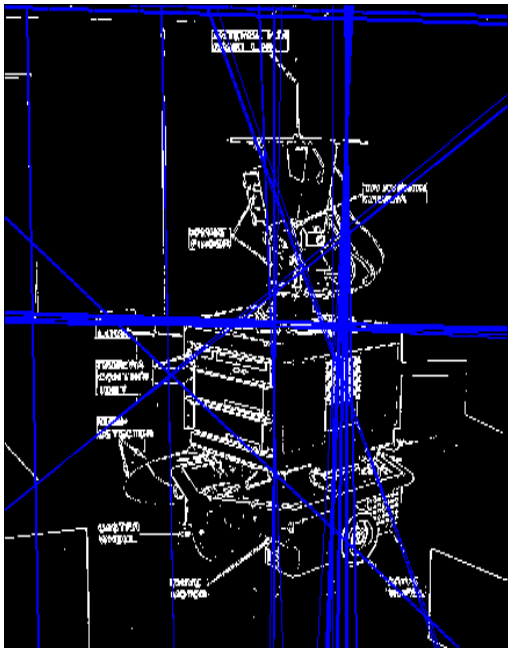
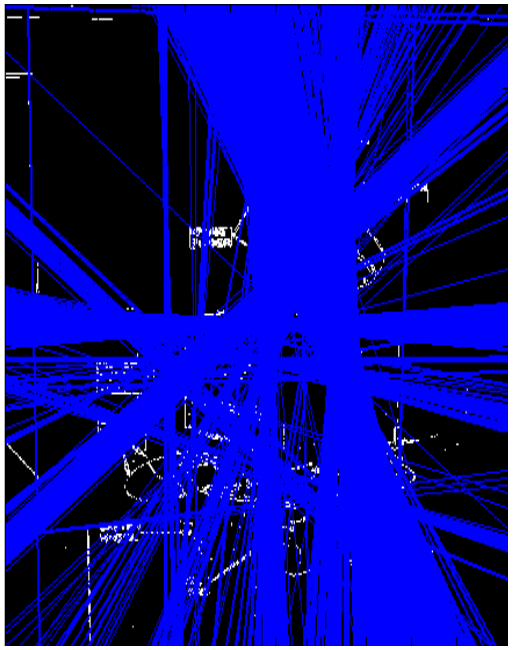
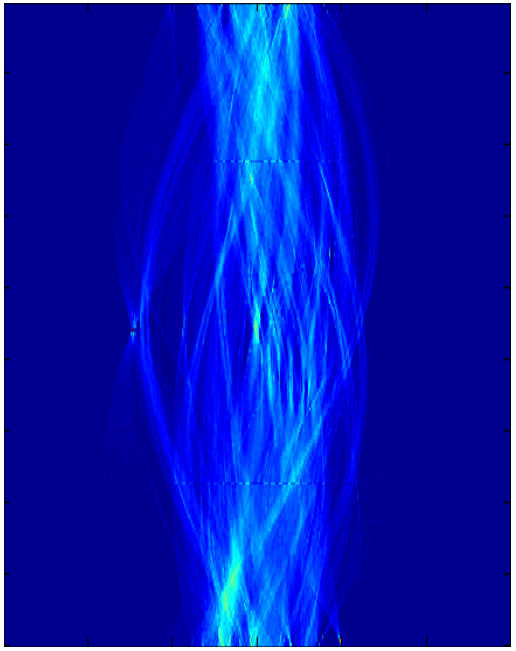
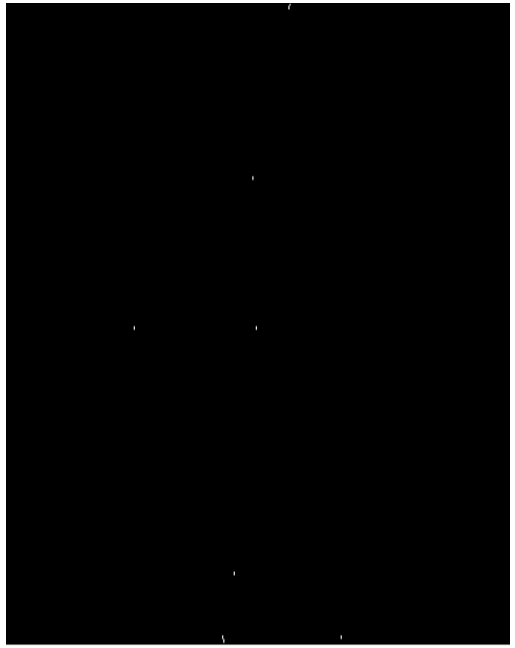
```
Create ϕ and w for all possible lines
Create an array A indexed by ϕ and w
for each point (x,y)
  for each angle ϕ
     w = x*cos(ϕ)+ y*sin(ϕ)
     A[ϕ,w] = A[ϕ,w]+1
  end
end

where A > Threshold return a line
```

A simple example






Hough Transform

- There are generalised versions for ellipses, circles
- For the straight line transform we need to supress non-local maxima
- The input image could also benefit from edge thinning
- Single line segments not isolated
- Will still fail in the face of certain textures

Circle Hough Transform

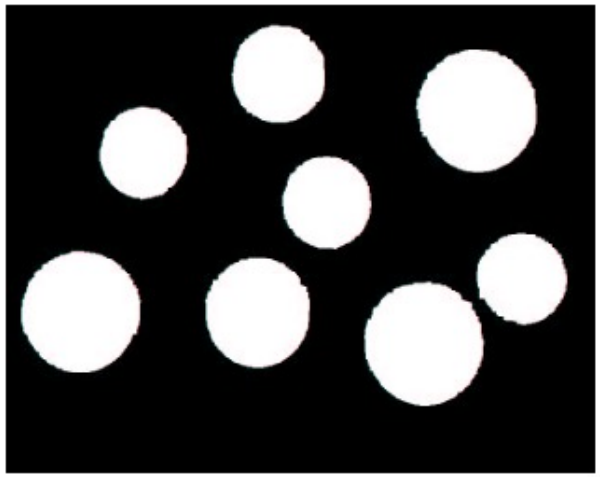
The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius R and center (a, b) can be described with the parametric equations

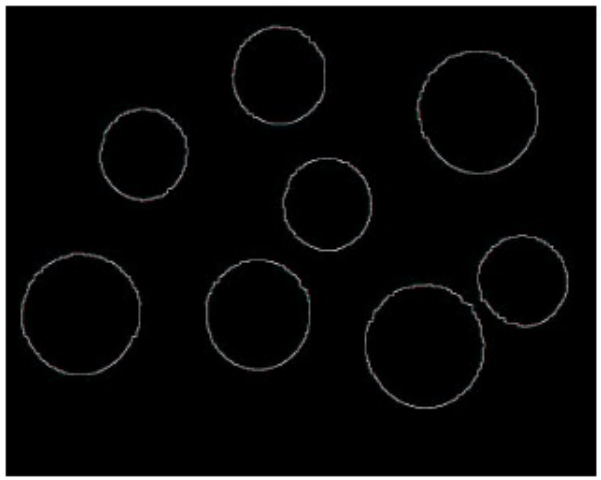
$$\begin{aligned} x &= a + R\cos(\theta) \\ y &= b + R\sin(\theta) \end{aligned}$$

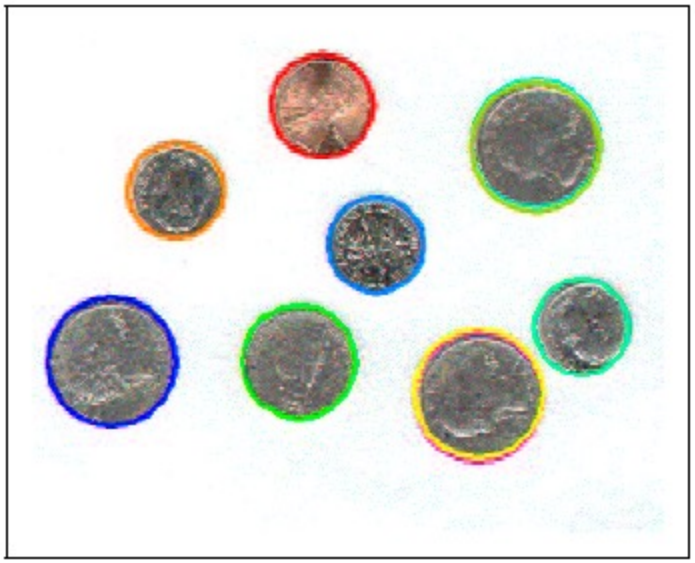
When the angle θ sweeps through the full 360 degree range the points (x, y) trace the perimeter of a circle.

If an image contains many points, some of which fall on perimeters of circles, then the job of the search program is to find parameter triplets (a, b, R) to describe each circle. The fact that the parameter space is 3D makes a direct implementation of the Hough technique more expensive in computer memory and time.

Example of Circle Hough Transform







k	R	C_x	C_y	Count
1	58	327	479	394
2	64	301	190	392
3	73	90	190	379
4	73	569	436	369
5	73	504	151	331
6	73	504	155	310
7	54	164	383	303
8	73	569	433	287
9	54	620	230	269
10	54	388	322	268