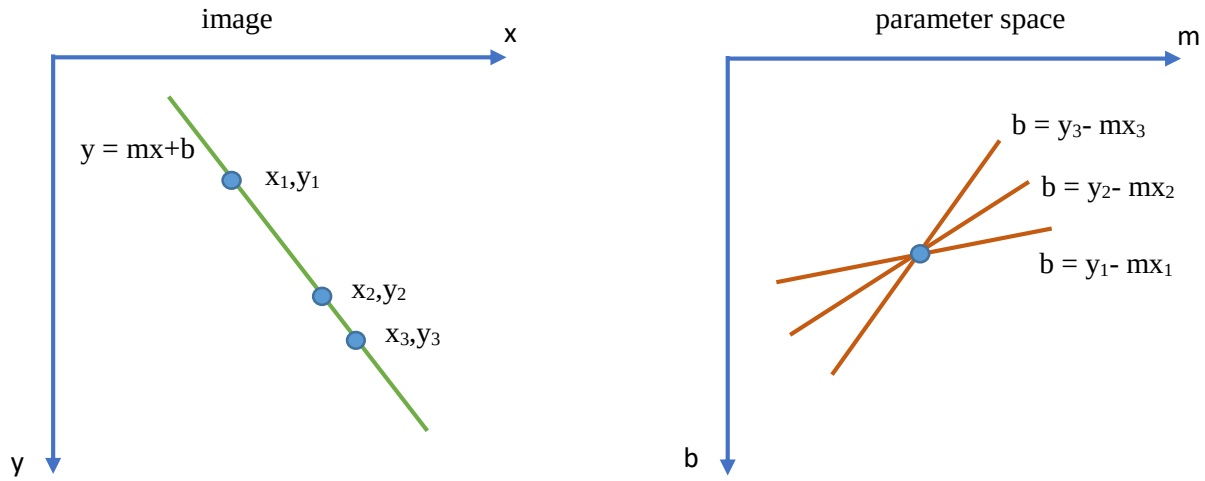
The Hough Transform for Lines

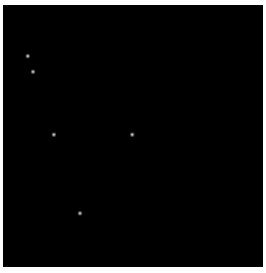
The Hough Transform can be used to detect lines in an image. A line in this case is defined as a group of collinear points aligned in a particular orientation. This method is robust and can handle discontinuities and other imperfections. We will study the line version first but the Hough transform can be used for any shape that can be expressed in parametric form, e.g. circles and parabolas.

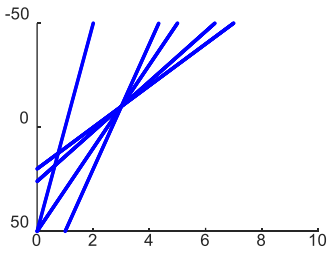
Every point in the image plane corresponds to a line in the parameter space.



Every point (edge pixel) in the image yields a line in the parameter space. Since there can be many different lines going through each point in the image we can interpret these lines in the parameter space as votes for those possible lines in the image. After all points in the image space have been mapped to lines in the parameter space we can then tally the votes to arrive at an estimate of the line in the image. The location in the parameter (m,b) space with the most votes will be the estimate for parameters of the line in the image. We will quantize the parameter space and use an accumulator array for the voting.

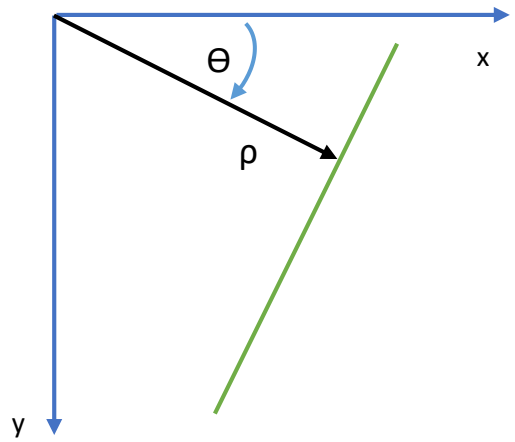
Below is an example of several points in the image (left) and their corresponding lines in the parameter space (right). Notice how the three lines corresponding to the four collinear points in the image intersect in the parameter space.





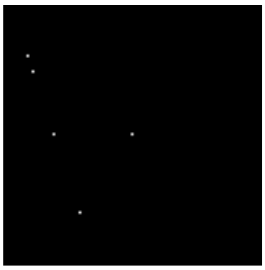
Since the above method will not support vertical lines (due to infinite slope) we will discuss a variation of this approach that works for any angle. We can describe a line in polar coordinates with a distance from the origin and an angle with the x axis.

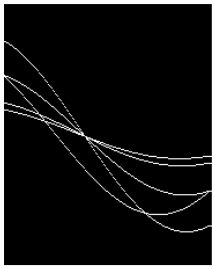
$$\rho = x \cos(\Theta) + y \sin(\Theta)$$



This has two advantages: it doesn't have the infinite slope problem for vertical lines and the resolution is uniform across lines of different angles.

The **ρ** – **Θ** parameter space is shown below on the right for the same image we used above.





The algorithm in polar coordinates:

1. Find edges in the image (e.g. use the Canny Edge Detector).
2. Initialize accumulator array A(ρ, Θ) to zero.
3. For each edge pixel (x,y), increment all cells in A that satisfy $\rho = x \cos(\Theta) + y \sin(\Theta)$.
4. Local maxima in A(ρ, Θ) correspond to lines.
5. Draw the detected lines on the image for verification.

Implementation details:

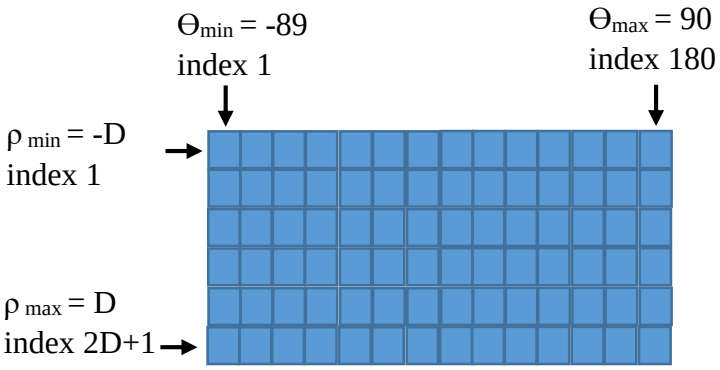
Step 1: Use MATLAB's Canny Edge Detector to obtain a BW image.

Step 2: Accumulator array A can be initialized and used with the following convention:

ρ is in pixels and Θ is degrees.

$\rho_{max} = D$ which is the diagonal of the image in pixels, i.e. D = sqrt(x_dim^2 + y_dim^2)

$\rho_{min} = -D$ and $\rho_{max} = D$



Step 3: Go through all edge pixels in the image

for each value of Θ in the range -89 to 90 calculate

$\rho = x \cos(\Theta) + y \sin(\Theta)$ (use MATLAB's cosd and sind for degrees)

increment entries in the array corresponding to the ρ, Θ pairs.

Step 4: If there is a single line in the image, you only need to find the maximum in A.

For multiple lines, find the ρ, Θ pairs for local maxima in A with a method similar to the one we used in template matching (zero out the neighborhood and find next max).

Step 5: For each maxima point found in A, draw a line on the image

for x from 1 to x_dim calculate

$y = (\rho - x \cos(\Theta)) / \sin(\Theta)$

plot the x, y pairs (avoid using an angle of zero with this method or make a special case where x = ρ for all y if Θ is zero)