# Table of Contents

Hough Transforms....1
Hough Transform for Lines....1
An Example....1
The Algorithm....2
In Polar Coordinates....2

# Hough Transforms

The Hough transform is an algorithm used to find simple forms in images, such as lines and circles. This transform usually takes its input from the result of an edge detection algorithm, in the form of a binary image (such as Canny's technique for finding connected edge pixels).

## Hough Transform for Lines

Any edge point in a binary image could be part of a set of possible lines. If we parametrize a line by its slope and intercept then an edge point from the image is transformed to a line in the slope-intercept space. Consider the line equation

$$y = ax + b$$

in the $(x, y)$ space. We can parametrize it in the following way:

$$b = -ax + y$$

and consider it as a line in the slope-intercept space $(a, b)$. Given an edge point $(x_1, y_1)$, we can form a line in the $(a, b)$ space as:

$$b = -ax_1 + y_1$$

If one converts all such edge points in the slope-intercept space, one obtains a number of various lines. If all the edge pixels fit a line equation, then all the lines in the $(a, b)$ space will have a common intersection $(a_1, b_1)$. And conversely, this intersection point yields the line of pixels in $(x, y)$ space:

$$y = a_1 x + b_1$$

## An Example

Consider the points $p_1 = (1,2)$ and $p_2 = (0,1)$. In $(x, y)$ space, these two points form a line expressed as $y = 1 \cdot x + 1$. What line in $(a, b)$ do we obtain with point $p_1$? We simply use the $(x, y)$ values of the point int the equation $b = -ax + y$ and obtain the following line in $(a, b)$ space:

$$b = -1 \cdot a + 2$$

Similarly, point $p_2$ yields the line $b = 1$. We now find the intersection point of these two lines as $(1,1)$. The line fitting the two points in $(x, y)$ is immediately

obtained as:

$$y = 1 \cdot x + 1$$

## The Algorithm

- Quantize (discretize into cells) the parameter space $(a, b)$ . These cells are usually called accumulator cells.
- Count the number of times a line in $(a, b)$ space obtained with a point in $(x, y)$ space intersects a given cell:
  - For each edge point in the image, find the values $(a, b)$ in the range $[a_{min}, a_{max}], [b_{min}, b_{max}]$ defining the line corresponding to this point.
  - Increase the value of the accumulator cells for these $(a, b)$ points.
- Lines are found as local maxima in the accumulator cells.

## In Polar Coordinates

It is preferable to use the polar coordinate representation of a line such as

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

In the polar coordinate space, both parameters $\rho$ and $\theta$ are bound. This is not the case with the usual line representation with slope and intercept and causes quantization problems. Using polar corrdinates limits the range of $\theta$ to $\pm 90$ degrees and $\rho$ to the length of the image diagonal $N\sqrt{2}$ (for an $N \times N$ image).

- In this representation, each point $(x_i, y_i)$ in the image yields a sinusoid in the $(\rho, \theta)$ plane
- If we have $m$ collinear points on a line $\rho = x_i \cos\theta + y_i \sin\theta$ then we obtain $m$ sinusoidal curves that intersect at $(\rho_i, \theta_i)$

The algorithm in polar coordinates is similar to the previous one:

- Partition the plane $(\rho, \theta)$ into accumulator cells $A[\rho, \theta]$
- Initialize all cells to zero (cell at $(i, j)$ corresponds to parameter values $(\rho_i, \theta_j)$ )
- For each edge point $(x_k, y_k)$ do
  - Let $\theta_j$ take all the possible $\theta$ values
  - Solve for $\rho$ as $\rho = x_k \cos\theta_j + y_k \sin\theta_j$
  - Round $\rho$ to the closest cell value $\rho_i$
  - Add one to $A[i, j]$

- At the end of the iteration, $A[i, j]=c$ means that $c$ edge points fit the line $\rho_i = x\cos\theta_j + y\sin\theta_j$