Hough Transform

Neal Bayya, George Tang, & Neil Thistlethwaite

November 2018

1 Introduction

The Hough Transform is a feature-extraction algorithm for identifying shapes in images, classically lines. The general process involves a "vote casting", whereby each edge pixel that is being considered to be part of a shape is allowed to cast "votes" in parameter space. At the end of the process the possible shapes (parameter configurations) with the most votes are considered to be the ones that actually exist in the image. This effectively reduces the problem by allowing possibilities to be checked only once, rather than once for every time a candidate is visited.

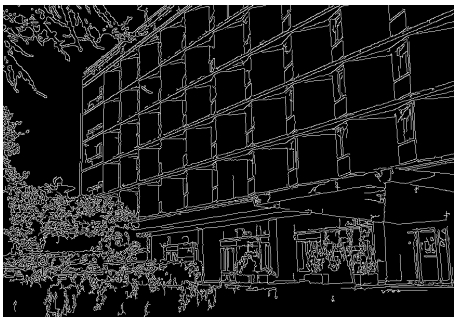

Figure 1: Original Image

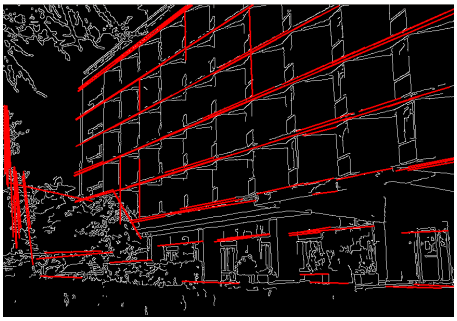

Figure 2: Lines Detected via Hough Transform

2 Line Detection

As an example, let us consider the simple case of line detection. First, we must define our parameter space that the points will vote in. Each point in the parameter space must correspond to a unique line, thus one clear choice would be to represent the lines in slope-intercept form, using (m, c) as the parameter space. Interestingly enough, if we draw a line in (m, c) for each edge pixel (defined by an x and y), then the intersection will tell us the fit a slope and intercept to the line. Refer to Figure 3 for a visualization of line fitting with the (m, c) space. One problem we run into is that the space (m, c) is not bounded; near-vertical lines cause the slope to go to infinity.

Our solution for this is to instead use *Hesse normal form*, which expresses a line as a location in 2D space and a direction that the line extends in. Specifically, we use the form

$$r = x \cos\theta + y \sin\theta \tag{1}$$

Where r and θ define the line as shown in Figure 6. That is, to go from a (r, θ) pair to a line, you extend out from the origin r units at an angle of θ above the x axis (shown in blue), then construct the line perpendicular (shown in red). With a little mathematical manipulation, it can be shown that the provided equation does indeed represent the red line. One neat advantage of this representation is that θ can be computed directly from the gradient angle in edge detection. This reduces the complexity of searching for a fitted line as we no longer have to search over different angles.

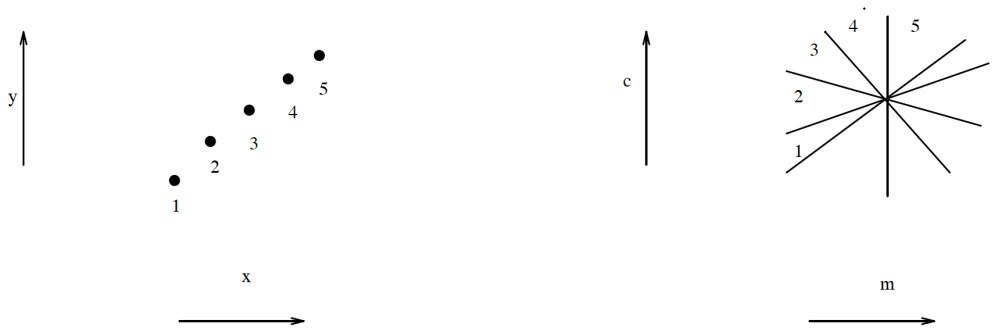

Figure 3: Edge pixels of straight line on left each translate to a line in (m, c) space. Lines intersect at one (m, c) coordinate.

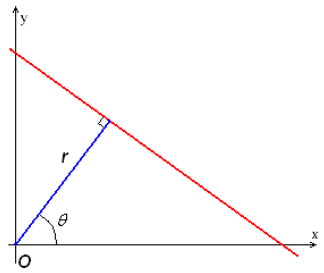

Figure 4: Line Representation in *Hesse normal form*

To perform Hough Transform, we initialize an 2-D accumulator array $A[r, \theta]$ in the parameter space. Each edge pixel will cast a number of votes for specific (r, θ) pairs into the accumulator function to model the line. At the end, the point maxima in the accumulator will represent lines. Since there's an infinite choices for r and θ, we discretize the accumulator by picking a finite number of angles and values of r to consider. In other words, the accumulator array is discretized by making bins of some size when casting votes; this size is usually dependent on and directly relates to the size of the image you are dealing with. Refer to Algorithm 1 for the implementation details of finding the lines in an image using Hough Transform.

Algorithm 1 Hough Transform Line Detection

function LINEAR HOUGH TRANSFORM
  Initialize an accumulator array in the parameter space $A[r, \theta]$
  for each edge point (x,y) **do**
    $\theta \leftarrow arctan(\frac{S_y}{S_x})$ ▷ S_y, S_x are gradients in y and x directions
    $r = x \cos\theta + y \sin\theta$
    $A[r, \theta] += 1$ ▷ Increment the accumulator
  Find r, θ parameters of maxima in A[r, θ] ▷ Numer of maxima corresponds to number of lines
  ▷ If necessary, monitor which edge pixels contribute to the determined lines

Notably, the Hough Transform of a single point is a unique sinusoid in the parameter space, each point along the sinuosoid representing a different line that the point could be a part of. Effectively, our parameter representation has changed the problem of detecting lines into detecting intersection points of sine waves, as the points where many sine functions intersect will be the lines that actually exist in the image.

3 Circle Detection

To deepen our applications of Hough Transform, let's try fitting circles in our image. A circle can be represented by the equation $(x-x_0)^2+(y-y_0)^2 = r^2$. Our parameter space would thus be represented by our three unknowns: (x_0, y_0, r). However, we can use θ from the gradient to reduce the complexity in this case as well. The center location (x_0, y_0) can be computed directly from (r, θ) as $(x - rcos(\theta), y - rsin(\theta))$. Using θ we only have to search over varying radius sizes. Algorithm 2 outlines Hough Transform for circles.

Algorithm 2 Hough Transform Circle Detection

function CIRCLE HOUGH TRANSFORM
  Initialize an accumulator array in the parameter space $A[x_0, y_0, r]$
  for each edge point (x,y) **do**
    for r $= r_{min}$; $r \leq r_{max}$ **do**
      $\theta \leftarrow arctan(\frac{S_y}{S_x})$ ▷ S_y, S_x are gradients in y and x directions
      $x_0 = x - rcos(\theta)$
      $y_0 = y - rsin(\theta)$
      $r = x\cos\theta + y\sin\theta$
      $A[x_0, y_0, r] += 1$ ▷ Increment the accumulator
  Find x_0, y_0, r parameters of maxima in $A[x_0, y_0, r]$ ▷ maxima corresponds to circles
  ▷ If necessary, monitor which edge pixels contribute to the determined circles

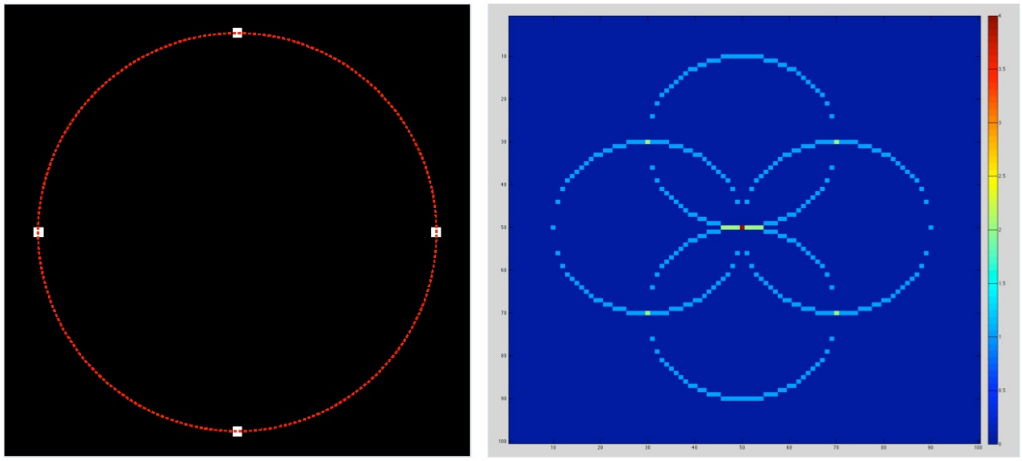

Figure 5: Four edge points on a circle processed using Hough Transform

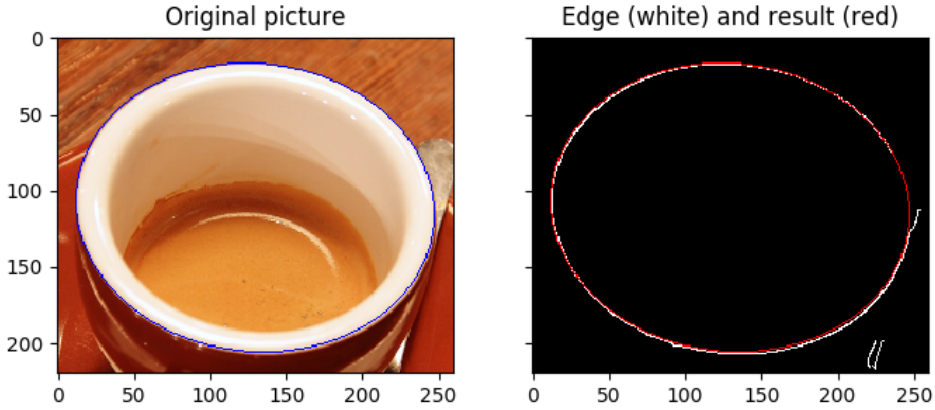

Figure 6: Hough Transform to detect coffee mug

4 Generalized Shape Detection

As it turns out, the Hough Transform can be generalized for detecting any parameterizable shape in an image. The key is choosing a well-defined parameter space (for instance, without any of the infinite discontinuities in the (m, b) parameterization of lines). Using these steps,you can easily fit edges to complex shapes such as ellipses. Algorithm 3 outlines the general procedure for applying Hough to various analytical shapes.

Algorithm 3 General Hough Transform Detection

function GENERAL HOUGH TRANSFORM
  Define parameter space P
  Initialize an accumulator array in the parameter space $A[P]$
  for each edge point (x,y) **do**
    for $p_1 = p_{1min}$; $p_1 \leq p_{1max}$ **do**
      for $p_2 = p_{2min}$; $p_2 \leq p_{2max}$ **do**

        ... ▷ Nest for loops until all parameters can be identified
        Find parameter values $p_1, p_2, p_3, ...p_n$ from x, y and the iterated variables
        $A[p_1, p_2, p_3, ...p_n] += 1$ ▷ Increment the accumulator
  Find $p_1, p_2, p_3, ...p_n$ parameters of maxima in $A[P]$

5 Shape Constraints

It is also possible to set constraints on the shape that the Hough Transform detects. For instance, in the example of circles, suppose we only wanted to detect circles with a diameter between 30 pixels and 50 pixels. We can impose this constraint by simply restricting the domain of r in the accumulator to $[30, 50]$. Anything outside will not be incremented (or iterated through in the innermost loop). Likewise, we could impose a constraint on θ in the line detection example if we only wanted to detect lines in certain directions.

6 Summary

The Hough Transform is a powerful and robust algorithm for shape detection when *you know what you're looking for*, with many applications from something as contrived as detecting the rungs of a ladder to detecting street signs or markings on the road in an autonomous vehicle.