YONSEI UNIVERSITY
Department of Computer Science

**CSI4116 Computer Vision, SPRING 2018**

**Lab 06**

**DUE:** 2018.05.18 11:59:59pm

**Student ID:** 2017843466

**Student Name:** Jimenez-Loza, Jibram Yajaciel

This paper will outline Jibram Jimenez-Loza's attempt at the Hough Transform.

- Basic idea
- The algorithm
- Examples

# 1 Basic idea

As outlined in the class notes, we start off with an edge detected image that gives a general idea of what objects are in the image. However, we want the set of pixels that make up straight lines as these can be very useful features of an image. Straight lines use the for y=mx+b and rewriting it can give us the b = -mx+y equation where we have x and y as parameters (as an image's pixel locations are described by two variables x and y). As every individual point can be described by an infinite many of lines, it is best to consider all the arrangments of a line on the single point and pick the line with the most intersections with other points on the image, creating the best line possible.

# 2 The algorithm

After messing around with the included off-the-shelf edge detection functions, the scikit-image.feature.canny one resulted in the nicest, albeit still rather awful, results. It was fairly close but the lines are fairly thick and unwieldy as well as one important line is missing from the points image. However, after messing with some of the hyperparameters, it seems the lines thin away after changing a few if the values. I simply changed the sigma of the canny filter and then ran this image through the hough transform.

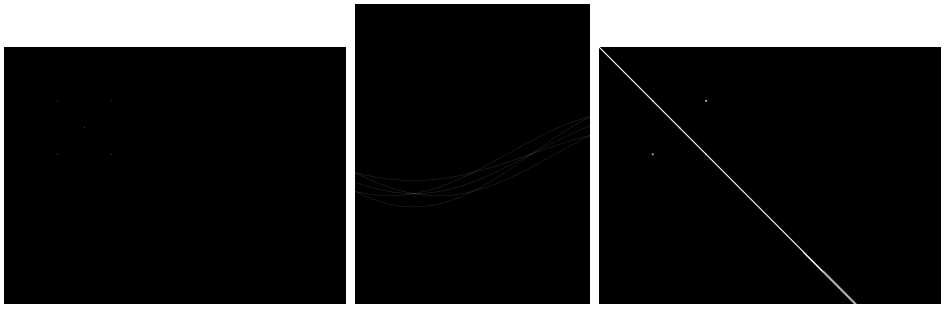
The algorithm seemed to be generalized by these following steps

- Edge detect an image
- Prepare hough space
- Run hough transform
- Threshold
- Draw lines

Since we can't simply model a bunch of values in the y = mx + b form, we can use polar coordinates that use sinusoidal functions to create our mapping for each pixel. We then draw the lines over the image.

## 3 Examples

The following images are the original points image, the hough space of said image, and the result of the hough transform ran on the image.



the following images are the original rect image, the hough space of said image, and the result of the hough transform ran on the image.