CS512 Homework 3 - Report

Lin Zhuo

A20379609

1. Problem statement

In this assignment the image processing is applied on the images containing straighlines. The objective is to apply Hough transform for obtaining the parameter plane, which has the parameters of distance and angle in polar coordinate.

The Hough transform is helpful for detecting the prodominent straight lines in the image and the approach of model fitting is also to be applied.

2. Methods

Detect edge pixels to generate a binary edge image.

In this procedure the original image is transformed to a resultant binary image in which the detected edges are gray scale color 255 (white) and the rest area are 0 (black).

Two approaches were applied((E.g. using Canny, Sobel). The binary edge image can be obtained by convolving the gray scale image with Sobel filter in the x and y directions together with normalizing and mapping the image gray scale value to 0 or 255 with a specific threshold.

Differences of binary edge detection were checked and used in the following parts by applying the openCV function `cv2.Canny()`.

Apply the Hough transform to detect straight lines.

After detecting the binary edges, in this step polar coordinate in the parameter plane is used to apply the Hough transform. Main steps are:

- Determine the range of ρ and θ. ρ ranges from -max_dist to max_dist where max_dist is the diagonal length of the input image. θ ranges from $-90°$ to $90°$. By default, there are 180 bins and each bin represent one degree. It is possible to adjust the number of bins in the next steps for the tradeoff of accuracy, space and speed.
- Hough accumulator of ρ vs θ. The accumulator of the parameter plane is a 2D array with the number of rows equal to the number of ρ values and the number of columns equal to the number of θ values.

- Voting in the accumulator. For each candidate point in the image and for each θ value, find the nearest ρ value (binned the value to a certain bin) and increment the value of that index in the accumulator. Each element tells how many points/pixels contributed "votes" for potential line candidates with parameters (ρ, θ).
- Peak finding. Local maxima in the accumulator indicate the parameters of the most prominent lines in the input image. Here the peaks can be detected by applying a threshold or a relative threshold (values equal to or greater than some fixed percentage of the global maximum value).

Determine the edge pixels belonging to each detected line and refine the line parameters estimate using least square error fitting.

Employ a threshold of distance to assign the pixels to a specific point in the parameter plane which is corresponding to a line in the image.

Assign a point to belong to a line when its distance to the line is within the threshold.

The explicit line can be written as $ax + by + c = 0$.

The parameters are $l = [a, b, c]^T$.

The objective is to get the sum of all the distances of the points to the above line model:

$E(l) = \Sigma(l^T x_i)^2$

that is, $\partial E(l)/\partial l = 0$

$E(l) = \Sigma_i(l^T x_i)^2 = \Sigma l^T x_i x_i^T l = l^T \Sigma(x_i x_i^T) l$

$\frac{\partial E}{\partial l} = 2\Sigma(x_i x_i^T) l = 0$

The equation to solve is:

$$\Sigma(x_i x_i^T) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

The solution is the eigenvector belonging to the eigenvalue that has the smallest absolute value (in ideal case the eigenvalue should be zero.)

Using color draw detected line segments and color the pixels belonging to them.

In the code, a threshold is define to group the points from the binary edge image to be assigned to a certain line or to no lines at all.

For each found lines through Hough transform, it can be expressed as (θ, ρ), the distance from a point in the image (x, y) to it is $d = xcos(\theta) + ysin(\theta)$. If $|\rho - d| < \tau$, the point belongs to the line.

Add a mode: display the result before and after applying least squares refinement.

The comparison is given in the result and analysis section.

Add a mode: display the paramter plane.

The votes obtained from the Hough Transform are normalized to [0,255] and displayed as the gray scale image.

Interactive thresolding added:

- The adjustable hysteresis thresholding: adjust the nunber of edge pixels detected
- The bin size in the Hough transform
- The local maxima determination threshold

3. Result and analysis

3.1 Canny edge detection

In the canny edge detection the two hysteresis thresholds are adjusted interactively. The results of different combination of thresholds are shown below (Figure 1~4).

3.2 Hough transform

The paramter plane is normalized to [0,255] and plotted to the gray scale as shown for example in the following Figure 5.

After hough transform and selecting the local maxima as representation of lines, the lines are drawn onto the image represented by white lines. (Figure 6)

The results of different bin sizes are shown in Figure 7,8,9.

When the bin size is too big, there could be errors in the line direction detection due to the loss of line information.

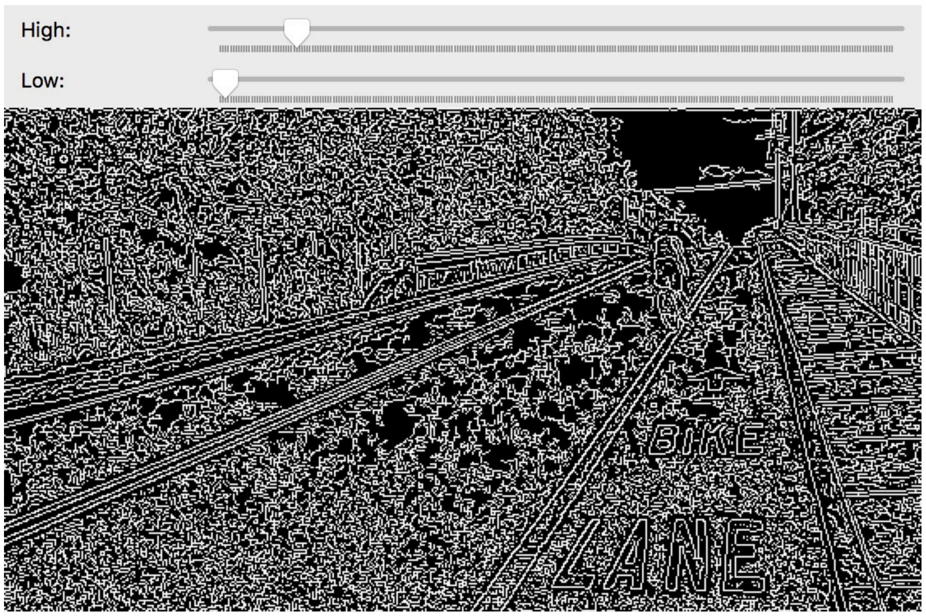

Figure 1: Canny edge detection: low = 5, high = 30

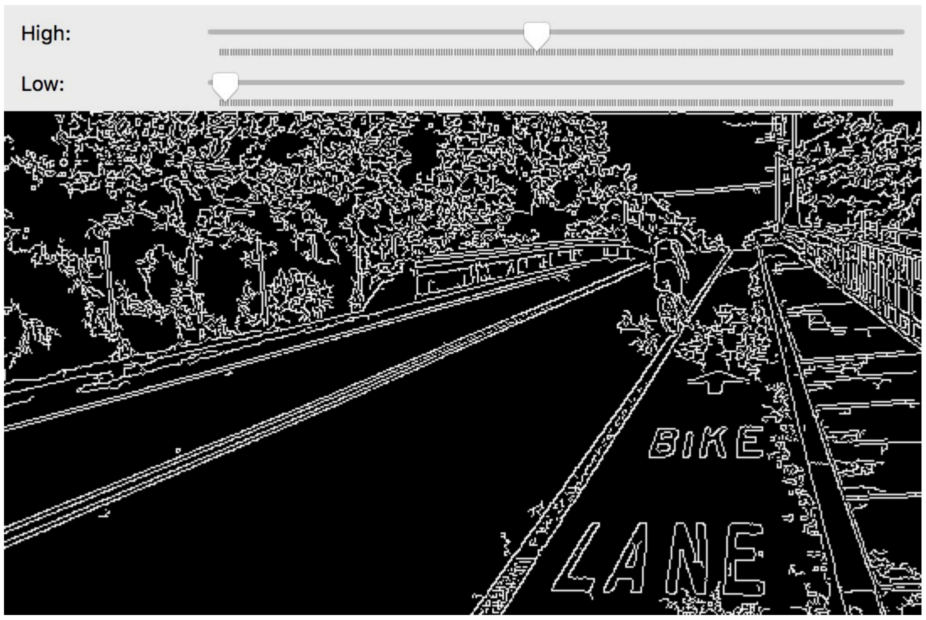

Figure 2: Canny edge detection: low = 5, high = 120

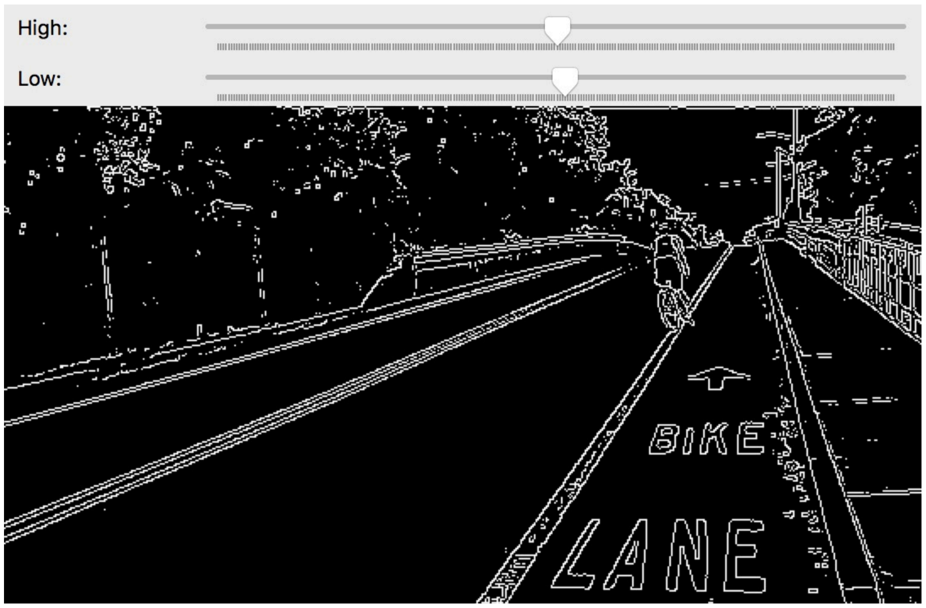

Figure 3: Canny edge detection: low = 120, high = 120

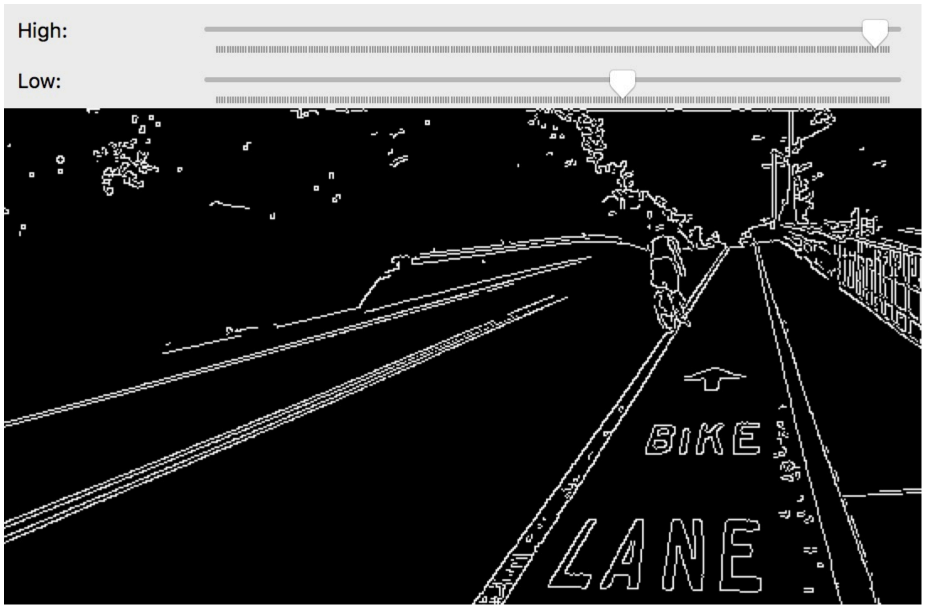

Figure 4: Canny edge detection: low = 150, high = 250

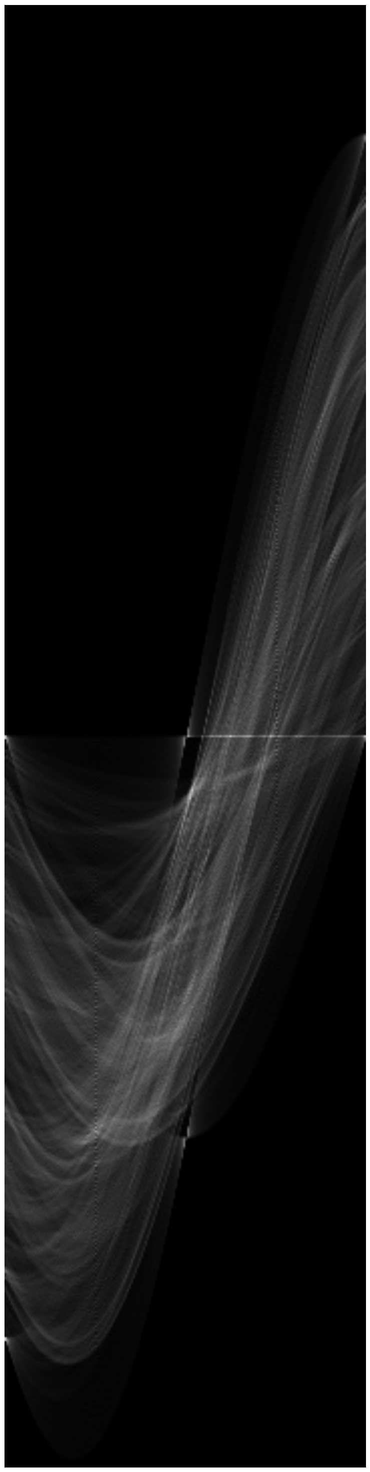

Figure 5: Parameter plane

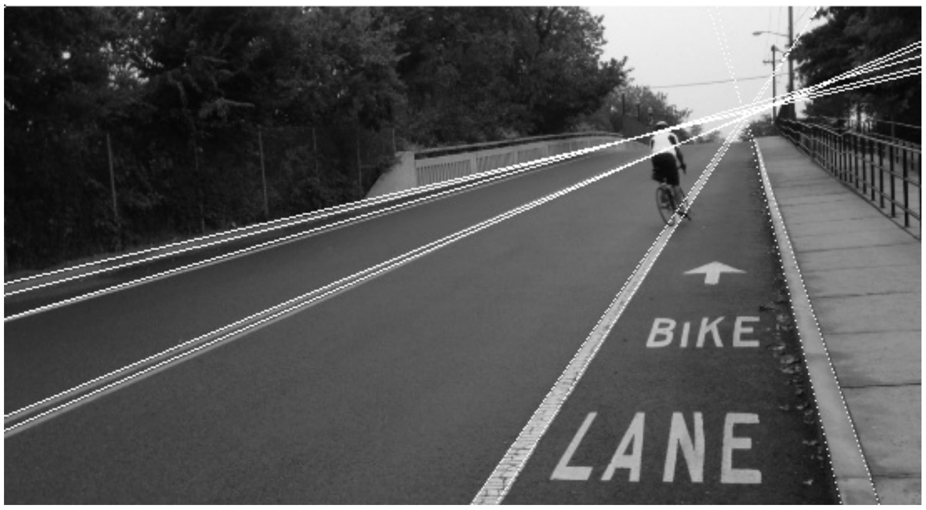

Figure 6: The lines detected by the Hough transform

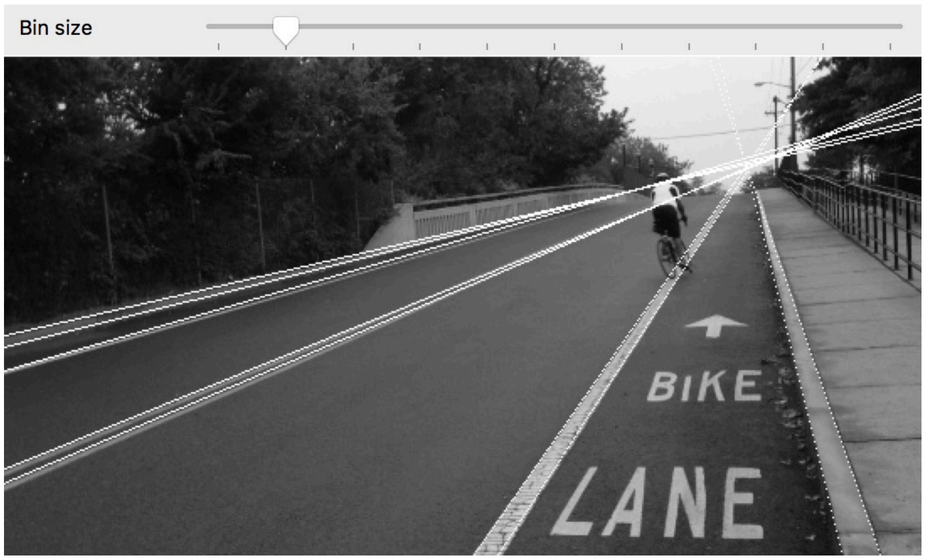

Figure 7: The lines detected by the Hough transform - bin size: 2

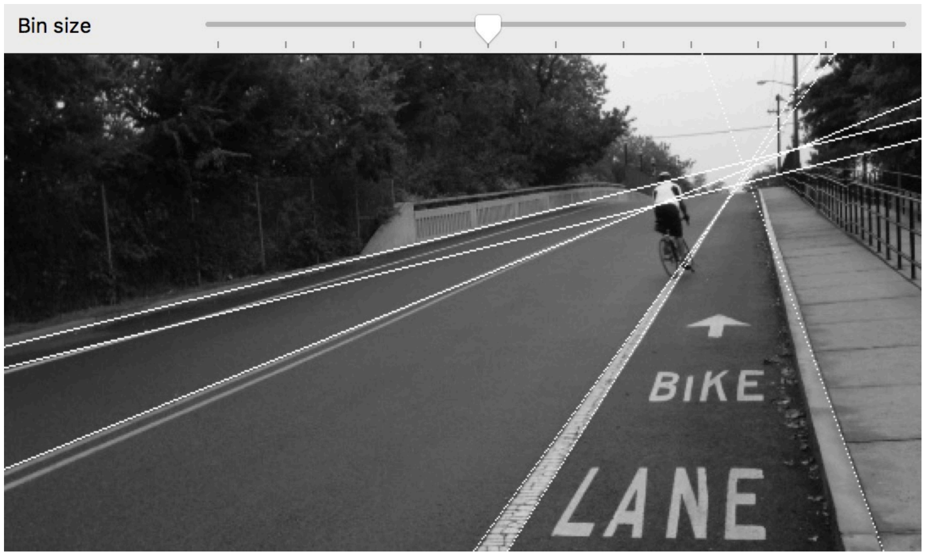

Figure 8: The lines detected by the Hough transform - bin size: 4

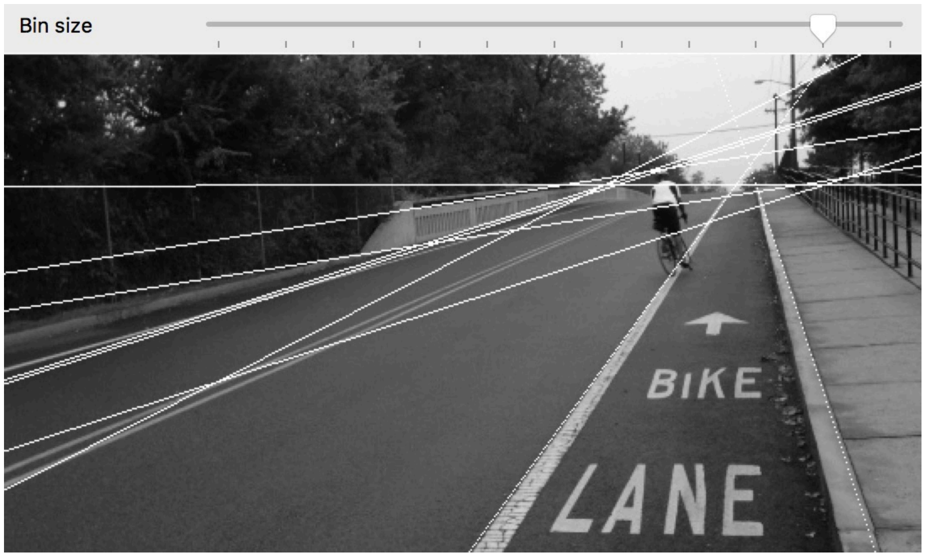

Figure 9: The lines detected by the Hough transform - bin size: 9

3.3 Least Square Error (LSE) Fitting

The points assigned to each line are plotted with different colors and the fitted lines via LSE are in white color. The result is shown in Figure 10.

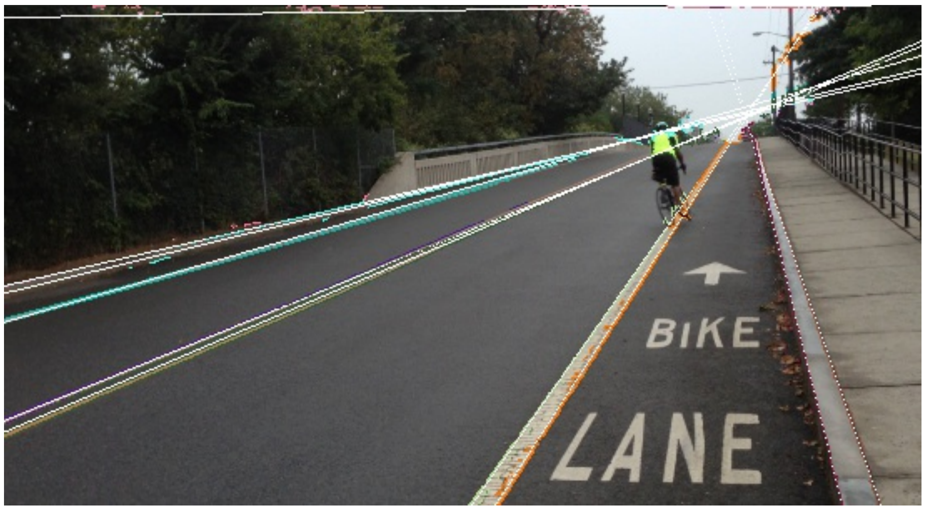

Figure 10: The lines refined by LSE

4. References

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

https://alyssaq.github.io/2014/understanding-hough-transform/