ENPM673: Project 2

Tanuj Thakkar
Master of Engineering in Robotics
University of Maryland - College Park
tanuj@umd.edu

I. Problem 1 - Histogram Equalization

In this part of the project, different histogram equalization methods are implemented to improve the quality of the image sequence. Varying lighting conditions affect the information that can be extracted from an image sequence. With histogram equalization, the contrast and visual appearance of the features in the scene can be enhanced. Two histogram equalization methods are implemented for this part, (1) Simple Histogram Equalization and (2) Adaptive Histogram Equalization.

Fig. 1: Sample used for Histogram Equalization

A. Simple Histogram Equalization

Simple histogram equalization is a technique to make the histogram of the entire image as uniform as possible. This will increase the dynamic range of the image and produce and appearance of high contrast. The following steps were implemented -

1) Since we only care about the intensity values of the pixels and not the color/hue/saturation values, we convert the image to the HSV color space as shown in Fig. 2

Fig. 2: Image in HSV color space

2) The Value (V) channel of the HSV color space holds the intensity information of the capture image. Therefore, we only apply histogram equalization to this channel shown in Fig. 3
3) We compute the histogram of pixel values from the V channel as shown in Fig. 4

Fig. 3: V channel of the image

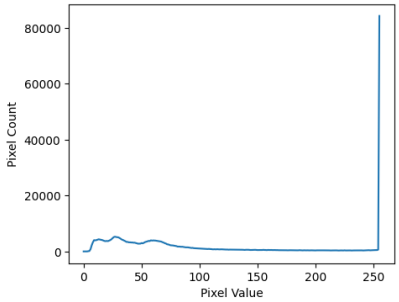

Fig. 4: Histogram of V channel

4) After computing the histogram, we estimate the cumulative density function (CDF) using the computed histogram. The CDF is basically a normalized cumulative sum of the histogram. The CDF is then used to estimate the equalized pixel intensity values to produce a uniform histogram as shown in Fig. 5

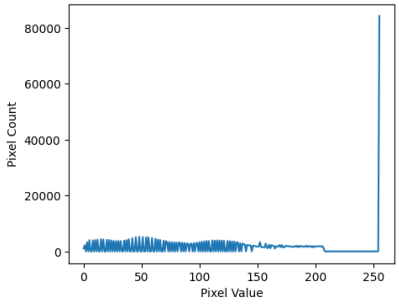

Fig. 5: Equalized Histogram of V channel

5) We then replace the estimated V channel in the original HSV image and convert it back to obtain a BGR image as shown in Fig. 6

It can be observed that the result obtained through simple histogram equalization produces and image which has much

Fig. 6: Simple Histogram Equalization Result

brighter feature and lighter shadows. The result of the entire image sequence is available here.

B. Adaptive Histogram Equalization (AHE)

The simple histogram equalization technique works but when the scene contains regions that are much lighters or darker than the majority of the image, those regions will not be efficiently enhanced. To tackle this issue, the adaptive histogram equalization method is used. In AHE, instead of using the same transformation function (or the CDF) for the entire image, we divide the image into blocks and apply histogram equalization on each block shown in Fig. 7.

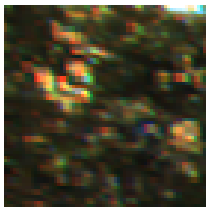

Fig. 7: 64×64 Block

After applying histogram equalization on each block, the result shown in Fig. 8 is obtained for adaptive histogram equalization (AHE).

Fig. 8: Adaptive Histogram Equalization Result

The problem here is the blocks tend to get over-amplified. For this, Contrast Limited Adaptive Histogram Equalization (CLAHE) is implemented. The following steps were followed to implement CLAHE -

1) When computing the histogram for each block shown in Fig. 7, we set a clipping limit over which we clip the count of that particular pixel value and store the additional count in a common residue. In the end, the residue is normalized and added to the entire histogram to produce Fig. 9
2) CDF similar to simple histogram equalization is used to produce equalized histogram as shown in Fig. 10

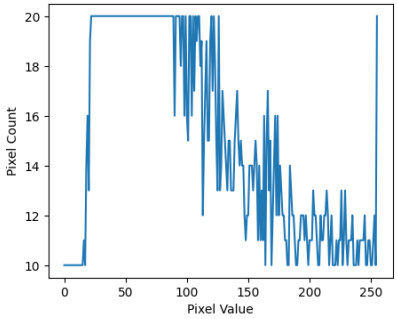

Fig. 9: Contrast Limited Histogram

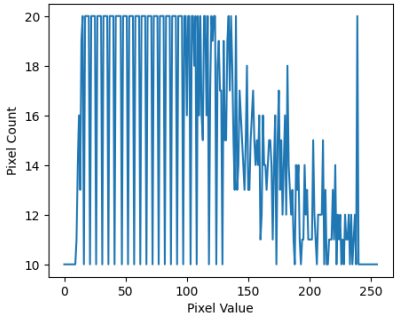

Fig. 10: CLAHE Histogram

3) Repeating for all the blocks of the image, the result obtained through CLAHE is shown in Fig. 11

Fig. 11: CLAHE Result

In comparison to Fig. 6, the result of CLAHE seem to be more uniform, however the regions seem to have less contrast. The performance of the algorithm also depends on the clipping limit and the block size used. The result of the entire image sequence for AHE is available here, and for CLAHE is available here.

II. Problem 2 - Lane Detection

This problem includes a video sequence of a vehicle driving on a straight lane. The challenge is to detect the solid and dashed lane markings on the road. The following pipeline was implemented to solve this task -

1) To filter edges of the lane lines, we first convert the frame to a gray-scale color code and use Canny edge-detection technique to obtain the edges in the scene. However, to remove noise, we first apply Gaussian blur to the gray-scale frame.

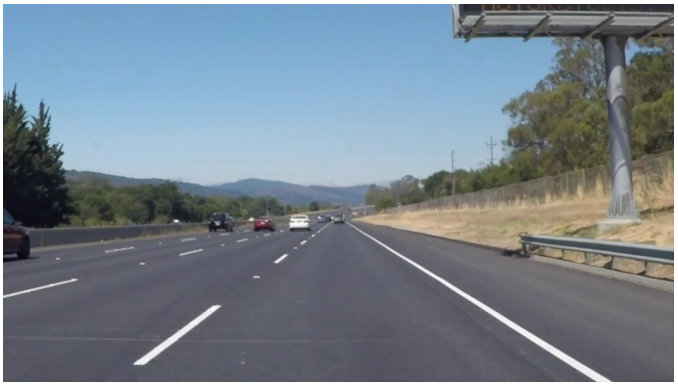

Fig. 12: Sample used for Lane Detection

Fig. 13: Canny Edge-Detection

2) We then mask out the region of interest (ROI) to get Fig. 14

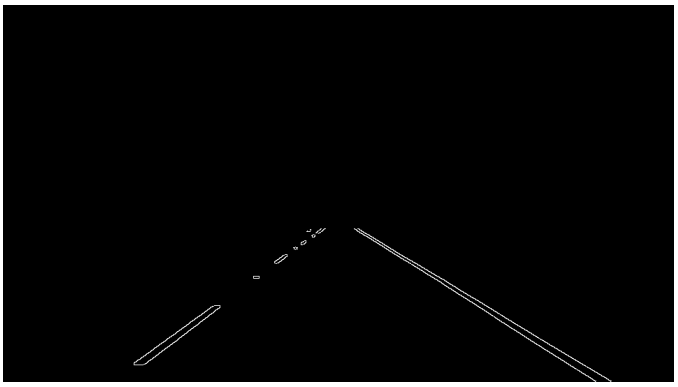

Fig. 14: Region of Interest

3) After obtain the ROI, we apply hough transform to obtain hough lines as shown in Fig. 15.
Hough Transform & Hough Lines: Using the Hough Transform, straight lines can be detected. Lines are expressed in polar coordinates, (r, θ), in Hough Transform. For each point, there is a family of lines passing through it. When we plot the possible lines represented using (r, θ) passing through a point (x, y), we get a curve. When two curves for different points intersect, the probability of them being on the same straight line increases. This is how Hough Lines work and are employed to detect straight lines.

Fig. 15: Hough Lines

4) The obtained frame lines are to be classified into solid and dashed lane lines. To achieve this, the slope of the longest Hough line which in most cases belongs to the solid lane marking, S, is computed. Then, the direction of the slope, i.e., whether the slope is positive or negative, of all the other Hough lines, s_i, are compared with S and classified accordingly.
5) After classifying all the Hough lines into solid and dashed lane markings, we fit straight lines on the data to obtain a single lane marking for each class. For this, endpoints of each Hough line are used and a straight line is fit on them to obtain the result shown in Fig. 16

Fig. 16: Lane Detection Result

The result of the entire video sequence is available here.

III. Problem 3 - Curvature Estimation

In this part of the project, again video sequence involving a vehicle driving on the freeway has to be processed to estimate the curvature of the lane. To accomplish this, the following pipeline was enforced -

1) First, to detect yellow and white lane line markings as shown in Fig. 17, we convert the frame from BGR to HSV color space. Since the lane line markings have a

Fig. 17: Sample used for Curvature Estimation

high saturation, we can use the S channel and perform thresholding to filter out regions with high saturation. Additionally, both yellow and white colors have a high RED color intensity which can also be used to achieve more accuracy. Finally, a bitwise-and operation is performed between the two thresholds to obtain Fig. 18

Fig. 18: Lane Marking Filtering

2) Next is perspective correction. Due to the way computational photography works, the camera's perspective from the way it is mounted is not the correct representation on the real world, similar to what we see from our eyes. To obtain the correct perspective, whether the lane is a straight one or curved, we need a birds-eye-view. For this, we use homography to transform the perspective and achieve Fig. 19. As it can be observed, now the curve of the lane is visible.
3) To detect the locations of the lane line markings, histogram of Fig. 19 shown in Fig. 20 is used. The peaks of the histogram provide the center locations of both the lane lines.
4) Utilizing the locations of both the lane lines, a sliding window search is performed across the height of Fig. 19, to estimate local means of the curves as shown in Fig. 21
5) The local means previously estimated are then used to fit curves through the lane lines as shown in Fig. 22. To

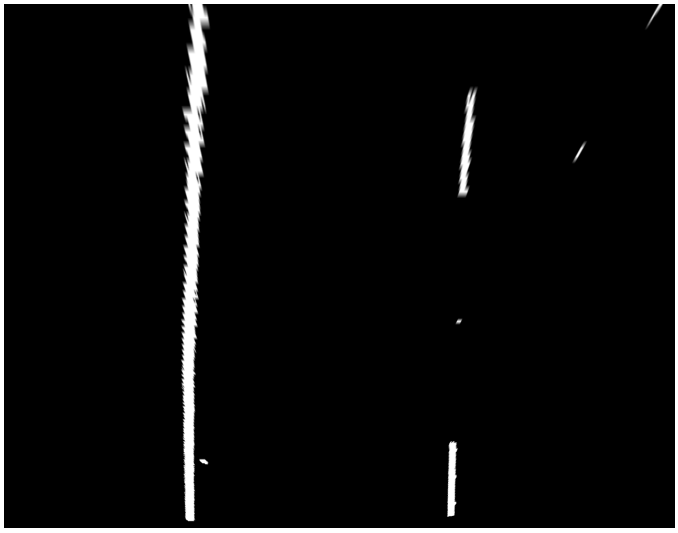

Fig. 19: Birds Eye View

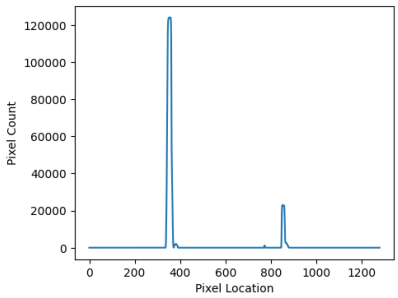

Fig. 20: Birds Eye View Histogram

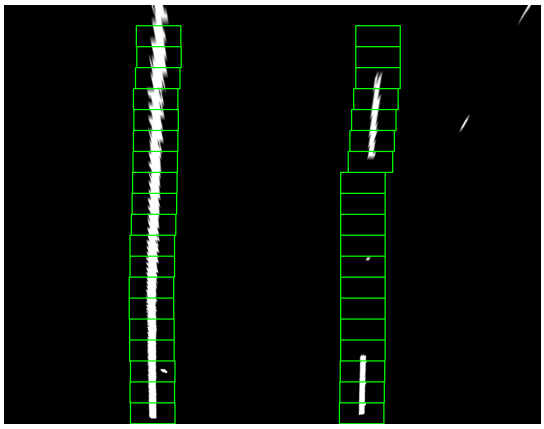

Fig. 21: Sliding Window Search

deal with noise in the data, we take a sliding average over the last N frames to estimate the coefficients of the curve. This will also help tackle situations were the lane filtering part of the pipeline isn't able to produce accurate results.
6) Using the equation of a second-degree polynomial, we can then compute the radius of curvature of the lane lines as follows,

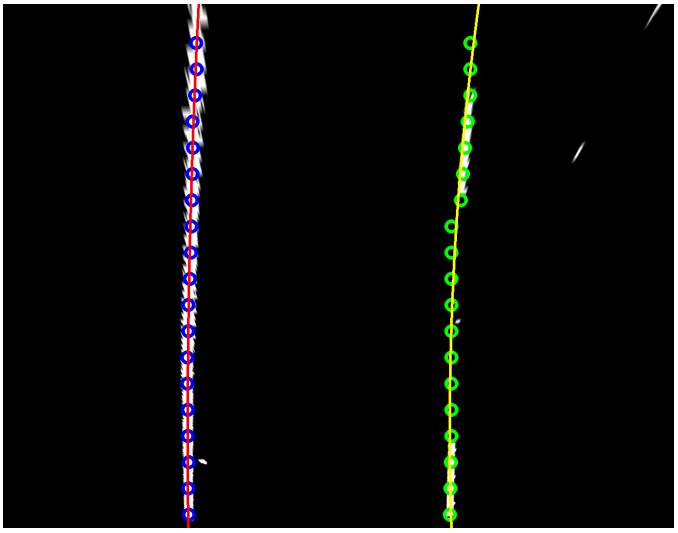

Fig. 22: Sliding Window Search

$$R = \frac{(1 + (\frac{dy}{dx})^2)^{3/2}}{|\frac{d^2y}{dx^2}|}$$

where, $y = ax^2 + bx + c$.

7) To project the lane back to the original frame, we use inverse homography, i.e., from birds-eye-view to the original perspective as shown in Fig. 23

Fig. 23: Sliding Window Search

The result of the entire video sequence is available here.

REFERENCES

[1]. Histogram Equalization and CLAHE
[2]. Hough Transform
[3]. Curvature Estimation