Department of Electrical & Computer Engineering
ELG5378 Image Processing and Image Communication

Edge Detection and Its Application in Lane Detection

Prepared by: Yinruo Jiang & Wei Cao
Student Number: 3002748145 & 7495980
Instructor: Professor Jiying Zhao
Date: April 19, 2022

Abstract

Both the advanced driver assistance system and self-driving systems require the application of edge detection as its fundamental component in the lane-detection system. Detection of important features (e.g., lines, curves and corners) from the image can be directly applied to the lane detection in the intelligent vehicle system to improve safety on the road. This paper studies and reviews various commonly used edge detection methods and some feature extraction techniques to identify lane markings on the road in the real time. We implemented two different approaches referenced from two papers, respectively. Compared with the first conventional lane detection approach, the second advanced approach has the ability to work consistently in challenging conditions with better robustness.

Contents

1 Introduction

Images are ubiquitous in our lives. Edge detection is one of important research areas that has widespread applications in various fields, such as feature extraction, patter recognition, and so on. We depend on edge detection for recognizing the contours of the fingerprint, detecting anomalies in the medical images, and identifying the lane line on the road. These examples, and many others that readily come to mind, are ample proof of the importance of edge detection in medicine, science and technology.

Considering that at least 90% of all auto accidents originate from human errors [14] and that, on average, crashes resulting from unintentionally drifting out of the lane can be one of important contributing factors, it is not difficult to accept that today many passenger vehicles are equipped with high levels of automatic safety technologies to improve safety on roads, such as lane deviation warning, automatic parking and collision avoidance. Lane detection as one of the core application components is heavily executed in these advanced driver assistance systems and also in the self-driving systems.

A digital image is a two-dimensional matrix of pixels for simplicity, and the value of each pixel represents the brightness of the corresponding point in an image. Edge is the place where has abrupt changes in intensity values, and it usually occurs between the boundary of objects. The edge detection method is the operation of detecting abrupt changes in pixel intensity to determine an edge. In our report, to generate the initial results on the lane detection, we first followed the conventional approach based on Canny edge detection and Hough transform that is sourced from this paper "Simple Robust Road Lane Detection Algorithm" [9]. Furthermore, in order to improve the algorithm robustness, we researched and implemented the second advanced lane detection approach based on perspective transformations and histogram analysis that was proposed in this paper "Lane detection technique based on perspective transformation and histogram analysis for self-driving cars"[11]. We conducted our experiments on this widely used dataset of *CarND-Advanced-Lane-Lines* from Udacity[12].

This paper consists of five sections. Section 1 is the introduction. Section 2 studies and reviews various edge detection techniques. Section 3 details the implementation of two different approaches. The experimental results are discussed in Section 4 and conclusion in section 5.

2 Literature Review

Most of the edge detection methods address the following three main problems [3]. First, noise in a digital image can decrease the image quality and distort the processing results, making it difficult to find accurate boundary of an object. To remove noise caused by high-frequency signals, the image intensities need to be smoothed by a low-pass filter to suppress noise. Second, the differentiation technique amplifies and sharpens the edges and creates more easily detectable edge candidates. Third, the threshold selection needs to be established to determine which edge pixels should be discarded as noise and which should be retained.

The main properties of robust lane detection techniques should possess the following merits [15]. First, the lane can be detected under most of challenging scenarios , such as severe occlusion and extreme lighting conditions. Second, it should be flexible to handle not only the straight roads but also the curved ones. Third, it should use the parallel constraint as a guidance to improve the detection in terms of robustness to noise. Lastly, it should produce consistent and reliable results quantitatively.

The following subsections review several literature on the topics of edge detection methods that are commonly used to detect lane lines on the road.

2.1 Gradient-based edge detection

The essential idea behind this technique is to find pixels that have large gradient magnitude on the first derivative of the image intensity.

The gradient is a measure of change in a function. Edge strength can be calculated by a gradient vector which packages G_x and G_y, which are the partial derivatives of intensity along the x-axis and y-axis directions of a two-dimensional image function $f(x, y)$ [13]. In other words, the gradient's components measure how significant pixel values are changing with distance in the x and y directions.

Gradient vector is given by:

$$\nabla f = grad(f) = \begin{bmatrix} Gx \\ Gy \end{bmatrix} \tag{1}$$

The gradient vector magnitude is the potential edge identified at a pixel location of the sampled input image.

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{2}$$

In reality, to calculate the partial derivatives would increase computational costs. Instead, it would be simpler and quicker to calculate the gradient vector magnitude by simple addition and subtraction of edge operators on top of the input image.

The Sobel operator[1] is one of the most commonly used edge detectors.

The mask of Sobel operator is given as:

$$Gx = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, Gy = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3}$$

Figure 2.1 shows results of identifying edges of a panda using a Sobel operator under two scenarios. The image (a) is the smoothed image using the Gaussian filter while the image (b) is the input image when the filtering step is omitted. The output results (b) and (d) demonstrated that many false edges would be detected as a result of the noise.

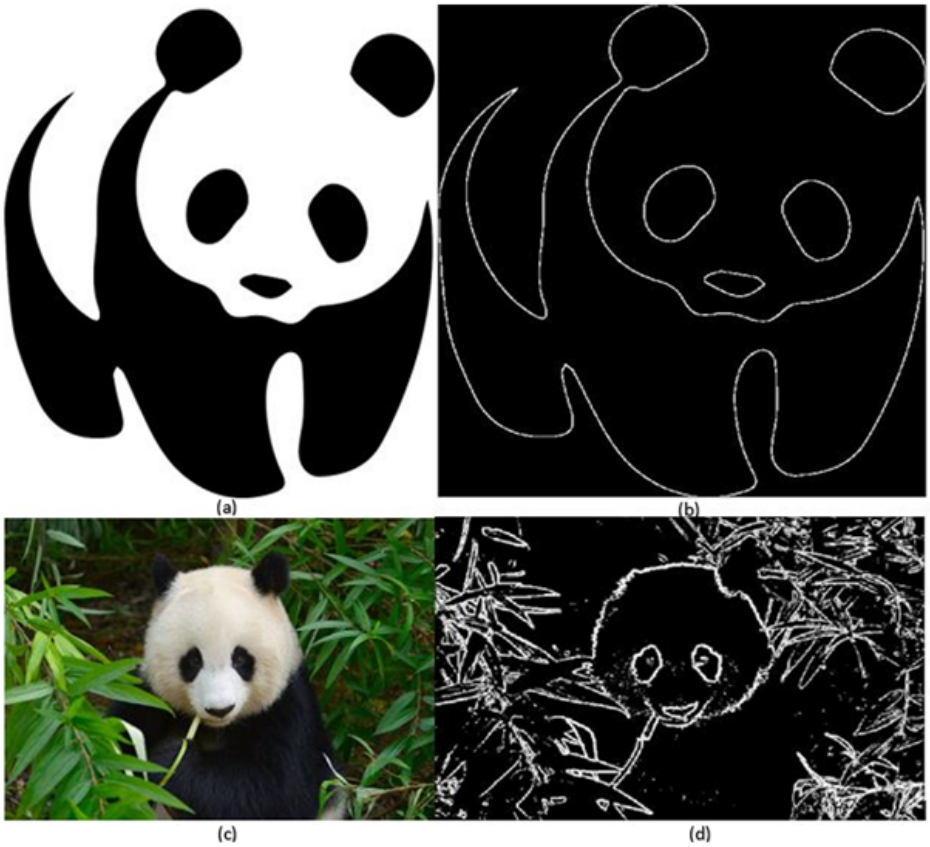

Figure 2.1: Example using the Sobel operator to detect edges of a panda. (a) Filtered image. (b) Output of the filtered image. (c) Original image. (d) Output of the original image.

2.2 Laplacian of Gaussian

The Laplacian method highlights the location of an edge in zero crossing by calculating the second derivative of an image intensity and shows a stronger response in the edge detection [1], compared with the first derivative method showing a maximum or minimum peak corresponding to the center of the edge in the original image. However, all the gradient-based techniques are very sensitive to noise. To avoid the effect of noise,

the scientist Marr and Hildreth [10] suggested to combine the Gaussian filter with the Laplacian operator for edge detection. The Gaussian filter can smooth an image and reduce distorted noise.

The output of the Laplacian of Gaussian operator $\nabla^2 \mathrm{G}$ is given as:

$$\nabla^2 \mathrm{G} = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] \tag{4}$$

where ∇^2 is the Laplacian operator and G is a 2-D Gaussian filter function. The value of σ decides the amount of blurring in the Gaussian filter.

Although the Gaussian smoothing filter removes the high frequency noise, it also smooths the edges and other sharp intensity discontinuities in an image. If the value of σ becomes larger, the better the result of noise removal but at the same time the image would also lose important edge information, ultimately affecting the performance of an edge detector.

2.3 Canny Edge Detection

The Canny edge detection algorithm was developed by John F. Canny in 1986 [4]. This algorithm is divided into five steps. First, the Gaussian filter is used to smooth the image and suppress noise. The second step is to calculate the intensity gradient in the smoothed image by applying the Sobel operator. The next step is to convert potential edges to sharp edges by applying non-maximum suppression to preserve the local maximum and remove unwanted pixels that might not be part of the edge. After that, the pixels that survive to this point are mostly candidates for true edges in the image, but the Canny algorithm uses double thresholding to confirm that edge pixels greater than the upper threshold level are marked as edges; edge pixels less than the lower threshold level are removed and edge pixels between the two thresholds are marked as edges if the pixel is adjacent to a pixel above the upper threshold. Because the Canny approach only marks edge pixels if they possess edge strengths that pass a double-threshold hysteresis criterion, it has become one of the mainstream techniques widely used in many fields as the edges detected are very close to its true edges.

2.4 Hough Transform

The Hough transform (HT) is a widespread technique to detect straight lines and connect discontinuous lines [5]. The HT transforms a sampled input image $f(x, y)$ from the image space with x- and y-axis to a parameter space matrix in the (ρ, θ) domain using the equation as,

$$\rho = x \cos\theta + y \sin\theta, \tag{5}$$

where ρ is the distance from the origin of the closest point of the line, and θ is the angle of the perpendicular projection from the origin to the line.

The HT transforms a point (x, y) of the input image into a sinusoid curve in the parameter space, and the point of intersection from the sinusoid curves in the parameter space matrix becomes a candidate line in the image space. Then, the voting procedure is used within the parameter space matrix in the form of grids to determine the grid with the maximum number of points, which corresponds to the edge pixels in the input image. However, the noise and potential false edges within an image will also be voted through the parametric units of false lines, so the Hough transform is also sensitive to noise. If the noise is removed before using the Hough transform, it is still a power technique to detect and group similar lines together.

2.5 HLS Model

When a colour space is detected and segmented, the Hue, Saturation, Lightness (HLS) color model is more intuitive to represent the color and easier to manipulate for the sake of efficient computation than the Red, Green, Blue (RGB) color model, which describes a color as a combination of the three primary components, may increase computational burden [7][11]. Objects with the specified colors from an image can be easily detected and segmented when the image is in the HLS color space. Given an input image in the RGB model, the first step is to convert to HLS color space using the equations below.

Find the minimum and maximum values of R, G and B.

Calculate the luminance value by adding the maxima and minima and then the total is divided by two.

The next step is to find the saturation

$$V_{\max} \leftarrow max(R, G, B), \tag{6}$$

$$V_{\min} \leftarrow min(R, G, B), \tag{7}$$

$$L \leftarrow \frac{V_{\max} + V_{\min}}{2}, \tag{8}$$

$$S \leftarrow \begin{cases} \frac{V_{\max} - V_{\min}}{V_{\max} + V_{\min}} if(L < 0.5) \\ \frac{V_{\max} - V_{\min}}{2 - V_{\max} - V_{\min}} if(L \geq 0.5) \end{cases} \tag{9}$$

$$H \leftarrow \begin{cases} \frac{G-B}{V_{\max} - V_{\min}} & \text{if } (V_{\max} = R) \\ 120 + \frac{60(B-R)}{V_{\max} - V_{\min}} & \text{if } (V_{\max} = G) \\ 240 + \frac{60(R-G)}{2 - V_{\max} - V_{\min}} & \text{if } (V_{\max} = B) \end{cases} \tag{10}$$

After an input image in RGB color space is transformed into HLS color space, certain desired objects or regions of the converted image that have the specified colors can be detected, segmented and extracted for subsequent processing.

2.6 LAB Model

The LAB model [8], also referred to as the CIELab color space, is defined by the International Commission on Illumination and describes all the colors visible to human eye. is used to segment the complete region. The 'a' channel is the color balance between green and red, and the 'b' channel is the color balance between blue and yellow. The 'L' channel, or Lightness, can be understood as a greyscale and closely match human perception of lightness. Because of this factor, the LAB model can be used to make accurate color balance correction by modifying the 'a' and 'b' components or adjusting the lightness contrast using the 'L' component. When converting RGB colors to any of the CIELab color space, the conversion can only be done indirectly using XYZ as an intermediate mode.

The RGB-to-LAB color space conversion is given as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{11}$$

$$X \leftarrow X/X_n, where X_n = 0.950456, \tag{12}$$

$$Z \leftarrow Z/Z_n, where Z_n = 1.088754, \tag{13}$$

$$L \leftarrow \begin{cases} 116Y^{\frac{1}{3}} - 16 & \text{if } (Y > 0.008856) \\ 903.3Y & \text{if } (Y \leq 0.008856) \end{cases} \tag{14}$$

$$a \leftarrow 500(f(X) - f(Y)) + \delta, \tag{15}$$

$$b \leftarrow 200(f(Y) - f(Z)) + \delta, \tag{16}$$

where

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{if } (t > 0.008856) \\ 7.787t + 16/116 & \text{if } (t \leq 0.008856) \end{cases} \tag{17}$$

and

$$\delta = \begin{cases} 128 & \text{for eight-bit images,} \\ 0 & \text{for floating-point images} \end{cases} \tag{18}$$

2.7 Perspective Transformation

The perspective transformation algorithm [11] can convert the forward and backward perspectives of the source images into a bird's eye view of the whole surrounding area around a vehicle. This technique can effectively help drivers perceive and detect the surrounding environment of the on-road vehicle and predict possible risks and avoid

collisions. The essential idea is to map the pixel coordinates of the region of interest from the source images to the new coordinates of pixels in the destination image.

(x', y') are the transformed coordinates, while (x, y) are the input coordinate points. The transformation matrix (M) can be seen as a combination of rotation, scaling, translation vector, projection vector and so on.

$$\begin{bmatrix} t_{\mathrm{i}} x'_{\mathrm{i}} \\ t_{\mathrm{i}} y'_{\mathrm{i}} \\ t_{\mathrm{i}} \end{bmatrix} = M \cdot \begin{bmatrix} x_{\mathrm{i}} \\ y_{\mathrm{i}} \\ 1 \end{bmatrix} \tag{19}$$

Once the transformation matrix is calculated, we can calculate all the new coordinates to generate the final transformed image using entire input coordinates.

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right), \tag{20}$$

where $dst(x, y)$ are the coordinates of pixels in the destination image and $src(x, y)$ are the coordinates of pixels in the source image.

2.8 Histogram-based segmentation

The technique of histogram-based segmentation [6] is also a very important tool for lane detection as it produces a very clean binary image with most of the lane markers on it. The distribution and growing region of the histogram can be predicted from previous location values and vanishing points. During the process, the system can calculate the mean value of the pixel level distribution of the road, as well as the maximum and minimum values of such distribution. Values above this maximum are detected as lane markers and values below the minimum are assumed to be lane.

2.9 Sliding window search method

The multiple sliding window algorithm technique [2] is to have windows formed over some part of data, and these windows can slide over the data to capture different portions of it. This technique is very important to efficiently detect the moving lanes regardless of the sharp curves and dashed lanes and formulate a set of adjacent pairs. The first step is to identify the maximum probability region of its existence by observing the histogram of the top-down view, so the left and right lanes are shown as peak values in the histogram of the image, and the positions of subsequent sliding windows are determined based on the average value of the point in the current sliding window. The system runs the sliding windows search to capture the pixel coordinates of the lane lines.

3 Implementation Design

3.1 Problem Analysis

In this report, we will use two traditional methods to detect highway lanes in two different conditions. First method mainly detects the single, straight, white, solid highway lane boundaries without shadow [9] and in the normal light intensity. Second method primarily focuses on the yellow and white, curved highway lane boundaries with some shadow and in the normal light intensity.

3.2 Detection Algorithm

In the first condition, we are using the canny operator and Hough transform to detect the lane lines which main character is white and solid. The primary reason behind this choice is white lane lines which has the high contrast with background. Using the Canny operator will have a good performance. In addition, it is common to utilize Hough Transform to detect straight lines.

In the second condition, we are using the HSL and L*a*b model, perspective transformation and histogram analysis to detect curved express lanes. This algorithm is extremely useful for extract information about lane marking, because in the world, the color of most lanes is yellow and white [11].

3.3 Algorithm Implementation

3.3.1 Implementation Steps for First Conventional Method on Detecting Straight Highway Lanes

- Image Capture: the system will capture the color images from camera facing the highway which is behind the windscreen [9].

- Pre-processing: at the stage of pre-processing, system will convert the captured images into gray scale, apply Gaussian filter to suppression noise, and apply canny operator to extract edge.

  - Turning the image into gray scale could lead to the computational reduction, by using this formula:

    $$Grayscale = 0.299R + 0.587G + 0.114B \tag{21}$$

    It could turn three color channels into one. Therefore, the system only needs to compute the pixel with one intensity variable ranging from 0 to 255 which will increase the computational speed. In addition, canny operator "works with monochromatic image" [9].

  - Apply Gaussian filter which its kernel size is 5*5. In the normal light intensity, white lanes have high contrast condition, Gaussian kernel size doesn't need too large. The larger Gaussian filter would use more pixel in the image then blurrier in the image.

- Apply canny operator: Canny Low threshold is to use to remove the unnecessary edge, therefore, for high contrast, we can set it higher, otherwise, we can set it lower. And for high threshold is to suppression the high contrast. In this step, we set the low and high threshold as 50 and 150.

- Region of Interest: manually set the region of interest to reduce computational cost. In this stage, we are using matplotlib (python library) to plot the image with x and y axis and read the position value of region of interest. In this project, we select three points which positions are (200, image height), (1100, image height) and (550, 250). Then set the value of region of interest to 255 and set the rest of region to 0 which forms mask. Using this mask and apply the bit-wise operation "and" between canny image and mask to extract the line image of region of interest. After that operation, the system captures many lines in the image.

- Hough Transformation: to get straight line, we apply Hough transform to "connect discontinuous line and differentiate different lines" [9]. We use the HoughLinesP function in OpenCV, and it involves two important parameters: minLineLength and maxLineGap. minLineLength is used to set how long the line could be line. Make it very small means even a point will be counted.To optimize this algorithm, system collects the slope and intercept of all lines, calculate the average. After that, one smooth straight line will display in the line image.

- Image output: using addWeight function in python to overlap line image to original image. Final video has been uploaded in website: https://www.youtube.com/watch?v=BZkcHg-aUR0

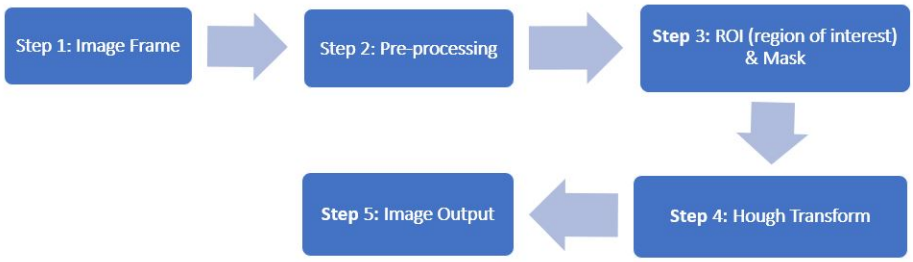

Figure 3.1: General Flowchart of lane detection algorithm

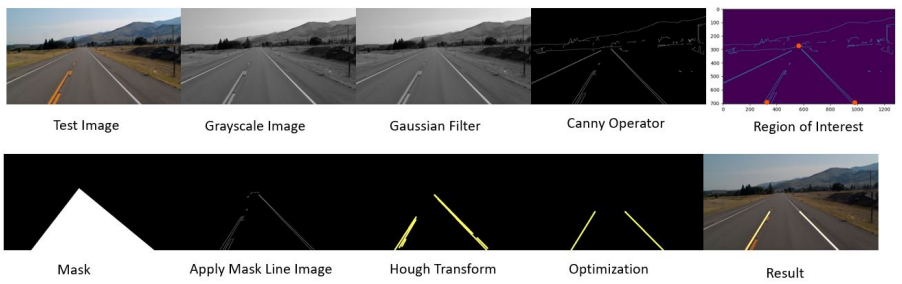

Figure 3.2: The process of implement the conventional method to lane detection

3.3.2 Implementation Steps for Advanced Conventional Method on Detecting Curved Highway Lanes

- Camera Calibration: Due to the property of camera lens, lights lay often lend a little too much at the edges of a curved lens of a camera, and this creates the distortion of the edges of the images. Even though it is not easy for human being to detect, these errors still need to be addressed when developing a robust algorithm [11]. Usually, we use chessboard to do camera calibration as it has regular, high contrast pattern. We use findChessboardCorners() and drawChessboardCorners() function in Python to detect and draw corners. Then we read twenty images to get a reliable calibration. Using calibrateCamera() function to get the camera matrix and distortion coefficient. Figure.3 illustrates the camera calibration using chessboard image.

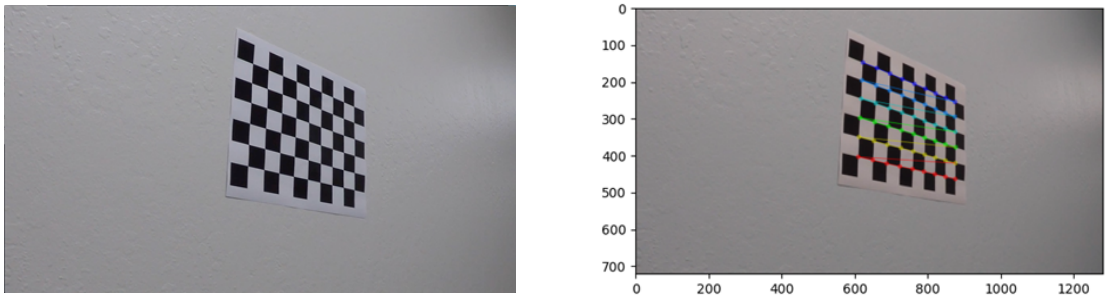

Figure 3.3: Camera Calibration using Chessboard

- Image Distortion Correction: using the undistort() function which parameters are original image, camera matrix and distortion coefficients, and get an undistorted image. Figure.3 illustrates the difference between original image and undistorted image.

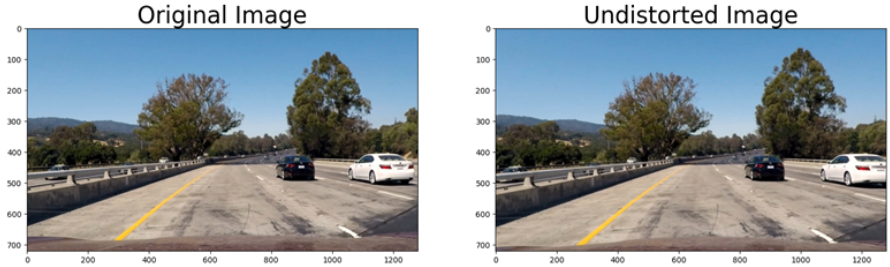

Figure 3.4: Difference between original image and undistorted image

- Perspective Transformation: two parallel lines would converge into a point in the image. Using perspective transformation could improve lane detection, as some features could be identified with higher accuracy in different perspective [11]. Getting the bird-view image enables us to fit a curved line to the lane lines. In addition, this easy for us to select region of interest. In this step, we select quadrangle vertices in the source image as source points and corresponding vertices in the destination image as destination points and using the getPerspectiveTransform() and warpPerspective() function to implement the perspective transformation.

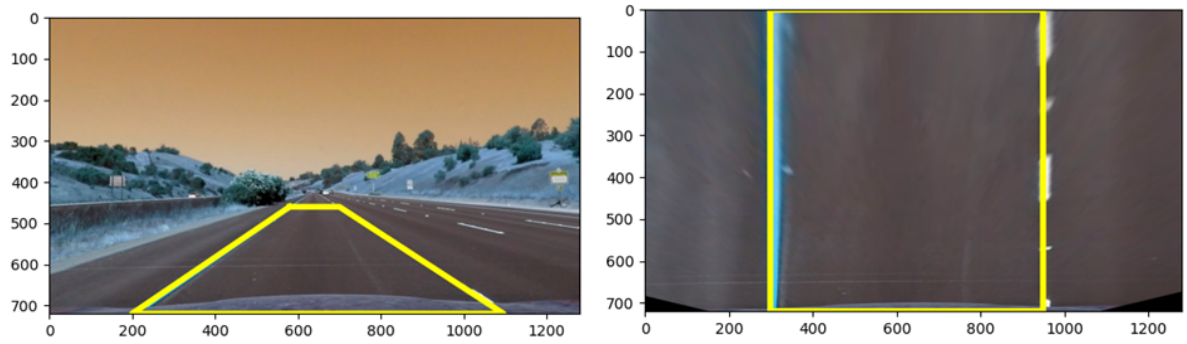

Figure 3.5: Source points and destination points

- HSL and L*a*b Model: The system uses HSL, L channel to extract the white line and L*a*b, b channel to extract yellow line. Then combine these two extract images. This is one of extraction methods, for other scenario, we could apply many different edge detection operators and deep learning method.

Figure 3.6: Color Segmentation

- Histogram of Image: Using the histogram image is to detect the maximum probability of the existence of lane lines [11]. In the following histogram, it means the amount of white points in certain pixel position.We can see from it: the left peak around x=350, which is the left lane line, and right column around x=950, which is right lane line. Therefore we can get the approximate position of left and right lane lines.

Figure 3.7: Histogram Image

- Sliding Window Search Methods: Starting points of two windows (left and right) are the the approximate position in the histogram image [2].The next sliding window positions are defined by the average of points in the current moving windows [2]. However, this method is mainly suitable on slightly curved express lanes, rather than sharp changed road. If the lane has a sharp left or right turn, it is impossible to build the next window, which is based on the average points of current window.

Therefore, it will happen the next window just stack up to the current window which totally not considering the sharp change.In addition,we are using video to track the lane lines, therefore we could use the correlation of previous frame and next frame, as the continuity of video. Then we could improve the stability of lane lines and reduce the computation.

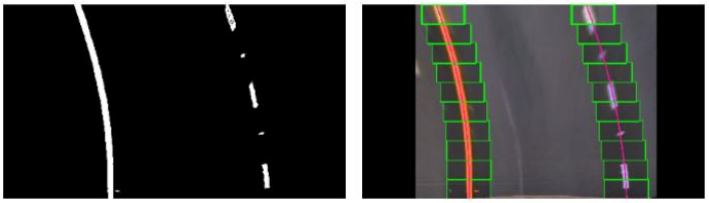

Figure 3.8: Sliding Window

- Inverse Perspective Transform: to inverse the bird-view image which has already draw the lane lines. And overlay the bird view image on undistorted image by using addWeight function. Lastly, output the video. Final video has been uploaded: https://www.youtube.com/watch?v=LrVTdjvNzD0

Figure 3.9: Final Result

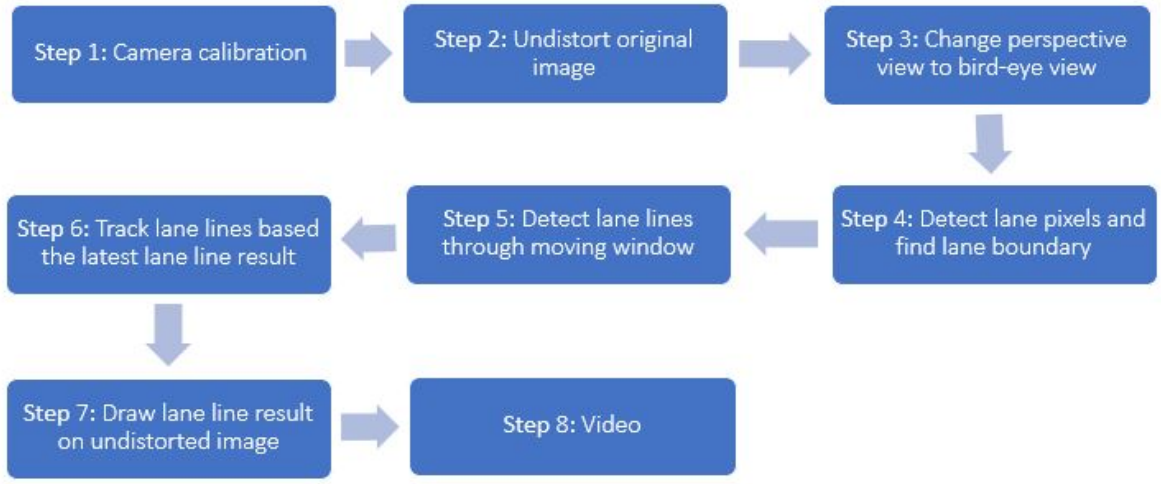

Figure 3.10: Flowchart for advanced method

4 Algorithm Performance

4.1 Performance comparison of conventional and advanced algorithms

We apply these two algorithms in two different lane lines, and get four images, including straight lane lines using conventional and advanced algorithms and curved lane lines using conventional and advanced algorithms.We can observe when apply conventional method on curved highway, due to the limitation of Hough transform, it fails to detect the lane lines.

Figure 4.1: Comparison with different conditions

4.2 Limitations and Shortcomings

Even though in the advanced method, the system detects the lane lines using advanced method in high accuracy, it still contains many limitations:

- Manually select the source and destination points, therefore, could not achieve fully automation
- Can not deal with steep turn due to the limitation of sliding window method
- Can not apply in the extreme light condition and condition called "no-visual-clue"

5 Conclusions

Edge detection is a vast subject and, especially, many techniques need to be executed to eventually detect the lane lines in the challenging scenarios. In this paper, we implemented the first conventional approach to extract road features with the Canny edge detection and group similar straight lines to generate left and right lanes with the Hough transform. We noticed some shortcomings of the first conventional approach. To overcome its limitations, we implemented the second advanced lane detection approaches by applying a more robust lane detection algorithms that can generate stable detection results in difficult scenarios.

References

[1] Abdulbasit Alazzawi. Edge detection-application of (first and second) order derivative in image processing: Communication. *Diyala Journal of Engineering Sciences*, 8(4):430–440, 2015.

[2] Keerti Chand Bhupathi and Hasan Ferdowsi. An augmented sliding window technique to improve detection of curved lanes in autonomous vehicles. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pages 522–527. IEEE, 2020.

[3] Alan C Bovik. *The essential guide to image processing*. Academic Press, 2009.

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[5] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[6] Juan Pablo Gonzalez and Umit Ozguner. Lane detection using histogram-based segmentation and decision trees. In *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493)*, pages 346–351. IEEE, 2000.

[7] D Hema and Dr S Kannan. Interactive color image segmentation using hsv color space. *Science and Technology Journal*, 2020.

[8] M Kazemi and Y Baleghi. L* a* b* color model based road lane detection in autonomous vehicles. *Bangladesh Journal of Scientific and Industrial Research*, 52(4):273–280, 2017.

[9] Chan Yee Low, Hairi Zamzuri, and Saiful Amri Mazlan. Simple robust road lane detection algorithm. In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–4. Ieee, 2014.

[10] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[11] Raja Muthalagu, Anudeepsekhar Bolimera, and V Kalaichelvi. Lane detection technique based on perspective transformation and histogram analysis for self-driving cars. *Computers & Electrical Engineering*, 85:106653, 2020.

[12] Udacity GitHub repository. Carnd-advanced-lane-lines. `https://github.com/udacity/CarND-Advanced-Lane-Lines`, 2021.

[13] GT Shrivakshan and Chandramouli Chandrasekar. A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues (IJCSI)*, 9(5):269, 2012.

[14] Bryant Walker Smith. Human error as a cause of vehicle crashes. `http://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes`, 2013.

[15] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image Vis. Comput.*, 22:269–280, 2004.