VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

LOGIC DESIGN PROJECT (CO3091)

Topic:
Edge Detection Application in Lane Detection

Advisor: Tran Ngoc Thinh
Student: Bui Trung Duc - 1852324.

HO CHI MINH CITY, Dec 2022

Contents

1 Introduction

Edge detection is an image processing technique, used to determine the boundaries (edges/corners) of objects or sets of pixels in an image. Edge/corner is one of the most important features associated with images. We can get a good idea of the basic structure of an image through its edges/corners. Therefore, edge/corner detection is an important and popular technique in image processing applications in particular and computer vision in general.

The main goal of the project is to apply edge/angle detection techniques to a problem in computer vision. It is through using the camera to detect road markings (Lane detection). This is a very important application in a driver assistance system (ADAS).

Figure 1: *Advanced Driver Assistance System*

2 Some basic concepts in image processing

2.1 Pixel - Picture Element

The real image is a continuous image in terms of space and luminance value. To be able to process images by computer, it is necessary to digitize the image. In this process, the concept of Pixels is used to represent the elements of the image. Here, it is also necessary to distinguish the concept of pixels often referred to in computer graphics systems. To avoid confusion, we temporarily call this pixel concept a device pixel.

The concept of device pixels can be considered as follows: when we observe the screen (in graphics mode), the screen is not continuous but consists of many small points, called pixels. Each pixel consists of a pair of x, y, and color coordinates. The x, and y coordinate pairs make up the resolution. As computer monitors have many different resolutions, currently popular are VGA monitors with a resolution of 640x480 or XSGA with a resolution of 1024x768.

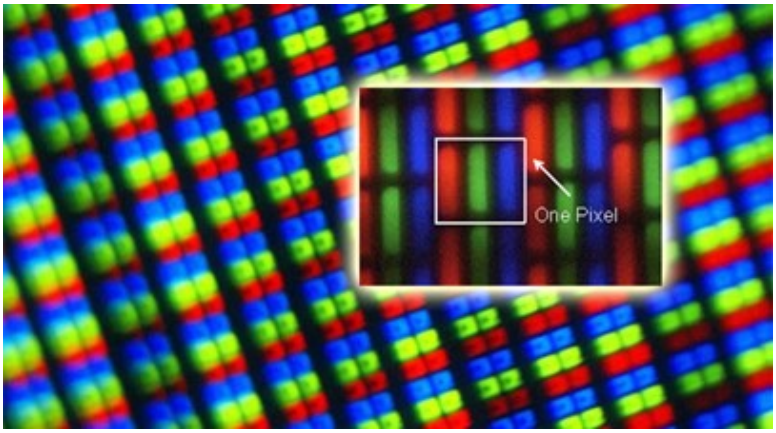


Figure 2: *One pixel*

2.2 Color Image

Color images contain color information for each image element. Usually, this color value is based on color spaces, in which the commonly used color space is RGB corresponding to the three channels of red - green- blue. Depending on the number of bits, used to store color we have different colors, for example, 8-bit, 16-bit, and 24-bit (True Color). If we use less than 24 bits to store colors, we must have a Color Palette, which is similar to a Lookup Table that allows mapping between a position in the table to a combination of the RGB color space. For example, using 8 bits corresponding to 256 colors, we must have a mapping table of 256 colors corresponding to 256 combinations of Red - Green - Blue.

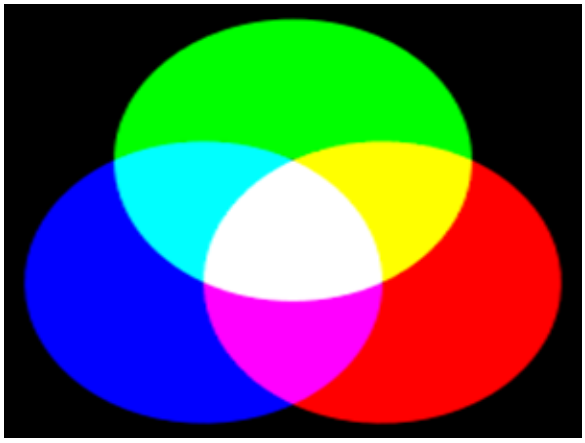


Figure 3: *RGB Color*

2.3 Gray Level

The gray level is the result of encoding the respective luminous intensity of each pixel with a numerical value. A grayscale image is an image that has a gradual grayscale transition from white to black. In fact, the gray level value is a combination of three RGB values (Red-Green-Blue). Normally, each pixel in a polygraphy image is usually encoded with 8 bits, corresponding to 256 gray levels.

2.4 Binary Image

A binary image with only 2 levels: black and white. In other words, each pixel of a binary image can only be 0 or 1.

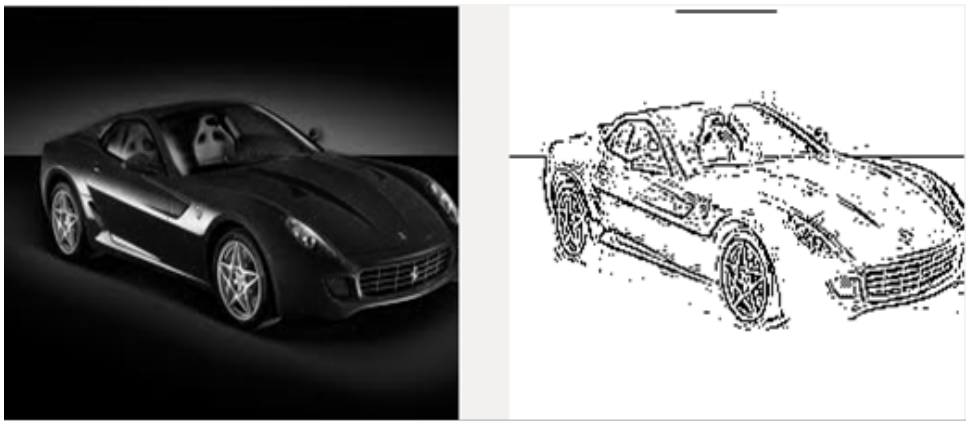


Figure 4: *Binary Image*

2.5 Canny Edge Detection

Canny Edge Detection is an algorithm used to detect a series of edges in an image. It was developed by John F. Canny in 1986. The algorithm is divided into 5 basic steps:

- Noise reduction.
- Gradient calculation.
- Non-maximum suppression.
- Double threshold.
- Edge Tracking by Hysteresis.

2.5.1 Noise reduction

Because edge detection is very sensitive to noise, one way to remove this is to apply Gaussian blur to smooth the image. There are many convolution techniques used with Gaussian Kernel (3x3, 5x5, 7x7,...). The size of the kernel depends on the blur effect, the larger the kernel, the more blurred it will be. Here is the image after applying Gaussian blur.

Original image (left) — Blurred image with a Gaussian filter (sigma=1.4 and kernel size of 5x5)

2.5.2 Gradient calculation

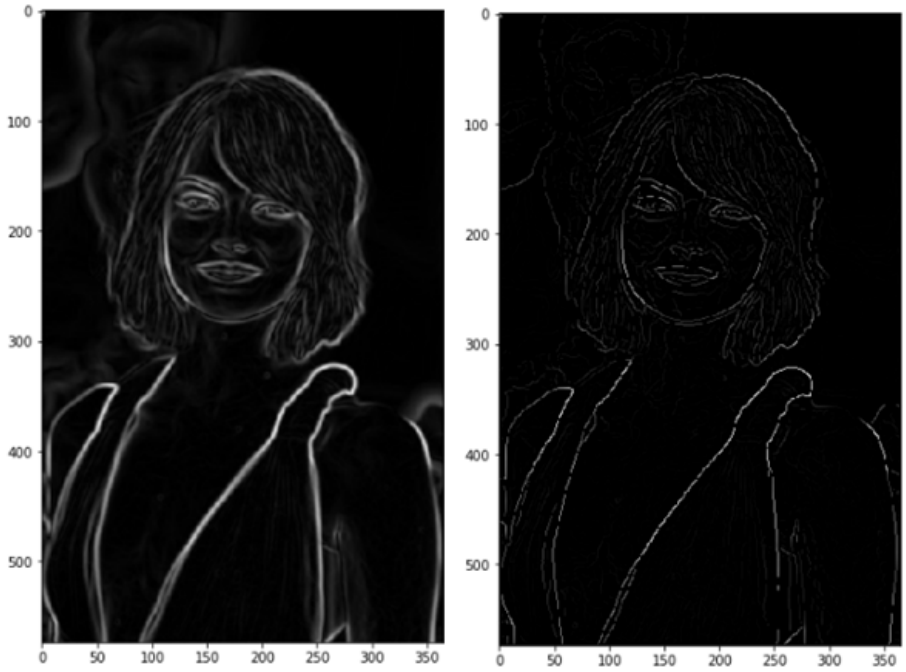
This step is to detect edge strength by directly calculating the derivative of the image. The edges correspond to different varying intensities of the pixels. The easiest way to detect it is to apply a filter that highlights the degree of change in the x and y directions on the coordinate axes.

Blurred image (left) — Gradient intensity (right)

2.5.3 Non-maximum suppression

This step is used to thin the edges with the following simple principle: the algorithm will go through all the points of the derivative intensity matrix and find the pixels with the maximum value in the edge directions.
Implementation steps:

- Initialize a 0 matrix with the same dimension as the derivative intensity matrix.
- Determine the direction of the edges.
- Check if pixels in the same direction are more intense than the currently generated pixels being processed.
- Returns the image processed with the Non-Maxium SuppressMax algorithm.

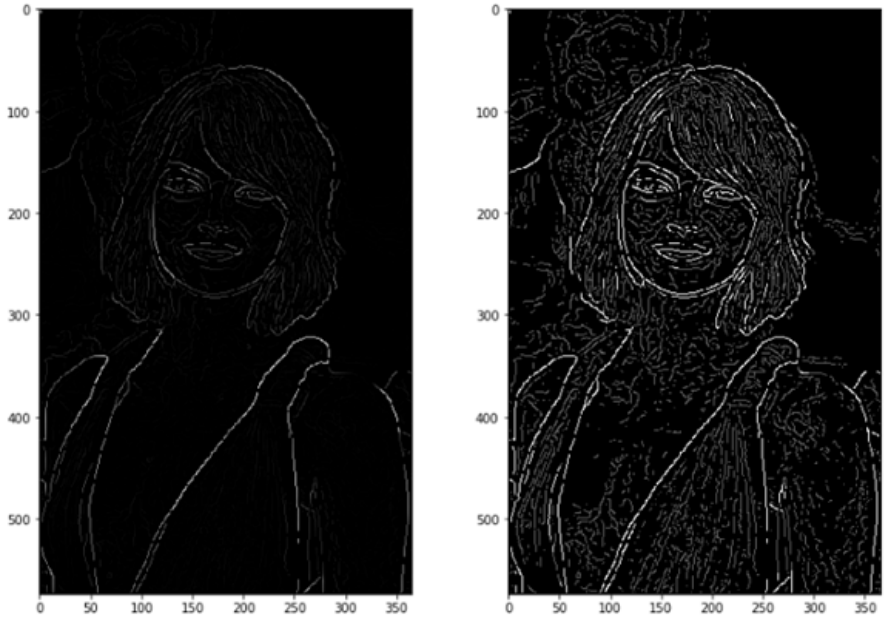


Result of the non-max suppression.

2.5.4 Double threshold

The goal is to identify three types of pixels: strong, weak, and irrelevant. A strong pixel is a high-intensity pixel that definitely contributes to an advantage. Weakness means the pixels do not have enough intensity value to form an edge. The non-relevant means that the pixel is not related to the edge.
Algorithm:

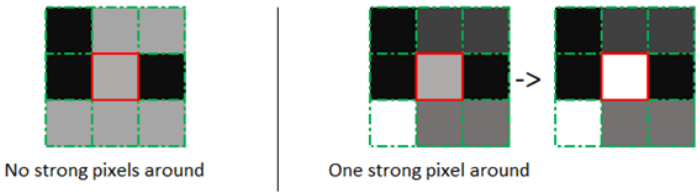
- The high threshold is used to identify strong pixels.
- The low threshold is used to detect irrelevant pixels.
- All pixels with values in the middle of the threshold are weak pixels.



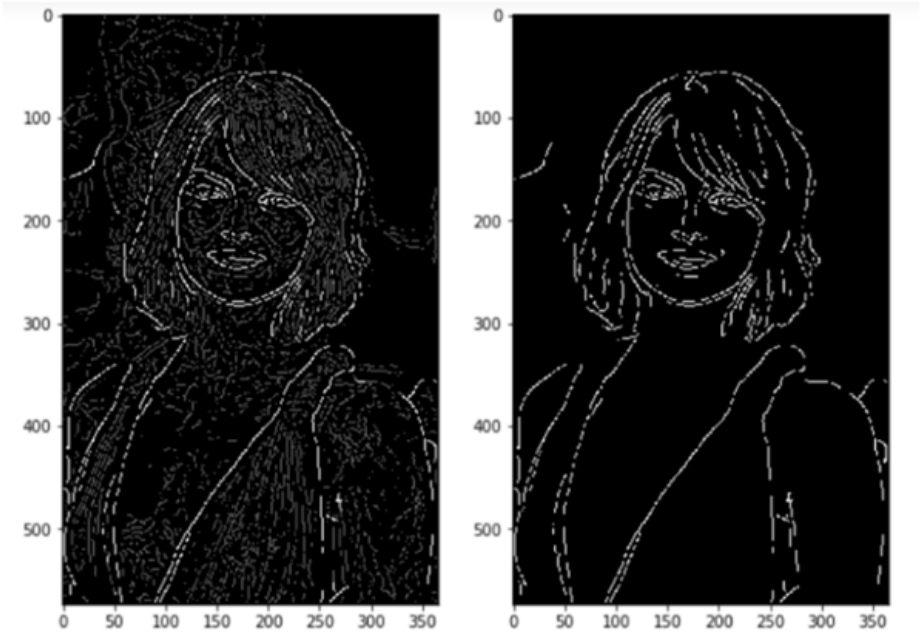
Non-Max Suppression image (left) — Threshold result (right): weak pixels in gray and strong ones in white.

2.5.5 Edge tracking by hysteresis

Based on the result of the threshold, hysteresis will transform weak pixels into strong; if and only if there is a weak pixel surrounded by strong pixels.



The final result:



Results of hysteresis process

2.6 Grayscale

If each color of a pixel is described by three components red, green, and blue (R, G, B), then there are three common ways to convert an image from a color image to a grayscale image.

Lightness: we will get the threshold by taking the sum of the maximum and minimum values of all pixels and then dividing by : (max(R,G,B) + min(R,G,B))/2

Average: average of 3 points: (R+G+B)/3

Luminosity: 0.21R+0.72G + 0.07B

2.7 Hough Transform

Hough Transform is an effective line detection algorithm in image processing. We will use the Hough Transform algorithm to detect the line in the image using the OpenCV library.

2.7.1 Theory

The general idea of line detection in this algorithm is to create a mapping from the image space (A) to a new space (B) where each line in space (A) corresponds to a point in space (B).

The basic line equation will be represented by 2 parameters a and b as follows: y=ax+b. However, with this representation, the value of the tilt angle ranges from -∞ to +∞. For example, to get the equation of the line Oy (x=0), a must approach ∞. The Hough Transform algorithm requires that the values of a and b lie within a specified range (or are bounded above and below), we must use a polar coordinate to represent the line equation. The equation: ρ=xcos(θ)+ysin(θ).

Considering that in the polar coordinate equation, the value of angle ∞ can be intercepted in the interval [0, π). In fact, the image space is a finite space (bounded by the edges of the image), so the value ρ is also bounded.

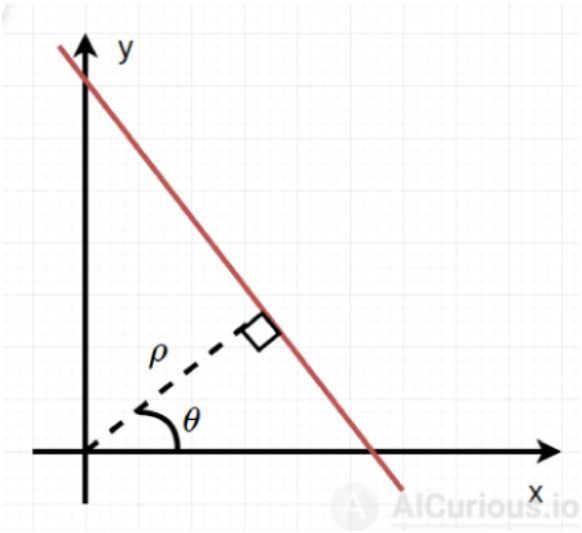


Figure 5: *Line in coordinate system*

2.7.2 Mapping between image space (A) and Hough space (B)

From a line in the image space (A) with two parameters ρ and θ, we will map the Hough space (B) to a point.

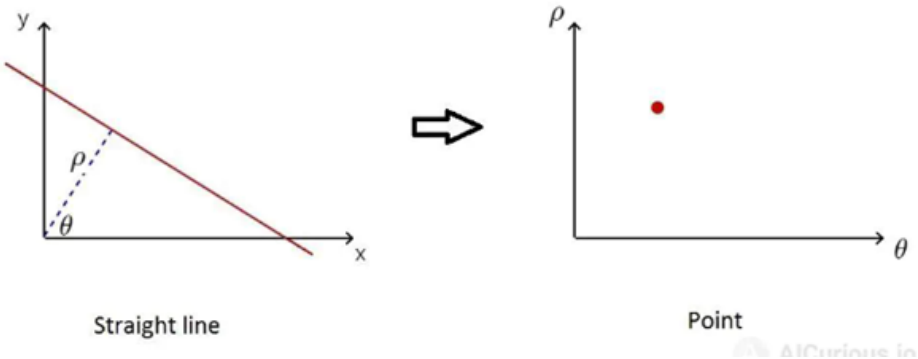


Figure 6: *Mapping one point from image space to Hough space*

Points lying on the same line are represented as sinusoids that intersect at a point in Hough space. This is where the idea of the Hough Transform algorithm originates. We will rely on these intersections to inversely deduce the equation of the line in the image space.

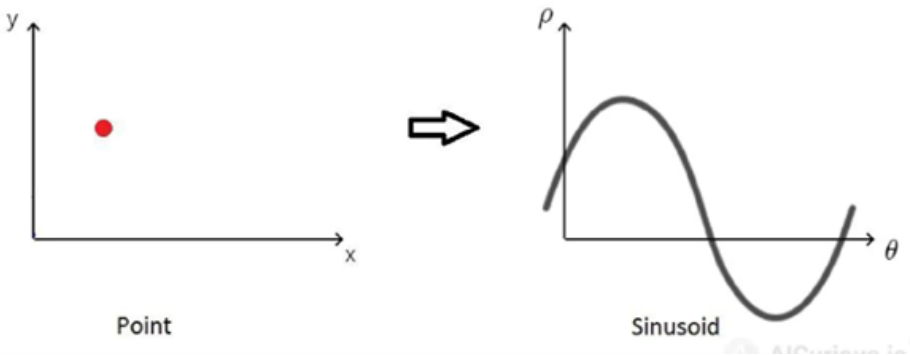


Figure 7: *Sinusoid*

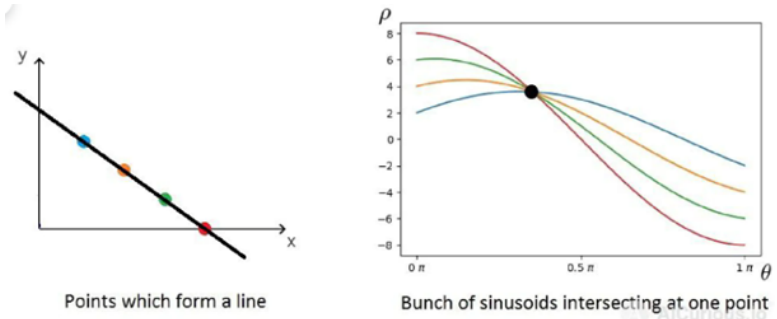


Figure 8: *Mapping multiple collinear points from image space to Hough space*

Each different line will form a bright spot (where many sinusoids intersect) on Hough space. Below is the representation of two lines in Hough space.

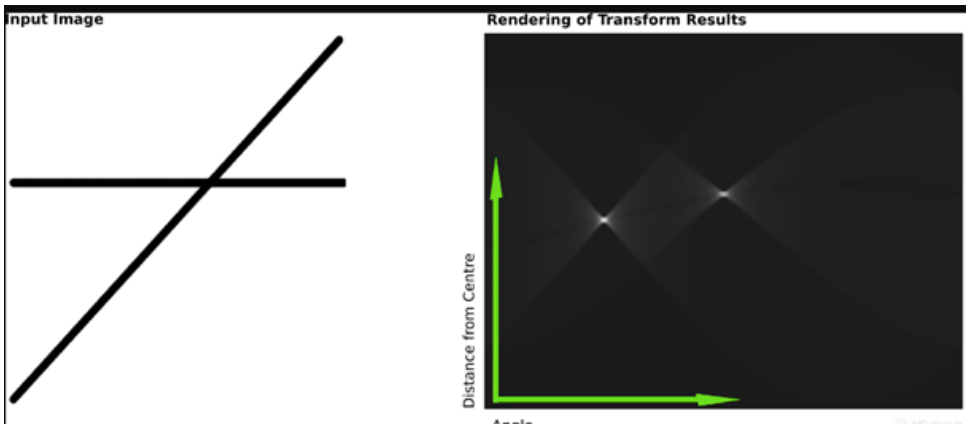


Figure 9: *Representing 2 straight lines in Hough space*

3 Implementation

3.1 Color mask

Figure 10: *Original image*

Looking at the photo, we can see that one of the most obvious characteristics of the road markings is that they are white or yellow compared to the darker line color.

Defines a color mask that allows color selection based on pixels in the image. The desire is to select only the white and yellow pixels and set the rest of the image to black.

Image processing techniques are applicable to RGB Color Space but for color selection, HSV

space is much better.

A yellow mask was created from the HSV image to select pixels with a color between 15 and 25 and oversaturated above 60. These values were determined from the images corresponding to the yellow line markers. The white mask is created from an RGB image where all three color channels of the pixel must be higher than the threshold value of 200. This correctly selects the white spots on the road surface.

Two masks are applied using bitwise or to retain both white and yellow pixels. All other pixels are set to black. With this selection, all lane pixels are selected correctly. There was also some noise, mostly from the hay and the engine of a white car on the highway. This will be removed with further processing.

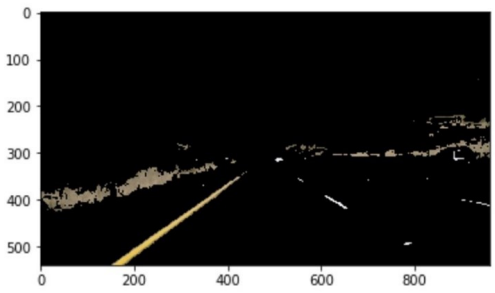


Figure 11: *Image after threshold*

3.2 Image smoothing

When the image is processed and keeps only white and yellow pixels, the color is no longer important so the image is turned into grayscale for easier processing.

To prepare for edge detection, it is useful to smooth the image so that artificial edges are not detected due to noise. For this, Gaussian Blurr is applied with kernel 5 as well as other defaults.

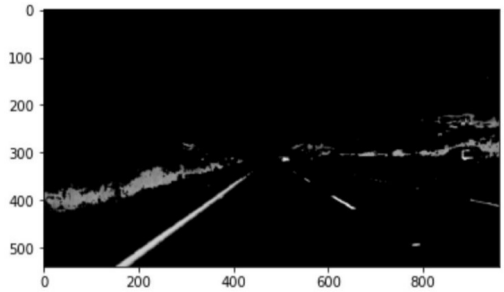


Figure 12: *Image smoothing*

3.3 Edge detection

Edges were detected using the Canny Edge filter application for grayscale images. The Canny edge filter is basically calculating the gradient on the image with respect to the X and Y directions, the resulting matrix represents the difference in intensity between adjacent pixels.

The algorithm will first detect strong edge (strong gradient) pixels above the high threshold, 150 for our image, and reject pixels below the low threshold, here chosen as 50. They are connected to the strong edges. The output is a binary image with white pixels showing the detected edges and black for the rest.

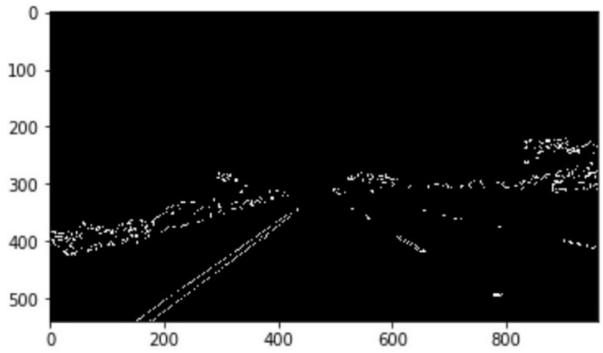


Figure 13: *Edge detection*

3.4 Select zone of interest.

In the resulting edge processing image, the edges are correctly defined, but there are lots of other unnecessary edges as well. They are mostly from objects that lie outside the road or maybe edges that define the boundaries of other cars. To get rid of this, I took advantage of the continuous area of interest in the image.

A polygon is defined based on the assumption that the camera is mounted in a fixed position on the vehicle. A selected trapezoid starts at the bottom of the image and goes toward the center. Values are determined visually from the images.

The Zone_OF_INTEREST function creates a mask using CV2.fillpoly(Mask, vertex, 255). The original mask is a Zeroes matrix of the same dimensions as the grayscale image. Fillpoly creates a polygon based on the given vertices and puts all pixels within 255. The mask is applied using bitwise_and to the grayscale image.

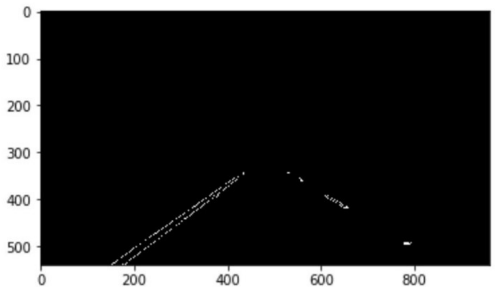


Figure 14: *Select and just display zone of interest*

3.5 Selection of zone of interest from pixel edges

As described in the logic. The Hough transform takes pixels from the (x, y) coordinates of the image space and turns them into hough space. A straight line in the image space y = mx + b can be represented as a point (b, m) in hough space. Similarly, in Hough space, each image space point is transformed into a line, which represents all possible image space paths that pass through the selected point.

When multiple lines intersect in a point (or larger defined area) in hough space, this indicates that the corresponding points in the image space are Colinear.

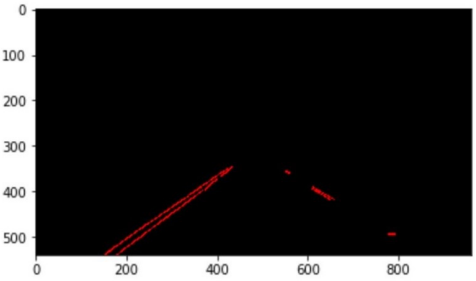
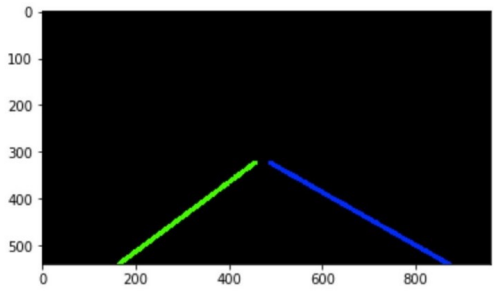


Figure 15: *Draw line detected*

3.6 Draw a straight line.

To draw a single line on the left and one on the lines on the right, the draw_lines function is built to extrapolate from the multiple individual lines found using the Hough transform.

Lines from the Hough transform are grouped by category left and right based on the calculated gradient. Once grouped, the mean slope is calculated along with the standard deviation. To remove lines that do not match the rest, only lines with a consistent slope are kept.



Using the two intersection points, an extrapolated line is drawn on the original image. The line is semi-transparent so it can be visually inspected to verify if the line aligns with the lane markings. For better visualization, the left line is green while the right one is blue.

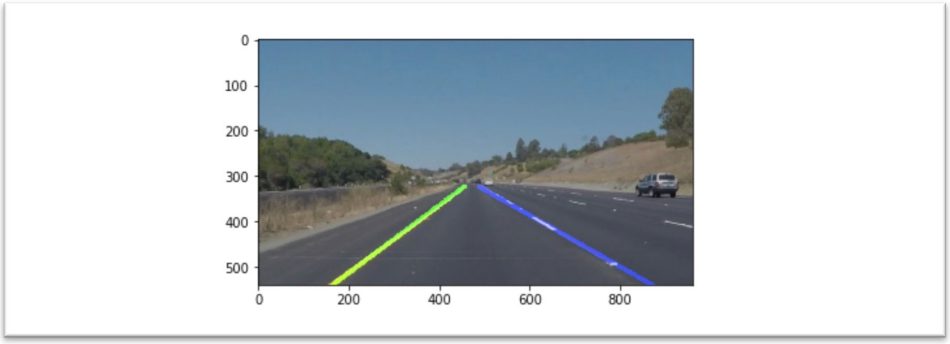


Figure 16: *Final Result*

4 Source code and Github Link

https://github.com/DucOriginal/Logic-Design-Project.git

5 Conclusion

Because it uses image processing, it is only true for images with a specific gray level. Influenced a lot by the environment, so for each environment, we determine each color threshold so that it is appropriate to distinguish the lane and surrounding colors.

6 Difficulty and Development

Because of the limited time of the project, I can only do it on my local computer, not on hardware devices. In the future, I will develop the system into a more complete product.