Artificial Intelligence (CS-308)

Project Report

Course Instructor:	**Dr Javed Iqbal**
Project Name	Traffic Road Lane line Detection
Issue Date:	04 April 2020
Submission Date:	30 June 2020

Submitted by:	Registration#	Name
	19-CS-44 19-CS-40	Farhan Ali Muhammad Moin Bukhari
Obtained Marks:		

University Of Engineering and Technology, Department of computer science, Taxila

ABSTRACT

Many technical improvements have recently been made in the field of road safety, as accidents have been increasing at an alarming rate, and one of the major causes of such accidents is a driver's lack of attention. To lower the incidence of accidents and keep safe, technological breakthroughs should be made. One method is to use Lane Detection Systems, which function by recognizing lane borders on the road and alerting the driver if he switches to an incorrect lane marking. A lane detection system is an important part of many technologically advanced transportation systems. Although it is a difficult goal to fulfil because to the varying road conditions that a person encounters, particularly while driving at night or in daytime. A camera positioned on the front of the car catches the view of the road and detects lane boundaries. The method utilized in this research divides the video image into a series of sub-images and generates image-features for each of them, which are then used to recognize the lanes on the road. Several methods for detecting lane markings on the road have been presented. Keywords: Lane Detection, Artificial Intelligence, Computer Vision, Traffic Safety

INTRODUCTION

Traffic safety is becoming increasingly crucial as urban traffic grows. People exiting lanes without respecting the laws cause most accidents on the avenues. The majority of these are the outcome of the driver's interrupted and sluggish behavior. Lane discipline is essential for both drivers and pedestrians on the road. Computer vision is a form of technology that enables automobiles to comprehend their environment. It's an artificial intelligence branch that helps software to understand picture and video input. The system's goal is to find the lane markings. Its goal is to provide a safer environment and better traffic conditions. The functionality of the proposed system can range from displaying road line positions to the bot on any outside display to more advanced applications like recognizing lane switching soon to reduce concussions caused on roadways. In lane recognition and departure warning systems, accurate detection of lane roads is crucial. When a vehicle breaches a lane boundary, vehicles equipped with the predicting lane borders system direct the vehicles to avoid crashes and issue an alarm. These intelligent systems always provide safe travel, but it is not always necessary that lane boundaries are clearly visible, as poor road conditions, insufficient quantity of paint used for marking the lane boundaries, and other factors can make it difficult for the system to detect the lanes accurately. Other factors can include environmental effects such as shadows cast by objects such as trees or other automobiles, or streetlights, day and nighttime conditions, or fog caused by invariant lightening conditions. These factors cause problem to distinguish a road lane in the backdrop of a captured image for a person. To address the issues raised above because of lane boundary adjustments. The algorithm used in this work aims to recognize lane markings on the road by feeding the system a video of the road as an input using computer vision technology, with the primary goal of lowering the number of accidents. Accidents caused by irresponsible driving on the roads can be avoided with the installation of a system in automobiles and taxis. It will ensure the safety of the children on school buses. Furthermore, the driver's performance may be tracked, and Road Transportation Offices can use the system to monitor and report driver irresponsibility and lack of attention on the roadways.

ASSUMPTIONS AND DEPENDENCES

Assumptions

Assumptions The following assumptions were made in preparing the Project Plan:

- All road Constructed should meet the highway authority standards
- The Road should be in good Quality
- We assume that the weather will be sunny with adequate lighting on the road
- Assuming the system used for lane detection is sufficiently fast and memory is not limited.
- The Lane Paint in the road should be visible
- Yellow Lane Color: define the left side of road beginning
- White Lane Color: define the right side of road ending

Dependencies

Resource-based planning dependencies:

1. For training of data, high GPU computational power is required, so time for training is directly connected to system raw power.
2. Output from the camera mounted on the moving vehicle will have great impact on detection result.
3. Lane visibility in weather condition will also have high impact of model accuracy for lane detection.

Logical planning dependencies:

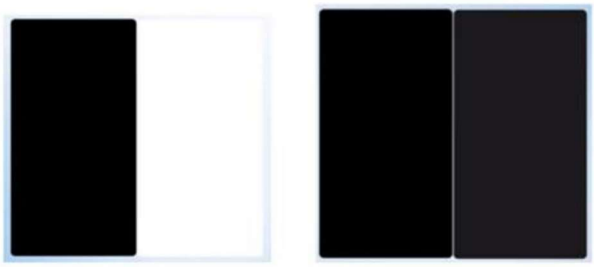
- Python Libraries should be pre-installed for working of this module.
- All operating System files should be updated.
- System should have sufficient Memory for module operation

METHODOLOGY

Using Python and Jupyter Notebook, the project involves detecting lane lines in an image.

The Canny Edge Detection Technique

The purpose of edge detection is to locate object boundaries within photographs. A detection is used to look for areas in an image where the intensity changes dramatically. An image can be recognized as a matrix or an array of pixels. A pixel represents the intensity of light at a specific spot in the image. Each pixel's intensity is represented by a numeric value ranging from 0 to 255; a value of zero indicates that something is completely black, while 255 indicates that something is completely white. A gradient is a pattern of varied brightness pixels. A significant gradient indicates a steep change, whereas a slight gradient indicates a shallow change

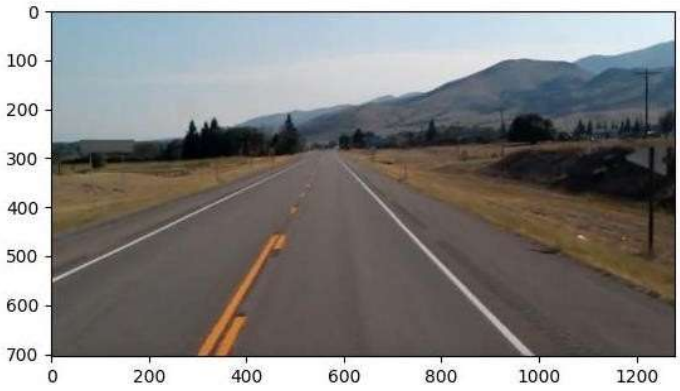


And there is a bright pixel in the gradient image anywhere there is a sudden shift in intensity (rapid change in brightness), i.e., wherever there is a strong gradient. We get the edges by tracing out all these pixels. This notion will be used to detect the edges in our road image.

Original Image

We will load and read our image into an array:

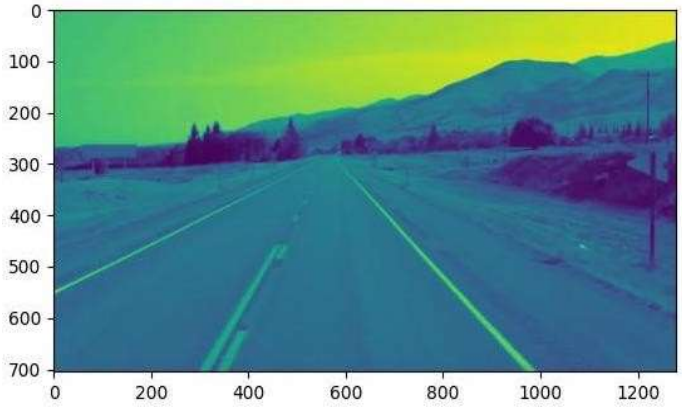
```
image = cv2.imread('road.jpg')
```



We then convert the image to grayscale:

```
gray_image=cv2.cvtColor(image,cv2.COLOR_RGBG
GRAY)
```

Now the image is converted to grayscale:

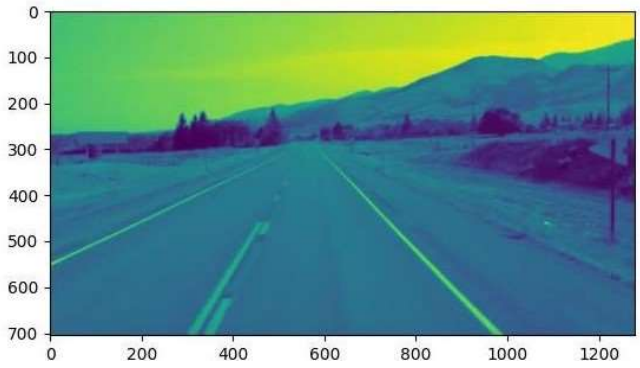


Gaussian Blur

A grayscale image's pixels are each defined by a single number that indicates the pixel's brightness. The common solution for smoothing an image is to change the value of a pixel with the average value of the pixel intensities around it. A kernel will average out the pixels to reduce noise. This kernel of normally distributed numbers (np.array ([[1,2,3],[4,5,6], [7,8,9]]) is applied to our entire image, smoothing it out by setting each pixel value to the weighted average of its neighbors. In our case we will apply a 5x5 Gaussian kernel:

```
blur=cv2.GaussianBlur(gray_image,(
5,5),0)
```

Below is the image with reduced noise:



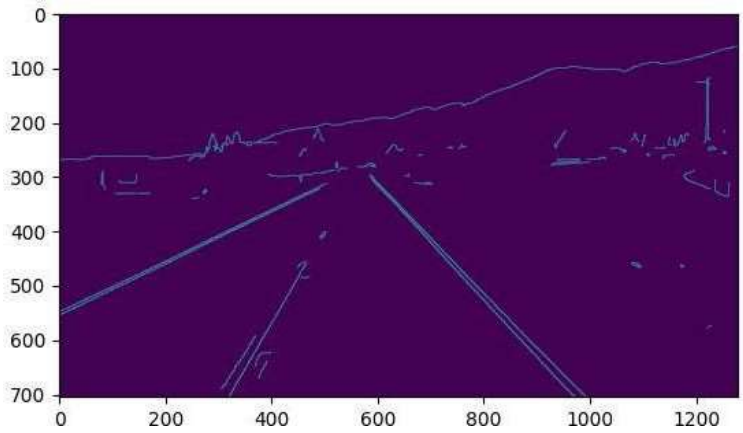
Edge Detection

An edge is a region in a picture where the intensity/color between neighboring pixels in the image changes dramatically. A steep change is a significant gradient, while a shallow change is the opposite. In this sense, an image may be thought of as a matrix with rows and columns of intensities. This means that a picture can also be represented in 2D coordinate space, with the x axis traversing the width (columns) and the y

axis traversing the image height (rows). The Canny function measures the change in brightness between neighboring pixels by performing a derivative on the x and y axes. In other words, we're calculating the gradient (or brightness change) in all directions. It then traces the strongest gradients with a series of white pixels.

```
canny_image = cv2.Canny(blur, 100, 120)
```

Below is the image after applying the Canny function:



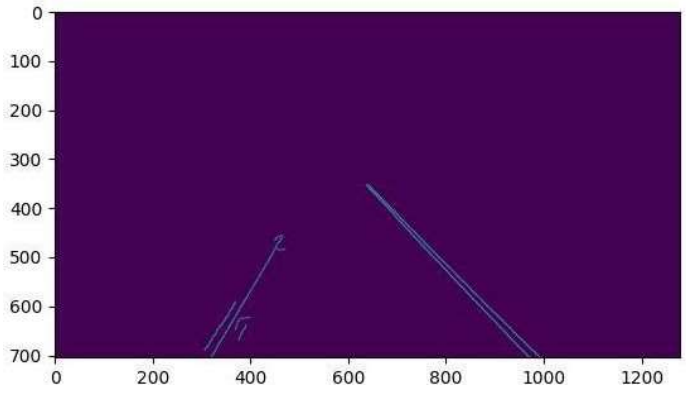
We can separate nearby pixels that follow the strongest gradient using the low_threshold and high_theshold functions. It is allowed as an edge pixel if the gradient is more than the upper threshold; if it is less than the lower threshold, it is rejected. If the gradient falls between the criteria, it is only approved if it is linked to a strong edge. The white line depicts a location in the image where there is a high change in intensity above the threshold, whilst the fully black areas relate to low variations in intensity between adjacent pixels.

Region of Interest

The image's dimensions are chosen to include the road lanes and identify the triangle as our region of interest. Then a mask with the same dimension as the image is constructed, which is effectively an array of all zeros. Now we'll fill the triangular dimension in this mask with 255 to make our region of interest dimensions white. Now we'll combine the canny image with the mask in a bitwise AND operation to get our final region of interest.

```
def region_of_interest(img, vertices):
    mask =
    np.zeros_like(img)
    match_mask_color =
    255
    cv2.fillPoly(mask, vertices,
    match_mask_color) masked_image =
    cv2.bitwise_and(img, mask)
    return masked_image
cropped_image = region_of_interest(canny_image, np.array ([region_of_interest_vertices], np.int32))
```

Below is the image after doing a bitwise operation with the canny image and the mask, also called the masked_image:



Hough Transform

Now we apply the Hough transform approach to determine the lane lines by detecting straight lines in the image. A straight line is described by the following equation:

y= mx + b.

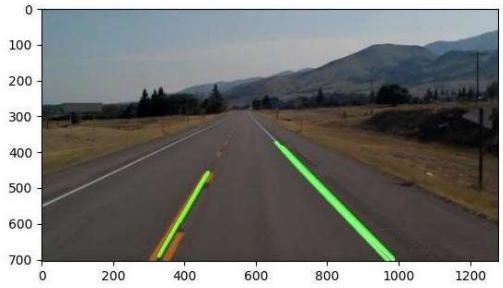
The line's slope is simply a climb over a run. If the y intercept and slope are known, the line can be plotted as a single dot in Hough Space. There are numerous lines that can pass through this dot, each with different 'm' and 'b' values. Each point can be crossed by several different lines, each with a different slope and y intercept value. There is, however, one line that connects both sites. We can figure this out by looking at the point of intersection in adequate space, because that point of intersection represents the 'm' and 'b' values of a line that crosses both locations in Hough Space. To identify the lines, we must first divide our Hough space into a grid. Each bin in the grid corresponds to the line's slope and y intercept value. Every point of intersection in a Hough Space bin will receive a vote from the bin to which it belongs. Our line will be drawn from the bin with the most votes. The slope of a vertical line, on the other hand, is infinite. So, to express vertical lines, we will use polar coordinates instead of cartesian coordinates. So, the equation of our line becomes:

Below is the combined image:

```
def draw_the_lines(img, Lines):
    img = np.copy(img)
    blank_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

    for line in Lines:
        for x1, y1, x2, y2 in line:
            cv2.line(blank_image, (x1, y1), (x2, y2), (0, 255, 0), thickness=4)
    img = cv2.addWeighted(img, 0.8,
blank_image,1, 0.0) return img
lines = cv2.HoughLinesP(cropped_image, rho=2, theta=np.pi/60, threshold=50, lines=np.array([]),
minLineLength=40, maxLineGap=80)
```

```
image_with_lines = draw_the_lines(image, lines)
```



Now the same methodology can be applied on the video which contains a few frames or images using video capture function and reading each frame in the video with help of the while loop like:

```
cap = cv2.VideoCapture('test_video.mp4')

while cap.isOpened():
  ret, frame =
cap.read() frame =
process(frame)
  cv2.imshow('frame', frame)
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

CONCLUSIONS

To achieve edge detection, we used the Jupyter notebook and algorithms like the Canny Function. Then we created a zero-intensity mask and mapped our region of interest using the bitwise approach. The Hough Transform method was then used to detect the straight lines in the image and identify the lane lines. We utilized polar coordinates since Cartesian coordinates could not give an adequate slope for vertical and horizontal lines. Finally, we merged the original image with our zero-intensity image to show lane lines.

FUTURE WORKS

Because we use modular implementation, updating algorithms is simple, and work on the model can be continued in the future. We insert the model's pickle file into the relevant locations, which can then be simply transferred to goods. As a result, compiling the full big code might be avoided easily. We can also enhance the idea by creating a new future in which the road can be identified in the dark, or at night. In daylight, the color recognition and selection process are highly effective. Adding shadows will make things a little noisier, but it won't be as rigorous as driving at night or in low light (e.g., heavy fog). And this research can only recognize lanes on bitumen roads, not on the loamy soil roads that are ubiquitous in Indian villages. As a result, this project can be improved to detect and avoid accidents on loamy soil roads found in communities.

REFERENCES

[1] Shopa, P., N. Sumetha and P.S.K Pathra. "Traffic sign detection and recognition using OpenCV",
International Conference on Information Communication and Embedded Systems (ICICES2014), 2014

[2] "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixelwise labelling," arXiv
preprint arXiv:1505.07293, 2015.

[3] J. Long, E. Shelhamer, and T. Darrell, "LANE DETECTION TECHNIQUES" – A Review." in Proceedings of the
IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[4] M.Cordts,M. Omran,S.Ramos, T.Rehfeld,M. Enzweiler, R.Benenson,U. Franke,S.Roth, and B.Schiele, "The cityscapes dataset for semantic urban scene understanding," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, And P.H Torr, "A Layered
Approach To Robust Lane Detection At Night." 2015, pp. 1529–1537.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv:1512.03385, 2015.