Journal of Positive School Psychology
2022, Vol. 6, No. 3, 6973–6979

Road Detection and Segmentation using OpenCV

Ankit Singh [1], Pranab Hazra [2], Moupali Roy [3], Abhijit Ghosh [4]

[1,2,3,4] *Electronics and Communication Engineering Department,*
Narula Institute of Technology, Kolkata, India
Email: [1]ankitrobin7@gmail.com, [2]pranabhazra2017@nit.ac.in, [3]moupali.roy@nit.ac.in,
[4]abhijit.ghosh@nit.ac.in

Abstract

In today's world, road detection and segmentation plays an important role in navigation system and generally used to detect new roads and patterns in the region/area. Its main aim is to provide navigation to the autonomous vehicle and robot on the ground. With the help of road detection and segmentation, it will be useful for us in finding correct road path where the vehicle can move as supportive vehicles thus preventing any collision with an obstacles on road. In this paper, a new technique for road detection and segmentation is proposed by using OpenCV(Open Source Computer Vision Library) .OpenCV is basically a library functions which aimed to provide a real-time computer vision. It consist of various methods through which an autonomous vehicle can detect obstacles on road and navigate it accordingly. It have higher accuracy with an average of 90-95%.

Keywords— Navigation, Road Detection, Segmentation, CNN, OpenCV, Grayscale, Canny edge.

I. INTRODUCTION

According to the World Health Organization, every year around 1.3 million of people die due to road accident. Between 30 and 60 million people suffer from fatal and non-fatal injuries [3]. It also causes damage to millions of public property resulting as economic losses. Therefore, in order to prevent it many automobile firms took their stand. Automobiles companies like Volvo Corporation, Tesla,Inc. , Nissan and many others have come forward and declared that they will participate for the program which was meant for development of the automatic vehicle. Volvo Corporation have declared that by 2020, using driving assistance system and warning, it will decrease the number of chances of having any serious accidents and injuries by one of its new cars by [1].

With the increase of population, it had been seen that large proportion of youth moving towards driving cars and at the same time it increase the chances of accidents or loss of lives. Intelligent vehicle technologies have strongly developed for this purpose. These technologies works on sensor such as Lidar, vision sensors, Radar[7]. Basically Lidar and Radar works in catching any obstacles which comes on its way whereas for lane detection and vehicle detection vision sensors is used. By using theses, it enables the vehicles to avoid collisions and

support a warning system. Nowadays these sensors are cheaper, smaller and gives high accuracy .It have strong and good graphical processing units (GPU). Many big automobiles companies started working on autonomous vehicles which runs without any drivers on roads. For these factors, lane detection using sensors becomes an interesting topic for researchers.

II. WORKING PROCEDURE

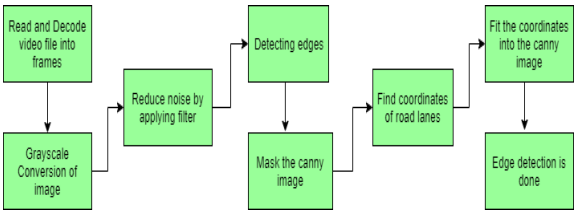


Figure 1: Lane detection involves the following steps:

III. METHODOLOGY

The methodology includes detecting of lane lines in an image using Python and OpenCV. OpenCV means "Open-Source Computer Vision", which contains useful package for analyzing images [14].

For loading and reading our test image into an array we will use: **image = cv2.imread('test_image.jpg')**

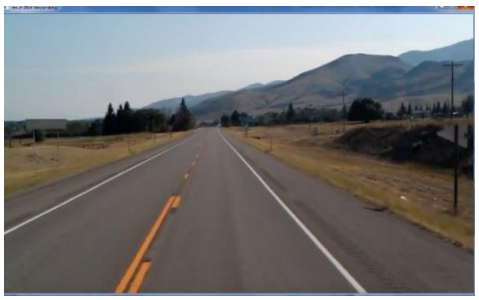


Figure 2: Test image

For converting our test image into a grayscale image we will make first copy of our original image using Numpy:

lane_image= np.copy (image)

3.1 GRAYSCALE CONVERSION

It is necessary to convert to grayscale image because as colour image is made of pixels contains 3 channels –Red ,Green and Blue and each and every pixel in the image is a combination of these three intensities while in grayscale image contain only one channel with each pixel belonging to one intensity value that ranges from 0-255 [6]. The reason for using grayscale image is that for processing a single channel is faster than processing a three channel colour image with less computationally intensive.

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

Figure 3: Image is converted into grayscale:

3.2 GAUSSIAN BLUR

After the conversion into grayscale image now it's necessary to reduce noise, while for detection of edges it is necessary to catch accurately all the possible edges of images ,we must filter out noises as it can create false edges ultimately effect edge detection and thus it is necessary to filter it out and smoothen the image. Filtering out the image noise and smoothening will be done by Gaussian filter. By using Kernel convolution, we can write this line of code inside of our editor [17].

blur =cv2.GaussianBlur (gray,(5,5),0)

Here we are applying Gaussian Blur on grayscale image with 5x5 kernel and the size of the kernel is dependents upon specific situation and taking 5x5 kernel is a good size for most cases and it return new type of image that we simply call blur applying the Gaussian blur by convolving image with a kernel of Gaussian values reduces noise in our image [9].

Figure 4: Image with reduced noise

3.3 CANNY EDGE DETECTION

It's basically used to identify edges in our image and calculates the gradient in all direction of our blurred image and to trace our strongest gradients as series of white pixels. **cv2.Canny(blur,low_threshold,high_threshold)**

These threshold in argument allows us to isolate the adjacent pixels that follow the strongest gradients.

We follow 3 steps for the gradient in threshold:-

1) If the gradient is greater than upper threshold then it is accepted as an edge pixel .

2) If it is below the lower threshold, then it is rejected.

3) If it is between the thresholds then it will be accepted only if it is connected to a strong edge [22].

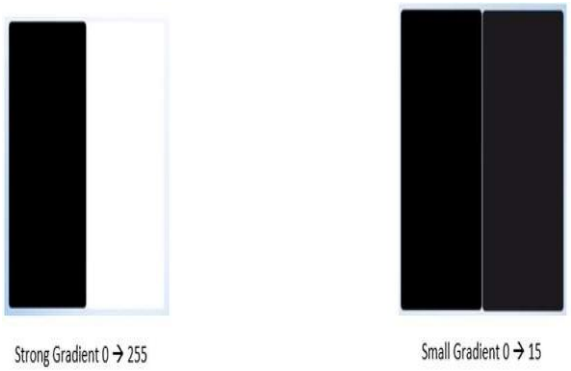


Figure 5: Change of Gradient

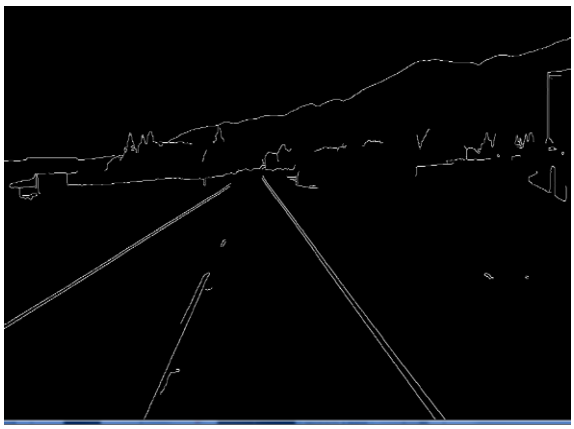


Figure 6: Canny image

3.4 REGION OF INTEREST

Now with the help of our axes we're going to set a limit by which we can extent our field of view based on their region of interest which traces a triangle with vertices of 300 pixels along the X axis and 800 pixels along the Y axis which would simply be the bottom of the image,1200 pixels along the X axis and once again the bottom of the image 800 pixels out the Y the very bottom and the last vertex will simply be 650 pixels along the X and 350 travelling on the Y axis ultimately tracing a triangle that isolates the region where we want to identify the lanes lines [13].

So goal is to create completely black image with same dimensions of our road image and fill part of its area with a triangular polygon. Now we will perform a bitwise AND operation with our canny image and the mask which will result in our final region of interest and define a function def region_of_interest .

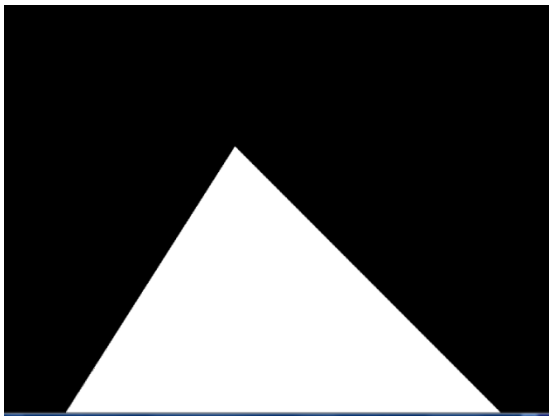


Figure 7: Masked Image

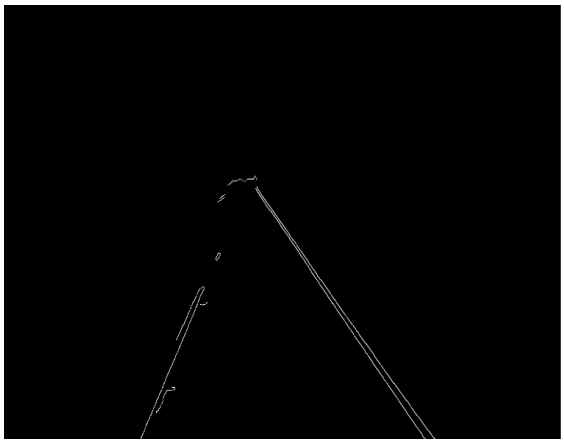


Figure 8: Image after doing a bitwise operation with the canny image and the masked image

3.5 HOUGH TRANSFORM

By using the HoughLine transform method we will going to see the working of Hough transform for line detection. It's used as extraction process of the image for its analysis. [20].

Basics of Hough line Method

Let us take a straight line which can be represented by

y = mx + c

and in parametric form, as

$r = x\cos\theta + y\sin\theta$ where

R = The perpendicular distance from origin to line.

θ = The angle formed by this perpendicular line and horizontal axis measured in counter-clockwise.

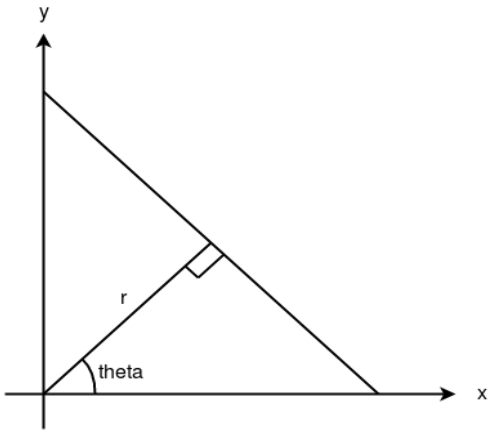


Figure 9: Representation of Hough Transform

So in order to represent a line it can be showed in two terms,
(r, θ).
We goanna split Hough space into a grid each bin inside of our grid corresponding to the slope and y-intercept value of a candidate line. Like taking about the points in x-y axis that corresponds the point of intersection are inside a single bin we're going to count the number of point of intersection in the bin, so bin which have maximum number of point of intersection will be the required line since it have been voted as line of best fit whatever be the value of M and B [11].

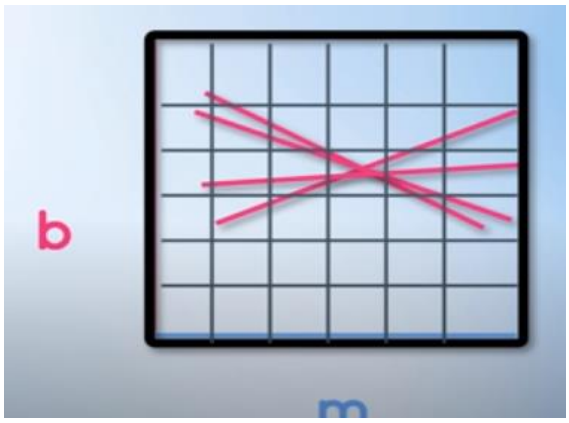


Figure 10: Hough space B versus M graph

3.5.1 Implementation of Hough Transform
Instead of using Cartesian coordinates, we will be using polar coordinates for showing vertical lines.
Result we get will be:

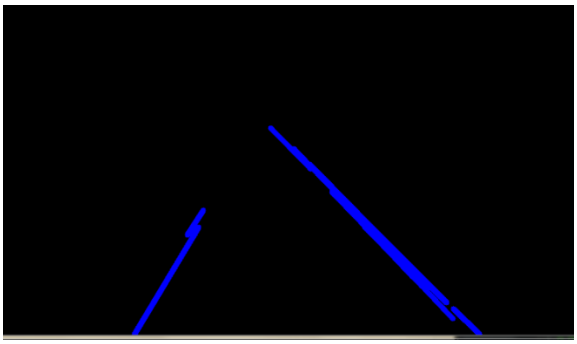
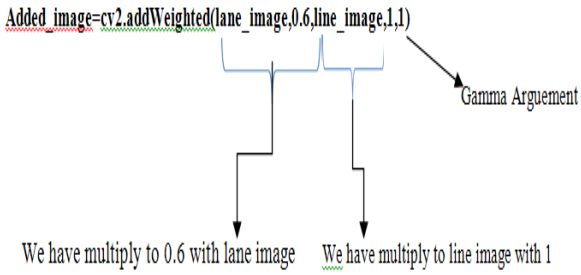


Figure 11: Image of the blue colored lines drawn on a zero-intensity image

Here detection of line is showed using Hough Transform and displayed it on black image. Now in our final step, we are going for blending process where we put this image to our original color image so that this lines should be shown on the lanes instead of black screen [18].



Now when we add these two arrays then line_image have 20% more weight which means that the line will be more clearly defined when we blend the two images while lane_image will be more darker [8].

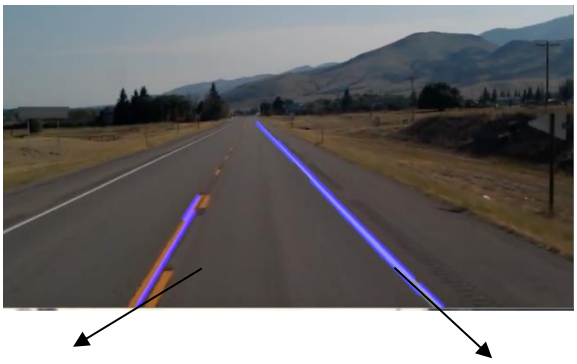


Figure *12: Lane image and line image on original color* image

3.5.2 Optimization of Hough Transform
Instead of having multiple lines, we can average out their slope and Y-intercept into a single line that traces out both of our lanes [4].

averaged_lines=average_slope_intercept(lane_image,lines)
Now we take
left_fit [ ] = which is equal to coordinates of the averaged lines on the left.
right_fit[ ] = which is equal to coordinates of the averaged lines on the right [19].
We now reshape the each line into a one dimensional array with four elements
x1 , y1, x2 ,y2 = line.reshape(4)
After that we use polyfit which fit this polynomial to our X and Y points and return a vector of coefficients which describe the slope and Y-intercept i.e
parameters =np.polyfit((x1, x2),(y1, y2),1)
Averaging the both line in single right and left array.
Since from above we have find the slope and Y-intercept but we don't know where we can draw our line on image i.e we don't have x1,y1,x2,y2 to fit each line .So
to fit each line [5].So for that we define a function make_coordinate (image, line_ parameters)

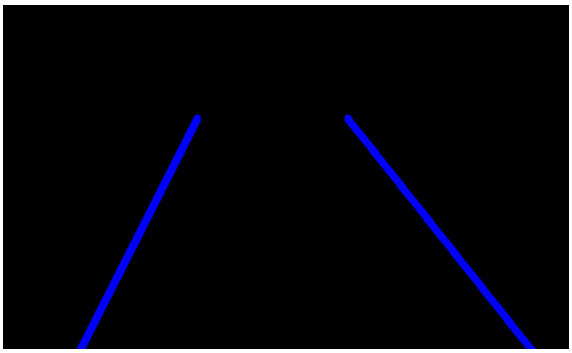


Figure 13: Averaged line image

Here instead of many lines it is averaged outline on each side.

Figure 14: Blending our line image with real image

3.6 Working Video

To capture the video in our workspace we need to create video capture object by setting [15]
Cap =cv2.VideoCapture ("test2.mp4")
Here after downloading video and giving its directory where video is kept
And putting while(cap.isOpened()) and capturing has been initialized we'll enter into a loop where use cap.read() to decode every video frame and what this return is Boolean and frame which we are projected to [23].

```
averaged_lines=average_slope_intercept(frame,lines)
line_image
=display_lines(frame,averaged_lines)
combo_image=cv2.addWeighted(frame,0.8,line_image,1,1)
cv2.imshow("result",combo_image)
 if cv2.waitkey(1) & 0xFF ==ord('q'):
     break
cap.release()
cv2.destroyedAllWindows()
```

Hence we will get a motion output of lane detection detecting each line of lane.

IV. CONCLUSION

In this paper, we introduced a new lane detection process where with the help of various methods like Gaussian Blur, Region of Interest and Canny edge Detection. The main objective is to provide the drivable lane for smooth and complicated roads. Its somehow decrease the chances of road accidents and thus saving human lives. Road detection and segmentation process provide an area of knowledge which can be developed in future times for development and welfare of human lives.
Many big automobiles companies have already started working and investing their resources in this area with coming of artificial intelligence in today's world, autonomous vehicle
prove to be reliable to it. Since the real struggle came when these automobile vehicle runs on road where many factors like improper lane marking ,bad lighting effects its movement and accuracy. In the future, we will continue to exploit more advanced road detection ways to improve its performance [24].

V. ACKNOWLEDGMENT

The authors acknowledge with thanks the support from Narula Institute of Technology, Agarpara, Kolkata, India for encouraging in research work and giving scope in this regard. We also thankful to the department of Electronics and Communication Engineering of Narula Institute of Technology, for carrying out the study. A special thanks goes to Dr. Anilesh Dey, Head of the Department, Electronics & Communication Engineering.

Authors acknowledge the immense help received from the scholars whose articles are cited and included in references of this manuscript. The authors are also grateful to authors / editors / publishers of all those articles, journals and books from where the literature for this article has been reviewed and discussed.

REFERENCES

1. D. Pomerleau, “RALPH: rapidly adapting lateral position handler,” in Proceedings of the Intelligent Vehicles '95. Symposium, pp. 506–511, Detroit, MI, USA, 2003.
2. J. W. Lee, C. D. Kee, and U. K. Yi, “A new approach for lane departure identification,” in Proceedings of the IEEE IV2003 Intelligent Vehicles Symposium, pp. 100–105, 2003.
3. J. W. Lee and U. K. Yi, “A lane-departure identification based on LBPE, Hough transform, and linear regression,” Computer Vision and Image Understanding, vol. 99, no. 3, pp. 359–383, 2005.
4. C. J. Chen, B. Wu, W. H. Lin, C. C. Kao, and Y. H. Chen, “Mobile lane departure warning system in,” in Proceedings of the 2009 IEEE 13th International Symposium on Consumer Electronics, pp. 1–5, 2009.
5. A. Borkar, M. Hayes, and M. T. Smith, “Robust lane detection and tracking with Ransac and Kalman filter,” in Proceedings of the 2009 IEEE International Conference on Image Processing, ICIP 2009, pp. 3261–3264, November 2009.
6. J.-G. Wang, C.-J. Lin, and S.-M. Chen, “Applying fuzzy method to vision-based lane detection and departure warning system,” Expert Systems with Applications, vol. 37, no. 1, pp. 113–126, 2010.
7. H. Xu and H. Li, “Study on a robust approach of lane departure warning algorithm,” in Proceedings of the EEE International Conference on Signal Processing System (ICSPS), pp. 201–204, 2010.
8. H. Chen and Z. Jin, “Research on Real-Time Lane Line Detection Technology Based on Machine Vision,” in Proceedings of the 2010 International Symposium on Intelligence Information Processing and Trusted Computing (IPTC), pp. 528–531, Huanggang, China, October 2010.
9. H. Aung and M. H. Zaw, “Video based lane departure warning system using hough transform,” in Proceedings of the International Conference on Advances in Engineering and Technology, pp. 85–88, Singapore, 2010.
10. J. Fritsch, T. Kuhnl, and A. Geiger, “A new performance measure and evaluation benchmark for road detection algorithms,” in Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC '13), pp. 1693–1700, IEEE, The Hague, The Netherlands, October 2013.
11. P.-C. Wu, C.-Y. Chang, and C. H. Lin, “Lane-mark extraction for automobiles under complex conditions,” Pattern Recognition, vol. 47, no. 8, pp. 2756–2767, 2014.
12. C. Mu and X. Ma, “Lane detection based on object segmentation and piecewise fitting,” TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 12, no. 5, pp. 3491–3500, 2014.
13. S. Srivastava, M. Lumb, and R. Singal, “Improved lane detection using hybrid median filter and modified hough transform,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 1, pp. 30–37, 2014.
14. V. Gaikwad and S. Lokhande, “Lane Departure Identification for Advanced Driver Assistance,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 910–918, 2015.

15. F. Yuan, Z. Fang, S. Wu, Y. Yang, and Y. Fang, “Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis,” IET Image Processing, vol. 9, no. 10, pp. 849–856, 2015.
16. J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao, “Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting,” Pattern Recognition, vol. 59, pp. 225–233, 2015.
17. P. N. Bhujbal and S. P. Narote, “Lane departure warning system based on Hough transform and Euclidean distance,” in Proceedings of the 3rd International Conference on Image Information Processing, ICIIP 2015, pp. 370–373, India, December 2015.
18. J. Son, H. Yoo, S. Kim, and K. Sohn, “Real-time illumination invariant lane detection for lane departure warning system,” Expert Systems with Applications, vol. 42, no. 4, pp. 1816–1824, 2015.
19. M.-C. Chuang, J.-N. Hwang, and K. Williams, “A feature learning and object recognition framework for underwater fish images,” IEEE Transactions on Image Processing, vol. 25, no. 4, pp. 1862–1872, 2016.
20. Y. Saito, M. Itoh, and T. Inagaki, “Driver Assistance System with a Dual Control Scheme: Effectiveness of Identifying Driver Drowsiness and Preventing Lane Departure Accidents,” IEEE Transactions on Human-Machine Systems, vol. 46, no. 5, pp. 660–671, 2016.
21. A. Mammeri, A. Boukerche, and Z. Tang, “A real-time lane marking localization, tracking and communication system,” Computer Communications, vol. 73, pp. 132–143, 2016.
22. Y. Ma, J. Xu, Y. Zhang, F. Liu & X. Luo, “Vector Separability Measurement Based Fast Feature Selection for Detecting Images Information Hiding”, IETE Technical Review, pp 56-71, https://doi.org/10.1080/02564602.2020.17699 27 May 2020.
23. C. Qin, Z. Qian, X. Li and J. Wang, “Artificial Intelligence Oriented Information Hiding and Multimedia Forensics”, IETE Technical Review, Vol.38:1, pp 1-4, 22 Feb 2021.
24. S.Tripathi, T. S. Sharan, S. Sharma & N. Sharma, “An Augmented Deep Learning Network with Noise Suppression Feature for Efficient Segmentation of Magnetic Resonance Images”, IETE Technical Review, https://doi.org/10.1080/02564602.2021.1937349, 09 June 2021.