Road lane detection through image and video processing using edge detection and Hough transform for autonomous driving purposes

Samane Sharifi Monfared
Bahçeşehir University
7 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Bilal Sedef
Bahçeşehir University
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Lavdie Rada
Bahçeşehir University
5 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Road lane detection through image and video processing using edge detection and Hough transform for autonomous driving purposes

Samane Sharifi Monfared*, Bilal Sedef*

Bahcesehir University, Besiktas, Istanbul and 34349, Turkey

Abstract

Nowadays, autonomous vehicles are on the board of fast development to facilitate driving for humankind. For performing this, the vehicles need to detect the lane of the roads to stay on suitable places to prevent car accidents. In this paper, we aim to implement and develop a computer vision method for road lane detection on both image and video using the OpenCV library. We are going to achieve this goal by a well-known method of line detection named Hough probabilistic transform. Before Hough transform in this study, we are using gray scale image, camera calibration, masking filter as preprocessing techniques and letter on, Canny edge detection as a method of edge detection. We firstly implement our method on image and then we will continue our work on video which is a more realistic case study, and we will show our detection with a red line on the screen [1].

Keywords: Lane detection, Image processing, Video processing, Canny edge detection, Hough probabilistic transform, Autonomous driving

1. Introduction

In this modern era, we cannot imagine our lives without using cars and vehicles and no one can deny the importance of good driving to stay safe and sound while chasing the road. Although, driving is an enjoyable task in short paths, we all agree that driving a car is a time taking, annoying, and dangerous work in long paths that may be followed with accidents and causing traffics, etc. Thus, in the past decade, scientist have tried to purpose the Advanced Driver Assistant Systems (ADAS) for autonomous vehicles to help driving be a safer and more fun task to do.

As a basic approach, ADAS is using cameras placed on the forehead of the car to collect images and videos for autonomous driving. In this study, we perform a Hough transform based approach to detect our lines. After having images and videos from the camera, our goal is to detect the lanes of the road in the best way.

* Corresponding author. Tel.: +90-531-514-8870 / Tel.: +90-505-615-2277
E-mail address: samane.monfared@bahcesehir.edu.tr / bilal.sedef@bahcesehir.edu.tr

Firstly, we get help from grayscale and blurring to perform some preprocessing on our image, after that, we implement mask filtering to focus on the area of interest. Furthermore, we implement Canny edge detection on our image to detect our mane edges which consist of our lane. Then, we perform Hough line transform as our main operator and we will paint our lines into an empty image. At the end, we will show our lanes on the original image with red color. After this process, we will perform the same method on the video which is more realistic [1].

2. Materials And Methods

In this paper, we used methods of both image and video processing to identify the lane of the road and show it on image and video using OpenCV library. For this aim, we used following features as our approach in the way that following figure is shown [2].

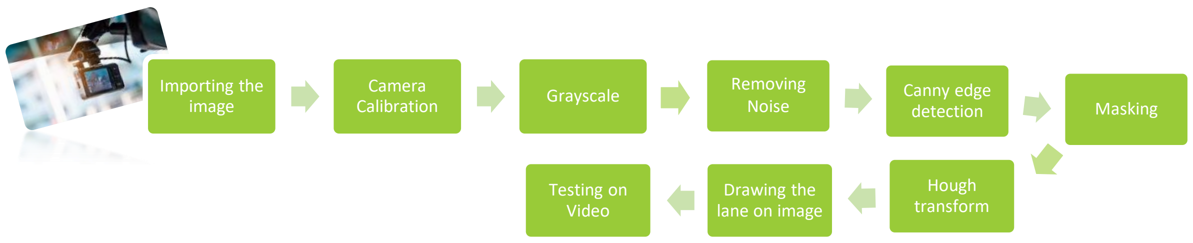

Fig. 1 Processes used in this paper for lane detection.

2.1 Camera calibration

Many cameras have the problem of lens distortion. We use camera calibration to solve this problem in our project. Distortion or in another word, camera sectioning is one of the consequences of parameters of lens and image sensors as we can see an example in Figure. 2 [3]

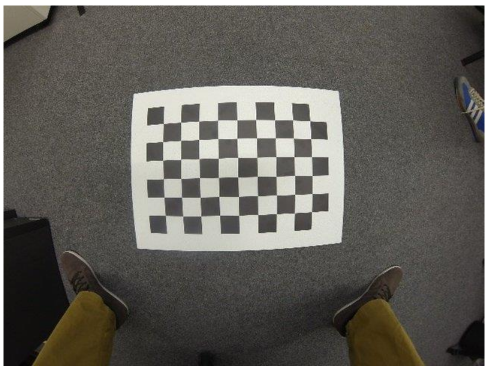

Fig. 2 Camera sectioning [3]

2.2 Gray scale

We all know that the lanes on the roads are in light colors mostly white and yellow. With this in mind, we can say that a gray scale image is more suitable to detect the lighter lines on a darker road. Furthermore, a gray scale image is a help in the edge detection phase using Canny edge detection.

We had shown the gray scale of Figure 3 on the Figure. 4. In RGB images we have 3 values for Red, Green, and Blue from 0 to 255 which means around 0 is for the darker values and around 255 is for the lighter, this means that black is [0,0,0] and white is [255,255,255]. For making a gray scale image from a color image we need to get the average of these three values, and this will give us the gray scale of our image [4].

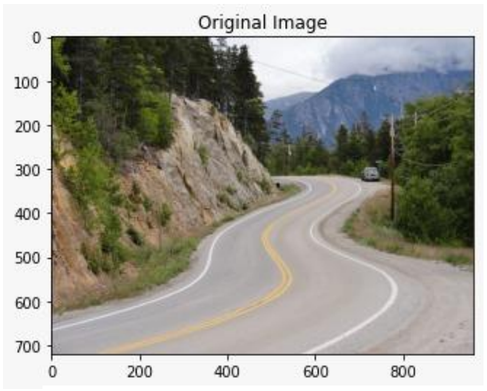

Fig. 3 Our test image, it is one of the most dangerous roads in US. Image from: Udacity

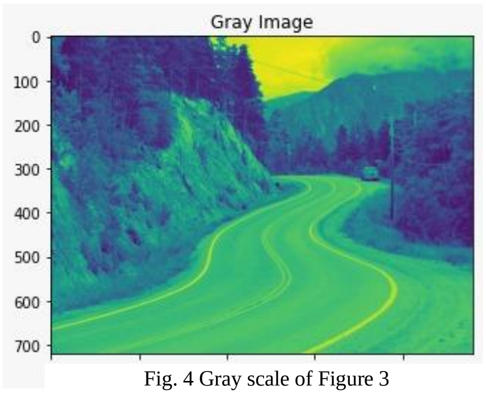

Fig. 4 Gray scale of Figure 3

2.3 Removing the noise

In our project we use methods based on derivative. Thus, considering that derivative is very sensitive to noise, we should manage the noise before performing any mathematical operation. One way of handling the noise of the image is blurring the picture using a small amount of Gaussian blur. We applied Gaussian kernel in 5x5 windows size. As you know the smallest the kernel size, the more invisible the blur, thus, with this small window, we will have smother images. It can be seen in the Figure 5.

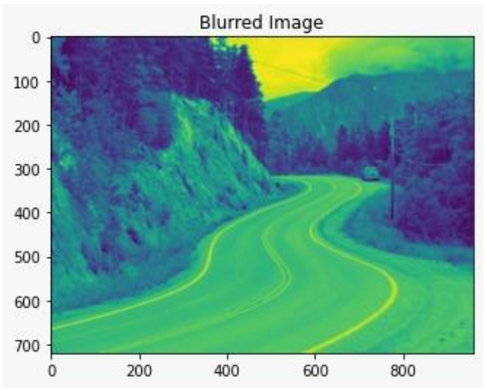

Fig. 5 Removing the noise with gaussian kernel blur

2.4 Masking the region of interest

Considering that our images and videos come from a camera placed on the forehead of the car named Dashcam, we can choose a certain part of our image as the area of interest to do lane detection. It is a triangle defined by the parameters of the lens of our imaginary camera [5]. The region of interest in our test image had been shown in Figure 6.

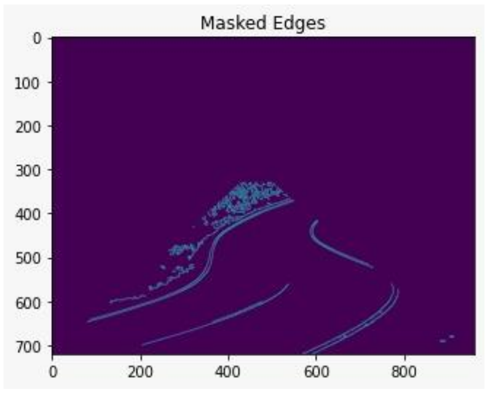

Fig. 6 Masking the region of interest from point of view of Dashcam

2.5 Canny edge detection

Canny edge detection is a well-known edge detection method that is mostly used to find boundaries in computer vision purposes from their derivative. In this method proposed with John F. Canny, we use the gradient of pixels to detect dense regions when we have changed intensity [6][7]. We can see the result of the Canny edge detector in our test image in Figure. 9. Here again we can see that a Grayscale image is useful because it can be shown as a matrix of x and y that we can perform mathematical operations on it.

2.5.1 Calculating the gradient

After having a smoother image, we can get the derivative of the image in the x direction and in the y direction to detect the edges using a Sobel kernel. Now, we have two images include first derivatives of our image. Thus, we can calculate the gradient using (1) with values of these two images. As we can see in the Figure. 2, having derivatives of the image will show us the edges of our picture itself, however getting the gradient is giving us the direction of our edges (2), as well as the edges itself [7]. Furthermore, in the next operations, thanks to gradient we can detect unwanted edges to remove. We can see an example of this operation in Figure. 7 and Figure. 8.

$$Edge\ Gradient\ (G) = \sqrt{{G_x}^2 + {G_y}^2} \tag{1}$$

$$Angle\ (G) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

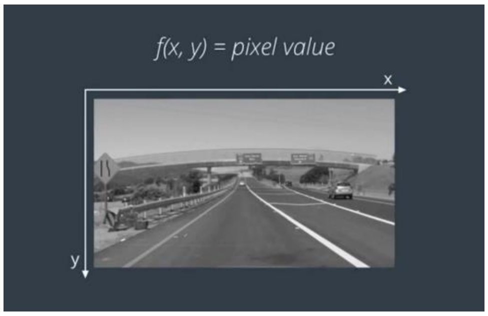

Fig. 7 Function of a Gray scale. image from Udacity

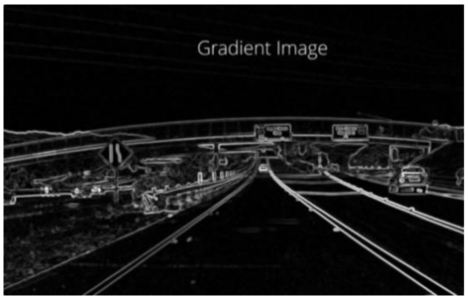

Fig. 8 Gradient image of figure 8.

2.5.2 Non-maximum suppression

As we have the values of gradients and directions, we can process all the pixels of our image to delete any pixel that is not a real edge. Thus, we are checking each pixel with maximum value in its neighbourhood to be in the same direction with the gradient of that pixel [7].

2.5.3 Edge Tracking by Hysteresis

As the last task of the Canny edge detection, we want to make sure that our edges are real edges. For this aim we define a minimum value for our intensity gradient which each edge below this value is not an edge and we define a maximum value for our intensity which declare the edges above this value are edges for sure. For the values between this minimum and maximum we decide based on their connectivity [2].

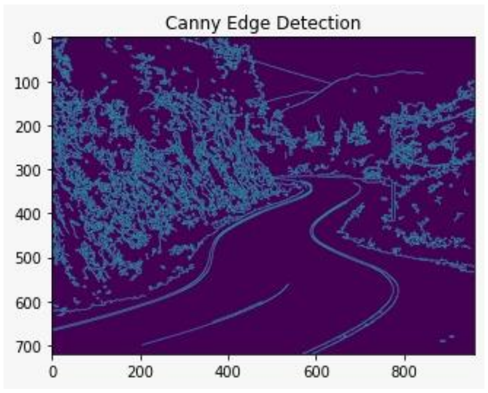

Fig. 9. The result of Canny edge detector on our test image

2.6 Hough probabilistic transform

Hough transform is a novel algorithm to build lines which is used in many autonomous driving projects. Using Hough transform, we can manage to connect edge pixels from Canny edge detector together and make a line. As the out put of Hough transform, we have all the dashed and solid lane lines in the given image [8].

In this project, we made use of the Hough Transform algorithm as one of the most important factors that enabled us to achieve our main goal. So, we want to talk more about the mentality underlying Hough Transform. Hough transform technique is a frequently used object detection method with a very high efficiency. Hough transform allows to perform digital image processing mathematically. The Hough Transform was developed by Paul Hough in 1962. This method is used to find the presence, position and angles of shapes. The Hough Transform technique is mostly used to detect lines in any image. Hough Transform is used in 3 different forms as linear, circular and general.

There are many applications in image processing where it can be used to find geometric shapes from numerical data in images. Finding the iris, finding the plate, finding the ball on the field, finding the portrait on the wall, finding the pen on the table or finding the smooth/uneven defective area on the object can be detected using Hough transform algorithms.

The biggest advantage of the Hough transform is that it can give successful results regardless of the noise in the image. Thanks to the parameters it contains, the Hough transform reduces the time spent searching for points that define the boundaries of objects in the image. Despite these advantages, the Hough transform also has some disadvantages. Due to the high data density in large-size images, it may take longer than usual to detect objects in the image.

The Hough transform works with the logic of determining which geometric shape the lines detected in the image belong to, by interpreting the results statistically. Shape detection using the Hough transform can generally be summarized with the following steps [9][10].

- Edges are determined in the source image.
- The image is made binary (black, white) using a thresholding method.
- For each edge pixel, the possible shapes of each edge pixel are graded by increasing the polar coordinate values of the possible geometric shapes that the point may be on, one by one on a matrix used.

- Since the shapes with the highest matrix value are the ones with the most votes, they are likely to be found in the image.

In Hough Space, the horizontal axis represents the slope, and the vertical axis represents the interceptor of a line. Hough Space is a 2D plane. If an edge is detected and its line is drawn, a line on that line is represented as y = ax + b (Hough, 1962). In the Hough Transform, the edging stripes and their intersections are characterized and joined on a point in Hough space. An infinite number of lines can pass through an edge point (x_i, y_i) in the edge image. Therefore, an edge point produces a line b = ax_i + y_i in Hough Space (Leavers, 1992). In the Hough Transform algorithm, Hough Space is used to determine whether a line is present in the edge image. [11]

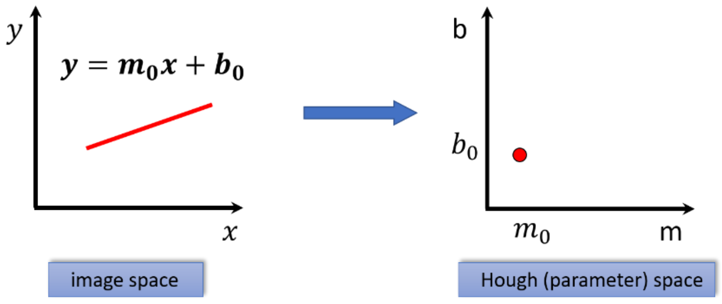

Fig.10 Hough Transform

In Hough Space, lines are represented as y= ax+b and these lines of defect, slope, etc. intersect at a single point. As per the formula, vertical lines converge to infinity, so the algorithm does not work to detect vertical lines (Leavers, 1992). In terms of computer science, this means that a computer would need an infinite amount of memory to represent all possible values of a. To avoid this problem, a straight line is represented by a line called the normal line that passes through the origin and is perpendicular to this straight line. The form of the normal line is ρ = xcos(θ) + ysin(θ); where ρ is the length of the normal line and θ is the angle between the normal line and the x-axis.

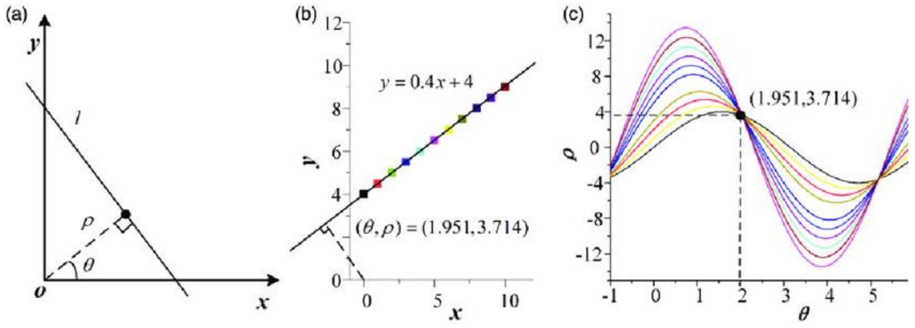

Fig. 11 Example straight line to normal line transform

Thus, we will not have to represent the Hough Space with the slope a and the intersection of b. We will now represent by ρ and θ, where the horizontal axis is for θ values and the vertical axis is for ρ values. Mapping the edge points to the Hough Space is again a similar process. However, an edge point (x_i, y_i) is now represented by a cosine curve in Hough Space instead of a straight line (Leavers, 1992). Thanks to this normal representation of lines, the problem of infinite straight lines passing through a single point is also eliminated.

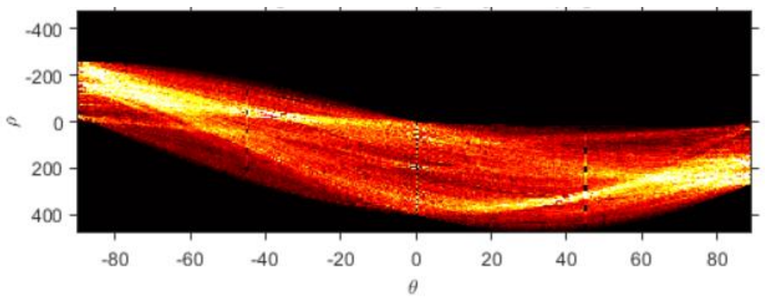

Fig 12. Cosine Transform of edge lines

As we mentioned above, an edge point produces a cosine curve in Hough space. From this, if we map all edge points in an edge image to Hough Space, a large number of cosine curves will result. The basic logic here, and indeed the Hough Transform feature that we will use for line detection, is that when two edge points are on the same line, the cosine curve corresponding to their transformation will intersect on the same (ρ, θ) pair, and this intersection will be seen in their normal representation. In our algorithm, where we look for a certain number of intersections, when we see a sufficient number of intersections (ρ, θ) on pairs, Hough Transform detects that it is a line [12].

2.7 Drawing average equation of lanes

After having lines, we need to get the average equation of the lines to show the lanes on the output and have smoother lines at the end. We perform this operation knowing that our left lane and right lane should be separated. Then we will draw a solid average for our lane lines. The Figure 13 is our result on the test image.

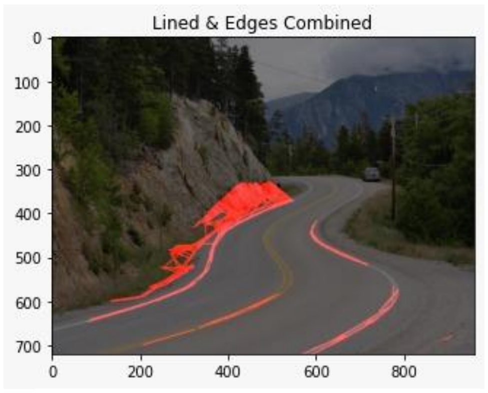

Fig. 13 Result image

2.8 Testing on video

Lane detection on video is slightly different from lane detection on a still image. We must do all the algorithms we apply on a still image in real time and continuously. It should process the frames of the images in a video separately, identify the corners with canny edge detection on these image frames, mask the image, and determine the lines with a certain length with the Hough transform, then find the slopes of these lines to decide whether they are on the right or left side, and finally we must overlay it on the real image by specifying the road lines as the path and then repeat this for the next frame [13] [2].

3. Implementing the methodology

3.1 Finding lanes of the road on the image

For detecting lanes on the road from the point of view of a dash camera, we had decided to collect our own data to sets our code. These 4 pictures are taken form deserts in Iran, with different modes and sizes of image to test that we can see in Figure 14. First, we had imported them in our python code then we had implemented methods discussed in the Material and methods section one by one.

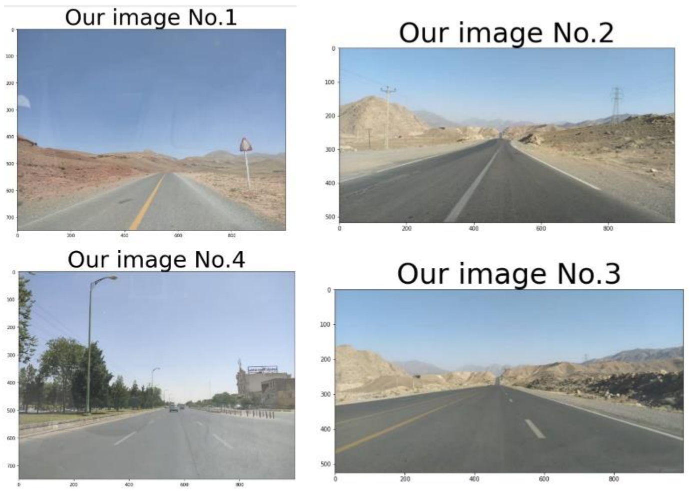

Fig. 14 Test images for our research. Images taken by Samane Sharifi Monfared

3.1.1 Camera calibration

We first did camera calibration on our images using new version of OpenCV library. We had used the cv2.undistort() function for this aim.

3.1.2 Gray scaling

Again, we had used OpenCV library for applying gray scale on our images using cv2color() function with the parameter of COLOR_RGB2GRAY.

3.1.3 Noise removal

We had used gaussian blur with kernel size 5 on the gray scaled images, using cv2.GasussianBlur() function from OpenCV.

3.1.4 Canny edge detection

We had used cv2.Canny() function of OpenCV with low threshold of 50 and high threshold of 150 on the blurred images.

3.1.5 Masking the region of interest

First, we had made a black image with the exact size of our image, using np.zeros_like(our image) function from Numpy library. Then, we get the length and the height of our image with image.shape and we use these two values to make a triangle of the region of our interest. Now, we can make a mask using cv2.fillPolly() function. Then, we use cv2.bitwise_and() to overlay our result of Canny edge detector with masked picture.

3.1.6 Hough transform

Now we perform Hough transform to extract our edges to lines, using cv2.HoughLinesP() function of OpenCV. Keeping this in mind that Hough Transform need parameters to get defined, we had defined distance resolution in pixels of Hough grid as 1, angular resolution in radians of the Hough grid as π/180, minimum number of intersections in Hough grid cell as 10 and the maximum gap in pixels between connectable line segments as 30. Again, we need a blank image to draw our lines provided by Hough transform on it and we use a for loop to draw each line we had from Hough transform using cv2.line() function. In Figure 15, an example of Hough transform is shown.

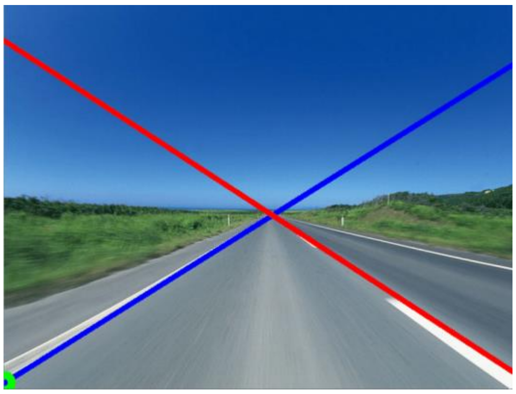

Fig. 15 Hough transform on lane detection example [15].

3.1.7 Drawing the lines in the original images

At the end, we had used cv2.addWeighted function of OpenCV to draw our final detections of lanes on the road of each image. Our result in 4 test images, is shown in Figure 16.

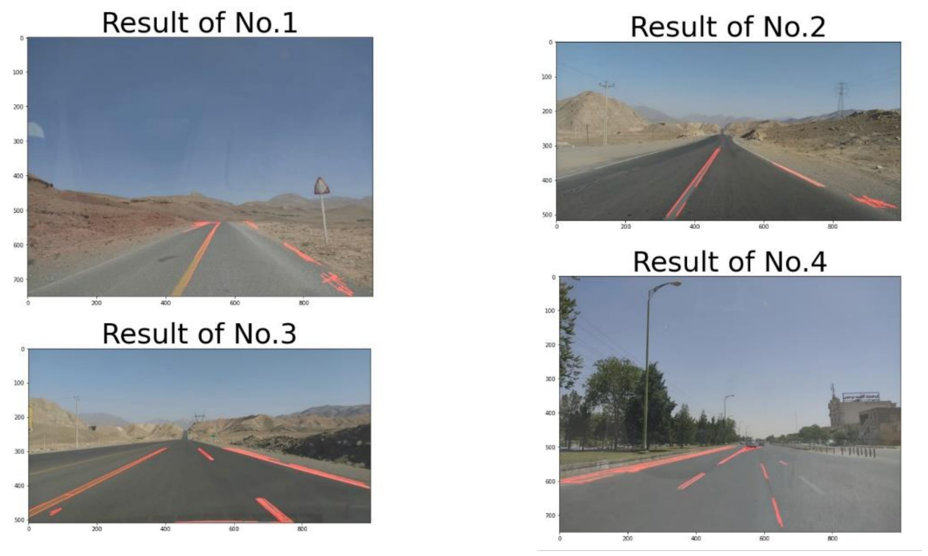

Fig. 16 The result of our lane detection on our test data

3.2 Finding lines of the road on the video

In video processing, what we need to do before all the steps is to detect all the lines in the image frame as we can see an example of this process in Figure 17. Here again, we make use of the Hough transform and Canny edge detection algorithms.

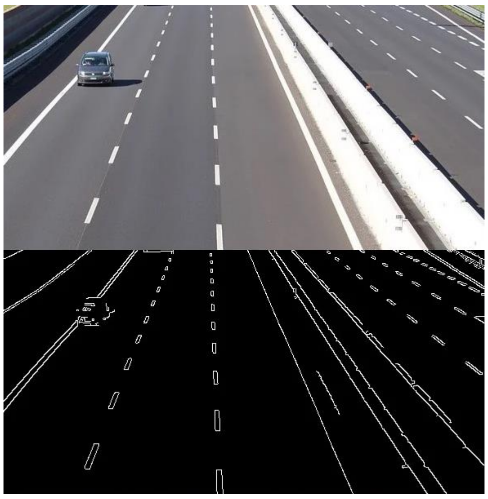
Fig. 17 Detection of all the lines

3.2.1 Finding the Slopes of Lines

Since this is a video rendering, it will not be enough for us to just find the road lines. For our vehicle to make a correct judgment about the route, we need to understand whether the road lines coincide with the right or left side of the vehicle. We do this by calculating the slopes of the road lines we have determined.

$$slope = (y2 - y1)/(x2 - x1) \tag{3}$$

Here, the slope represents the lines identified by Hough Transform, each considered separately. X and y values represents the lower and upper left and right points of lines. Our slope threshold is 0.5 to count a line as curved. After counting a line as curved and calculating its slope, we need to also appoint it as a left lane or right lane. For this, we need to divide the image to half, vertically.

$$center = img.shape[1]/2 \tag{4}$$

If a line's x values are higher than the center value and the slope is higher than 0, we can safely appoint this line as a right lane. Vice versa as a left lane.
After appointing lines as right or left lanes, while using their averaged x and y values, we apply polynomial fitting to create a first-degree polynomial.

$$right\_m, right\_b = np.polyfit(right\_x, right\_y, 1) \tag{5}$$
$$left\_m, left\_b = np.polyfit(left\_x, left\_y, 1) \tag{6}$$

We solve this equation by choosing a y value to find our new x values which will be used to create a linear line across the image frame. After these steps, we will have a linear line which represents the right or left lanes overlayed on a video as real time [14].

4. Experimental result

We had used two factor to determine our method in this study. The accuracy of our lane detection in various cases which is shown in Figure. 13 and time taken to execute the result which is shown in Table 1.

Table 1.

Test Case	Time of execution
Test Image 1	Almost 0s
Test Image 2	Almost 0s
Test Image 3	Almost 0s
Test Image 4	Almost 0s
Test Video 1 (10 s long)	54s
Test Video 2 (7 sec long)	52.3s
Test Video 3 (12 sec long)	1 min 1s

5. Conclusion and further research

As it is shown in previous figures, our results in lanes detection in images our videos without many curves in their lane were incurably amazing, especially when it comes to images and videos from highways. However, we can say that we need some improvement in the case of curve detection. Another negative point of our project was that sometimes our algorithm will detect some other objects near the lanes as lanes. As the following research, we can do some improvements include use of second-degree polynomial to detect curved lanes and use more accurate lane detections method to achieve better lane detection.

Contributions:

Responsibilities of By.	**Mr. Bilal Sedef**	**Mrs. Samane Sharifi**
Coding (Pre-processing)	50%	50%
Coding (Hough transform)	50%	50%
Coding (Lane detection on Image)	25%	75%
Coding (Lane detection on Video)	75%	25%
Reporting (Pre-processing)	50%	50%
Reporting (Hough transform)	50%	50%
Reporting (Lane detection on Image)	25%	75%
Reporting (Lane detection on Video)	75%	25%
Total (Average)	**50%**	**50%**

References

[1] C.Y. Kuo, Y.R. Lu and S.M. Yang, On the Image Sensor Processing for Lane Detection and Control in Vehicle Lane Keeping Systems, International Program on Energy Engineering, National Cheng Kung University, Tainan 70101, Taiwan, 8 April 2019, https://www.mdpi.com/1424-8220/19/7/1665/pdf.

[2] Archit Rastogi, Computer Vision: Lane Finding Through Image Processing, 6 Sep 2020, https://medium.com/swlh/computer-vision-lane-finding-through-image-processing-516797e59714

[3] Qi, Wang & Li, Fu & Zhenzhong, Liu. (2010). Review on Camera Calibration. 3354 - 3358. 10.1109/CCDC.2010.5498574.

[4] Kanan, Christopher & Cottrell, Garrison. (2012). Color-to-Grayscale: Does the Method Matter in Image Recognition. PloS one. 7. e29740. 10.1371/journal.pone.0029740.

[5] Chin-Pan Huang, Chaur-Heh Hsieh, Jin-Long Li. A Robust Lane Detection Method Using Adaptive Road Mask. Department of Computer and Communication Engineering, Ming Chuan University, Taiwan. (2015) https://www.academia.edu/10351901/A_Robust_Lane_Detection_Method_Using_Adaptive_Road_Mask

[6] Samuel, Moveh & Mohamad, Maziah & mad saad, Shaharil & Hussein, Mohamed. (2018). Development of Edge-Based Lane Detection Algorithm using Image Processing. JOIV: International Journal on Informatics Visualization. 2. 19. 10.30630/joiv.2.1.101.

[7] Canny J. F., 1986. A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Machine Intel, vol. PAMI-8, no. 6, pp. 679-697.

[8] Matas, Jiri & Galambos, Charles. (1998). Progressive Probabilistic Hough Transform.

[9] Rice, John & Hubbard, Graham & Beamish, Paul. (2007). Strategic Management: Thinking, Analysis and Action, Third Edition, Pearson Prentice Hall, Frenchs Forest.

[10] Fokkinga, Maarten. (2011). The Hough transform. J. Funct. Program.. 21. 129-133. 10.1017/S0956796810000341.

[11] Bahathiq, Adil. (2010). The fallopian tube in infertility and IVF practice. Cambridge University Press USA..

[12] R. O. Duda, and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", the Comm. ACM, Vol.15, No.1, 1972, pp. 11-15.

[13] Kim, Sunghoon & Park, Jeong-Ho & Cho, Seong & Park, Soonyoung & Lee, Kisung & Choi, Kyoungho. (2007). Robust Lane Detection for Video-Based Navigation Systems. 2. 535 - 538. 10.1109/ICTAI.2007.20.

[14] Zuroski, Kathyrn. (2021). Finding the slope of a line using Interactivate.

[15] Lim, Kai Li & Braunl, Thomas. (2019). A Methodological Review of Visual Road Recognition Procedures for Autonomous Driving Applications.

[16] Sharma, Anmol & Kumar, Maneesh & Gupta, Rajkamal & Kumar, Rajesh. (2021). Lane Detection using Python.

[17] Pansambal, Suvarna & Kumar, Udaya. (2019). Lane Datasets for Lane Detection. 0792-0796. 10.1109/ICCSP.2019.8698065.

[18] Lu, Pingping & Cui, Chen & Xu, Shaobing & Peng, Huei & Wang, Fan. (2021). SUPER: A Novel Lane Detection System. IEEE Transactions on Intelligent Vehicles. PP. 1-1. 10.1109/TIV.2021.3071593.

[19] Mongkonyong, Peerawat & Nuthong, Chaiwat & Siddhichai, Supakorn & Yamakita, Masaki. (2018). Lane detection using Randomized Hough Transform. IOP Conference Series: Materials Science and Engineering. 297. 012050. 10.1088/1757-899X/297/1/012050.

[20] Dai, Xingjian & Xie, Jin & Qian, Jianjun & Yang, Jian. (2020). Lane Detection Combining Details and Integrity: an Advanced Method for Lane Detection. 1-6. 10.1145/3446132.3446145.

[21] Fan, C. & Xu, J. & Di, S. & Shi, X.. (2013). Lane detection based on parallel Hough transform. Journal of Computational Information Systems. 9. 4893-4900. 10.12733/jcis6333.

[22] Xiao, F., Jin, L. & Haopeng, W. (2010). A study of image retrieval based on hough transform, 3rd IEEE Computer Science and Information Technology (ICCSIT), Chengdu, China