TECHNICAL UNIVERSITY OF CLUJ-NAPOCA – COMPUTER SCIENCE DEPARTMENT

Driving assistance systems:

Lane line detection

Péntek Tamás

Technical University of Cluj-Napoca

Computer Science Department

Cluj-Napoca, Romania

pentek.tomy@gmail.com

1. Introduction

Traffic accidents have become one of the most serious problems. Vehicle crashes remain the leading cause of accidental death and injuries in most traffic congested countries. Most of these transportation deaths and injuries occur on the nation's highways. The reason is that most accidents happen due to negligence of the driver. A large number of accidents can be avoided if a dangerous driving condition is detected early and warned to other drivers.

Developing a driver assistance system is very important in the context of road conditions. Suppose there is a system with integrated motion camera and an integrated onboard computer with the vehicle, a simple driver guidance system based on frame by frame analysis of the motion frames can be developed and thereby generate the alarm signals accordingly, so that the

driving can be made much safer. Advanced Driving Assistance System (ADAS) has recently been proposed to predict driver's intent, warn the driver about possible lane departure, and assist lane keeping, and it has been implemented on several vehicles in automobile industry. The road lane detection and object detection are also the other important way that we can improve the safety in roads.

As the name itself indicates is a process of detecting the lanes where the traffic circulates. For advanced driving assistances, the lane detection is one of the essential functions. The lane detection has become very specific term that implies the utilization of different sensors, cameras and algorithms to perform this functionality. The lane detection is a process that have to be effective and have to be done in real time. A good quality of lane should not be affected by shadows (trees, buildings), the change of light condition, the dirt left on the road surface. If we have to teach a computer to understand the road scene, the first step is lane detection. Everything else we detect from cars to zebra lines exists in context to the lanes on which we are driving.

The lane detection will provide driving person and passengers confidences even in the different lighting and different environments situations. Using this technology, we can provide the safety in roads to achieve a safer environment in any traffic condition.

2. Related work

The first research paper [1] describes a method for lane detection and for controlling a vehicle to keep the lane on the road. The vision-based lane detection is in three common steps: (1) image transformation from the sensor's frontal view to bird's eye view, (2) edge-based lane feature detection, and (3) lane markers regeneration or reconstruction in the processing image. In lane detection, the sensor image from a dashcam first has to be transformed from frontal view into bird's eye view so that the lane markings on become parallel for accurate vehicle positioning. Figure 2a illustrates the inverse perspective mapping where the sensor image is transformed by inverse perspective mapping (IPM) into a bird's eye view image. The horizontal position of the left and right lane marking can be determined by the column intensity sum of the

threshold image, and two lines LL and LR passing through xL and xR, respectively, are the initial guess of the position of the left and right lane marking. Their slope can be determined by the two windows of width w/2 and height b at horizontal position as shown in Figure 4a. By the intensity center of each of the two windows, the slope of LL and LR can be determined as indicated in Figure 4b.

(a) sensor image (b) bird's eye view image

Figure 2. Illustration of the inverse perspective mapping (IPM) by converting (**a**) the region of interest (in box) in the sensor image to (**b**) the bird's eye view image for detecting the parallel lane markings.

Lane markings close to the vehicle are approximately vertical after the transformation, thus the horizontal position of the left and right lane marking can be determined by the column intensity sum of the threshold image, and two lines LL and LR. This is the initial guess of the position of the left and right lane marking. Their slope can be determined by the two windows of width w/2 and height b at horizontal position as shown in Figure 4a. By the intensity center of each of the two windows, the slope of LL and LR can be determined as indicated in Figure 4b.

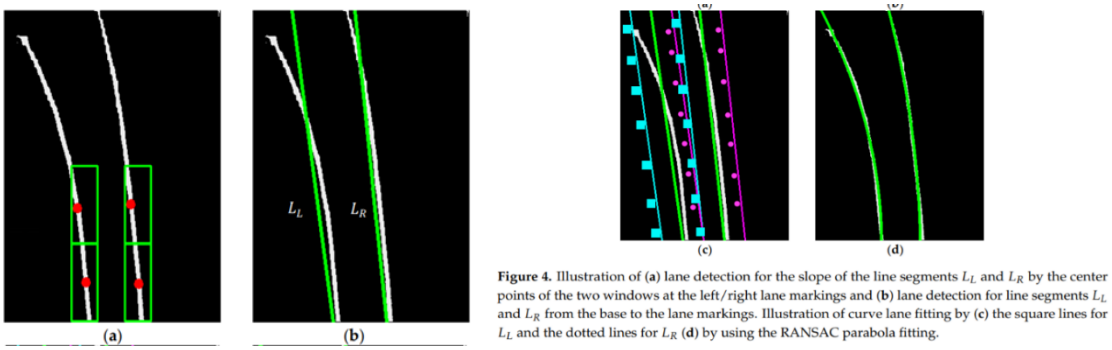


Figure 4. Illustration of (**a**) lane detection for the slope of the line segments L_L and L_R by the center points of the two windows at the left/right lane markings and (**b**) lane detection for line segments L_L and L_R from the base to the lane markings. Illustration of curve lane fitting by (**c**) the square lines for L_L and the dotted lines for L_R (**d**) by using the RANSAC parabola fitting.

In order to obtain a better position estimate in curve lane markings, random sample consensus (RANSAC) parabola fitting is applied around LL and LR for lane feature detection. Using the central limit theorem, a parabola can be estimated accurately as shown in Figure 4d. The highest accumulated intensity value in each window can be used to locate the position of each lane markings. Accurate vehicle position with respect to the lane center is necessary for lane control to minimize the cross-track error and keep the vehicle driving safely in the desired lane.

The second research paper [2] is a Master's Thesis, in which is presented a detailed description about road lane-detection based on heuristic algorithm and vehicle-detection. The method starts by thresholding each frame by keeping only the brightest regions of the image. A clustering methodology is used to group the detected points and a best fit line is then fitted to the remaining points. Firstly, is described Hough-based detection and Color-based detection individually. In order to do Hough-based detection, the image is converted into gray scale, applying Region of Interest (ROI), using global thresholding, using edge detection, and finally find lines by Hough Line Transform. The result of lane detection using Heuristic Algorithm is considerably better than method that is just based on Hough Line Transform. The captured color image is converted to gray scale by using Optimum Global Thresholding with the help of Otsu's method which solves the problem of illumination, and make method less sensitive to scene conditions. The main principle of Otsu's method is choosing threshold to maximize the interclass variance of the black and white pixels. To find lane boundaries, one of the edge detection methods, called Canny Edge Detection, is used and the detected boundaries are shown in figure 11.

Figure 11 Canny Edge Detection

The most important characteristics of Canny method are that the error rate of this method is low because this algorithm uses double thresholding, hysteresis thresholding. Standard Hough line

transform is used to find the lines on the image. The results of edge detection step are shown in Figure 17.

Figure 17 Line Detection Using Hough Line Transform

Color-based detection is used to extract more information about the lines and line boundaries based on their color information to improve the efficiency of detection. First step is Color Segmentation, and second step is Noise Reduction. In lane detection using HSI model, saturation component S and intensity component I are more important than hue component H, and give more consequential information. Captured image from the roads contains white lines and gray background of road. High contrast between gray color of road and white color of lines cause higher values of saturation S and intensity I components rather than hue component H. To remove noise from the segmented image, Morphological operation is used: Opening operation to remove small objects that are responsible for the noise, and then Closing operation to make the lane boundaries clearer and softer. The output after removing the noise from segmented image is shown in Figure 19.

Figure 19 color segmentation

A clustering methodology is used to group the detected points and a best fit line in the mean least squares sense is then fitted to the remaining points. In figure 20 the result of the presented algorithm is shown. This system demands low computational power and memory requirements,

and is robust in the presence of noise, shadows. The resulted images can be used for lane tracking or road following.

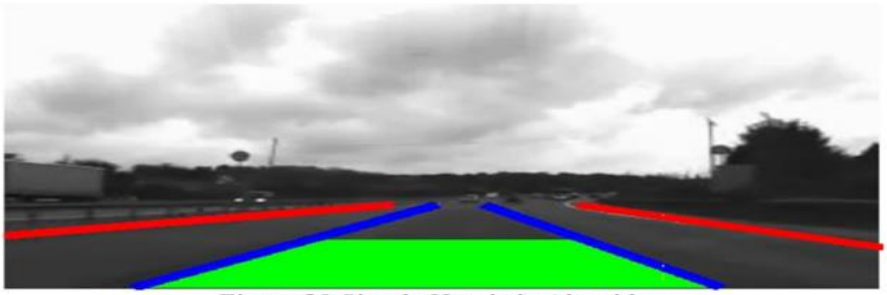


Figure 20 Simple Heuristic Algorithm

The third research paper [3] presents real-time lane detection and lane departure prediction. The input of the system is a color video frame. It transforms color image into gray-scale image for every frame. Only the pixels in Region of Interest (ROI) were extracted for edge detection. The ROI region was about 1/3 part of a full frame. The linear Hough transform (HT), a popular line detection algorithm, is widely used for lane detection. The HT is a parametric representation of points in the edge map. It consists of two steps, "voting" and "peak detection". In the process of voting, every edge pixel P (x, y) is transformed to a sinusoidal curve. The resulting values are accumulated using a 2D array, with the peaks in the array indicating straight lines in the image. In the process of peak detection, it involves analysis of the array accumulation to detect straight lines. Hierarchical pyramidal approaches have been proposed to speed up the HT computation process through parallelism. These hierarchical approaches filter candidates to be promoted to the higher levels of hierarchy by the threshold of the accumulation spaces. For each candidate that qualifies, they perform a complete HT computation again. To speed up the HT by parallelizing the process, additional costs are incurred for recomputing HT at every level. In this research paper, a modified approach is proposed to accelerate the HT process in a computationally efficient manner, thereby making it suitable for real-time lane detection. After the process of edge detection, next step is lane line identification. After all points in left line and right line are traversed and some parameters are calculated, all the elements of parameter space are already calculated. If the value of the parameter is more than the threshold after the setup of parameter space, the road lane is existing. The parameter will be kept for the next step to draw the corresponsive lane. If the value of parameter is less than the threshold, the road lane is not existing. The position deviation of lane obtained from two or more initial detecting process is small, so the position of lanes is predictable. After the lane recognition process, the prediction of lane departure starts to work. The departure prediction is only made in even frames for reduction of computation.

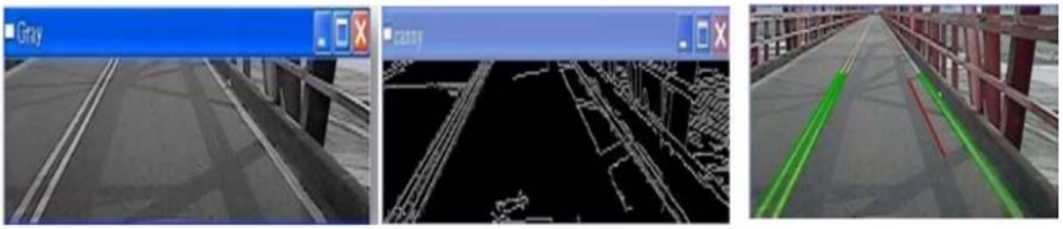


Figure 6. (a) After the process of gray image in ROI region; (b) After the process of edge detection in ROI region; (c) Prediction of road lane line.

The fourth research paper [4] describes a lane detection algorithm which uses a different color space, gradient filter and histogram to achieve the goal. One way to separate and detect objects in an image is to use color transforms and gradients to generate a filtered-down thresholded binary image. For color transforms, were tested three color spaces, HSL, LAB, LUV, in order to find out which one is best at filtering the pixels representing the lane line. The result was that the B channel of the LAB color space and the L channel of the LUV color space are the best. The Sobel gradient filter was also used, this is a type of gradient filter that uses Gaussian smoothing and differentiation operations to reduce the effects of noise.

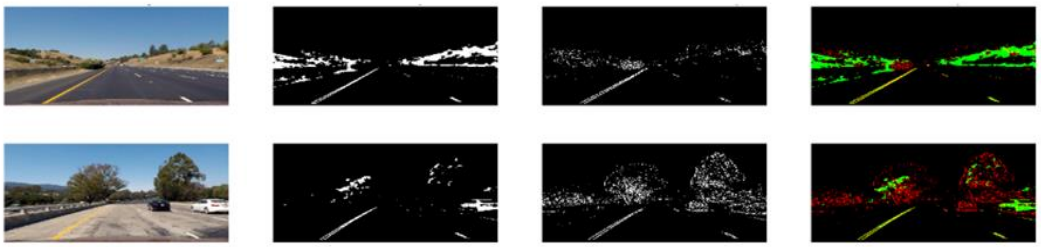


Figure 4. Original undistorted images in the first column, the b/L channel thresholding in the second column, the Sobel gradient filter in the third column, and the two filters combined in the last column.

To fix this perspective distortion of the image, it was transformed to have a perspective of the image from above, also known as birds-eye view. After that a binary thresholding was applied to the perspective transformed lane line segment. The next step was to find a good starting point to look for pixels belonging to the left lane line and pixels belonging to the right lane line. One approach is to generate a histogram of the lane line pixels in the image. The histogram should have two peaks each representing one of the lane lines. The image below shows two example histograms generated from two binary images:

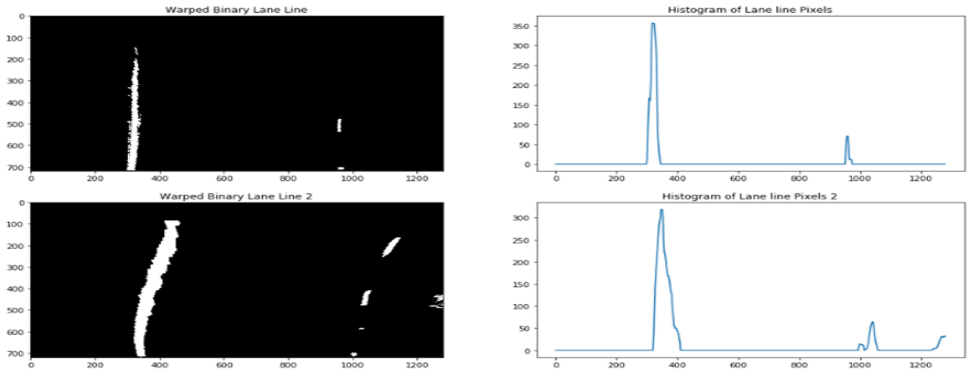


Figure 7. Binary thresholded images and histogram of pixels in thresholded images.

The locations of the two peaks are then used as a starting point. A sliding window search technique was used: it starts from the bottom and iteratively scans all the way to the top of the image, adding detected pixels to a list. If a sufficient number of pixels is detected in a window, the next window will be centered around their mean position, in that way is followed the path of the pixels throughout the image. After that is fit a polynomial through the points, generating a smooth line. The image below shows the sliding window technique in action, with the polynomial fit through the detected lane pixels:

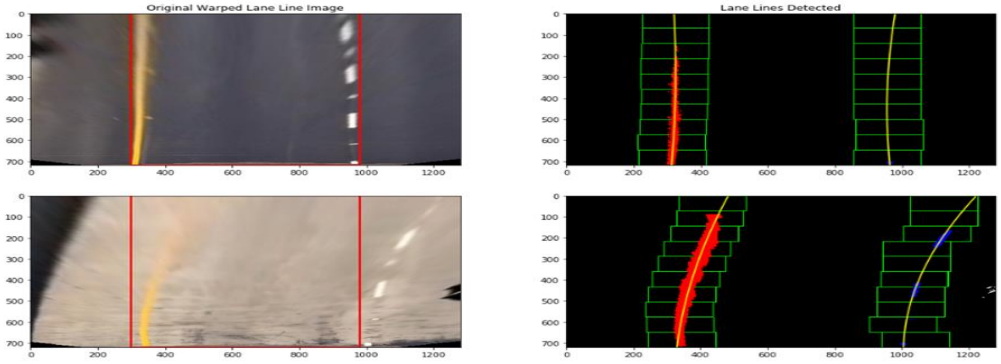


Figure 8. Warped images and polynomial fit produced using sliding window technique.

Using the location of the two detected lane lines and the assumption that the camera is located at the center of the image, the position of the car relative to the lane was calculated. Last step is to project a filled polygon onto the image where lane line boundaries are.

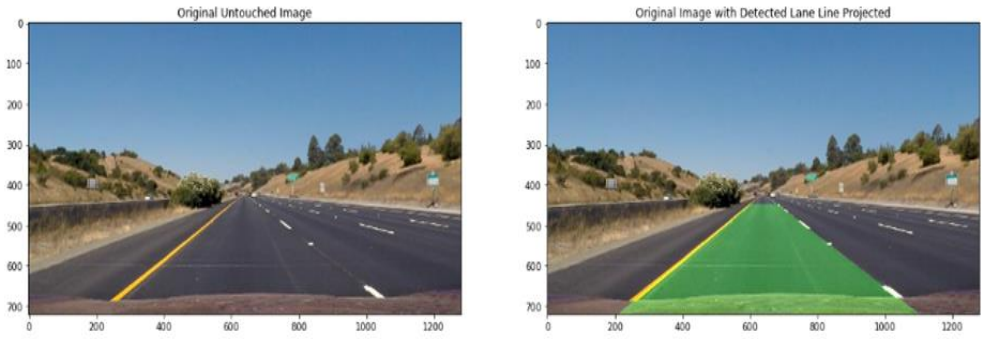


Figure 11. Projection of our lane line onto the original video.

The fifth research paper [5] presents lane and front collision detection using a mask of image, the vanishing point and the center-line estimate. On the images of roads, the lane lines appear to intersect at the horizon. This is called the vanishing point. In the top view, the points near the vanishing point or the horizon are further apart than they are in the front view. Hough Transform is used to identify lines and shapes in images and to turn the edge points into lines. Using subsequently the vanishing point, a set of source points was created and mapped (corners of the red polygon) to the destination points (corners of the top-view image).

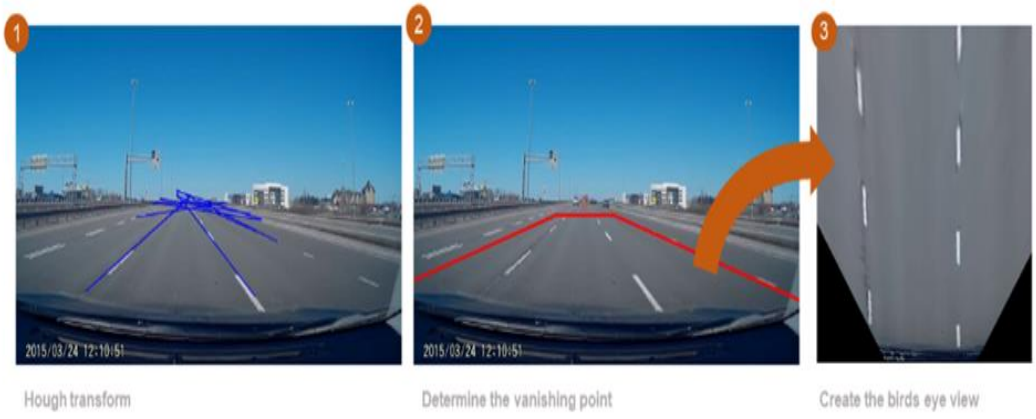


Fig 3 Creating a birds-eye view

Because HLS color space is more manageable, HLS converted image is used as the starting point for extracting the image mask. After that is applied a low/upper threshold for the white and

yellow mask, which is updated every few seconds because the background and lighting conditions are changing.

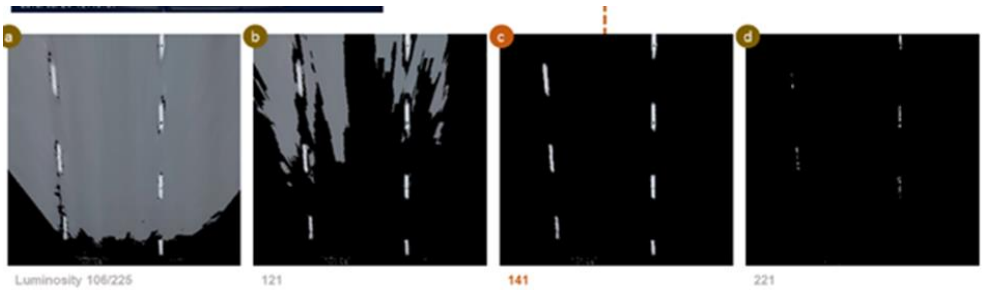
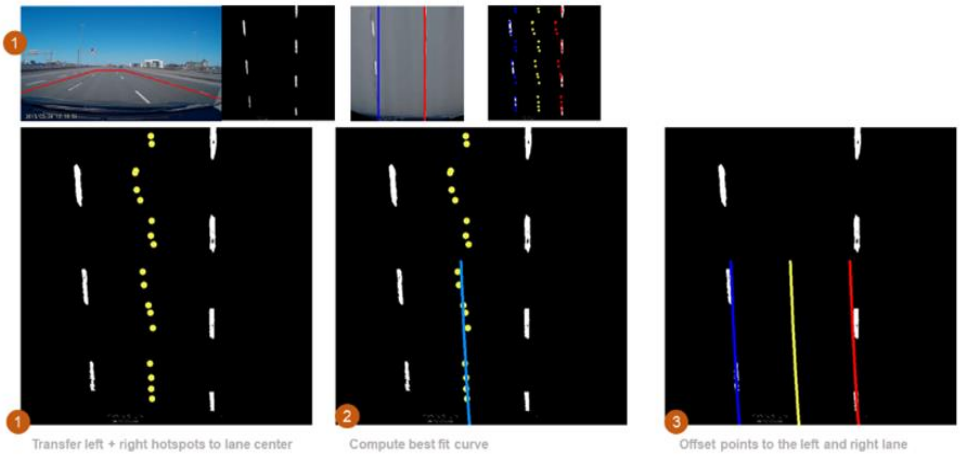


Fig 4 Using Thresholds to create a mask

The start of the left and right lanes is determined using histograms of the masked images. Sometimes the lane might curve left/right so it's more careful to use lower portion near the car to compute the histogram. On most roads, the lanes are of a standard width of 3.5 to 3.75 m. This information helps to map the pixel coordinate system in the top view image to the real-world coordinate system the image represents. After that the pixels containing the lane lines are extracted using sliding window mechanism: every pixel has a boundary box and is determined the midpoint of these boundary boxes. Next step is to fit a curve to the lane coordinates. One way to moderate this is to use the centroid of the points in each window and use the centroids to estimate the curve (Figure 8).



Both the left and right lane lines can estimate two separate curves. However, using the center-line estimate is better: it pools information from two lanes this helps produce an estimate even if one of the lane-line is missing or erroneous; the lane-lines are parallel treating, but as two separate entities lose this information.

3. Proposed solution

The proposed solution is based on some steps from [5] and [6]. The solution which successfully realizes the proposed objectives has three main steps: image preprocessing, edge detection and lane line detection. This solution uses a lot of functions from Open Source Computer Vision Library (OpenCV). In the following, these steps will be described in details.

First step from *image preprocessing* step, is to adjust the contrast of the original image. This is done by convertScaleAbs function from OpenCV, which has a parameter for brightness control, this parameter was not changed and another parameter for contrast control. After this step we get a better image, on which the road and the lane lines are better separated. Next step is to convert the BGR image into HSV image, because I observed that the algorithm gives a better solution if I use an image represented in HSV. After that we need to isolate the value (V) channel because it best enhances the lane lines.

The next step is *edge detection*, which is done by Canny, which is a multi-stage algorithm. First step in this algorithm is noise reduction, which is done by Gaussian filter. Edge detection is susceptible to noise in the image, so this step is very important. The image is then filtered with a Sobel kernel in both horizontal and vertical direction, from these images, we can find edge gradient and direction for each pixel.

The Sobel function calculates image derivatives using an extended Sobel operator. The Sobel operator [9] is used in edge detection algorithms where it creates an image emphasizing edges. The Sobel operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high-frequency variations in the image. Sobel is a type of gradient filter that uses Gaussian smoothing and differentiation operations to reduce the effects of noise.

After using the Sobel operator and non-maxima suppression, to remove any unwanted pixels which may not constitute the edge, hysteresis thresholding is applied on the image. For

this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. After this algorithm we get only strong edges in the image. These steps for edge detection are done by Canny function from OpenCV, which has 3 parameters: first parameter is the input image, the second and the third parameters are the min and max threshold values, which are used for hysteresis thresholding.

We can get an improvement on the resulting image from Canny algorithm if we use Hough Line Transform. The Hough Line Transform is a transform used to detect straight lines. The purpose of the technique is to find imperfect instances of objects, in our case lane lines, within a certain class of shapes by a voting procedure. In this project was used the Probabilistic Hough Line Transform, which is a more efficient implementation of the Hough Line Transform. The OpenCV function has a lot of parameters, like: minimum line length, maximum allowed gap between points on the same line, a threshold for votes, distance and angle resolution, etc.

At this moment, on the image white pixels represent parts of the lane line and black pixels the background. For *lane line detection* I used some functions for having a great final result on the image. First step is to pick the area in the image on which the lane line detection will be applied. For this, we need to pick four points in a trapezoidal shape that would represent a rectangle when looking down on the road from above. This can be done programmatically and manually by playing with the dimension of the trapezoid. The points need to be sorted clockwise, starting from top-left. This trapezoid will represent the ROI (Region of Interest) of the lane line detection algorithm. Another important step, is to connect the interrupted lines from road, to have a solid line on each side of the lane. For this is used the slope-intercept equation from two points, which approximates a line between these points. After we have a solid line on the right and left side, we can use a function which colors the area between these two lines in the region of interest of the resulting image.

4. Implementation

The most important function from this project is called processingPipeline which has a single parameter: the source image which will be processed. First step is contrast adjustment on the source image, which is done by convertScaleAbs function from OpenCV. This has a parameter for brightness which was not changed (default value: 0) and a parameter for contrast which is increased a little bit for a better separation of the dark colors from the light colors.

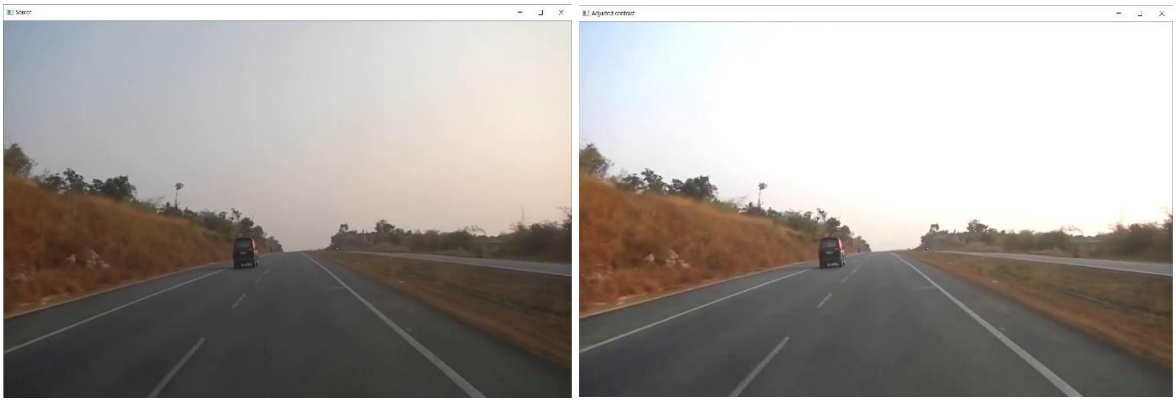


Figure 4.1 Difference between source image and contrast adjusted image

After that, this image is transformed in HSV color space using cvtColor function and then is split in 3 different images: hue, saturation and value. The value component is the most important because this is used for Canny algorithm.

For the final Canny algorithm, I set the lower threshold parameter for 70 and the higher threshold for 100. Initially I tried other threshold values, but this combination had the best result. The source image is a little bit dark and because of this the threshold values represents darker gray colors.

The region of interest on the image is determined by an array of four points, which represents a trapezoid, because the lane in front of the car is wider and farther becomes slimmer, therefore a trapezoid form is an ideal choice. The get_roi method cuts of from the image what is outside of the region of interest.

Figure 4.2 Result of the Canny algorithm

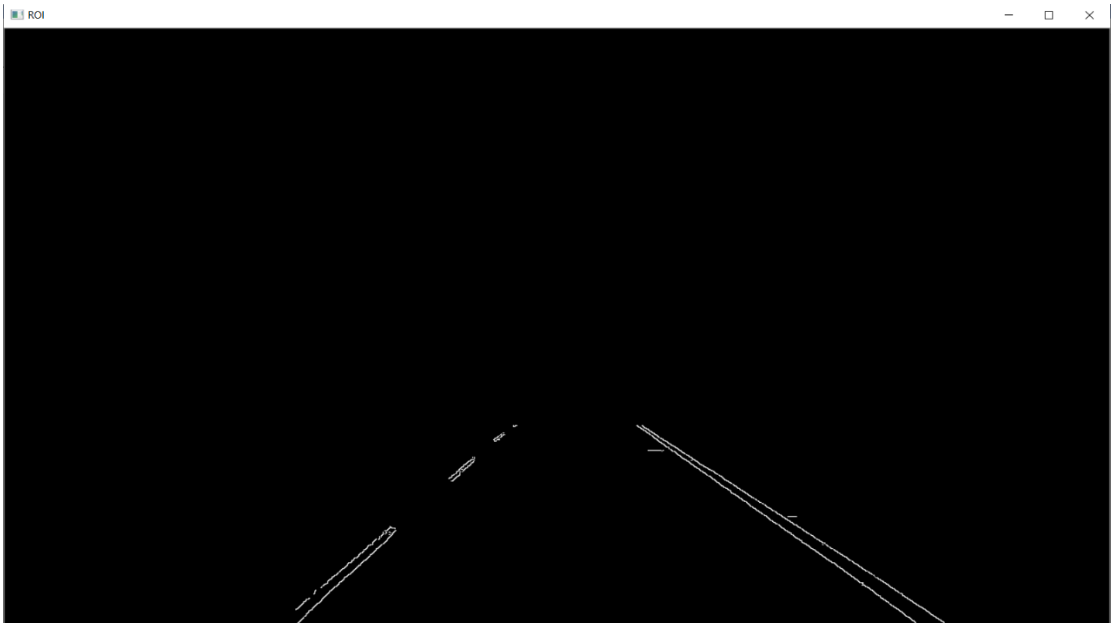


Figure 4.3 Result after get_roi method is applied

After we have the ROI in the image, we apply the HoughLinesP function from OpenCV, which will generate an array of lines. These lines, together with the source image are used as parameters for draw_lines function, which draws the lines on the original image.

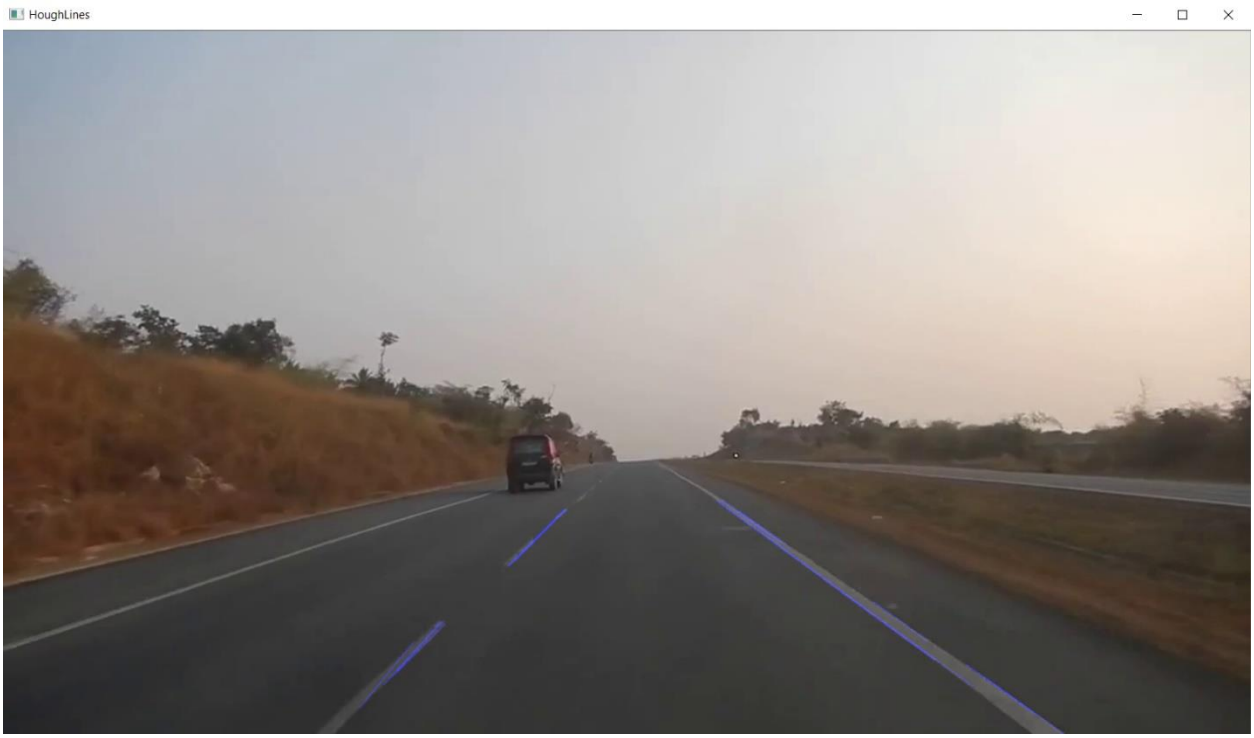


Figure 4.4 Lines detected by Canny and Hough Lines algorithms

After we have these lines, next step is to connect interrupted lines to have one single solid line on the left and right side. For this, we call getLeftRoadLine and getRightRoadLine functions, which compute one line from these multiple lines on each side of the road. The final step is to color the area between the lines with a selected color.

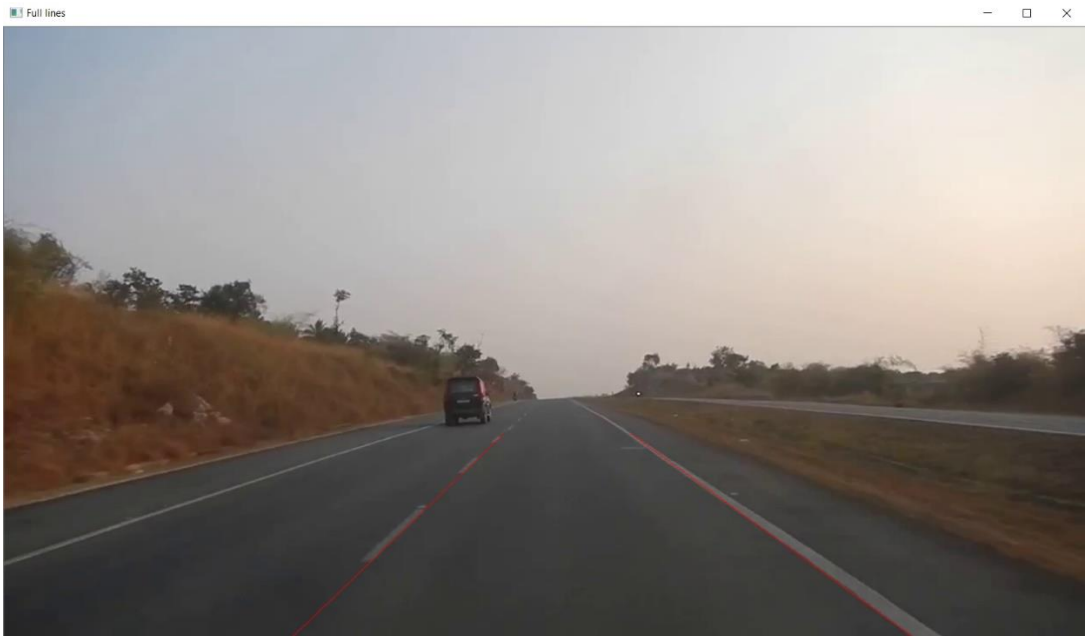


Figure 4.5 The result after getLeftRoadLine and getRightRoadLine are called

5. Final results

To test the lane line detection algorithm, we have two functions: photoTest and videoTest. PhotoTest is used for image inputs and videoTest for video inputs, inputs with more frames. Both functions are based on processingPipeline function, which processes the input image.

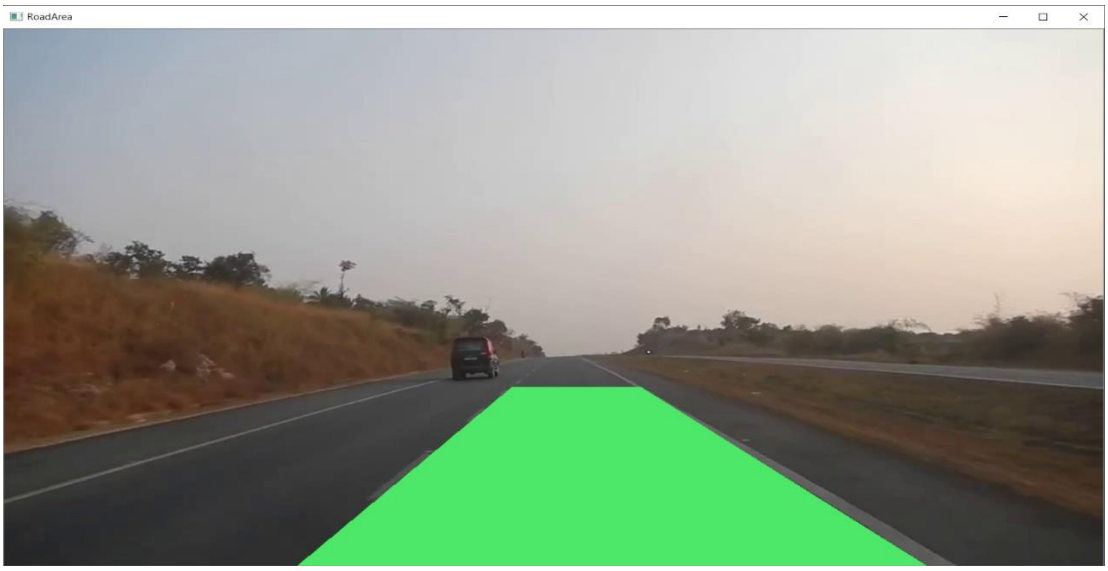


Figure 4.6 Test image 1

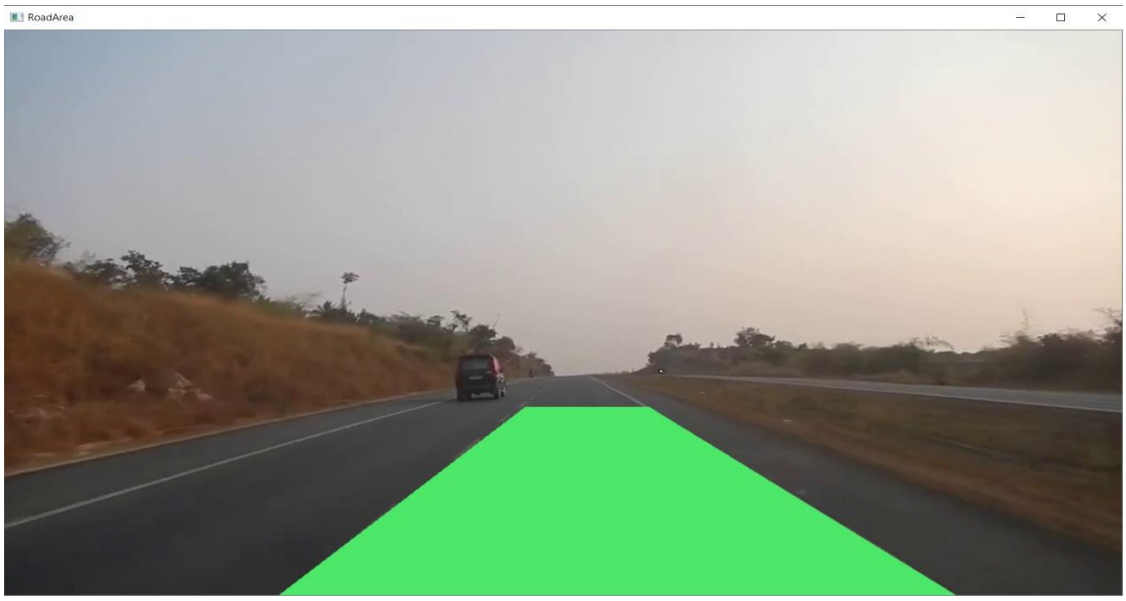


Figure 4.7 Test image 2

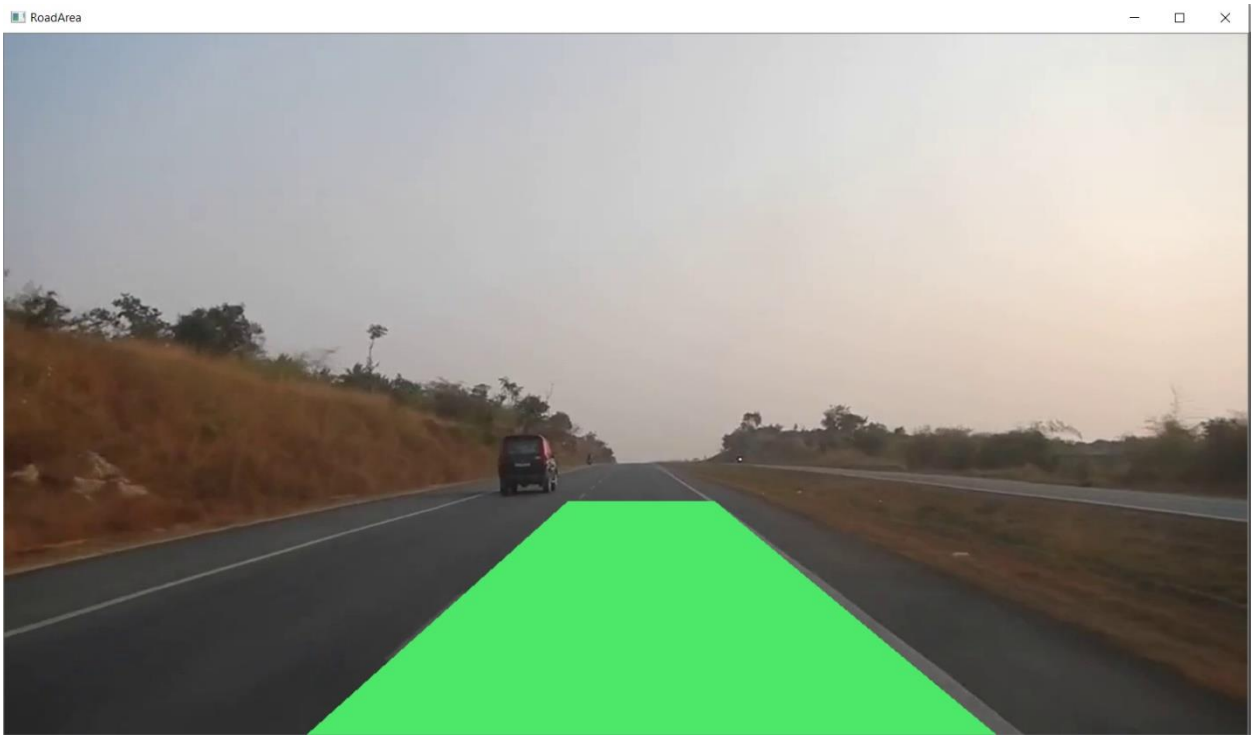


Figure 4.8 Test image 3

As we can see, the above images are some frames from a video on which was applied the lane line detection algorithm. The algorithm doesn't work perfectly, there are some problems on the left and right side of the colored area. In some cases, the lines are in this colored region, in other cases the lines are outside of the colored area. This situation is because the slope-intercept equation from two points doesn't approximate perfectly the line or these two points are not selected properly. However, this algorithm works pretty good in most cases and has an adequate performance.

6. Conclusion

In conclusion, I can say that this project is very good start for a complex, effective lane line detection algorithm with high performance which can be built in a car. A well-designed, cost-effective lane keeping system is key to the development of autonomous delivery system.

This algorithm can be improved with a lot of techniques, for example: sliding window for line detection, polyfit for a more precise line, radius of curvature for curves and some other color spaces, color filters for a better separation of lines from the road. Future development will simulate more complicated road conditions: tunnel, rugged ground, different light conditions. Another improvement would be to use deep learning for lane line detection or to extend the project to detect other cars or traffic signs too.

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", April 2019
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6479783/

[2] Muhammad Shafique, Muhammad Fahim, Prashant Pydipogu, "Robust lane detection and object tracking", School of Engineering, Blekinge Institute of Technology, June 2013
https://www.diva-portal.org/smash/get/diva2:829209/FULLTEXT01.pdf

[3] Takialddin Al Smadi, "Real-Time Lane Detection for Driver Assistance System", Department of Communications and Electronics Engineering, College of Engineering, Jerash University, July 2014

https://www.researchgate.net/publication/273985782_Real-Time_Lane_Detection_for_Driver_Assistance_System

[4] Moataz Elmasry, "Computer Vision for Lane Finding", April 2018

https://towardsdatascience.com/computer-vision-for-lane-finding-24ea77f25209

[5] Shuvashish Chatterjee, "Copilot — lane and front collision detection", August 2019

https://towardsdatascience.com/copilot-driving-assistance-635e1a50f14

[6] https://chatbotslife.com/self-driving-cars-advanced-computer-vision-with-opencv-finding-lane-lines-488a411b2c3d

[7]https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[8] https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

[9] https://en.wikipedia.org/wiki/Sobel_operator

[10] https://opencv-python tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html