Lane Detection System
Using Image Processing

Mohammad Mirzanejad

2016 Summer

Introduction:

Currently, there are more than 20 million cars on the roads of Iran, and this number is more than one billion in the world. Statistics make integrated security an inseparable part of the vehicles of many car manufacturers and there are many rules and standards for vehicle security.

In recent years, artificial intelligence has introduced successful examples of smart self-driving cars to the world by entering the world of vehicles. With the deployment of self-driving cars, the lives of millions of people will be saved from the danger of death, and the world's transition to clean energy will dramatically increase.

In this project, we are going to design a system that after receiving the images from the camera installed in the front of the car and processing these images, will detect the driver's sudden change of direction due to fatigue or human error and through the front signs This system can be used in the car and warn the driver. In this system, first, the camera images are sent to the computer and then they are analyzed using image processing techniques, and the vehicle's deviation from the path is detected. The images are processed in real-time and in case of a sudden change of direction, a warning command is given to the driver. Of course, the practical implementation of such a system requires access to the car's network system and control commands must be sent to the car's multiplex system to warn the driver. For this project, hardware implementation is omitted and this matter is postponed to the future.

Image processing

When we drive, with the help of our eyes, we consider the road lines as a constant reference for decisions and driving the car. This is the first thing that is implemented in the design of self-driving cars by different algorithms.

Here, we receive an image similar to the image on the right, and with image processing algorithms, we determine the path (line) we are moving on - similar to the image on the left. We used the videos that we recorded from the inside of the car while driving on the roads during the day and at night. These images have very high noise due to the high speed of the car, the inappropriate location of the camera and the light, and the real environment.

Sample output of the system for real images taken from driving in which ambient light and noise are evident; The right side of the input images and the left side of the processed images for detecting the current driving lane for the day (top) and night (bottom)

Gray spectrum and noise removal

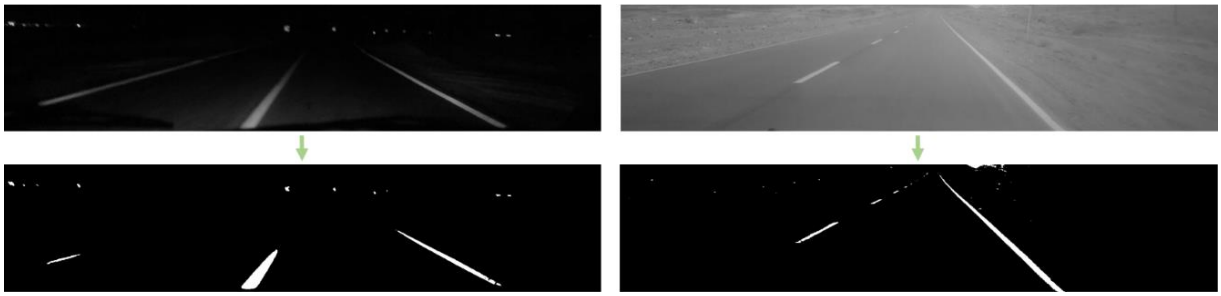
We used OpenCV computer vision framework, image processing techniques, and Python to process images. In the first step, we remove the upper and lower parts of the images that do not include important information about the road. Because most algorithms use the difference in color change (that is, the light or the impact of an object on a part of the image, and here the road markings), at this stage we convert the image to a gray spectrum; That is, we convert the value of each pixel in the three color channels of the image (red, green and .blue or RGB) to a value between 0 and 255.

Converting the input color image received from the camera to a gray spectrum .image and removing the top and bottom in the day (right) and night (left) image

Finding Delineation and Separation

Before detecting edges, we need to clearly define what we are looking for in the image. Road markings are always white or yellow. Looking further into the captured images, a simple method to isolate the white dashed lines is to extract the pixels that have a higher value than the rest of the adjacent pixels or within the image. Therefore, we select the points that have a value slightly higher than the upper quarter of the value of all pixels in the image

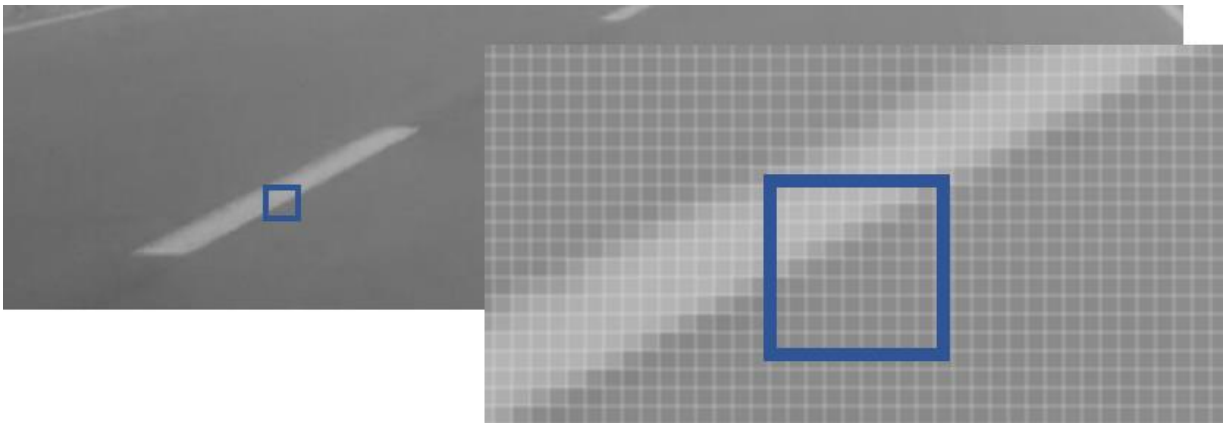
In the following, because the color yellow is not easily separated in a three-channel RGB image and these values may be removed in the value extraction mode by quartiles, we change the display format of the image to HSV and extract similar points with yellow colors of the HSV image in the gray spectrum image as the desired points (you can use Photoshop or other similar online tools to find the color spectrum). Finally, the sum of the selected white and yellow points is used for further analysis by applying a noise removal filter by averaging the neighboring pixels.



Extracting pixels needed to identify road markings in day (right) and night (left) images

Detecting the edges of lines

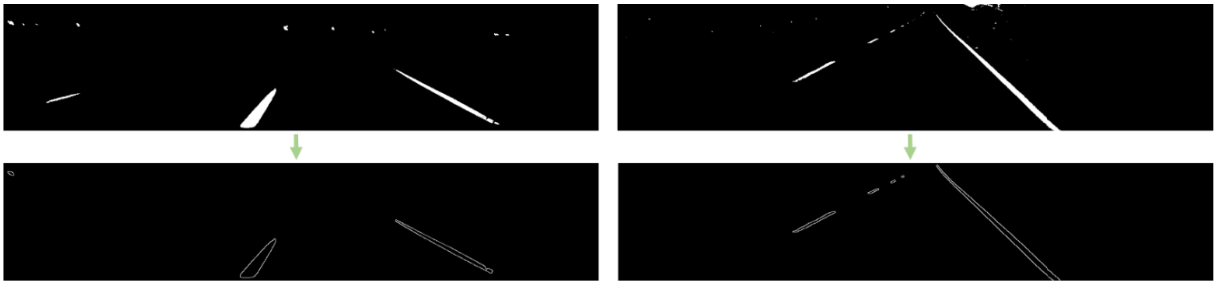
After we have preserved the main lines in the image and removed other values, we need to detect the main edges of the lines. To do this, we must first answer Taking a closer look at the data around an edge, we can conclude that edges are areas where pixel values change rapidly. Therefore, edge detection is finding pixels that have significantly different values compared to their neighbors.



An example of line detection using a gradient of changing pixel values for an edge

Fortunately, in the field of image processing, this problem has already been solved. The Canny edge detector algorithm does this in the same way by finding the gradient values at a certain threshold along the image.

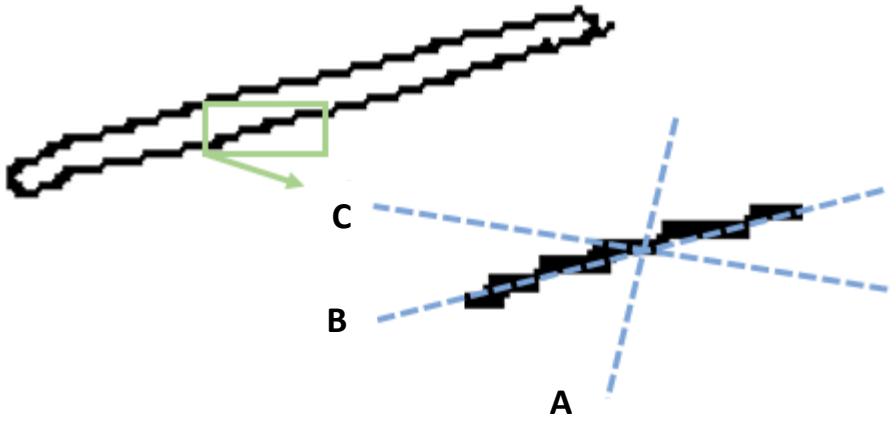
Therefore, we don't need to think about the color values of the images, only the value changes are important to us, and we can find the edges in the images of the previous step with this algorithm.



Sample output for edge detector algorithm for road markings in day (right) and night (left) images

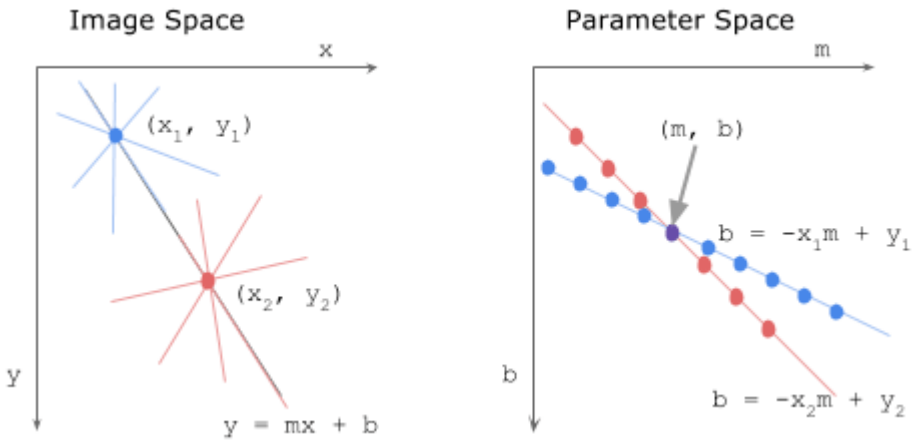
Detecting road lines

So far, we have identified the pixels that show the edges of the lines. Now, by connecting these pixels together, we need to identify the lines defining the path. Like the previous step, this problem can be solved with a mathematical theory. Let's take a look at the found edges again:



Edge found in a specified area (green); The candidate lines for the edge are marked in blue in the image

A close look at the edges, it is obvious that the best line that displays an edge is the line that includes the largest number of pixels from the edge (line b in the image above). Finding this line is a difficult problem. Because the number of these lines may be too many it is necessary to check all possible lines. For this purpose, we can use the Hough transform. In this transformation, we convert all the edge pixels into another mathematical representation. After complete conversion, each pixel in "image space" becomes a line or curve in " Hough space"; That is, in Hough space, each line is displayed as a point in the image.



A representation of the image space and the corresponding Hough space with parameters (source)

Therefore, we do not need to solve the problem of finding a line that passes through all the neighboring pixels, and it is enough to find a line that is in the Hough space and finally map this line to the image space.

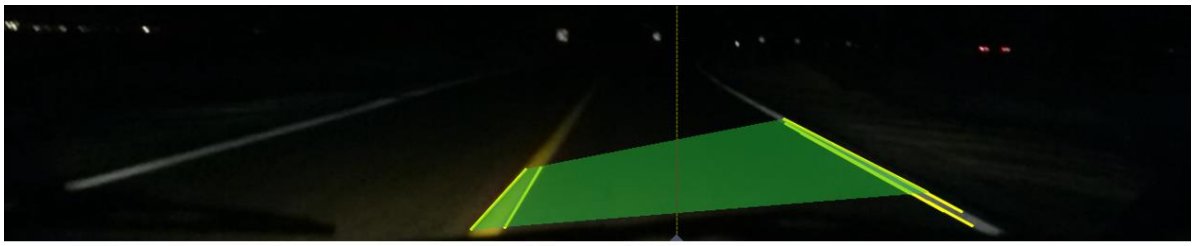
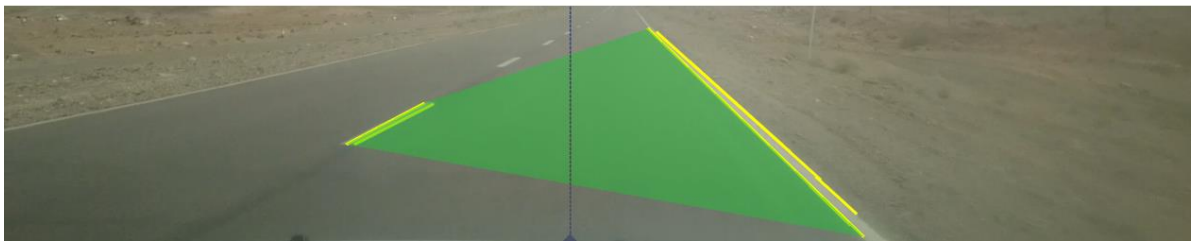

Lines found (yellow) using edge pixels after mapping from Hough space to image space and placed on the original images.

Decision-making at the final stage

The goal is to create a criterion for decision-making. If everything has been implemented correctly so far, we will have lines that are representative of the road markings for the route we are traveling through. First, we divide these lines into two groups right and left lines. An obvious difference between the two groups of lines is the direction of the slope of these lines. The slope of each line measures the angle. Horizontal lines have zero slope and vertical lines have infinite slope. So, the slope of the found lines has a size between these two values, because they have an angle from the bottom of the image toward the center, concerning the horizon.

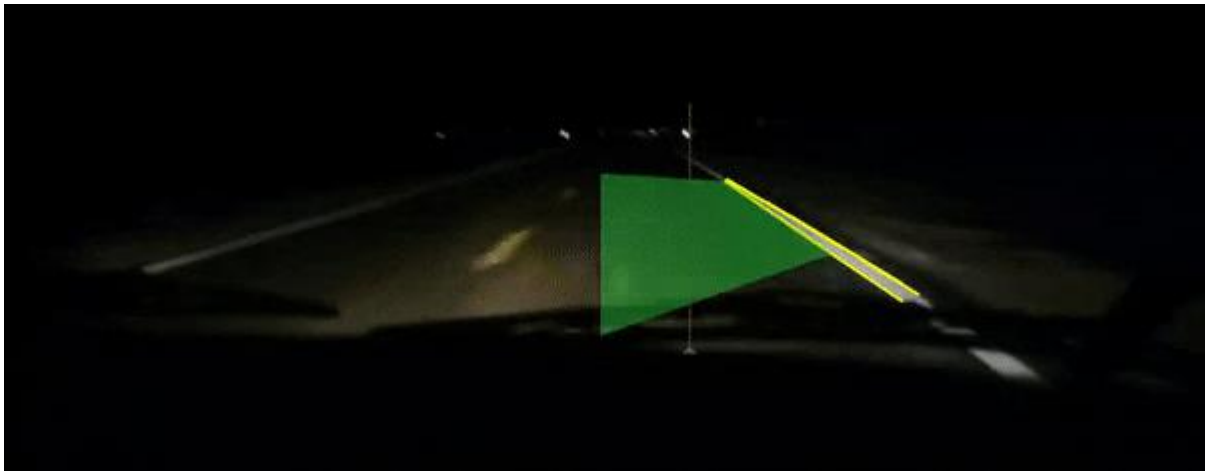
The direction of the slope of a line shows that by moving from left to right, is the line inclined up or down? Lines with a negative slope move downwards and lines with a positive slope move upwards. This is what happens in the normal coordinate system. In the coordinate system we use, the center point is in the upper left corner of the image; That is, here the direction of our slope will be reversed.

So the lines on the right have a positive slope and move down. Observations show that lines with very low slopes are probably not good lines for finding the decision line. Therefore, we try to choose the lines that have a slope of more/less than (+/-) 0.4.

The final result of the image processing to find the decision-making line of vehicle deviation from the road (dashed line) at night (top) and day (low)

The decision line, which represents the deviation of the vehicle from the moving path, is the average of the two groups. For this line, the average value of the highest and lowest x of each group will determine the value of x. The last step here is to receive images from the camera. It is enough to receive each frame of the video separately and repeat all the above steps for it. Of course, in the case of video, it is possible to follow different objects more carefully or to consider each frame in the time frame compared to the other frames, which we will not discuss here due to the length of the article. If you are interested in receiving these images through the network between mobile and laptop, you can also check these codes.



Driver lane change detection system using computer vision for realistic images of driving at night – Note: This is a gif file and is supposed to show the final output of the system, if you see it as a non-animated picture, please find the real video in the resources folder with the name of “final output video.gif”.

We made it! Now we have a system that can recognize the path of the car. It is enough to consider the location of the camera in front of the car if the deviation of this line is greater than the calibrated amount; It means that the car changed direction suddenly, and warning commands should be given to the driver.