Detail SW Design Specification document

Capstone Design (soc4150-002)

Team: tripLM (group16-3)

ID: U1610146 Mirzashomol Karshiev
ID: U1610143 Mirpulatjon Shukurov
ID: U1610137 Mardon Zarifjonov
ID: U1610125 Lazizbek Qahhworov

Table of Contents:

1. Introduction
2. Use Case Diagram
3. Explanation

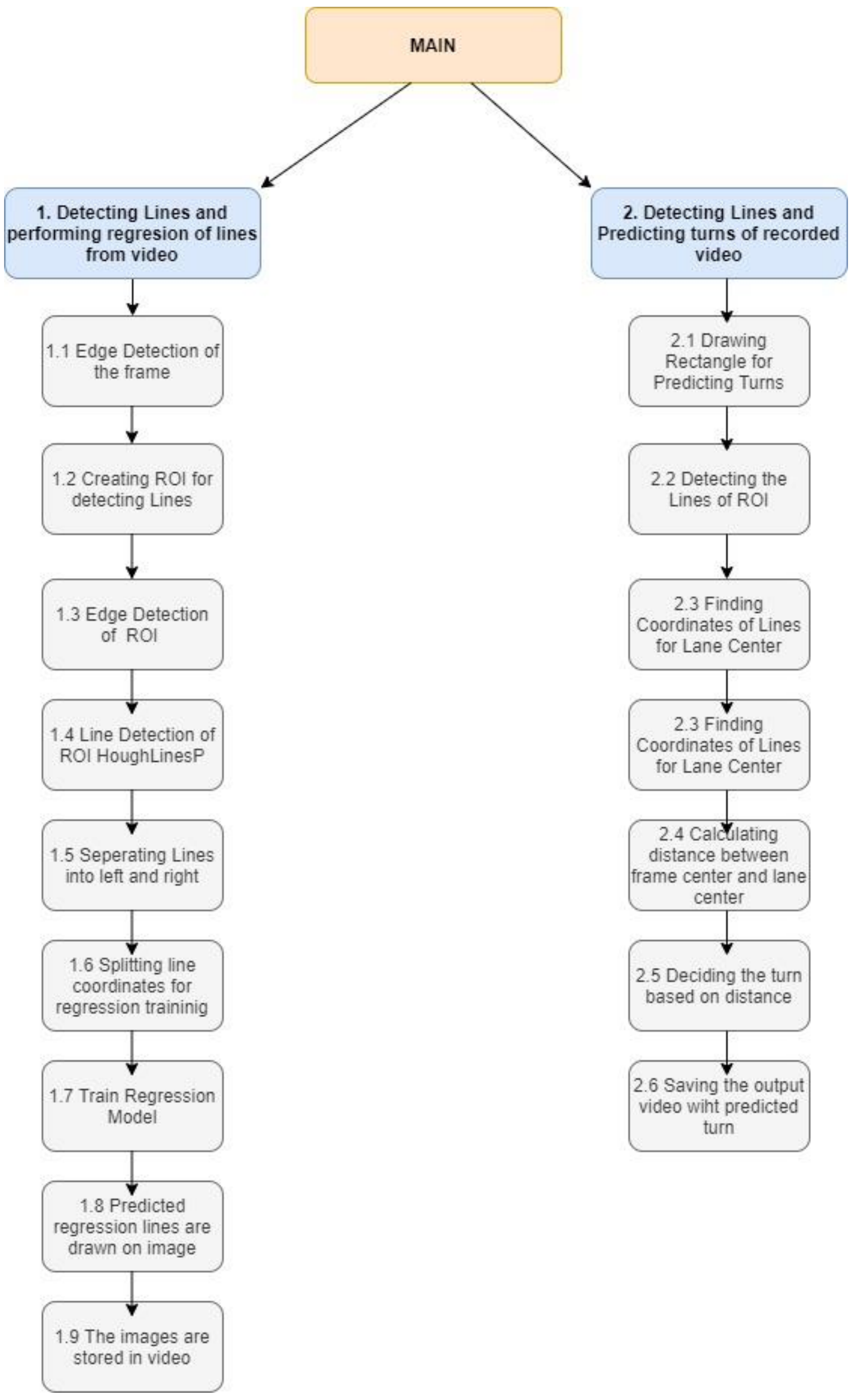
1. Introduction:

This SW design specification of a program that is written in Python language to detect the lines from the video and based on those lines predict the turn. These tasks are achieved with the help of TensorFlow and OpenCV libraries. The purpose of this project is to gain knowledge that can be applied in future studies that are related to self-driving cars or projects that are related to Image Processing and Deep Learning. Moreover, this project gives an understanding of fundamental procedures that are performed using OpenCV when working with images and a basic understanding of linear regression models. The knowledge obtained from this software design specification can be applied in self-driving cars or robots.

2. Use Case Diagram:

This is diagram of our software. That consists of 2 main parts.

1. Detecting Lines from the video and performing regression of those lines and saving the video.
2. Detecting the lines from saved video and predicting the turns based on those lines.



MAIN
1. Detecting Lines and performing regresion of lines from video
1.1 Edge Detection of the frame
1.2 Creating ROI for detecting Lines
1.3 Edge Detection of ROI
1.4 Line Detection of ROI HoughLinesP
1.5 Seperating Lines into left and right
1.6 Splitting line coordinates for regression traininig
1.7 Train Regression Model
1.8 Predicted regression lines are drawn on image
1.9 The images are stored in video
2. Detecting Lines and Predicting turns of recorded video
2.1 Drawing Rectangle for Predicting Turns
2.2 Detecting the Lines of ROI
2.3 Finding Coordinates of Lines for Lane Center
2.3 Finding Coordinates of Lines for Lane Center
2.4 Calculating distance between frame center and lane center
2.5 Deciding the turn based on distance
2.6 Saving the output video wiht predicted turn

3. Explanation:

The above diagram shows the flow design of our software.
The design consists of 2 main parts:

1. Detecting Lines from the video and performing regression of those lines and saving the video
2. Detecting the lines from saved video and predicting the turns based on those lines

1.1 Edge detection of the frame. First frame converted to grayscale then it blurred. After which it converted to binary image. Then it's edge detected using cv2.filter2D() function

1.2 Creating ROI for detecting lines. We take 4 points to create polygon which later is combined with dark image and masked with real image.

1.3 Edge detection of ROI. It is done on ROI which is binary format. Using the same filter2D, edge detected.

1.4 Lines detection of ROI by using HoughLinesP() algorithm which detects lines by going through the image.

1.5 Seperating lines into left and right based on the coordinates of lines if the coordinates are righter than image center that means it is right line if the coordinates are left side than image center it is left line.

1.6 Splitting line coordinates for training regression line. We use linear multi-variable regression for stabilizing lines.

1.7 The model is trained with learning rate 0.000001 and returns predicted vales for corresponding x coordinates.

1.8 Predicted lines are drawn on the original image.

1.9 The frames are saved in the video.

2.1 We draw blue rectangle for predicting the turn of the car. Based on the coordinates

2.2 It detects the Lines of ROI by using HoughLinesP()

2.3 Finds the coordinates of Lines inside the ROI. The lane center is found by calculating (x1+x2)/2

2.4 Calculating the distance between frame center and lane center.

2.5 Based on distance value we decide the turn whether it is slightly left or left.

2.6 Saving the output images into video.