INHA University in Tashkent

Capstone Design

SOC4150

Software Design Specification

Team Name: Senior Challengers

Team Members:

Mirkamol Khamidov (leader)	U1610142
Doston Mardiev	U1610063
Kuvonchbek Bobokulov	U1610122
Aziza Abdurakhmonova	U1610038

Table of Contents

1. Introduction

This document explains the project's SW design requirements by showing a system's functionality and how that functionality gets converted into a physical implementation, while satisfying a constrained design matrix. The purpose of the document is delivering the design decisions and considerations taken while completing a comprehensive assignment on analyzing a movement of the car based on a target video.

2. Document Outline

The planned perusers of this report will be the teacher, however the particular language utilized in this record shows that it will be appreciated and seen well by the crowd with understanding of software development, with a primary focuson Python programming language. This infers likewise that the linguistic structure of the previously mentioned language ought to be natural to the perusers, as certain parts of the source code has been included for perception of solid plan details.

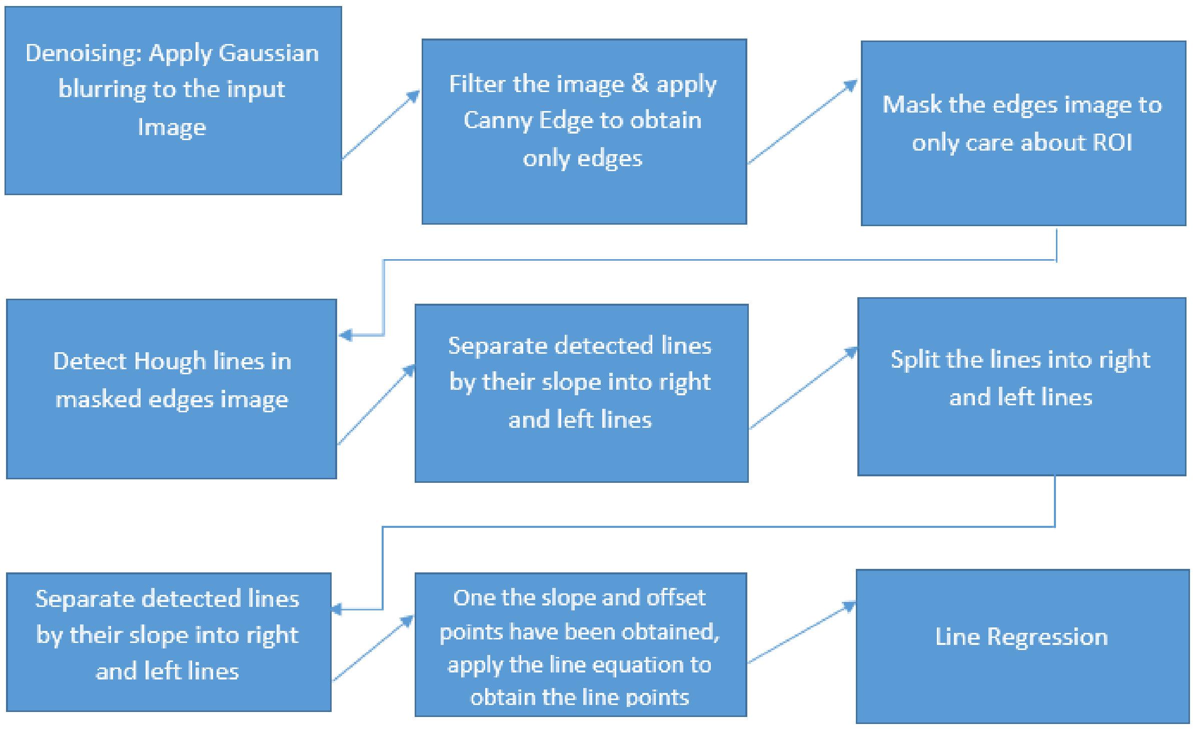
3. Document Description

3.1 Introduction

The SW specification design document provides the description of the overall framework, including the outline of features and the suggested solution's limitations / constraints. Mostly this document serves as a guidebook that should serve as a guidebook that keeps developers and designers within the frame of criteria required. Apart from that, it can be used as a manual for the new members of the team work so as they could easily get familiarize themselves with the design and development specifications.

3.2 System Overview

The proposed system's design involves an number of image processing manipulations for images and video data. The solution proposed should interpret the video material and based on this, some research should be performed on digitized image frames and manipulated to extract the meaning. The algorithm input is an image taken from a pre-recorded video. The video stream is divided into frames and then there is frame-by-frame processing.

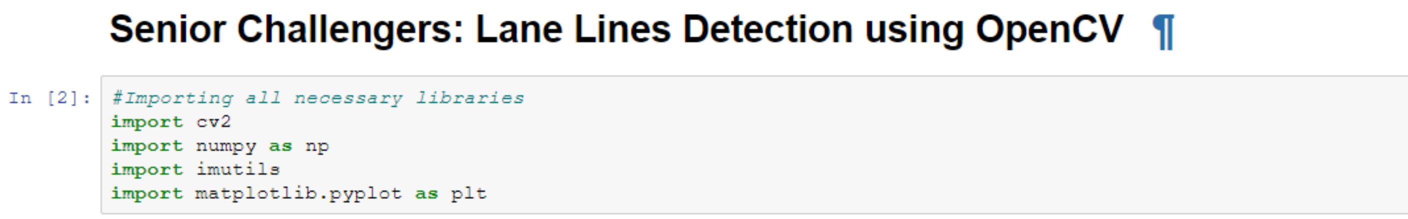

Denoising: Apply Gaussian blurring to the input Image
Filter the image & apply Canny Edge to obtain only edges
Mask the edges image to only care about ROI
Detect Hough lines in masked edges image
Separate detected lines by their slope into right and left lines
Split the lines into right and left lines
Separate detected lines by their slope into right and left lines
One the slope and offset points have been obtained, apply the line equation to obtain the line points
Line Regression

3.3 Design Considerations

One of the primary functionalities to be achieved throughout the project was lane detection. Lane detection is a well-researched area of computer vision with applications to autonomous vehicles and driver assistance systems. Mainly, the principle of lane detection works in two fundamental steps:

1) Data preprocessing, that includes noise filtering and etc
2) Drawing the lines upon the original image

Importing needed libraries:



Senior Challengers: Lane Lines Detection using OpenCV ¶

```
In [2]: #Importing all necessary libraries
        import cv2
        import numpy as np
        import imutils
        import matplotlib.pyplot as plt
```

- **Cv2 library:** OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

- **Numpy:** NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.
- **Imutile:** We use it A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV
- **Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy

Denoising: Gaussian blurring

The removing of noise from the image is called Image denoising.

All denoising algorithm are based on Noise Model.

Noise Model v(i) = u(i) + n(i) ; v(i): observed value, u(i): true value, n(i): noise value Types of DenoisingAlgorithms: All the denoising algorithms are achieved by averaging.

Most used Filters types:

• Spatial domain filter

• Gaussian filtering

• Anisotropic filtering (AF)

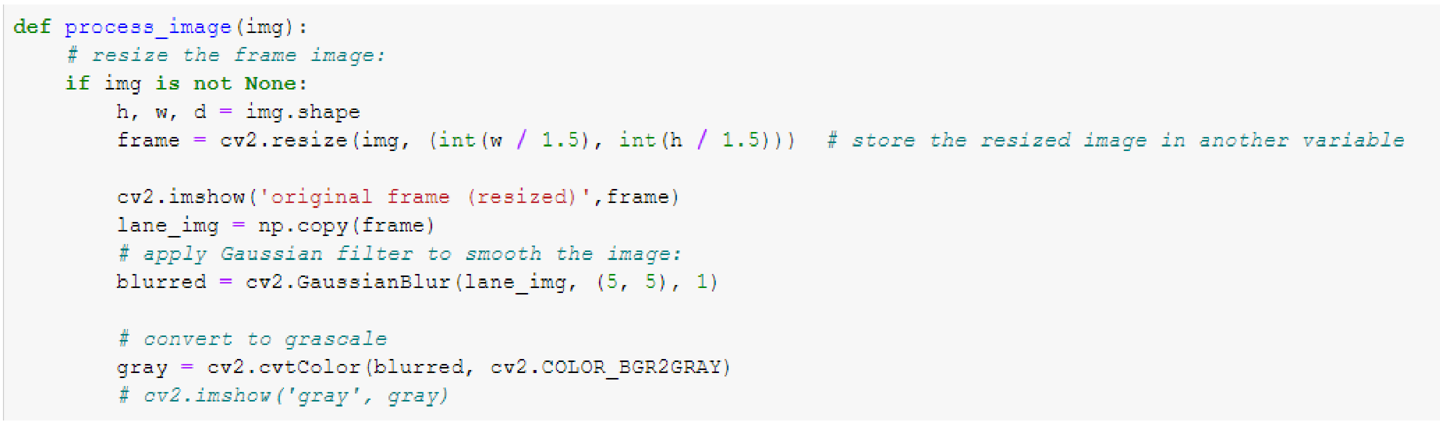
• Neighboring filtering

• Total Variation minimization

• Non-Local-Means (NL-means) algorithm

Gaussian Filtering The image isotropic linear filtering boils down to the convolution of the image by a linear symmetric gaussian kernel. The image method noise of the convolution with a gaussian kernel Gh is $u - G_h * u = -h^2 \Delta u + o(h^2)$, for h small enough.

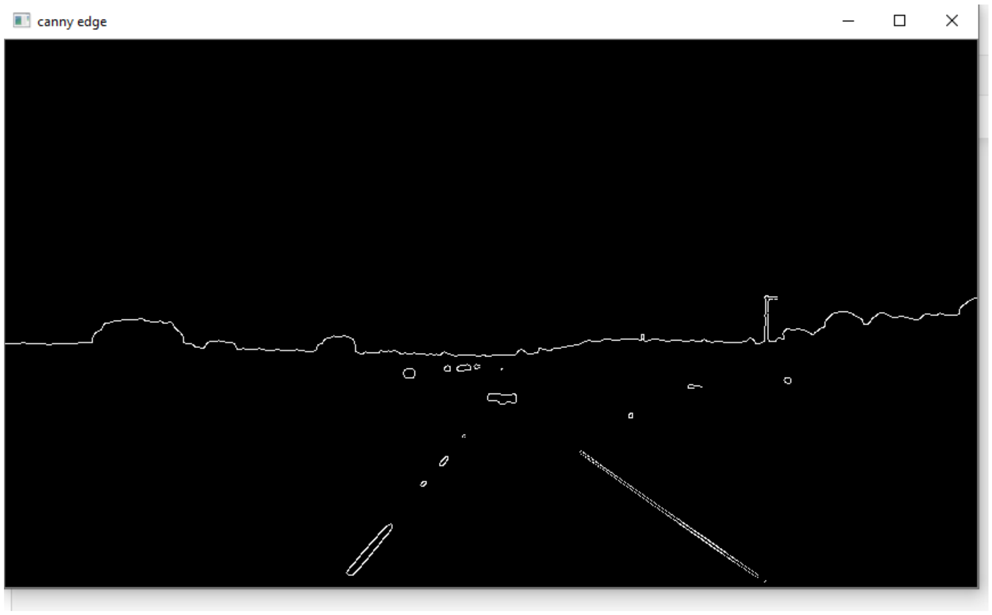
Gaussian Filtering Gaussian convolution is optimal in flat parts of the image. Drawback of Gaussian Filtering, Edges and textures are blurred.

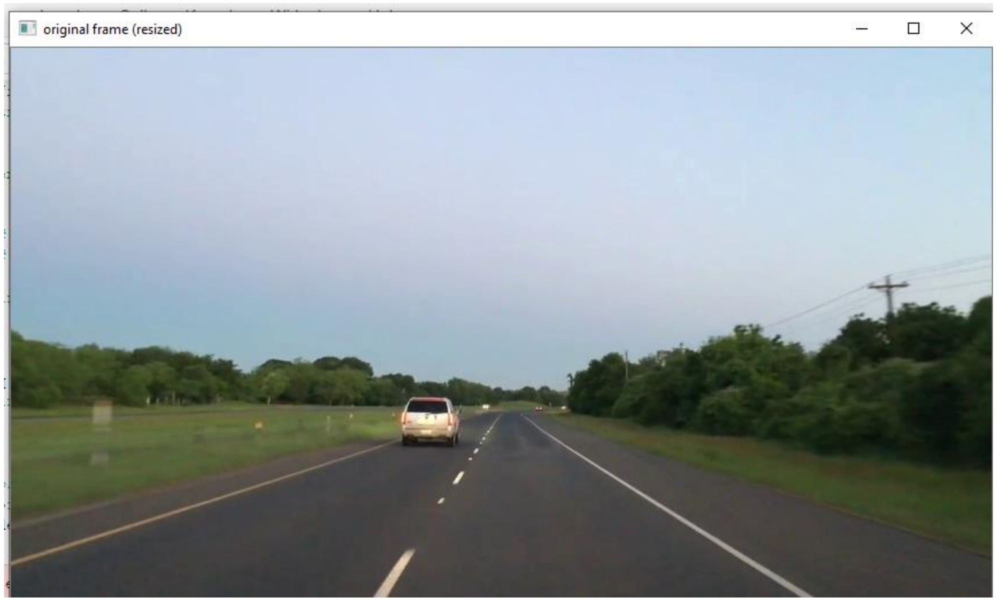


```python
def process_image(img):
    # resize the frame image:
    if img is not None:
        h, w, d = img.shape
        frame = cv2.resize(img, (int(w / 1.5), int(h / 1.5)))  # store the resized image in another variable

        cv2.imshow('original frame (resized)',frame)
        lane_img = np.copy(frame)
        # apply Gaussian filter to smooth the image:
        blurred = cv2.GaussianBlur(lane_img, (5, 5), 1)

        # convert to grascale
        gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)
        # cv2.imshow('gray', gray)
```

Canny Edge detection from binary image:

The Canny edge detector is an edge detection operator that uses a multistage algorithm to detect a wide range of edges in images





```
def auto_canny(image, sigma=0.33):
    # compute the median of the single channel pixel intensities
    v = np.median(image)
    # apply automatic Canny edge detection using the computed median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)
    # return the edged image
    return edged
```

In this case function “auto_canny” gets original image and calculates the median of the single channel pixel intensities. After that apples automatic canny edge detection using the computed median. Finally returns edged image.

Mask the image

In order to mask the image first we must identify the region that we have to mask. To do so ROI(Region of Interest) is used.

```
def ROI_masking(img, vertices):
    mask = np.zeros_like(img)
    # channel_count = img.shape[2]
    match_mask_color = (255,)   # * channel_count
    cv2.fillPoly(mask, vertices, match_mask_color)

    # we can use cv2.bitwise_and() to do masking:
    masked_img = cv2.bitwise_and(img, mask)
    return masked_img
```

```
        ROI_frame = [

            (200,500), (800,500),(450,300)

        ]

        masked_img = ROI_masking(canny, np.array([ROI_frame], np.int32))
        cv2.imshow('masked cropped edge detected', masked_img)

        masked_img = morph_closing(masked_img)
```

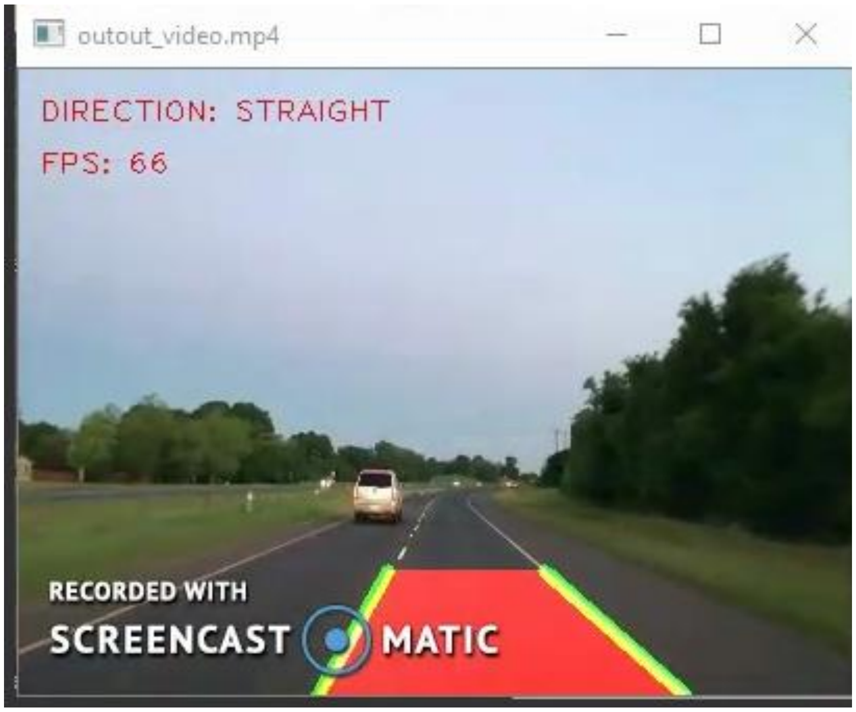
To use Hough Lines Transform, processed image should be binary.

• Lines will be expressed in terms of Polar System in Hough Lines Transform

• Firstly grayscale the image and then to detect edges. Such mask of edges can be then fetched to the Hough Lines method which should output a set of straight lines found on an image.

• y=ax+b

• Mapping from Image space to Hough space should be done. • All points in the line should be separated into left and right according to their placement



DIRECTION: STRAIGHT
FPS: 66

Final advance in our program is the representation of the lines on the casing. OpenCV gave the uncommon capacity to adhering to a meaningful boundary to which we have to give the beginning and consummation focuses just as the shading and thickness of the lines. We likewise settled on the choice to draw the polygons between the lines which gives increasingly visual portrayal of the street the vehicle is passing. We use "fillPoly" work which takes as information the focuses which limit the square shape. We return the edge with lines and square shape drawn on it.

For detecting the edges in the image, ***Canny Edge Detection*** technique is used. The Canny algorithm is one of the most efficient and effective methods for detecting the edges in an image. In fact, the principle of its work lies on the main characteristic of the edges, namely on their features. Edges are portions of an image that create sharp contrast with respect to the remaining portions. By focusing on these edges, we can pull out more important information faster than if we were looking at the whole picture. Kenny's algorithm performs the extraction of the high contrasted parts of the image from the rest parts by highlighting them thoroughly. Before applying it, it is necessary to perform several preprocessing methods, such as noise reduction and others.

3.6 **Goals and Guidelines**

The project's aim is to study the target video as a detailed task, and write a program to understand the car's movement and predict the turn based on the number of techniques and algorithms used. A comprehensive project is made up of a list of assignments that are actually step-by-step requirement actions comprising an ultimate aim.

4. Glossary

Python is an interpreted , high-level , general-purpose programming language.

OpenCV a library of programming functions mainly aimed at real-time computer vision.

RGB a color model that defines a given color according to its red, green, and blue components.

Gaussian blur a general-purpose blur filter used for removing noise from the image.

Hough Transform a feature extraction technique used in image analysis, computer vision.

Canny Edge Detection a multi-step algorithm that can detect edges with noise suppressed at the same time.

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrice s.

Readability means that the code is easy to follow, logically.

Reliability is easy to test and update without easily breaking with other developers. Optimized Performance the process of making a computer system, work as effectively as possible Image Denoising is removing a noise in the image using techniques Binarization is the process of converting a pixel image to a binary image.

5. References

https://www.tutorialspoint.com/opencv/opencv_overview.htm

https://drive.google.com/file/d/1RMq9j_-mxkqPX_C4OYdoJmdfnp15ef0S/view?usp=drivesdk

https://medium.com/sanchit-gupta/lane-detection-with-turn-prediction-1773c4819541