TOSHKENT SHAHRIDAGI INHA UNIVERSITETI
INHA UNIVERSITY IN TASHKENT

SW Specific Design Document

Capstone Design (SOC4150)

Comprehensive Assignment
(Line Detection)

Team Members:

1.	Anvarjon Yusupov	U1610026
2.	Boburjon Iskandarov	U1610054
3.	Oybek Amonov*	U1610176
4.	Kamronbek Jurayev	U1610261

TOSHKENT SHAHRIDAGI INHA UNIVERSITETI
INHA UNIVERSITY IN TASHKENT

Contents

INTRODUCTION

Driver support system is one of the most important features of the modern vehicles to ensure driver safety and decrease vehicle accident on roads. Apparently, the road lane detection and turn prediction are the main challenges in this project. In order to conquer these challenges, we managed to develop very solid and secure software system. We combined powerful features of Python programming language with OpenCV computer vision library and as result acquired multifunctional code that compresences movement of the vehicle.

SYSTEM OVERVIEW

FUNCTIONAL REQUIREMENTS

Project assignment provided functional requirements in details. Thus, our team focused on accomplishing those tasks in first place. Video frames are given as our input. Those inputs are gone through several manipulations:

1)Image resizing, 2) Color correction, 3) Image Denoising, 4) Edge Detection, 5) Masking, 6) Lines detection and their separation, 5) Turn prediction

Those are main functional requirements in our term project

HARDWARE REQUIREMENTS

This final project is considered as software project. Thus, there are no hardware requirements besides functional computer. However, this project must have been software and hardware hybrid project. That did not happen due to certain events.

SOFTWARE REQUIREMENTS

The project is fully developed by using Python programming language. Although Python is considered as slow, it is still powerful language with simple syntax and extensive libraries. During this project we required access to functions of three libraries:

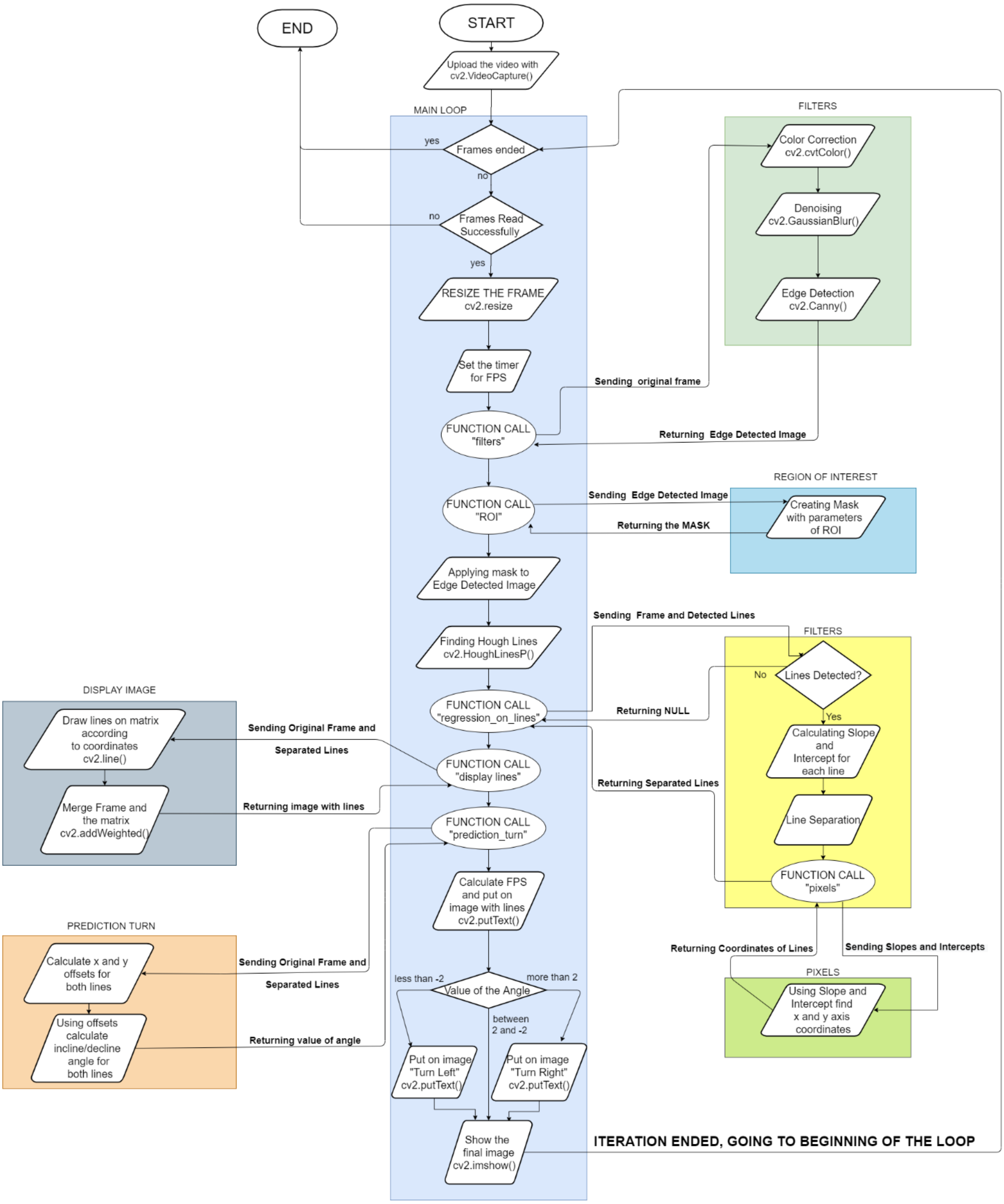
1) OpenCV library – open source computer vision library. All the actions regarding image manipulation such as image display, color correction, blurring and so on, were implemented with this particular library. So cv2 library was arguable the main software requirement.

2) NumPy - python library for numerical manipulations. This library was fundamental in matrix manipulations. We regularly used matrices to store the values of pixels in an image, as well as to find the average value of intercepts and slopes.

3) math – for mathematical manipulations. We used only one function from this library, however *math.atan()* that we used was the core operation in turn prediction phase. We used it to calculate incline/decline angle of the lines using offsets, because it requires to apply arc tangent trigonometric function.

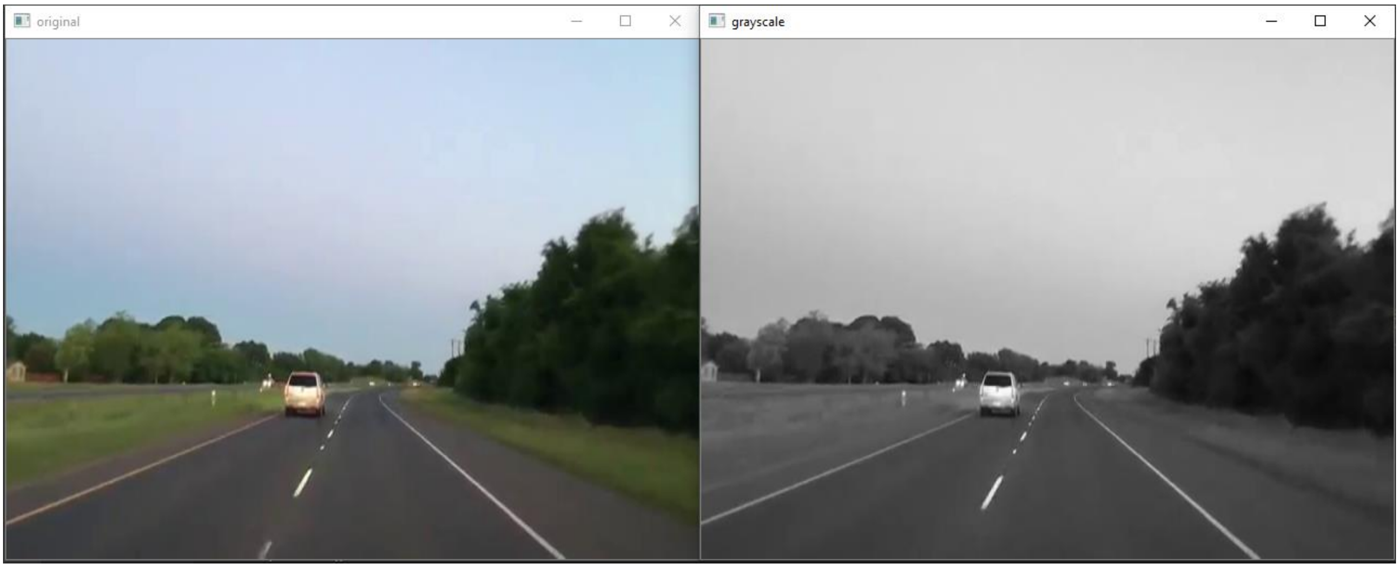
SOFTWARE DESIGN AND IMPLEMENTATION

We understand from the beginning that this project would involve multiple advanced techniques that must be precisely ordered and written. Thus, we thoughtfully structured our program in order to achieve best efficiency. The fact that all the steps were already given in task was very handy. So basically, we completed each task step by step, while trying to keep the code clean and easy to read. The flow chart of our code is given below.

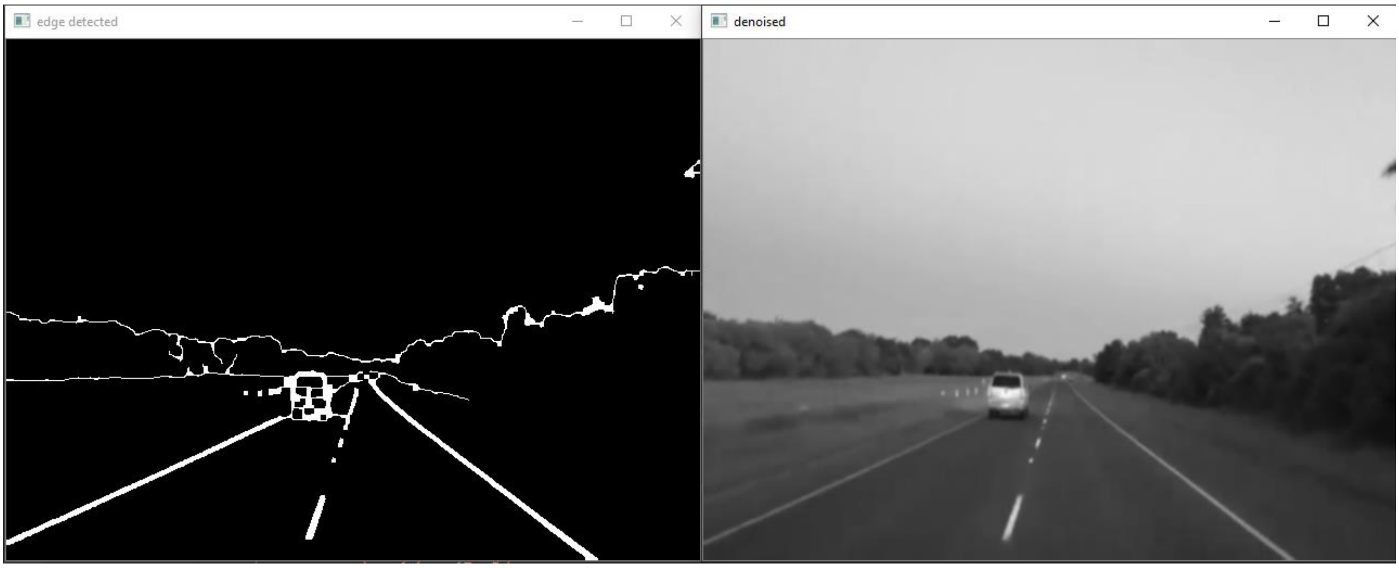

END
START
Upload the video with cv2.VideoCapture()
MAIN LOOP
yes
Frames ended
no
no
Frames Read Successfully
yes
RESIZE THE FRAME cv2.resize
Set the timer for FPS
FUNCTION CALL "filters"
FUNCTION CALL "ROI"
Applying mask to Edge Detected Image
Finding Hough Lines cv2.HoughLinesP()
FUNCTION CALL "regression_on_lines"
FUNCTION CALL "display lines"
FUNCTION CALL "prediction_turn"
Calculate FPS and put on image with lines cv2.putText()
less than -2
more than 2
Value of the Angle
between 2 and -2
Put on image "Turn Left" cv2.putText()
Put on image "Turn Right" cv2.putText()
Show the final image cv2.imshow()
FILTERS
Color Correction cv2.cvtColor()
Denoising cv2.GaussianBlur()
Edge Detection cv2.Canny()
Sending original frame
Returning Edge Detected Image
REGION OF INTEREST
Sending Edge Detected Image
Creating Mask with parameters of ROI
Returning the MASK
Sending Frame and Detected Lines
FILTERS
No
Lines Detected?
Yes
Returning NULL
Calculating Slope and Intercept for each line
Returning Separated Lines
Line Separation
FUNCTION CALL "pixels"
Returning Coordinates of Lines
Sending Slopes and Intercepts
PIXELS
Using Slope and Intercept find x and y axis coordinates
DISPLAY IMAGE
Draw lines on matrix according to coordinates cv2.line()
Merge Frame and the matrix cv2.addWeighted()
Sending Original Frame and Separated Lines
Returning image with lines
PREDICTION TURN
Calculate x and y offsets for both lines
Using offsets calculate incline/decline angle for both lines
Sending Original Frame and Separated Lines
Returning value of angle
ITERATION ENDED, GOING TO BEGINNING OF THE LOOP

Although, in the flow chart some details are skipped, it still accurately represents structure our project:

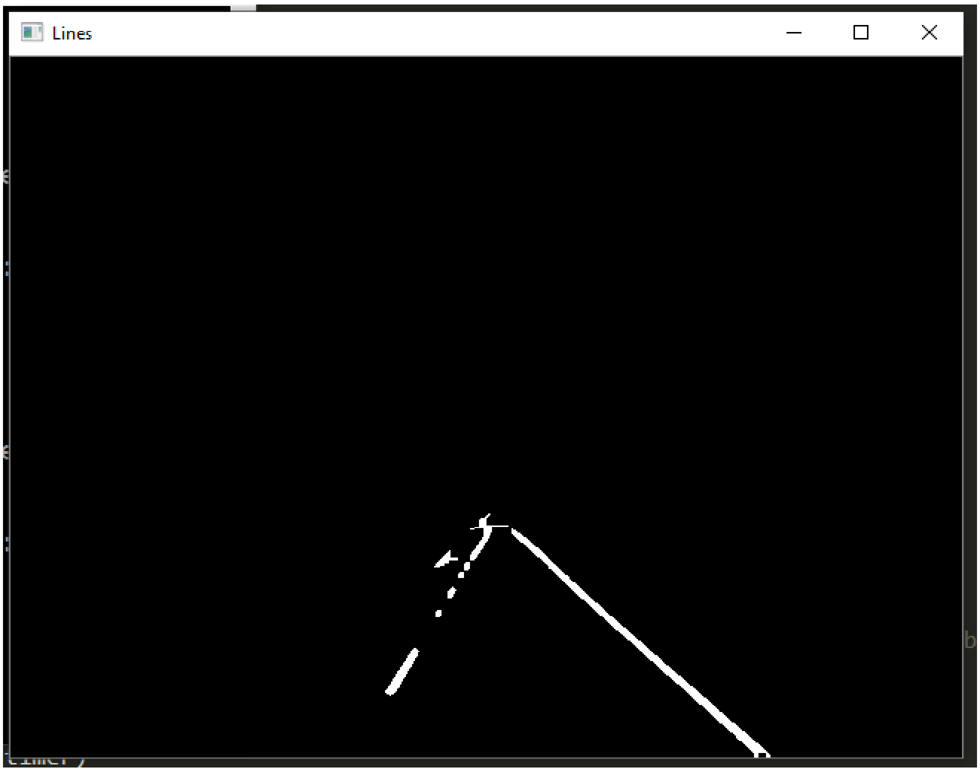
One of the first tasks we have accomplished was turning each frame from BGR imaginary to grayscale one, we did with cv2.im



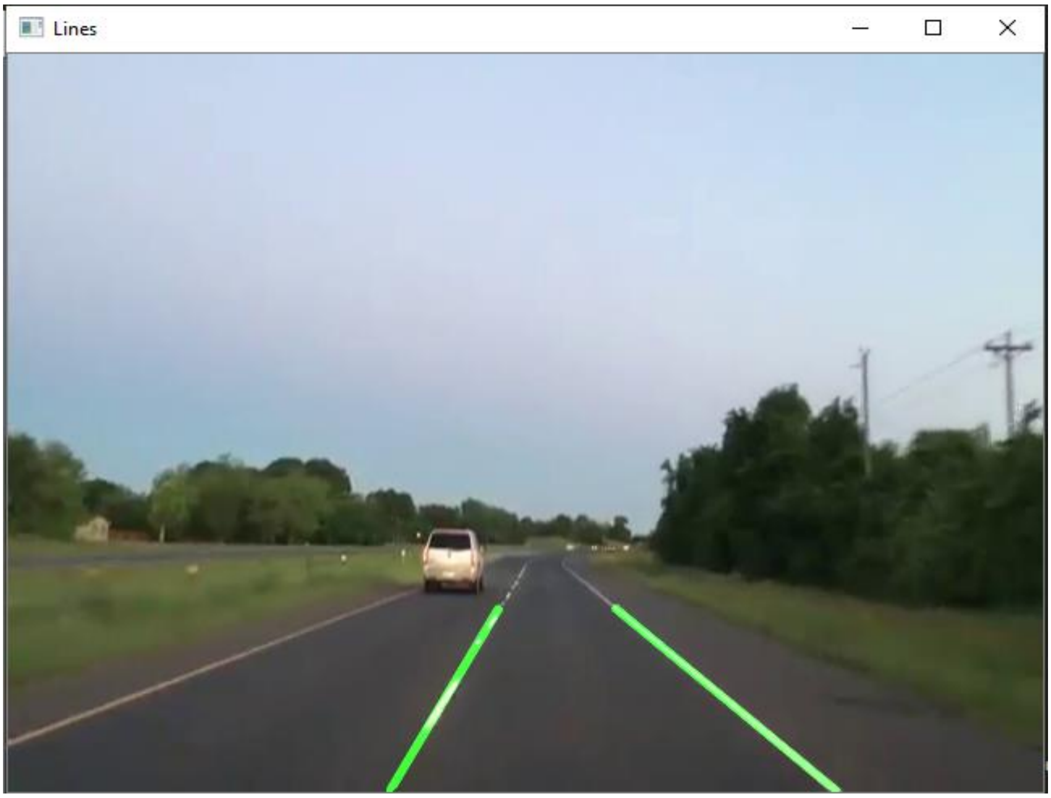
The next steps were denoising the image and finding the edges:



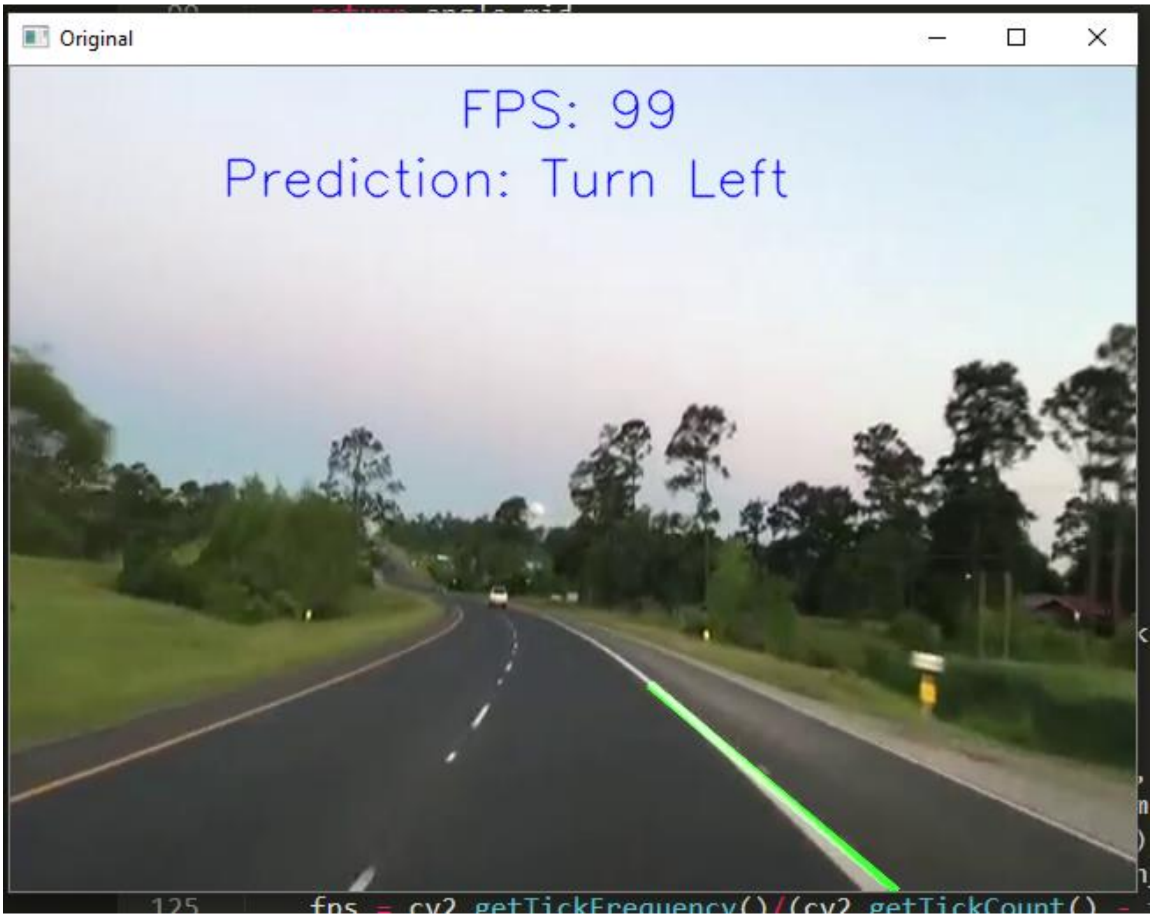
After that we applied masking to our edge detected image to have only region of interest and as result



It was essential phase, because after that we found Hough lines on image above. Those Hough lines afterwards were separated into two lines right and left lines. Those lines afterwards were displayed on the original frame:



Last two tasks we accomplished were predicting the turn and calculating FPS. The first one we achieved by calculating incline/decline angles of both lines. FPS was calculated by dividing frequency of cycles and number of cycles in each iteration.



```
FPS: 99
Prediction: Turn Left
```

```
fps = cv2.getTickFrequency()/(cv2.getTickCount() -
```

CONCLUSION

Developing this task allowed me and my team members to enhance skills in software engineering and team communication. In this project, we implemented our knowledge not only in advanced topics of computer vision including edge detection and Hough spaces but also in properties of linear functions. Each one of us become more familiar with the with Python programming language and expand knowledge in usage of OpenCV library. Working on this code, like no other was very useful in exploring innovative modern technologies. Within the project we build such features as image resizing, color correction, denoising, masking, image edge detection, Hough lines detection and turn prediction. All of that feature is structured in an efficient code that currently is functional only as software program. However, our team is highly interested in Implementing it to embedded system and unleash its full potential.

REFERENCES

[1] https://docs.python.org/3/library/math.html?highlight=math%20library

[2] https://numpy.org/devdocs/user/quickstart.html

[3] https://docs.opencv.org/master/d6/d00/tutorial_py_root.html