VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

Project Report on

"CRUISE CONTROL USING IMAGE ANALYSIS"

Submitted in partial fulfillment of the requirements for the VII Semester

Bachelor of Engineering

in

ELECTRONICS AND COMMUNICATION ENGINEERING

For the Academic Year

2020-2021

BY,

AASHIKA CHAKRAVARTY 1PE17EC003

UNDER THE GUIDANCE OF

DR. SUBHASH KULKARNI

Principal, PESIT-BSC

Department of Electronics and Communication Engineering PESIT - BANGALORE SOUTH CAMPUS

Hosur Road, Bengaluru - 56010

PESIT - BANGALORE SOUTH CAMPUS

HOSUR ROAD, BENGALURU - 560100

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project work titled "Cruise Control using Image Analysis" carried out by "**Aashika Chakravarty**" bearing the USN **1PE17EC003**, in partial fulfillment for the award of Degree of Bachelors (Bachelors of Engineering) in Electronics and Communication Engineering of Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/ suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Signature of the Guide	Signature of the HOD	Signature of the Principal
Dr. Subhash Kulkarni	**Dr. Subhash Kulkarni**	**Dr. Subhash Kulkarni**
Principal, PESIT-BSC	**HOD, ECE**	**Principal, PESIT-BSC**

External Viva

Name of the Examiners

1.

2.

ECE Department, PESIT-BSC, Bangalore

ABSTRACT

Autonomous cars are the future. Smart cars anticipated to be driver less, efficient and crash-avoiding; ideal urban car of the future.
To reach this goal, automakers have started working in this area to realize the potential and solve the challenges currently in this area to reach the expected outcome.

It has various potential benefits, including increased productivity, more efficient emergency and delivery services, fewer accidents and driver errors, and its positive impact on the environment.

We also have to look at the challenges and restrictions that we may face on the way, like limited vision (due to the camera and its quality), lack of powerful hardware, and the fact that this technology is still in development makes it difficult for complete accuracy.

Our goal is to contribute to this field, emulating advancements made by companies like Tesla and build a miniature car that can safely navigate and drive itself, staying within the lane, and watching out for pedestrians and traffic signals and acting accordingly.

We will be attempting this project using various technologies and algorithms and testing each one for the best performance, computation time, and accuracy and implement it.

Contents

1 Introduction **1**

1.1 INTRODUCTION

1.2 PROBLEM STATEMENT

1.3 MODULAR BREAKDOWN

2 Literature Survey **2**

3 Hardware and Software Requirements Specification **4**

3.1 HARDWARE

3.2 SOFTWARE

4 System Design **5**

4.1 SYSTEM ARCHITECTURE

5 Implementation **10**

6 Result Analysis **12**

7 Conclusion and Future Scope **14**

7.1 Conclusion

7.2 Future Scope

References **15**

A Review Process **16**

A.1 Phases of Review

B Project Planning **18**

List of Figures

Page No.

Fig1. Output of canny edge detector. 6

Fig2. The output from lane detection highlighting the path to follow. 7

Fig.3 Block representation of the YOLOv4 model 7

Fig.4 YOLOv4 object detector 8

Fig.5 Output of the object detector 9

Fig.6 3-D render of the completed model car 11

Fig.7 Canny Edge detection, gradient and direction 13

Fig.8 Hough Transform 13

Chapter 1

Introduction

1.1 Introduction

Self-driving vehicles will bring about the largest societal revolution. With numerous benefits and solutions to current daily issues, they're on the road to become a staple in the automotive industry.

1.2 Problem Statement

Our aim is to build and program a small car capable of detecting the lane and operating in real time by responding to daily traffic situations.

1.3 Modular Breakup of Project

1. Preprocessing
2. Lane and Edge Detection
3. Object Detection
4. Deployment on hardware
5. Bot control
6. Refining detection techniques to improve accuracy

Chapter 2

Literature Survey

1. Vision-based Lane Detection and Tracking for Driver Assistance Systems: a Survey by Hui Zhou School of Electrical and Electronic Engineering Nanyang Technological University Singapore, 639798 in this paper, we review the vision-based lane detection and tracking methods complemented with other sensor information when necessary. Approaches that adopt conventional computer vision techniques are reviewed and compared according to the separate functional modules in a generic framework.

2. Lane Detection and Tracking for Intelligent Vehicles: A Survey by Mehdi FENICHE National School of Applied Sciences Kenitra- Nowadays, Lane detection and tracking modules are considered as central requirements in every Intelligent Transportation System (ITS) development. The extracted lane information could be used in several smart applications for lane
keeping systems, lane departure warning and avoiding collisions with other vehicles

3. The YOLOv3: An Incremental Improvement is a paper presented by Joseph Redmon, Ali Farhadi, University of Washington, 2018. was instrumental in developing the object detection aspect of our project we used the same model architecture and trained it to identify 50 different object classes to enable our device to be capable of not only detecting obstacle but also identifying it.

4. The YOLOv4 is a paper by Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao Institute of Information Science Academia Sinica, Taiwan released in 2020 . It is a one-stage object detection model that improves on YOLOv3 with several new features and modules introduced in the literature.

5..Deep Residual Learning for Image Recognition; by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Microsoft Research was used in deploying the object detection for our project we used the same model architecture and trained it to identify 50 different object classes to enable our device to be capable of detecting and identifying obstacles. However this was not used in our application as the hardware used couldn't provide an acceptable output rate (in terms frames per second *fps*) for smooth operation.

The above mentioned papers even if the results and research were not actually used, were instrumental for us in gaining insight on how various aspects of image processing, Computer Vision, Deep neural networks and the architecture of various models work.

Chapter 3

Hardware and Software Requirements Specification

3.1 Hardware

- Nvidia Jetson Nano 2GB
  It's a 128-core GPU, Quad-core CPU developer kit. It has 2GB of memory, with a microSD slot.
  It has multiple USB ports (type A [3.0, 2.0] and micro-B 2.0), Ethernet and wireless connectivity, a camera connector, and an HDMI display, along with multiple other features
  Its dimensions are 100mm x 80mm x 29mm

- Web camera
  To capture the road and process the images received.

- Power bank
  18,000 mAh power source in order to power the bot.

- Miniature automobile Chassis and DC motors.

- Motor control drivers(by Adafruit) to control the movement model car (bot)

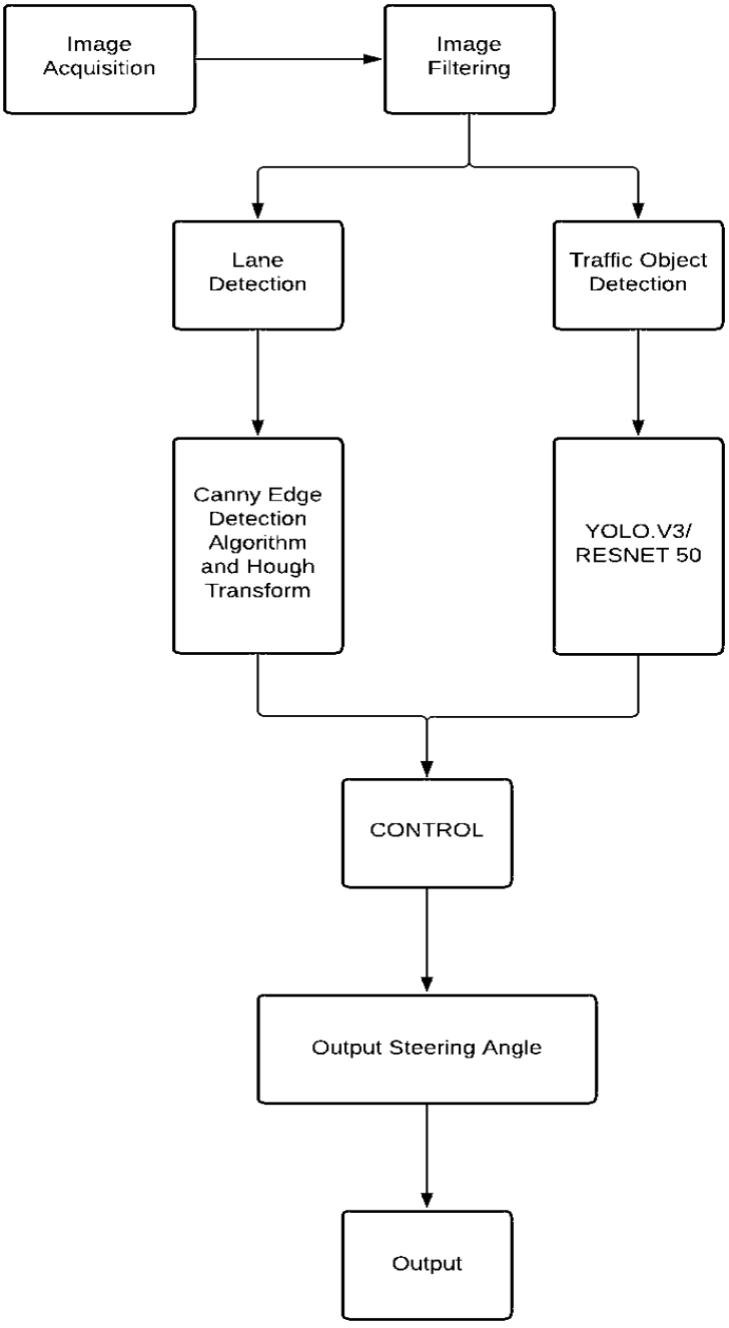
3.2 Software

- Python is the programming language of choice.
- Tensorflow, Keras are the required framework and api used to develop computer vision model.
- Opencv is the image processing library used for feature extraction and other image processing operations.
- A custom version of Ubuntu provided by Nvidia OS

Chapter 4

System Design

4.1 SYSTEM ARCHITECTURE



1. Image Acquisition:

The first stage of any vision system is the image acquisition stage. The video is captured from Camera which is mounted on the car. High resolution video is captured in real time to get higher accuracy. We have used webcam for our project demonstration.

2. Image Filtering:

Before detecting the lane lines in image, we pre -process the image by applying various filters. From the video captured in real time, each frame in the video is applied with a Gaussian blur filter with a fixed kernel size of 5*5. This is generally applied to remove noise in the image.
After Gaussian blurring, the frames are then converted to HSV format, so that we get true colors of the image.

3. Canny Edge Detection Algorithm and Hough Transform:

Canny Edge Detection is a popular edge detection algorithm. It is a multi-stage algorithm and each stage is as follows:

3a. Finding Intensity Gradient of the Image:

Image is filtered with a Sobel kernel in both horizontal and vertical directions to get the first derivative in the horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:
Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3b. This stage decides which edges are really edges and which are not. 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. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.

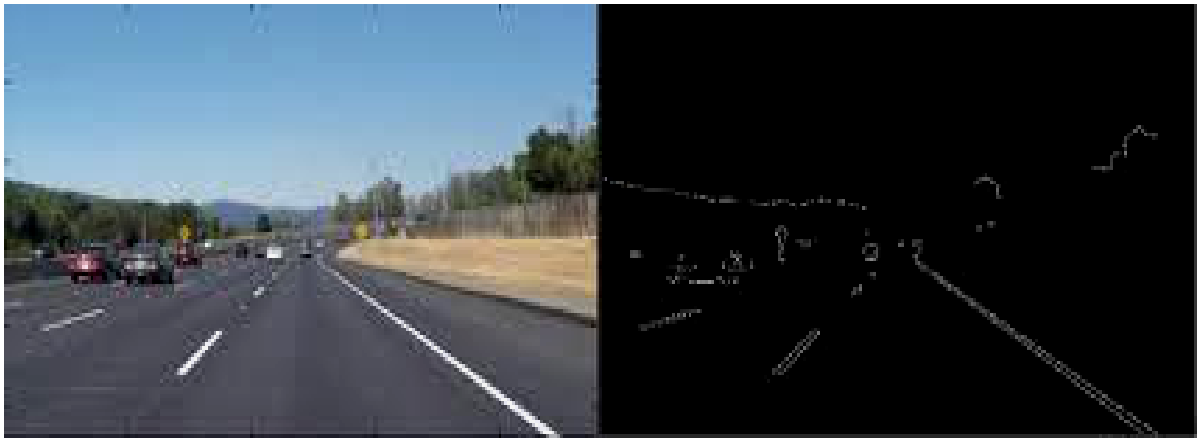


FIG1.

3c. Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. Any line in the x-y plane can be represented by a point in the Hough plane. So, the complexity to process a line is eliminated. The point in the plane gives us the slope and the intercept. Probabilistic Hough Transform is an optimization of Hough Transform. It doesn't take all the points into consideration, instead take only a random subset of points
and that is sufficient for line detection. Just we have to decrease the threshold.

Fig.2

.

4. Real time traffic objects detection using YOLO V4:

You Only Look Once real time objects detection system. The architecture of YOLO is shown in FIG3. It consists of several convolution neural networks stacked back-to-back.

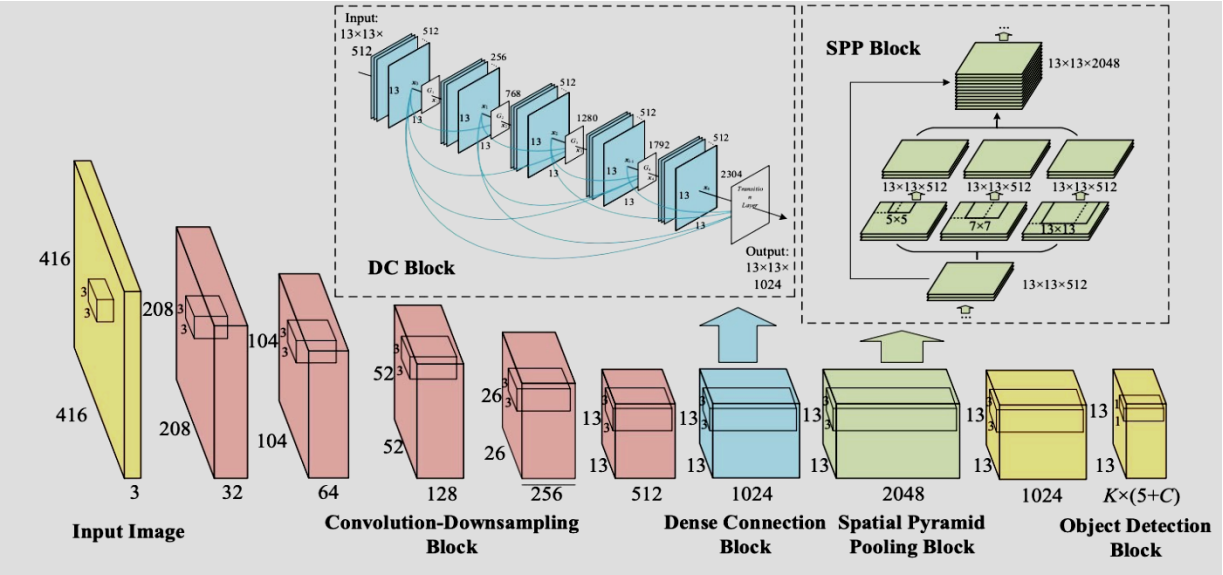


Fig.3

YOLO uses features from the image and predicts bounding boxes simultaneously.
The image is divided into an S X S grid and each gird produces B bounding boxes
and their confidence scores. The confidence score is an indicator of how accurate the model thinks it is about what is actually enclosed in the box and what it has predicted.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the mid-point of the box with respect to the bounds of the grid cell. The w and h are the width and height of the object and is relative to the whole image.

An ordinary object detector is composed of several parts (shown in fig. 4):

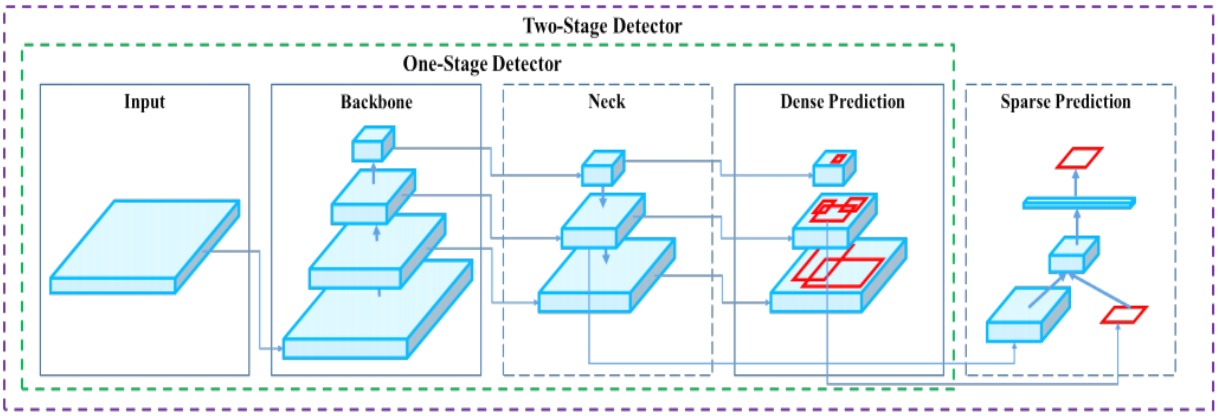


Fig4

1. Input: Image, Patches, Image Pyramid

2. Backbones: VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50, CSPDarknet53

3. Neck:
    1. Additional blocks: SPP, ASPP, RFB, SAM
    2. Path-aggregation blocks: FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM

4. Heads:
    1. Dense Prediction (one-stage):
        1. RPN, SSD, YOLO, RetinaNet (anchor based)
        2. CornerNet, CenterNet, MatrixNet, FCOS (anchor free)

2. Sparse Prediction (two-stage):
1. Faster R-CNN, R-FCN, Mask RCNN (anchor based)
2. RepPoints (anchor free)

The YOLO model in use is trained for our traffic object dataset so that it detects the traffic objects. The traffic objects such as cars, vehicles, traffic lights etc. are detected. Based on the decision to be taken the car identifies the objects location from the video feed.

Fig .Output of the object detector

Chapter 5

Implementation

The software developed is deployed on our hardware. We use NVIDIA's Jetson Nano as the heart of our project. Jetson Nano is a Single Board Computer (SBC) used for embedded applications and AI IoT that delivers. Jetson Nano has the performance and capabilities we need to run modern AI workloads as a result of having a discrete 128 CUDA core Nvidia-Maxwell GPU along with a Quad core ARM CPU, giving a fast and easy way to add advanced AI to the next product.

Webcam is mounted onto the chassis of the model car, which offers video footage of 720p with 30fps.

The Jetson nano is booted with newest version of the custom Ubuntu OS available on the official Jetson nano page. It is done by etching the image downloaded to a micro-SD card and inserting it into the Jetson nano

Once booted the OS comes with its own installation of python tensorflow, keras, pytorch along with many other tools required for deep learning, machine learning, and Computer vision applications. The code developed for Lane detection and Object detection from the discussions earlier is copied onto the device and the code is run.

As we need to control the speed of the car and steer we use an Adafruit motor driver. This motor driver is extremely versatile which makes it useful for a wide variety of applications. The GPIO pins on board the jetson nano is used to communicate to the Adafruit motor controller module on how to maneuver the car/bot along the custom track built for testing.

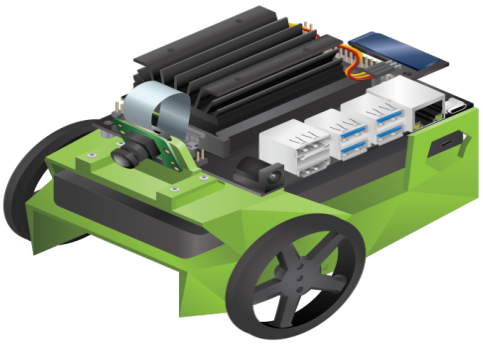


Fig.6

Once the entire model is connected it will look like as shown in the *Fig.6* is now ready to be powered on using Li-ion rechargeable battery pack deployed for testing on the test track.

Chapter 6

Result Analysis

Automation and Higher levels of autonomy have the potential to reduce risky and dangerous driver behaviors and can help reduce the number of crashes on our roads

In this project, computer vision techniques were used for object detection, collision avoidance and lane detection of autonomous driving systems. Real time traffic object detection is implemented using deep learning.

Traffic Object Detection: On the course of working on this project we deployed various object detection algorithms and measured the rate at which they can run on the presented hardware platform and we are to choose the one which can give reasonably faster results with a high accuracy for efficient working of the system. When comparing results of Yolo v3, Yolo v4 and ResNet we have then concurred that the YoloV4 model is the right choice for our use case.

Lane Detection: Real time lane detection is a difficult task to be computed without deep learning, but the algorithm which we have used in our project has given us a higher accuracy with a less delay.
The image is taken and blurred at the initial to remove noise present in it while accessing it. Gaussian filter has been used for this purpose, which acts as smoothing filter for the raw image. Extra care has been taken to convert the image into HSV format to extract the true colors in the image. Converting to HSV, gives us a unique image for various environment problems such as low illumination, extra sunlight and much more.
Pre- processing steps are done to remove noise and to get a higher accuracy in the output, here output refers to detection of lanes. Canny Edge detection algorithm is used to detect the edges in the image. Canny edge detector has given us proper edges as compared to other image processing algorithms such as LoG, high pass filtering and sobel operator.

The expression for edge gradient and the direction of edge is given below.

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Fig .7

To grab the co-ordinates of edge detected by Canny edge detector we use Hough transform. Any line in x-y plane can be represented by a point in Hough plane as follows.

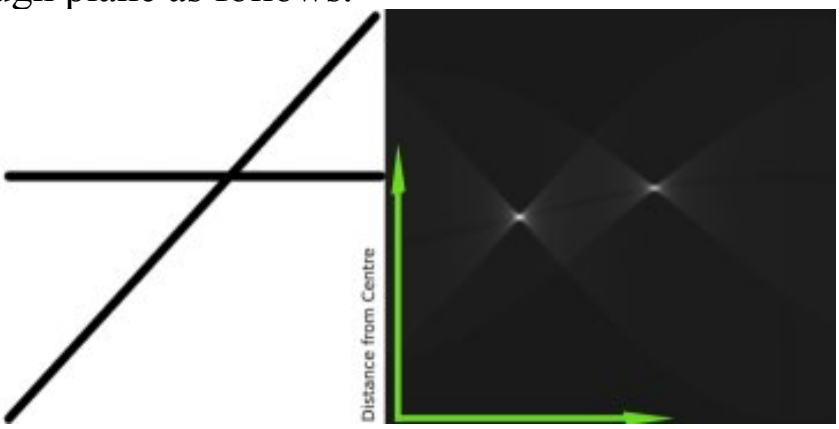


Fig.8

This reduces complexity as we need to just process a point instead of a line. This increases visual understanding as many lines can be thought as points.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

We were able to implement a fully functioning miniature model of a self driving car, capable of detecting the road, and identifying and avoiding collisions with obstacles on the road, by stopping until the obstacle is removed

7.2 Future Scope

Future endeavors and improvements on this project can be as follows

- Faster response time and increased frame rate in order to improve accuracy and performance on the road in real-time situations
- Ability to read and follow road signs, keeps up with speed limits, and read traffic signals and act accordingly.
- Ability to be able to maneuver around obstacles by calculating relative distance and required steering angle.

References

1. Lane Detection and Tracking, For Intelligent Vehicles: A Survey Mehdi FENICHE National School of Applied Sciences Kenitra-Moroccomehdifeniche@gmail.com, Tomader MAZRINational School of Applied Sciences, Kenitra-Morocco, IEEE/ICCSRE2019, 22-24 July, 2019, Agadir, Morocco.

2. Real-Time Lane Detection and Tracking for Advanced Driver Assistance Systems. Hazrat Bilal1, Baoqun Yin1, Jawad Khan2, Luyang Wang1, Jing Zhang1, and Aakash Kumar Proceedings of the 38th Chinese Control Conference, July 27-30, 2019, Guangzhou, China.

3. Lane_detect2Curve Path Prediction and Vehicle Detection, in Lane Roads Using Deep Learning for, Autonomous Vehicles G. Pavithra, N.M. Dhanya February 2019

4. Object Detection for Autonomous Vehicles Gene Lewis, Stanford University, Stanford, CA

5. YOLOv3: An Incremental Improvement; by Joseph Redmon, Ali Farhadi, University of Washington, 2018 [https://arxiv.org/abs/1804.02767]

6. YOLOv4: Optimal Speed and Accuracy of Object Detection; by The YOLOv4 is a paper by Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao Institute of Information Science Academia Sinica, Taiwan released in 2020
.

7. .Deep Residual Learning for Image Recognition; by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Microsoft Research

Appendix A

Review Process

A.1 Phases of Review

Project was evaluated at phases with major checkpoints as follows:

Pre-Review: ***Defining Problem Statement and Feasibility analysis Review***

1. Understanding of the problem statement.
2. Technical understanding of domain.
3. Identification of differentiating features.
4. Feasibility of conversion to product or paper.

1st Review: ***Literature survey and Project plan Review***

1. Clarity in understanding of the problem/project.
2. Completion of Literature Survey.
3. Identification of sub-blocks and their interaction.
4. Timeline for completion of project using Gantt chart.

2nd Review: ***Module level design and Test Plan Review***

1. Detail design of each module.
2. Integration and module test plan.
3. Availability status of required Hardware and Software components.
4. 20% Completion of Block/Sub module implementation.

3rd Review: *Project Progress Review*

1. Adherences to project plan.
2. Completion of module interaction interface design.
3. 50% Module level implementation.
4. Demonstration of completed modules using primitive interfaces

4th Review: *Project Progress and Finishing plan Review*

1. 70% Module level implementation
2. Presentation of completed test data as per test plan.
3. Packaging plan.
4. Final demonstration plan.

5th Review: *Module completion and Integration Review*

1. 100% Module implementation.
2. Completion of Integration.
3. Presentation of test data as per test plan.
4. Adherences to project plan.

6th Review: *Demonstration Review*

1. Quality of packaged full demo presentation.
2. Integration test data as per test plan.
3. Quality of documentation made across project.
4. Group presentation / communication skill.

Appendix B

Project Planning

B.1 Image acquisition

B.2 Lane Detection

B.3 Object Detection

B.4 Hardware implementation

B.5 Control

B.6 Refinement to improve accuracy