SELF DRIVING CAR USING TENSORFLOW

Submitted in partial fulfillment of the requirements
of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND TELECOMMUNICATION

BY

WAQQAS THAKUR	16DET72
USMAN SHAIKHNAG	16DET70
MD.AABID PATEL	16DET109
ARMAN ANSARI	15ET14

UNDER THE GUIDANCE OF

PROF. ZARRAR AHMED KHAN

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING
ANJUMAN-I-ISLAM'S KALSEKAR TECHNICAL CAMPUS
Sector 16, New Panvel, Navi Mumbai-410206
University of Mumbai

2018-2019

SELF DRIVING CAR USING TENSORFLOW

Submitted in partial fulfillment of the requirements
of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND TELECOMMUNICATION

BY

WAQQAS THAKUR	16DET72
USMAN SHAIKHNAG	16DET70
MD.AABID PATEL	16DET109
ARMAN ANSARI	15ET14

UNDER THE GUIDANCE OF

PROF. ZARRAR AHMED KHAN

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING
ANJUMAN-I-ISLAM'S KALSEKAR TECHNICAL CAMPUS
Sector 16, New Panvel, Navi Mumbai-410206
University of Mumbai

2018-2019

Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals . I would like to extend my sincere thanks to all of them.

We am highly indebted to **Prof. Zarrar Khan** for their guidance and constant supervision as well as for providing necessary information regarding the project also for their support in completing the project.

We would like to express my gratitude towards my parents Staff of Anjuman-I-Islam's Kalsekar Technical Campus for their kind co-operation and encouragement which help me in completion of this project.

We My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

WAQQAS THAKUR
USMAN SHAIKHNAG
MD.AABID PATEL
ARMAN ANSARI

ABSTRACT

Deep Learning has led us to newer possibilities in solving complex control and navigation related tasks. The paper presents Deep Learning with back propagation autonomous navigation and obstacle avoidance of self-driving cars, applied with Deep Q Network to a simulated car an urban environment. The approach uses two types of sensor data as input: camera sensor and laser sensor in front of the car. It also designs a cost-efcient high-speed car prototype capable of running the same algorithm in real-time. The design uses a camera and a Hokuyo Lidar sensor in the car front. It uses embedded GPU (Nvidia-TX2) or CPU for running deep-learning algorithms based on sensor inputs.

Keywords: Deep Learning, Lidar sensor

Contents

List of Figures

Chapter 1

Introduction

1.1 HISTORY OF SELF DRIVING CARS

Figure 1.1: Self Drive

The autonomous car has been in the headlines for a decade and still continues to dominate auto headlines. The autonomous car has attracted the researchers, robotics communities and the automobile industries. Human driving is accident-prone. We are drunk, drowsy and tired. Our inefciency to make better on-road decisions cause road accidents, property loss and deaths. The autonomous car brings us the possibilities to replace accident-prone human driver by providing comfort and safety. The automated car concept was originated in 1920, but it evolved truly in 1980 with CMUs Navlab and ALV projects. CMUs effort resulted in the ALVINN in 1989, the rst try in car autonomy which used a neural network to drive. The United States government invested in three projects for autonomous car which were demonstrated by US Army (Demo-I, Demo-III) and DARPA (Demo-II) in the 2000s. From 2004 to 2007, DARPA organised 3 car autonomy challenges which got attentional lover the world at that time. The automobile industries like Ford, Mercedes, Volkswagen, Audi, Nissan, Toyota, BMW and Volvo came to play a bigger role in the autonomous car area by announcing their plans for it. In 2014, Google introduced Google car with sensor fusion technology (mainly camera and lidar sensor). One year later, Tesla brought their autonomous car in 2015. Nvidia has recently introduced a self-driving car with end-to-end convolutional neural network. In 2017, Audi stated latest A8 would be autonomous at up to speed of 60 km/h using Audi AI. The driver doesnt require safety checks such as frequently gripping the steering wheel. The Audi A8 is claimed to be the rst production car to reach level 3 autonomous driving and Audi would be the rst manufacturer to use laser scanners in addition to cameras and ultrasonic sensors for their system. On the 7th November 2017, Waymo announced that it has begun testing driver-less cars without a safety driver at the driver position. However, there is an employee in the car. And the list still continues with Uber, Tata, Intel and other automobile industries. Driving is a complex problem. The evolution of autonomous car involves complex modules, which uses traditional control algorithms, SLAM based navigation systems and Advanced Driver Assistant Systems in recent days. But all the previous methods fail to replace human driver completely. But increasing computing power of GPUs, accurate sensors and deep neural networks has made complete autonomy is realisable. But recent failure of Uber and Tesla, and the listed disengagements in show that the autonomous car research needs more attention in the upcoming years to achieve better than human-level driving. In the context of human-level control, presented a deep reinforcement learning approach which redesigned Q-Learning using a deep neural network. It introduces 4 steps to unstable Q-learning: Experience Replay, Target Network, Clipping Rewards and Skipping Frames. It achieved human-level performance (sometimes even better than human) tested in different complex game environment. The breakthrough achievement of DQN has motivated us to implement it in car autonomy for human-level performance. In this paper,we present a DRL-based autonomous driving strategy for urban environment using Deep Q Network. We use sensor fusion on two types of input sensor data: vision based camera sensor and a laser sensor. The approach is tested in simulation environment, which is designed using Unity Game Engine. We also design a car prototype capable of running our approach in real-time. The prototype uses Hokuyo Lidar Sensor and camera sensors embedded with Nvidia Jetson TX2 embedded GPU.The paper structure is as follows: section I introduces autonomous car and the neural network approaches in car autonomy, section II presents the background and mathematical representation of Deep Q Network. Section III describes proposed methodology with the network architecture and congurations, action denitions and simulation environment. It also presents the learning curve during network training. Section IV discusses our car prototype capable of running DQN in real-time.

1.2 INTRODUCTION TO MACHINE LEARNING

Machine Learning is an idea to learn from examples and experience, without being explicitly programmed. Instead of writing code, you feed data to the generic algorithm, and it builds logic based on the data given. For example, one kind of algorithm is a classification algorithm. It can put data into different groups. The classification algorithm used to detect handwritten alphabets could also be used to classify emails into spam and not-spam. As stated by Tom M. Mitchell A computer program is said to learn from experience E with some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.. Consider playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

Examples of Machine Learning

There are many examples of machine learning. Here are a few examples of classification problems where the goal is to categorize objects into a fixed set of categories.

Face detection: Identify faces in images (or indicate if a face is present).

Email filtering: Classify emails into spam and not-spam.

Medical diagnosis: Diagnose a patient as a sufferer or non-sufferer of some disease.

Weather prediction: Predict, for instance, whether or not it will rain tomorrow.

1.2.1 NEED OF MACHINE LEARNING

Machine Learning is a field which is raised out of Artificial Intelligence(AI). Applying AI, we wanted to build better and intelligent machines. But except for few mere tasks such as finding the shortest path between point A and B, we were unable to program more complex and constantly evolving challenges.There was a realisation that the only way to be able to achieve this task was to let machine learn from itself. This sounds similar to a child learning from its self. So machine learning was developed as a new capability for computers. And now machine learning is present in so many segments of technology, that we dont even realise it while using it.[3]

Finding patterns in data on planet earth is possible only for human brains. The data being very massive, the time taken to compute is increased, and this is where Machine Learning comes into action, to help people with large data in minimum time.[3]

If big data and cloud computing are gaining importance for their contributions, machine learning as technology helps analyse those big chunks of data, easing the task of data scientists in an automated process and gaining equal importance and recognition.[3]

The techniques we use for data mining have been around for many years, but they were not effective as they did not have the competitive power to run the algorithms. If you run deep learning with access to better data, the output we get will lead to dramatic breakthroughs which is machine learning.[3]

1.2.2 KINDS OF MACHINE LEARNING

There are three kinds of Machine Learning Algorithms.

a. Supervised Learning

b. Unsupervised Learning

c. Reinforcement Learning

1.3 INTRODUCTION TO COMPUTER VISION

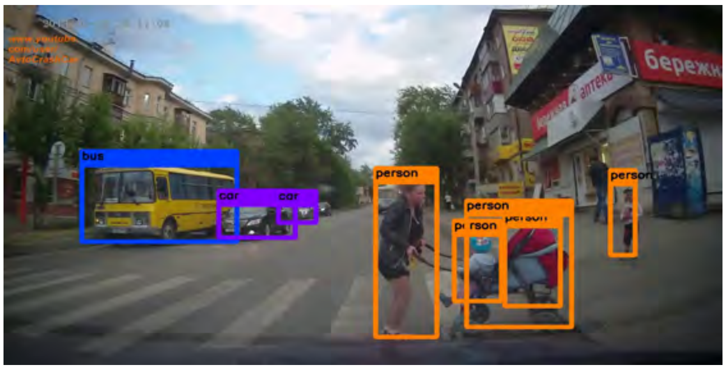


Figure 1.2: Object detection

Humans use their eyes and their brains to see and visually sense the world around them. Computer vision is the science that aims to give a similar, if not better, capability to a machine or computer.[]

Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding.[]

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. [2] Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.[2]

As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.[2]

In the late 1960s, computer vision began at universities that were pioneering artificial intelligence. It was meant to mimic the human visual system, as a stepping stone to endowing robots with intelligent behavior. In 1966, it was believed that this could be achieved through a summer project, by attaching a camera to a computer and having it ”describe what it saw”.[2]

What distinguished computer vision from the prevalent field of digital image processing at that time was a desire to extract three-dimensional structure from images with the goal of achieving full scene understanding. Studies in the 1970s formed the early foundations for many of the computer vision algorithms that exist today, including extraction of edges from images, labeling of lines, non-polyhedral and polyhedral modeling, representation of objects as interconnections of smaller structures, optical flow, and motion estimation.[2]

The next decade saw studies based on more rigorous mathematical analysis and quantitative aspects of computer vision. These include the concept of scale-space, the inference of shape from various cues such as shading, texture and focus, and contour models known as snakes. Researchers also realized that many of these mathematical concepts could be treated within the same optimization framework as regularization and Markov random fields. By the 1990s, some of the previous research topics became more active than the others. Research in projective 3-D reconstructions led to better understanding of camera calibration. With the advent of optimization methods for camera calibration, it was realized that a lot of the ideas were already explored in bundle adjustment theory from the field of photogrammetry. This led to methods for sparse 3-D reconstructions of scenes from multiple images. Progress was made on the dense stereo correspondence problem and further multi-view stereo techniques. At the same time, variations of graph cut were used to solve image segmentation. This decade also marked the first time statistical learning techniques were used in practice to recognize faces in images (see Eigenface). Toward the end of the 1990s, a significant change came about with the increased interaction between the fields of computer graphics and computer vision. This included image-based rendering, image morphing, view interpolation, panoramic image stitching and early light-field rendering.[2]

Recent work has seen the resurgence of feature-based methods, used in conjunction with machine learning techniques and complex optimization frameworks.[2]

1.4 INTRODUCTION TO DEEP LEARNING AND NEURAL NETWORK

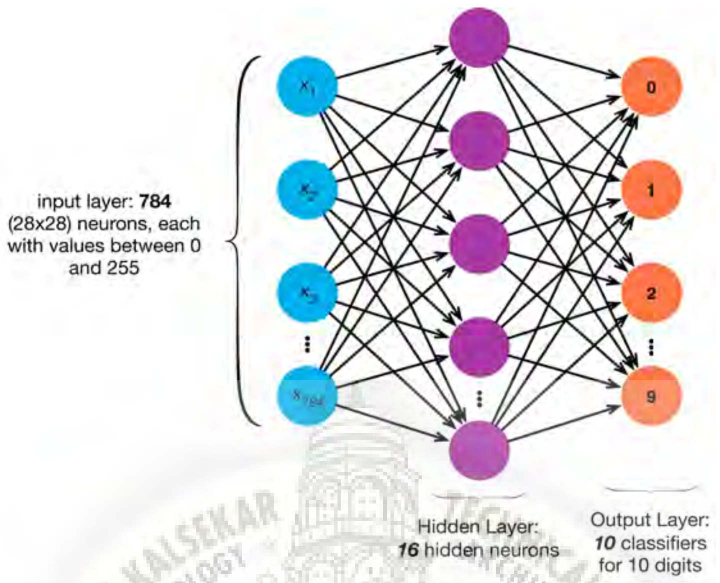

input layer: **784** (28x28) neurons, each with values between 0 and 255
x_1
x_2
x_3
x_{784}
0
1
2
9
Hidden Layer: ***16*** hidden neurons
Output Layer: ***10*** classifiers for 10 digits

Figure 1.3: Neural network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.[5]

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. (Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.)[5]

What kind of problems does deep learning solve, and more importantly, can it solve yours? To know the answer, you need to ask questions:

What outcomes do I care about? Those outcomes are labels that could be applied to data: for example, spam or not spam in an email filter, good guy or bad guy in fraud detection, angry customer or happy customer in customer relationship management.[5]

Do I have the data to accompany those labels? That is, can I find labeled data, or can I create a labeled dataset (with a service like AWS Mechanical Turk or Figure Eight or Mighty.ai) where spam has been labeled as spam, in order to teach an algorithm the correlation between labels and inputs?[5]

1.4.1 A FEW EXAMPLES

Deep learning maps inputs to outputs. It finds correlations. It is known as a universal approximator, because it can learn to approximate an unknown function f(x) = y between any input x and any output y, assuming they are related at all (by correlation or causation, for example). In the process of learning, a neural network finds the right f, or the correct manner of transforming x into y, whether that be f(x) = 3x + 12 or f(x) = 9x - 0.1. Here are a few examples of what deep learning can do.[5]

Classification

All classification tasks depend upon labeled datasets; that is, humans must transfer their knowledge to the dataset in order for a neural to learn the correlation between labels and data. This is known as supervised learning.[5] Detect faces, identify people in images, recognize facial expressions (angry, joyful) Identify objects in images (stop signs, pedestrians, lane markers) Recognize gestures in video Detect voices, identify speakers, transcribe speech to text, recognize sentiment in voices Classify text as spam (in emails), or fraudulent (in insurance claims); recognize sentiment in text (customer feedback) Any labels that humans can generate, any outcomes you care about and which correlate to data, can be used to train a neural network.[5]

Clustering

Clustering or grouping is the detection of similarities. Deep learning does not require labels to detect similarities. Learning without labels is called unsupervised learning. Unlabeled data is the majority of data in the world. One law of machine learning is: the more data an algorithm can train on, the more accurate it will be. Therefore, unsupervised learning has the potential to produce highly accurate models.[5]

Search: Comparing documents, images or sounds to surface similar items. Anomaly detection: The flipside of detecting similarities is detecting anomalies, or unusual behavior. In many cases, unusual behavior correlates highly with things you want to detect and prevent, such as fraud.[5]

Predictive Analytics: Regressions

With classification, deep learning is able to establish correlations between, say, pixels in an image and the name of a person. You might call this a static prediction. By the same token, exposed to enough of the right data, deep learning is able to establish correlations between present events and future events. It can run regression between the past and the future. The future event is like the label in a sense. Deep learning doesnt necessarily care about time, or the fact that something hasnt happened yet. Given a time series, deep learning may read a string of number and predict the number most likely to occur next.[5]

Hardware breakdowns (data centers, manufacturing, transport) Health breakdowns (strokes, heart attacks based on vital stats and data from wearables) Customer churn (predicting the likelihood that a customer will leave, based on web activity and metadata) Employee turnover (ditto, but for employees) The better we can predict, the better we can prevent and pre-empt. As you can see, with neural networks, were moving towards a world of fewer surprises. Not zero surprises, just marginally fewer. Were also moving toward a world of smarter agents that combine neural networks with other algorithms like reinforcement learning to attain goals.[5]

With that brief overview of deep learning use cases, lets look at what neural nets are made of.[5]

Chapter 2

Literature Survey

2.1 PAPERS REFERED

We refered the following papers while developing the project

1. Paper 1

   Title

   Implementation of Vehicle Detection Algorithm for Self-Driving Car on Toll Road Cipularang using Python Language

   Outcomes

   The result shows this algorithm needed to be add some method that can adaptively changing the parameters during day and night adaptively. Because constant parameters can only be used in the same lighting conditions. Overall the implementation method in Python Language can successfully detect the vehicle.

2. Paper 2

   Title

   Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment

   Outcomes

   In this paper, we presented a reinforcement-learning based approach with Deep Q Network implemented in autonomous driving. We dened the reward function and agent (car) actions and trained the neural network accordingly to maximise the positive reward. We also discussed our sensor fusion technique: data concatenation of Lidar and Camera. We presented the simulated urban environment that was designed using Unity Game Engine. We tested the car for autonomous driving with the trained network in simulation. The simulation tests showed that the trained neural network was capable of driving the car autonomously by taking proper decisions (choose an action from action set) in the environment. Finally, we designed a car prototype based on our simulation experiments which is capable of running a DQN in real-time.

3. Paper 3

Title

VisualBackProp: efcient visualization of CNNs for autonomous driving

Outcomes

In this paper we propose a new method for visualizing the regions of the input image that have the highest inuence on the output of a CNN. The presented approach is computationally efcient which makes it a feasible method for real-time applications as well as for the analysis of large data sets. We provide theoretical justication for the proposed method and empirically show on the task of autonomous driving as well as other applications that it is a valuable diagnostic tool for CNNs and substantially contribute to our understanding of end-to-end learning systems.

4. Paper 4

Title

Realtime Vehicle and Pedestrian Tracking for Didi Udacity Self-Driving Car Challenge

Outcomes

The system described in this paper was implemented using Robotic Operating System (ROS) framework. On one side this was a hard constraint from committee, on the other side this framework supported the modularity of the system, making it possible to split application into several processes (nodes) which could be executed on different CPU cores and work concurrently. Additionally, the framework provided debug and visualization utilities.

Chapter 3

Software Requirements Specification

3.1 IMAGE PROCESSING

3.1.1 OPEN CV LIBRARY IN PYTHON

Figure 3.1: Open CV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence its free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

3.2 MACHINE LEARNING MODEL

3.2.1 TENSORFLOW LIBRARY IN PYTHON

Figure 3.2: Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Googles AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

3.2.2 KERAS IN PYTHON

Figure 3.3: Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Supports both convolutional networks and recurrent networks, as well as combinations of the two. Runs seamlessly on CPU and GPU. Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.6.

Chapter 4

Datasets

4.1 STREET SIGN DATASET

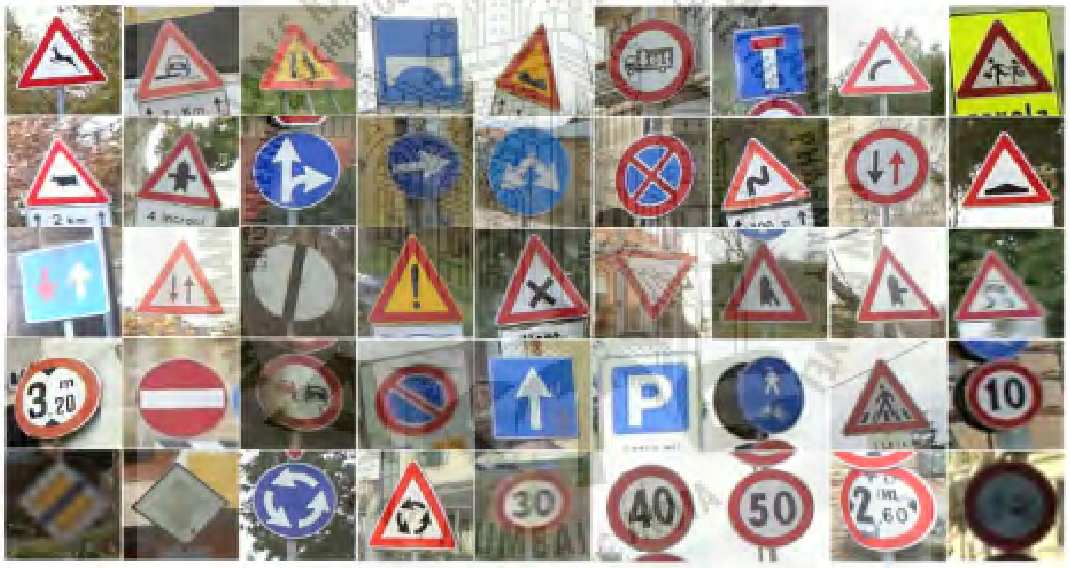


Figure 4.1: German traffic sign data-set

The above figure shows a portion of the data set we used for street sign detection, It is the German traffic sign data-set open sourced on GITHUB. It contains a total of around 56000 different images with labels and has 48 different classes for classification. It has One directory per class, Each directory contains one CSV file with annotations ("GT-¡ClassID¿.csv") and the training images, The training images are grouped by tracks, Each track contains 30 images of one single physical traffic sign

4.2 DATA COLLECTION FROM SIMULATOR

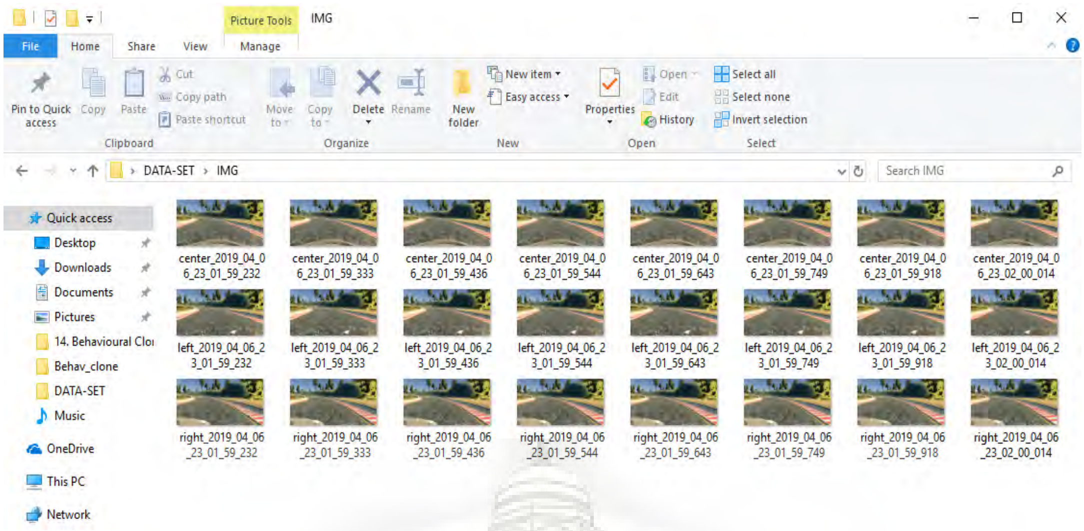
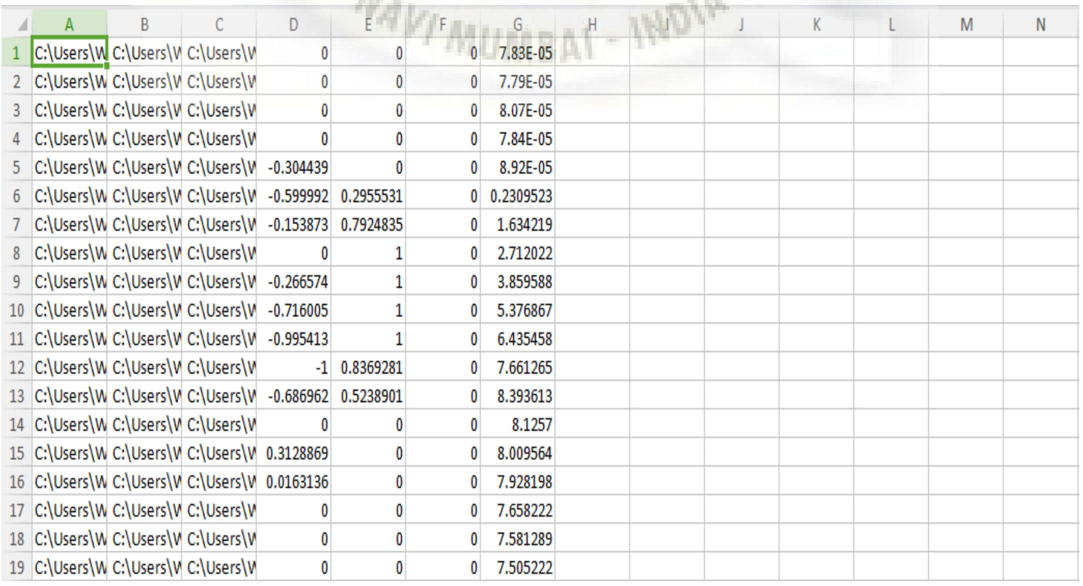

Picture Tools
IMG
File
Home
Share
View
Manage
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
DATA-SET › IMG
Search IMG
Quick access
Desktop
Downloads
Documents
Pictures
14. Behavioural Clo
Behav_clone
DATA-SET
Music
OneDrive
This PC
Network
center_2019_04_06_23_01_59_232
center_2019_04_06_23_01_59_333
center_2019_04_06_23_01_59_436
center_2019_04_06_23_01_59_544
center_2019_04_06_23_01_59_643
center_2019_04_06_23_01_59_749
center_2019_04_06_23_01_59_918
center_2019_04_06_23_02_00_014
left_2019_04_06_23_01_59_232
left_2019_04_06_23_01_59_333
left_2019_04_06_23_01_59_436
left_2019_04_06_23_01_59_544
left_2019_04_06_23_01_59_643
left_2019_04_06_23_01_59_749
left_2019_04_06_23_01_59_918
left_2019_04_06_23_02_00_014
right_2019_04_06_23_01_59_232
right_2019_04_06_23_01_59_333
right_2019_04_06_23_01_59_436
right_2019_04_06_23_01_59_544
right_2019_04_06_23_01_59_643
right_2019_04_06_23_01_59_749
right_2019_04_06_23_01_59_918
right_2019_04_06_23_02_00_014

Figure 4.2: Track images

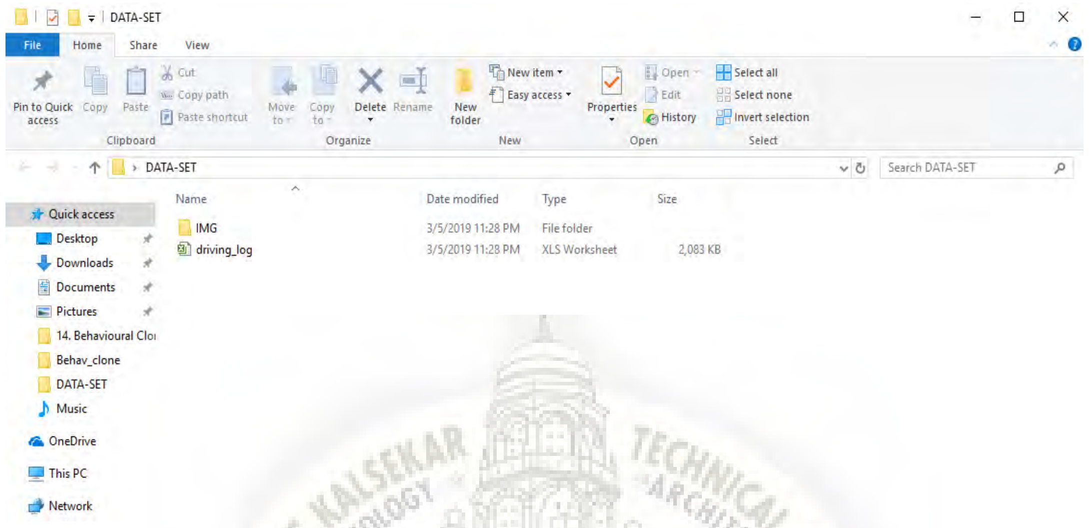
The above image shows the sample images captured by running the car around the training track, there are three images captured at every instance, the front or the center image from the center camera on the bonnet of the car, the right image captured by the right side camera and the left image captured by the left camera the images are captured by the simulator itself after clicking record and selecting folder where the dataset needs to be created, there are two things created one folder in which the images are and second the CSV file where the image path is given along with the steering angle at that instance the following figure shows the csv file format.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.83E-05							
2	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.79E-05							
3	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	8.07E-05							
4	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.84E-05							
5	C:\Users\W	C:\Users\W	C:\Users\W	-0.304439	0	0	8.92E-05							
6	C:\Users\W	C:\Users\W	C:\Users\W	-0.599992	0.2955531	0	0.2309523							
7	C:\Users\W	C:\Users\W	C:\Users\W	-0.153873	0.7924835	0	1.634219							
8	C:\Users\W	C:\Users\W	C:\Users\W	0	1	0	2.712022							
9	C:\Users\W	C:\Users\W	C:\Users\W	-0.266574	1	0	3.859588							
10	C:\Users\W	C:\Users\W	C:\Users\W	-0.716005	1	0	5.376867							
11	C:\Users\W	C:\Users\W	C:\Users\W	-0.995413	1	0	6.435458							
12	C:\Users\W	C:\Users\W	C:\Users\W	-1	0.8369281	0	7.661265							
13	C:\Users\W	C:\Users\W	C:\Users\W	-0.686962	0.5238901	0	8.393613							
14	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	8.1257							
15	C:\Users\W	C:\Users\W	C:\Users\W	0.3128869	0	0	8.009564							
16	C:\Users\W	C:\Users\W	C:\Users\W	0.0163136	0	0	7.928198							
17	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.658222							
18	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.581289							
19	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	7.505222							

Figure 4.3: CSV file

The csv file has different columns, the first column gives the image path the next three columns shows the steering angle and the last column shows the speed of the car at that imstance. The following image shows the folder which is created after runniing the car in the training track



Name	Date modified	Type	Size
IMG	3/5/2019 11:28 PM	File folder	
driving_log	3/5/2019 11:28 PM	XLS Worksheet	2,083 KB

Figure 4.4: data set folder

Chapter 5

Data Preprocessing

The image data we collected from the simulator is raw and we cannot use this data directly to create a model, We need to preprocess this data in order to extract important features from the data there are many steps in preprocessing the image data in our project

5.1 IMAGE COLOR CONVERSION

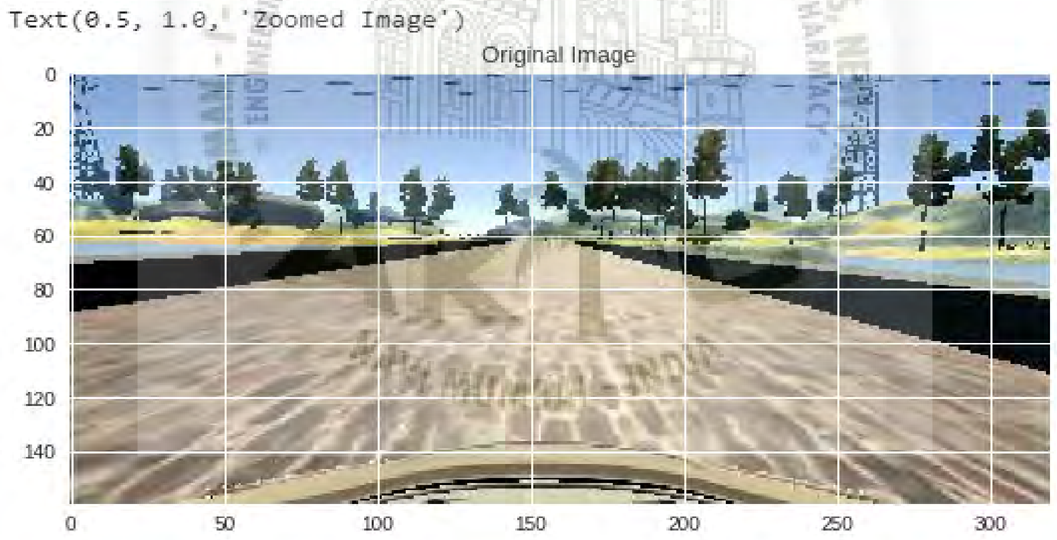

```
Text(0.5, 1.0, 'Zoomed Image')
```

Figure 5.1: Original image

We used
img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
command to convert RGB image to UV image

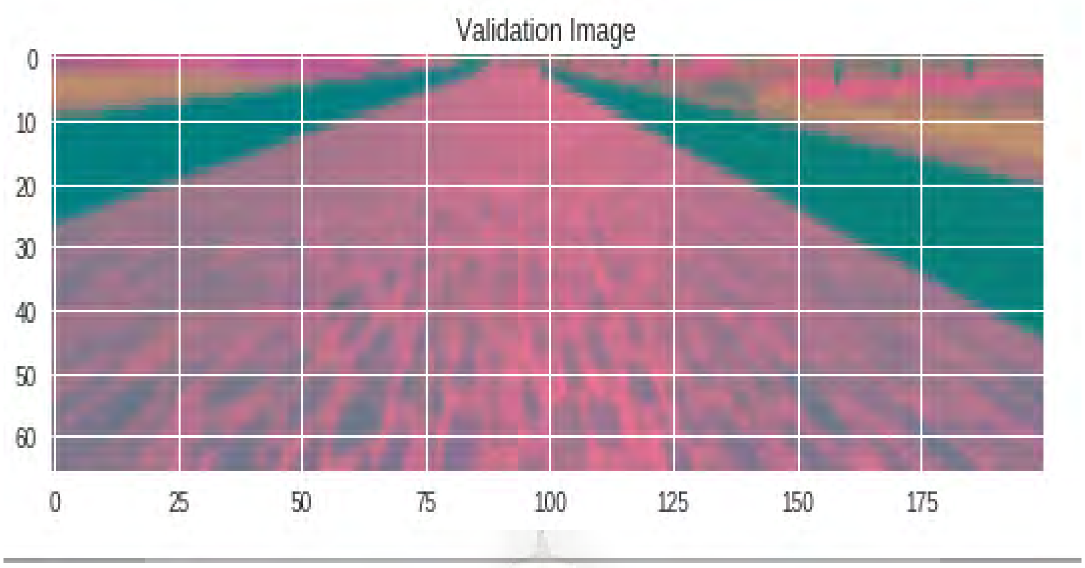


Figure 5.2: UV converted image

The images from our dataset are good but when we convert the image to UV image we get more accurate features of the image so we use the python commands to convert the image to UV images. The above figues shows the difference between the original and converted images

5.2 DATA BALANCING

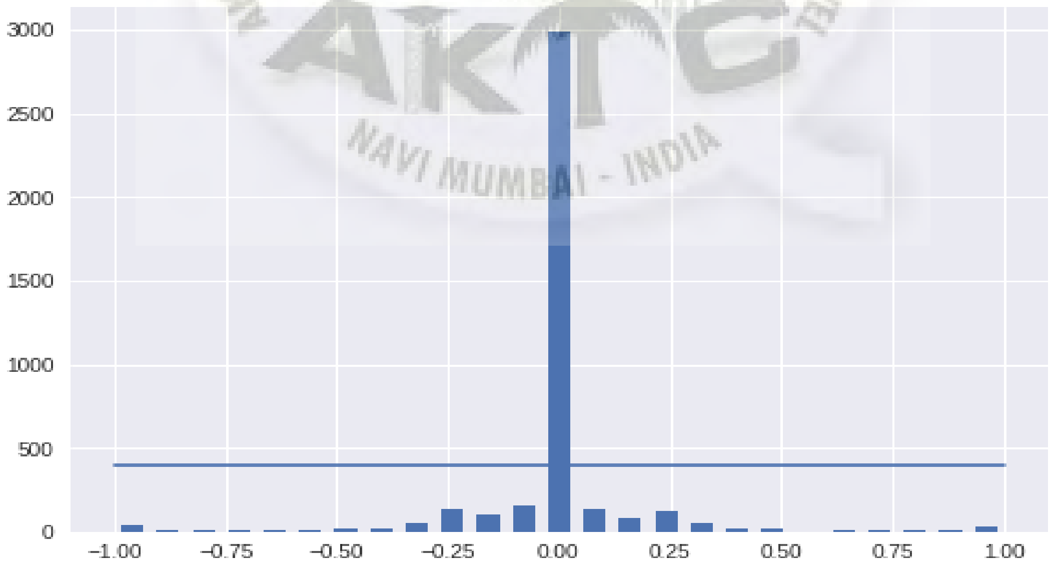


Figure 5.3: Original Dataset Graph

We can see by the graph that our dataset is a lot biased to straight driving that is zero steering angle and if we train the model on this data set we may run into problems like straight driving and the car getting out of the track so we need to reduce the amount of center images in order to balance our dataset the following figure show us the dataset after the balancing operation still the center driving is dominanat but we need more center driving than left or right driving in order to keep the car stable.

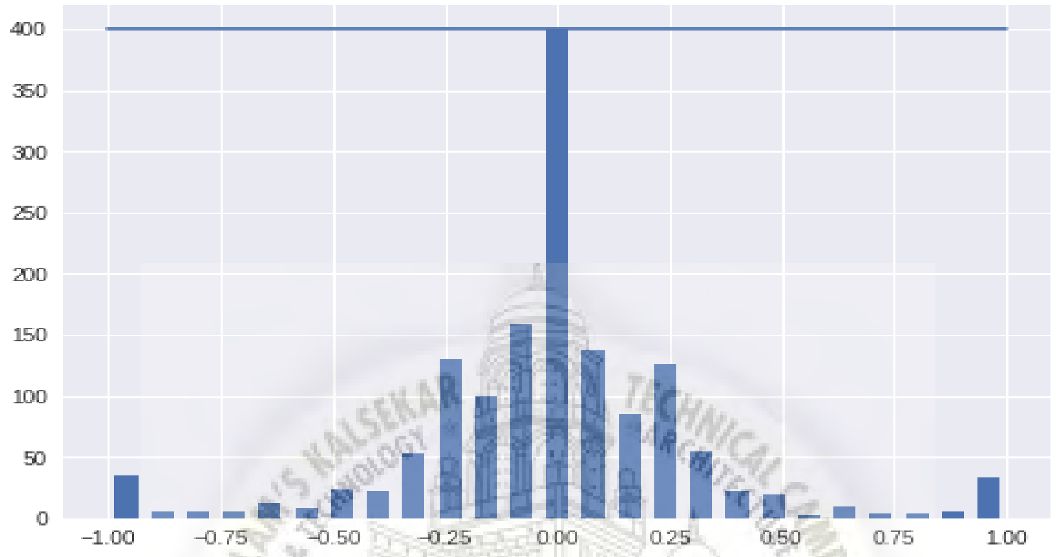


Figure 5.4: Balanced Dataset Graph

5.3 TRAIN-TEST SPLIT

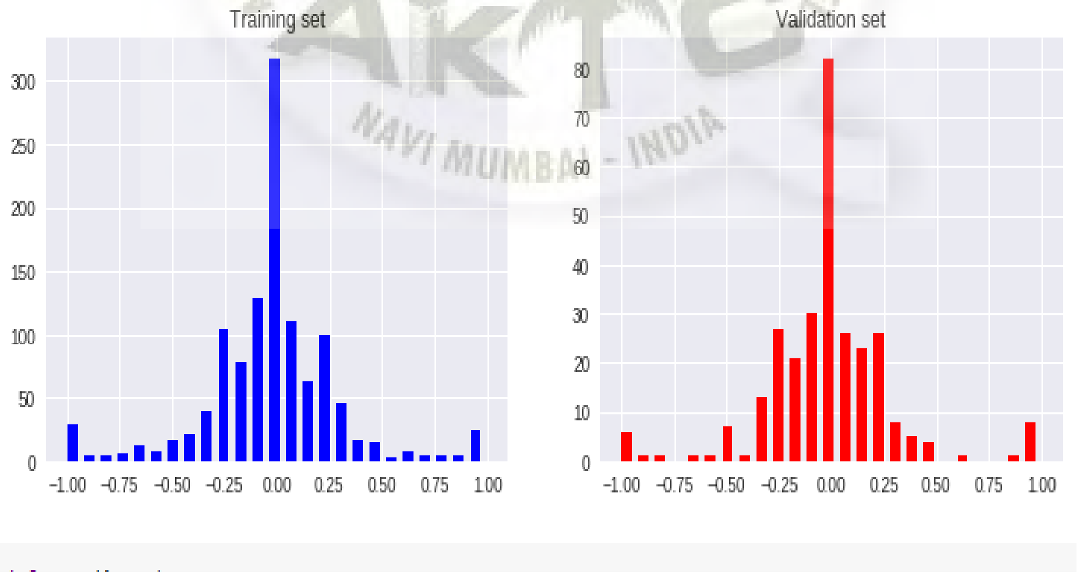


Figure 5.5: Train-Test Split

Before training the model we need to split the data into two different parts the training data and the testing data the following graph show the train-test split graph.

We can see that the balance in both the graphs are almost equal so we know that the split is good. the split is done in a ratio of 1:4 or 1:5 it depends.

Chapter 6

Image Augmentation

Once the data set is created we have lot of images to work with, these images make a pretty good model and it works for a some part of the track but we need a bigger dataset to increase the accuracy even more and avoid the car getting out of the track even due to slight misalignment so we need Image augmentation techniques to increase our data set diversity even more. There are four types of image augmentation techniques that we used in our project, They are as follows

6.1 IMAGE PANNING

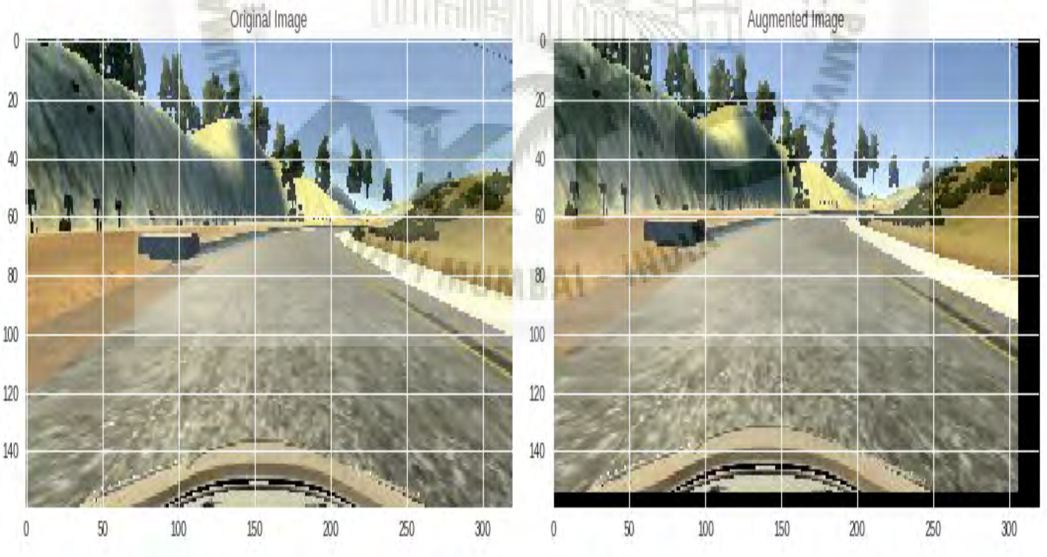
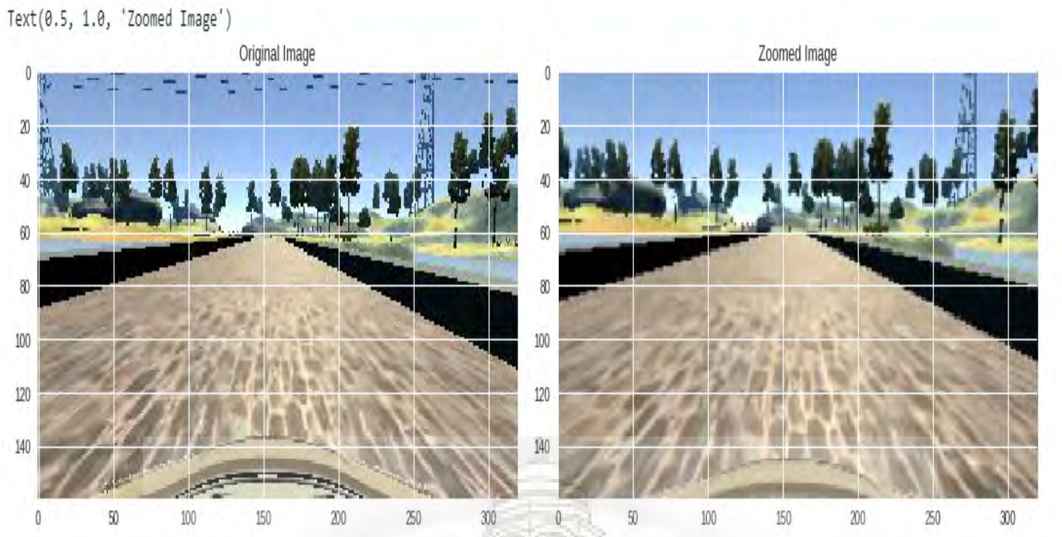


Figure 6.1: Image Panning

Image panning is a technique in which the images are either shifted left, right, up, down or combination of two or more of the above types of panning

6.2 IMAGE ZOOMIMG

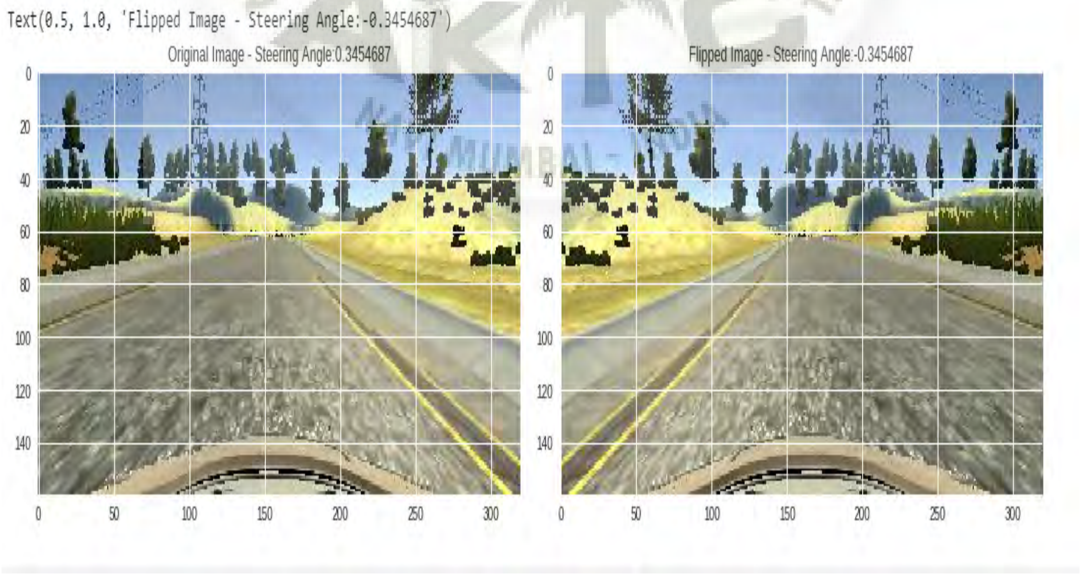


```
Text(0.5, 1.0, 'Zoomed Image')
```

Figure 6.2: Image Zooming

Image Zooming is a technique in which the images are zoomed in or zoomed out according to the specification of the user, And the amount of zoom in or zoom out is also specified

6.3 IMAGE FLIPPING



```
Text(0.5, 1.0, 'Flipped Image - Steering Angle:-0.3454687')
```

Original Image - Steering Angle:0.3454687
Flipped Image - Steering Angle:-0.3454687

Figure 6.3: Image Flipping

Image Flipping is a technique in which the images are flipped either horizontal or vertical and the steering angle is also reversed in this case for example if the original image has the steering angle of +0.022 then its flipped image will have steering angle of -0.022

6.4 IMAGE BRIGHTNESS MANIPULATION

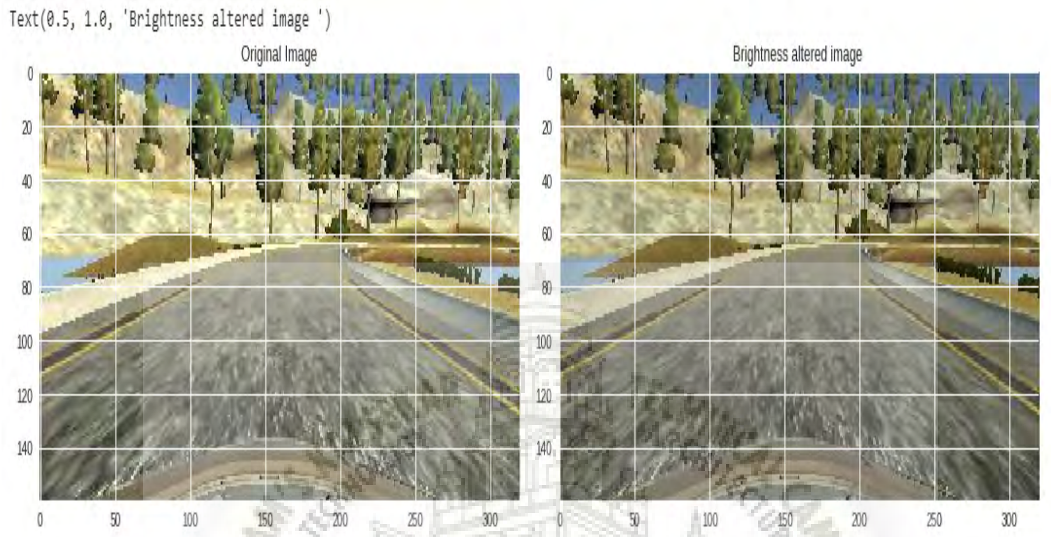


Figure 6.4: Image Brightness Manipulation

Image brightness manipulation is a technique in which the image brightness is manipulated we need this technique because not every time the real road will have same lighting conditions so the brightness of the images are either increased of decreased

Chapter 7

System Design

7.1 BLOCK DIAGRAM

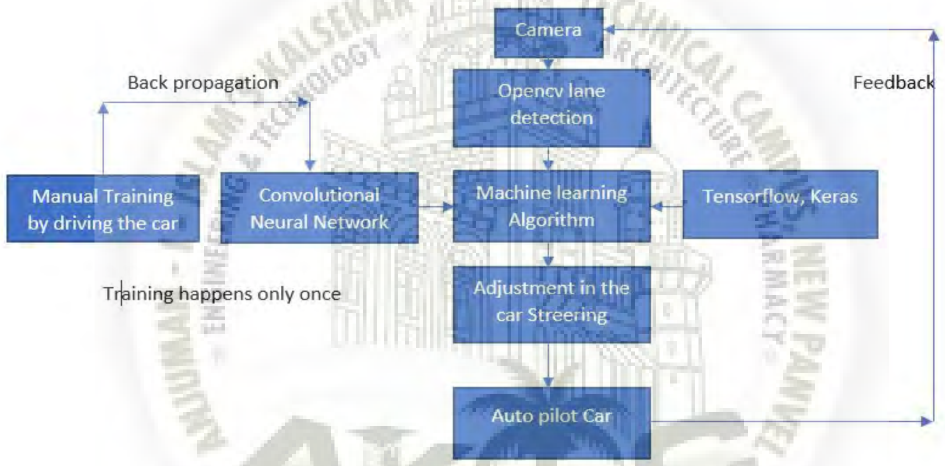

Camera
Opencv lane detection
Machine learning Algorithm
Tensorflow, Keras
Adjustment in the car Streering
Auto pilot Car
Feedback
Back propagation
Manual Training by driving the car
Convolutional Neural Network
Training happens only once

Figure 7.1: Block diagram of the system

The Above block diagram shows the basic block diagram of our system where in we take the input from the camera, The image from the camera is the live from the camera mounted on the bonnet of the car so the image will be of the road lane in front of the car in which the is being driven. The image is processed using Open cv lane detection algorithm and the output of the detected lane is fed as input to the Machine learning algorithm which will receive another input from the convolutional neural network.

The convolutional neural network is designed to keep the car inside the desired lane. The convolutional neural network is trained using back propagation technique in which we drive the car multiple times through different tracks and keep recording the video and updating the data set.

The Machine learning algorithm after receiving all the required inputs will the accurately calculates the position of the lanes and the degree to which we need to steer the car the output is corrected after each training cycle and we get the autopilot control.

7.1.1 LANE DETECTION

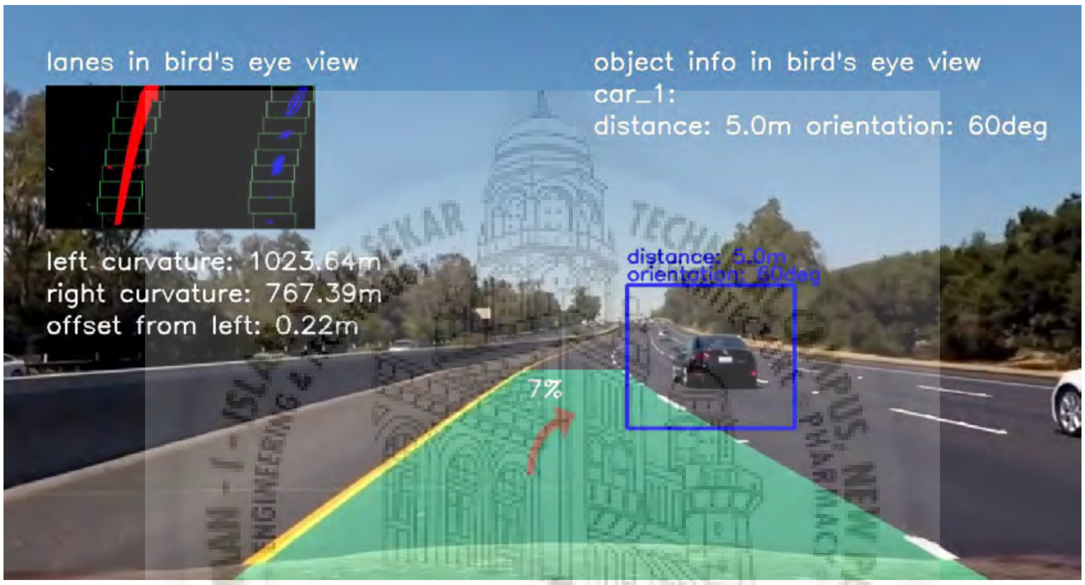


Figure 7.2: Lane detection

The above figure show how the lane markings are seen by a computer vision the openCV tool is used along with python language to read the frames of a live video feed and then perform some operations like gaussian blur, edge detection and hough transform to get the lane markings accurately.

7.1.2 STEERING MECHANISM

drive 1.png

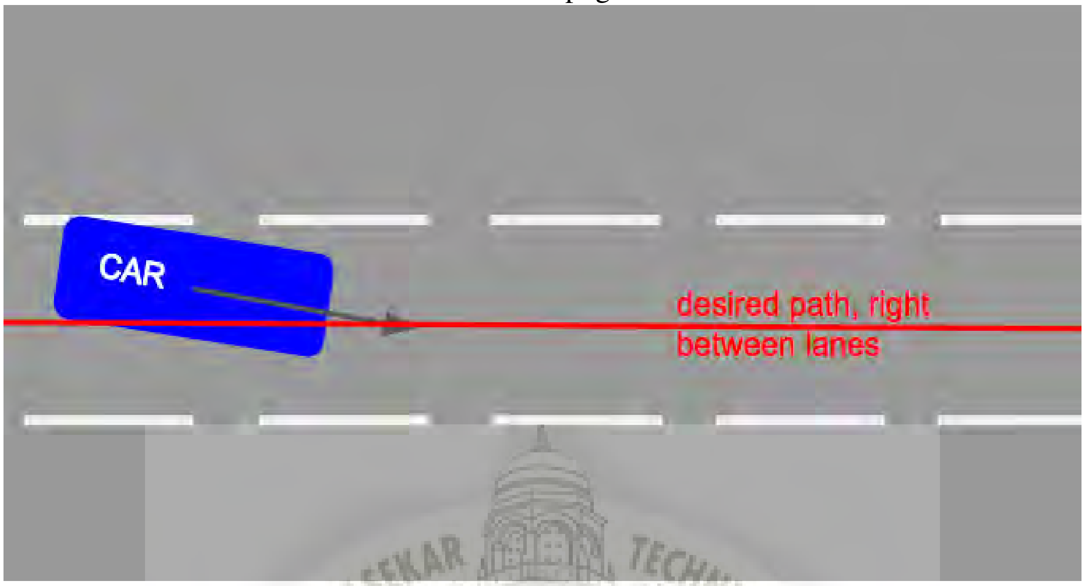


Figure 7.3: Driving assist

The above figure shows how a car is assisted to drive on its own after the model is trained when the car starts getting out of the lanes the driving assist starts.

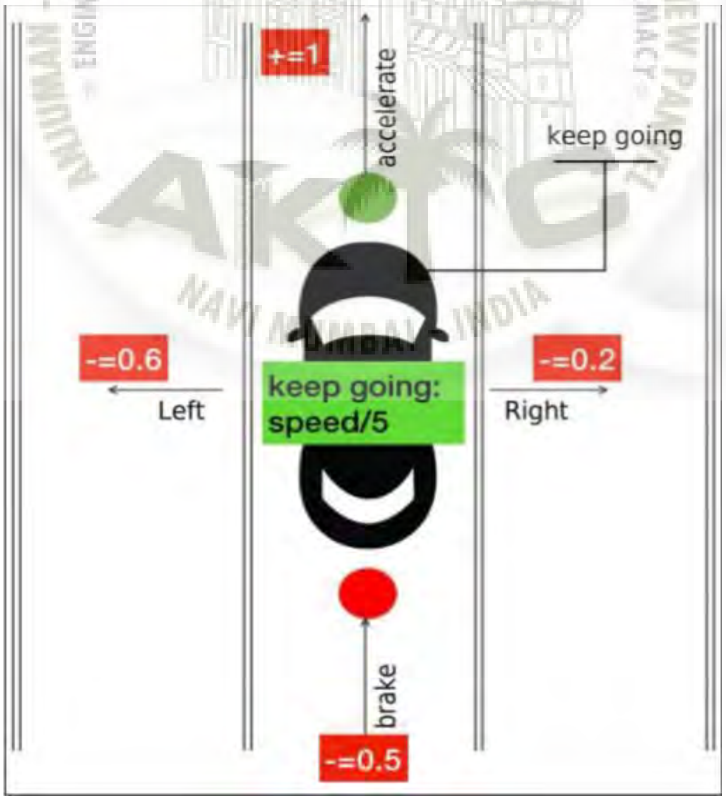


Figure 7.4: Driving assist

Chapter 8

Results

8.1 OUTPUTS

8.1.1 DETECTED LANES

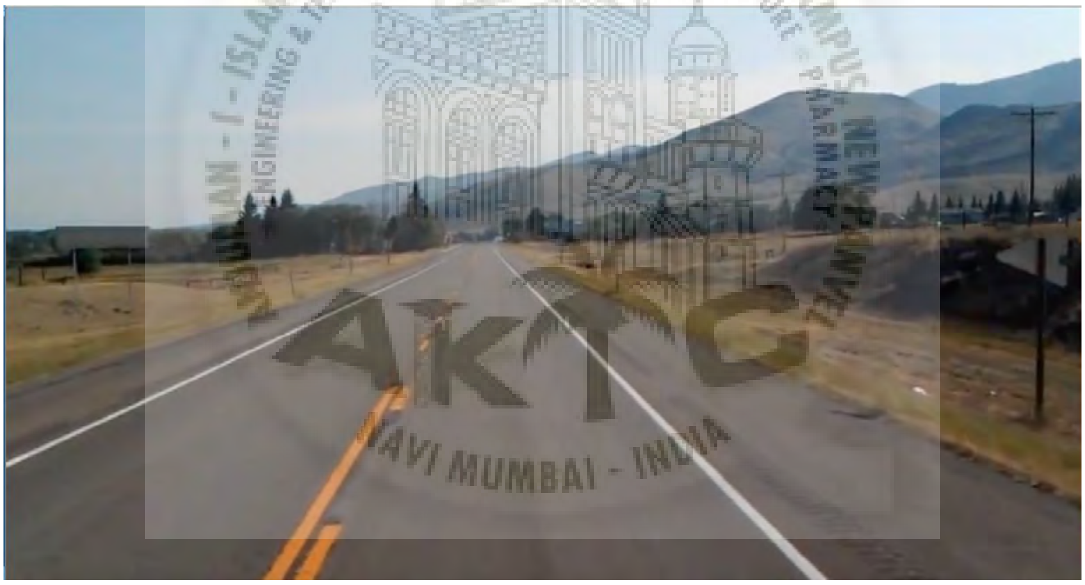


Figure 8.1: The input image

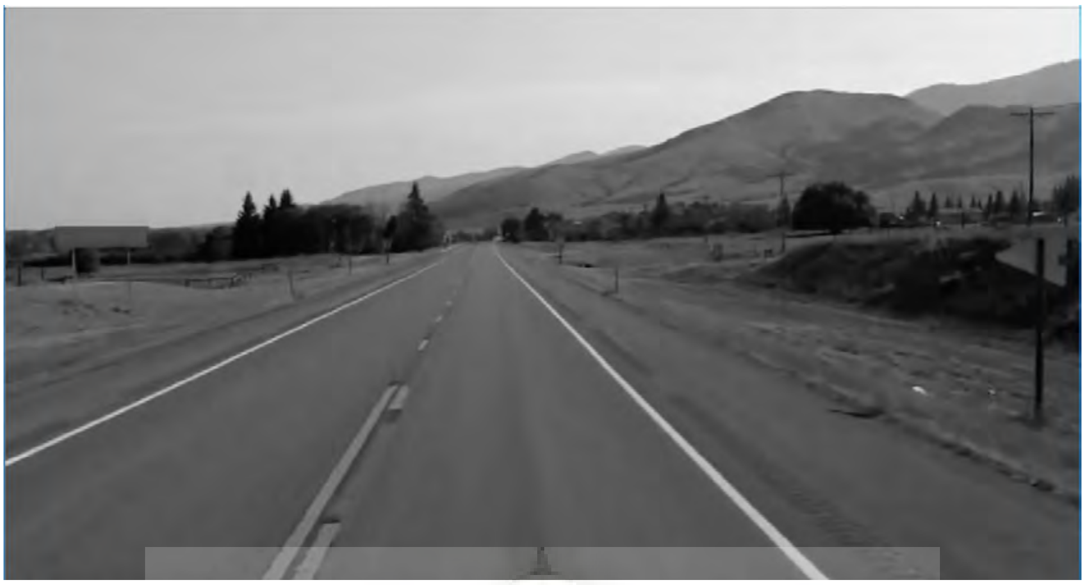


Figure 8.2: The grayscaled image

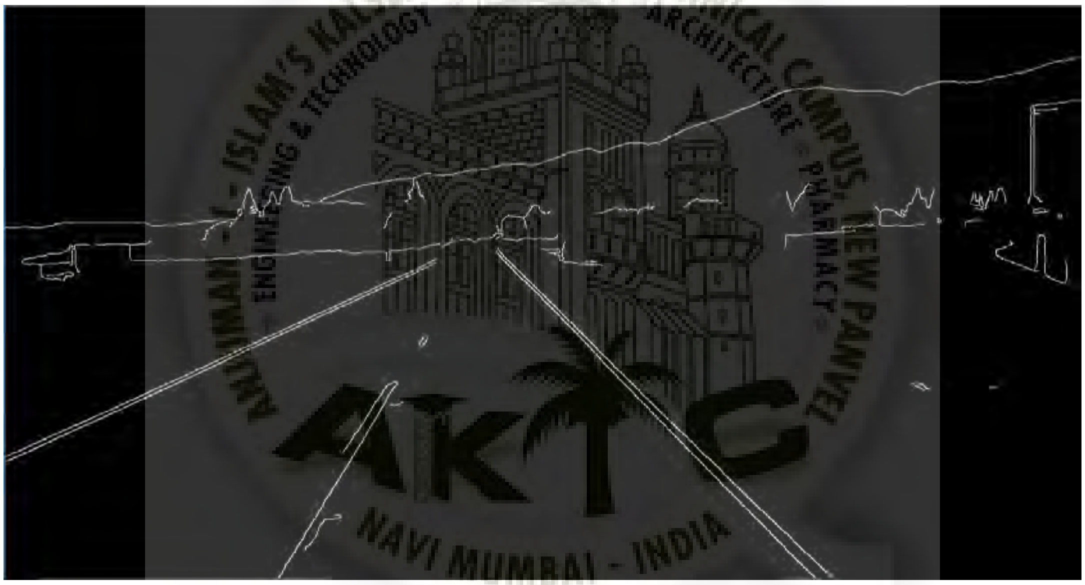


Figure 8.3: The edge detected image

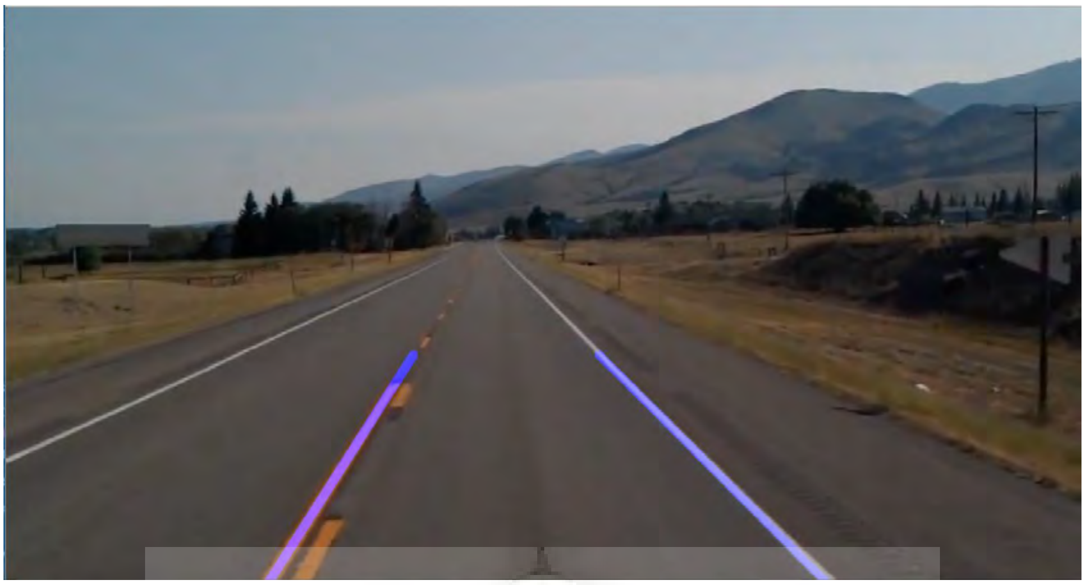


Figure 8.4: The lane detected image

8.1.2 THE TRAINING OUTPUT

The following figure shows the output of the autonomous car on the Traininng track on which it was train and then tested manually

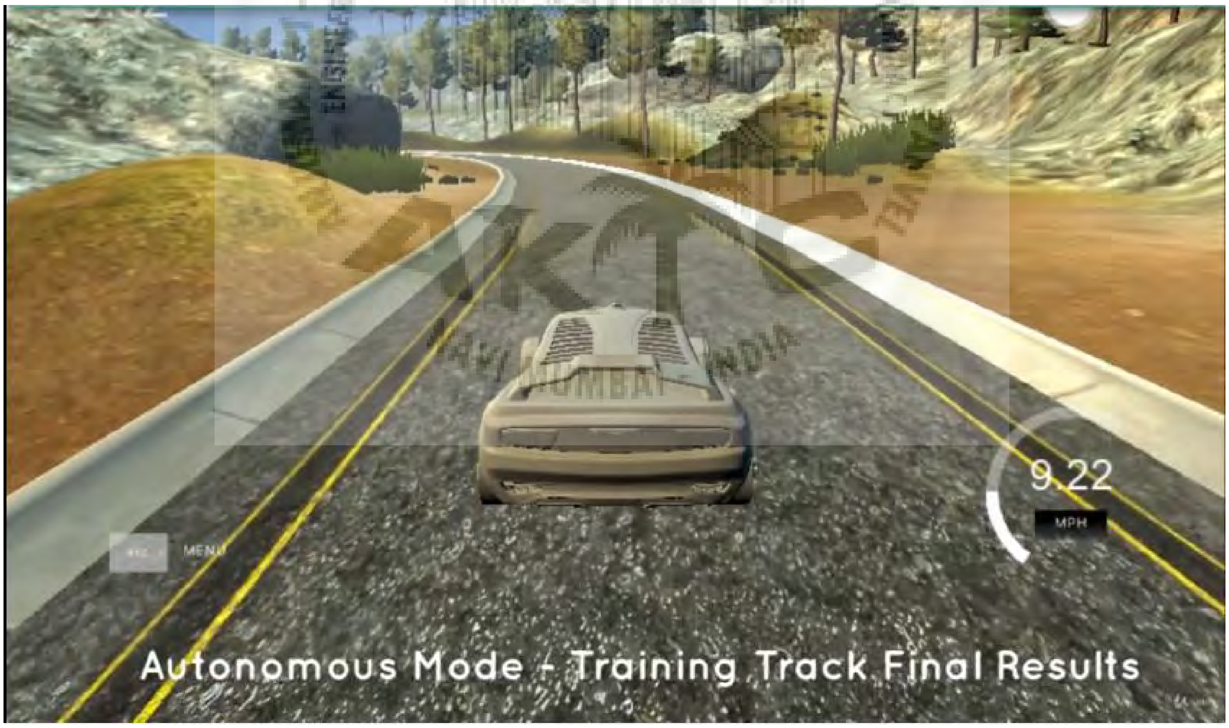


Figure 8.5: The training track

8.1.3 THE TESTING OUTPUT

The following image is taking from the output obtained after we test our algorithm on a test file or video

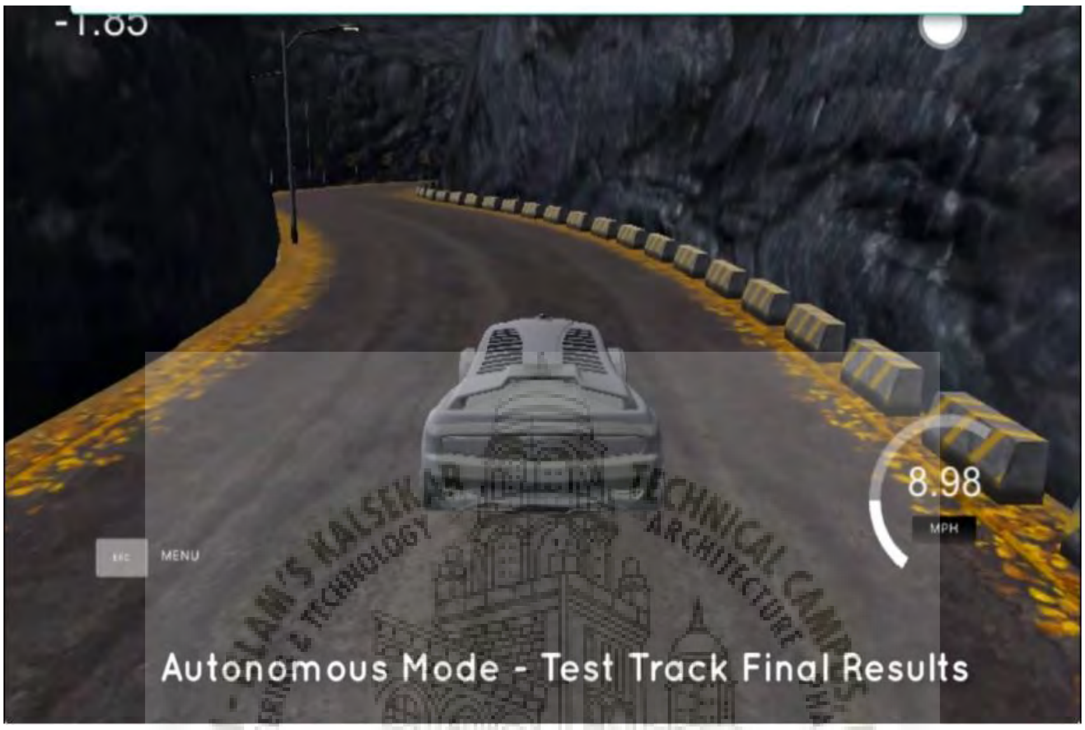
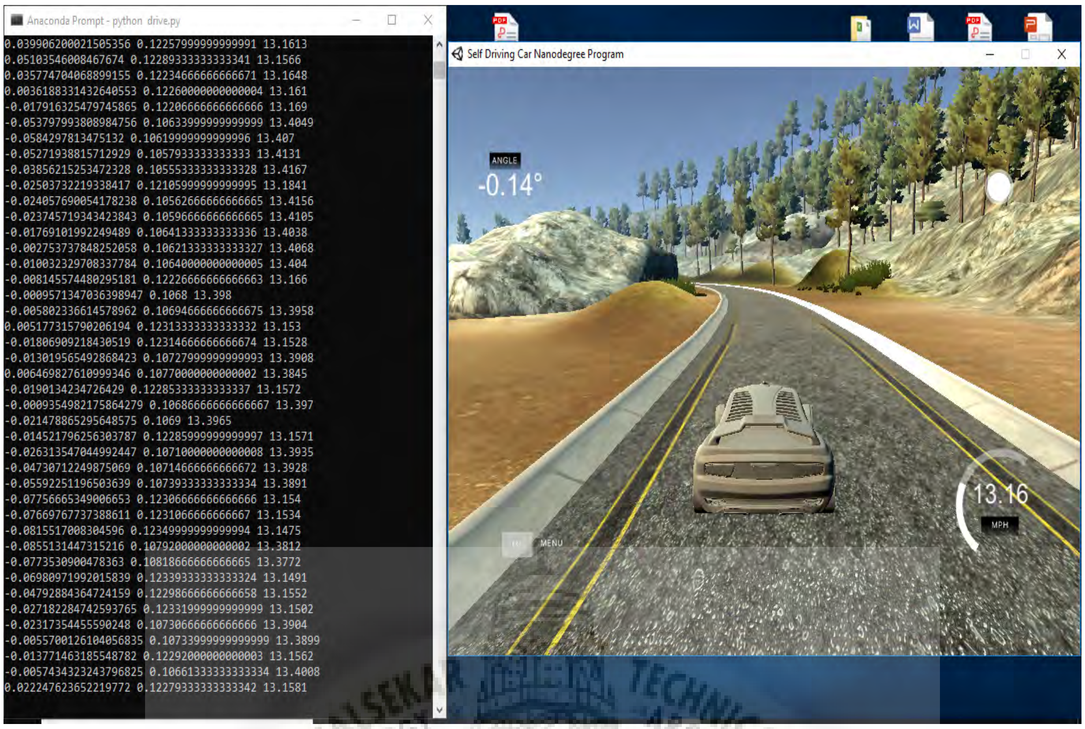


Figure 8.6: The testing track

8.1.4 THE FINAL OUTPUT

The following image is the actual output we got after we trained the model and we ran it on our simulator with the help of socket and flask programming by creating a client and server bu directional data flow


Anaconda Prompt - python drive.py
Self Driving Car Nanodegree Program
ANGLE
-0.14°
13.16
MPH
MENU

```
0.039906200021505356 0.12257999999999991 13.1613
0.05103546008467674 0.12289333333333341 13.1566
0.03577470406889155 0.12234666666666671 13.1648
0.003618833143264053 0.12260000000000004 13.161
-0.017916325479745865 0.12206666666666666 13.169
-0.053797993808984756 0.10633999999999999 13.4049
-0.0584297813475132 0.10619999999999996 13.407
-0.05271938815712929 0.10579333333333333 13.4131
-0.03856215253472328 0.10555333333333328 13.4167
-0.02503732219338417 0.12105999999999995 13.1841
-0.024057690054178238 0.10562666666666665 13.4156
-0.023745719343423843 0.10596666666666665 13.4105
-0.01769101992249489 0.10641333333333336 13.4038
-0.002753737848252058 0.10621333333333327 13.4068
-0.010032329708337784 0.10640000000000005 13.404
-0.008145574480295181 0.12226666666666663 13.166
-0.0009571347036398947 0.1068 13.398
-0.005802336614578962 0.10694666666666675 13.3958
0.005177315790206194 0.12313333333333332 13.153
-0.01806909218430519 0.12314666666666674 13.1528
-0.013019565492868423 0.10727999999999993 13.3908
0.006469827610999346 0.10770000000000002 13.3845
-0.0190134234726429 0.12285333333333337 13.1572
-0.0009354982175864279 0.10686666666666667 13.397
-0.021478865295648575 0.1069 13.3965
-0.014521796256303787 0.12285999999999997 13.1571
-0.026313547044992447 0.10710000000000008 13.3935
-0.04730712249875069 0.10714666666666672 13.3928
-0.05592251196503639 0.10739333333333334 13.3891
-0.07756665349006653 0.12306666666666666 13.154
-0.07669767737388611 0.1231066666666667 13.1534
-0.0815517008304596 0.12349999999999994 13.1475
-0.0855131447315216 0.10792000000000002 13.3812
-0.0773530900478363 0.10818666666666665 13.3772
-0.06980971992015839 0.12339333333333324 13.1491
-0.04792884364724159 0.12298666666666658 13.1552
-0.027182284742593765 0.12331999999999999 13.1502
-0.02317354455590248 0.10730666666666666 13.3904
-0.0055700126104056835 0.10733999999999999 13.3899
-0.013771463185548782 0.12292000000000003 13.1562
-0.0057434323243796825 0.10661333333333334 13.4008
0.022247623652219772 0.12279333333333342 13.1581
```

Figure 8.7: The final output

The command prompt display the output steering angle, the trottle and the speed of the car every time the model updated it

Chapter 9

Conclusion and Future Scope

9.1 CONCULSION

After implementation of this project we conclude that the 'Autopilot Vehicles' are the future Which can be a useful resource if used with care and precautions are taken while making the algorithm and adequate amount of training on different environments are done with care. We are done till the part of lane detection and street sign detection and are planning to do the total self driven cars

9.2 FUTURE SCOPE

According to us the future scope might be

- Implementing our project on hardware.
- Adding LiDAR and proximity sensors.

References

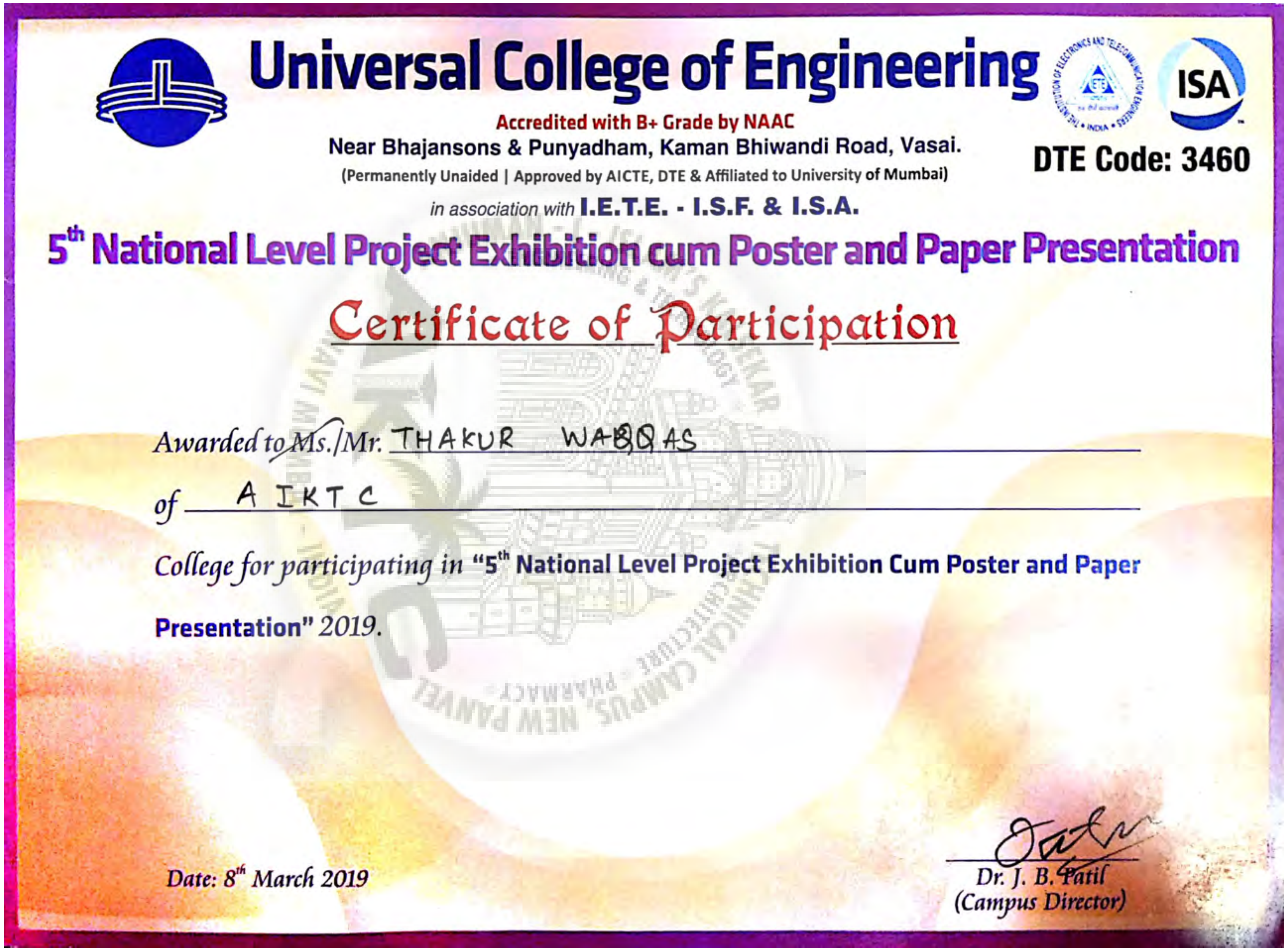
[1] *Implementation of Vehicle Detection Algorithm for Self-Driving Car on Toll Road Cipularang using Python Language*; Mochamad Vicky Ghani Aziz, Hilwadi Hindersah, Ary Setijadi Prihatmanto

[2] https://en.wikipedia.org/wiki/Computer_vision

[3] https://towardsdatascience.com

[4] http://www.bmva.org/visionoverview

[5] https://skymind.ai/wiki/neural-network



Universal College of Engineering

Accredited with B+ Grade by NAAC

Near Bhajansons & Punyadham, Kaman Bhiwandi Road, Vasai.

(Permanently Unaided | Approved by AICTE, DTE & Affiliated to University of Mumbai)

DTE Code: 3460

in association with **I.E.T.E. - I.S.F. & I.S.A.**

5th National Level Project Exhibition cum Poster and Paper Presentation

Certificate of Participation

Awarded to ~~Ms.~~/Mr. THAKUR WAQQAS

of AIKTC

College for participating in **"5th National Level Project Exhibition Cum Poster and Paper Presentation"** *2019.*

Date: 8th March 2019

[illegible]
Dr. J. B. Patil
(Campus Director)

Figure 9.1: Certification