

FINAL PROJECT REPORT

EEE F266

ON

**Designing and Building An
Autonomous Vehicle Using Neural
Networks / Computer Vision**

BY

**Aamer Abdul Rahman
2017AAPS0217U
ECE**

Under the guidance of Dr. Anand Kumar

AT



**BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E**

BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

Duration: Ongoing

Date of Start: 27-AUG-2019

Date of Submission: 14-DEC-2019

Title of the Project: Designing and Building An Autonomous Vehicle Using Neural Networks / Computer Vision

Student Name: Aamer Abdul Rahman

Student ID: 2017AAPS0217U

Discipline of Student: B.S. (Hons.) in Electronics and Communications Engineering

Name of the Faculty: Dr. Anand Kumar

Key Words: Neural networks, Convolutional nets, Deep learning

Project Areas: Artificial intelligence, Convolutional neural networks, Machine learning

Abstract: The aim is to design and simulate various functions that are required for a vehicle to be autonomous. Image processing tools are utilized to detect and isolate lanes from images so that the ego vehicle may perform appropriate maneuvers. R-CNN are used to detect vehicles and pedestrians in real world simulations via MATLAB. Through the use of fuzzy logic, object avoidance is implemented coded via MATLAB is simulated in the simulation software VREP. Finally, a mobile robot controlled via an Arduino is programmed to detect incoming objects via proximity sensors and avoids collision.

Signature of Faculty

Signature of the Student

Date:

Date:

ACKNOWLEDGEMENTS

I would like to express my deepest sense of gratitude and thanks, to my Supervisor **Dr. Anand Kumar**, Professor of Electrical and Electronics Engineering Department, BITS Pilani, Dubai campus, United Arab Emirates, for her valuable guidance and encouragement during the course of this Project.

I am grateful to our Director, **Prof. Dr. R.N. Saha**, for his motivation, encouragement and support to pursue my dissertation.

Above all, I thank the Lord for giving me the strength to carry out this work to the best of my abilities.

Signature of the Student

Date:

CONTENTS

Abstract.....	1
Acknowledgement.....	2
Table of contents.....	3
List of Figures.....	4

Chapter 1 INTRODUCTION

1.1 Overview.....	5
1.2 Autonomous Vehicles.....	5
1.2.1 Levels of Autonomy.....	6

Chapter 2 PI-CAR KIT

2.1 Overview.....	7
2.2 Components.....	7
2.2.1 Raspberyy-Pi.....	8
2.2.2 Robot HATS.....	9
2.2.3 Motor drivers.....	10
2.2.3.1 PCA9865 Motor Driver.....	10
2.2.3.2 TB6612.....	11
2.2.4 SF0180 Servo	11
2.2.5 DC Gear Motor	12
2.2.6 USB Accelerator.....	12

Chapter 3 CONVOLUTIONAL NEURAL NETWORKS

3.1 Introduction.....	13
3.2 Architecture.....	14

Chapter 4 LANE DETECTION

4.1 Introduction.....	15
4.2 Implementation.....	15
4.2.1 Isolating colour of lane.....	15
4.2.2 Detecting edges of lane lines.....	17

Chapter 5 TRAINING A NETWORK TO DETECT VEHICLES

5.1 Overview.....	21
5.2 CNN vs R-CNN.....	21
5.3 Training the R-CNN.....	22

Chapter 6 TRACKING VEHICLES AND PEDESTRIANS IN VIDEO

6.1 Overview.....	23
6.2 Workflow.....	23
6.3 Supporting Functions.....	23
6.4 Tracking Vehicles.....	24
6.5 Tracking Pedstrians.....	25

Chapter 7 COLLISION AVOIDANCE WITH FUZZY LOGIC

7.1 Fuzzy Logic.....	26
7.2 Fuzzy Logic Controller.....	26
7.3 Mamdani and Sugueno Model.....	26
7.4 Methodology.....	27
7.5 Result.....	28

Chapter 8 Conclusion and Future Scope.....29

References.....	30
-----------------	----

LIST OF FIGURES

1. Figure 1 Sunfounder PiCar Kit.....	7
2. Figure 2 Circuit diagram.....	8
3. Figure 3 Raspberry Pi.....	9
4. Figure 4 Arduino Uno.....	9
5. Figure 5 Robot HATS.....	9
6. Figure 6 PCA9865.....	10
7. Figure 7 TB6612.....	11
8. Figure 8 Servo motor.....	11
9. Figure 9 Coral USB Accelerator.....	12

10. Figure 10 Regular neural network vs convolutional neural network.....	13
11. Figure 11 Architecture of CNN.....	14
12. Figure 12 Lane detection system.....	15
13. Figure 13 RGB colour space.....	15
14. Figure 14 HSV colour space.....	16
15. Figure 15 RGB to HSV.....	16
16. Figure 16 Blue colour masked.....	17
17. Figure 17 Isolation of region of interest.....	18
18. Figure 18 Representation of line.....	19
19. Figure 19 Lines drawn on lane.....	19
20. Figure 20 Bounding box detected by RCNN.....	23
21. Figure 21 Accuracy Rating.....	23
22. Figure 22 Bounding boxes detected by RCNN in Video.....	25
23. Figure 23 Bounding boxes around pedestrians detected by RCNN.....	26
24. Figure 24 VREP Scene.....	28
25. Figure 25 Membership Functions.....	29

Chapter 1

INTRODUCTION

1.1 Overview

Operating a vehicle has always been a relatively simple task for the experienced driver. An individual uses his senses to observe and measure their surroundings and perform the appropriate action in response to the immediate situation such as pressing the brake pedal or accelerator and turning the steering wheel. However, when it came to designing a system that can autonomously operate a vehicle, it becomes complicated. The human brain can extrapolate many more considerations when facing a problem such as this and can obtain a lot of data from as little as the three or four variables. Neural networks are utilized to try to mimic brain function and when trained properly, they are able to make sense of the large amounts of data collected and deliver the desired output. The objective is to create a system, that makes use of neural networks, computer vision and image processing, that is capable of autonomously driving the car to the destination tackling the different criteria's such as lane keeping, pedestrian detection and object tracking.

1.2 Autonomous Vehicles

Autonomous vehicles should be able to operate and guide itself without the need for people to intervene. Campbell et al. describes modern autonomous vehicles as vehicles that can sense their local environment, classify different kinds of objects that they detect, can interpret sensory information to identify appropriate navigation paths whilst obeying transportation rules. Such vehicles combine data collected from sensors such as LIDAR, camera, radar and sonar and then software is used to make sense of the data received from the environment to operate, navigate, and drive the vehicle. Automated driving systems have been in the works since the 1920's with first car in the world that was radio controlled. Since then notable advancements have been made wherein vehicles are able to give appropriate responses to real world situations. Many cars sold on the market are already equipped with mechanisms that bring the vehicle to certain levels of autonomy. Vehicles with partially autonomous systems may require the driver to intervene if the system is not able to process certain environments or situations. On the other hand, fully autonomous vehicles may not even require steering wheels [1].

1.2.1 Levels of Autonomy

Six levels of autonomy have been outlined by The Society for Automotive Engineers to be used by automakers, supplier and policymakers to determine a system's sophistication [2].

Level 0 – Zero Automation: The driver completely in control of the vehicle. No autonomous driver aids have been implemented.

Level 1 – Assistance Systems: The driver is still mostly in charge of the vehicle. However, the driver may call upon technology like adaptive cruise control for support. This system determines the distance from the car in front using lasers or radars and adjusts the throttle accordingly to maintain proper distance. A level 1 autonomous vehicle can control either acceleration/braking or steering, but not both simultaneously.

Level 2 – Partial Automation: Level 2 autonomous vehicles are able to control steering and acceleration/braking at the same time for short periods without driver input. The cars can stay in lane and are able react to warning systems. However, the driver still has to be driving and alert.

Level 3 – Conditional Automation: At this level, the autonomous system is able to handle most of the driving situations. However, when it encounters a situation it can't navigate through, the control is transferred back to the driver.

Level 4 – High Automation: There is a significant leap in functionality from level 3 to level 4. Level 4 vehicles can monitor all roadway activities and perform all critical driving functions for the duration of the trip while operating in operational design domain. However, the driver needs to be aware while travelling and may have to alternate with the system in certain driving conditions.

Level 5 – Full Automation: Level 5 autonomous system functioning is equivalent to a car being controlled by a driver. The car should be capable of navigating through any roads in any conditions as a human driver could operate.

Chapter 2

Pi-Car Kit

2.1 Overview

To accomplish the task of building a miniature autonomous vehicle, a programmable car kit that will be able to take in video feed and be compatible with machine learning algorithms is needed. The PiCar-V kit being manufactured by Sunfounder was chosen as it met the required criteria and provided a user friendly interface to work with. The PiCar-V is based on the Raspberry Pi or Arduino that can be connected to various sensors such as proximity sensors or a wide angle camera and is an open source robot learning kit [3].

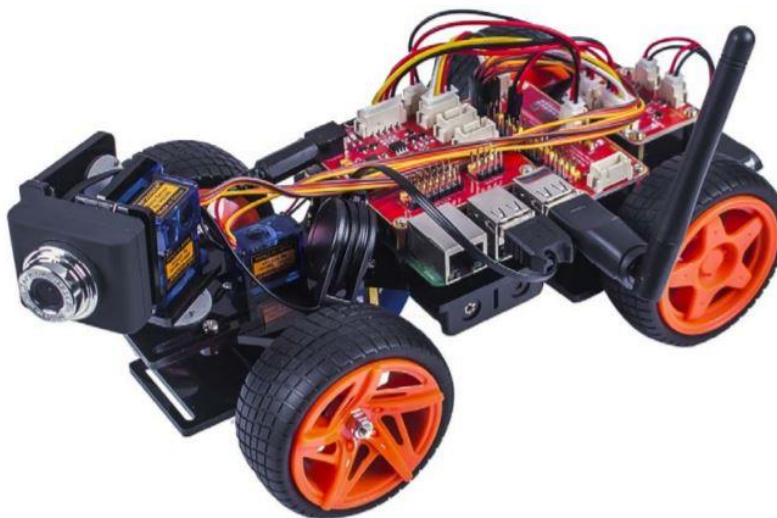


Figure 1 Sunfounder PiCar Kit

2.2 Components

The kit consists of several PCB's that are interconnected with the Raspberry Pi and the motors. There are three servo motors and two DC motors. Each of the components are discussed in detail below.

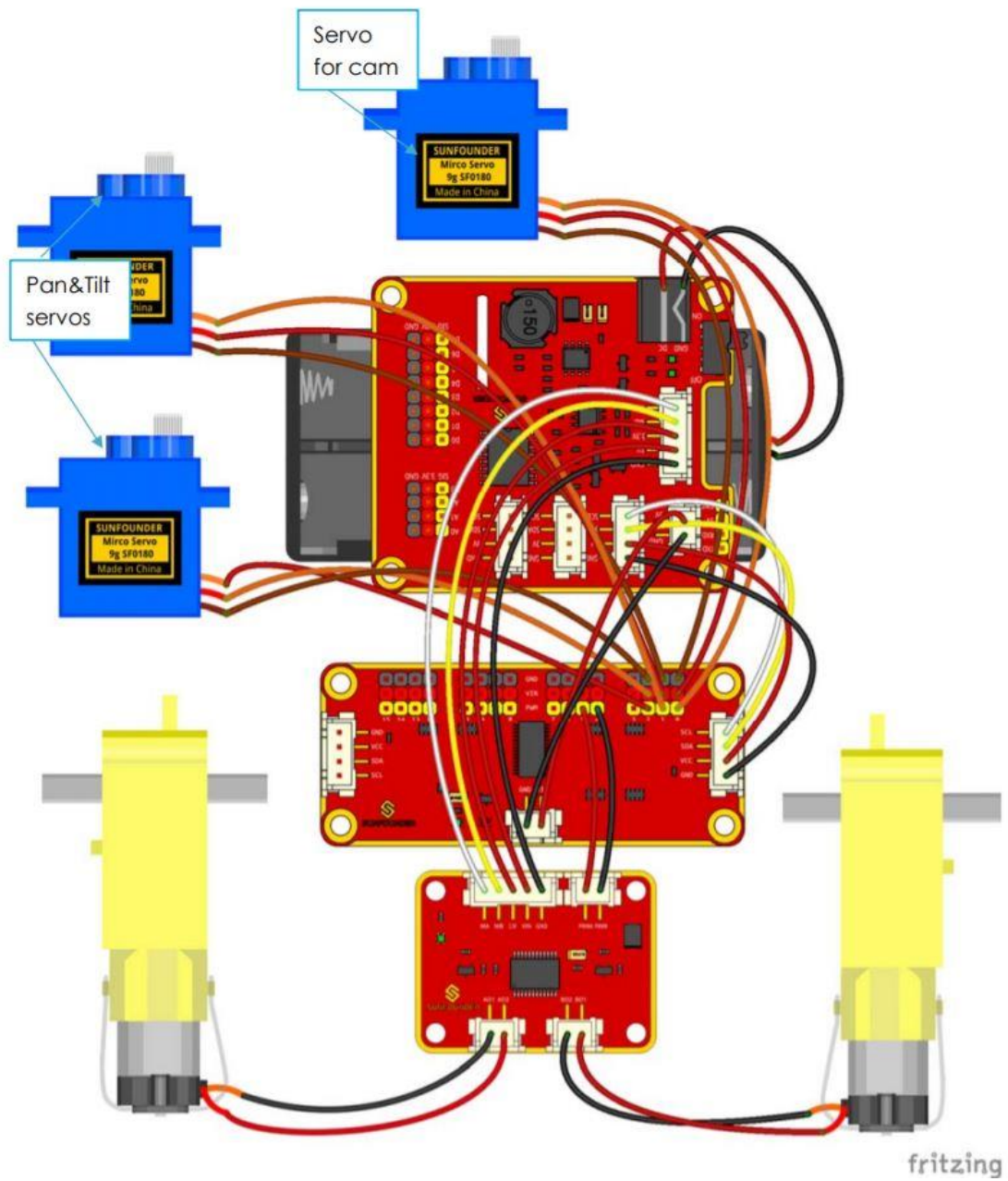


Figure 2 Circuit diagram

2.2.1 Raspberry Pi

The Raspberry Pi is a microprocessor that can be plugged into a monitor or screen and is controlled via a standard keyboard and mouse. It became popular thanks to its low cost, small size and versatility. It can be programmed via python or scratch. It functions in the same way as a traditional computer in the sense that it runs on an operating system wherein we can perform functions such as playing videos, connecting to and browsing the internet, playing games, editing documents and spread sheets and so on. It operates on a downsized version of Linux and consists of a set of GPIO, that stand for general purpose input and output pins that enable the control of various electrical components to create and operate physical computing. The Raspberry PI is an open source system that traditionally runs on Raspbian, a variant of the Linux operating system [4].



Figure 3 Raspberry Pi

Its processor can run up to speeds of 1.4 GHz and has on-board memory (RAM) of 1 GB. It does not come with storage to hold the operating system and instead consists of a micro SD slot where the user can boot Linux from. The board consists of 4 USB ports. HDMI and composite video are supported for video output along with a 3.5 mm jack for audio output. Lower-level output is provided by a number of GPIO pins that support various protocols such as I2C. This particular model has an 8P8C Ethernet port and on board Wi-Fi [4].

2.2.2 Arduino UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion shields and other circuits. The board has 14 digital I/O pins (6 capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE, via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo [16].

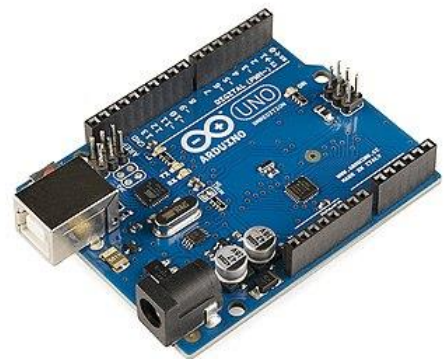


Figure 4 Arduino

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer. While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB to serial driver chip. Instead, it uses the Atmega16U2 as a USB to serial converter.

2.2.2 Robot HATS

The Robot HATS that is designed to be compatible with a Raspberry PI. The power sources is connected to this component. It then supplies power to the PI via the GPIO ports. The circuit is designed with an ideal diode that enables the Raspberry PI to be controlled by either a USB cable or from the DC power source without damaging the TF card caused by the batteries running out of power. The component operates via I2C communication and the PCF8591 is used as the ADC chip [5].

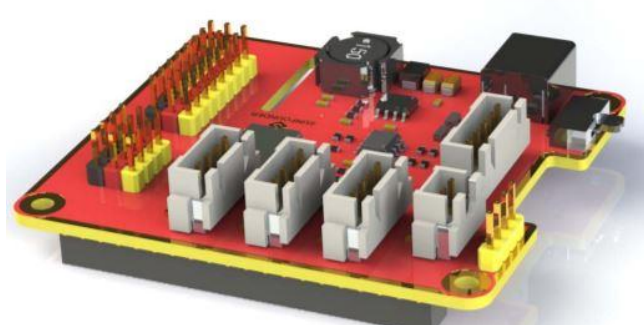


Figure 5 Robot HATS

- Digital ports for digital sensors where signal voltage is equal to 3.3V and VCC voltage is equal to 3.3V.
- Analog ports for 8 bit ADC sensors with reference and VCC voltage equal to 3.3V
- 3.3V I2C bus communication ports
- 5V power supply ports to PWM driver.
- UART port: 4-wire UART port, 5V VCC
- TB6612 motor control ports: includes 3.3V for the TB6612 chip, 5V for motors, and direction control of motors MA and MB; working with SunFounder TB6612 Motor driver.
- DC power por with input voltage from 8.4 to 7.4V and limited operating voltage from 12V to 6V.

2.2.3 Motor Drivers

Motor Driver boards act as current amplifiers. The motor and the microcontroller are bridged by driver circuits. IC's integrated with discrete components are used to design motor drivers. The signal input to the motor driver from the microcontroller is a low current signal. The motor driver bridges the two components by amplifying this current. The higher amplified current is then fed into the motor [5].

2.2.3.1 PCA9865 Motor Driver

The PCA9865 motor driver is a 12-bit I2C bus PWM driver that consists of 16 channels. It supports independent PWM output power and is easy to use 4-wire I2C port for connection in parallel, distinguished 3-colour ports for PWM output [5].

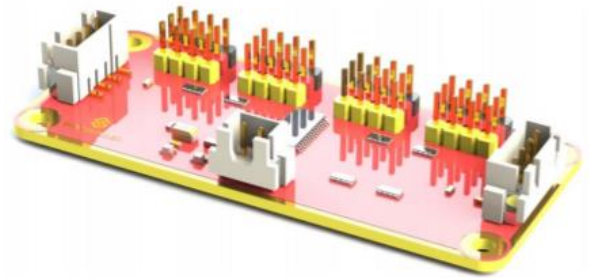


Figure 6 PCA9865

- Ports for PWM output: Directly connected to servo via the independent PWM output port
- Port for I2C bus channel that is compatible with 3.3 or 5V
- Power input for PWM limited to 12V

2.2.3.2 TB6612

The TB6612 uses a dual H-bridge motor controller that consists two input slots per H-bridge. The speed of the motor can be controlled with the PWM drivers available for each individual input. The logic connections run at 2.7-5V. This logic voltage is set apart from the motor voltage [5].

- Ports for power and control: Consists of pins that supply power to the chip and motor and pins that control the motors direction upon changing polarity.
- Ports for PWM control of motors: Input ports for PWM signals that control speed of motors.
- Port for motor output: Dual output pins for respective motors.

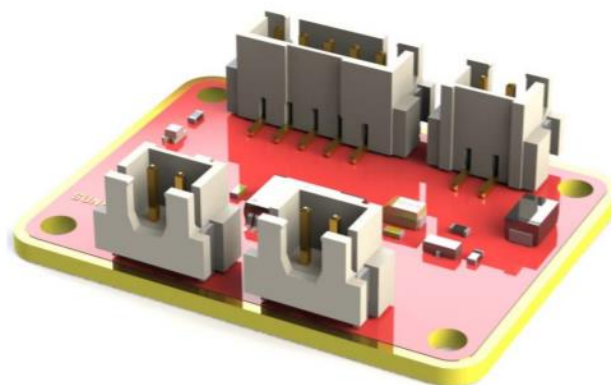


Figure 7 TB6612

2.2.4 SF0180 Servo

The SF0180 Servo is a three wire digital servo motor with a 180 degree field of rotation. It accepts PWM signals of upto 60Hz from the microcontroller or processor to which it is connected to [5].

Item	V = 4.8V	V = 6.0V
Operating speed (no load)	0.12 Sec/60°	0.09 Sec/60°
Running current (no load)	140 mA	200 mA
Stall torque (locked)	2.0 kg-cm	2.5 kg-cm
Stall current (locked)	520 mA	700 mA
Idle current (stop)	5 mA	5 mA



Figure 8 Servo

2.2.5 DC Gear Motor

It's a DC motor where the shaft is attached to a gear train the reduces speed and increases torque [5].

Motor	Model	F130SA-11200-38V
	Rated Voltage	4.5V-6V
	No-load Current	≤80mA
	No-load Speed	10000±10%
Gear Reducer	Gear Ratio	1:48
	Speed (no-load)	≈200rpm (≈180rpm in test)
	Current	≤120mA

2.2.6 USB Accelerator

The Edge TPU is a small ASIC designed by Google for high performance ML inferencing on low-end devices. It can easily execute the latest mobile vision models including MobileNet V2 at 100+ fps. The Edge TPU supports TensorFlow Lite easily. The first-generation Edge TPU can easily execute deep feed forward neural networks (DFF) including convolutional neural networks (CNN), which makes it the best choice for different vision-based ML applications [12].



Figure 9 Coral USB Accelerator

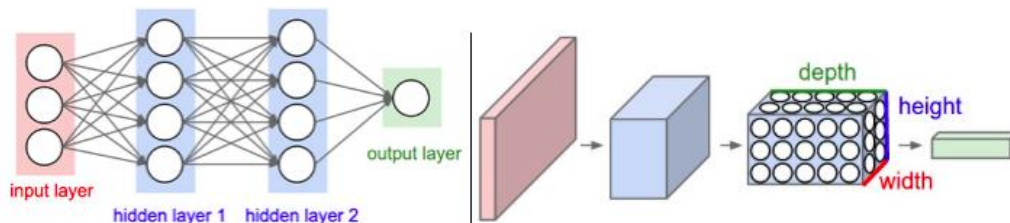
Chapter 3

Convolutional neural networks

3.1 Introduction

The most commonly used type of neural network in deep learning is the convolutional neural network. Convolution is a mathematical term that refers to linear operations between matrixes. This type of neural network consists of layers that are 3 dimensional, the dimensions being width, height and depth. Unlike simple neural networks, the neurons in each layer only connect to a small part of the succeeding layer. The final output is then reduced to a single vector of probability scores, organized along the depth dimension. CNNs also reduce upon the number of parameters as compared to ANN [7].

Figure 10 Regular neural network vs convolutional neural network



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

3.2 Architecture

CNNs are made up of two components:

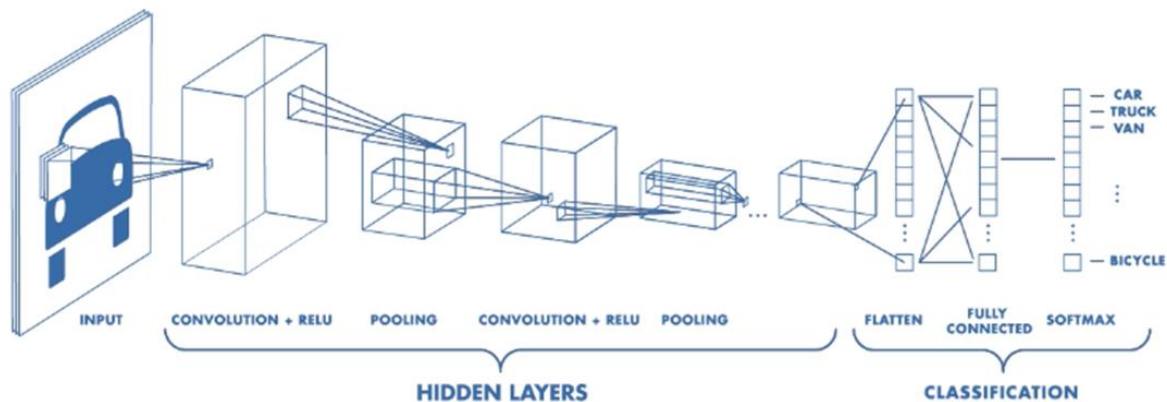
The Feature extraction component (Hidden layer)

In this stage, the CNN passes the data through stages of pooling and convolutional operations wherein the various features of the frames are detected. This is the part where the network is able to detect different features of the input. For example, if you had a picture of a person, it would be able to detect the person's face, arms and legs [8].

The Classification component

This component of the CNN consists of layers that are fully connected that functions as a classifier to recognize and classify the detected features into their respective classes. The image is assigned a probability of what the network predicts it is. This classification layer only accepts one dimensional input. Hence, the three dimensional data is converted into one dimension. [8]

Figure 11 Architecture of CNN



Chapter 4

Lane Detection

4.1 Introduction

Today, cars on the market that have two features onboard, namely, Adaptive Cruise Control (ACC) and some forms of Lane Keep Assist System (LKAS). Adaptive cruise control uses radar to detect and keep a safe distance with the car in front of it. Lane Keep Assist System is a relatively new feature, which uses a windshield mount camera to detect lane lines, and steers so that the car is in the middle of the lane.



Figure 12 Lane detection system

4.2 Implementation

A lane keep assist system has two components, namely, perception (lane detection) and Path/Motion Planning (steering). Lane detection's job is to turn a video of the road into the coordinates of the detected lane lines. One way to achieve this is via the computer vision package, OpenCV. In this implementation, the captured frames will undergo through the following image processing techniques [8].

4.2.1 Isolating colour of lane

The lanes will be marked tape of desired colour. As different parts of the tape may be lit with different light, different shades of the colour will be captured by the camera. Selecting these shades of colour in the RGB colour space is possible but will be difficult since all three parameters, i.e. red, green and blue, will change for different shades of one colour. It is extremely difficult to arbitrarily dictate how much of each primary colour composes it when looking at a particular colour.

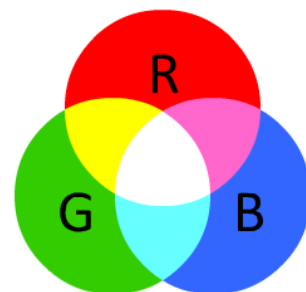


Figure 13 RGB colour space

In the HSV color space, the colours (hue or tint) are described in terms of their brightness value and shade (saturation or amount of gray). The HSV color wheel always consists of these components but sometimes appears as a cylinder or cone [9].

HUE

Hue is one of the components of the HSV colour space and indicates the colour part of the space. It is indicated by a number in between 0 and 360 degrees [9].

- Between 0 and 60 degrees: **Red** .
- Between 61 and 120 degrees: **Yellow**.
- Between 121-180 degrees: **Green**.
- Between 181-240 degrees: **Cyan**.
- Between 241-300 degrees: **Blue**.
- Between 301-360 degrees: **Magenta**.

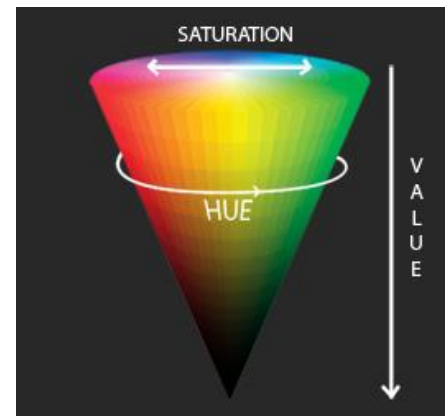


Figure 14 HSV colour space

SATURATION

The amount of gray in a particular colour is described by the Saturation. It ranges from 0 to 100 percent. A faded effect is produced upon reducing this component towards zero as more gray is introduced. In certain cases, the saturation may be ranged from 0-1 instead of 0-100, where 1 is the primary colour and 0 is gray. [9]

VALUE (OR BRIGHTNESS)

The intensity or brightness of the colour is described by value. Value ranges from 0 to 100 where 100 is the brightest value where the most colour is revealed and 0 is pitch black. [9]

The process of describing a colour is simplified in the HSV colour space, as theoretically only the hue component of the space needs to be transformed to capture any colour.

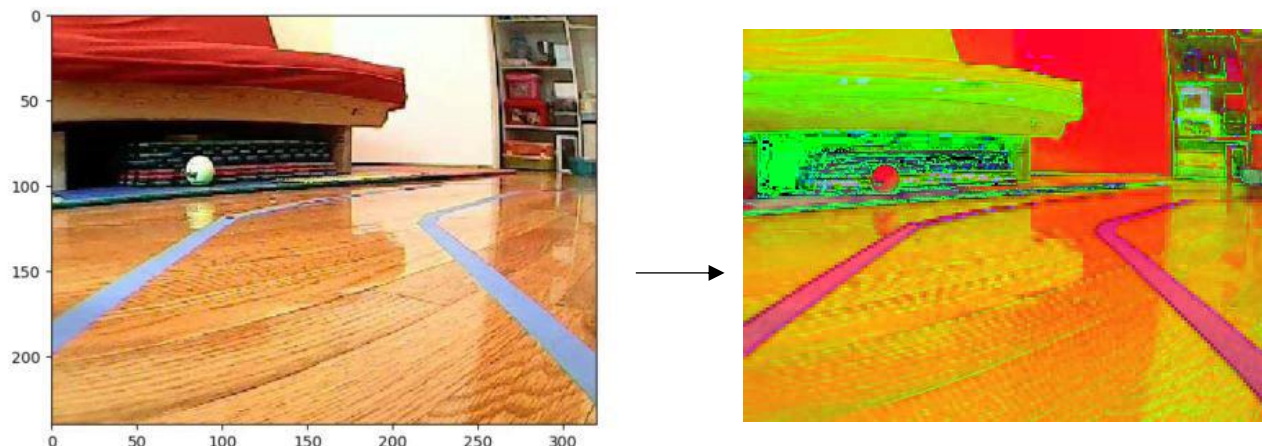


Figure 15 RGB to HSV

Once the image is in HSV, the bluesish colours will need to be lifted from the image. This is done by specifying a range of the color blue. In Hue color space, the blue color is in about 120–300 degrees range, on a 0–360 degrees scale. You can specify a tighter range for blue, say 180–300 degrees, but it doesn't matter too much. [9]



Figure 16 Blue colour masked

4.2.2 Detecting edges of lane lines

After masking, the next step is to detect edges in the blue mask, i.e. detecting along the blue masks. The Canny edge detection function is a powerful command that detects edges in an image. In the code below, the first parameter is the blue mask from the previous step. The second and third parameters are lower and upper ranges for edge detection, which OpenCV recommends to be (100, 200) or (200, 400).

The amount of data to be processed can be dramatically reduced upon using the Canny edge detection algorithm [10] which extracts useful structural elements from different objects in the frame. This method is popularly used in computer vision applications. Canny observed that even for diverse vision systems, the requirements for the application of edge detection has remained relatively similar. Therefore, edge detection

algorithms can be used in a wide range of solutions upon satisfying these requirements. The general requirements for edge detection are:

- The detection needs to catch as many edges from the frame accurately. That is, edges should be detected with low error rate.
- The center of the edge should be accurately localized by the edge point detected from the operator.
- Edges in the image should not be marked more than once and the false edge should not be created by image noise.

Canny used the calculus of variations to satisfy these requirements. This technique is used to find a function which optimizes the given functional. The sum of four exponential terms are used to describe the optimal function in Canny's detector, however, the optimal function can also be approximated upon finding the first derivative of a Gaussian function.

Canny's edge detection [10] is among the most popular edge detection algorithms owing to the simplicity in the process of implementation and its optimality to meet the three criteria for edge detection. This detection algorithm is one of the most strictly defined methods that provides good and reliable detection.

The Canny edge detection algorithm can be divided into 5 different steps:

- The noise is removed by applying a Gaussian filter
- Intensity gradients of the image is calculated
- Spurious response to edge detection is removed by applying non-maximum suppression
- Potential edges determined by applying double threshold
- Hysteresis to track edges: Edges that are relatively weak and not connected to strong edges are suppressed.

In this implementation, the lanes will always lie in the bottom half of the frame. Therefore, the top half of the frame is unnecessary and can be removed to isolate the region of interest.

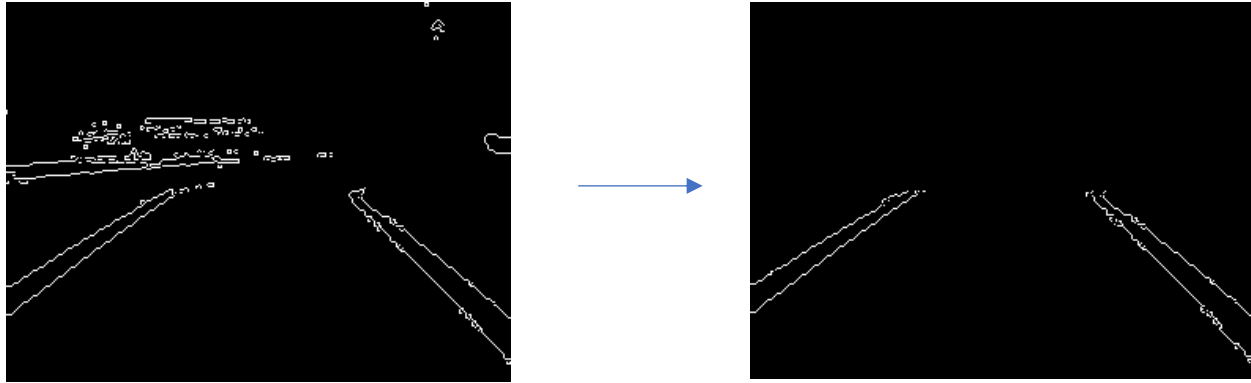


Figure 17 Isolation of region of interest

To a computer, these edges are just a group of white pixels on a black background. From these pixels, lines and curves need to be extracted. This is done with the help of an algorithm called the Hough Transform. With this function, lines will be formed along the edges of the lanes.

Hough Transform [11] is used to create a mathematical form to represent any detected shape. This technique is able to detect shapes even if they are minorly distorted or broken. $y = mx + c$ can be used to represent a line or $p = x \cos(\theta) + y \sin(\theta)$ in parametric form where p is the perpendicular distance from origin to the line, and θ is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise.

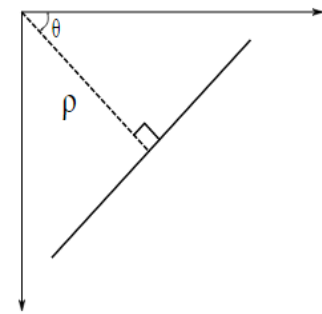


Figure 18 Representation of line

ρ will be positive and the angle will be less than 180 if the line is passing below the origin. On the other hand, the angle is taken less than 190 and ρ is taken as negative if the line is greater than 180. Horizontal lines will have 90 degree and vertical lines will have 0 degrees. ρ (p) and θ can be used to represent any line. A 2D array or accumulator is created to hold the two values and is set to zero initially. The columns of the array represent θ and the rows of the array represent ρ . The desired accuracy will determine the size of the array. 180 columns will be required if the accuracy of the angles is to be one degree. The diagonal length of the image is the maximum length possible for ρ . So, the diagonal length of the image is taken for 1 degree accuracy.

Consider an image with a horizontal line at the middle 100x100 pixel density. The first point on the line is taken. Substitute the values of θ from 0 to 180 in the line equation. The value of the accumulator is incremented by one for every (ρ , θ) pair, in the corresponding (ρ , θ) cells. The (50,90) cell in the accumulator will now be equal to one.

Upon moving on to the second point in the line, the value in the cells corresponding to (p, theta) are incremented. Hence, the value in the cell (50,90) will now be equal to two. This process is repeated for every subsequent point on the line. Each time, the value in the cell (50,90) will be incremented while the other cells need not necessarily increase. In the end of this process, the (50,90) will have the highest value. Upon searching for the maximum value in the accumulator, the cell (50,90) will be returned, indicating there is a line at a distance 50 from the image at an angle of 90 degree [11].

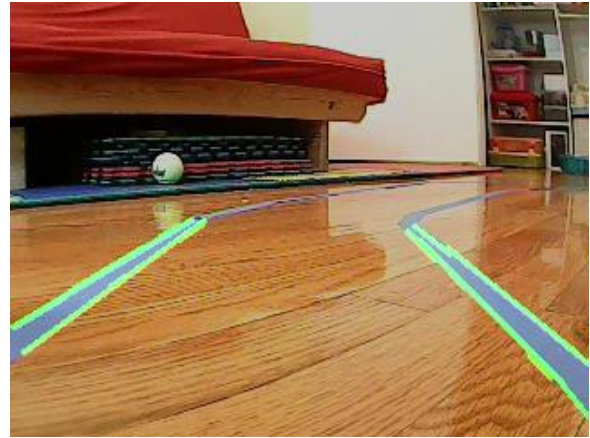


Figure 19 Lines drawn on lane

Chapter 5

Training a network to detect vehicles

5.1 Overview

Being able to detect and track cars is necessary for the ego vehicle to operate autonomously. For training the neural network to detect vehicles in still images, faster R-CNN's are utilized.

5.2 CNN vs R-CNN

CNN's are primarily used to classify images, where as R-CNN's [14], R standing for region, are used to detect objects. Traditional CNN's [13] can classify objects but cannot convey where they are. It is possible to regress bounding boxes using a CNN, but that is only possible for one object at a time. This is because regressing multiple boxes causes interference.

In R-CNN's, one object is focused at a time, so a single object of interest dominates in a given region, this minimizes interference. The regions detected are then resized so that equally sized regions are fed into the CNN for classification and bounding box regression.

5.3 Training the R-CNN

To train this model, a dataset consisting of 295 pre-labelled images of vehicles are used. The images consist of one or more labelled instances of a vehicle.

The detector is trained in four steps. The first and second step train the region proposal and detection networks. The third and fourth steps join the first two networks and creates a single network for detection. The network training options are specified using training options.

As the dataset consists of images with different sizes, the minibatch size is set to 1. This ensures that these images won't be processed together.

Setting a temporary location for checkpoint path will allow you to pause training of the network and lets you later resume from where you left off. A pretrained CNN called ResNet-50 is used for feature extraction.

On training the R-CNN [13] with the specified parameters, the network was able to detect cars with an average accuracy of 86%. The time taken to complete training totaled to 4 hours.

```
% Options for step 1.
options = trainingOptions('sgdm', ...
    'MaxEpochs', 5, ...
    'MiniBatchSize', 1, ...
    'InitialLearnRate', 1e-3, ...
    'CheckpointPath', tempdir);
```



Figure 20 Bounding box detected by RCNN

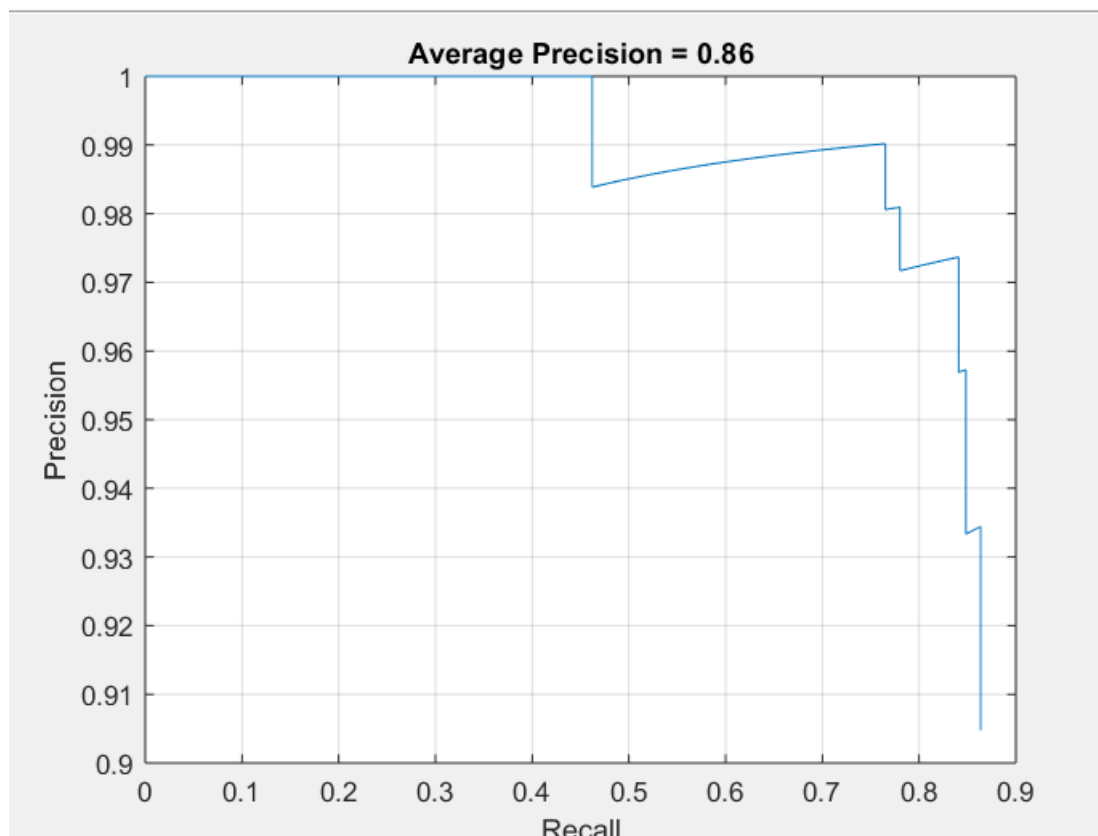


Figure 21 Accuracy Rating

Chapter 6

Tracking vehicles and pedestrian in video

6.1 Overview

In the real world, vehicles need to be identified and tracked from video input rather than still frames. This is done by identifying the vehicles using the previously trained R-CNN in each frame of the video and constantly updating the tracker with the detection results.

6.2 Workflow

The tracking workflow consists of the following steps:

- 1) Define camera intrinsics and camera mounting position.
- 2) Load and configure a pretrained vehicle detector.
- 3) Set up a multi-object tracker.
- 4) Run the detector for each video frame.
- 5) Update the tracker with detection results.
- 6) Display the tracking results in a video.

In this example, a pretrained ACF vehicle detector [17] is used and configured to incorporate camera information. By default, the detector scans the entire image at multiple scales. By knowing the camera parameters, the detector is configured to detect vehicles on the ground plane only at reasonable scales. The detector only tries to find vehicles at image regions above the ground plane. This can reduce computation and prevent spurious detections. At each time step, the detector is run, the tracker is updated with detection results, and the tracking results are displayed on the video.

6.3 Supporting functions

`setupTracker` function creates a `multiObjectTracker` to track multiple objects with Kalman filters. The following functions are considered when `multiObjectTracker` is created:

FilterInitializationFcn: Sets up models for predicting likely motion and measurements. The objects are bound to have a constant velocity motion in this scenario.

AssignmentThreshold: Used to determine how far the detections can fall from tracks. By default, the value of this parameter is set to 30. If detections that should've been assigned to tracks but aren't, this parameter is increased. If detections that are too far have been assigned to tracks, this value is decreased. This example uses 50.

NumCoastingUpdates: The number of times a track is updated or coasted before being deleted. This parameter has a default value of 5.

ConfirmationParameters: Track is conformed using this parameter. With every unassigned detection, a new track is initialized. As some of these detections may be false, the tracks were initialized as Tentative. The track needs to be detected at least M times in N tracker updates for the track to be confirmed. M and N are determined based on the visibility of the objects. 3 out of 5 updates have been used for this example.

6.4 Tracking vehicles

Using the trained R-CNN and MATLAB's automated driving toolbox, the networks was able to detect and track vehicles from video input.



Figure 22 Bounding boxes detected by RCNN in Video

6.5 Tracking pedestrians

First, this example computes the cost of assigning every detection to each track by using the `bbboxOverlapRatio` measure. As people move toward or away from the camera, the centroid point alone cannot accurately describe their motion. The cost takes into account the distance on the image plane, and the scale of the bounding boxes. This prevents assigning detections that are far away from the camera to tracks that are closer to the camera, even if the predicted and detected centroids coincide. The choice of this cost function eases the computation without resorting to a more sophisticated dynamic model. The results are stored in an M by N matrix, where M is the number of tracks, and N is the number of detections [17].



Figure 23 Bounding boxes around pedestrians detected by RCNN

Chapter 7

Collision Avoidance with Fuzzy Logic

7.1 Fuzzy Logic

Fuzzy logic [15] performs computations by taking into account the impreciseness of communication in the real world. As opposed to Boolean logic, that is 1's and 0's, values are taken based on a 'degree of truth'. Inspired by the biological processes of human cognition and perception, fuzzy logic was theorized on the idea of relative graded membership functions membership functions that relate to the extent and degree of belongingness.

7.2 Fuzzy Logic Controller

Fuzzy logic controller [15] is a control system based on fuzzy logic or fuzzy sets which analyses analog input values in terms of logical variables that take on continuous values between 0 and 1. Therefore, the vagueness and imprecise nature of the boundaries makes them useful for approximation models.

The conventional controller consists of four steps namely fuzzification, knowledge base, fuzzy reasoning and defuzzification. The first step in the fuzzy controller is to define the input and output variables of the fuzzy controller. Fuzzy logic controllers use a very flexible set of if-then rules and the controller rules are usually formulated in linguistic terms. Thus, the use of linguistic variables and fuzzy sets implies the fuzzification procedure, that is, the mapping of the input variables into suitable linguistic values. The final step is defuzzification which converts fuzzy based linguistic terms to scalar output values.

7.3 Mamdani and Sugeno Model

The most commonly used fuzzy inference technique is the Mamdani fuzzy [18] method which was proposed, by Mamdani and Assilian. It was proposed as a first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators. The task of the standard Mamdani fuzzy logic controller is to find a crisp control action from the fuzzy rule base and a set of crisp inputs.

The Sugeno [19] Fuzzy model was proposed by Takagi, Sugeno, and Kang in an effort to develop a systematic approach to generating fuzzy rules from a given input-output dataset. A typical fuzzy rule in a Sugeno fuzzy model has the form,

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = f(x, y)$$

where A and B are fuzzy logic sets while $z=f(x,y)$ is a crisp function.

7.4 Methodology

An environment for the navigation of the robot is created in the VREP software for the simulation and performance of the robot in an object filled environment. The robot used for this purpose is Pioneer 3dx, a small lightweight two-wheel two motor differential drive robot ideal for indoor laboratory or classroom use. Further, three ultrasonic sensors are attached at the left, right and front parts of the robot to detect objects.

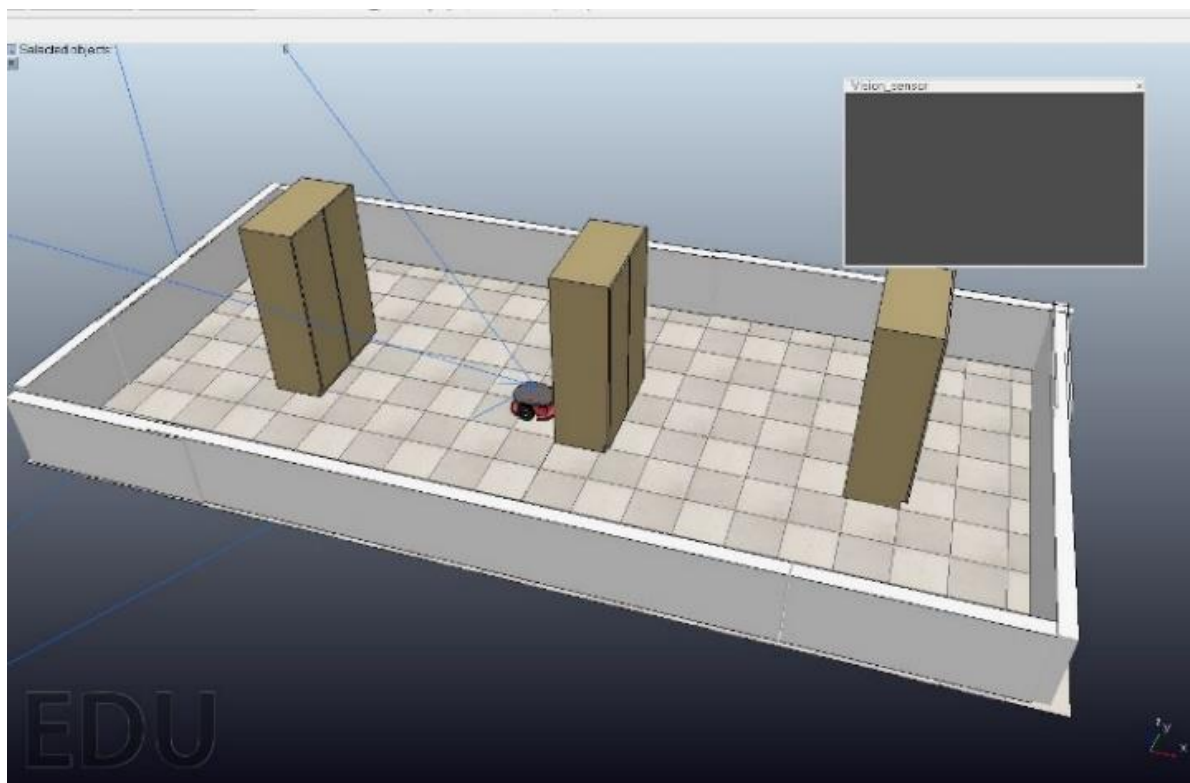


Figure 24 VREP Scene

The various fuzzy stages like suitable identification of input and output parameters along with linking them to the fuzzy based variables is carried out in the fuzzy logic controller and is implemented in Matlab using the fuzzy logic toolbox. A Mamdani based controller

with three inputs and two outputs is obtained with the inputs being the left, right and the front ultrasonic sensor and the outputs correspond to the velocity of the left and right motor. The membership function and the ranges for the input and output variables were assigned after various trials to get the accurate result. Further the rulebase for the relation between the input, output and fuzzy variables is framed by using fuzzy rules and is created in the controller. The matlab file is linked with VREP scene by code and is thus executed and the result is noted.

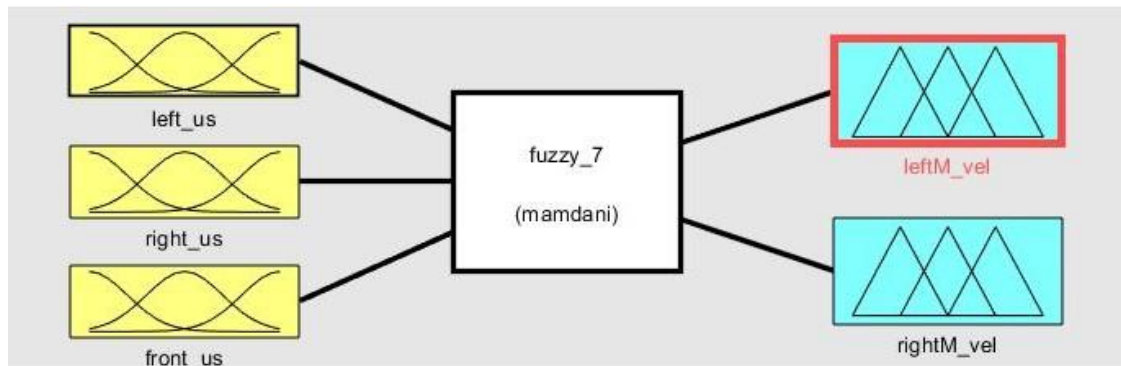


Figure 25 Membership Functions

7.5 Result

The created VREP scene with the robot was executed and it is observed that the robot could successfully avoid objects in its vicinity. Thus, the aim was achieved and the collision avoiding robot was simulated on software. This design allows the robot to navigate in an unknown environment by avoiding collisions, which is a primary requirement for any autonomous mobile robot.

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

This paper goes over different methodologies that can be implemented towards the design of an autonomous vehicle. It covers the basic functioning of neural networks and goes in-depth on how deep learning works and how it's implemented for solving different models. Using MATLAB's automated driving toolkit, an R-CNN was trained to identify vehicles and pedestrians in individual frames and then subsequently track them over the length of the video input. Using image processing techniques, the lanes in a frame were isolated and corresponding lines were plotted on the lane in order for the ego vehicle to make appropriate maneuvers. Finally, using Matlab's fuzzy logic toolbox and the VREP simulation software, an object avoidance system based on fuzzy logic was designed and successfully implemented.

8.2 Future Scope

The models explored and used in this project can be further developed to be implemented in real world models of vehicles in order to achieve certain levels of autonomy. Over the last 50 years, there have been significant advancements in technology that helped speed up the development towards completely autonomous vehicles. Although the future regarding autonomous vehicles remains uncertain in the face of rapidly changing legal and political institutions, there is no doubt that car manufacturers will continue to develop and implement advanced technology as the race towards autonomy progresses.

References

1. Campbell, M. (2010). Autonomous driving in urban environments: approaches, lessons and challenges.
2. Society of Automotive Engineers. (2018). Levels of Driving Automation.
3. SunFounder. (n.d.). Retrieved from <https://www.sunfounder.com/smart-video-car-kit-v2-0.html>
4. Raspberry Pi. (n.d.). Retrieved from <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
5. Sunfounder. (n.d.). *Smart video car for Raspberry Pi handbook*.
6. ["CS231n Convolutional Neural Networks for Visual Recognition"](https://github.com/cs231n/convnet). *cs231n.github.io*. Retrieved 2018-12-13
7. ["Convolutional Neural Networks \(LeNet\) – DeepLearning 0.1 documentation"](https://github.com/LISA-Lab/deeplearning01). *DeepLearning 0.1. LISA Lab*. Retrieved 31 August 2013.
8. Chahal, A. (2018) In Situ Detection of Road Lanes Using Raspberry Pi <https://pdfs.semanticscholar.org/cc87/79b99f4d11faf3e32313380167c4d317ec18.pdf>
9. Howard, Jacci (2019) The HSV Colour Model in Graphic Design <https://www.lifewire.com/what-is-hsv-in-design-1078068> , Lifewire
10. OpenCV, Canny Edge Detection, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html
11. OpenCV, Hough Line Transform, https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html
12. Coral, USB Accelerator, <https://coral.withgoogle.com/products/accelerator/>
13. ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al. <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
14. Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., <https://arxiv.org/abs/1311.2524>

15. Fuzzy Logic, Standford Encyclopedia, <https://plato.stanford.edu/entries/logic-fuzzy/>
16. Arduino, Opensource, <https://opensource.com/resources/what-arduino>
17. A Robust Vehicle Detection Approach based on Faster R-CNN Algorithm, Tourani et al., <https://ieeexplore.ieee.org/document/8785988>
18. Mamdani Fuzzy Model, Research Hub, <http://researchhubs.com/post/engineering/fuzzy-system/mamdani-fuzzy-model.html>
19. Sugeno Fuzzy Model, Research Hub, <http://researchhubs.com/post/engineering/fuzzy-system/takagi-sugeno-fuzzy-model.html>

Turnitin Originality Report

- Processed on: 14-Dec-2019 22:00 +04
- ID: 1234439755
- Word Count: 5257
- Submitted: 3

Submission By Aamer Abdul

Similarity Index

12%

Similarity by Source

Internet Sources:

8%


Publications:

4%

Student Papers:

9%

[exclude quoted](#) [exclude bibliography](#) [exclude small](#)

[matches](#) mode:  Change mode [print](#) [download](#)

2% match (Internet from 13-Dec-2019)

https://en.wikipedia.org/wiki/Arduino_Uno

2% match (Internet from 02-Aug-2019)

<https://www.sunfounder.com/learn/SunFounder-PiCar-S/appendix-picar-s.html>

2% match (Internet from 29-Jan-2017)

<http://au.mathworks.com>

1% match (Internet from 17-Dec-2007)

<http://www.ncptt.nps.gov>

1% match (student papers from 27-Aug-2008)

[Submitted to University of Bradford on 2008-08-27](#)

1% match (student papers from 30-Apr-2018)

[Submitted to Texas A&M University, College Station on 2018-04-30](#)

1% match (student papers from 13-Feb-2019)

[Submitted to University of Southern California on 2019-02-13](#)

<1% match (Internet from 25-Jun-2019)

<https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>

<1% match (student papers from 31-May-2015)

[Submitted to BITS, Pilani-Dubai on 2015-05-31](#)

<1% match (publications)

[Ion Iancu, Mihaela Colhon. "Mamdani FLC with Various Implications", 2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2009](#)

<1% match (publications)

[Laurine A. Ashame, Sherin M. Youssef, Salema F. Fayed. "Abnormality Detection in Eye Fundus Retina", 2018 International Conference on Computer and Applications \(ICCA\), 2018](#)

<1% match (student papers from 27-Apr-2008)

[Submitted to University of Bradford on 2008-04-27](#)

<1% match (student papers from 17-Oct-2019)

[Submitted to Jabatan Pendidikan Politeknik Dan Kolej Komuniti on 2019-10-17](#)

<1% match (student papers from 20-Apr-2019)

[Submitted to University of Sydney on 2019-04-20](#)

<1% match (student papers from 13-Feb-2019)

[Submitted to University of Southern California on 2019-02-13](#)

<1% match (student papers from 01-Dec-2019)

[Submitted to Assumption University on 2019-12-01](#)

<1% match (student papers from 31-May-2018)

[Submitted to M S Ramaiah University of Applied Sciences on 2018-05-31](#)

<1% match (Internet from 23-Feb-2019)

https://baadalsg.inflibnet.ac.in/bitstream/10603/107457/16/16_chapter8.pdf

<1% match (publications)

[Kyriakidis, M., R. Happee, and J.C.F. de Winter. "Public opinion on automated driving: Results of an international questionnaire among 5000 respondents", Transportation Research Part F Traffic Psychology and Behaviour, 2015.](#)

<1% match (student papers from 06-Dec-2016)

[Submitted to Institute of Technology, Nirma University on 2016-12-06](#)

<1% match (student papers from 05-Jun-2011)

[Submitted to BITS, Pilani-Dubai on 2011-06-05](#)