BACHELOR'S PROJECT:

- ***Title:*** ***Autonomous Vehicle implementation with End to End Deep Convoluted feed forward neural network and S.L.A.M***
- *Student:* *Md Mohidul Hasan 15071717*
- *Supervisor:* *DR. Raimond Kirner*
- *Branch:* *Software Engineering*
- *Department:* *Department of Computer Science*
- *Validity:* *Until the end of winter semester 2019/20*
- *Date:* *30th April, 2019*

Acknowledgements:

First and foremost, I would like to thank my supervisor Dr. Raimond Kirner for granting me the opportunity to explore such a vast and interesting field of Machine Learning and neural computation and for always being available and pointing me to the right direction.
I would like to express my gratitude to all my friends and family for helping me make this project a reality.
Last but not the least I would like to acknowledge with much appreciation, the staff of the school of computer science, University of Hertfordshire for granting me access to the school resources at all times.

Declaration Statement:

I hereby declare that, the thesis presented, is my own work and all the sources of information in accordance with the guide line for adhering to principles of ethics when expanding an academic thesis have been cited.

I acknowledge that my thesis is subject to the rights and obligations owned by University of Hertfordshire, School of Computer Science.

April, 30th, 2019

Abstract:

The proposition of the project undertaken is to design, develop and implement an RC car employing a Raspberry pi (Computer), deep convoluted neural networks and SLAM. The objective is to successfully employ the concept of behaviour cloning within the field of vehicle operation exercising miscellaneous computer vision algorithms alongside concepts like Machine learning. A single front facing camera have been used, images from which will be passed onto the Supervised CNN to retrieve predicted steering commands. The proposed model is an illustration of supervised learning.

The report simply explains the entire developmental process of the project. The resolution of the project would be to demonstrate a computer algorithm operating a vehicle with human like precision. This will be done by employing deep end to end convoluted neural networks model proposed by NVidia (Karol Zieba NVIDIA Corporation, 2016), open source computer vision library known as open cv, and an RC car with raspberry pi and a driver board. The NVidia CNN model is able to generalize and obtain abstract models of different road features from within any image with almost human like precision with just one input parameter.

SLAM algorithms make use of LIDAR sensor, which will be the lesser focus of the project. SLAM algorithms are generally applied to retrieve a map of the environment; the vehicle is being driven into simultaneously localising itself within that map. An extended version of the Kalman filter is applied to the LIDAR data to acquire the map. Different objects located within the environment are also positioned within the map recovered.

Abbreviations:

CNN (Convoluted Neural Network)
SLAM (Simultaneous Localisation and Mapping)
LIDAR (Light Detection and Ranging)

Contents:

Introduction:

Supervised deep learning algorithms such as CNN's have brought remarkable success in the fields of computer vision and artificial intelligence. In many cases these algorithms have out-performed classical feature extraction based A.I algorithms for classification problems like semantic segmentation, object detection and labelling (Image captioning). These network of algorithms can extract features from a well-defined image with numerous levels of abstraction which would otherwise be very difficult for human beings to extract and classify employing generalised rules. With the increasingly large amount of open sourced available datasets and with the increasing power of computation employing GPU's and TPU's, Computers can train these models effectively and quickly to yield increasingly better results. In today's world this technology can be seen in most places , whether it be Google's search engine, or google news's or in medical science to classify malignant and benign tumour's or even in housing business to predict the price based on some set and defined parameters. A field that is benefitting with the emergence of this technology is the automotive engineering. Self-driving cars are being engineered by all most all of the fortune 500 companies like TESLA, WAYMO or UDACITY.

figure 1 : Udacity self-driving car engineer (Udacity)

Autonomous vehicles have already started to revolutionize transportation. Self-driving vehicles are well equipped to save lives in situations where it is not possible for a human driver to avoid the accidents purely because of the long ranged sensors like, Ultrasonic, Lidar or the Camera etc. These sensors provide vehicles with almost super human like abilities regardless of the fact that machines comprise of a far better reaction time then human beings. This is why companies like google and Udacity are providing the general public with open sourced resources for Machine learning like TENSOR FLOW or UDACITY SELF DRIVING SIMULATION and educating them to contribute towards the solutions for the problem of vehicle autonomy.

Hypothesis:

The goal of this project is to develop a low cost prototype using raspberry pi RC car and end to end deep CNN.

The car should be able to drive with a well-defined track identifying the lanes from processing the image captured by the camera mounted over the car. The track will be indicative of a simplified road. The only put parameter to the agent will be images captured by the camera. The agent will then process the image and output a steering angle based on the input. As the model accepts images as the only source of input, the thesis of this undertaking will be to design, construct and develop a model that can successfully control the RC car by providing steering commands based on appropriate input images.

A separate model has also been trained employing supervised classification algorithms to process and identify traffic signals. Similar to the model mentioned above, it will only take images as appropriate input, process it within the model and classify the image as output argument.

While obstacle avoidance and detection is a major problem when considering self-driving cars, but combining it with the steering angle as an output for the model within one single controller loop is beyond the scope of this thesis. Although, a LIDAR device will be used for obstacle avoidance and slam within a separate loop which will not affect the loop with the steering commands or the agents.

An inexpensive RC car chassis along with a cheap raspberry pi with 4 DC motors will be used. The software will be written using python machine learning libraries for the purpose extendibility.

Definitions:

Neural Network: A neural network in simple terms is defined as a network of simplified processing units that share similar functionalities with biological neurons. The processing ability is obtained by adapting to a set of training data extrapolating useful patterns and storing them as weights. (Gurney, 1997)

Feed Forward Neural Networks: Within a neural network model if there is no feedback loop or feedback directed from the output of the model towards its input, then the model can be described as feed forward neural network. (SAZLI)

CNN: A CNN also known as a convent, is a subclass of deep feed forwards neural networks designated to analysing visual data or images. (University)

Artificial Intelligence: A branch of Computer science that deals with the science and engineering of intelligent computer programs to understand human intelligence but also not confined within the domain of methodologies that are biologically inspired. (-Prakhar Swarup, 2012)

Machine Learning: According to Tom M. Mitchell "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". (wiki, 2018)

Classification: Within machine learning and statistics, classification is defined as the process of identifying the category among a set of categories to which an observable data belongs to, based on a set of training instances inclusive of the categories defined. (wikipedia)

Supervised Learning: Supervised learning is the task of learning a specified function that maps a certain input variable to an output variable based on tuples of large input output pairs. It is a subset of machine learning. (wikipedia, 2019)

Deep Learning: Deep learning also known as deep structured learning or hierarchical learning is a segment of machine learning methods based on the number of layers within the developed model of the neural network. (wikipedia)

SLAM: Within the fields of robot mapping and odometry, navigation , SLAM is the computational model of simultaneously mapping an unknown environment while simulating the agent's location within the environment. (ross)

Project background:

Levels of Autonomous Driving:

There are five levels of autonomous driving according to National Highway and Traffic administration. (Forbes)

Level 0(No automation): Vehicle is completely driven by a human. (Forbes)

Level 1(Driver assistance): The driver is being assisted by the vehicle with certain features but not at the same time. E.g: keeping lane, controlling cruise or assisted breaking. (Forbes)

Level 2(Partial automation): The vehicle can simultaneously steer, accelerate and decelerate by itself. Although human driver is expected to do other manoeuvres when required relating to driving. E.g: Tesla Autopilot（Forbes）

Level 3(Conditional automation): The vehicle can perform driving in its entirety in most cases. The human driver is expected to take control when the system fails or the conditions are not favourable for autonomous driving. E.g: Waymo|(Google)（Forbes）

Level 4(High automation): The vehicle can perform driving in its entirety in most cases. However, the human driver is not expected to take control when the system fails or the conditions are not favourable for autonomous driving. The vehicle should be able to stop, or come at a safe position if the human driver fails to retrieve vehicle control. E.g: Google's latest Waymo project is supposed to level 4 although it is in its testing phase. (Forbes)

Level 5(Full automation): The vehicle driving system is able to navigate and drive in all cases and circumstances. (Forbes)

This thesis is aimed at developing a driving system with a level of automation between 2 and 3. Navigation will be completely autonomous while decelerating will only work when an obstacle of some sort is detected. (Forbes)

Research into hardware:

Raspberry Pi 3 b +:

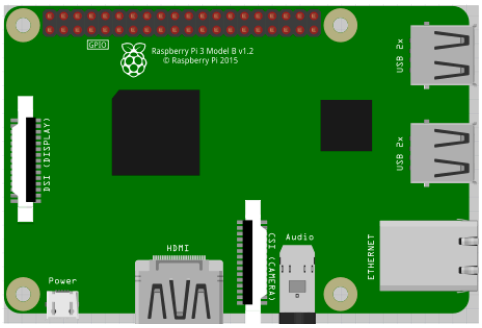

fig: raspberry pi 3 b+

A Raspberry Pi is a low cost bank card sized computer that can be plugged into any monitor or screen. It also uses general keyboard and mouse used for a desktop's.

It consists of 40 GPIO pins or header that allows the user to connect to it slave circuit and modules, sensors like the ultrasonic sensor, IR (Infrared) sensor or LIDAR. It also consists of USB slots to connect webcams or designated pi camera slot to connect to a pi camera.

Within the GPIO header different protocols like IDEPROM or UART (universal asynchronous receiver-transmitter) or I2C bus is encoded with pins like SCL, SDA, TXD, RXD.

It also consists of a 1.3 Ghz quad-core processor and 1 GB of RAM (Random Access Memory).

GPIO (General purpose input and output):

fig : Pi GPIO header

The GPIO pins in raspberry pi allow the users to interface physical devices like DC motors, ultrasonic sensors, LIDAR, camera, IR with the Linux kernel employing python programming language within the ROS (Raspian or Robot operating system). Using python RPI.GPIO library input to these physical systems can be controlled.

There are 2 pin numbering system for GPIO headers known as the BOARD pin numbering system or the BCM (Broadcom soc numbering system). BCM generally differ for different raspberry pi versions while BOARD pins follow the exact same pattern engraved within the pi GPIO header.

For the scope of this thesis the BOARD pin numbering system has been used.

Lidar:

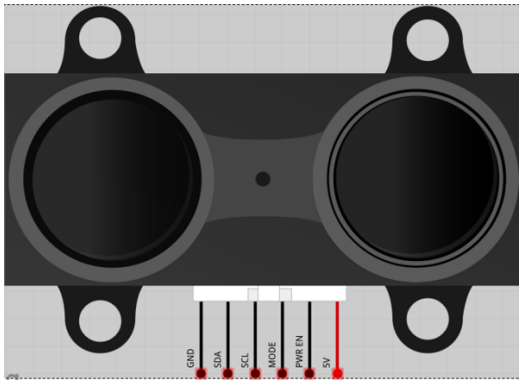

fig: LIDAR

Lidar sensors are generally used to map unknown environments with extreme precision. These sensors generally use a pulsed lased to measure the distance between the object and the sensor itself. These generated pulses are used to then measure the surface of the object with precision.

The LIDAR used for this thesis employs UART protocol as such communication between the pi and the LIDAR is very difficult to establish. It consists of 4 pins namely TX (transmitter), RX (Receiver) , VCC (Voltage) and Ground.

The data received from this device will be used to implement the SLAM algorithm. It can also be used for Object detect and obstacle avoidance.

USB – Camera:

The only source of input for our model will be images captured by the camera module mounted at the top of the vehicle. It will be connecting to one the USB ports and python imaging library alongside Open cv will be used for image manipulation, augmentation and processing.

Power Management:

A 12-volt battery is being used to power the driver board used as the HAT for the pi. Another separate 12-volt portable charger is being used to power the pi itself.

DC motors and driver Board:

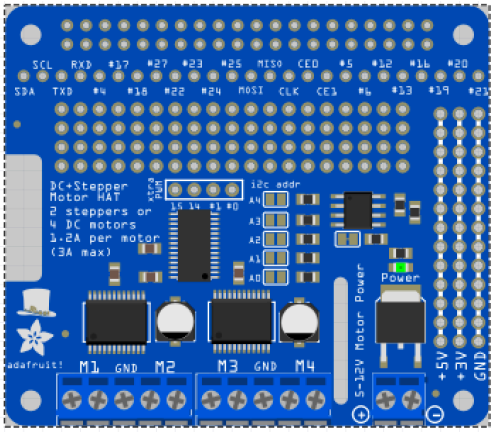

fig: Ugeek Stepper Motor Hat

A stepper motor hat with dual L298 H-bridge motor drivers are used to control 4 DC motors. This driver board used as the HAT for the pi as such interfacing with pi becomes a lot easier and maintainable.

Python scripts are run within the pi terminal to control the 40 GPIO pin header and the DC motors connected to the driver board. The driver board consists of 2 input units for each motor, which will then be used to drive the motors forwards and backwards.

Motor driver consists of PWM (Pulse width modulation) pins as well which will be used to control the speed of the vehicle.

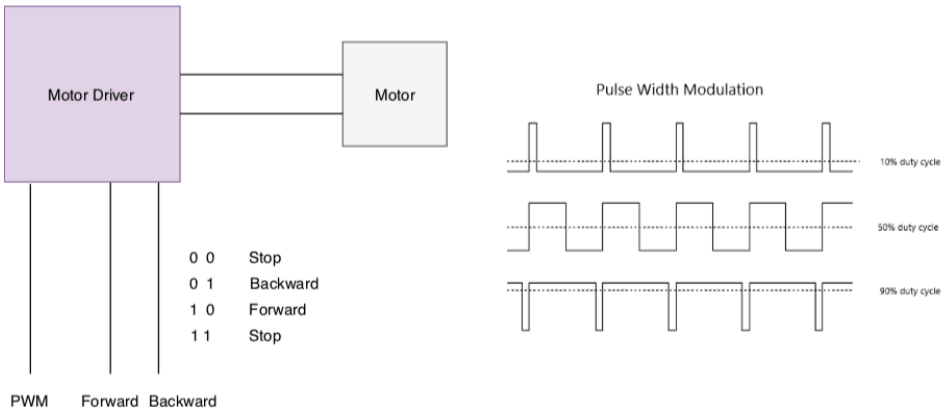

fig: Motor driver connection / Voltage control using PWM

Hardware Configuration:

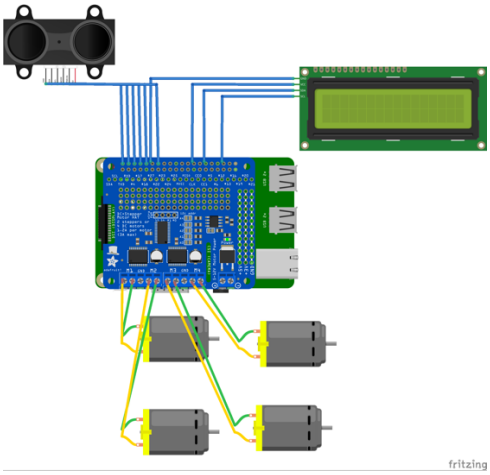

fig: Abstract diagram of the Hardware Structure of the vehicle

All the devices connect above consists of specific pin configuration. These input pins are programmed using python and rpi GPIO python library to read and write data to the registers of the driver board. The program is then placed within the pi that in turn controls the pi GPIO channels.

Sensors	TX	RX	Sensors	SCL	SDA
Lidar	14	15	LCD	5	3

Fig: Pin configuration for the sensors

PWM pins in pi:

Within pi GPIO header, there consists of specialised pins known as the PWM pins.
Generalised gpio pins cannot be used as pwm pins as their frequency is very low.
Pwm pins are those pins that consist of a higher frequency than normal pins.

Pi Setup:

The raspberry pi needs to be configured properly before it can be used. The operating system known as ROS (Raspian) needs to be installed. This is done by flashing a USB drive with the ROS DMG file. The pi is connected to a monitor, keyboard and mouse. The usb with the OS is then flashed within the micro SD card of the pi. A complete tutorial for setting up ROS on pi can be found online.

Assembling car:

A smart car chassis is being used to mount all the necessary components of the vehicle. It consists of well-defined holes used to mount the pi, mount of the camera, DC motors, Lidar and LCD.

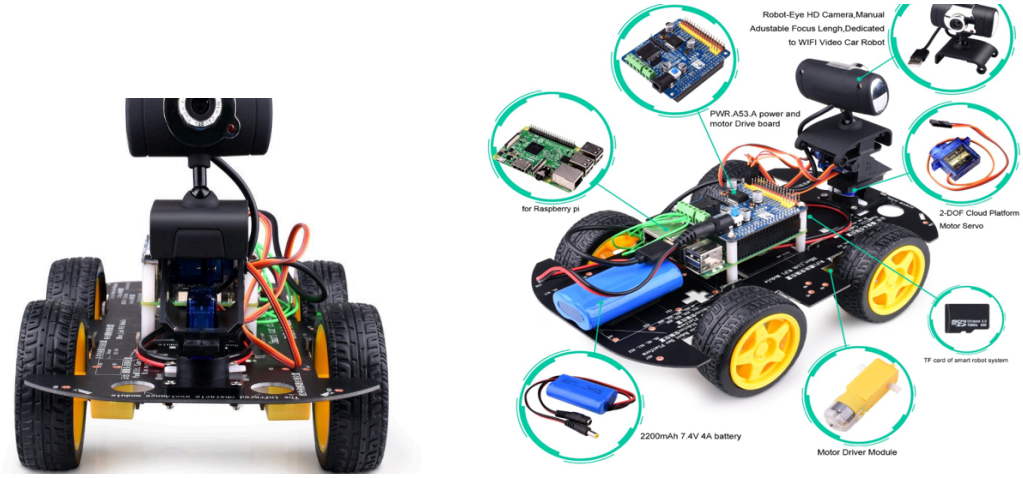

fig: Car chassis and mount

At the top front section of the car, the camera is mounted on top of a mount connected to the chassis body. The LIDAR is mounted on top of the Camera. 2 batteries, one at the front and the other at the bottom is placed.

System Analysis and design:

Ideas and concepts can sometimes become over complicated and confusing. Employing abstraction to disintegrate the idea into smaller sets of problems to make the entire development process feasible, especially in cases of complex developmental procedures. Below a set of activity, use case, sequence and class diagrams have been displayed.

Use Case Diagram:

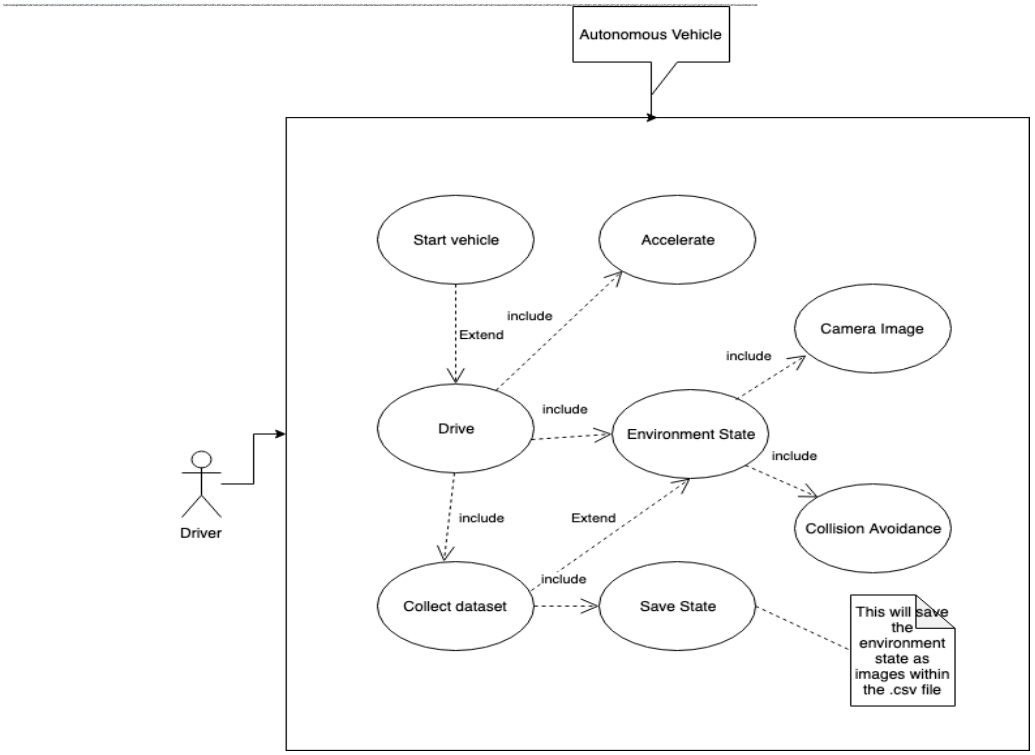

fig: Use case diagram of the proposed thesis

Sequence Diagram:

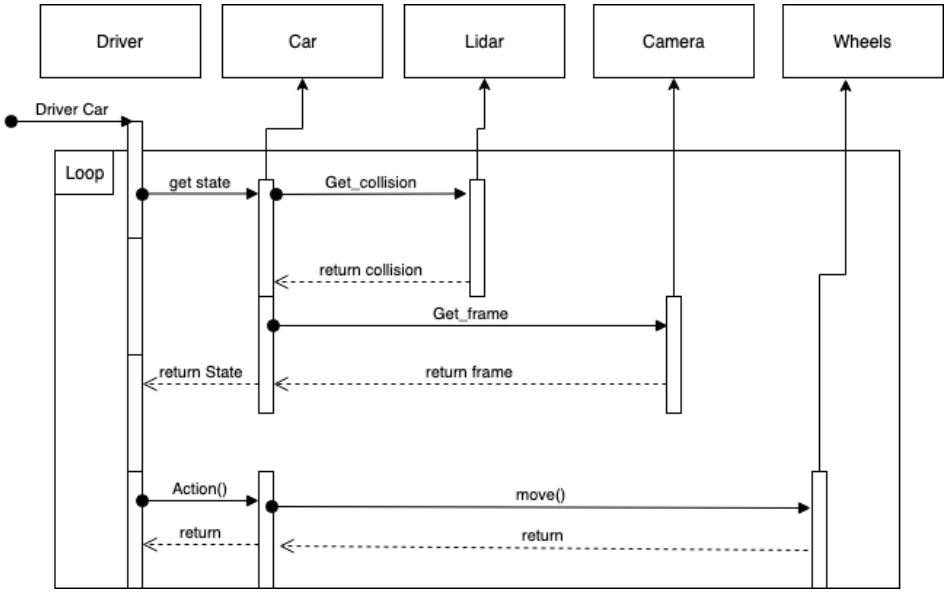

fig: Sequence diagram

Class Diagram:

These diagrams generally help with defining numerous variables and properties. But within the thesis, it'll be used and referenced as the proposed subroutines belonging to the system.

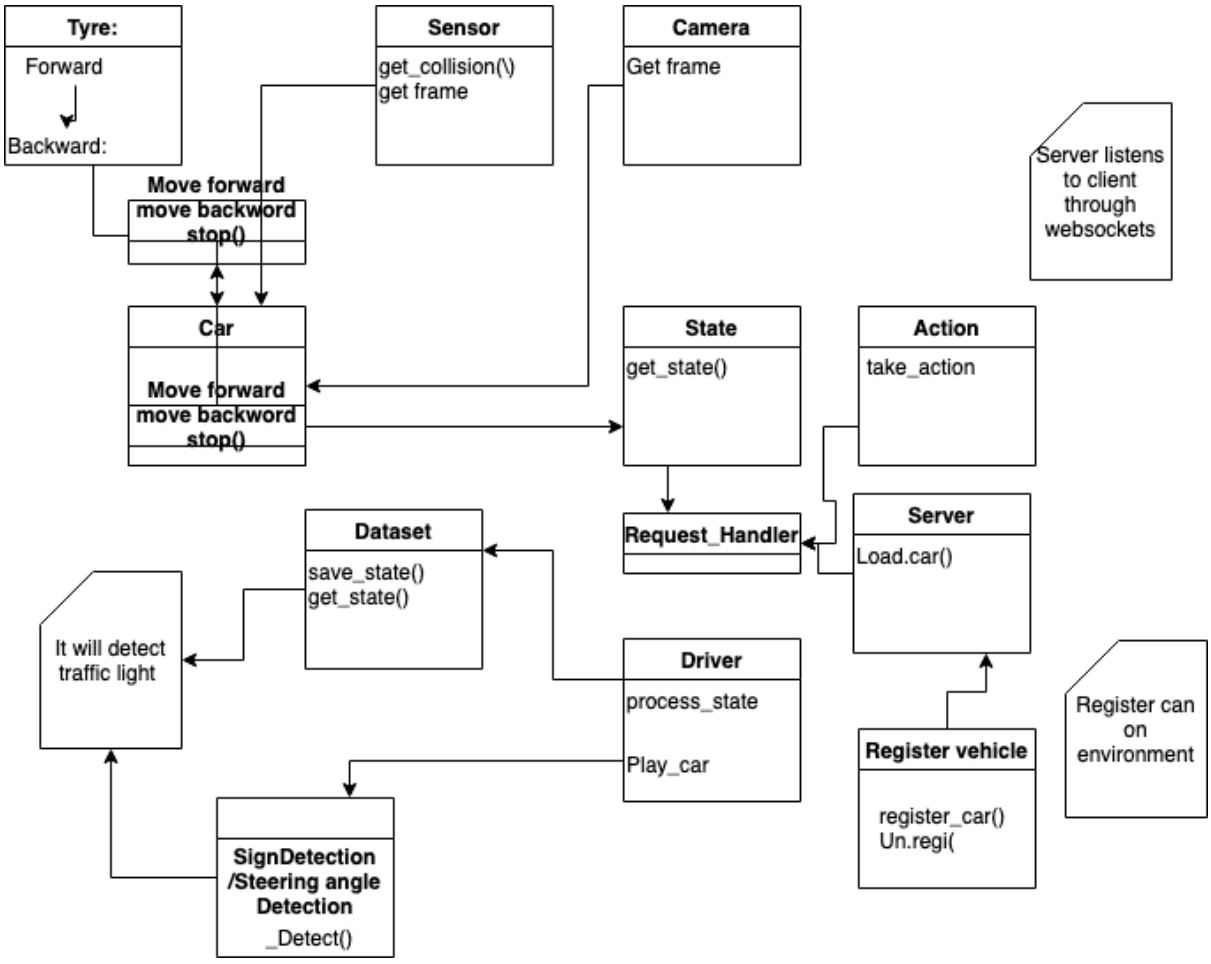

fig: Class Diagram

Activity Diagram:

The vehicle will initiate its sensors and camera after automatically discovering the network. It will then take input from the model regarding steering actions (Left, Right, Forward, Backward). The current state of the car will be generated after executing the collision avoidance function as a result of which, every time the car detects objects or obstacles it comes to a safe state of "STOP". While no obstacles or collision, the vehicle will be progressing based on the prediction steering command by the trained model (model will be discussed within the Deep Learning and AI section).

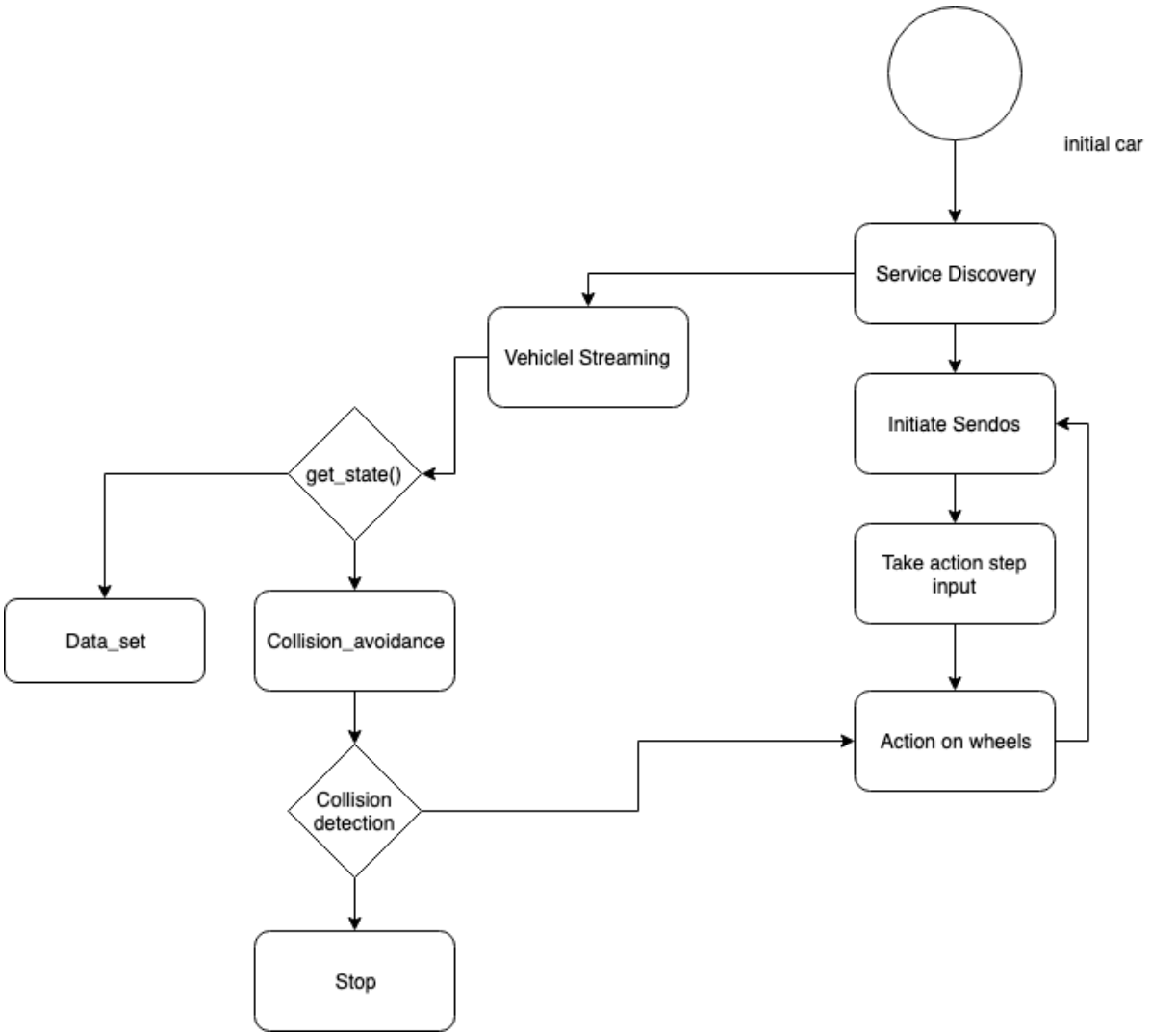

fig: Activity diagram

Software Interface:

This chapter describes in detail, all the necessary software used to establish connection and interface with different modules employed by the autonomous vehicle.

System Architecture:

The main application or control loop is run within the pi. A web interface have also been implemented through which users can interact with the system. The vehicle can be controlled manually through the browser or a simulation has also been implemented, that can be used to run a pertained ML model driving a vehicle. The simulation is based on UDACITY self-driving car model.

Used Technologies:

The entire software stack has been written on python 3.6 except the front end of the web application for manual control of the vehicle. The front-end was built employing HTML, CSS, Bootstrap and JavaScript. A lot of standardised python libraries have been used for sockets and threads
Third party python libraries like Flask and Cherry have been used for backend section of the web app.
The model was trained used general python libraries like numpy, keras, sklearn, pandas and Tensorflow.

Car's Main loop:

The main loop within the main.py file runs indefinitely. At the beginning it looks for obstacles within the track. While there is no obstacles within the track the vehicle gets most recent steering angles from the model and feeds it into the actuators. Within the main loop logic is also added for 3 traffic sign's namely : stop, red light and cross sign (No road forward). A second ML model is also trained using multi class classification algorithms to predict traffic signs.

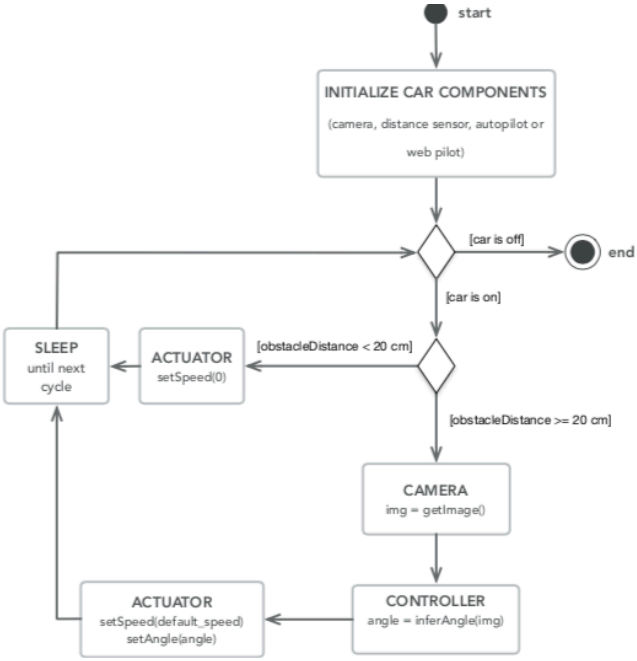

fig: Car's life cycle

While there are no obstacles detected or no traffic signs, the vehicle takes in and process output variables as steering angles from the trained model to drive autonomously within the track.

Controlling the Servo:

The camera at the front of the car Chassis is mounted on top of 2 servos'. This was done to effectively increase the degree of freedom of the camera. These servos are connected to the pwm channels on the driver board as the pi not efficient in terms of pwm channel generation.

```
#!/usr/bin/python

from Raspi_PWM_Servo_Driver import PWM
import time

# ===========================================================================
# Example Code
# ===========================================================================

# Initialise the PWM device using the default address
# bmp = PWM(0x40, debug=True)
pwm = PWM(0x6F)

servoMin = 150  # Min pulse length out of 4096
servoMax = 600  # Max pulse length out of 4096

def setServoPulse(channel, pulse):
  pulseLength = 1000000                   # 1,000,000 us per second
  pulseLength /= 60                       # 60 Hz
  print ("%d us per period" % pulseLength)
  pulseLength /= 4096                     # 12 bits of resolution
  print ("%d us per bit" % pulseLength)
  pulse *= 1000
  pulse /= pulseLength
  pwm.setPWM(channel, 0, pulse)

pwm.setPWMFreq(60)                        # Set frequency to 60 Hz
while (True):
  # Change speed of continuous servo on channel 0
  pwm.setPWM(0, 0, servoMin)
  time.sleep(1)
  pwm.setPWM(0, 0, servoMax)
  time.sleep(1)
```

fig: Example servo control code

LIDAR:

The LIDAR sensor has been used for obstacle avoidance since implementation of SLAM algorithms have become out of scope for the proposed thesis. A file named lidar.py contains all the necessary program code to retrieve desired output from within the LIDAR device. A code snippet of it has been added below for clarification purposes. Python serial package has been imported and used.

```
# -*- coding: utf-8 -*
import serial

ser = serial.Serial("/dev/ttyAMA0", 115200)

def getTFminiData():
    while True:
        count = ser.in_waiting
        if count > 8:
            recv = ser.read(9)
            ser.reset_input_buffer()
            if recv[0] == 'Y' and recv[1] == 'Y': # 0x59 is 'Y'
                low = int(recv[2].encode('hex'), 16)
                high = int(recv[3].encode('hex'), 16)
                distance = low + high * 256
                print(distance)



if __name__ == '__main__':
    try:
        if ser.is_open == False:
            ser.open()
        getTFminiData()
    except KeyboardInterrupt:   # Ctrl+C
        if ser != None:
            ser.close()
```

fig: Example Lidar code

A LIDAR device as mentioned above employs UART protocol. Within this protocol there are 2 pins that are being used, namely serial transmit (TX) and serial read (rx) . The TX pin of the LIDAR is connected to the RX pin of the pi while the RX pin of the Lidar is connected to the TX pin of the pi for establishing a successful 2 way data communication service.

LCD display:

An LCD (Liquid Crystal Display) device employs I2C bus. It is essential a protocol employed by low speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems.

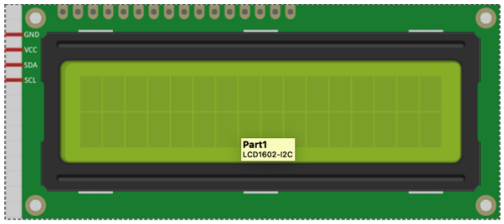

fig: LCD display

Within the LCD there are 4 existing pins known as the SCL (serial clock) , SDA (serial data) , VCC and ground. The SCL and SDA pins are connected to the corresponding SCL and SDA pins on the driver board.

Web Interface:

Highly sophisticated web development architecture known as the MVC or Model, view and controller architecture has been used to design and develop the web interface. Flask package has been used for developing the backend of the web app while bootstrap for the GUI (Graphical user interface).
Users can interact with the vehicle through the GUI. It is made possible to drive the vehicle forwards, backwords, left and right by making use of the buttons and sliders within the GUI or by using a ps3 controller. Users can also turn the recording on or off, modify speed and see the camera input in real time. Once the car is started the interface can be found at ip_address:5000.

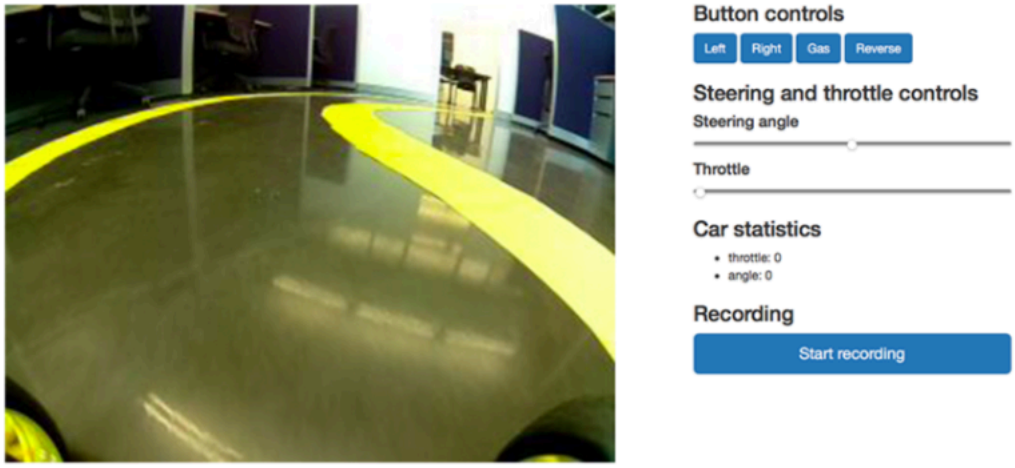

fig : frontend of the web interface

Command Line Interface:

The car can be run in 2 modes. Either in manual mode where the user has to manually drive the car around or in auto pilot mode where the controller takes control of the vehicle.
Within the auto pilot mode, a python script is used to make the vehicle run around a lane using computer vision lane detection algorithm.
On the contrary, the user has to manually take control of the vehicle and drive it using keyboard w, a, s, d keys for forwards, left, back and right.

Dataset Recording:

Collecting the appropriate training data is one of the most crucial tasks undertaken within the scope of the thesis. Not only is it important for the entirety of the thesis, but also for the CNN model which will be trained using the dataset recorded.

Dataset can be collected via the web interface making good use of the recording button or via the command line interface by running the autopilot.py script. The entire process is controlled by an object called the recorder. It creates a folder pyramid which reflects the current date and the number of records. A single dataset entry is saved on every tick of the car's main loop. The frequency can be changed in file config.py.

Since this is a supervised learning problem, we need to have an input object and a corresponding output value. In this case our input is an image and the output is a json file containing the *speed* and *steering angle* at the *time* the picture was taken. These files are later used to train the machine learning model.

The initial training data was collected using a simulation. Udacity self-driving car simulation was built upon unity game engine. It allows the user to drive a car within a real road with extraneous circumstances, while recording the entire drive. These videos are later on processed into images and a corresponding data.csv file is created that has the images as the input value and the steering angle as the labels.

fig: Visualisation of training data, acquired from MNIST dataset

Approaches to Steering in Autonomous Driving:

Currently there exists 3 ways to deal with the problem of autonomous driving.
They are:

- Non-AI approach (Manual engineering)
- AI approach (CNN, Reinforcement Learning DQN)
- Combination of AI and non AI

Non-AI approach (Manual engineering):

The non AI approach makes use of control theory to keep the vehicle within the lane. It calculates the steering angle based on environment it is driving on using open sourced computer vision algorithms. One of the most popular methods of control theory is PID (Proportional, integral and derivative) controller. The controller works within a loop calculating the error value e (t) with every iteration as the difference between the vehicle's feedback and the next command signal. Afterwards, a correction value is calculated and applied.

The corrected value u (t) consists of 3 arguments namely proportional, integral and derivative and can be computed from the error e (t). (Zhao, et al., 2012)

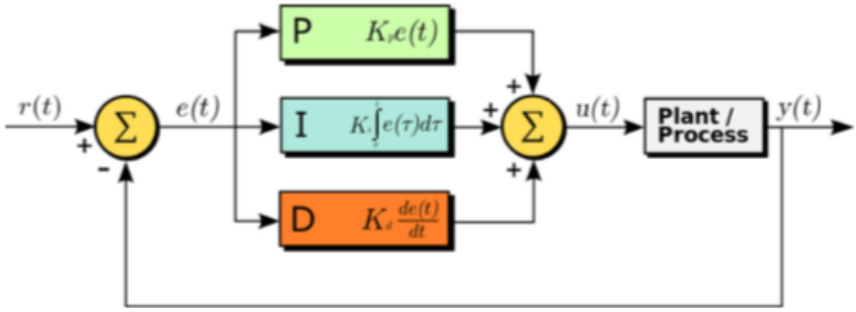

fig: Computation of the corrected value within a PID loop controller

The entire mathematical model is given below:

$$u(t) = k_p\, e\,(t) + K_i \int_0^t e\,(\tau)\, d\tau + K_d \frac{d\, e(t)}{dt}$$

Each parameter has their own coefficients k_p, k_i and k_d needs to be finely tuned for the model to perform well. An in depth analysis of the parameter tuning is out of scope for the thesis undertaken as the approach mainly interested in is the AI approach. For further details refer back to (Levine) , (Chong, et al.) , (Chandni, et al., 2017) and (Zhao, et al., 2012)

AI Approach:

Unlike the non AI approach, The AI approach does not compute the optimal steering value based on control logic and equation rather, it depends on an intelligent system to predict the best steering angle based on previously processed data.

Such agents can be developed and trained using Machine Learning specifically deep learning techniques that can process large data with the aim of recognizing and extrapolating road

features with the most precise level abstraction with the goal of predicting steering angles the vehicle should perform in order to follow the track.

With the rise of deep neural networks and fortune 500 companies like Google, waymo reshaping the idea of driving and integrating it with the field of machine learning and achieving success, new possibilities are in order. The combination of ANN's being a universal approximator including breakthrough in CNN models alongside new and improved GPU models, makes the creation of an NN – based control algorithm a reachable goal.

End to End Deep Learning:

The most standardised approach towards solving the problem of autonomous driving is to break it into several levels of abstraction such as lane marks detection, pedestrian detection, Traffic signal detection, path/motion planning and motor control. This approach to autonomous driving merges all the techniques mentioned above into one simple elegant algorithm. The image below best illustrates the two concepts:

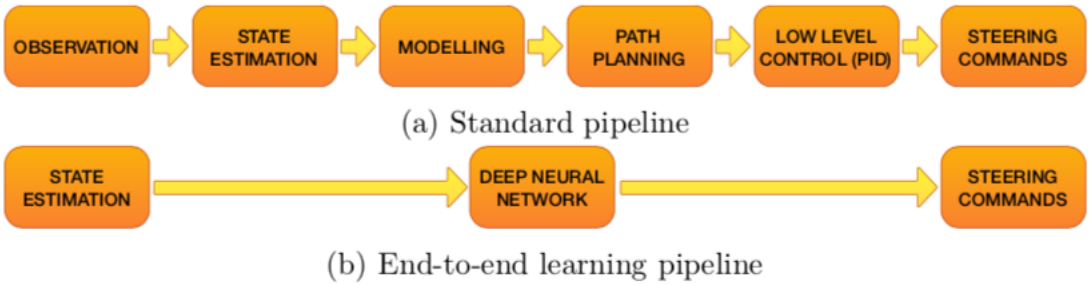

fig: Difference between modularized and AI approach

The first ever self-driving car built with neural network (ALVINN) was done by Pomerleau and it can be dated back to 1989. He employed a single NN with one deep layer composed of 29 nodes. His research greatly facilitates the idea of proposing a solution to the problem of autonomous driving employed miscellaneous end to end neural networks. In 2004 Defence Advanced Research Projects Agency (DARPA) came up with the open source project known as DAVE (DARPA Autonomous Vehicle) which is significantly small RC car project that uses NN to drive within a terrain an simultaneously avoid obstacles.

This project has revolutionised NVidia's self-driving cars project. Within the paper "End-To-End Learning for Self-Driving Cars", NVidia provides a state of the art Deep feed forward end to end CNN model built specifically for self-driving vehicles. The testing of their model displayed promise while it drove on the highway with a controlled amount of traffic.

Real world self-driving cars:

Companies that are conducting research on self-driving cars in modern times are not really interested in an end to end deep learning model as collecting large datasets of images of real world scenarios can be often times unrealistic. It was shown within a study conducted in 2016 that every case where extreme precision is required, amount of training data to train an end to end model based on that case will increase exponentially compared to the modularised model. Within the modularised model, the problem is scaled down to a number of separate modules consisting different levels of abstraction where lane track and or pedestrian detection for example are implemented using multiple ML and or manual engineering algorithms to yield the maximum accuracy.

Another important difference between real world self-driving cars and the RC car used within the scope of this thesis is in the form and size of the input. Companies like google waymo or tesla accumulate data from a wide variety of sensors to concisely precisely localise the vehicle within the environment. These accumulated data is then combined to create a visual model of the vehicles surroundings.

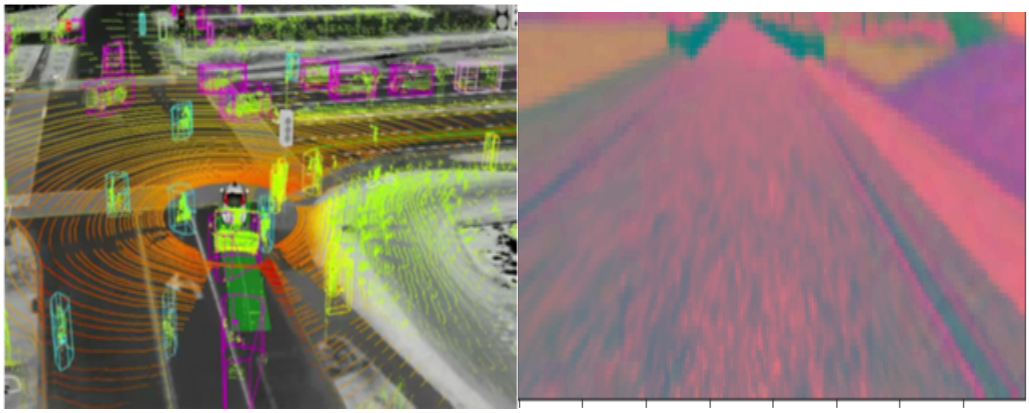

fig: Waymo Sensor Fusion/ Image segmentation with imgaug

Expensive technologies such 360-degree Lidar along with an array of cameras and ultrasonic sensors are being used to generate 360-degree model of the environment. Tesla model S uses 12 ultrasonic sensors, 5 camera's (one at the back and four facing front) and a radar sensor to track the speed of the object in front of the vehicle.

fig: tesla model S

Such vehicles also employ Bayesian SLAM algorithms with extended Kalman filters, in turn combining all the sensor data to generate the 360-degree sensor fusion image.

Thesis Model:

The Goal of this thesis will be to implement an RC car that can autonomous drive itself within an artificially designed track with no traffic. As it is very different to the real world Model, the NN architecture will not complicated by employing a modularised model. Although modularisation will still be implemented in a smaller scale by training a model to predict the steering angle and the other to predict traffic signals. It will then be integrated within the RC car's main control loop along-side certain control logic to drive the vehicle autonomously within the well-defined track. This provides an opportunity to display the strength and clarity of an end to end model when applied to the problem of autonomous driving.

Principles of Deep Learning:

This chapter aims at introducing the reader to the concepts of Deep learning and the benefits of CNN. This chapter begins by explaining the structure of an artificial neuron. Then the reader is introduced with the concepts of feed forward neural networks followed by the theoretical foundations of CNNs which will be heavily used within the practical part of this thesis.

Basics of an Artificial Neural Network :

Similar to the biological half, the artificial neural networks consist of a variety of interconnected nodes or often referred to as neurons that are able to compute their corresponding activation functions based on the activation functions of the preceding layer and their weights and biases.

The underlying idea is that many independent simple unit working together towards a singular goal can often mimic the existence of an intelligent whole much like the ants in many ways. One of the major concepts of deep neural networks is distributed representation or abstraction. The idea is that each input of a system is denoted as a collection of many features and each feature is considered to be a representations of numerous inputs. For example, to construct a classifier that can recognize books, copies and rubbers and each object can only be either red, green or blue. In an attempt to solve the problem, 6 neurons can be created where they can learn the concept of being an object or colour. Every time the number of objects and colour that needs to be identified is increased, the size of the model exponentially increases as well.

Whereas with distributed representations we can scale it down to just 6 neurons where 3 neurons learn the physical shape or aesthetics of an object while the rest learn the method of classifying between colours.

Artificial Neurons:

A comparison between an artificial neuron with a real neuron is discussed below:

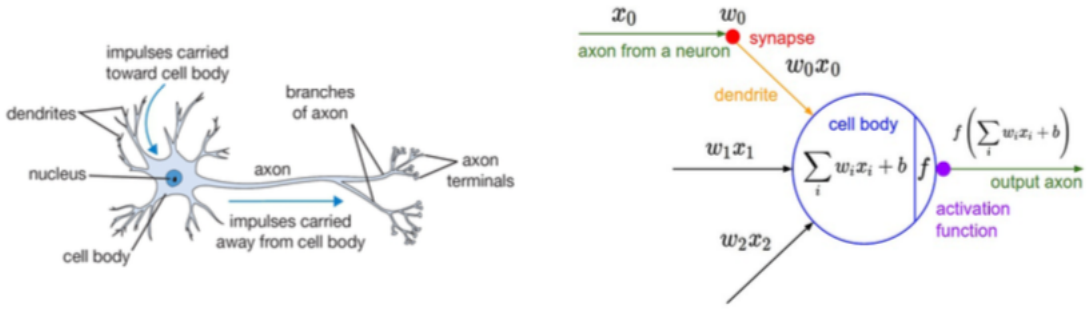

fig: a comparison between a biological unit of brain (left) and an artificial neuron (right)

The output of a single artificial neuron is calculated in the equation below, where x is the input, w is the weight associated with the input, b refers to the bias unit or threshold of that specific neuron and σ is the activation function of that neuron.

$$y = \sigma\left(w_i x_i + b\right)$$

Activation Functions:

In order to fit a non-linear set of input data, a non-linear activation function needs to be used. The most popular activation functions used in today's world are the sigmoid, relu (rectified linear unit), tanh (hyperbolic) and the elu function. They all have their own short comings and are used in various circumstances to fit various purposes.