CHAPTER 1

INTRODUCTION

An autonomous vehicle is a vehicle equipped for detecting its condition and working without human association. A human traveler isn't required to assume responsibility for the vehicle whenever, nor is a human traveler required to be available in the vehicle by any stretch of the imagination. A self-sufficient vehicle can go anyplace a conventional vehicle proceeds to do everything that an accomplished human driver does. Lane keeping is a key component for self-driving vehicles and it is important to perform this task safely and efficiently.

Autonomous vehicles rely on different kinds of sensors like ultrasonic sensor, LiDAR, radar and many others to obtain the inputs from its surroundings in order to take relevant driving decisions. Regardless of numerous sensors introduced on self-sufficient vehicles for example, radar, LiDAR, ultrasonic sensor and infrared cameras, the normal color cameras are still significant for their minimal effort and capacity to acquire rich data. Given a picture caught by camera, one of the most significant assignments for a self-driving vehicle is to locate the best possible vehicle control contribution to keep up it in path. The conventional approach separates the undertaking into a few sections, for example, path recognition, path planning and control logic, and they are frequently examined independently.

The path markings are typically distinguished by some image processing techniques, for example, color enhancement, Hough transform, edge detection, and so forth. Path planning and control logic are at that point performed dependent on the path markings distinguished in the first stage. Right now, execution profoundly depends on the highlight extraction and translation of the picture information.

As the pictures taken through camera are two dimensional it will be hard to decide the distance from obstacles so joining different sensors like LiDAR, ultrasonic sensors, infrared cameras and numerous others will improve the precision of the autonomous vehicle. This will make the autonomous vehicle more reliable and efficient.

CHAPTER 2

INPUT Sensors

A vehicle performs autonomous driving through a blend of sensors like LiDARs, cameras, wheel odometry and inertial measurement unit. The sensors used are as follows:

2.1 LiDAR (Light Detection and Ranging)

Lidar is a remote sensing method that uses light in the form of a pulsed laser to measure ranges to the Earth. A LIDAR instrument principally comprises of a laser, a scanner, and a specialized GPS receiver. Airplanes and helicopters are the most ordinarily utilized platforms for acquiring LIDAR data over broad areas. Two kinds of LIDAR are topographic and bathymetric. Topographic LIDAR typically utilizes a near-infrared laser to map the land, while bathymetric lidar utilizes water-penetrating green light to also measure seafloor and riverbed elevations.

LIDAR can likewise be utilized in any circumstance where the structure and state of Earth's surface should be known, and can even gauge a few gases and particles in the air. Its flexibility and high goals give it applications in antiquarianism, climate monitoring, city planning, meteorology, mining, and considerably more.

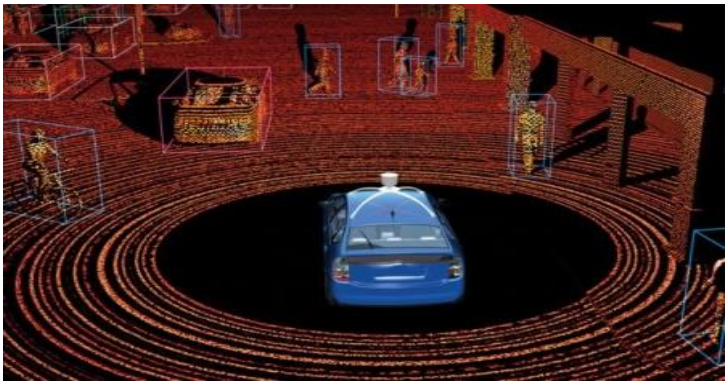

Fig I: Visualization of a LiDAR point cloud

2.2 Hall Effect Sensor

A Hall Effect sensor is a gadget that is utilized to gauge the magnitude of a magnetic field. Its yield voltage is directly proportional to the magnetic field quality through it.

Hall Effect sensors are utilized for proximity sensing, positioning, speed detection, and current sensing applications. Frequently, a Hall sensor is combined with threshold detection so that it acts as and is called a switch. Commonly seen in industrial applications such as the pictured pneumatic cylinder, they are also used in consumer equipment; for example some computer printers use them to detect missing paper and open covers. They can also be used in computer keyboards, an application that requires ultra-high reliability. Another use of a Hall Sensor is in the creation of MIDI organ pedal-boards, where the movement of a 'key' on the pedal-board is translated as an on/off switch via Hall Sensors.

Fig II: Hall Effect Sensor

2.3 Radar

The fundamental separation of radar is that it utilizes radio waves rather than a laser to detect objects. This enables radar to measure velocities of surrounding objects straightforwardly, offering a basic preferred position in the car condition. LiDAR frameworks would need to depend on an intricate examination to accomplish a similar result. What's more, when radar waves travel through the air, less force is lost contrasted with the light waves, which means radar can work over longer separations. Radar has also already been in use for years for powerful military purposes in airplanes and battleships.

Radar maintains functionality over all climate and lighting conditions. Nonetheless, the technology has customarily been constrained by low goals, a hindrance that made radar defenseless to bogus alerts and incapable to distinguish stationary

items. As of recently, that is. The technology has advanced into high goals abilities as of late.

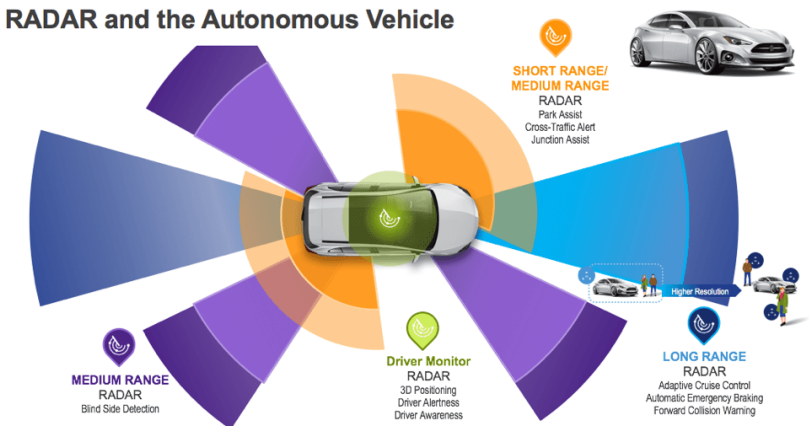

Fig III: RADAR and autonomous car

2.4 Cameras

From photographs to video, cameras are the most precise approach to make a visual portrayal of the world, particularly with regards to self-driving vehicles. Autonomous vehicles depend on cameras set on each side — front, back, left and right — to join together a 360-degree perspective on their environment. Some have a wide field of view — as much as 120 degrees — and a shorter range. Others center on a progressively restricted view to give long-range visuals. A few cars even integrate fish-eye cameras, which contain super-wide lenses that give an all-encompassing perspective, to give a full image of what's behind the vehicle for it to park itself.

Fig IV: Camera

CHAPTER 3

ALGORITHM

3.1 Cone driving algorithm

1. procedure CONEDRIVE (cones in range)
2. init steering range to [-25,25]
3. for all cones in range do
4. evaluate collision range with cone
5. exclude the collision range from steering range
6. end for
7. if steering range is empty then
8. Stop
9. else if all steering range _ threshold then
10. select largest steering range
11. else if all steering range > threshold then
12. select steering angle with minimum change in current direction
13. end if
14. drive toward center of steering range
15. end procedure

3.2 Road and Lane Detection

1. procedure LANEDETECT(histogram vector from camera)
2. undistort image frame from lens distortions
3. Sobel filter image frame as filtered image
4. threshold filtered image as binary image
5. obtain histogram vector for binary image on y-axis
6. split histogram vector into left half and right half
7. for each histogram vector do
8. find peak position on y-axis
9. init sliding window at bottom of image at peak position
10. while sliding window not at top of image do

11. find mass center of sliding window as line point
12. move window to the mass center
13. move window iteratively on x-axis towards top of image
14. end while
15. fit line point with second-order polynomial
16. end for
17. end procedure

CHAPTER 4

Navigation Systems utilized

4.1 Odometry

The SAE vehicle has been fitted with Hall Effect sensors on each wheel which send data through a comparator and an OR gate, and generate a pulse train to an Arduino Nano. The sensors count pulses for each wheel and report them to the low-level controller with timestamps. The linear velocity and rotational velocity are then evaluated by a low-level controller with the feedback from the steering sensor. The accumulated pose information is then combined with the wheel odometry of the SAE car. The goal for implementing this wheel odometry is to provide basic offline localization within a low-level system and to be further fused with other sensors such as IMU, LiDAR and cameras for a more accurate global positioning.

4.2 Dead Reckoning

In navigation, dead reckoning is the way toward ascertaining one's present position by utilizing a previously decided position, or fix, and propelling that position dependent on known or assessed speeds over passed time and course. Having an accurate state estimation is crucial for making optimal decisions for future control inputs to effectively navigate the environment. Dead reckoning on the vehicle is achieved through the Xsens MTi-G-710 [9], which is an inertial measurement unit (IMU) that is equipped with a global navigation satellite system (GNSS) running at 50 Hz. However, these sensors are susceptible to noise and imperfections which introduce uncertainty to the measurements. Hence, we introduce an extended

Kalman filter (EKF) to fuse data from these sensors with that from odometry using a model of the car's dynamics to obtain a more precise estimate of its state. The EKF linearizes these non-linear functions using a first-order Taylor series approximation, where it is approximated according to the following equation

$$f(u_k, x_{k-1}) \approx f(u_x, x_{k-1}) + \frac{df(u_k, \mu_{k-1})}{dx_{k-1}}(x_{k-1} - \mu_{k-1})$$

where u and μ are the mean and the estimate of x, respectively. X_k is the current pose at time instance k and f is a non-linear transition function that converts the past state to the current state, state x is composed of the car's x-y coordinates and orientation φ.

4.3 LiDAR System

The vehicle utilizes an array of Light Distance and Ranging (LiDAR) systems. This consists of a SICK LMS111-1010 [13] and an ibeo LUX 4 connected through an Ethernet switch to the Nvidia Jetson TX1. The LMS111 scans a single layer at 50 Hz while the LUX scans four layers at 10 Hz featuring inbuilt object detection and tracking. The data is published by each of the LiDAR's ROS drivers and processed to achieve desired functionalities.

The LUX is mounted above the driver and pitched towards the ground slightly such that the lower layer scans the ground 20 meters ahead of the vehicle. It is utilized to achieve road edge and object detection at a distance. Road edge detection is achieved by analyzing the depth information in one of the layers and checking it for both smoothness and slope. The central data points and those near them are considered and checked to confirm that they meet the slope criteria (the road should be relatively flat so no great changes in depth should be noted in a line). Iteratively, further and further points are considered in a stepwise process where the correlation coefficient is considered at each point. The road edge is the point at which the correlation coefficient is the highest whilst the slope condition is still being met. This approach was improved with the implementation of a Kalman filter which creates a time-averaged estimate of the road edge position assisting in the prediction of the current road edge.

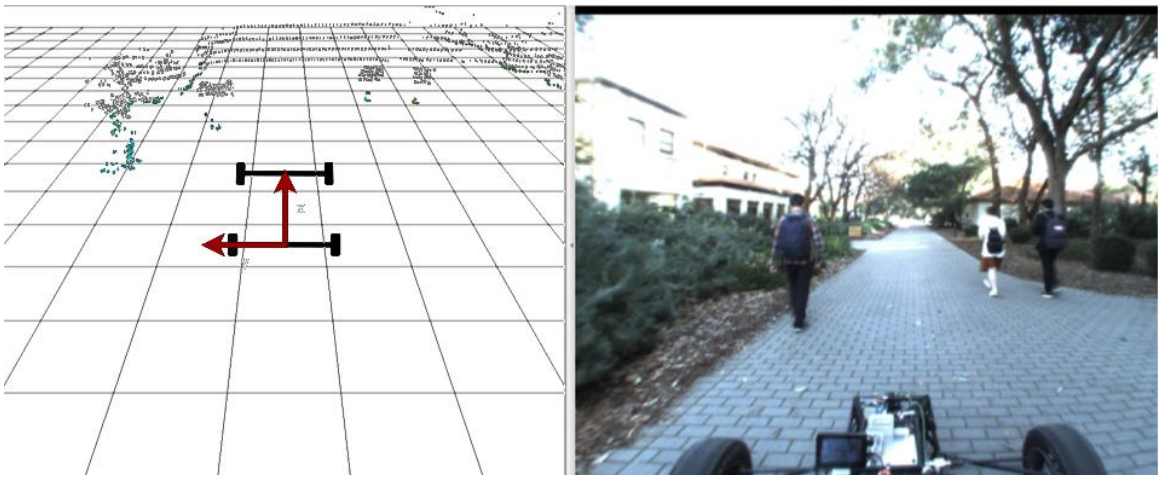

Fig V. Point cloud generated from the LMS111 (colored) and the LUX (4-layers, white) (left) and the scene where it was captured (right).

The in-built object detection and tracking of the LUX will be used for fusion with the obstacles reported through processing of the LMS111's data. The comparison of positions of objects reported by the LUX and LMS111 will increase the likelihood of detecting an object. The LMS111 giving data of the objects on a low and horizontal plane and the LUX giving data of objects in the mid to far range. In turn, this object information will also be fused with that of the camera vision.

4.4 Camera System

A pair of FLIR Blackfly GigE cameras are mounted on the vehicle's frame to perform tasks related to visual navigation, such as semantic segmentation and visual odometry. These cameras are fitted with Fujinon f/1.2, 2.8–8 mm wide aperture varifocal lenses , and are individually capable of capturing a wide field of view. To suit our application, these cameras use 1.3MP 1/3" global shutter CCD image sensors that will not be affected by any distortions caused by the rolling shutter effect. The cameras are connected to a Gigabit Ethernet switch that connects to the Jetson TX1, interfacing them through Blackfly's ROS node where it is configured to record at 10 Hz per channel. The application focuses on using monocular vision that is paired with measurements from the LiDAR system in order to achieve depth perceptions.

CHAPTER 5

Path Planning

It is important to design the way on which the vehicle needs to venture out to arrive at the destination. There are numerous techniques to accomplish path planning. The method utilized in paper [1] is explained beneath.

The control system was programmed to deliver path planning routines to drive either through a series of predefined waypoints, or in between a series of traffic cones placed on either side of the vehicle.

5.1 Waypoint Driving

The underlying idea behind waypoint driving is to drive a set of predefined points in between the starting position P1 (x = 0, y = 0 and orientation φ = 0) to the destination position P2, which can be obtained through the differences in GPS coordinates. These waypoints can either be stored in Cartesian coordinates in an array or in this test case, they are selected based on the driver's preference by selecting position points on RViz , which are then confirmed on the console. To ensure that all points can be driven smoothly with the consideration of the correct heading to the subsequent point, a spline approach has been implemented within this design to generate the desired path.

Once the waypoints are chosen, two static paths are shown on RViz The first path (green) consist of straight lines that interlink all the points with the arrow at the end showing the destination heading. The second path (purple) is the desired path which consists of a smoothed curvature that passes through all of the waypoints, the generation of this path is based on the Hermite spline interpolation technique with the required parameters which are the four vectors — current position P1, target position P2, tangent of departure from current position T1 and tangent of approach from target position, T2; along with four Hermite basis functions $H_n(u)$.

$H_1(u) = 2u^3 - 3u^2 + 1$, $H_2(u) = -2u^3 + 3u^2$, $H_3(u) = u^3 - 2u^2 + u$, $H_4(u) = u^3 - u^2$

where $0 \leq u \leq 1$ which represents the start to finish motion. We then construct the resultant path f by calculating the product between the vectors and the Hermite basis functions

$$f(x, y, \varphi) = H_1 P_1 + H_2 P_2 + H_3 P_3 + H_4 P_4$$

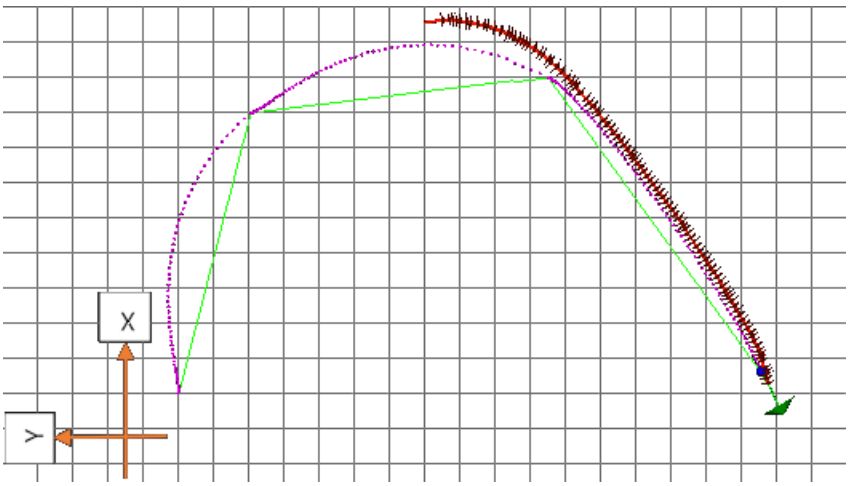

Fig VI: Calculated waypoints on RViz

The actual simulated driving pattern (red line) is determined based on the distance measurement between the current position of the vehicle against the desired path (purple) with a predefined tolerance range, and limited maximum turning angle ranged between $\pm 25^{\circ}$. The desired path is constituted by a finite number of points generated from the Hermite spline interpolation.

5.2 Cone Driving

The current iteration of the path planning procedure uses obstacle detection of the cones to determine the correct path. The cone driving module accepts cone locations from either the map, LiDAR or camera, classifying them as objects. Then, the vehicle navigates to drive within the track formed by cones safely without collision. Using a range of the maximum turning circle of the car, of both a left-hand turn and right-hand turn, it then looks at which predicted paths will intercept cones. The vehicle dynamics are thus limited during motion planning such that the steering angle does not exceed 25°. The algorithm will iterate through all cones within the car's range and calculate the best collision free path to undertake, as detailed in Algorithm 3.1.

Handcrafted features that combine a linear classifier are utilized for cone detection using OpenCV. The histogram of oriented gradients (HOG) descriptor across an image was utilized to find and segregate regions of interest (ROIs) that may encompass a cone, which is then used as inputs for a support vector machine (SVM) classifier. For all regions that are positively classified, the hue layer is threshold with an orange value, as our system is benchmarked using orange cones. We finally apply a histogram to the threshold image and then obtain the position of the cone within the image frame. However, in order to fuse this classification result with other sensors, the detected cones must be presented in the global reference frame. This is done by applying a perspective transform to the image, and with the assumption that the vehicle is driving on a flat plane. The position of the cones in the global frame can then be obtained by projecting these cones onto a horizontal ground plane.

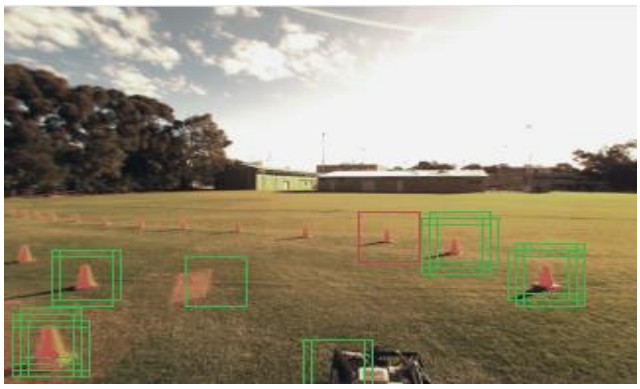

Fig VII: Visual cone detection showing the detected cones in bounding boxes.

CHAPTER 6

Road and Lane Detection

The system uses either semantic segmentation or edge detection to detect road edges and lane markings, depending on the environment's complexity. Using semantic segmentation also enables obstacle recognition which can be integrated with the LiDAR system. Environments are complex when there is a lack of uniformity in pose, features and illumination.

While it is possible to solely rely on modules within OpenCV here, the performance of using handcrafted features alone for image recognition is restricted by variations in image quality, brightness, and contrast. In order to improve its performance under these environments, we selected SegNet for semantic segmentation due to its high compatibility and ease of implementation. Its ability to perform pixel-wise classification for road scene objects complements the drawbacks of a single image processing scheme.

The architecture of SegNet uses a convolution encoder and decoder setup that classifies objects into one of the following classes—sky, building, column-pole, road-marking, road, pavement, tree, sign-symbol, fence, vehicle, pedestrian and bicyclist; with a class average classification accuracy of 65.9%. However, OpenCV is simultaneously used to perform image processing, with the first step being camera calibration to get an undistorted image. This is achieved using a chessboard image and finding its corners to get two accumulated list — a 3D point in real world space and 2D point in an image plane. We then use the camera calibration function in the OpenCV library to obtain the camera calibration and distortion coefficients.

Fig VIII: Semantic segmentation results during a test drive.

The road edges detection process finds lane markings at both sides of the car. It was noted that lane marking detection can also be performed solely using OpenCV functions, especially in non-complex, uniform environments with minimal illumination variations, thereby reducing its computation requirements. Algorithm 3.2 describes this approach.

CHAPTER 7

Significance of CNN

Given an image captured by camera, one of the most important tasks for a self-driving car is to find the proper vehicle control input to maintain it in lane. The traditional approach divides the task into several parts such as lane detection, path planning, and control logic, and they are often researched separately. The lane markings are usually detected by some image processing techniques such as color enhancement, Hough transform, edge detection, etc. Path planning and control logic are then performed based on the lane markings detected in the first stage.

When utilizing CNN, an end-to-end learning takes the raw image as input and outputs the control signal automatically. The model is self-optimized based on the training data and there are no manually defined rules. These become the two major advantages of end-to-end learning: better performance and less manual effort. Because the model is self-optimized based on the data to give maximum overall performance, the intermediate parameters are self-adjusted to be optimal. Moreover, there is no need to detect and recognize certain categories of pre-defined objects, to label those objects during training or to design control logic based on observation of these objects. As a result, less manual efforts are required.

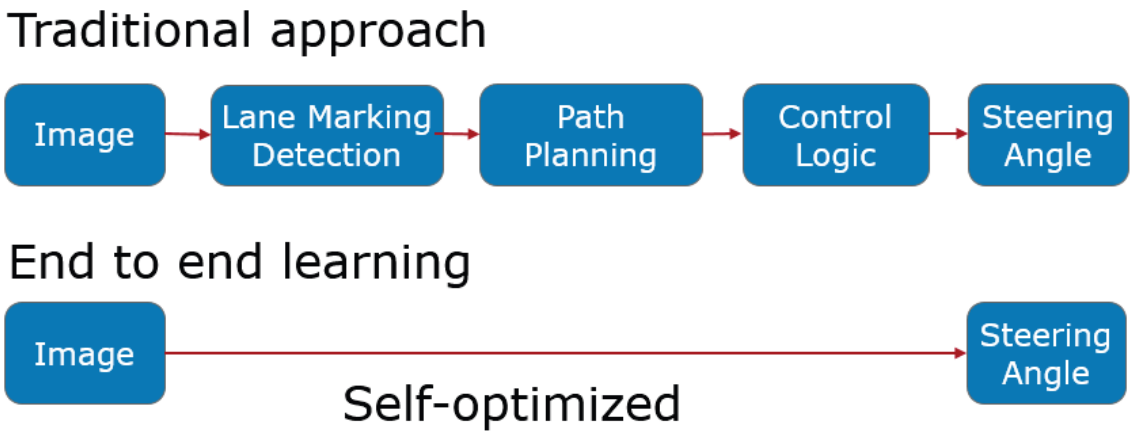

Fig IX: Comparison between the traditional approach and end-to-end learning.

Various steps associated with CNN implementation is as per the following:

- Data collection and pre processing

- CNN development
- CNN training using a part of the data collected
- CNN testing

The data used in the paper [2] are from comma.ai driving dataset. The dataset contains 7.25 hours of driving data, including 11 video clips recorded at 20 Hz and some other measurements such as steering angle, speed, GPS data, etc.

The image frames are of size 320 × 160 pixels, and are cropped from original video frames. The original frames are not provided by the dataset.

Fig X: An example of image frame from the dataset.

Before training the CNN model, the data need to be further processed. During the training stage, one important issue needs to be addressed that the data used for training is highly unbalanced as shown in Fig XI. As highway roads tend to be mostly straight and the portion of curved road is at a small percentage, the trained model based on these unbalanced data may tend to driving straight while still have low losses. To remove such bias, the data of curved roads are up-sampled by five, where curved roads are defined by where the absolute steering angle values are larger than five degrees. The data are then randomly shuffled before training.

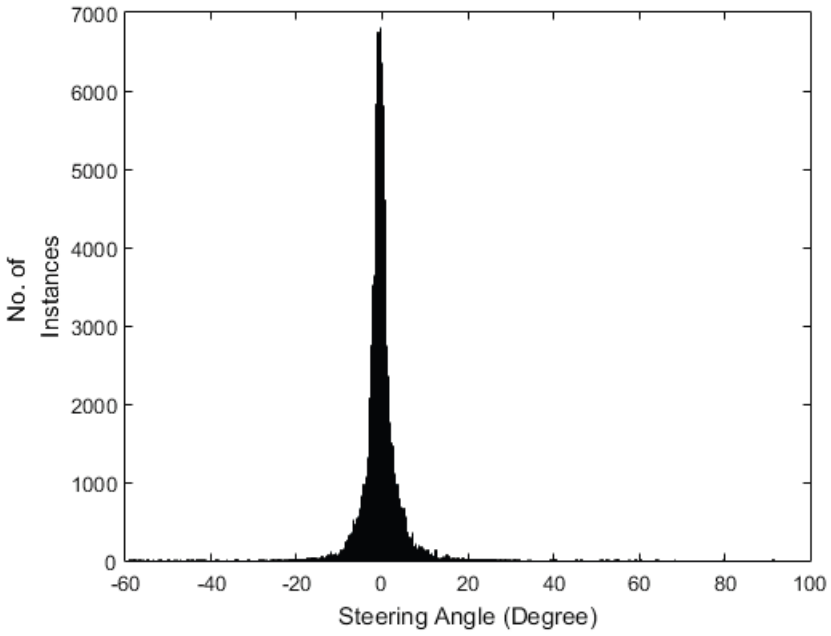

Fig XI: Histogram of steering angles in training data.

The CNN model consists of three convolutional layers and two fully connected layers. The input layer is raw RGB image, and output layer is the predicted steering angle for the input image. The first convolutional layer uses a 9×9 kernel and a 4×4 stride. The following two convolutional layers use a 5 × 5 kernel and a 2 × 2 stride. The convolutional layers are mainly for feature extraction and the fully connected layers are mainly for steering angle prediction, but there is no clear boundary between them since the model is trained end-to-end. Dropout layers are used for preventing over-fitting. There are no pooling layers because the feature maps are small.

The trained model was evaluated using two test video clips containing 25K frames. For each frame, the predicted steering angle is compared with the ground truth value.

CHAPTER 8

CONCLUSION

Lane Keeping is a significant viewpoint in autonomous driving. The software framework developed is for a high-level control system that is designed for autonomous vehicles that is both modular and scalable. These projects can therefore be low-cost while allowing users to adapt the software to the vehicle's and environment's needs. The system aims to be holistic by incorporating all the necessary modules required for autonomous driving, including sensor interfaces and fusion, localization, path planning, visual navigation and road detection. There are a few methods used to build up a system to perform lane keeping yet CNN ends up being the effective technique to execute this element. The end-to-end learning way to deal with lane keeping for self-driving autos that can automatically deliver legitimate steering angles from image outlines caught by the front-view camera is exceptionally efficient. The test outcomes show that the model can deliver generally exact steering of vehicle.

CHAPTER 9

Learning Outcomes

Lane Keeping is an indispensable piece of driving, so autonomous vehicles must be equipped with this function. Cameras alongside different sensors like liDAR, radar and so forth will make the vehicle accumulate the data increasingly successful and effectively. The collection of data is the fundamental part in training the software to turn out to be increasingly precise and reliable. It is imperative to have a collection of data in all the conditions of driving for the vehicle to perform better.

The convolution neural network is a proficient and less complex answer to actualize lane keeping in the autonomous vehicle. The traditional approach used for lane keeping divides the task into several parts such as lane detection, path planning, and control logic, and they are often researched separately. The Convolution Neural Network dispenses with the intermediate steps associated with the traditional approach and makes the procedure progressively less difficult. The Convolution Neural Network has different layers like the convolutional layers, fully connected layer, dropout layer, pooling layer, optimizers and activation functions. The convolutional layer is usually used for feature extraction, activation function is used to fit the output into a specific range and the fully connected layer is used to get the steering angle of the vehicle.