January 2019

Ros-Based Sensor Fusion And Motion Planning For Autonomous Vehicles: Application To Automated Parkinig System

Yuanzhe Li
Wayne State University, lyzkevin96@gmail.com

ROS-BASED SENSOR FUSION AND MOTION PLANNING FOR AUTONOMOUS VEHICLES: APPLICATION TO AUTOMATED PARKINIG SYSTEM

by

YUANZHE LI

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2019

MAJOR: ELECTRICAL ENGINEERING

Approved By:

_____

Advisor Date

_____

Co-Advisor Date

ACKNOWLEDGEMENTS

I would like to thank everyone who has supported me during the time I worked on my thesis.

I would like to express gratefulness to Dr. Caisheng Wang, Dr. Chin-An Tan and Dr. Hao Ying for serving as my committee.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES vi

LIST OF TABLES ix

1. CHAPTER I INTRODUCTION 1

1.1. System architecture 5

1.1.1. Overall diagram of ADAS system 7

1.1.2. Function blocks 8

1.2. Literature Review 10

1.2.1. Motivation 10

1.2.2. Literature review 13

1.3. Objective and scope of thesis 23

2. CHAPTER II SENSOR SELECTION AND SENSOR FUSION ALGORITHMS 25

2.1. Criteria for sensor selection 25

2.1.1. Environmental conditions 25

2.1.2. Distance and Speed Tracking 26

2.1.3. Curve 26

2.1.4. Efficiency and energy consumption 27

2.2. Camera 28

2.3. Radar 29

2.4. LiDAR 34
2.5. Ultrasonic Sensors 37
2.6. Driver Status Monitoring Sensors 38
2.6.1. Required Hardware 39
2.6.2. Interfacing with Vehicle Systems and Current/Potential Issues 40
2.6.3. Sensor Limitation Impact on Driver Feedback 40
2.7. Localization sensor and IMU 41
2.8. Sensor fusion algorithm 41
3. CHAPTER III CASE STUDY 45
3.1. Automated perpendicular/horizontal parking 45
3.1.1. Perception 46
3.1.2. Object Detection Algorithms – Convolutional Neural Networks 55
3.1.3. Another Object Detection Algorithm – HOG and SVM approach 56
3.1.4. Sensor fusion 63
3.1.5. Route planning, behavior decision making and motion planning 69
3.1.6. Vehicle control 71
4. CHAPTER IV EXPERIMENTAL AND SIMULATION RESULTS 73
4.1. ROS Environment setup 73
4.2. MATLAB Simulation Environment 76
4.3. Experimental results 79

4.3.1. Surround view camera experiment 79

4.3.2. GNSS vehicle tracking experiment 81

5. CHAPTER V CONCLUSION AND FUTURE WORK 84

5.1. Conclusions 84

5.2. Future work 85

5.2.1. Algorithm improvement 85

5.2.2. Use cases in other ADAS feature scenarios 85

REFERENCES 95

ABSTRACT 101

AUTOBIOGRAPHICAL STATEMENT 103

LIST OF FIGURES

Figure 1 Automated parking system [1] 2

Figure 2 ADAS Architecture 6

Figure 3 ADAS Architecture in details 7

Figure 4 Sensor mount locations 8

Figure 5 SAE J3016 Levels 0 - 2 of Driving Automation [9] 14

Figure 6 SAE J3016 Levels 3 - 5 of Driving Automation [9] 16

Figure 7 2007 DARPA Urban Challenger Winner [17] 19

Figure 8 ADAS sensor performance comparison [30] 25

Figure 9 multiple sensor coverage diagram (in-scale) 27

Figure 10 Spider chart of camera performance 28

Figure 11 Spider chart of Radar performance 30

Figure 12 Bosch Mid-Range Radar SNR vs. Distance [31] 33

Figure 13 Spider chart of LiDAR performance 35

Figure 14 Spider chart of ultrasonic sensor performance 38

Figure 15 Spider chart of Driver Status monitoring performance 39

Figure 16 Planned ADAS System Structure 42

Figure 17 HMI Display UI and Tablet 44

Figure 18 Automated parking use case 45

Figure 19 Vehicle dynamics model 46

Figure 20 Picture of commercial parking lot taken from webcam 48

Figure 21 Different types of parking spots 48

Figure 22 Camera perception algorithm flow chart 49

Figure 23 Left, right, top, bottom, near and far in computer graphics [37] 50

Figure 24 Camera calibration results 51

Figure 25 HSV Color space [38] 51

Figure 26 Image after perspective transform and color space conversion 52

Figure 27 Picture after line detection 55

Figure 28 Heat map of vehicles in picture 58

Figure 29 Comparison between HOG+SVM and Neural Network algorithms vehicle detection results 60

Figure 30 On-road test of Neural Network algorithm 61

Figure 31 Partial results for evaluating performance of vehicle detection algorithms 62

Figure 32 Image of parking lot after lane detection and object detection 63

Figure 33 Distorted vehicles in surround view camera system [40] 64

Figure 34 Reference path generated by RRT 71

Figure 35 Diagram of the speed PID controller 72

Figure 36 Diagram of the feed-forward steering PID controller 72

Figure 37 Vehicle model built in Gazebo with different view angle 74

Figure 38 Surround view camera simulation 75

Figure 39 On-vehicle GPS data acquired from rostopic list 75

Figure 40 LiDAR detection result in simulation environment 76

Figure 41 RQT Graph of the simulation environment 76

Figure 42 Auto-parking Simulation result in different view angles 78

Figure 43 Lane Keeping Assist simulation based on ROS and MATLAB 79

Figure 44 Camera mounted on the vehicle for experiment 80

Figure 45 Parking spot and object detection in a parking lot 81

Figure 46 GNSS Sensor for vehicle localization 82

Figure 47 RTK base station near Warren waste water treatment plant on Google Map .. 82

Figure 48 Vehicle location GNSS signal during experiment 83

Figure 49 Simple HD Map creation using GNSS sensor 83

Figure 50 Adaptive cruise control: use case scenario on local roads 86

Figure 51 Adaptive cruise control: use case scenario on highways 86

Figure 52 Traffic jam assist use case 87

LIST OF TABLES

Table 1 Luxury Vehicle ADAS Feature Summary [44] 4

Table 2 List of functionalities needs to be achieved 10

Table 3 Perception and Sensor Fusion Algorithm in ADAS Features 12

Table 4 Specifications of front view camera on autonomous vehicles 28

Table 5 Preferred front Radar specs [31] 30

Table 6 Comparison among different types of LiDARs 35

Table 7 Preferred front LiDAR specs [31] 36

Table 8 Performance and processing speed of two algorithms 58

Table 9 Description of sensor fusion output parameters 68

1. CHAPTER I INTRODUCTION

The automotive industry is currently putting high focus on Advanced Driver Assistance Systems (ADAS) and automated driving vehicles. ADAS helps drivers by conducting complex driving tasks in different scenarios, even mitigating dangerous situations. Currently, ADAS is widely used in production and development vehicles. ADAS such as AutoPilot can achieve partial automation that allows the vehicle to drive by itself on highways, even under traffic jam. Companies such as Waymo and Uber already started to use their fleet as autonomous taxis [11].

By sensing the surrounding environment and creating an up-to-date world model, the vehicle will be able to establish spatial and temporal relationship between the vehicle and environment. The world model created in real time consists of two parts: kinematic information of the ego vehicle, and the objects around the vehicle, such as velocity and location, etc. for both ego vehicle and objective vehicle. The information provided by the world model will be further used for path planning and by the control units to generate potential paths and maneuvering the vehicle. To create the world model using perception and sensor fusion, localization of the ego vehicle and object tracking are two of the most important tasks. Multiple sensors mounted on the vehicle are responsible for localizing the vehicle and detect/track objects. For example, Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU) acquire the location of the vehicle, and help define the start point and track the vehicle coordinates trajectory. While cameras, Radars and LiDAR are used for object detection and tracking. Sensor data fusion is also a crucial part of the world model realization. The sensor fusion algorithm collects data from multiple sensors mounted on the vehicle. By processing and fusing the data from

the sensors, the output of the sensor fusion algorithm is a list of moving objects to be displayed in the world model. In this thesis, the sensor fusion algorithm will not be processing raw sensor data. Instead, detected but untracked object list will be processed using the sensor fusion algorithm to generate an associated object list with a high confidence level.

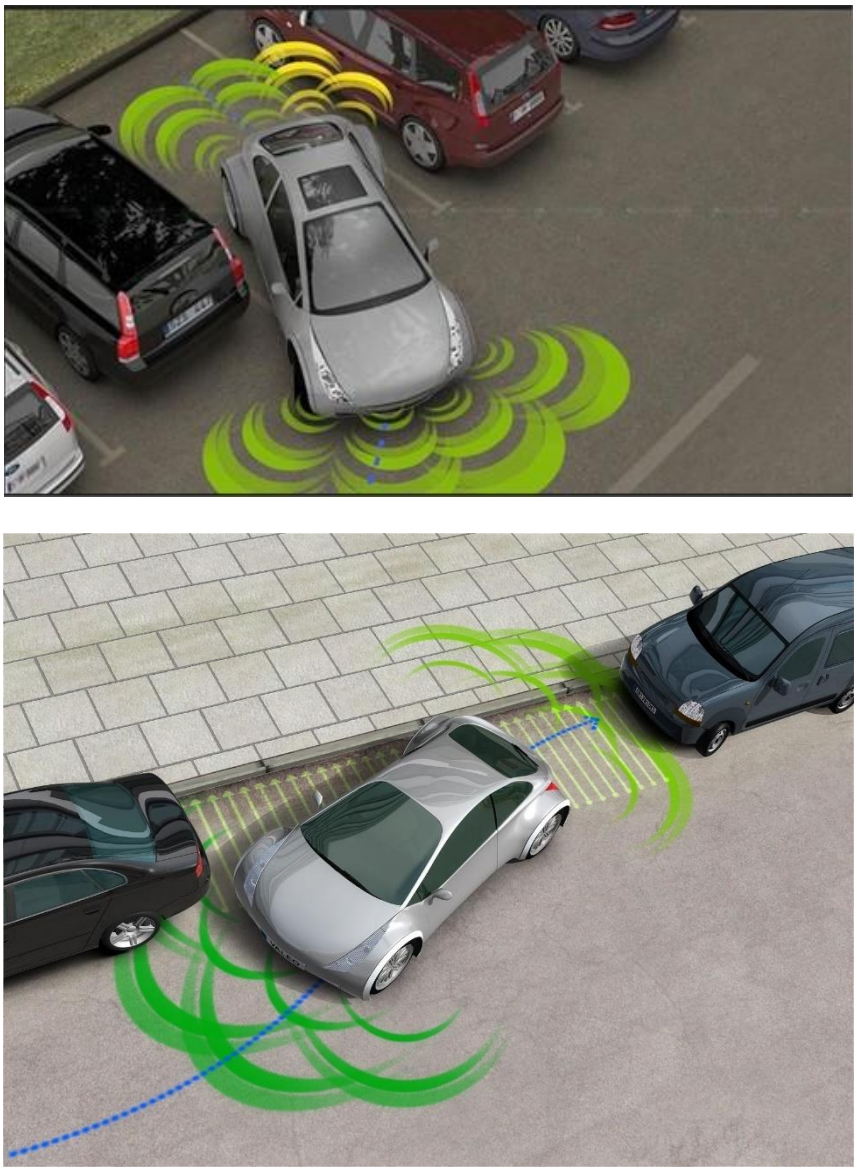


Figure 1 Automated parking system [1]

Auto-parking system is a relatively new feature in ADAS. Auto-parking can be

generally divided into two categories: Auto Parking Assist and Auto Parking Valet. Currently the auto-parking system in an ADAS can only be considered as a partial Auto Parking Assist, such as perpendicular and parallel parking. The system is less automated than other features such as Adaptive Cruise Control, Traffic Jam Assist, etc. The reason is that an auto-parking system requires more a sophisticated sensing and data fusion process [2, 3]. In an auto-parking system, the world model usually gets an object list from perception sensors, such as surround-view cameras and ultrasonic sensors. Vehicle position and velocity are sent to the world model in real time, wheel encoders and IMU are usually responsible for localization and velocity measurement. Current sensor fusion algorithm cannot meet the requirement to fuse data in order to get accurate and precise location of the vehicle as well as the surrounding environment. For auto-parking application, vehicle equipped with cameras and ultrasonic sensor fusion has limited information about the location and speed of the ego vehicle. Accumulative errors will cause the vehicle to depart from the previous planned trajectory [4]. Thus, an improved method for eliminating the accumulative error in vehicle localization should be developed.

Most of the luxury vehicles sold are equipped with multiple ADAS features. Regardless of the system performance, the common ADAS features of popular vehicles on the market are listed in the table below. Adaptive Cruise Control (ACC) and Lane Keeping Assist (LKA) have become standard on almost all luxury vehicles and some mid-class vehicles. Traffic Jam Assist/Pilot (TJA/YJP) are offered on limited models of luxury vehicles due to its performance stability as well as cost benefit issues. Even though all vehicle manufactures require drivers to pay attention to the road as a mandate

while these features are enabled, they still help drivers with great driving experiences and significantly increased driving safety, especially for long-distance driving.

Sensor fusion is critical for ADAS features. The key of an ADAS is to detect the surrounding environment and make proper decisions to control the vehicle in order to achieve some specific functionalities. Sensors mounted on the vehicle acquire tons of data from the environment. These data from different sensors will not be useful unless processed by a sensor fusion algorithm. Automated Parking Assist (APA), requires the fusion of multiple sensors [24]. For example, APA feature needs all types of sensors used in other ADAS features. However, it requires much higher accuracy in order to control the vehicle in a relatively small area. APA is a very useful function that helps park or even take over the whole parking process. This functionality is relatively new and yet not many vehicles are equipped with it.

Table 1 Luxury Vehicle ADAS Feature Summary [44]

Test Vehicle	**ACC**	**LKA/LDW**	**TJA**	**TJP**	**HWA**	**HWP**	**APA**
Tesla Model X	yes	yes	yes	yes	yes	yes	yes
AUDI Q7	yes	yes	yes	no	yes	no	yes
Volvo XC90	yes	yes	no	no	yes	no	no
Jaguar I-pace	yes	yes	no	no	yes	no	no
BMW X5	yes	yes	yes	no	no	no	no
Infiniti QX50	yes	yes	yes	no	yes	no	no
Acura RLX	yes	yes	yes	no	yes	no	no
Cadillac CT6	yes	yes	no	no	no	no	no

MB AMG GLS63	yes	yes	no	no	yes	no	no

1.1. System architecture

An ADAS consists of multiple layers of perception, decision making and control. Sensors and sensor data processors are to perceive the surrounding environment via object detection and tracking algorithms. The sensor fusion processor analyzes the data collected from sensor data processor and outputs a moving object list with robust detection and tracking, feeding to the world model. The world model is also built and updated in the fusion processor. The decision maker is responsible for generating an optimized path based on the world model information. The dynamics controller has all the vehicle dynamics information and controls the vehicle to move along the pre-defined trajectory.

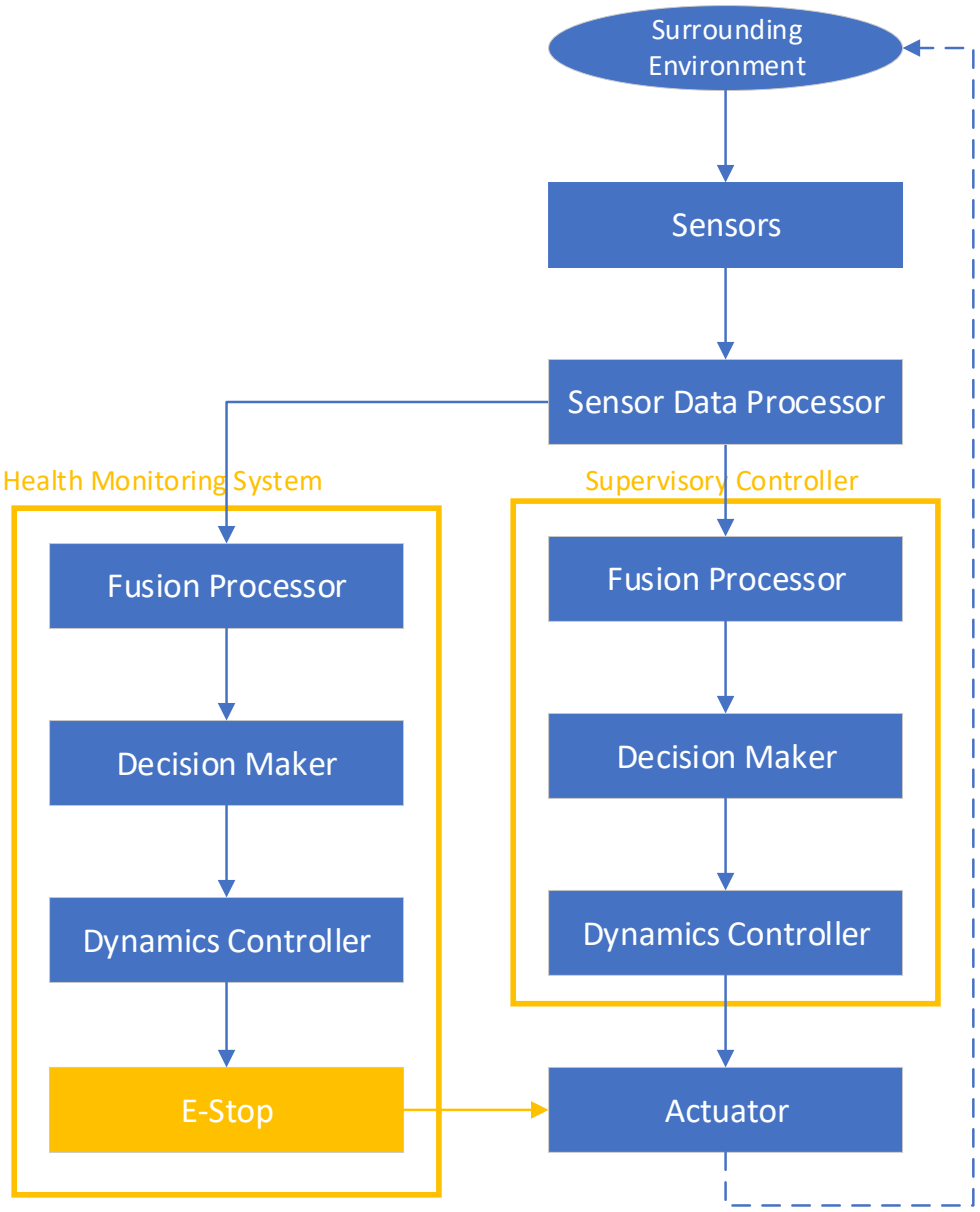


Figure 2 ADAS Architecture

The whole system is designed to have two major parts: the nominal system and the health monitoring system. Each part has its own fusion processor, decision maker and vehicle controller. When a failure occurs anywhere within the nominal system, the health monitoring system will take over the vehicle control and alarm the driver. The health monitoring system runs concise but robust algorithms that have same functionalities with

the nominal system. Actuators are the parts that control the vehicle, such as VCU throttle/brake controller and steering wheel encoder, etc.

1.1.1. Overall diagram of ADAS system

As previously introduced, an ADAS is designed to have six different components, along with a redundant health monitoring system for safety concern. As shown in the diagram below, each component has its own specific functionalities. The functionalities of each component will be discussed in detail in the following.

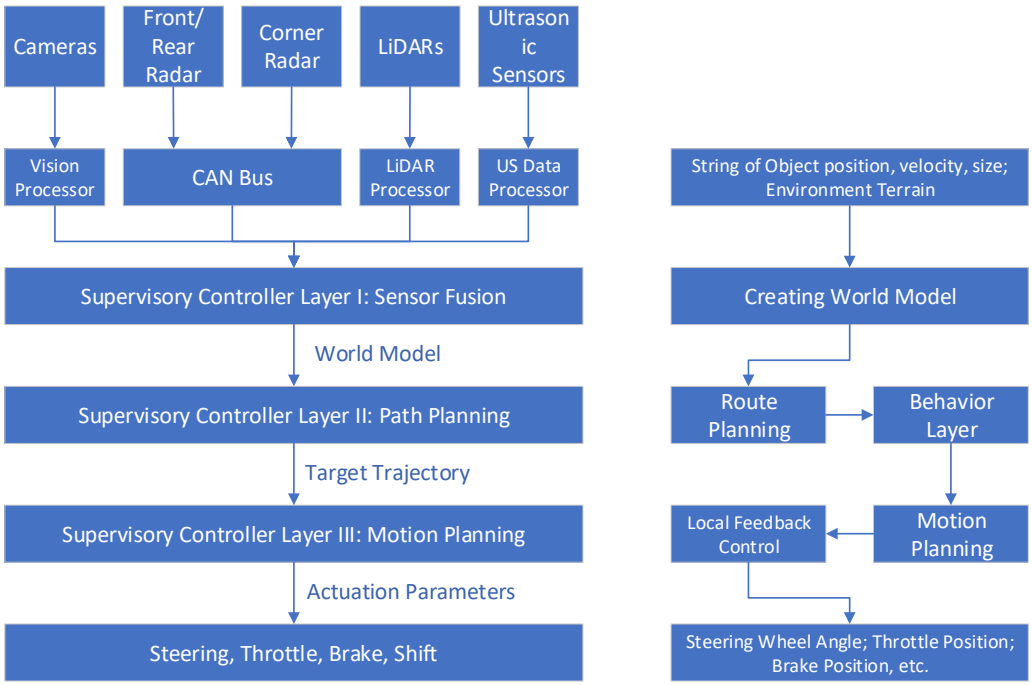


Figure 3 ADAS Architecture in details

System mounting location is shown in Fig. 7. Sensors and processors are marked in the diagram with different color labels. Cameras mounted on the wind shield are for the front object detection and tracking. With calibrated stereo camera we can do simple distance estimation. Radars are good at object detecting and speed estimation. Four of the corner radars are responsible for cross traffic detection, while the front and rear radars are for preceding and following object detection. The LiDAR mounted on the roof is used for

creating a machine-readable environment map by outputting point clouds, which can be used for benchmarking and reference. Sensor specification and sensor selection criteria will be discussed in Chapter 2.

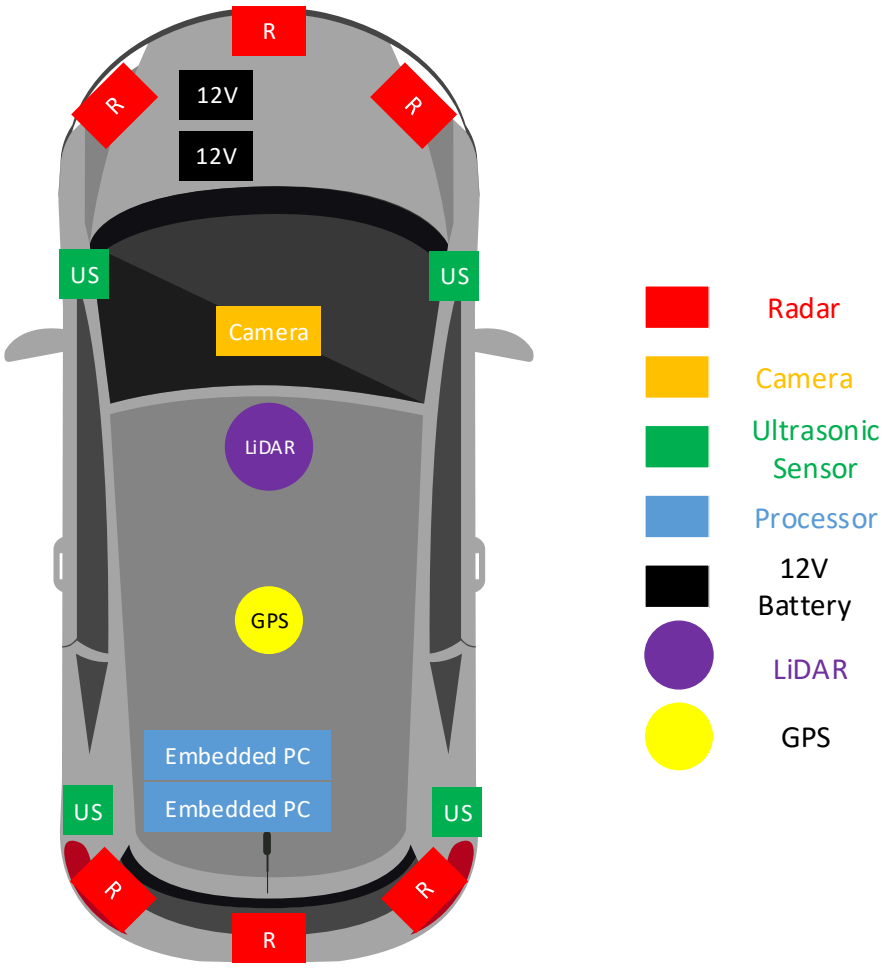


Figure 4 Sensor mount locations

1.1.2. Function blocks

1.1.2.1. Sensors

Sensors perceive information from the surrounding environment and feed acquired information to other parts of the system. Cameras, radars, Lidars and ultrasonic sensors are common sensors seen on autonomous vehicles.

1.1.2.2. Fusion Processor

The sensor fusion processor gets information from sensors and use sensor fusion algorithm to create a world model. The output of the sensor fusion processor is a list of moving objects. The objects on this list are fused based on sensor data and could be different from any object detected by sensors.

1.1.2.3. Decision Maker

The decision maker performs decision making and path planning based on the world model created by the fusion processor.

1.1.2.4. Dynamics Controller

The dynamics controller has all the information of the vehicle dynamics and inertial sensors. This controller is responsible for motion planning and vehicle dynamics control.

1.1.2.5. Actuator

Actuators are the mechanical or electrical parts controlled by the dynamics controller and maneuver the vehicle. Basically, actuators on autonomous vehicles are motors on steering wheel and brake pedals, and the ECU that controls throttle.

1.1.2.6. Health Monitoring System

Health monitoring system is a backup system of the nominal system. When an unexpected situation happens, such as sensor obscure, extreme weather, or nominal system malfunction, the health monitoring system is responsible for detecting errors and taking over control of the vehicle. Concise but robust ADAS algorithms run on the health monitoring system.

In ideal situations, the safety requirement should be achieved by the system by defining some parameters. For example, for ACC, the cruise follow distance, autosteer enabled/disabled should be defined. The table below lists the parameters to be defined

either by the system or the driver that sets a performance threshold for the system.

Table 2 List of functionalities needs to be achieved

Parameter Name	Options
Cruise Follow Distance	1 to 7 (follow distance is speed dependent)
Autosteer	Enable, Disable
Auto Lane Change	Enable, Disable
Speed Limit Warning	Off, Display, Chime
Speed Limit	Relative, Absolute
Speed Limit - Offset	Offset [mph], 0 to 5mph
Forward Collision Warning	Off, Late, Medium, Early
Lane Departure Warning	Enable, Disable
Automatic Emergency Braking	Enable, Disable

1.2. Literature Review

1.2.1. Motivation

Autonomous vehicles represent a fast-emerging technology to improve mobility as they offer a safer and more comfortable driving experience for users. The development of automated driving systems has focused on multiple features and the use of multiple sensors such as cameras, radars, LiDARs, ultrasonic devices, and combinations of sensor technologies to provide useful information from the environment to automate or assist various dynamic driving tasks, which are generally classified into mobility and safety. Automated driving spans a wide range of automation levels — from advanced driver assistance systems (ADAS) to fully autonomous driving being planned for the near

future.

An important objective of ADAS is to provide essential information to assist drivers to ensure a safe driving experience. These preventive measures for both vehicle occupants and pedestrian protections must take place before the actual impact or accident occurs. Existing triggering and assistive systems such as the electronic stability program (ESP) use sensor data to identify vehicle instabilities and provide fast preventive actions. However, in some accidents the ESP is not able to indicate any abnormal driving dynamics [5]. A common scenario is when a driver is making a right turn and focusing his/her attention on on-coming vehicles from the left, while forgetting pedestrians who might be approaching the same intersection to cross the road. Other accidents may involve pedestrians being distracted while using smartphones and not paying full attention to on-coming traffic while crossing at intersections. Based on the traditional technologies, no signal will be triggered and sent to assistive systems such as the brake assist under these situations to prevent the accidents. A warning system to identify potential accidents early enough to allow the driver or an assist system to take appropriate actions would thus be desirable. The warnings can be either audio or by some LED display (e.g., green and red indicate potential and imminent accidents, respectively).

For vehicles equipped with ADAS features, to deliver preventive safety applications as well as driver assist functionalities, it is required to have necessary information of ego vehicle status and surrounding environment. The development of perception and sensor fusion algorithms will be based on several ADAS features: APA, ACC, LKA and TJP. Perception and sensor fusion are critical parts of these ADAS features, as summarized in the table below.

Table 3 Perception and Sensor Fusion Algorithm in ADAS Features

ADAS Features	Realization Steps	Steps involving perception and sensor fusion
APA (Automated Parking Assist)	1) Localization 2) Detect parking spots 3) Path planning 4) Actuation 5) Go back to 1	1, 2
ACC (Adaptive Cruise Control)	1) Localization 2) Detect lane, preceding and nearby vehicles 3) Path planning and decision making 4) Actuation 5) Go back to 1	1, 2
LKA (Lane Keeping Assist)	1) Detect lanes (bird-eye view algorithm) 2) Calculate lane center 3) Calculate position deviation 4) Calculate steer angle (steer angle – yaw rate map) 5) Apply steering wheel 6) Go back to 1)	1, 2, 3

TJA (Traffic Jam Assist)	1) Localization 2) Detect lane, preceding and nearby vehicles 3) Path planning and decision making 4) Actuation 5) Go back to 1	1, 2

The objective of this research is to develop novel sensor fusion algorithms for auto-parking application in autonomous vehicles. Sensor fusion algorithms can help improve the perception accuracy and vehicle localization accuracy. Incorporating motion planning algorithms and vehicle control methods, the auto-parking feature can be realized.

1.2.2. Literature review

Over the past few decades, autonomous driving has been an important research topic. Autonomous vehicle can be defined as vehicles that can move from a start point to a given destination, without the need of any interaction from the driver [6]. The autonomous vehicle is able to move in space with an environment model created by detecting information from the surrounding environment, as well as inside the vehicle cabin. According to the US National Highway Traffic Safety Administration (NHTSA) [7] and the SAE International standard, two main levels of autonomous classifications have been developed. As discussed in Chapter 1, the SAE defined classification method was accepted by NHTSA and became the publicly accepted [8].

The SAE standard divides the automation into 6 levels, from Level 0 – no automation to Level 5 - Full self-driving automation. All vehicles, including vehicles being sold on the market and prototype vehicles in the development stage, could be classified into the

above 6 different levels of autonomous vehicles.

Figure 5 SAE J3016 Levels 0 - 2 of Driving Automation [9]

1.2.2.1. Level 0 – No automation

Vehicles with no automation will be considered as Level 0 autonomous vehicles. In this case, the driver has 100% responsibility of controlling the vehicle. The driver must monitor the surrounding environment and maneuver the vehicle any time during driving. However, some warning system can be added to the vehicle, such as blind spot detection, front collision warning, etc.

Nearly all vehicles on the road today are Level 0 autonomous vehicles. Although a lot of new vehicles are equipped with different kinds of warning systems, these warning systems cannot control the vehicle. For example, blind spot detection, detects if there is any vehicle in the blind spot of the vehicle, but does not provide any suggestions to the driver, such as keeping or changing lane. These systems work passively and merely provide drivers with assistive information to make better decisions.

1.2.2.2. Level 1 – Driver-assistance

Vehicles equipped with driver-assistance functionality will be considered as in Level 1. The driver is fully responsible for the control of the vehicle, similar to Level 0. However, the car can take partial control of the vehicle in some specific scenarios, such as highway with clear lane markers. The driver must pay attention to the surrounding environment and be ready to take over control of the vehicle at any time.

Features such as Adaptive Cruise Control (ACC) and Lane Departure Warning (LDW) can be considered as Level 1 autonomous features. Level 1 autonomous vehicles already exist on market.

1.2.2.3. Level 2 – Partial Automation

Level 2 autonomous vehicles contains all Level 1 features, whereas Level 2 features contains more advanced automation, including Lane Keeping/Centering Assist, Automatic Emergency Braking, ACC with stop and go, etc. Similar to Level 1, these functions work only in specific conditions, such as well-illuminated highways and local roads. Level 2 vehicles will monitor the surrounding environment when certain features are turned on. When the vehicle detects obstacles or events that the system is not able to respond, the control of vehicle will be handed back to the driver immediately, with audio

or light signals.

Even though the vehicle can self-drive in some certain conditions, the driver is still responsible for the overall control of the vehicle and should pay full attention to the traffic all the time. Tesla's autopilot system was the first commercial Level 2 autonomous system on the market [10].

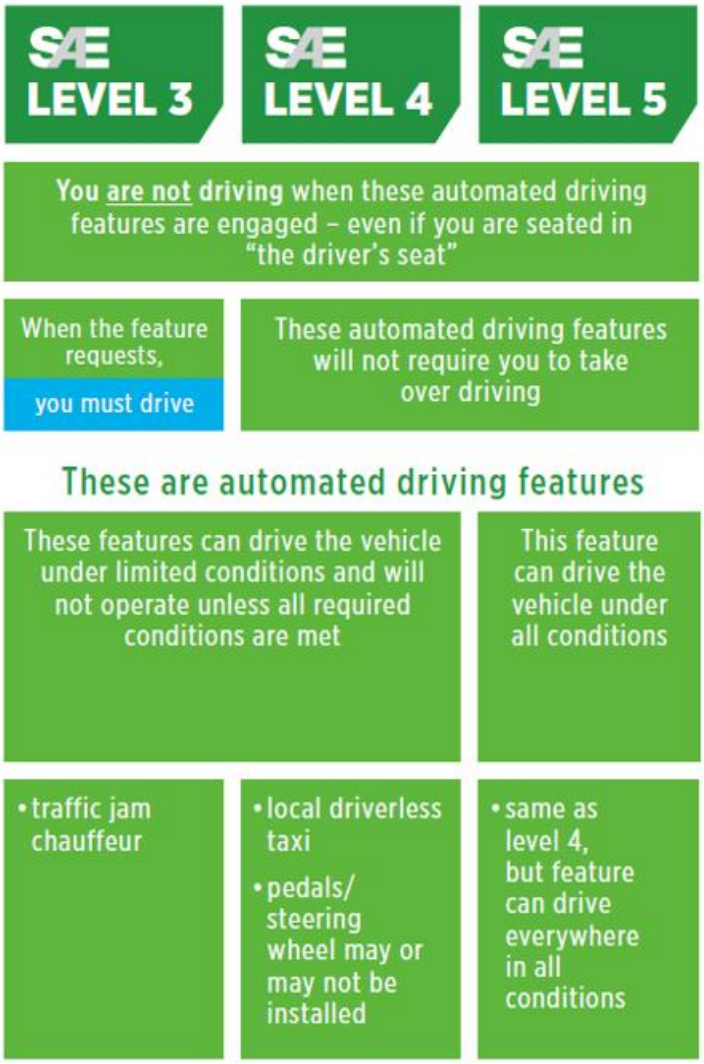


Figure 6 SAE J3016 Levels 3 - 5 of Driving Automation [9]

1.2.2.4. Level 3 – Conditional Automation

With more sophisticated functionalities, Level 3 vehicles can be considered as self-

driving vehicles under more complicated conditions. A vehicle with Level 3 automation is capable of monitoring multiple objects in the surrounding environment, such as traffic lights, pedestrians and traffic signs, etc. The vehicle can drive in most highways and part of local roads without involving the driver. In industry, an important criterion for Level 3 automation is that the vehicle can perform lane changing without the driver turning the blink signal [11].

The driver can rely on the vehicle in some scenarios; yet the driver should keep his/her hands on the steering wheel all the time as a safety protocol. Level 3 vehicles might still be affected by some extreme weather conditions. Some companies, such as Google and Uber [12] have already developed prototype vehicles that reach Level 3 automation in 2013 and is aiming at production. Automated Parking Valet (APA) is considered to be a Level 3 automation feature [11, 13].

1.2.2.5. Level 4 – High Automation

In this case, Level 4 autonomous vehicles can be switched to self-driving mode by the driver and perform self-driving in most road and weather conditions. The driver is not required to pay full attention to the surrounding environment, meaning that the vehicle can take full control of the steering/throttle/brake of the vehicle. If the system determines that there is an emergency situation that it cannot handle, warnings will be given to the driver and the driver will take over the control of the vehicle.

Tesla has claimed that the autopilot system is hardware ready for Level 4 automation [?]. However, due to legal issues, wireless software update for the autopilot system has not been released yet at this moment [?]. Auto Parking Valet (APV) is considered to be a Level 4 automation feature [11, 13].

1.2.2.6. Level 5 – Full self-driving Automation

Vehicles with full self-driving capability will be classified as Level 5. In this case, the vehicle can be designed with no driver. Steering wheel, shift lever, throttle and brake pedal can be removed from vehicle. The vehicle is able to drive itself with or without human inside the vehicle cabin. The vehicle is equipped with redundant autonomous system with multiple sensors mounted on the vehicle. The sensors will be able to provide precise information of surrounding environment in different illumination and weather conditions.

Furthermore, Level 5 automation vehicles support connected vehicle features, such as communicating with infrastructure and other vehicles using cloud. The vehicle will be connected to the cloud for traffic and road condition information, and HD maps. Software updates will also be downloaded from the cloud.

Numerous competition projects related to topics of ADAS and autonomous vehicles have been conducted by universities and research institutes since the late 80's. In 1988, Carnegie Mellon University's NAVLAB project vehicle was among the first that demonstrated ADAS features [14]. The NAVLAB vehicle was able to perform a lane center keeping (LCK) feature using computer vision algorithms. Research on autonomous vehicles has become a hot topic since them. A few decades later, in 2004, a famous competition – DARPA Grand Challenges rolled out. The objective of this competition for each team was to develop a driverless vehicle that is able to travel through deserts. Seven teams attended this competition, yet none finished even 5% of the full length of the course [15]. In 2007, DARPA Urban Challenge aimed at developing a self-driving car that can travel on urban roads, adding more complexity and challenge to

this competition [16]. A heavily equipped Chevrolet Tahoe developed by Carnegie Mellon University and General Motors won the competition.

Figure 7 2007 DARPA Urban Challenger Winner [17]

Starting 2009, tech-companies begin to develop commercial self-driving cars. Google being the first, Uber, Lyft and Tesla started to develop prototype self-driving cars. Among all these companies, Google was the first company to start the project and has the most experience on it, with about 8 million miles of accumulative mileage by the year of 2018. Tesla works more on marketing, as their 2016 Model S with Autopilot system became the first Level 2 autonomous vehicle on the market.

Sensor fusion is a critical method in perception system of autonomous vehicles. The definition of sensor fusion has been changing over the decades [19]. Several terms such as “sensor fusion”, “data fusion”, “multi-sensor data fusion”, “multi-sensor integration”, etc. have been prevalently used in literature to refer to data that is obtained from multi-

channel of information. Wald proposed the term “data fusion” in 1998 to be used as a summary of the terminologies listed above [20]. Dasarathy proposed to use a new term “information fusion”, which has not been used extensively before [21], as an overall terminology for fusion of any kind of data. Sensor fusion is a subset of information fusion, and the definition of sensor fusion is

“the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually”. [19]

Sensor fusion has been applied in autonomous driving systems and ADAS in multiple perspectives. Currently, autonomous driving systems use different levels of sensor fusion technology. For example, fusion architecture is divided in to 4 levels in a perception system of an autonomous vehicle: low level, map level, object detection level and track level [22].

- Low level: Raw data collected from sensors is converted into machine readable format. Detection provided by each sensor is fused in raw-data level to generate a map for further processing.
- Map level: Simultaneous Localization And Mapping (SLAM) is done in the map level with each sensor output. The generated maps from each sensor are combined to get a fused map. Notice that the map created in the map level is not the predecessor for object detection level. Actually, they are processed in parallel.
- Object detection level: In this level, specific objects will be detected from sensor outputs. Sensor will perceive the surrounding environment and provide a

list of moving objects. Combining lists from each sensor will increase object detection precision and accuracy. An enhanced list will be sent to track level for further processing.

- Track level: After obtaining a list of moving objects from the object detection level, the track level fusion algorithms will extract time features from the list and produce a list of tracked objects. Track level increases track accuracy and helps prevent false tracks.

An environment model is created after the four levels of fusion. A complete model contains two models from the four levels of fusion: one from low level/map level and one from object detection level/track level. The first one involves path planning and navigation functions, usually a HD map and a destination is required for this model [23]. Here we will only talk about the second model, which is the world model containing information of objects around the vehicle.

A variety of sensors can be put onto a vehicle. Due to some limitations, we choose only cameras as sensors. Cameras mounted on different locations of the vehicle will be integrated into a sensory system for parking spot detection.

Camera and ultrasonic sensor are the two major sensors that are widely used in automated parking assist systems. Ultrasonic sensors can evaluate distance from objects, while the video provided by cameras can be used for object detection [24].

Auto parking technology has already been used in production vehicles. Several OEMs started to equip their luxury vehicles with auto parking assist system [25]. Being the first to put auto parking on a production vehicle, Ford’s auto parking assist system is convenient indeed, supporting both parallel and perpendicular parking. However, like

most APA systems on market, it still has limitations. Most APA systems on market require the driver to be engaged during the whole time, while the only automatic part is the control of the steering wheel. The driver needs to drive back and forth to find the parking spot, control brake and throttle of the vehicle and pay attention to the surrounding environment.

During the recent years, many researches on improving the auto parking assist system were conducted. However, most auto-parking related systems are based solely on cameras. Cameras are used for parking spot detection and tracking during the parking process. The algorithm proposed were not robust enough for calculating some essential vehicle/environment states, such as the location of the vehicle, the location of the parking spot, the speed and heading of the vehicle, etc. C. Wang et al. proposed an auto parking system based on bird's eye view computer vision system in 2013 [26]. A camera-based parking spot detection and vehicle localization method was developed in this paper. The vehicle states were estimated using the camera data, without other sensor data.

In 2015, J. K. Suhr et al. proposed an indoor and underground parking space detection method using AVM camera image processing [27]. S. Lee introduced slot context analysis method to available parking spot detection [28]. Histogram of Gradients (HOG) algorithms is used in this method to extract features. A machine learning based available parking spot detection method was proposed by L. Zhang et al. in 2018 [29]. The research goal of this thesis is to develop a multi-sensor fusion method for auto-parking application.

1.3. Objective and scope of thesis

Research Question:

How to use sensor fusion to improve auto parking technology?

How to utilize path planning algorithms for auto parking?

Sub Questions:

What is the state-of-the-art of auto parking technology?

How is sensor fusion used in auto parking technology?

What does auto-parking system contain?

The objective of this work is to develop a vehicle-to-environment (V2E) system that would provide accurate detection of parking spots in a parking lot and planning a potential path to park the vehicle. The innovation of this system is that it utilizes multi-sensor with learning and recognition algorithms, including sensor fusion algorithms. Different sensors such as cameras, LiDAR, GNSS and ultrasonic sensors will be fused for better accuracy in terms of detection and localization. A series of road tests will be conducted to define an auto-parking assistance system and quantify the effectiveness of such a system in terms of successfully parking the vehicle into available parking spots.

The goal of sensor fusion is to adopt combinations of sensor technologies to provide accurate parking spot detection and provide potential maneuvering path to the driver. The system will use a combination of sensor technologies mounted on the vehicle and learning/recognition algorithms. Sensor fusion algorithm will determine an object list from raw sensor data. Fused objects, including vehicles, pedestrians and parking spot lines, will be integrated to the world model for further motion planning. Motion planning will work with or without a pre-defined map. A suggested trajectory will be generated as

an output of motion planning. The trajectory contains the route planned by path planning algorithms, as well as the speed profile of the vehicle. Perception and sensor fusion algorithms will be tested with a series of on road tests. Motion planning algorithms and vehicle controller will be simulated with visualization in MATLAB and ROS.

2. CHAPTER II SENSOR SELECTION AND SENSOR FUSION ALGORITHMS

Sensors are one of the most important parts in an ADAS. For a human driver, eyes, ears and other sensory organs such as tactile feelings are “sensors” for driving. These human “sensors” will perceive the surrounding environment to provide information to the brain for further processing. The responsibility of sensors in an autonomous vehicle is to perceive the surrounding environment for sensor fusion, motion planning and vehicle control. In this chapter, the criteria for sensor selection and sensor fusion algorithms will be established.

2.1. Criteria for sensor selection

Performance Aspect	Human	Radar	LiDAR	Camera
Object Detection	Good	Good	Good	Fair
Object Classification	Good	Poor	Fair	Good
Distance Estimate	Fair	Good	Good	Fair
Edge Detection	Poor	Poor	Good	Good
Lane Tracking	Good	Poor	Poor	Good
Visibility Range	Good	Good	Fair	Fair
Poor Weather Performance	Fair	Good	Fair	Poor
Dark or low Illumination Performance	Poor	Good	Good	Fair

Figure 8 ADAS sensor performance comparison [30]

2.1.1. Environmental conditions

- Extreme weather (heavy rain, snow, or fog): Reduces maximum range and signal quality (acuity, contrast, excessive visual clutter) for human vision, AV

visual systems (cameras, lidar).

- Excessive dirt or physical obstructions (such as snow or ice) on the vehicle: Interferes with or reduces maximum range and signal quality (acuity, contrast, physical occlusion of field of view) for human vision and all basic AV sensors (cameras, lidar, radar).
- Darkness or low illumination: Reduces maximum range and signal quality (acuity, contrast, possible glare from external light sources) for human vision and AV camera systems.
- Large physical obstructions (buildings, terrain, heavy vegetation, etc.): Interferes with line of sight for human vision and all basic AV sensors (cameras, radar, lidar).
- Dense traffic: Interferes with or reduces line of sight for human vision and all basic AV sensors (cameras, radar, lidar).

2.1.2. Distance and Speed Tracking

- Stereo cameras can detect distance, but detection accuracy isn't as good as radars.
- Radars were invented for speed tracking, especially acceleration, which cameras cannot easily detect. Even though speed information could be obtained using camera, speed sensor and IMU, more sensor fusion work is needed. The complexity of the system will also increase.

2.1.3. Curve

- Front radar is perfect for front object detection, but the performance of front corner will be reduced significantly when the vehicle enters a curve with quite

large curvature. In this case side sensors (either side camera or corner radar) are necessary.

2.1.4. Efficiency and energy consumption

- Energy consumption of radar could go as low as 5W each – according to Bosch MRR datasheet
- Detection result sent from radars to vehicle CAN bus could go at a speed of least 20Hz. Yet to speed up the processing speed of camera, resolution, in other words detection accuracy, will be compromised.

Sensors on autonomous vehicles are supposed to perceive information from the surrounding environment, mimicking human drivers. For autonomous vehicles, all areas around the vehicle are considered to be blind spots. Sensors such as cameras, Radars, LiDARs, etc. are to cover the blind spots and detect moving objects around the vehicle.

As shown in the diagram below, we can see that different sensors have different coverage areas. The large circle in blue is coverage areas of corner radars. The pie area in orange and red on the right are to be covered by cameras and front radars. The irregular shape on the left is rear radar coverage area. We will go into each type of sensor for detailed discussion on their specifications.

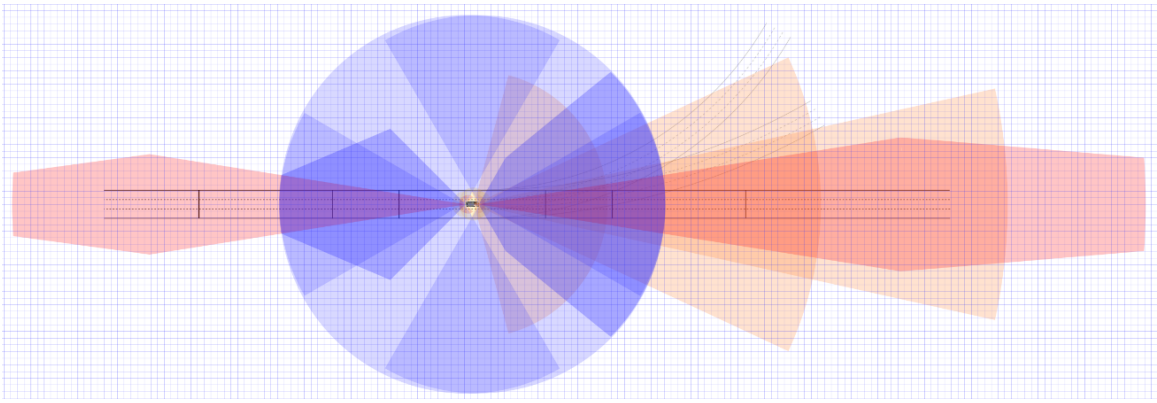


Figure 9 multiple sensor coverage diagram (in-scale)

2.2. Camera

Camera is one of the most important sensors in ADAS. Computer vision and machine learning algorithms can be used on camera captured video data for object detection and tracking. In autonomous vehicles, usually multiple cameras are deployed on different locations of the vehicle, such as front wind shield and side mirror, mimicking human driver's vision system. The vision system could be either mono- or stereo camera system.

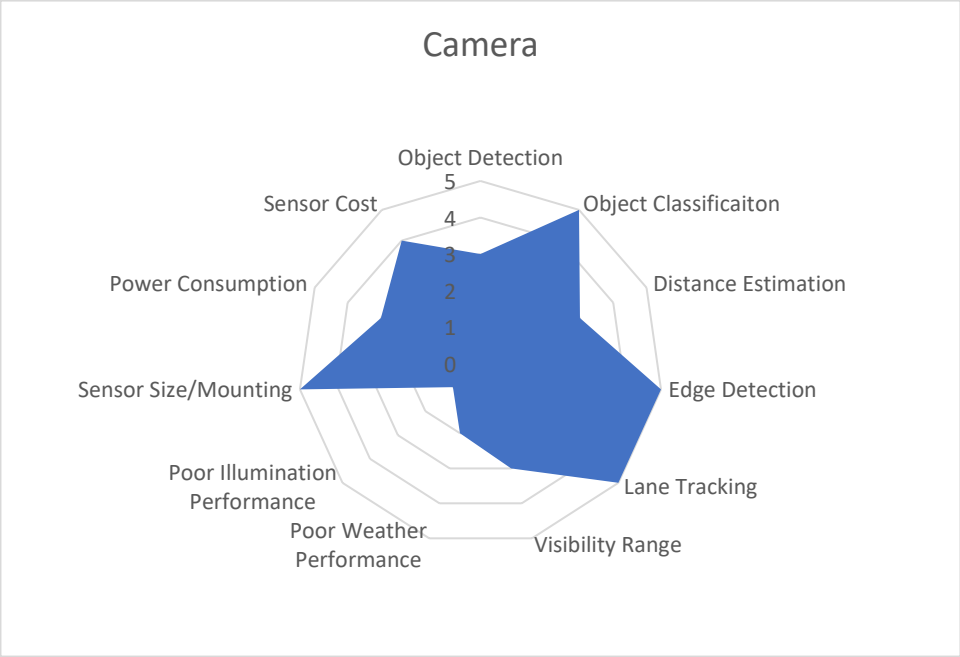


Figure 10 Spider chart of camera performance

Cameras equipped with object detection are capable of detecting objects of surroundings of the vehicle, such as vehicles, pedestrians. Due to the limitations of radars and LiDARS, road lanes and traffic signs can only be detected by cameras. The table below shows some preferred requirements of camera systems on autonomous vehicles.

Table 4 Specifications of front view camera on autonomous vehicles

Parameters	Preferred Value

Distance range	0.5…50 m
Detectable objects	Car/Truck/Bus/Motorcycle
Range resolution objects	0.5 m (@ 10 m range)
Range accuracy objects	0.2 m (@ 10 m range)
Angle resolution objects	2° (@ 10 m range)
Lateral object position accuracy	0.05 (@ 10 m range)
Speed range	±25 m/s
Speed resolution	1 m/s
Speed accuracy	0.2 m/s
Lateral line resolution	0.1 m (@ 10 m range)
Lateral center-line accuracy	0.05 m (@ 10 m range)
Azimuth angle field-of-view	±15°

The camera should have a detection range from 0.5m to at least 50m, between which pedestrians and vehicles might appear. The resolution of the camera might vary, depending on the computational capability of the vision processor. The higher the resolution, the higher the detection accuracy but also higher power consumption. Typical resolution for vehicle cameras is 800 pixels by 600 pixels and 1280 pixels by 720 pixels on development vehicles.

2.3. Radar

Radars, excellent range and range rate measurement sensor, play a very important role in autonomous vehicles. Radar generate electromagnetic waves and detect the reflection from objects. Although radars are not good at classifying objects – meaning they cannot estimate the contour of the object, they can measure object range and range rate

accurately. In extreme weather or poor illumination conditions, cameras might get blocked by rain or snow or even unable to capture pictures, while radar works robustly in these situations. Radars are widely used for adaptive cruise control and blind spot detection features.

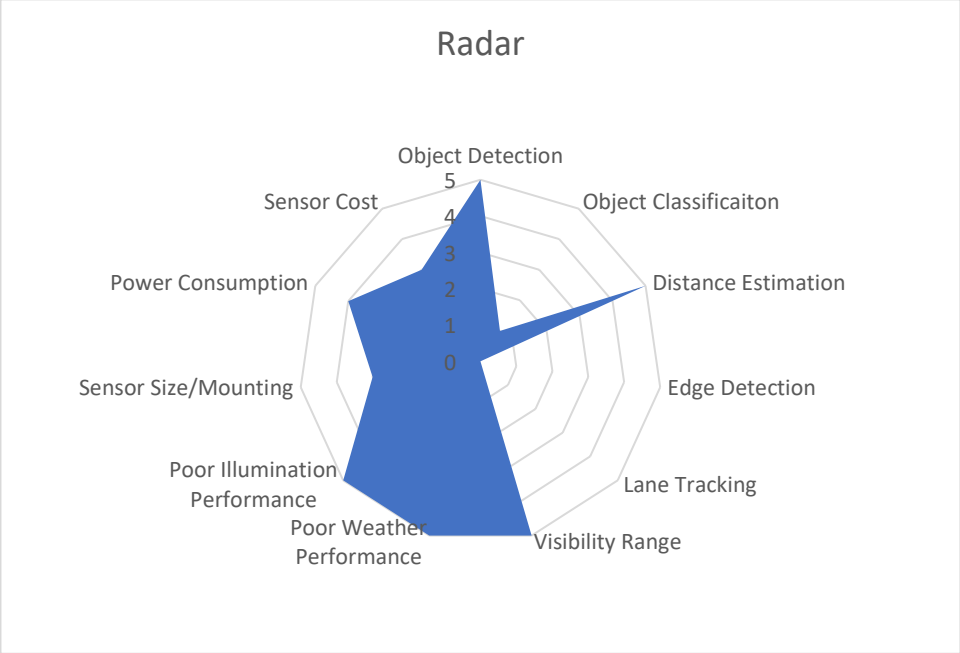


Figure 11 Spider chart of Radar performance

Table 5 Preferred front Radar specs [31]

Parameters	**Preferred Values**
Frequency range	76 ~ 77GHz
Azimuth Field of View	±7°; ±25°
Distance Range	0.5 – 100m
Distance accuracy (far range)	0.2m
Distance resolution (far range)	1m
Lateral object position accuracy	0.05m（@10m range）

Angle accuracy (far range)	0.3°
Angle resolution (far range)	1°
Speed range	-25m/s--+25m/s
Speed accuracy (far range)	0.1m/s
Speed resolution (far range)	1m/s
Acceleration estimation	0.2m/s^2 （delay<200ms)
Objects detectable	Car/Truck/Bus/Motorcycle
Object tracking	> 10 simultaneous tracked objects
Object classification	Type of object
Object size estimation	width
Object standing	Detection
Cut-in Cut-out detection	Left right
Update rate	50ms
Processing delay	< 100ms
Pitch angle compensation	-3°…+3°
Auto-alignment in azimuth	> 3°

In a typical ADAS system with front object tracking capability, a narrow FOV Mid-Range front radar sensor will be mounted on the center of the front fascia, together with a large FOV tri-focal camera [32]. The radar system can return object position with *dx* and *dy* coordinates, azimuth angle as well as object tracking list through CAN bus. The sensor fusion processor will be able to read radar detection data on CAN bus, merging the information from radar system with the graphic data to form a significant portion of raw

data used for sensor fusion.

Ideally, four wide-FOV short-range corner radar sensors will be mounted on the corners of the vehicle, are as follows.

Two front corner radars will detect whether if there are any potentially dangerous objects in the blind spot of the vehicle. The radars would be able to return the distance to the objects detected, and if either of the front corner detects the object is too close to the vehicle, the system will notify the driver by using audio or visual warning signals. The frequency of the warning signal will be decided according to the distance to the obstacles.

Two rear corner radars will be working on the rear object detection. When the vehicle is travelling at over 30mph (estimated), the two radars will detect rear objects. When objects are moving fast towards the vehicle, the system will notify the driver and potentially maneuver the vehicle to achieve safer driving conditions, such as releasing the brake or changing lanes. When vehicle is travelling under 30mph, the two radars will collaborate with ultrasonic sensor for rear-crossing traffic detection. For example, when the vehicle is parking, the radars will detect potential moving vehicles and pedestrians behind the vehicle. Warning will be given to the driver when distance between ego vehicle and object is too small.

The radar will improve the degree of sensor fusion. The sensor fusion process will be able to combine the objects' distance and position information from front radar system with the objects' position information from the camera graphics together. Based on detection results coming from radars, we can fuse the coordinate system of radar and camera to define a specific Region of Interest (ROI) for the camera. The camera only needs to run detection algorithm on the pre-defined ROI, filtering out irrelevant objects

such as trees, etc. Hence saving considerable amount of computational power compared to running detection algorithm over the whole image. This will significantly reduce the amount of data that object detection algorithm need to be working on, which will result in a higher processing speed, and reducing the safety concern due to data processing delay.

For example, mid-range radar systems usually include a relatively narrow FOV but a longer detection range [31, 33]. Short-range corner radars with a wider FOV to cover a wider but nearer area [33]. As shown in the following plots, the corner radar (left) detection area with a SNR over 40 is wider-spread while the front radar detection area with a SNR over 40 is narrower but longer. For the front radar, the estimated range limit would be around 100m.

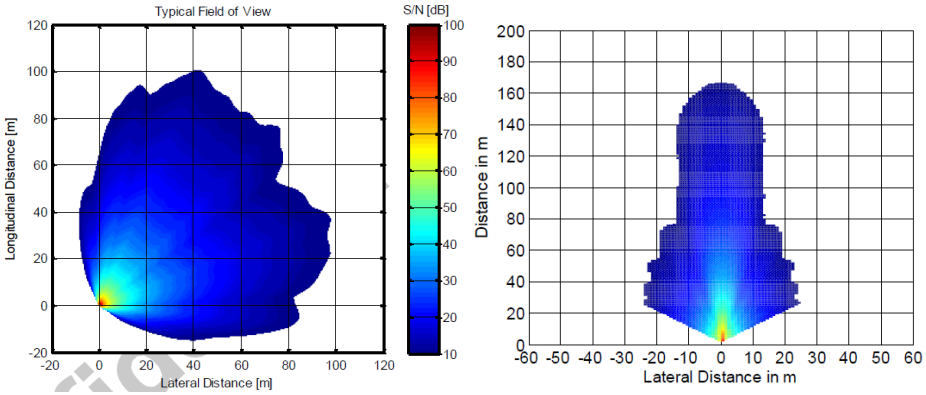


Figure 12 Bosch Mid-Range Radar SNR vs. Distance [31]

For distance detection and estimation, radar sensors are more powerful and accurate than cameras. The radar emits electromagnetic waves and, after a certain time interval, receives target object reflected waves, thus calculating the distance from the radar to the target object. In this way, the radar sensor is getting a clean distance value calculated simply by speed and time. By contrast, distance calculation for camera is not as “clean” as that for radar, even stereo cameras. Since the prerequisite for cameras to detect distance from an object is to have algorithms that know where bounding boxes for object

is in the frame. Note that the bounding box is always a rectangle, which means that not all pixels in the bounding box are part of the target object. Yet the distance calculation is usually based on every pixel in the bounding box or the size of the bounding box, which could be inaccurate due to the variation in object detector performance. A hardware-triggered stereo camera would definitely have a better performance in range and range rate detection but is still limited by the performance of object detector. In summary, the distance calculation of camera would always have some inaccuracies caused by the redundant pixels. As a result, we have expected our current temporary range and range rate detection approach, using a mono-camera for distance calculation, is not very accurate.

Another advantage of radar is that the returned targets' dimensions are more accurate than those from cameras, assuming a testing condition with relatively light interferences is provided. The targets' dimension calculations rely on the bounding boxes. And due to the limitation of object detector performance, the target dimension accuracy could also be limited. Whereas for radars, only a valid object surface would return electromagnetic waves that are used for dimension calculation. What's more, the cameras are vulnerable to bad weather conditions such as fog, heavy snow, rain, etc., whereas the radars are not as vulnerable.

On the other hand, radar sensors cannot distinguish the colors and patterns such as the lane marks, traffic signs, traffic lights, etc. which are all useful and important information for driver assistance systems.

2.4. LiDAR

LiDAR stands for Light Detection and Raging, LiDAR is a less common sensor

comparing with cameras and radars. LiDAR has never been deployed in a production car but is usually a standard choice on development vehicles. Many autonomous vehicle developers, such as Google, Nvidia, Baidu, etc., use LiDAR as their primary perception sensor. LiDAR can emit laser beams to measure object distance, outputting a point cloud map describing the surrounding environment. LiDAR is good for benchmarking and works as a reference data source for object detection and tracking.

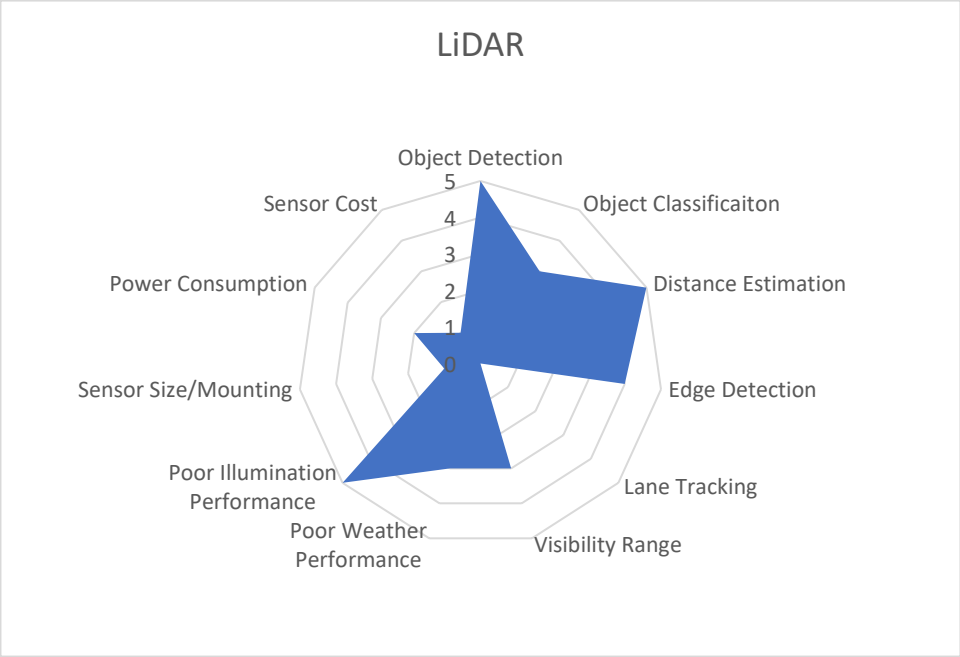


Figure 13 Spider chart of LiDAR performance

Currently there are three major types of LiDARs for autonomous vehicles: Flash LiDAR, Multi-beam LiDAR and Scanning Beam LiDAR. Each type has pros and cons, they are used in different ADAS applications and are usually mounted on different locations on the vehicle. Flash LiDAR and Multi-beam LiDAR are solid LiDARs.

Table 6 Comparison among different types of LiDARs

LiDAR Performance	**Flash LiDAR**	**Multi-beam**	**Scanning-beam**

Criteria		**LiDAR**	**LiDAR**
Resolution	High	High, in Parallel	Low, sequential
Range	Short	Far	Far
Bad weather performance	Scatter in fog	Good in all weather conditions	Scatter in fog
Power consumption	High	Low	High
Others	Confused by similar flashes	Not confused and no steering	Beam steering system needed

Table 7 Preferred front LiDAR specs [31]

Manufacturer	**Velodyne**	**Velodyne**
Model	VLP-16 Puck	HDL-64E S3
Radar Type	Scanning beam Lidar	Scanning beam Lidar
Channel Numbers	16	64
Measurement range	100m	120m
Accuracy	± 3 cm	± 2 cm
FoV (vertical)	+15° to -15°	+2.0° to -24.9° (26.9°)
Angular resolution (vertical)	2°	0.4°
FoV (horizontal/azimuth)	360°	360°
Angular resolution (horizonal/azimuth)	0.1° – 0.4°	0.08° – 0.35°

Rotation rate	5 - 20 Hz	5 - 20 Hz
Point output	Up to 0.3 million points/second	3D LiDAR Data Points Generated: - Single Return Mode: ~1,300,000 points per second - Dual Return Mode: ~2,200,000 points per second
Output Interface	100 Mbps Ethernet	100 Mbps Ethernet
GPS receiver	$GPRMC NMEA sentence from GPS receiver (GPS not included)	$GPRMC NMEA sentence from GPS receiver (GPS not included)
Power consumption	8 W	60 W
Operating voltage	9 -32 V DC	12 V – 32 V
Weight	830 grams (w/o cabling) / 590 gram (puck lite)	28 lbs. (12.7 Kg) (w/o cabling)
Vibration	5 Hz to 2000 Hz, 3G rms	Not Specified
Storage/Operating temperature	-10° to +60° C/- 40° to +105° C	-10° to +60° C/- 40° to +85° C

2.5. Ultrasonic Sensors

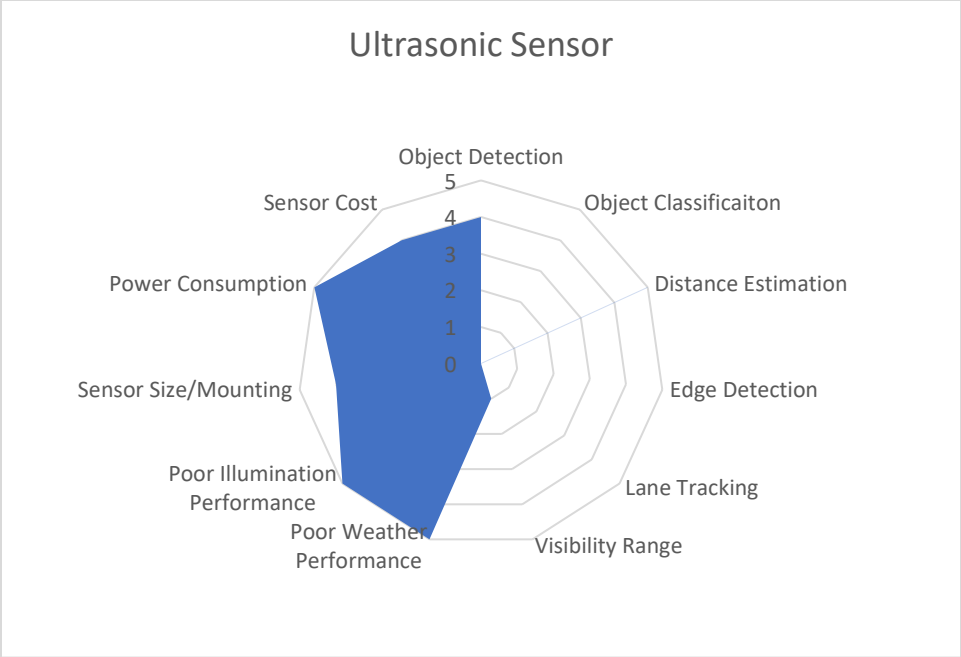


Figure 14 Spider chart of ultrasonic sensor performance

Ultrasonic sensors are commonly used in parking systems of vehicles. Similar to radars, ultrasonic sensors can detect object range by receiving and processing ultrasonic signal sent by itself. The frequency of each ultrasonic pulse is beyond human ear's hearing range, between 25kHZ and 50kHz. The physical model of the ultrasonic sensors can be represented by

$$\mathrm{p(x,w)} = -\frac{i_{wp}}{2\pi}\int_{s}^{1} v_z\,(y,w)\,\frac{e^{ikr}}{r}\,d_s(y)$$

where, in our case, p is the density of air, w is the angular frequency of the ultrasonic pulse transmitted by the sensor, v_z is the velocity of sound perpendicular to the transducer, r is the distance from the object to the transducer [34].

2.6. Driver Status Monitoring Sensors

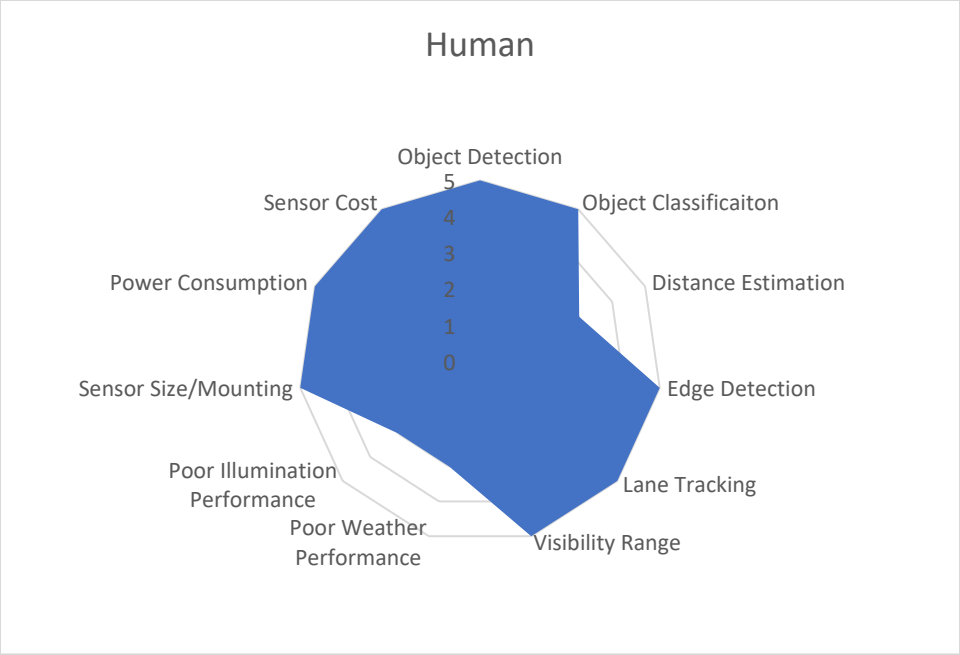


Figure 15 Spider chart of Driver Status monitoring performance

Development of a driver-status evaluation system is an important and popular topic in ADAS, in which the driver aggressiveness and drowsiness will be evaluated as a factor of deciding how much the ADAS system will become involved. An algorithm will monitor the acceleration and brake pedal signals and will generate an aggressiveness score every few seconds. Combining detection output of the sensor fusion algorithm, the ADAS will be provided with continuous reporting of front vehicle distance and speed, and the aggressiveness score on the driver, the decision on when to engage the driver alert will be made accordingly. Similarly, considering the distance and signal returned by the drowsiness detection module, the decision on when to engage the driver alert would be made.

2.6.1. Required Hardware

A decision-making system is needed to decide when and under what situations the driver need to be warned. The ADAS controller would be the decision maker. Some

driver feedbacks are realized using a display. The display should be small enough not to block driver's view while big enough to provide clear information to driver. For front collision warning, the feedback would be a red bounding box on the display plus alarm beeping sound emitted by a beeper. A driver drowsiness monitor is also planned to be used, and the feedback for drowsiness being detected would be a blinking red LED. To activate the LED and the beeper mentioned above, an I/O module that can be controlled by the ADAS controller is also needed.

2.6.2. Interfacing with Vehicle Systems and Current/Potential Issues

Signals indicating vehicle steering angle, steering angle ratio, vehicle speed, vehicle yaw rate and shift lever position are needed for Bosch radar sensors to work properly and have an optimized tracking functionality. The signal that indicates turning blinker status will also be needed for the lane departure warning system, to indicate if it is a driver intended lane departure activity if detected. Sports mode indication signal is needed for our ADAS application. When the car is in sports mode, the ADAS system will get less involved in the driver feedback.

2.6.3. Sensor Limitation Impact on Driver Feedback

For the camera, the lane departure warning function is based on the road lane detection, and an accurate lane detection algorithm is the prerequisite for a stable lane departure warning system. The driver feedback for lane departure is invalid if an accurate lane detection cannot be achieved. Scenarios like the driver is warned lane departure but the car is actually in the middle of the lane could happen. Under a poor illumination condition, the accuracy for lane detection and vehicle detection could be worse. The fact that the camera does not have a high enough resolution will make the far object detection

difficult. This could result in no data being usable as a cross check with radar returned data, thus increase the probability of having an invalid bounding box position for the driver feedback.

2.7. Localization sensor and IMU

Localization sensor and IMU are not new to autonomous vehicles, even conventional vehicles. They are not used for perception or decision-making algorithm, but for localization purposes. Unlike conventional vehicles controlled by human drivers, autonomous vehicle relies strongly on localization sensors. The accuracy of the world model depends on the accuracy of localization sensor. GNSS and IMU can also be used to verify if the vehicle is travelling along the pre-generated path within error tolerance.

The United States government currently claims 4-meter RMS (7.8-meter 95% Confidence Interval) horizontal accuracy for civilian (SPS) GPS. Vertical accuracy is worse. Autonomous vehicles usually require much higher accuracy. In this case, Real Time Kinematic is used for higher accuracy on most development vehicles [35].

2.8. Sensor fusion algorithm

Sensor fusion algorithms will be applied to mono-cameras mounted around the vehicle and GNSS sensor. The cameras are responsible for lane detection and object detection. Sensor fusion algorithm would be carried out on an ADAS controller, in our case a powerful laptop, that connects all the sensors.

Once the lanes are detected, lanes' coordinates will be sent from the vision processor to the sensor fusion processor within the ADAS laptop, where the lanes' coordinates will be transformed onto vehicle coordinate system. Meanwhile cameras will also send vehicle detection results to ADAS controller. Concurrently, vehicle detection results will also be

sent to the sensor fusion processor.

One critical issue for sensor fusion is time synchronization. The synchronized time stamp would ensure that the detections from each camera are from the same moment, thus to make sure the cross check between the output data from each camera is accurate.

There are two types of sensors, which we use for the ADAS, radars and the mono-camera. They work at the same time, thus measurements need to be taken for combining the data from different sources and get fused data, to enhance the robustness as well as the variety of useful information. As shown below, our ADAS system is a star-shaped structure and the master node would be the ADAS controller. The sensor fusion process is carried out within the ADAS controller.

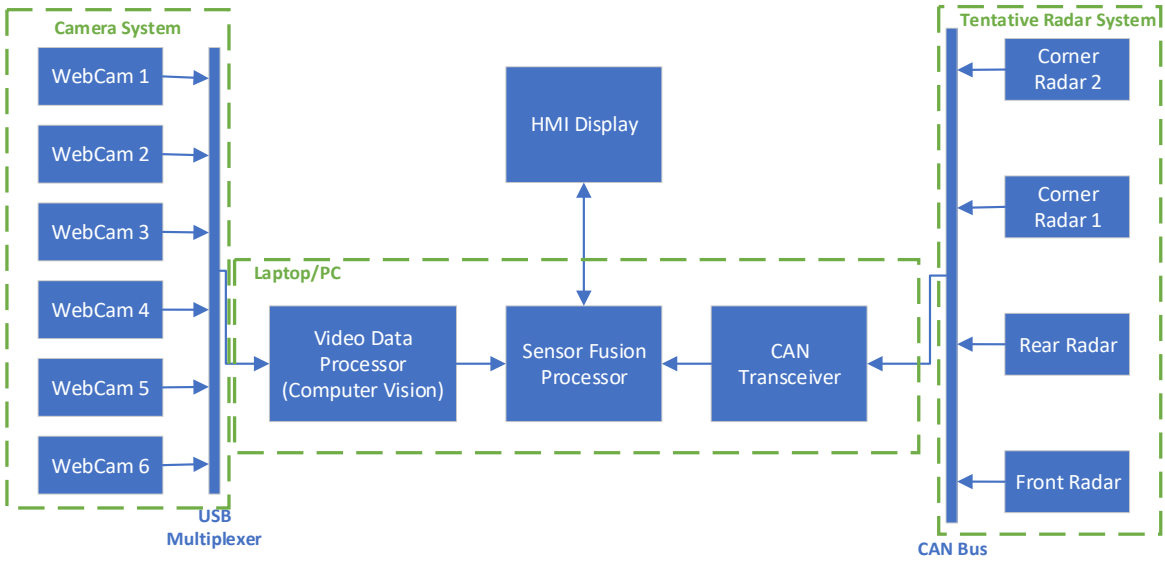


Figure 16 Planned ADAS System Structure

Our ADAS system would be a star-structured network and the controller would be the central node of the network. The controller would receive the CAN signals from every sensor as well as vehicle CAN network and would generate the CRCs for the radars. In addition, it would gateway all the vehicle dynamic information needed for radars. We

choose this structure with the following several considerations:

With this structure, we have more freedom of manipulating all the available signals that are currently needed for our ADAS system, as well as those are potentially needed for future algorithm modifications. A controller that supports model-based programming would greatly accelerate software development process for the following reasons: It gives us the convenience of choosing different programming method based on different applications. In addition, some algorithms are more suitable for State flow, which would give us a much more structured and intuitive UI. The ADAS controller would greatly help us in CAN communication. When the dbc file is loaded, the programming interface would give us the application level information directly and the user would know each signal name and routing straightforwardly. A dedicated ADAS controller would be responsible for all the sensor fusion algorithms and decision-making process, thus alleviate the calculation burden on the image processing unit, to which the frame rate and processing speed is a key factor.

The HMI display is designed to let the driver control the vehicle before and during the parking. Qt creator software and an Android tablet are used for creating the HMI display.

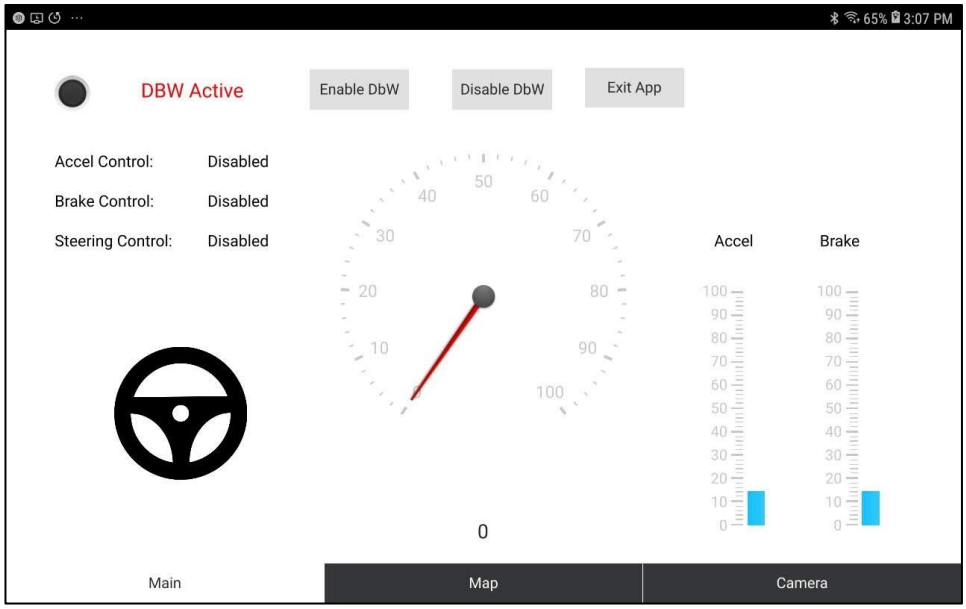


Figure 17 HMI Display UI and Tablet

3. CHAPTER III CASE STUDY

3.1. Automated perpendicular/horizontal parking

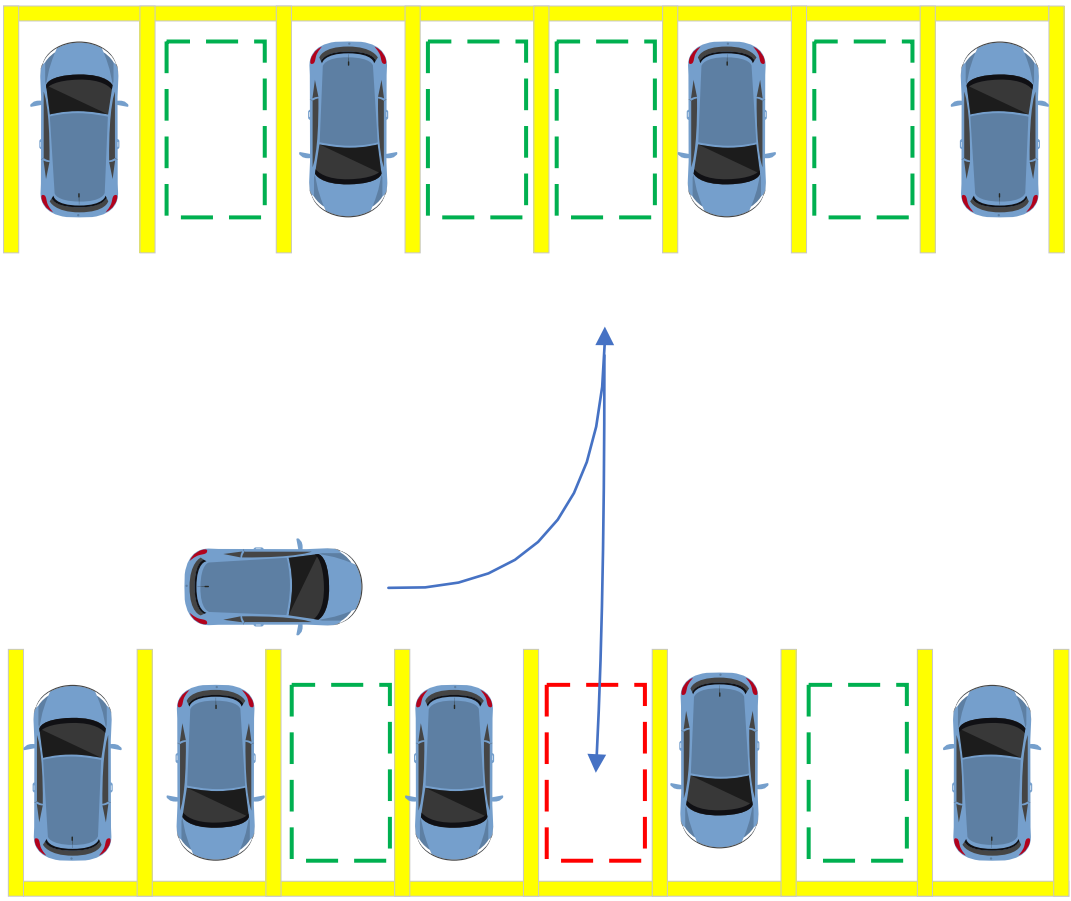


Figure 18 Automated parking use case

Auto parking system can be divided to 6 sub-systems. Perception, sensor fusion, route planning, behavior decision making, motion planning and vehicle control. A detailed illustration of each system will be conducted in this section.

Since the vehicle usually moves slowly during a parking process, we will use simple vehicle dynamics model to represent the vehicle for motion planning. Suspension dynamics will be ignored, a 3-DOF bicycle model will be used in motion planning analysis as well as simulation.

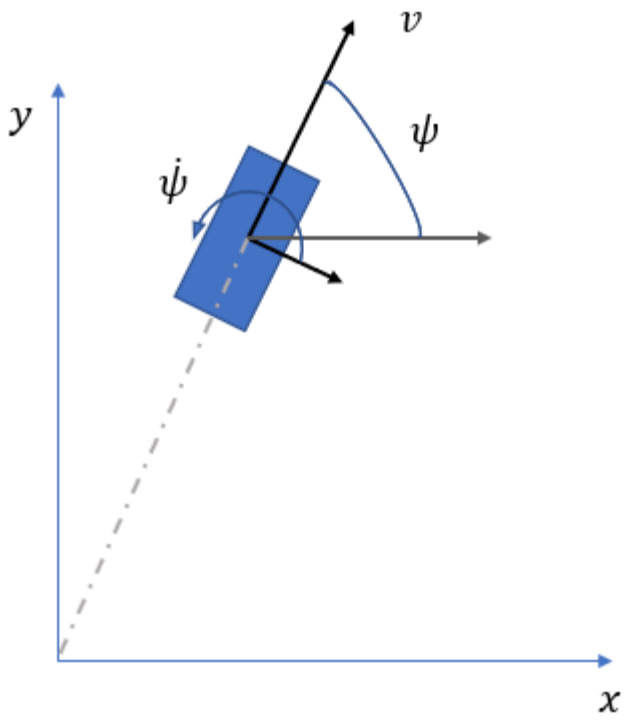


Figure 19 Vehicle dynamics model

The vehicle dynamics equation would be

$$\begin{bmatrix} x \\ y \\ \psi \\ v \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} x + \frac{v}{\dot{\psi}}(-\sin\psi + \sin(T\dot{\psi} + \psi)) \\ y + \frac{v}{\dot{\psi}}(\cos\psi - cos(T\dot{\psi} + \psi)) \\ T\dot{\psi} + \psi \\ v \\ \dot{\psi} \end{bmatrix}$$

In the equations, x and y are the reference point of the vehicle coordinate system, which is located on the rear differential of the vehicle; ψ is the heading angle of the vehicle; $\dot{\psi}$ is the yaw rate of the vehicle; v is the velocity of the vehicle; L is the wheelbase of the vehicle.

3.1.1. Perception

Perception is the first and one of the most important processes in auto-parking system.

In our case, we don’t have a HD map as a reference for available parking spot look-up. Therefore, sensors mounted on the vehicle detects other vehicles, available parking spots, as well as potential obstacles in the parking lot. Different types of sensors can be used in the auto parking system, such as cameras, ultrasonic sensors, LiDARs. Due to budget and technical issues, also in consideration of sensor fusion algorithm development, we choose to use multiple cameras as the major perception sensor.

Perception system can also be divided into two parts, available parking spot detection and object detection/tracking. Object detection and tracking algorithms are discussed in Chapter 4.

The core algorithm for parking spot detection is line detection. Similar to lane detection on public roads, parking spot detection algorithms will recognize geometric shapes such as lines and rectangles. Perspective transform, edge detection and line fitting are commonly used in lane detection algorithms [36]. Perspective transform will convert the camera image to a bird eye view image. The reason of perspective transform is to get a better view field of view of an image. The image below taken from a webcam will be used as an example for image processing.

Figure 20 Picture of commercial parking lot taken from webcam

However, lines in parking spots are more complicated than road lanes. For example, as shown in the figure below, parking spots has different shapes, orientations. The margin of each parking spot could be dashed, solid, or even double lines. Therefore, the detection algorithm should have the ability to recognize different types of parking spots.

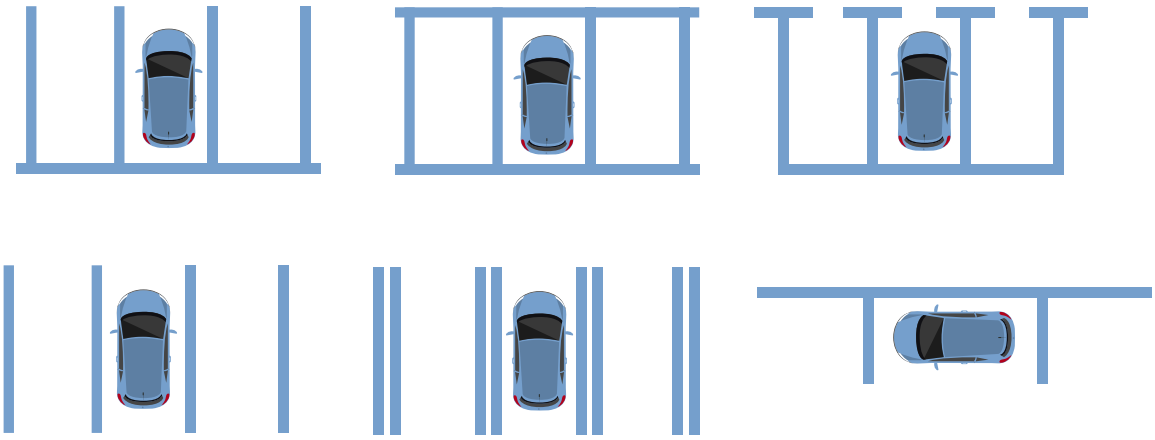


Figure 21 Different types of parking spots

The perception algorithm for cameras can be divided to six parts, as shown in the figure below. Note that camera calibration automatically calibrate distortion caused by camera

lens. Camera calibration is very important on cameras with wide field of view, such as fish eye cameras.

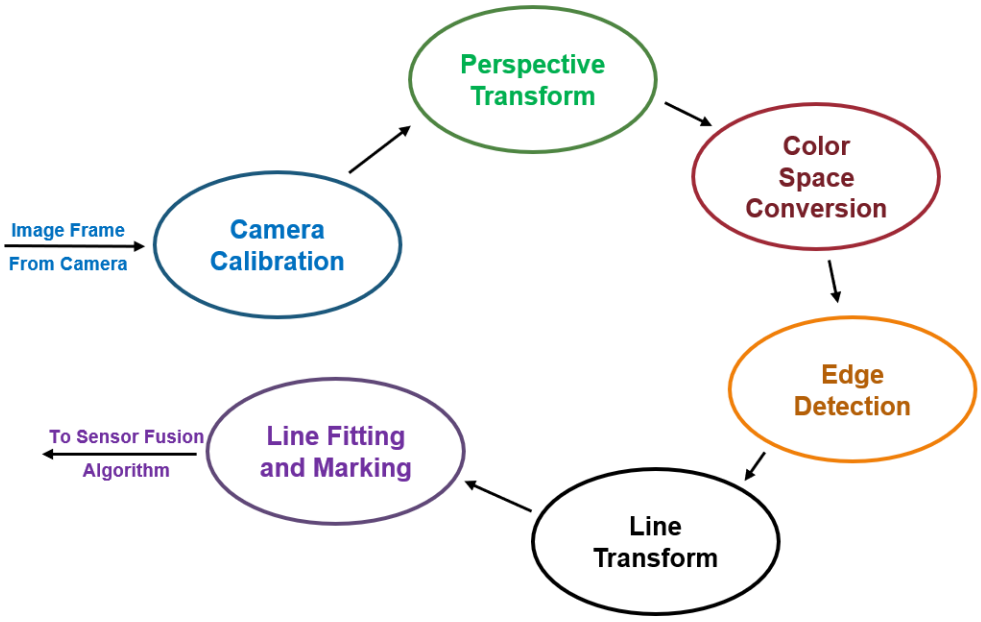


Figure 22 Camera perception algorithm flow chart

3.1.1.1. Perspective transform and color space conversion

The math behind perspective transform is simply multiplying matrices to an image. An image is made of pixels, each pixel has its value between 0 and 255, representing the greyscale of the image. An image with resolution of 1024×768 is a 1024-by-768 matrix with pixels as each element in the matrix. We can rotate, scale, shear and stretch the image by applying matrices to it.

Perspective transform can be considered as an orthographic projection. In computer graphics, the 6-tuple which defines the clipping planes, can be used as the orthographic transform matrix.

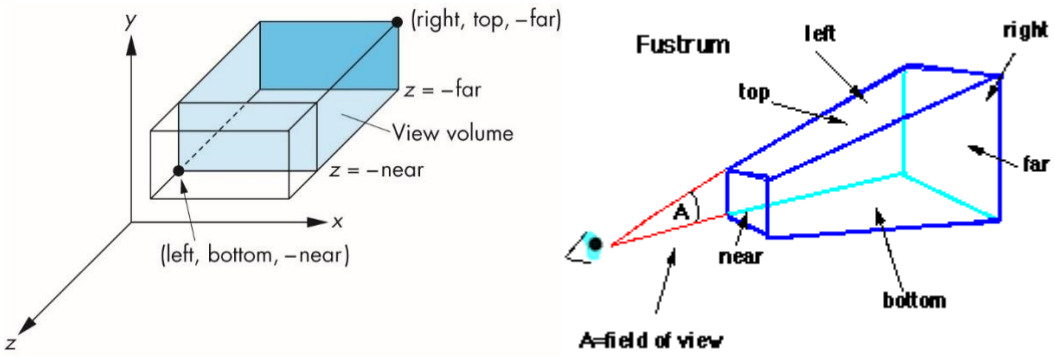
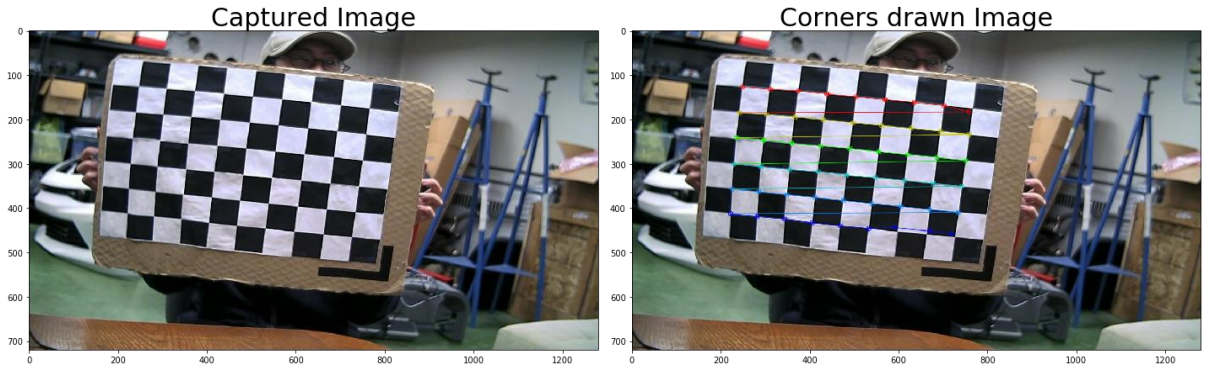


Figure 23 Left, right, top, bottom, near and far in computer graphics [37]

The orthographic transform matrix to be applied to an image is

$$P = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera calibration can also be considered as a process of perspective transform. Distorted camera image should be calibrated before processing. The black and white shapes within the chessboard are considered perfect squares for reference. By calculating the distortion rate, we can compute a correction matrix to be applied to each image frame later.



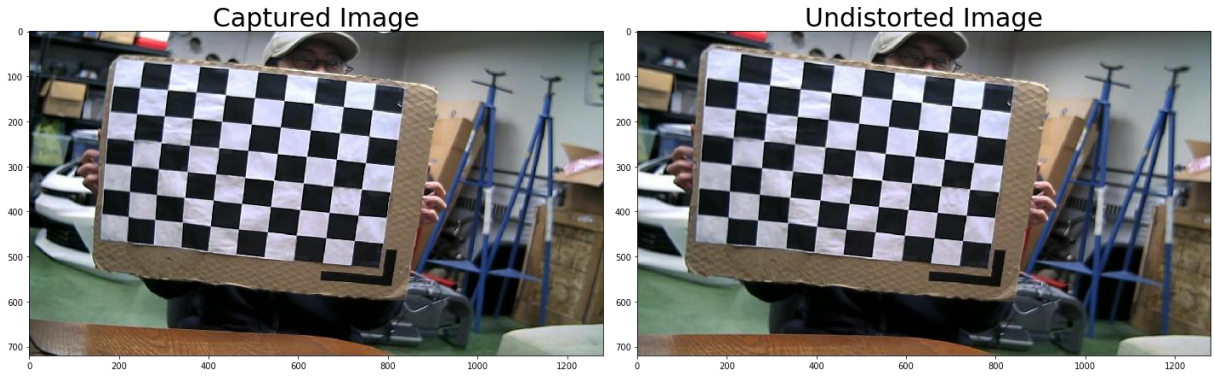


Figure 24 Camera calibration results

Color space conversion is also used in the perception algorithm. Parking spot lines painted on the ground are usually yellow and white. Conversion from RGB to HSV color space can distinguish these light colors from other colors. Similar to RGB, Hue Saturation Value (HSV) is a type of color space model.

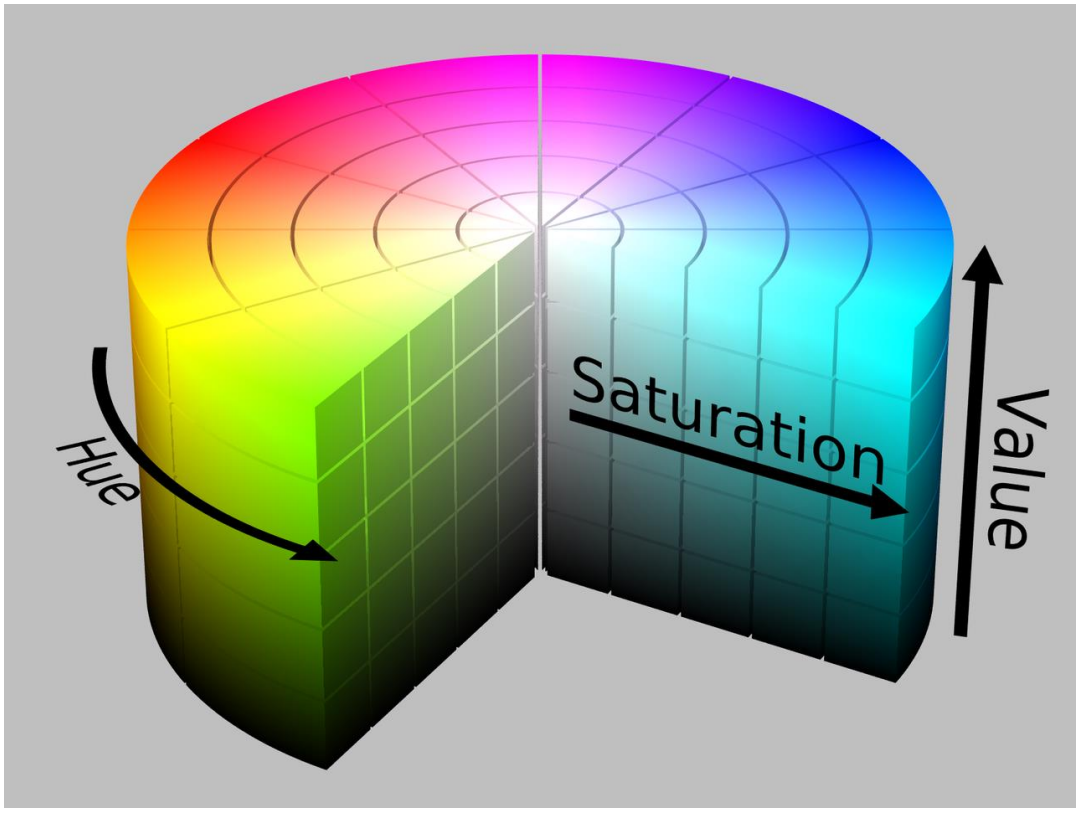


Figure 25 HSV Color space [38]

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(R, G, B)}{V}, if\ V \neq 0 \\ 0,\ \ otherwise \end{cases}$$

$$H \leftarrow \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)}, & if\ V = R \\ 120 + \dfrac{60(B - R)}{V - \min(R, G, B)}, & if\ V = G \\ 240 + \dfrac{60(R - G)}{V - \min(R, G, B)}, & if\ V = B \end{cases}$$

If $H < 0$ then $H \leftarrow H + 360$. The output of HSV color space is restricted to

$$0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 360.$$

After the perspective transform and color space conversion, we can see the yellow lines clearly even with human eyes.

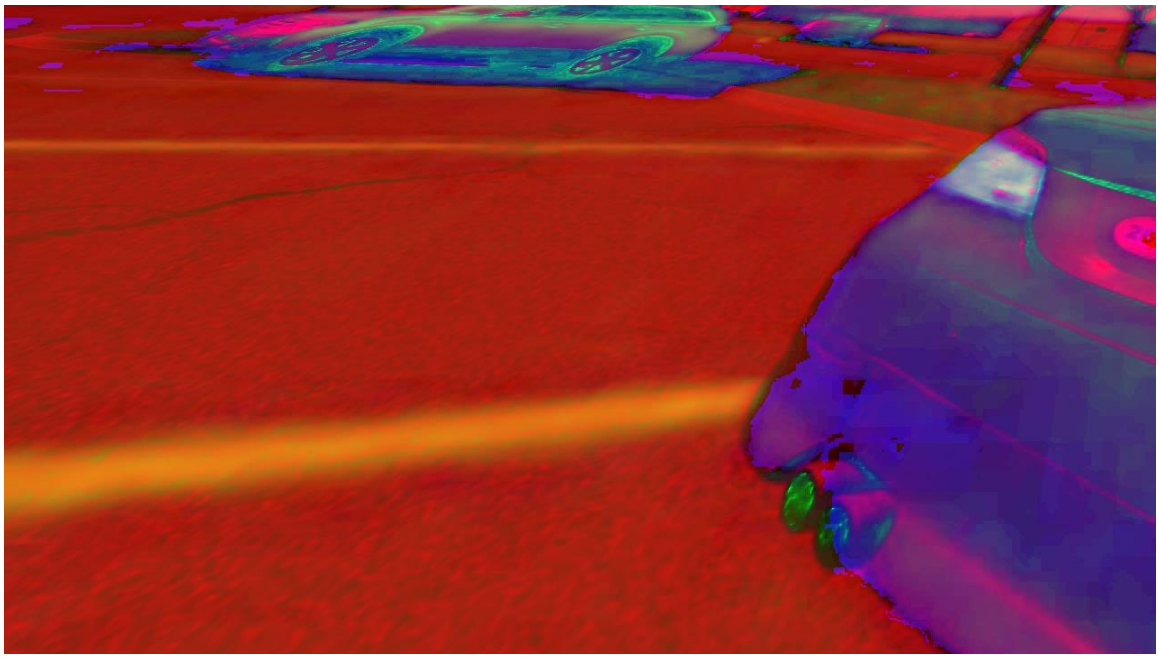


Figure 26 Image after perspective transform and color space conversion

3.1.1.2. Canny Edge Detection

Canny edge detection is used to detect features in an image, especially edge features. John Canny developed this edge detector in 1986 [39]. In order to detect edges in an image, a Gaussian filter will be applied to the image to smooth the image. Noise and useless feature details will be filtered out.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

Where

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{m^2+n^2}{2\sigma^2})$$

We can get the gradient of each image by performing any of the gradient operators such as Roberts, Prewitt, Sobel, etc.

$$M(n,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

And

$$\theta(m,n) = \arctan\left[\frac{g_n(m,n)}{g_m(m,n)}\right]$$

Then we can calculate the threshold M by

$$\mathrm{M_T}(m,n) = \begin{cases} M(m,n) & if\ M(m,n) > T \\ 0 & otherwise \end{cases}$$

Where T is chosen that all edge elements are kept while most of the noise is suppressed.

3.1.1.3. Hough Line Transform

After getting edge features from a canny edge detector, we need to fit lines in the image to visualize potential lanes. Lines in an image space can be defined as two parameters, either in Cartesian or Polar coordinate system.

For Hough transform, the parameters of lines will be discussed in the Polar system, as following,

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \frac{r}{\sin\theta}$$

$$r = x\cos\theta + y\sin\theta$$

For each pixel in the image (x_0, y_0), we can consider a group of lines that goes through that point as

$$r_\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$

And each pair of (r_θ, θ) represents a unique line that has (x_0, y_0) on it. We can draw a group of lines on this point, each plot will add a sine wave to the graph. We will consider only points such that $r > 0$ and $0 < \theta < 2\pi$.

For every point in the image, we can do similar iteration to find several sets of points that are on the same line. If more than one set of points were considered to be on one specific line, then there is high probability that we can declare a line detection. Hough transform tracks the intersection of each point set we found in the image. If the number of intersections goes over a pre-defined threshold, we consider there is a line.

After getting lines from Hough transform, we can create a bounding box using the four coordinates of the two side lines of a detected parking spot. Before completing the parking spot detection, the last step after Hough line transform is to convert the coordinates of the lines back to the original image. This can simply be done by performing inverse perspective transform.

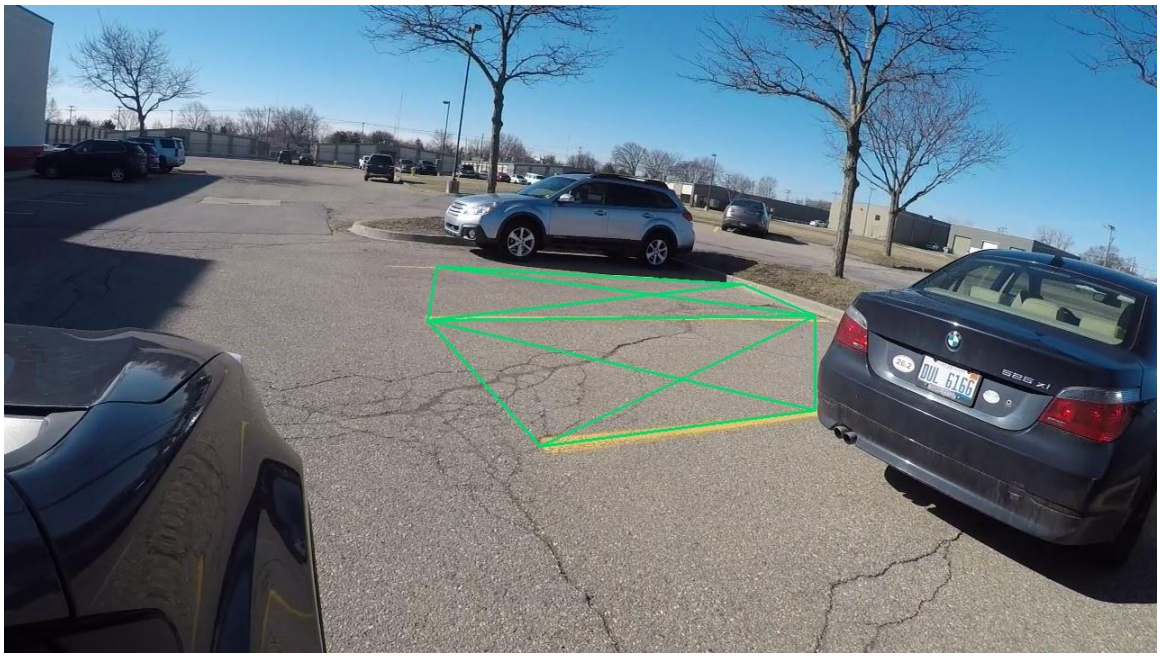


Figure 27 Picture after line detection

3.1.2. Object Detection Algorithms – Convolutional Neural Networks

The object detection and tracking algorithms are developed and applied on Nvidia GPU platform, which runs on a 16.04 Ubuntu OS, with a USB-webcam connected to laptop as a mono camera. The camera is mainly responsible for object detection and tracking. Object range and range rate measurement will also be part of its job, which will be discussed in the next section.

YOLO (You Only Look Once) is a real-time object detector equipped with deep fully convolutional neural network to generate bounding boxes, class labels and confidence scores. YOLO-network were used as object detection algorithm. This single network is applied to the full image, by first dividing the image into several sub-regions to predict probabilities as well as confidence scores for each region. Then bounding boxes are added to the image according to the confidence score. We designed a 34-layer convolutional neural network with two classes for vehicle feature extraction and

detection. The weights file for the networks was trained using GTI and KITTI vehicle picture dataset by PC with Nvidia GPU. The speed of YOLO-network is boosted using NVidia CUDA toolbox, which utilizes GPU on embedded Jetson TX2, increasing detecting speed from 2~3 FPS (CPU only) to 15~18 FPS (GPU only).

The coordinates of the bounding box are used for tracking objects. Kalman filter algorithm is used for vehicle tracking and labelling. We use OpenCV Kalman filter class and functions to predict the future position of the bounding box coordinates and reduce noise in the detected location. The tracking algorithm helps to associate multiple physical objects with their corresponding tracks. Different vehicles will be labeled with unique IDs throughout the whole detection period.

Range and range rate measurement can also be realized by using mono cameras, though predicting accuracy and robustness are not as good as radars. However, with the neural network algorithm, our detecting- and tracking-algorithms are robust enough to predict relatively accurate values for slow moving objects.

3.1.3. Another Object Detection Algorithm – HOG and SVM approach

We tested both HOG and YOLO on PC. Compared to an average of 15 FPS on Jetson TX2, YOLO can reach 50 FPS on a PC with Nvidia GeForce GTX 1066 for vehicle detection with the same algorithms.

For lane detection we improved our algorithm by adding perspective transform and creating a series of search windows over the lines. For perspective transform, OpenCV function *cv2.getPerspectiveTransform* (Python) and *cv2.warpPerspective* to generate a linear matrix transform. For search windows, which is like sliding-window technique, but with an extra action of splitting the windows into left and right halves.

For vehicle detection, Histogram of Oriented Gradients (HOG), and Support vector machine (SVM) approach is used by classifying vehicle using the HOG feature and color space feature. By performing a HOG feature extraction on a labeled training set of images, a linear SVM classifier could be trained. Linear SVM is a linearly scalable routine meaning that it creates an SVM model in a CPU time which scales linearly with the size of the train data set. It should be noticed that our training data set was created from the images that our team member took for previous ADAS tasks. Normalization of differences in magnitude between the color-based and gradient-based feature and sliding-window technique are used for trained classifier to search for vehicle in an image. This yields the result of images with adjacent boxes, as shown below.

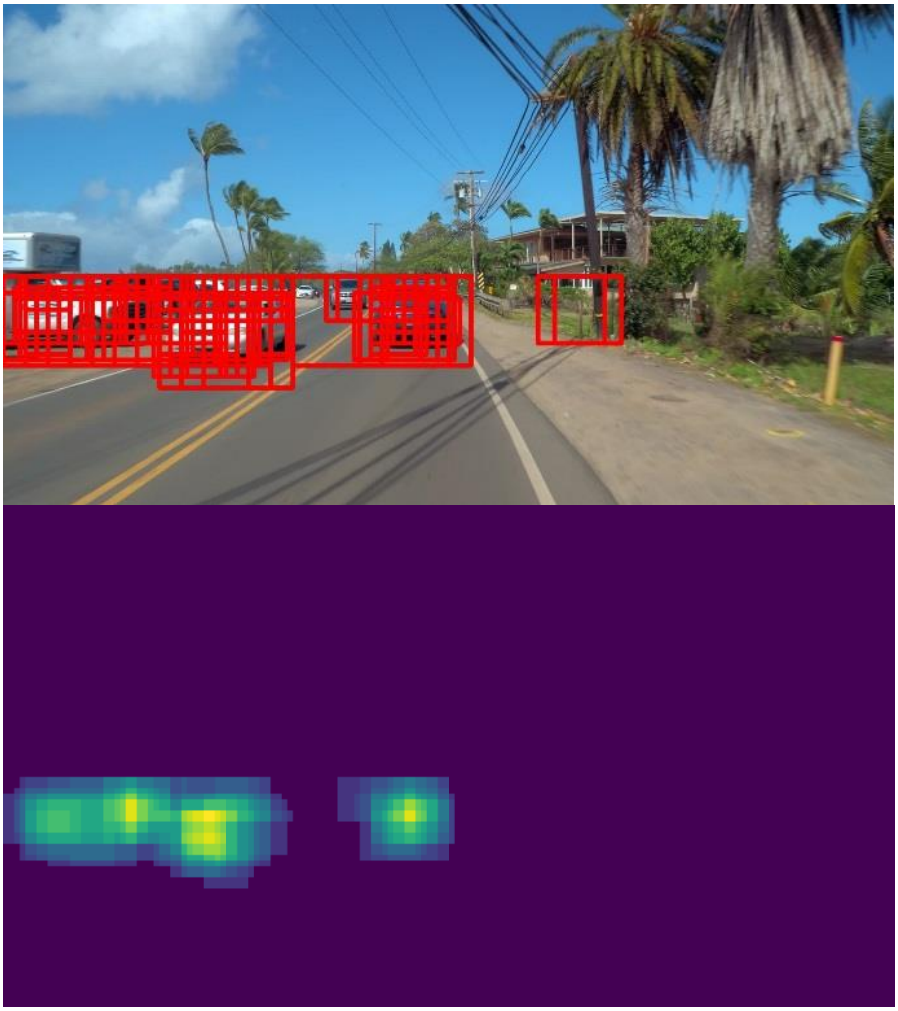


Figure 28 Heat map of vehicles in picture

However, the result is not satisfactory because there are many positives and multiple detections. Heatmap is a solution for this problem. By overlapping bounding boxes on the target image, calculating the accumulating rate of bounding boxes and create a heat map for vehicle detection. While the final results are satisfactory, yet the elapsed time for a vehicle detection is too long (0.2FPS). Furthermore, high contrast images would make our detector more accurate and robust.

Table 8 Performance and processing speed of two algorithms

Algorithms	True positive rate (%)	Processing speed (FPS)

		On PC	On Jetson TX2
HOG + SVM	~90	0.191	<0.01
Convolutional Neural Network	80.42	30-50	15-17

True positive rate is calculated by comparing detection result and ground truth labeled in the footage video captured by webcam on Jetson. Note that our neural network has quite high true positive rate when the object is quite close to the camera. As shown in the pictures, neural network algorithms start to detect vehicles at the distance of ~40 meters form the object (distance between each cone is 10 meters.

Figure 29 Comparison between HOG+SVM and Neural Network algorithms vehicle detection results

Figure 30 On-road test of Neural Network algorithm

We tested the neural network algorithm using pictures from Internet taken in different time and weather conditions. For example, sunny, snowy, foggy and rainy days, as well as daytime and nighttime traffic. Testing pictures have cars with different shapes and driving directions. Number of cars in each picture also varies.

Figure 31 Partial results for evaluating performance of vehicle detection algorithms

The output of the camera image processing algorithms contains obstacle position and raw data of each available parking spots. Especially for parking spot detection, some features will be the input of sensor fusion algorithms, which will be discussed in the next section.

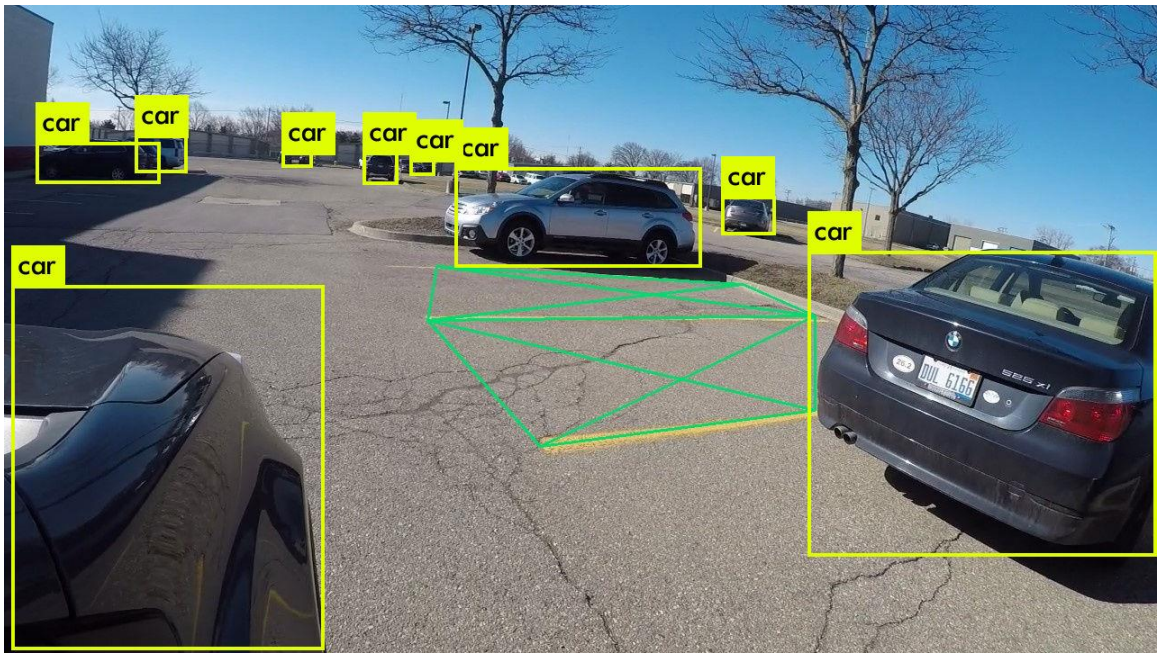


Figure 32 Image of parking lot after lane detection and object detection

3.1.4. Sensor fusion

Sensor fusion system is the critical part in the auto parking system. Sensor fusion algorithm collects objects list and parking spot features obtained from perception algorithms. GPS signal will also be fused to get the accurate location of the vehicle, i.e., the trajectory of the vehicle will be tracked when it starts to move. The information was used to build an environment model for localizing the car and identifying obstacles. Here we propose two approaches.

The first approach starts with Around View Monitor (AVM). Fusion image of four surround view cameras mounted on the four sides of the vehicle, we can obtain an around view picture by performing perspective transform, feature matching, then connecting and stitching four images to a large AVM image. The perception algorithm such as object detection and tracking, lane detection algorithms are conducted in the AVM image. The detection result (typically an object list) of perception algorithms can be directly put into

the world model for further processing.

The second approach is to do perception on each camera. Object detection/tracking and lane detection algorithms are performed by each camera and the detection results are put into the world model after perspective transform. This step requires the exact position of each camera. And the perception algorithm should have the ability to estimate the distance of the object.

In this research project, we will use the second method. To get a around view monitoring image, at least four high resolution cameras are needed. The location accuracy of the camera and dimension of the vehicle are critical to calibrating the camera. After calibration, the stitched surround view image is too distorted for object detection, as shown in the figure below.

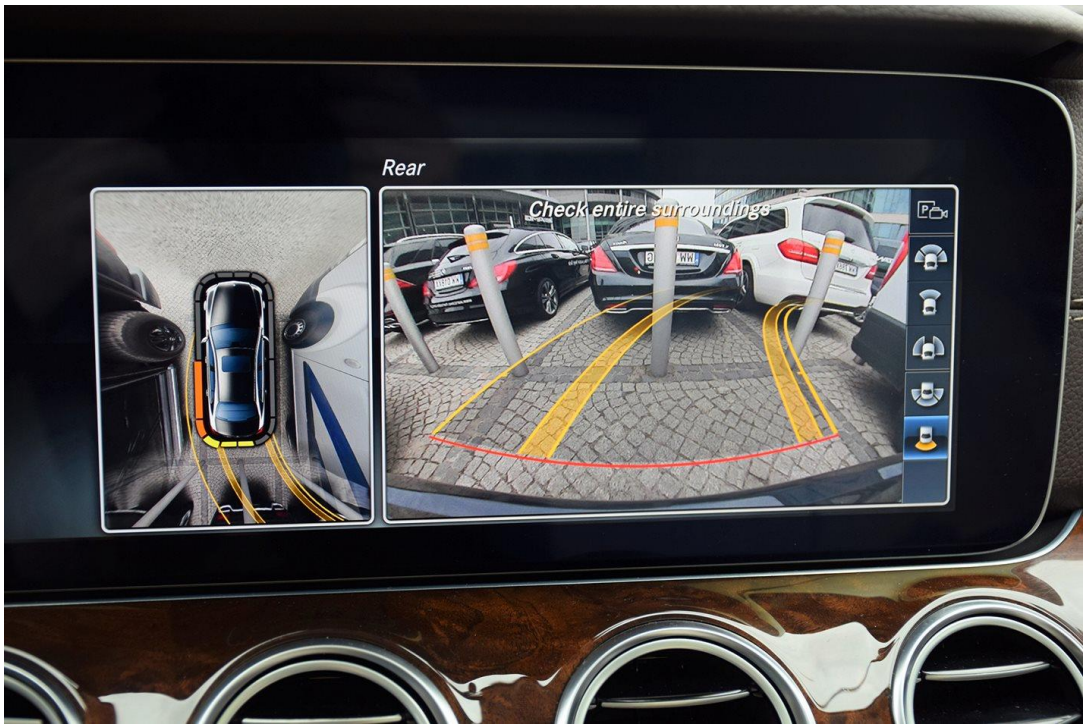


Figure 33 Distorted vehicles in surround view camera system [40]

In our case, we will use the second method to avoid complicated camera calibration. In

future work, we can mount high-resolution cameras permanently on a development vehicle to obtain the surround view and use other sensors to perform object detection.

During the parking process, the location of the vehicle is critical for path planning and localization. In the auto parking system, we will use GPS and IMU for localization. GPS will be used both in experiment and simulation. However, hacking into the IMU of a vehicle is difficult, IMU will only be used in the simulation environment.

As is discussed in the vehicle dynamics model earlier in this chapter, five parameters of the vehicle are used for describing the localization information of the vehicle. We will use a state vector to describe the vehicle model.

$$\mathrm{x_k} = [x\ y\ \psi\ v\ \dot{\psi}]$$

We will get readings of these five parameters from different sensors on the vehicle. X and Y are the position of the vehicle coming from the GPS, in Cartesian coordinate. ψ and $\dot{\psi}$ are the yaw angle and yaw rate of the vehicle coming from IMU, in polar coordinate. v is the speed of the vehicle. The value of speed comes from vehicle wheel encoder, and the heading of the vehicle comes from GPS.

As is described in the vehicle dynamics model, these five parameters are constrained by nonlinear functions. We will use extended Kalman filter for estimating the state of the vehicle. The mathematical model of an extended Kalman filter can be written as [41]

$$\boldsymbol{x}_k = g(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1}$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k$$

Where $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are noises occurred when processing and measuring the sensor data, respectively. They can be considered as mean Gaussian noises with Q and T covariance. Function g and h can be used to calculate the predicted state/measurement from the

previous estimate. At each step, a Jacobian matrix is applied between the prediction functions and the noise covariance.

From the vehicle dynamics model we can get that

$$\begin{bmatrix} x \\ y \\ \psi \\ v \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} x + \frac{v}{\dot{\psi}}(-\sin\psi + \sin(T\dot{\psi} + \psi)) \\ y + \frac{v}{\dot{\psi}}(\cos\psi - cos(T\dot{\psi} + \psi)) \\ T\dot{\psi} + \psi \\ v \\ \dot{\psi} \end{bmatrix}$$

Then we can take partial differential of the vehicle state matrix to get the Jacobian matrix of the vehicle state

$$\text{Jacobian(state)}$$

$$= \begin{bmatrix} 1 & 0 & \frac{v}{\dot{\psi}(-\cos\psi + \cos(T\dot{\psi} + \psi))} & \frac{1}{\dot{\psi}(-\sin\psi + \sin(T\dot{\psi} + \psi))} & \dots \\ 0 & 1 & \frac{v}{\dot{\psi}(-\sin\psi + \sin(T\dot{\psi} + \psi))} & \frac{1}{\dot{\psi}(\cos\psi - \cos(T\dot{\psi} + \psi))} & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & \frac{Tv}{\dot{\psi}}\cos(T\dot{\psi} + \psi) - \frac{v}{\dot{\psi}^2(-\sin\psi + \sin(T\dot{\psi} + \psi))} \\ \dots & \frac{Tv}{\dot{\psi}}\sin(T\dot{\psi} + \psi) - \frac{v}{\dot{\psi}^2(\cos\psi - \cos(T\dot{\psi} + \psi))} \\ \dots & T \\ \dots & 0 \\ \dots & 1 \end{bmatrix}$$

Extended Kalman filter can handle synchronization of different sensors. For example, in our case, the sample rate of GPS is 10Hz, while other sensors have a sample rate of 50Hz. Before we start fusing the data using extended Kalman filter, we will set up the initial environment, including initial uncertainty, noise covariance matrix, etc. Assuming

the initial parameters of the vehicle are 0, meaning when the driver activates auto-parking functionality, the vehicle is static, and the GPS is reset to the location where the driver stopped.

The next step is to define the sensor measurement noise, as is pointed out by Kelly, A., etc., the uncertainty estimates take on the significance of relative weights of state estimates and measurements. So it is not so much important that uncertainty is absolutely correct as it is that it be relatively consistent across all models [42]. We will define the measurement noise covariance R as

$$R = \begin{bmatrix} var(GPS_x)^2 & 0 & 0 & 0 \\ 0 & var\left(GPS_y\right)^2 & 0 & 0 \\ 0 & 0 & var(Speed)^2 & 0 \\ 0 & 0 & 0 & var(Yaw)^2 \end{bmatrix}$$

Where var() is the standard deviation of sensor measurement. Here we define the standard deviation is 6, 1 and 0.1, for GPS, speed and yaw rate, respectively.

The Process Noise Covariance Matrix Q is defined as

$$Q = \begin{bmatrix} s(GPS_x)^2 & 0 & 0 & 0 \\ 0 & s\left(GPS_y\right)^2 & 0 & 0 \\ 0 & 0 & s(Speed)^2 & 0 \\ 0 & 0 & 0 & s(Yaw)^2 \end{bmatrix}$$

Where s() is the maximum value a sensor can measure.

The extended Kalman filter has a loop that performs prediction and correction of the state parameters. Within each time-stamp in the loop, the prediction process will project the state and error covariance one time-stamp ahead, using the Jacobian matrix calculated each iteration.

$$x_{k+1} = g(x_k, u)$$

$$P_{k+1} = J_A P_k J_A^T + Q$$

The correction process will update the estimate via sensor measurement as well as the error covariance used in the prediction phase.

$$K_k = P_k J_H^T (J_H P_k J_H^T + R)^{-1}$$

$$x_k = x_k + K_k\big(z_k - h(x_k)\big)$$

$$P_k = (I - K_k J_H) P_k$$

The output of the sensor fusion algorithm contains parking spot features, as well as the ego vehicle information, as shown in the table below.

Table 9 Description of sensor fusion output parameters

Sensor fusion output	**Description**
i	Starting from zero, ID records the indices of all the parking spots found.
x_p, y_p	The orientation of the parking spot. The direction of the parking spot will determine vehicle motion planning.
d_i	Geometric feature of the parking spot
w_i	Geometric feature of the parking spot
m, n, p, q	Localize the parking lot in the frozen world coordinate.
ζ	Probability for the parking spot is available
$type\_i$	Different types of detected parking lot
v	Value of vehicle speed
ψ	Heading of vehicle speed

x_v, y_v	Relative position to the original point, where auto-parking started
$\dot{\psi}$	Yaw rate of vehicle

Sensor fusion result will be integrated in the world model. These features listed in the table are essential for vehicle maneuvering such as motion planning and local feedback control.

3.1.5. Route planning, behavior decision making and motion planning

With the perception and sensor fusion information, we can plan a path from the start point to the desired destination. In our case, the start point is the location of the vehicle when it detected an available parking spot. The destination is the center of the available parking spot.

Many path planning algorithms can be used in autonomous vehicle, such as Dijkstra algorithm, RRT algorithm, A* algorithm, etc. According to Reeds and Shepp [43], in a parking lot with obstacles, as long as there is a trajectory that can avoid all the obstacles between the initial and destination point, there is a Reeds-Shepp curve that allows the vehicle to move from start to end. To obtain a Reeds-Shepp curve, random searching algorithm are usually used to find a path for the vehicle. In our case, we will choose RRT as our path planning algorithm. RRT is the abbreviation for Rapid-exporting Random Trees.

Below is the pseudo code for RRT algorithm. The advantage of RRT algorithm is fast speed. During the auto-parking process, every time the vehicle moves, the path planning algorithm will re-generate a path for the vehicle to follow. This requires the path planning

algorithm to be fast in terms of processing speed.

Algorithm BuildRRT

Input: Initial configuration qinit, number of vertices in RRT K, incremental distance Δq)

Output: RRT graph G

```
G.init(qinit)
for k = 1 to K
  qrand ← RAND_CONF()
  qnear ← NEAREST_VERTEX(qrand, G)
  qnew ← NEW_CONF(qnear, qrand, Δq)
  G.add_vertex(qnew)
  G.add_edge(qnear, qnew)
return G
```

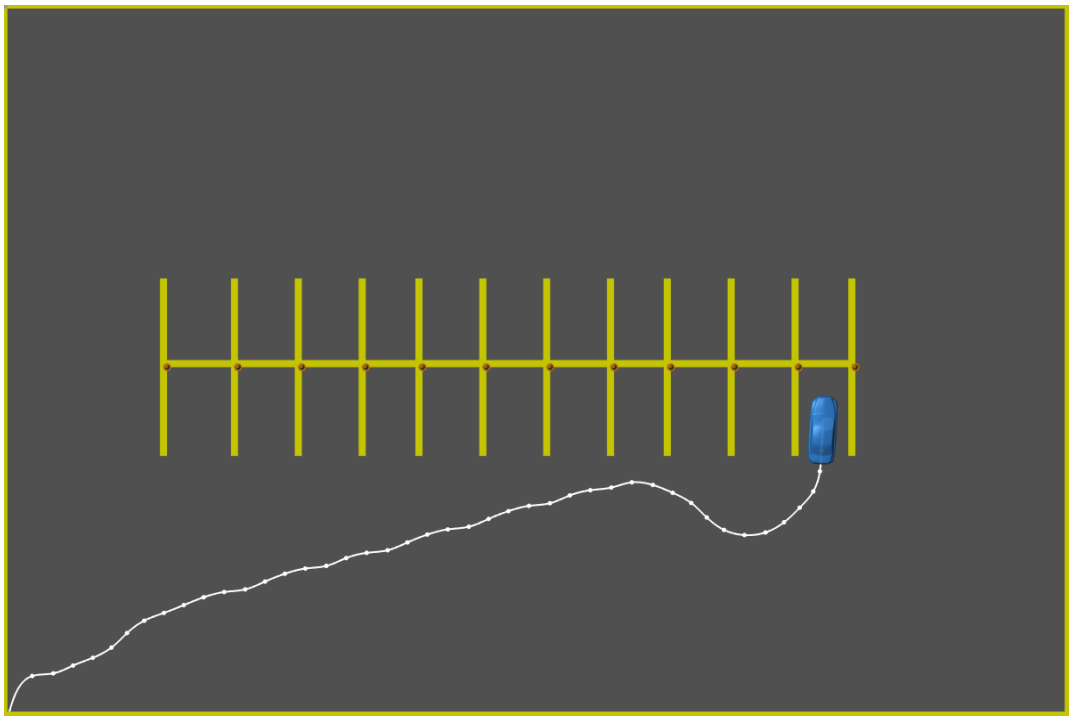


Figure 34 Reference path generated by RRT

3.1.6. Vehicle control

Motion planning determines reference velocity and reference pose of the vehicle. In the simulation environment, the vehicle is controlled by steering wheel, throttle and brake, same as vehicles we drive daily. In our case, vehicle control is a path follower. Using feedback control, the vehicle will be driven with a pre-defined path and speed profile, as discussed in the last section. In order to control the vehicle to follow the path created by path planning algorithm, a simple PID controller will be applied to control the vehicle.

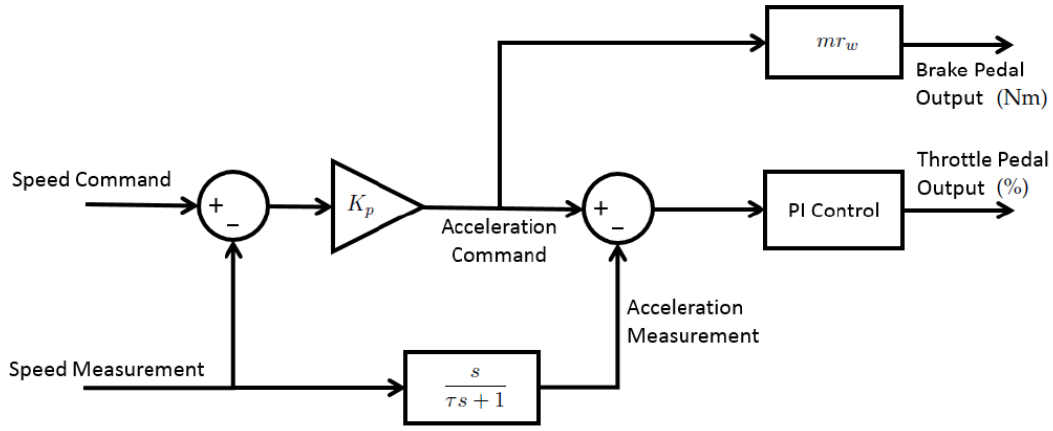


Figure 35 Diagram of the speed PID controller

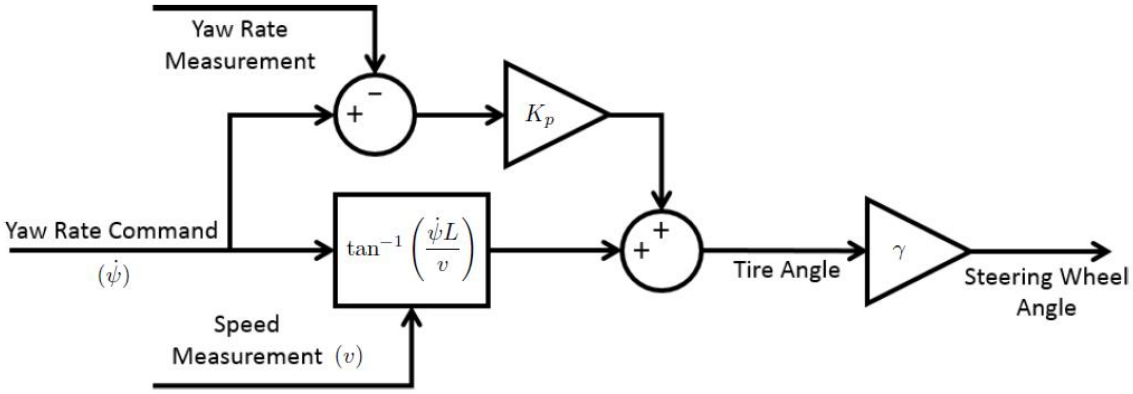


Figure 36 Diagram of the feed-forward steering PID controller

4. CHAPTER IV EXPERIMENTAL AND SIMULATION RESULTS

A universal simulation tool for ADAS feature testing and validation is developed in this chapter. The simulation tool, especially for the auto-parking algorithm simulation, is conducted in Robot Operating System (ROS). ROS is an open source simulation environment for robotics research application. In this simulation environment, vehicle simulation tool such as Gazebo and visualization tool such as Rviz can be used as the main development platform.

4.1. ROS Environment setup

In Gazebo, a plug-in open source software for robot visualization and simulation, vehicles and obstacles are simulated. In our simulation, a Lincoln MKZ is used as the ego vehicle. Vehicle dynamics discussed in the previous chapter can be defined in Gazebo. In this vehicle model, vehicle mass, center of mass and Ackermann steering mechanism are also simulated to make the model more realistic.

Multiple sensors are mounted on the vehicle, including a front view camera; a perfect GPS with Gaussian noise added; a surround view camera system with four fisheye cameras; an IMU; a mid-range Radar; 12 ultrasonic sensors and a Velodyne 32 beam scanning LiDAR. Although not all the simulated sensors are used in the auto-parking application, they can be reserved for future work. For example, the LiDAR and Radar can be used as major sensors for level 3 or level 4 feature development such as highway pilot. From the screenshots taken in Gazebo below, we can see how the vehicle is modeled and the two obstacles are for visualization purposes.

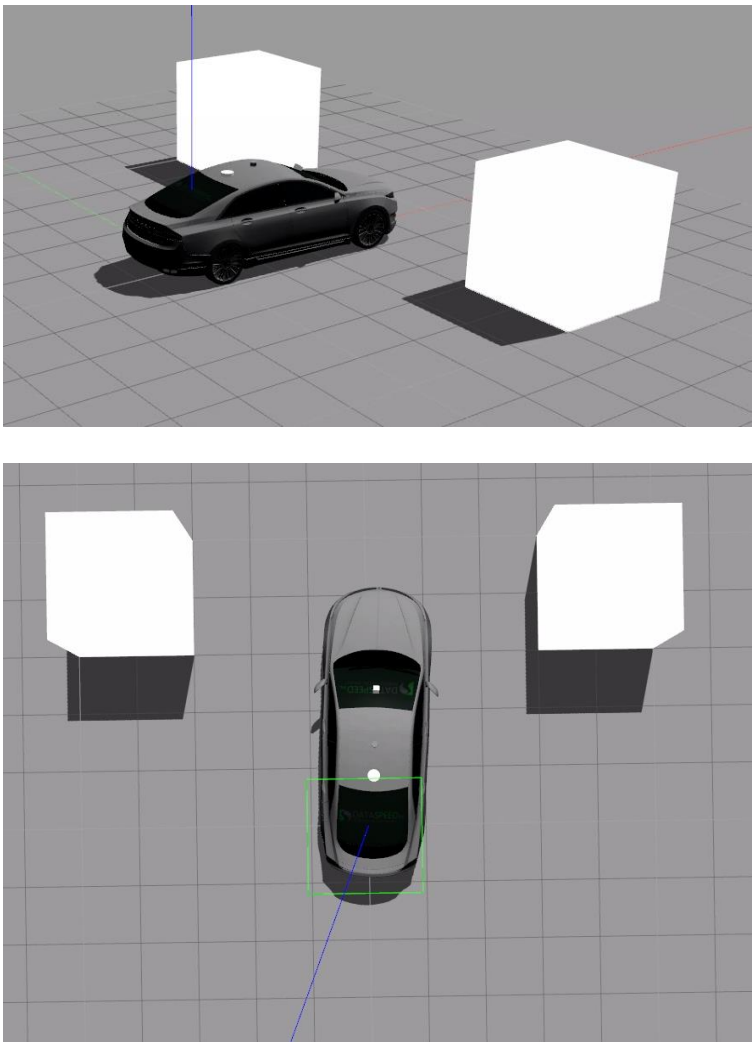


Figure 37 Vehicle model built in Gazebo with different view angle

As is discussed in the previous chapter, the auto-parking process can be divided to six parts. Perception, sensor fusion, route planning, behavior decision making, motion planning and vehicle control. All these functionalities can be simulated in ROS environment.

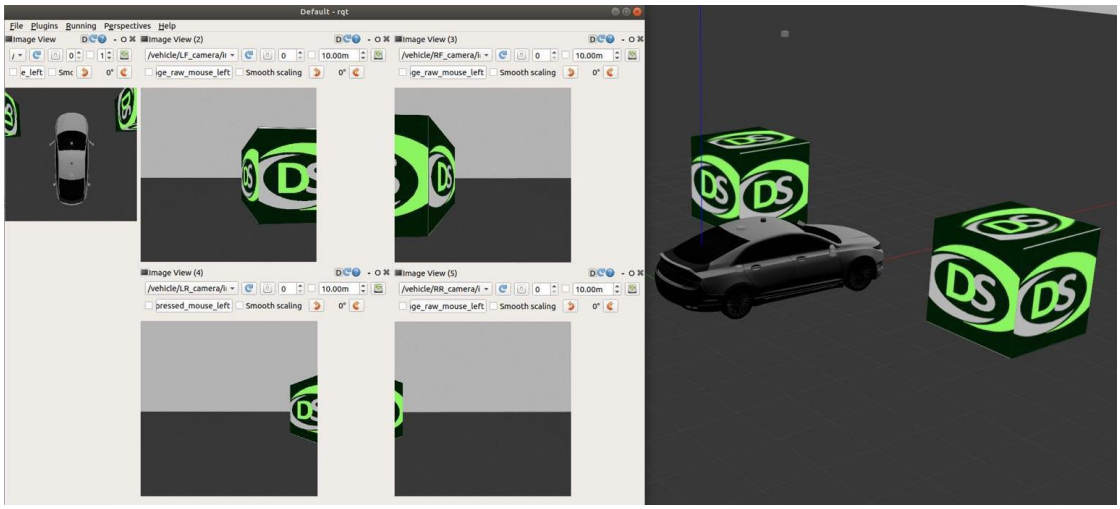


Figure 38 Surround view camera simulation

As is shown in the figure above, four surround view cameras are mounted on the simulation vehicle. A top-down view camera is deployed to monitor the status of the vehicle using a bird eye view. The image captured by the cameras can be used for image processing purpose, as discussed in previous chapters. GPS sensors can be setup on the vehicle, providing the speed, heading and position of the vehicle. In Gazebo, sensor data will be transmitted as a *rostopic*. Command *$ rostopic echo /vehicle/** can be used to visualize sensor data.

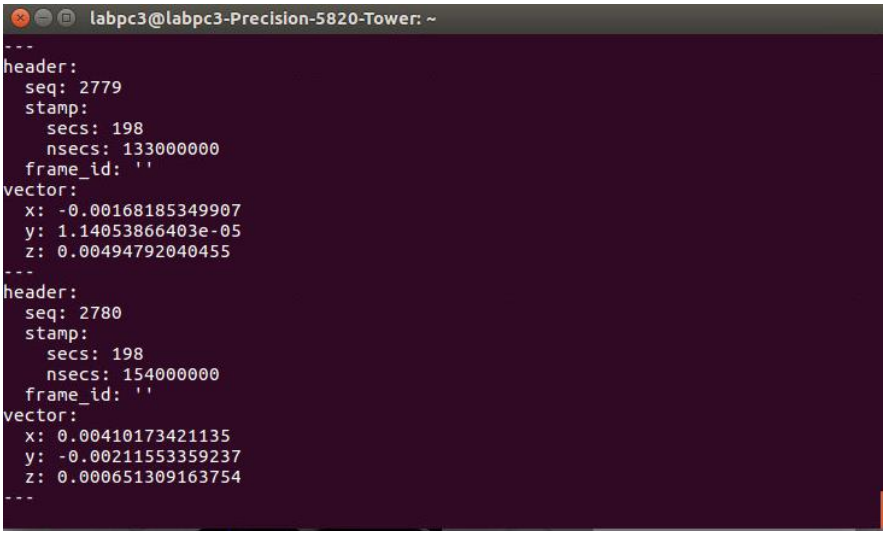


Figure 39 On-vehicle GPS data acquired from rostopic list

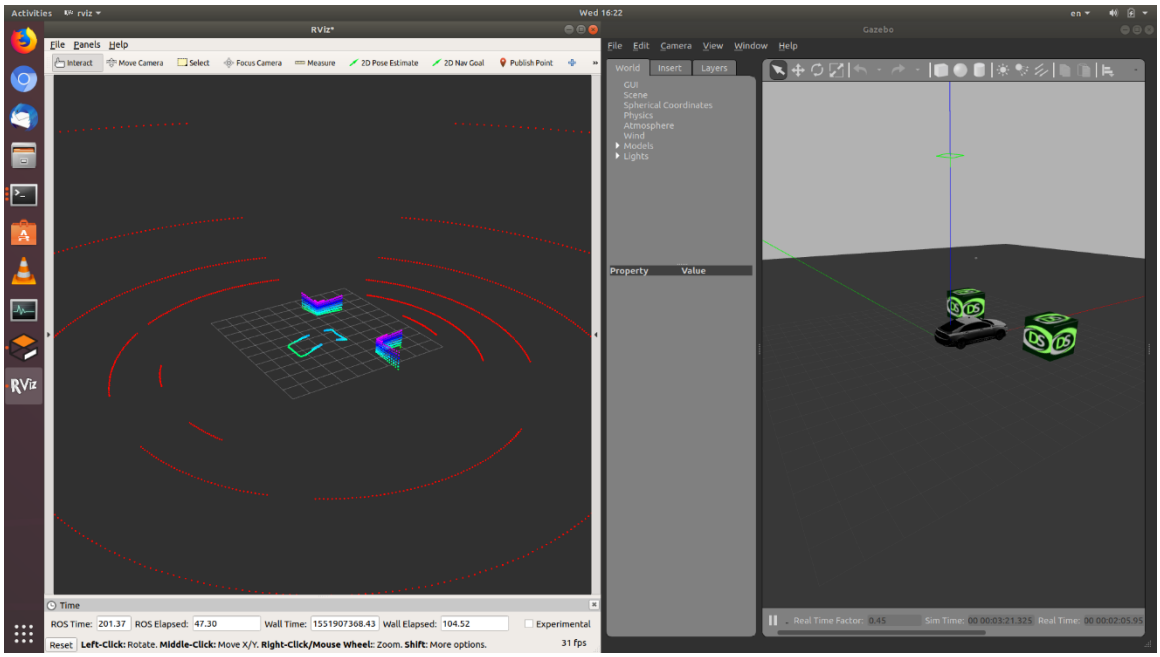


Figure 40 LiDAR detection result in simulation environment

As is shown in the figures above, virtual sensors mounted on the vehicle can be used to perceive information from the virtual world in Gazebo. Sensor fusion, motion planning and vehicle control will be conducted within the work flow below. The work flow graph is generated by command *$ rqt_graph*. This command can help us visualize all the nodes and topics from the simulation environment.

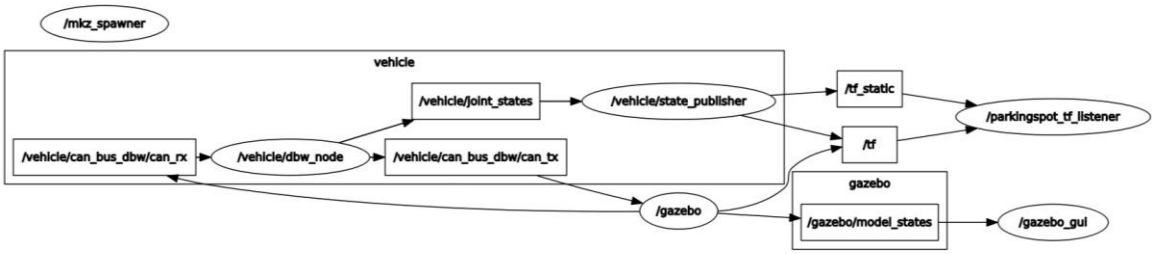
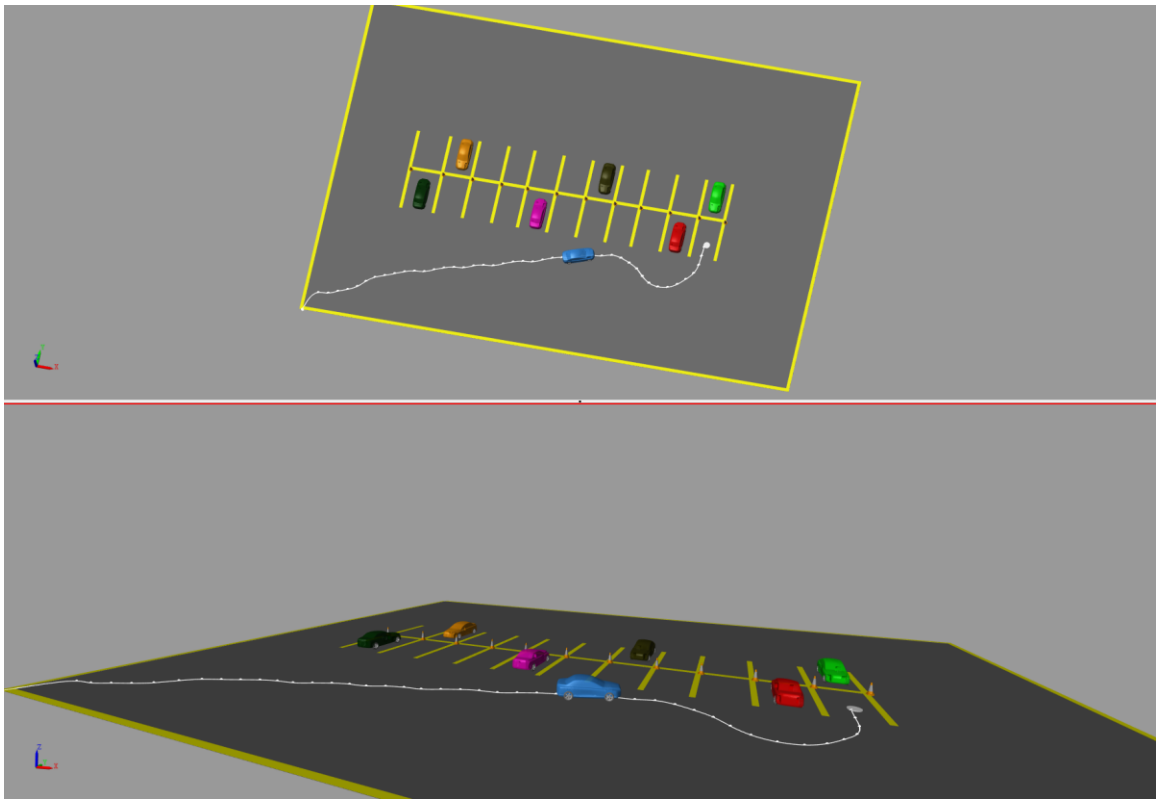


Figure 41 RQT Graph of the simulation environment

4.2. MATLAB Simulation Environment

Algorithm development in ROS can be performed in either C++ or Python. However,

we can also use MATLAB ROS Toolbox to communicate between ROS and MATLAB. In this case, we can develop all the algorithms in MATLAB Simulink and Simscape. In our case, the sensor fusion result processed in ROS will be transmitted to MATLAB as a ROS topic. Motion planning and vehicle control algorithm are conducted in Simulink. We can also visualize the parking process in MATLAB Simscape multi-body simulator.



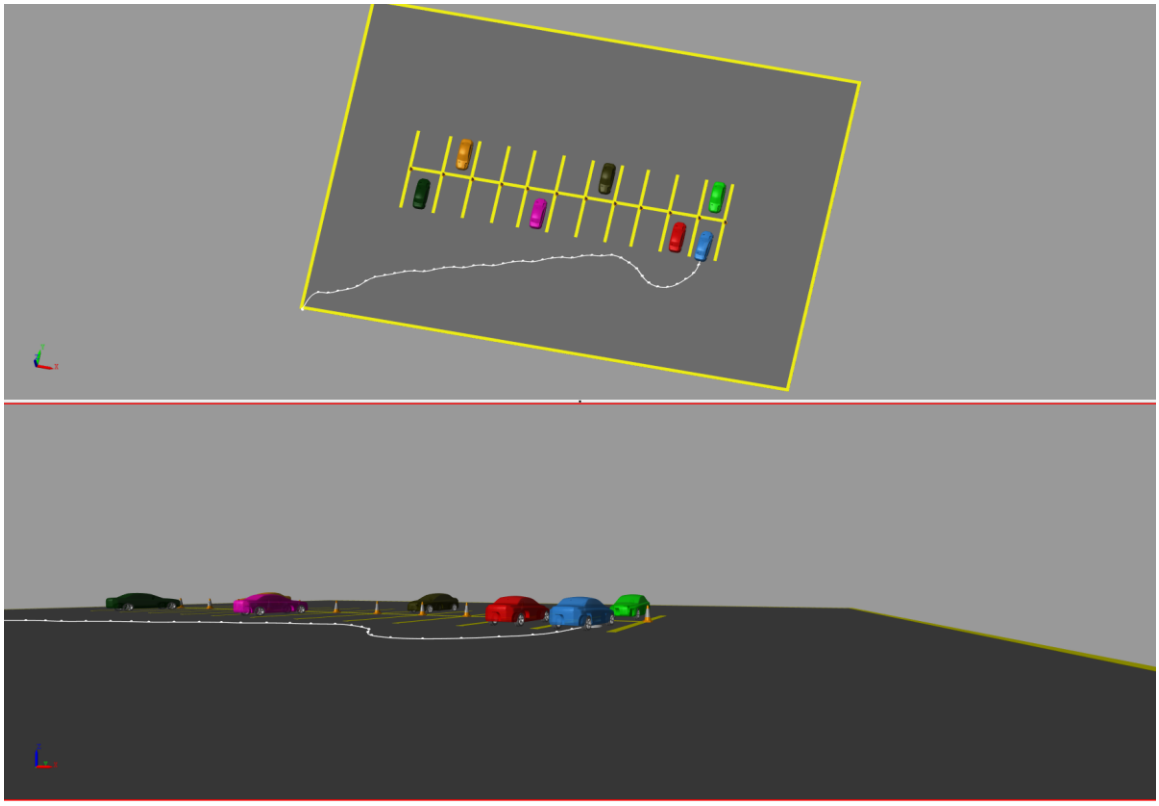


Figure 42 Auto-parking Simulation result in different view angles

Similarly, algorithm output from MATLAB, such as reference path and vehicle steering angle, can also be transmitted to ROS environment. In the following simulation experiment, lane detection result detected from ROS is transmitted to MATLAB, where path planning and vehicle control is performed. The generated path and steering angle are transmitted back to ROS, in order to control the vehicle to drive within the lane, as shown in the figure below. The whole system can reach a speed of 10 FPS, meaning 10 messages transmitted to ROS per second.

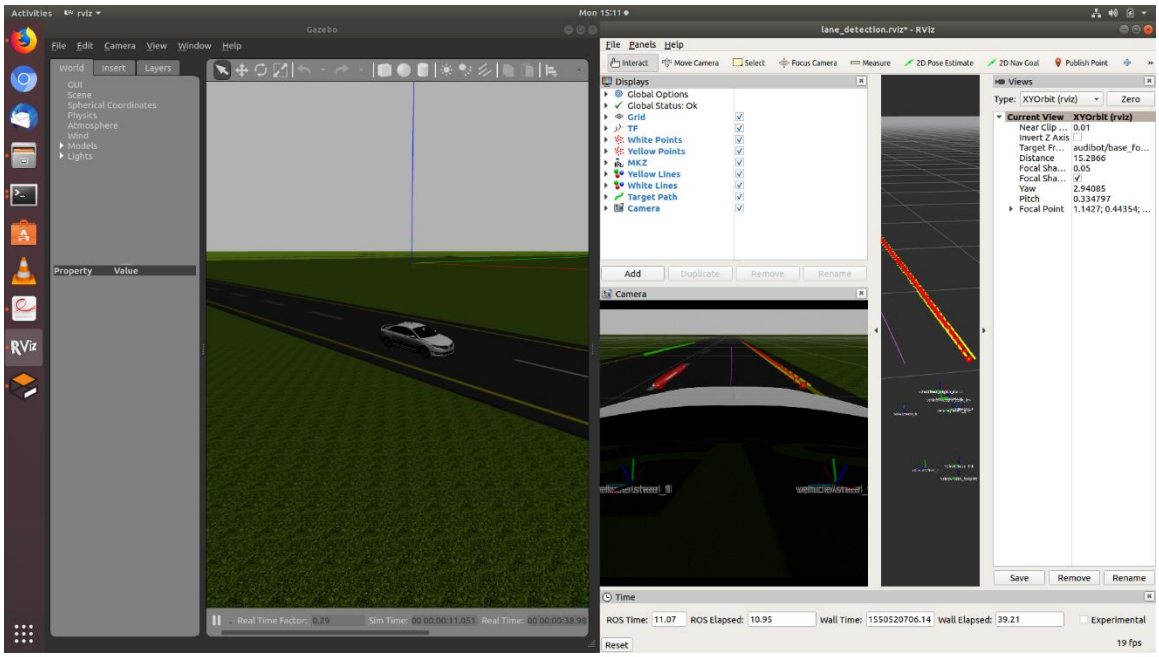


Figure 43 Lane Keeping Assist simulation based on ROS and MATLAB

4.3. Experimental results

4.3.1. Surround view camera experiment

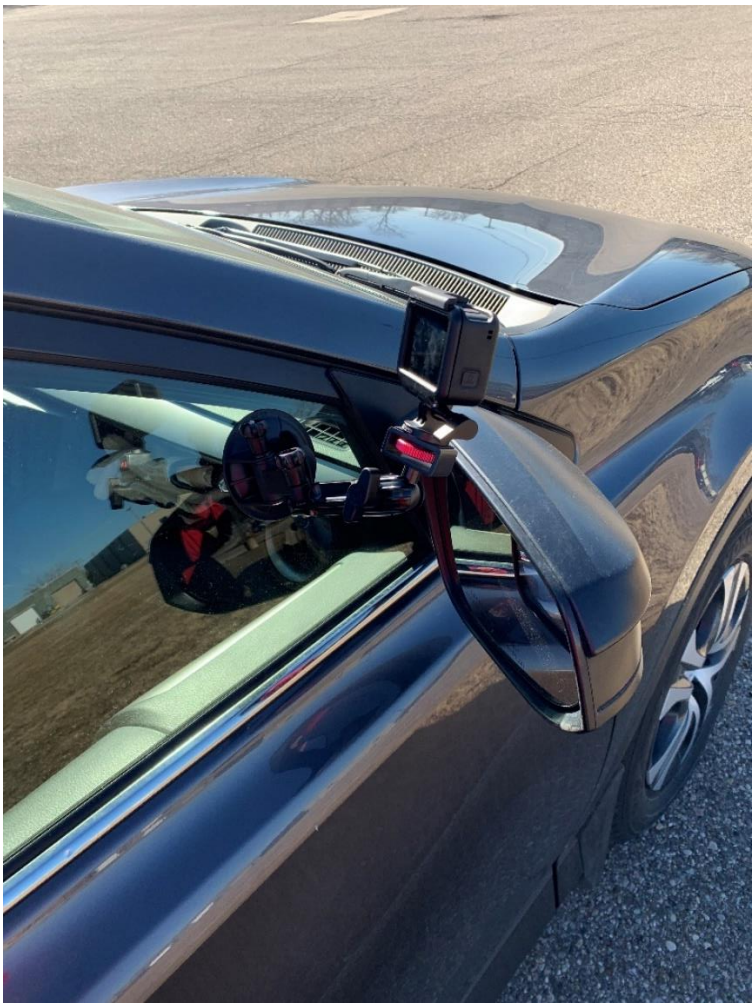


Figure 44 Camera mounted on the vehicle for experiment

The experiment for surround view camera parking spot detection is conducted in an open US parking lot with yellow solid line markers. A webcam is mounted near the wing mirror on the side window. The videos recorded from the webcam was stored for post processing. Parking spot detection and object detection are performed on the video.

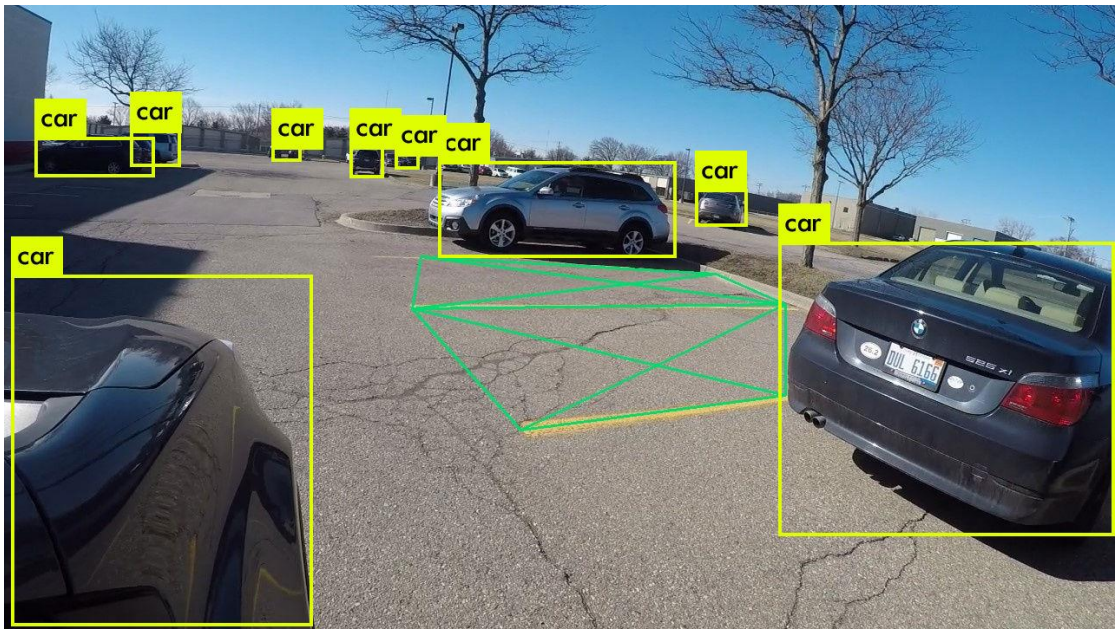


Figure 45 Parking spot and object detection in a parking lot

Comparing with ground truth labeled by human drivers, the true positive rate for parking spot detection after sensor fusion is 73%. Most of the false negative detection are due to non-robust computer vision algorithms. The detection algorithm is sensitive to cracks on the ground and vehicle shade that partially covered lines on the ground.

4.3.2. GNSS vehicle tracking experiment

Normal GNSS (GPS) receivers usually have a few meters of positioning errors, and not accurate enough for use in ADAS applications. RTK GNSS can achieve a better positioning accuracy with positioning errors of a few centimeters, but they are very expensive. In our experiment, we used an embedded GNSS sensor with antenna, developed based on Arduino.

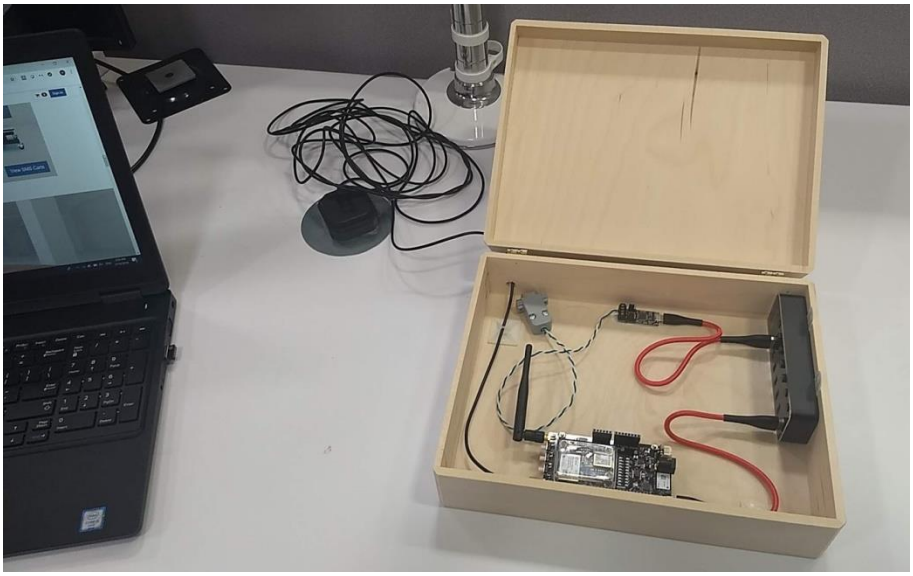


Figure 46 GNSS Sensor for vehicle localization

The GNSS is using the RTK base station in the City of Warren, Michigan. The base station is located at 42° 32' 02.45507" N, 83° 01' 13.34816" W.

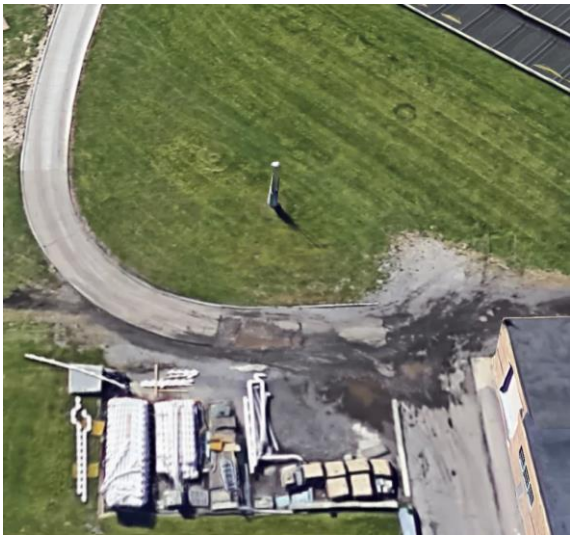


Figure 47 RTK base station near Warren waste water treatment plant on Google Map

With the GNSS sensor, we can get accurate position of the test vehicle, which is an essential assumption in the sensor fusion algorithm discussed in the previous chapters. During the experiment, the GNSS sensor is mounted on the top of the test vehicle. The vehicle was driven around in a parking lot. The GNSS has a 10Hz sample rate and after

applying Kalman filter to it, the trajectory is smooth. The trajectory of test vehicle is drawn on Google Map background.

Figure 48 Vehicle location GNSS signal during experiment

The GNSS sensor can be used to create an HD map for future work. The antenna was put on the parking spot lines, road curb and s light pole to create a scaled map for the parallel parking.

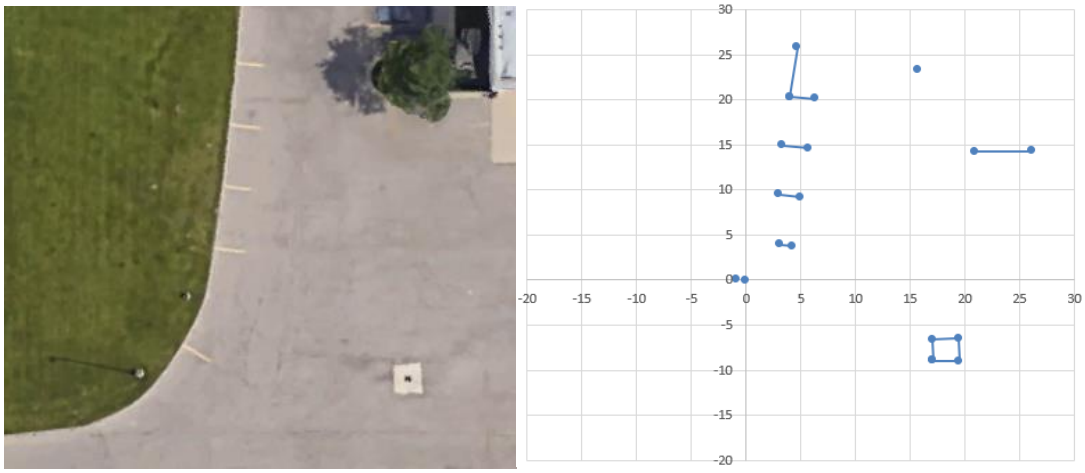


Figure 49 Simple HD Map creation using GNSS sensor

5. CHAPTER V CONCLUSION AND FUTURE WORK

5.1. Conclusions

This thesis presents an integrated design of Auto-Parking Assist in an Advanced Driver Assistance System. The modified architecture of the ADAS, especially for auto-parking feature, is introduced in this thesis. The system was broken down to four major sub-systems: sensor selection, perception and signal processing, motion planning and behavior decision making, and vehicle control and trajectory following. Each sub-system was discussed in detail in this thesis. A ROS based simulation environment has been developed to simulate the performance of the ADAS feature, based on individual tests on perception, sensor fusion, path planning, and vehicle control. Due to resource limitation, simple experiments such as camera image processing for available parking spot detection and GPS localization were conducted. More sophisticated experiment involving more sensors and processors are planned for future work.

Sensor selection criteria for autonomous driving have been established in this thesis. Different types of sensors have pros and cons to be utilized in different use cases, in terms of ADAS features, traffic conditions and surrounding environments. Cameras and localization sensors such as GPS are used for auto-parking application. Sensor fusion between cameras and GPS using extended Kalman filter are discussed in the thesis. Cameras mounted around the vehicle are used for detecting available parking spots. GPS is used as a localization sensor in order to obtain the accurate location of ego vehicle for minimizing accumulative error. The fused sensors will provide more accurate parking spot detection result as well as vehicle localization information. Motion planning algorithms are based on Reeds-Shepp and Rapid-exploring Random Tree (RRT). Based

on the detection and fusion result from the perception and sensor fusion system, an initial point and a destination point will be defined in the vehicle coordinate system. The RRT algorithm was used as a path planning algorithm to plan a feasible path that guides the vehicle to the parking spot. A PID controller is used to control the steering angle of the vehicle to follow the trajectory.

A simulation environment has been designed and setup up in Robot Operating System (ROS). Virtual vehicle, lane markers, parking lot and obstacles such as vehicles and pedestrians are defined in the simulation world. The simulation vehicle subjects to a bicycle vehicle dynamics model. Virtual sensors such as cameras, GPS receiver, LiDAR and ultrasonic sensors are simulated in ROS.

5.2. Future work

5.2.1. Algorithm improvement

The sensor fusion algorithm is based on the constant velocity vehicle model, which limits the motion planning and vehicle control of the system. For example, the path planning algorithm will only be able to generate a geometric path without generating a speed profile for the trajectory. At the same time, a more sophisticated vehicle controller can be developed to handle real-world experiments, such as Model Predictive Control, Fuzzy Control, etc.

5.2.2. Use cases in other ADAS feature scenarios

The ADAS architecture design in this thesis can also be migrated to other ADAS functionalities to achieve a higher level of automation. For example, Adaptive Cruise Control (ACC), Traffic Jam Assis (TJA), High Way Assist (HWA), etc. can adopt the sensor fusion and motion planning algorithm.

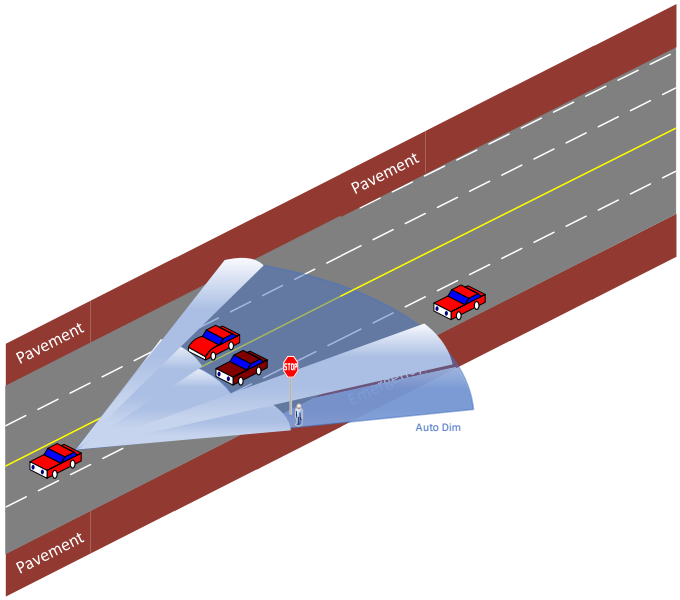


Figure 50 Adaptive cruise control: use case scenario on local roads

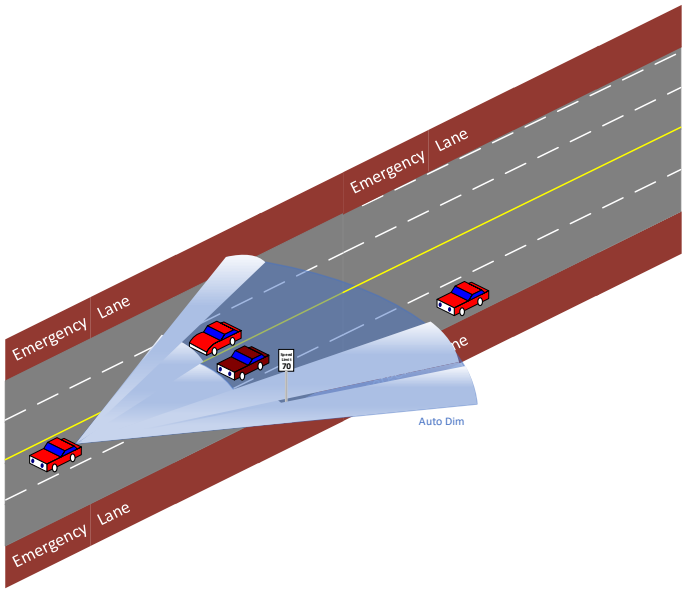


Figure 51 Adaptive cruise control: use case scenario on highways

Adaptive cruise control is commonly used as the primary ADAS feature on vehicles. When adaptive cruise control is enabled on highway or local express way, the ego vehicle will cruise at a pre-set speed and follow the preceding vehicle by automatically adjusting throttle and brake. If the preceding vehicle slows down or cut-in, the ego vehicle should

be able to detect the speed difference between itself and the preceding vehicle and adjust its speed accordingly.

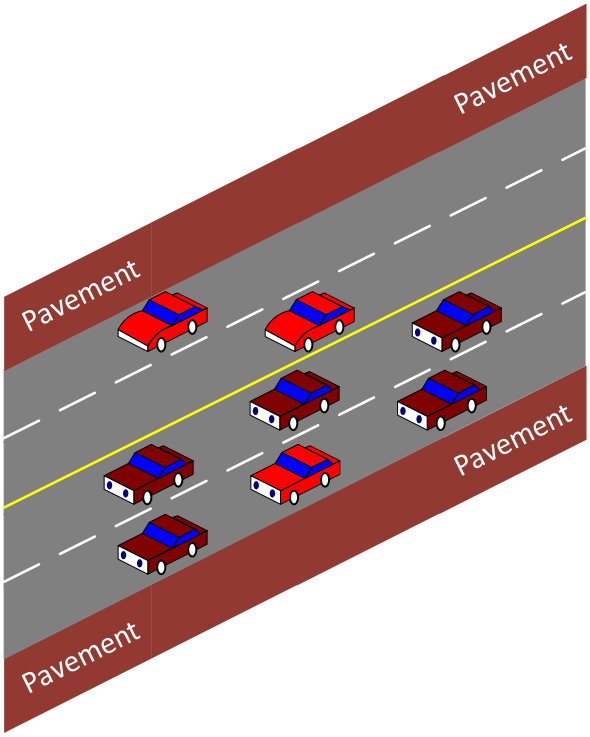


Figure 52 Traffic jam assist use case

Traffic jam assist is another important ADAS feature yet not many vehicles have in their ADAS package. The system will automatically adjust steering, throttle and brake to navigate the vehicle in traffic jam situations. Normally the system will drive the vehicle in the host lane and will not change lane. When the driver turns on the turning lights, the system can provide necessary information to the driver (such as blind spot detection results, etc.), but the system will not have any physical operations.

APPENDIX

Parking spot to world model coordinate transform

```
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <tf/transform_listener.h>

ros::Publisher parkingspot_publisher;

void transformPose(const tf::TransformListener& listener){
  //we'll create a point in the base_laser frame that we'd like to transform to the base_link
frame
  geometry_msgs::PoseStamped parkingspot;
  parkingspot.header.frame_id = "world";

  //we'll just use the most recent transform available for our simple example
  parkingspot.header.stamp = ros::Time();

  //just an arbitrary point in space
  parkingspot.pose.position.x = 0.0;
  parkingspot.pose.position.y = 0.0;
  parkingspot.pose.position.z = 0.0;
  parkingspot.pose.orientation.x = 0.0;
  parkingspot.pose.orientation.y = 0.0;
```

```
  parkingspot.pose.orientation.z = 0.3428978;
  parkingspot.pose.orientation.w = 0.9393727;
  try{
    geometry_msgs::PoseStamped base_point;
    listener.transformPose("vehicle/base_footprint", parkingspot, base_point);
    parkingspot_publisher.publish(base_point);
    ROS_INFO("world: (%.2f, %.2f. %.2f) -----> base_footprint: (%.2f, %.2f, %.2f) at
time %.2f", parkingspot.pose.position.x, parkingspot.pose.position.y,
parkingspot.pose.position.z, base_point.pose.position.x, base_point.pose.position.y,
base_point.pose.position.z, base_point.header.stamp.toSec());
  }
  catch(tf::TransformException& ex){
    ROS_ERROR("Received an exception trying to transform a point from \"world\" to
\"base_link\": %s", ex.what());
  }
}


int main(int argc, char** argv){
  ros::init(argc, argv, "parkingspot_tf_listener");
  ros::NodeHandle n;
  parkingspot_publisher =
n.advertise<geometry_msgs::PoseStamped>("parkingspot",1000);
  tf::TransformListener listener(ros::Duration(10));
```

```
  //we'll transform a point once every second
  ros::Timer timer = n.createTimer(ros::Duration(0.1), boost::bind(&transformPose,
boost::ref(listener)));

  ros::spin();

}
```

Transmit Brake Command from MATLAB to ROS

```
#! /usr/bin/env python
import rospy
from dbw_mkz_msgs.msg import BrakeCmd

class CmdKeeper:
    def __init__(self):
        rospy.init_node('BrakeCmd_Keeper')
        rospy.Subscriber('/vehicle/SimulinkBrakeCmd', BrakeCmd, self.recv_brakecmd)
        self.pub_brakecmd = rospy.Publisher('/vehicle/brake_cmd', BrakeCmd,
queue_size=1)
            self.start = False
            self.cmd = BrakeCmd()
            self.pubrate = rospy.Rate(50)
```

```
    def pub_cmd(self):
        while not rospy.is_shutdown():
            if self.start:
                self.pub_brakecmd.publish(self.cmd)
            self.pubrate.sleep()
    def recv_brakecmd(self, msg):

        self.cmd.pedal_cmd = msg.pedal_cmd
        self.cmd.pedal_cmd_type = msg.pedal_cmd_type
        self.cmd.enable = msg.enable
        self.cmd.clear = msg.clear
        self.cmd.ignore = msg.ignore
        self.cmd.count = msg.count
        self.start = True

if __name__ == '__main__':
    node_instance = CmdKeeper()
    node_instance.pub_cmd()
      rospy.spin()
```

Steering Command

```
#! /usr/bin/env python
```

```
import rospy
from dbw_mkz_msgs.msg import SteeringCmd


class CmdKeeper:
    def __init__(self):
        rospy.init_node('SteerCmd_Keeper')
        rospy.Subscriber('/vehicle/SimulinkSteerCmd', SteeringCmd, self.recv_steeringcmd)
        self.pub_steercmd = rospy.Publisher('/vehicle/steering_cmd', SteeringCmd,
queue_size=1)
            self.start = False
            self.cmd = SteeringCmd()
            self.pubrate = rospy.Rate(50)
    def pub_cmd(self):
            while not rospy.is_shutdown():
                    if self.start:
                            self.pub_steercmd.publish(self.cmd)
                    self.pubrate.sleep()
    def recv_steeringcmd(self, msg):


            self.cmd.steering_wheel_angle_cmd = msg.steering_wheel_angle_cmd
            self.cmd.steering_wheel_angle_velocity = msg.steering_wheel_angle_velocity
            self.cmd.enable = msg.enable
            self.cmd.clear = msg.clear
```

```
            self.cmd.ignore = msg.ignore
            self.cmd.quiet = msg.quiet
            self.cmd.count = msg.count
            self.start = True

if __name__ == '__main__':
    node_instance = CmdKeeper()
    node_instance.pub_cmd()
        rospy.spin()
```

Throttle Command

```
#! /usr/bin/env python
import rospy
from dbw_mkz_msgs.msg import ThrottleCmd

class CmdKeeper:
    def __init__(self):
        rospy.init_node('ThrottleCmd_Keeper')
        rospy.Subscriber('/vehicle/SimulinkThrottleCmd', ThrottleCmd,
self.recv_throttlecmd)
        self.pub_throttlecmd = rospy.Publisher('/vehicle/throttle_cmd', ThrottleCmd,
queue_size=1)
            self.start = False
```

```
        self.cmd = ThrottleCmd()
        self.pubrate = rospy.Rate(50)
    def pub_cmd(self):
        while not rospy.is_shutdown():
            if self.start:
                self.pub_throttlecmd.publish(self.cmd)
            self.pubrate.sleep()
    def recv_throttlecmd(self, msg):

        self.cmd.pedal_cmd = msg.pedal_cmd
        self.cmd.pedal_cmd_type = msg.pedal_cmd_type
        self.cmd.enable = msg.enable
        self.cmd.clear = msg.clear
        self.cmd.ignore = msg.ignore
        self.cmd.count = msg.count
        self.start = True

if __name__ == '__main__':
    node_instance = CmdKeeper()
    node_instance.pub_cmd()
      rospy.spin()
```

REFERENCES

[1] Patrascu, D. (2010, Sept 22). Valeo Electric Show Car to Shine in Paris. Retrieved from https://www.autoevolution.com/news/valeo-electric-show-car-to-shine-in-paris-24640.html.

[2] H. Satonaka, M. Okuda, S. Hayasaka, T. Endo, Y. Tanaka, and T. Yoshida, "Development of parking space detection using an ultrasonic sensor," in Proc. 13th World Congr. Intell. Transp. Syst. Serv., Oct. 2006, pp. 1–10.

[3] S. Hiramatsu, A. Hibi, Y. Tanaka, T. Kakinami, Y. Iwata, and M. Nakamura, "Rearview camera based parking assist system with voice guidance," presented at the Proc. SAE World Congr. Exhib., Detroit, MI, USA, Apr. 2002, Paper 2002-01-0759.

[4] Ji, G., Hu, J., Zhu, Y., Gao, Z., & Yu, C. (2014). A Novel Method to Reduce Accumulative Switching Errors in Multi-Sensor Incremental Measurement Systems. ECS Transactions, 60(1), 863-868.

[5] Lima, P. F., Pereira, G. C., Mårtensson, J., & Wahlberg, B. (2018). Experimental validation of model predictive control stability for autonomous driving. Control Engineering Practice, 81, 244-255.

[6] Hucko, F. (2017). The development of autonomous vehicles (Master Thesis). Retrieved from https://projekter.aau.dk/projekter/files/260085639/Master_Thesis___The_development_of_autonomous_vehicles.pdf.

[7] NHTSA. (2016). Automated Vehicles for Safety. Retrieved from https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[8] Ivanov, A. M., & Shadrin, S. S. (2018, February). Development of autonomous vehicles' testing system. In IOP Conference Series: Materials Science and Engineering (Vol. 315, No. 1, p. 012011). IOP Publishing.

[9] Shuttleworth. J. (2019, Jan 7). SAE Standards News: J3016 automated-driving graphic update. Retrieved from https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic.

[10] Alaniz. A. (2018, Aug 7). Tesla Model 3 Outperforms In New IIHS Level 2 Autonomy Tests. Retrieved from https://www.motor1.com/news/261555/tesla-model-3-autonomous-best/

[11] Solís-Marcos, I., Ahlström, C., & Kircher, K. (2018). Performance of an additional task during Level 2 automated driving: an on-road study comparing drivers with and without experience with partial automation. Human factors, 60(6), 778-792.

[12] Karnouskos, S. (2018). Self-Driving Car Acceptance and the Role of Ethics. IEEE Transactions on Engineering Management.

[13] Huang, C., Lu, R., Lin, X., & Shen, X. (2018). Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles. IEEE Transactions on Vehicular Technology, 67(11), 11169-11180.

[14] Thorpe, C.; Hebert, M.H.; Kanade, T.; Shafer, S.A. Vision and navigation for the Carnegie-Mellon Navlab. IEEE Trans. Pattern Anal. Mach. Intell. 1988, 10, 362–373.

[15] Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The robot that won the DARPA Grand Challenge. J. Field Robot. 2006, 23, 661–692.

[16] Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.N.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the Urban Challenge. J. Field Robot. 2008, 25, 425–466.

[17] DARPA Urban Challenge. Retrieved form https://www.darpa.mil/about-us/timeline/darpa-urban-challenge

[18] Tesla Motors. (2016, Dec 20). Accelerating the world to sustainable energy. Retrieved from https://www.tesla.com/presskit/autopilot

[19] Elmenreich, W. (2002). An introduction to sensor fusion. Vienna University of Technology, Austria, 502.

[20] L. Wald. A European proposal for terms of reference in data fusion. International Archives of Photogrammetry and Remote Sensing, XXXII, Part 7:651–654, 1998

[21] B. V. Dasarathy. Information fusion - what, where, why, when, and how? Information Fusion, 2(2):75–76, 2001. Editorial.

[22] Chavez-Garcia, R. O. (2014). Multiple sensor fusion for detection, classification and tracking of moving objects in driving environments (Doctoral dissertation, Université de Grenoble).

[23] Fairfield, N. (2009). Localization, mapping, and planning in 3D environments. Ph. D. dissertation.

[24] Suhr, J. K., & Jung, H. G. (2014). Sensor fusion-based vacant parking slot detection and tracking. IEEE Transactions on Intelligent Transportation Systems, 15(1), 21-36.

[25] Loveday, S. (2018, Dec 19). 20 Best Self-Parking Cars in 2018. Retrieved from https://cars.usnews.com/cars-trucks/best-self-parking-cars.

[26] Wang, C., Zhang, H., Yang, M., Wang, X., Ye, L., & Guo, C. (2014). Automatic parking based on a bird's eye view vision system. Advances in Mechanical Engineering, 6, 847406.

[27] Suhr, J. K., & Jung, H. G. (2016). Automatic parking space detection and tracking for underground and indoor environments. IEEE Transactions on Industrial Electronics, 63(9), 5687-5698.

[28] Lee, S., & Seo, S. W. (2016). Available parking slot recognition based on slot context analysis. IET Intelligent Transport Systems, 10(9), 594-604.

[29] Zhang, L., Li, X., Huang, J., Shen, Y., & Wang, D. (2018). Vision-Based Parking-Slot Detection: A Benchmark and A Learning-Based Approach. Symmetry, 10(3), 64.

[30] Schoettle, B. (2017). Sensor fusion: A comparison of sensing capabilities of human drivers and highly automated vehicles. Ann Arbor: University of Michigan.

[31] Bosch MRR Radar Specification Sheet. (2019, Jan 11). Retrieved from https://www.bosch-mobility-solutions.com/media/global/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-

systems/predictive-emergency-braking-system/mid-range-radar-sensor-(mrr)/product-data-sheet-mid-range-radar-sensor-(mrr).pdf

[32] Eichelberger, A. H., & McCartt, A. T. (2016). Toyota drivers' experiences with dynamic radar cruise control, pre-collision system, and lane-keeping assist. Journal of safety research, 56, 67-73.

[33] Continental ARS 408 Long Range Radar Specification Sheet. (2019, Jan 11). Retrieved from https://www.continental-automotive.com/getattachment/5430d956-1ed7-464b-afa3-cd9cdc98ad63/ARS408-21_datasheet_en_170707_V07.pdf.pdf

[34] Breyer, B. (1997). Ultrasonic measurements and technologies: Sensor physics and technology series 4: S. Kocis, Z. Figura, Chapman & Hall, 1996, 218 pp. Ultrasound in Medicine and Biology, 23(9), 1442.

[35] Wing, M. G., Eklund, A., & Kellogg, L. D. (2005). Consumer-grade global positioning system (GPS) accuracy and reliability. Journal of forestry, 103(4), 169-173.

[36] Deng, X. X., Wang, X. N., & Zhu, J. (2014). Available lane detection based on radon transform. Paper presented at the, 1046 415-424. doi: 10.4028/www.scientific.net/AMR.1046.415

[37] OpenGL Programming Guide, OpenGL ARB, Addison-Wesley 1993, ISBN 0-201-63274-8. (chapter 3; see insight)

[38] By HSV_color_solid_cylinder.png: SharkDderivative work: SharkD Talk HSV_color_solid_cylinder.png, CC BY-SA 3.0, Retrieved from https://commons.wikimedia.org/w/index.php?curid=9801673

[39] Canny, J. (1987). A computational approach to edge detection. In Readings in computer vision (pp. 184-203). Morgan Kaufmann.

[40] Bonnici, D. 360-degree parking monitors explained. (2018, Feb 27). Retrieved from https://www.whichcar.com.au/car-advice/360-degree-parking-monitors-explained

[41] Kalman, R. E. (1960). Contributions to the theory of optimal control. Bol. soc. mat. mexicana, 5(2), 102-119.

[42] Kelly, A. (1994). A 3D state space formulation of a navigation Kalman filter for autonomous vehicles (No. CMU-RI-TR-94-19). CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

[43] Dubins L E. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents[J]. American Journal of Mathematics, 1957, 79(3): 497-516

[44] DeMuro D. (Nov 2016). 7 Self-Driving Car Features You Can Buy Now (and Some You May Already Have). Retrieved from https://www.autotrader.com/best-cars/7-self-driving-car-features-you-can-buy-now-and-so-259333

ABSTRACT

ROS-BASED SENSOR FUSION AND MOTION PLANNING FOR AUTONOMOUS VEHICLES: APPLICATION TO AUTOMATED PARKINIG SYSTEM

by

YUANZHE LI

May 2019

Advisor: Dr. Caisheng Wang

Co-Advisor: Dr. Chin-An Tan

Major: Electrical Engineering

Research and development in autonomous vehicles are currently very active since these vehicles are expected to play an important role in future transportation. During the past decade, numerous Advanced Driver Assistance Systems (ADAS) features have been developed and implemented on production vehicles, such as Adaptive Cruise Control and Lane Keeping Assist. However, most ADAS features are aimed to assist driving on highways where the environments are usually more structured and decision making can be made more easily. Non-highway environments, such as in the case of self-parking, are unstructured and require more complicated analysis of the image information, localization and path planning. Most camera-based auto-parking features are limited to partial autonomy, not intelligent enough to park a vehicle automatically without the participation of a driver. Camera-based auto-parking systems can detect parking spots but cannot accurately localize the vehicle as well as the parking spot. ds. These algorithms are specifically designed for ADAS under low velocity and requires high

precision vehicle maneuvering, by increasing the localization accuracy, in terms of vehicle and parking spot location. The APA system will help the driver to search for available parking spots and self-drive the vehicle into the parking spot safely. The strategic architecture of the auto-parking system includes environment perception, sensor data fusion, motion planning, and vehicle control. Perception algorithms such as line detection and object detection are discussed in this thesis. Sensor data fusion and data association using extended Kalman filter for parking spot and vehicle location tracking are developed in this thesis. Rapidly-exploring Random Tree (RRT) motion planning algorithm is used to generate a path, leading the vehicle to park into the parking spot. Simulations for sensor data processing, data association and motion planning are conducted in a Robot Operating System (ROS) environment. A versatile virtual environment with vehicle dynamics model and control algorithms are developed in the simulation environment. Parking spot detection, vehicle behavior decision making, and motion planning are tested based on virtual sensor signals modelled in ROS. Simulation results show that the vehicle can self-drive into the parking spot without the participation of a driver. Vehicle localization field experiments based on GPS sensor fusion has been conducted in an open parking lot. Localization accuracy of ego vehicle is improved.

AUTOBIOGRAPHICAL STATEMENT

Name: YUANZHE LI

Education: May 2019, M.S. Electrical Engineering, Wayne State University, Detroit, Michigan, U.S.A.

June 2018, B. S. Mechanical Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang, China.