IMPLEMENTATION OF AUTONOMOUS VEHICLES AT MICRON SINGAPORE FACTORY

SUN SHUO

INNOVATION & DESIGN PROGRAMME

NATIONAL UNIVERSITY OF SINGAPORE

2019

IMPLEMENTATION OF AUTONOMOUS VEHICLES AT MICRON SINGAPORE FACTORY

SUN SHUO

A THESIS SUBMITTED FOR THE DEGREE OF

BACHELOR OF ENGINEERING

(MECHANICAL ENGINEERING)

INNOVATION & DESIGN PROGRAMME

NATIONAL UNIVERSITY OF SINGAPORE

DECLARATION

I hereby declare that this thesis is my original work

and it has been written by me in its entirety.

I have duly acknowledged all the sources of information

which have been used in this thesis.

SUN SHUO

19th November 2019

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Prof. Marcelo H. Ang Jr., for his guidance, and inspirations for the past one year's time working on this project. And I would like to express my thanks to Prof William Lee and Krittin Kawkeeree from NUS ARC for their continuous guidance and support throughout the entire project.

I would also like to express my gratitude to my supervisors at Micron, Joshua Lee, and Anthony Lai, who have supported us and provided guidance on the project directions. It was my privilege to be involved in this exciting research opportunity.

I would sincerely thank my seniors and colleagues at Micron, namely Kelvin Kang, Quan En Goh, Shun Da Yar, Wei Jie Tan, Alexis Yuan Hui Khoo, Brina Wey Tynn Shong, Hao Ren Guo, Max Yi Neng Ang, Nicole Kye Ting Low, Xing Ting Lin, and Zhi Jie Zhao, for their valuable friendship and encouragement. It was indeed an honour to have the privilege of working with all of you, and you are the reason for making all these achievements possible.

I would like to dedicate this thesis to my family, especially to my parents and my grandparents for their unconditional support and understanding throughout the years. They have always been my constant motivation and my source of strength.

Table of Contents

Summary

This thesis focuses on the obstacle avoidance problem for autonomous driving in the structured road environment, with emphasis on the path planning capability. The aim is to design and develop a new software system based on the previous version that can expand the autonomous functions of the vehicle to detect and avoid obstacles it encountered during the mission safely.

A new system architecture based on the previous version was proposed to upgrade the vehicle systematically. First, a visual object detection system is added on the vehicle. This new system is fused with the LiDAR detection system to improve the detection accuracy and robustness. Second, a novel method of representing the geometric and topological information of the lane was proposed to support the path planning functions. Next, a new planning module that consists of a trajectory planner, a behaviour planner and a collision checker is developed to allow the vehicle to plan its path and make decisions based on scenarios. In the trajectory planner, the frenét optimal planning algorithm is implemented and improved to suit this specific application. A method of using a finite state machine is applied in the behaviour planner to allow the vehicle to switch between different driving behaviours. The dynamic virtual bumper approach is used in the collision checker to provide a safer and more accurate prediction of future collisions. In the control module, the previous version of the control algorithms was improved to adapt to the new autonomous capabilities, and a new approach of using Model Predictive Control is developed as a second control option for the vehicle. In addition, a new graphical user interface is designed to improve users' experience during the ride.

The methods mentioned above have been developed and integrated into the vehicle. The vehicle has been tested on the real structured road driving environment successfully at Micron campus in Singapore.

Keywords: *autonomous vehicle, obstacle avoidance, perception, behaviour planning, trajectory planning, collision checking, trajectory tracking.*

List of Tables

List of Figures

List of Symbols

x, y, θ	Position and heading
v	Velocity
a	Acceleration
δ	Steering Angle
t	Time
L	Vehicle wheelbase, distance between the front and rear axle
u	Input
$\vec{x}$	State vector
e	Error
$k_p,\ k_i, k_d$	Proportional, integral, derivative gain
k	Stanley gain
s, d	Longitudinal and lateral position in the frenét frame
$\dot{s}, \dot{d}$	Longitudinal and lateral velocity in the frenét frame
$\ddot{s}, \ddot{d}$	Longitudinal and lateral acceleration in the frenét frame
$\dddot{s}, \dddot{d}$	Longitudinal and lateral jerk in the frenét frame
$\vec{r}$	Origin of the selected frenét frame
$\vec{n}_r$	Normal vector of the selected frenét frame
L_{fw}	Look-ahead distance
κ	Curvature
η, ψ	heading difference between the waypoint and the vehicle
r_{true}	Obstacle's true radius
$r_{with\ margin}$	Obstacle's radius with margin
v_x, v_y	Obstacle's velocity's *x* and *y* component
x_{img}, y_{img}	Obstacle's location on the image in pixels
w_{img}, h_{img}	Obstacle's bounding box's extends on the image in pixels

W, H	Resolution of the image
β	Obstacle's angular position
α	Camera's Field of View
ϕ	Camera's heading in the vehicle's local frame
$x_{object,car}, y_{object,car}$	Object's position in the vehicle's local frame
$x_{cam,car}, y_{cam,car}$	Camera's position in the vehicle's local frame
x_{object}, y_{object}	Object's position in the global map frame
d_x, d_y	*x* and *y* components of the waypoint's lateral unit vector
$w_{left}, w_{centre}, w_{right}$	distances from the waypoint to the boundaries
v_{min}, v_{max}	Minimum and maximum speed
S_x, S_y	Longitudinal and lateral trajectory
a_i, b_i, c_i, d_i	Coefficients of the quintic polynomial for the *i*-th waypoint
x_{next}, y_{next}	Position of the closest next waypoint
x_{prev}, y_{prev}	Position of the closest previous waypoint
$\vec{n}_s$	Vector from the previous waypoint to the next waypoint
$\vec{n}_d$	Unit vector pointing in the lateral direction
$\vec{m}$	Vector from the previous waypoint pointing to the vehicle
$proj_{\vec{n}_s}\vec{m}$	The projection of vector $\vec{m}$ on vector $\vec{n}_s$
$proj_{\vec{n}_d}\vec{m}$	The projection of vector $\vec{m}$ on vector $\vec{n}_d$
v_{ref}	Reference velocity
T_{min}, T_{max}	Minimum and maximum sampling time
$\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$	Coefficients of the quintic polynomial
J_s, J_d	Longitudinal and lateral total squared jerk
Δv^2	Squared speed difference
Δd^2	Squared lateral offset
d_{ref}	Desired lateral position
$r_{obstacle}$	Obstacle radius

$r_{vehicle}$	Vehicle radius
a_{min}, a_{max}	Minimum and maximum longitudinal acceleration
a_d	Maximum lateral acceleration
κ_{max}	Maximum curvature
$\Delta\varphi$	Difference in heading
Δs	Difference in arc length
$k_{overall}$	Stanley overall gain
$cost_{cte}$	Cross-track error cost
$cost_{\psi}$	Heading error cost
$cost_{\Delta v}$	Speed difference cost
$cost_{\delta}$	Steering usage cost
$cost_a$	Accelerator usage cost
$cost_{\Delta\delta}$	Steering change cost
$cost_{\Delta a}$	Accelerator change cost
w_{cte}	Cross-track error weight
w_{ψ}	Heading error weight
$w_{\Delta v}s$	Speed difference weight
w_{δ}	Steering usage weight
w_a	Accelerator usage weight
$w_{\Delta\delta}$	Steering change weight
$w_{\Delta a}$	Accelerator change weight
$\delta_{min}, \delta_{max}$	Minimum and maximum steering angle

List of Abbreviations

DARPA	Defense Advanced Research Projects Agency
OEM	Original Equipment Manufacturer
Micron	Micron Semiconductor Asia Pte. Ltd
ARC	Advanced Robotics Centre
NUS	National University of Singapore
STEM	Science, Technology, Engineering and Mathematics
AV	Autonomous Vehicles
DBW	Drive-by-Wire
LiDAR	Light Detection and Ranging Device
IMU	Inertial Measurement Unit
SLAM	Simultaneous Localization and Mapping
UX	User Experience
GUI	Graphical User Interface
ETA	Estimated Time of Arrival
HD	High Definition
KITTI	KITTI Vision Benchmark Suite
AP	Average Precision

IoU	Intersection of Union
FOV	Field of View
FPS	Frames Per Second
AMCL	Adaptive Monte Carlo Localization
RNDF	Route Network Definition File
PID	Proportional-Integral-Derivative
MPC	Model Predictive Control
ROS	Robot Operating System
Rviz	ROS visualiser
CG	Centre of Gravity
POMDP	Partially Observable Markov Decision Process

1. Introduction

1.1. Research Motivations

1.1.1. Mobility using Autonomous Vehicles

The concept of autonomous driving has been raised as one of the most topical and promising research topics in the last decade. In 2005 and 2007, the Defence Advanced Research Projects Agency organised two Grand Challenges in the US regarding autonomous vehicles. Many research institutes and universities joined these challenges and gained interests in this new field of research. Google was the first industrial company who showed interests in autonomous vehicles, and it recruited the winning team of the 2005 DARPA Grand Challenge from Stanford University in 2009. After Google, many famous tier-1 automotive company and OEMs joined this race, such as Audi, Daimler, Bosch, and BMW. Meantime, many self-driving car start-ups started to emerge like nuTonomy, Zoox, and GM Cruise. Until today, a series of breakthroughs in autonomous driving technology has been made. Many of them have developed vehicles with a high level of autonomy and started conducting trials in cities across the globe.

It has been widely believed that autonomous driving technology could provide improved road safety, reduced traffic time cost, and greater convenience to public transportation. According to the World Health Organization, in 2016, about 1.35 million people died due to road accidents globally [1]. While in the US, 94% of vehicle accidents were due to human choice or error [1]. By eliminating human errors, autonomous vehicles could effectively reduce the number of vehicle accidents and protect millions of lives. Statistics also showed that about 42 hours of life per person had been wasted on the road annually in the US [1]. From a financial perspective, the increase in productivity and leisure could be $1.2T per year in the US by saving people’s time spent on commuting and waiting [1].

Furthermore, self-driving cars offer a more convenient and accessible way to everyone, regardless of age and physical conditions, to travel without limitation. This technology has also inspired the Robo-taxi and the cab-sharing business to shape the future of mobility. In conclusion, people believe that autonomous driving technology owns the potential to improve the living standards and productivity of society profoundly.

1.2. Background

1.2.1. Project Launch

This autonomous vehicle project was initiated by both Micron Foundation and the Advanced Robotics Centre, the National University of Singapore as a joint project in support of STEM education and research. Micron Technology is an international semiconductor company based in Boise, Idaho. The company's scope of business is mainly related to the manufacturing, and commercialization of semiconductor products such as memory and storage products. As its factory expanding in Singapore, the need for innovative solutions in transporting team members and materials between facilities emerged. This project aimed to develop a fleet of autonomous vehicles for internal shuttle service within Micron facilities. This project also aims to explore possibilities of using Micron memory and storage solutions on autonomous vehicles, making decisions and enabling intelligence for autonomous driving. Micron Singapore campus provided a development and testing environment to carry out the project.

1.2.2. Prior Work

The project formally started in January 2019. During the first phase, from January to June, the team’s main objective was to develop three Drive-by-Wire (DBW) vehicles platforms and equip them with as proof-of-concept prototypes to demonstrate necessary autonomous capabilities. The first batch of interns retrofitted three electric golf buggies with steering motors, brake motors, throttle controllers to allow computers to fully control the driving motion of the vehicles vis electrical signals. As shown in Figure 1.1, the vehicles were each equipped with one 3D LiDAR on the top of the vehicle, and one 2D LiDAR on the front bumper to perform localization and obstacle detection functions. At the same time, there were two encoders mounted on the back wheels and an IMU sensor under the roof of the buggy to support localization. Furthermore, various electrical components such as additional batteries, power distribution boards, ventilation, were mounted on the buggies in support of all kinds of onboard sensors, actuators and computing devices.

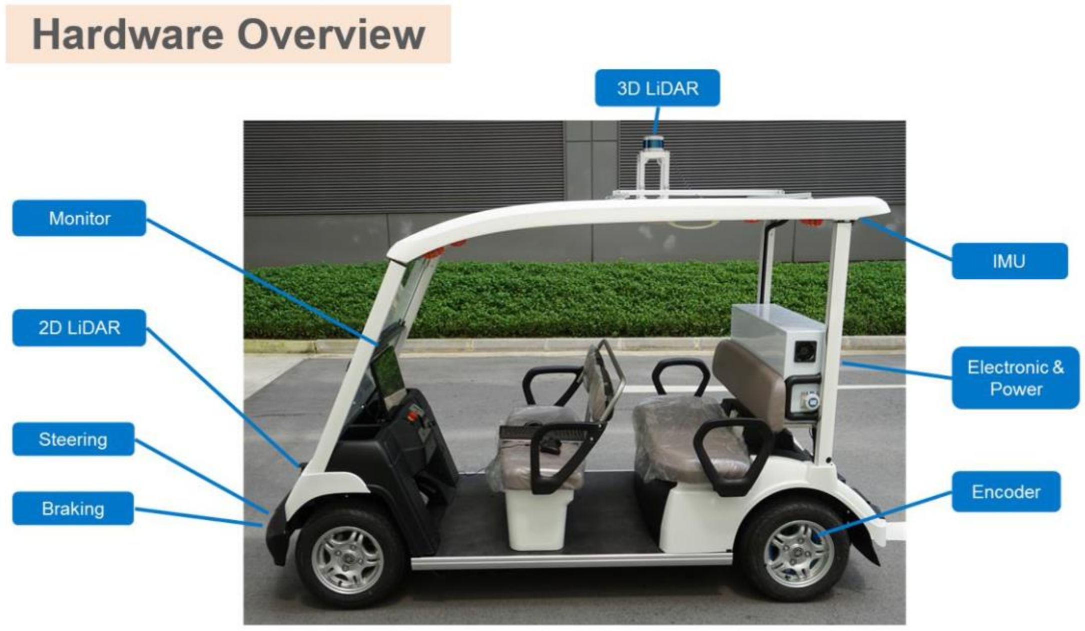

Figure 1.1: Autonomous buggy hardware overview

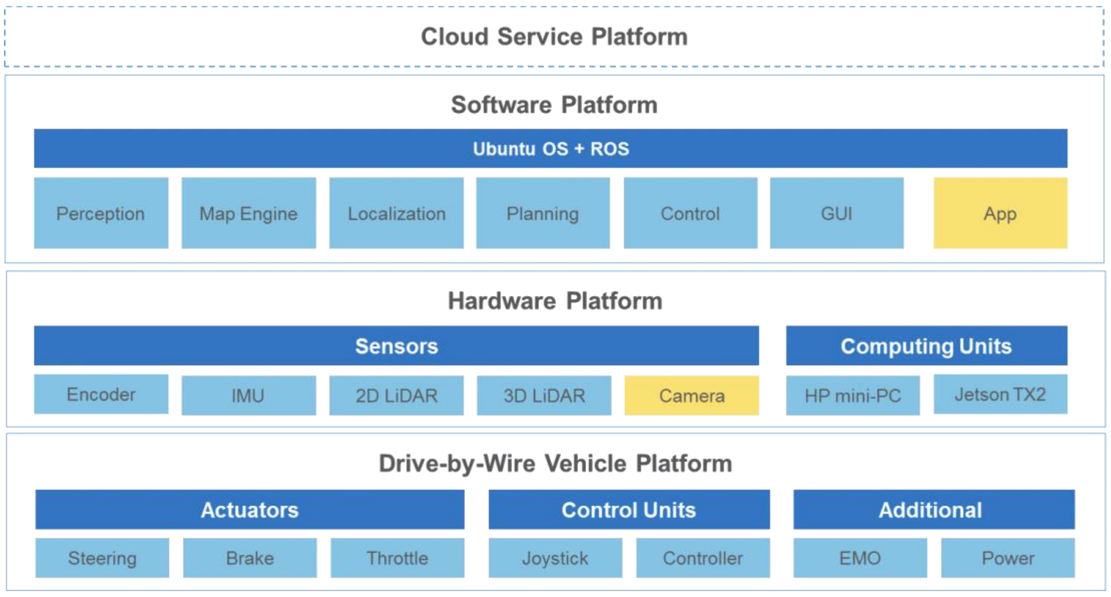

Figure 1.2: Overview of the autonomous buggy software stack

Figure 1.2 shows an overview of the autonomous driving software system that was developed during phase one. This software architecture enabled the buggy to perform necessary autonomous capabilities such as SLAM, path following, and obstacle detection. By the end of the first phase, the buggy was able to locate itself in the Micron campus, travel from point A to point B by following a predefined path, and stop when obstacles were detected. There were safety drivers on the buggy at all time who can switch to manual mode autonomous mode, and take over the control immediately in case of emergencies.

1.3. Scope of the Thesis

The main contribution of this Bachelor thesis is the solution to the problem of obstacle avoidance for autonomous vehicles in the structured road driving environment. The solution mainly comprises of the newly designed lane representation format, a behaviour planner and a trajectory planner in the planning module. This thesis also covers the design and development of new software architecture, several upgrades and improvements to the other modules such as perception and control. The methods described in this thesis are implemented and tested on the autonomous vehicles at Micron.

The first area that this thesis will cover is an overview of the new software architecture. A systematic plan of upgrades and improvements is proposed to enable the obstacle avoidance capability on the vehicle.

The second area that this thesis will cover is the implementation of the map and planning module, which enabled the new obstacle avoidance capabilities on the vehicle. A novel method of lane representation for autonomous driving is proposed to support path planning functions. The actual implementation of the frenét optimal planning algorithm and several new improvements to improve its performance are discussed in this thesis. Furthermore, a finite state machine is developed as the behaviour planning that handles decision-making process. A newly designed collision checker using dynamic virtual bumper is designed to improve vehicle safety and manoeuvrability.

This work also covers the upgrades and improvements to the perception, control, and the user interface module. A visual object detection system using existing 2D object detection model is presented in the thesis. A new control module using the Model Predictive Control algorithm is proposed to improve its trajectory tracking performance. In addition, a newly designed graphical user interface to improve user experience is included as well.

Each of these areas is incorporated into a holistic framework for autonomous driving in a structured road environment, with emphasis on the obstacle avoidance function. The new system described in the thesis has been successfully tested in the real world environment at Micron campus in Singapore.

1.4. Contributions

The main contributions of this thesis can be summarized as follows:

- A new software architecture with a systematic plan of upgrades and improvements to the old system is proposed to enable the obstacle avoidance capabilities.
- A novel lane representation format in the map module that incorporates geometry and topological information associated with lanes and can be used to support the planning module.
- Implementation of the frenét optimal planning algorithm in the trajectory planner to enable the trajectory generation and obstacle avoidance on the vehicle. A series of improvements to the algorithm to improve its flexibility and robustness and make it suitable for the specific application.
- A behaviour planner is designed in the form of a finite state machine which is in charge of the decision-making process and controls the vehicle's behaviour on the road. This behaviour planner takes into account the geometry of the lane, state of the vehicle and other obstacles on the road and chooses the most suitable vehicle state accordingly.
- A new collision checker that adopted the dynamic virtual bumper method to predict the incoming collisions. The vehicle's safety and manoeuvrability are improved by using the new collision checker.

- Implementation of Model Predictive Control in the control module in replace of the Proportional-Integral-Derivative controller and the Stanley controller to achieve a better trajectory tracking performance.
- A newly designed graphical user interface to allow the users to operate the vehicle and improve users experience.

These contributions are integrated on the vehicle and tested in the real driving environment at Micron. The new software system is able to detect obstacles, make a proper decision and generate a safe and smooth trajectory for the vehicle to follow. The vehicle can demonstrate the desired obstacle avoidance capabilities on the road.

1.5. Thesis Outline

This thesis focuses on the realisation of the obstacle avoidance capability on the vehicle in a structured road environment. After this introductory chapter, the rest of the thesis is organised as follows: Chapter 2 is a literature review of the research topics in autonomous vehicles field. Chapter 3 introduces the overall architecture of the new software system and the improvements to each module. The following two chapters (Chapter 4, 5), discusses the integration of the visual object detection system using cameras with the LiDAR obstacle detection system. Chapter 4 describes the new proposed sensor layout, including both new cameras and LiDARs. Chapter 5 discusses the selection of 2D visual object detection models and the fusion between camera and LiDAR detection results. In Chapter 6, a novel method of representing lanes is proposed to assist with the path planning module. Chapter 7 covers the implementation of frenét optimal planning algorithm in trajectory planner, the design of the finite state machine in behaviour planner, and the design of the collision checker using the dynamic virtual bumper method. Chapter 8 discusses the design of a Model Predictive Control algorithm as a second control option in the control module. Chapter 9 presents the design of the new graphical user interface for the passengers. Finally, Chapter 10 concludes this thesis and suggests directions for future works. Additionally, a user's survey is presented in Appendix A.

2. Literature Review

2.1. Autonomous Vehicle Architecture

A typical autonomous driving software stack consists of three major modules, perception, planning, and control [2]–[5], with each in charge of one specific area of tasks. The perception module is responsible for processing and fusion of the raw data from various sensors to fully understand the environment around the vehicle. Planning module is designed to enable the robot to make decisions about it next step's behaviour and trajectory based on all the information provided to it. In the end, those decisions will be executed by the control module, which controls the steering, accelerating and braking of the vehicle. In addition to the three models mentioned above, another two supporting modules that are commonly seen on autonomous vehicles are map module and fleet management module. The map module is designed to serve as a centralized database for all information relevant to the HD map and the fleet management module is in charge of the communication and coordination between vehicles and the command centre.

2.2. Perception

Perception tasks can usually be divided into three categories, including classification, detection, and semantic segmentation, with an increasing level of refinement [5]. Classification refers to the ability to differentiate the class that this object belongs to, such as a car, person, cyclist, barrier or part of a building. Based on the classification capability, detection can identify all the individual objects in the current frame, and semantic segmentation can provide fine classification at the pixel level.

Based on the primary sensors used, perception algorithms can be categorized into three major types, namely visual perception, LiDAR perception, and the combined methods [5]. The performance of these algorithms varies significantly and inherently depend on the nature and characteristics of the sensors they use [6]. Until today, the best object detection results achieved on the KITTI benchmark suite are dominated by LiDAR detection methods or LiDAR-camera mixed methods [7]. Among these algorithms, many state-of-the-art object detection methods using point cloud data reached above 70% AP, and some of them, such as [8], [9], even reached an impressive AP of above 77%. In comparison to LiDAR-based methods, the current state-of-the-art camera-based solution, [10], and its variations, only managed to reach an AP of 44.56%. Meanwhile, a large portion of solutions still tends to use cameras as a supplement of LiDAR point clouds, and the best method among them currently is [11].

Table 2.1: Comparison of sensors used in autonomous driving

Characteristics	**Camera**	**LiDAR**
Cost	Low	High
Reliability	Medium	High
Resolution	High	Low
Distance Measurement	Stereo Cameras RGB-D Cameras	Yes
Object Detection Accuracy	Low	High

When comparing the strengths and weaknesses of the camera and LiDAR (Table 2.1), the cost is often referred to as the most notable factor. The cost of a typical multilayer LiDAR could easily reach thousands of USD, while that of a camera with lens is many times cheaper. However, the ability to provide high reliability and accurate depth-sensing results make LiDAR a better candidate than a camera. It is a common practice to use both sensors together on the same autonomous vehicle to balance the robustness and the cost of the solution. It is also favourable for the industry to use many different kinds of sensors to supplement each other and provide redundancy [12], [13].

2.3. Vehicle and Lane Modelling

2.3.1. Kinematic Car Model

A widely used kinematic model for the wheeled vehicle is the Ackermann steering geometry model. This geometry assumes that during turning, all the wheels of a vehicle will turn at different radii but share the same centre of rotation. Under this assumption, there is no side slipping motion between the ground and wheels, which greatly simplified the wheels' motion to a pure rotational motion with no lateral acceleration.

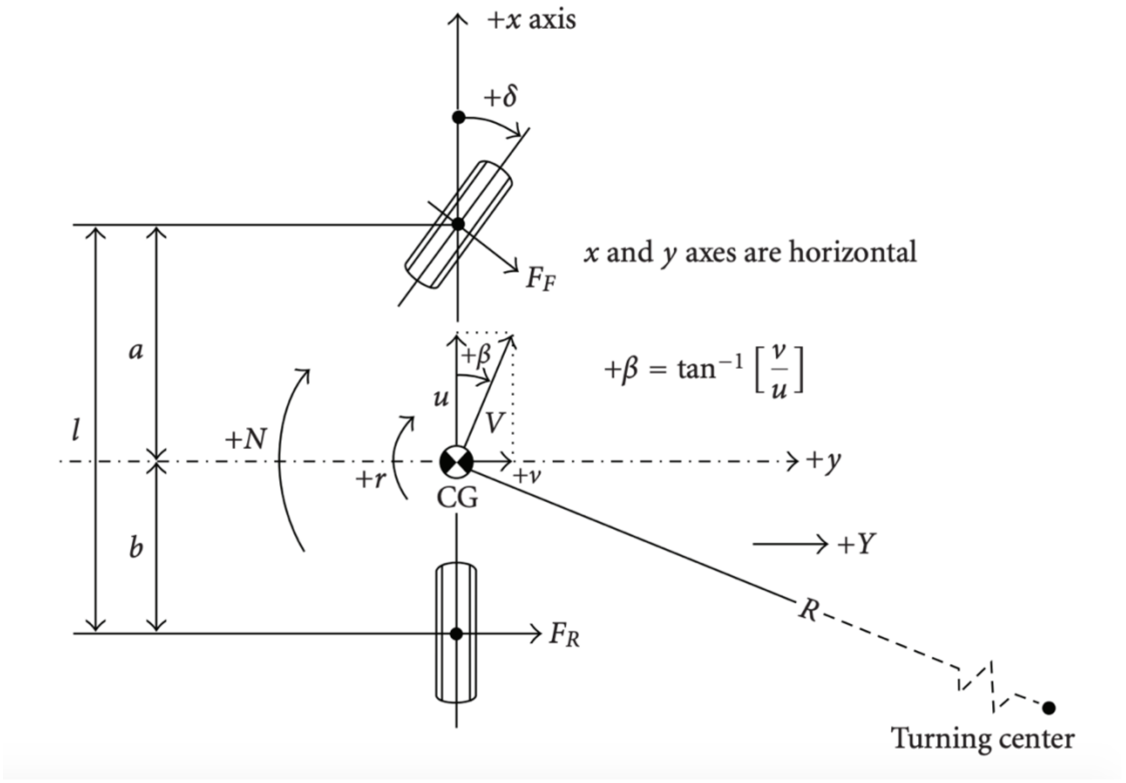

Figure 2.1: The bicycle kinematic model, adapted from [14]

In many occasions, a bicycle model is used as a simplified model for four-wheel vehicles. This model combines both the front wheels and both rear wheels to one pair of front wheel and rear wheel located along the centreline of the vehicle body, as shown in Figure 2.1. The vehicle's translational motion in the global inertia frame can be expressed in terms of its velocity v as:

$$\dot{x} = v \cos\theta \tag{2.1}$$

$$\dot{y} = v \sin\theta \tag{2.2}$$

The relation between the yaw rate ($\dot{\theta}$) and the steering angle (δ) can be derived based on this model:

$$\dot{\theta} = \frac{v}{L} \tan\delta \tag{2.3}$$

where *L* refers to the distance between the vehicle's front axle and the rear axle (wheelbase).

However, this method is a linear representation which disregards forces acting on the tires and slippage between the wheels and the ground. This simplification of the real-world conditions is not always accurate at different speed and steering angle conditions. To address this issue, a dynamic vehicle model was proposed. Nevertheless, due to its complexity in nature and nonlinearity, it is rarely used in real applications.

2.3.2. Lane Representation

Since most of the autonomous driving tasks are in a structured environment with clearly defined lane markings and traffic rules, autonomous vehicles must store these data in advance or be able to recognise this information on the fly. The HD map for autonomous driving typically includes both topological and geometrical representations of the road network, detailed to sub-lane level. To store road networks efficiently, many lane representation formats had been invented. Some methods that had been widely used were RNDF [15], Lanelets [16]. RNDF took advantage of a hierarchical structure, organising information into three different levels, from road segment level, lane level, to waypoint level. Each road segment contains several lanes (or open spaces), and each lane contains a series of waypoints which are represented by their absolute coordinates. Some descriptive parameters such as the width of each lane, stop lines, exit points and entry points. Lanelets method divides the whole road network into small lanelets, and each lanelet consists of two-lane boundaries that are represented by a series of connected nodes. Each node contains information about its previous nodes, next nodes, left and right neighbours, stopping line, speed limit, and other information deemed crucial for autonomous driving.

2.3.3. Frenét Coordinate

In most of the driving scenarios, lanes tend to have a curvature instead of being perfectly straight, which makes representing curved lanes or trajectories in a Cartesian coordinate (the normal map frame) tedious and complicated. However, human drivers tends to drive and perceive trajectories with respect to the curvature of the lane instead of the fixed map frame. It is natural for them to make decisions based on the centrelines of each lane. To simplify the calculation and mimic human behaviour, a widely used tool in path tracking control theory is the frenét frame (Figure 2.2), which is a moving coordinate system based on a reference curve, in this case, the centreline of the lane. The frenét coordinate is defined by two vectors, the tangential vector $\overrightarrow{t_r}$ and the normal vector $\overrightarrow{n_r}$, at each point on the reference curve. Frenét frame can guarantee the same tracking performance under the action of the following special Euclidean group:

$$SE(2) = SO(2) \times R^2 \tag{2.4}$$

This property of the frenét frame provides a convenient way of representing trajectories. Instead of solving the problem in the global Cartesian frame, it can be transformed into a dynamic local frenét frame with one point on the centreline of the lane as the origin. The position of vehicles or other objects on the road thus can be represented by a longitudinal component s and lateral offset d. The target trajectory, $\vec{x}(s, d)$, can be represented in frenét frame as:

$$\vec{x}\big(s(t), d(t)\big) = \vec{r}\big(s(t)\big) + d(t)\, \vec{n}_r\big(s(t)\big) \tag{2.5}$$

where $\vec{r}$ is the origin of the selected frenét frame.

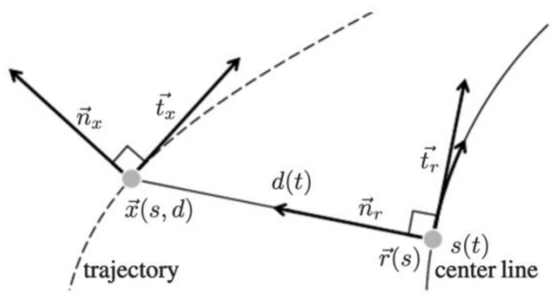

Figure 2.2: Transform from Cartesian coordinate to frenét frame, adapted from [17]

2.4. Planning

The planning module in autonomous driving typically consists of three planners, the route planner (or global planner), behaviour planner, and the motion planner (often referred to as trajectory planner or local planner), with each in charge of a task at different levels [5], [18], [19]. The global planner will search for an optimal global route to the destination based on certain criteria as the reference route for the downstream planning stack after receiving the destination and map data. Then, the behaviour planner will receive the global route, take into account the perceived information about the environment from the perception module, and make behavioural decisions such as follow, stop, overtake, shift lane, and yield, at real-time. In the end, the motion planner will generate a trajectory that is free of collision and comfortable for the vehicle to execute.

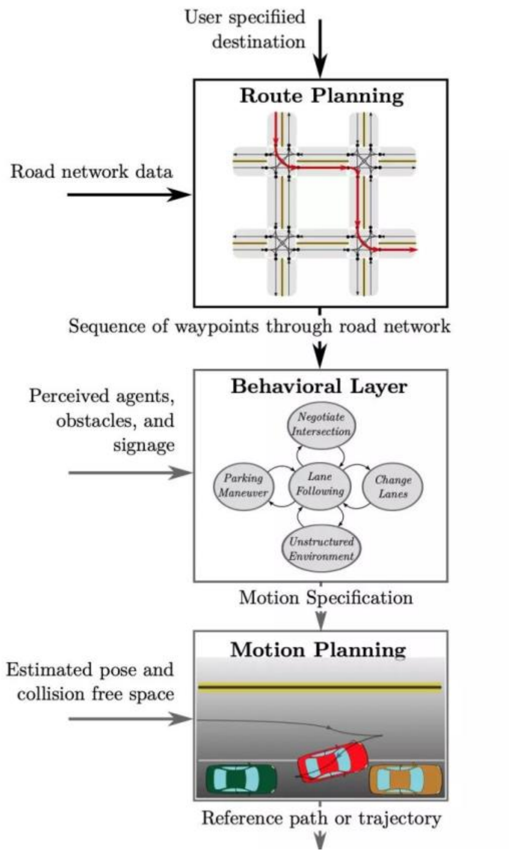
User specifiied destination
Route Planning
Road network data
Sequence of waypoints through road network
Behavioral Layer
Perceived agents, obstacles, and signage
Negotiate Intersection
Parking Maneuver
Lane Following
Change Lanes
Unstructured Environment
Motion Specification
Motion Planning
Estimated pose and collision free space
Reference path or trajectory

Figure 2.3: A typical hierarchy of path planning module, adapted from [18]

2.4.1. Route & Trajectory Planning

Route planners and motion planners are very similar, from the perspective of the algorithms used, except for the difference in the scale of the problem. As a fundamental problem in robotics, a significant number of algorithms have been developed to enable robots to find a path by themselves. Generally, path planning algorithms can be divided into five categories, namely, search-based methods, sampling-based methods, optimisation methods, parametric curve methods, and potential filed methods [18]. Among them, famous search-based methods such as A*, hybrid A* [3], and D* [4], first discretise the map into small grids and apply traditional searching algorithms to retrieve the solution. Sampling-based methods such as RRT [20] and its variations [21], [22], first randomly sample points in the state space and then connect the points based on some criteria and form a path among them. Optimisation methods directly formulate the path planning problem as a three-dimensional (two-dimensional space plus the time domain) optimisation problem, decouple the problem and then solve it at lower dimensions [23]. Parametric curve methods take the kinematic and kinetic constraints of the vehicle into account and generate smooth and continuous paths using different types of parametric curves, such as Dubins curves, Reeds-Sheep curves, polynomials, Bezier curves, and splines. Potential field methods like [24], [25] are inspired by the attractive forces and repulsive forces in magnetic fields and utilise the similar idea to navigate the vehicle through the pulling force of the goal and the pushing force of obstacles.

Table 2.2: A comparison of planning algorithms

Methods	**Completeness**	**Optimality**
Search-based	Complete	Optimal
Sampling-based	Probabilistically Complete	Asymptotically Optimal
Optimisation	Not Complete	Local Optimal
Parametric Curves	Not Complete	Local Optimal
Potential Field	Not Complete	Local Optimal

Two important considerations when assessing path planning algorithms are completeness and optimality. A comparison of these five types of planning algorithms in terms of their completeness and optimality is shown in Table 2.2. Completeness refers to the algorithm’s ability to guarantee a solution whenever the solution exists. Probabilistically complete means that the algorithm is able to find the solution if given enough time and the solution exists. Optimality differentiates whether an algorithm is able to find the optimal solution under a certain criterion. Algorithms that go through finite searching iterations and the result after each iteration is closer to the global optimal solution are called asymptotically optimal.

2.4.2. Behaviour Planning

Behaviour planner is in charge of making decisions for the vehicle to operate in a dynamic and complex environment. Currently, there are four different types of methods used as behaviour planners, including finite state machines, decision trees, knowledge-based inference system, and value-based decision processes [5], [18]. Finite state machine [3] and decision tree methods solidify the designed vehicle behaviour states into a fixed structure and switch between these states based on logical conditions or cost functions. Knowledge-based inference system establishes links between driving scenarios and the corresponding driving behaviours of a human driver. This prior knowledge is stored in the database offline and looked up by the system to inference the behaviour at runtime [26], [27]. Value-based decision processes usually model driving behaviours and define their utility or value functions to quantitatively assess each driving behaviour based on certain rules and strategies. For autonomous vehicles, these value functions usually include safety, comfort, and efficiency aspects of the driving. Famous methods in this field, such as [28]–[30], decide on the optimal driving behaviour based on these models.

2.5. Vehicle Control

The control module serves the function of converting the desired vehicle motion to suitable control signals to the actuators. The vehicle pose and the desired trajectory are sent to the control module and being processed to deduce the desired control variables. For a typical wheeled vehicle, the controlled variables are usually the acceleration (deceleration) and the steering angle, while their corresponding actuators are the throttle, the brake, and the steering wheel.

Classical control methods often refer to feedback controllers that are widely used in various control problems. Feedback control measures the system’s response and compensates for any disturbance deviating the control variables from the desired set-point. The feedback control can counteract adverse effects due to external disturbances, errors in system modelling, an imperfection in parameter tuning, and measurement noise. However, only using feedback terms in a controller is not sufficient to deal with all kinds of scenarios. Feedback controllers tend to have a delay in response as the errors measured are from the previous time step. Another limitation for purely feedback controllers is that they suffer from responding to all disturbances in the same way, while in many complex scenarios, it is more reasonable to respond differently.

2.5.1. PID Controller

The Proportional-Integral-Derivative (PID) controller is one of the most popular control methods in this field, as it is simple, powerful and does not require modelling of the system. The control output u is dependent on three individual terms, the proportional term, the integral term, and the derivative term:

$$u = K_p e + K_i \int_0^t e \, dt + K_d \frac{d}{dt} e \tag{2.6}$$

where e is the error measured by the sensors, K_p, K_i, and K_d are respectively the proportional gain, the integral gain, and the derivative gain. This method is commonly used in controlling the longitudinal motion of the vehicle, i.e. the accelerator and the brake. In practice, a PI controller or a PD controller is sometimes used in certain applications as a simplified version.

2.5.2. Pure Pursuit Steering Control

Pure Pursuit is a classic method for controlling the vehicle’s lateral motion. This method pursues a waypoint that has a certain look-ahead distance L_{fw} in front of the vehicle’s current position. As shown in Figure 2.4, the curvature κ of the arc starting from the centre of the vehicle’s rear axle to the look-ahead waypoint can be derived as:

$$\kappa = \frac{2 \sin \eta}{L_{fw}} \tag{2.7}$$

where η is the heading difference between the look-ahead waypoint and the vehicle.

Based on the kinematic bicycle model, the steering angle δ can be derived as:

$$\delta(t) = tan^{-1}\left(\frac{2L\, sin(\eta(t))}{L_{fw}}\right) \tag{2.9}$$

where L is the distance that the vehicle travelled with respect to the starting position. In practice, the look-ahead distance is often expressed as a function of the vehicle's velocity v:

$$\delta(t) = tan^{-1}\left(\frac{2L\, sin(\eta(t))}{kv(t)}\right) \tag{2.10}$$

where k is the tuneable gain parameter.

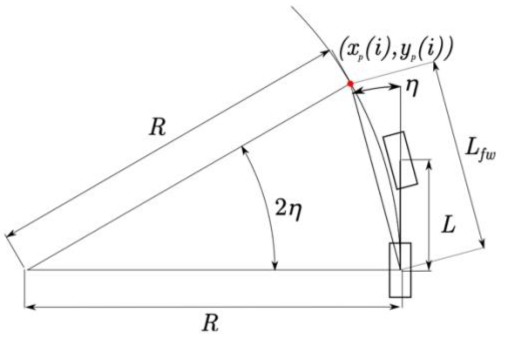

Figure 2.4: Pure pursuit steering control, adapted from [5]

Pure pursuit method is able to deal with external disturbances and significant offset error from the trajectory. However, this method heavily relying on the tuning of the gain parameter and suffer from corner-cutting and steady-state offset problem, especially at higher speeds, when the look-ahead distance tends to become too large.

2.5.3. Stanley Steering Control

Stanley steering controller is invented by the 2005 DARPA Challenge winning team from Stanford University. According to their paper, the desired steering angle comprises two terms, the heading difference term $\psi(t)$ and the cross-track error term $x(t)$. The heading difference term is the difference in heading between the closest next waypoint on the trajectory and the vehicle's current heading, while the cross-track error term refers to the vehicle's lateral offset at the front axle from the current trajectory. The final control output is given by:

$$\delta(t) = \psi(t) + tan^{-1}\left(\frac{k\,x(t)}{v(t)}\right) \tag{2.11}$$

where the k is a gain value and $v(t)$ is the vehicle's current velocity. The sign of the cross-track error term depends on which side the vehicle is on with respect to the heading of the trajectory.

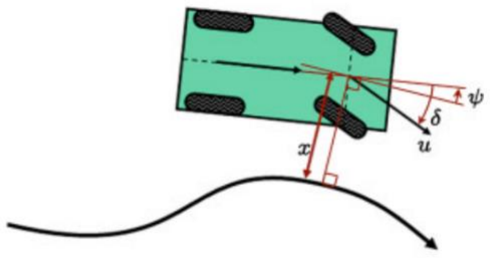

Figure 2.5: Stanley steering control geometry, adapted form [2]

Similar to the pure pursuit method, Stanley steering controller only has one gain parameter for tuning. However, the Stanley steering controller has better tracking results as it converges to zero cross-track error exponentially and does not cut corners as it uses the heading difference term and cross-track error term to gauge the compensation. It also performs better at higher speeds.

2.5.4. Model Predictive Control

Model Predictive Control (MPC) is a class of feedback control algorithms that uses a model of the system to predict the system's future behaviours and solve for a series of control outputs through an optimizer. Depending on the model it uses, MPC can be classified into linear and non-linear MPC. The typical structure of MPC is shown in Figure 2.6. The advantage of MPC is that it can handle heavily coupled multi-input multi-output (MIMO) systems that other control methods cannot solve. It also takes into account constraints, including hard constraints that the control law cannot violate and soft constraints that the system desired to obey. The most dominating factor that makes MPC a powerful controller is that it possesses the ability to predict into the future and preview the states of the system at a future time and make adjustments accordingly.

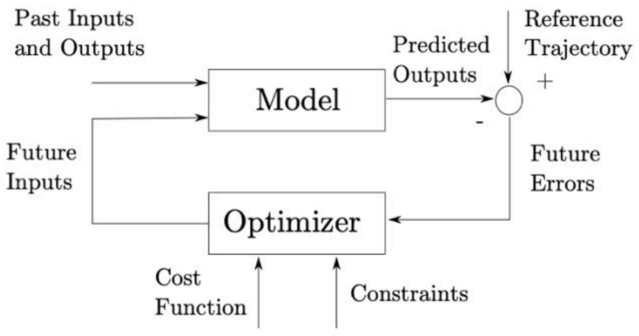

Figure 2.6: The typical structure of MPC, adapted from [5]

However, the disadvantage of this method is that it relies heavily on the accuracy of the modelling of the system, which sometimes might be difficult to achieve. Another limitation is that MPC often demands a huge amount of computing power and memory. In the case of operating an autonomous vehicle, the major challenges exist at deploying a non-linear MPC on the onboard computers.

3. Architecture Design

3.1. Driving Environment

The most dominating factor that guides the design of the architecture and functions of a type of autonomous vehicles is the driving environment. The selection of sensors and the level of complexity of algorithms largely depend on the actual environment of the application. Comparing to typical urban streets, which most of the existing research have been focusing on, the traffic condition in Micron has several key differences from that (as shown in Table 3.1).

The drivable space along the selected route in Micron was mostly structure road with little open space, which made it possible to apply existing algorithms that were designed for structure road environment. Another advantage it brought was that the majority of the other road users obey the traffic rules, which made it easier and faster for the decision process to find suitable driving behaviour.

Table 3.1: Comparison between different driving environments

Conditions	Micron Internal Road	Typical Urban Road
Road Type	Two-way 2-lane fire engine access	4-lane/ 6-lane dual carriageway
Lane Markings	Yes, except for certain area	Yes
Traffic Lights	No	Yes
Traffic Speed	< 20 km/h	60 km/h
Traffic Intensity	Vary from area	High
Major Traffic Participants	Container/Tank trucks Engineering trucks Special vehicles Forklifts Workers	Cars Cyclists Pedestrians

However, one challenge it posed was that roads in Micron are mostly two-way 2-lane fire engine access, with curbs on both sides of the road. This condition required high localization and mapping accuracy as well as high planning and control accuracy to keep the vehicle staying at the centre of the lane. The vehicle may not be able to rely on the widely used lane detection and traffic sign detection algorithms as there were no lane markings along curtain selected route. Since there were no traffic lights and little traffic signs, it was not necessary to equip the vehicle with traffic light and traffic sign detection algorithms. Instead, the HD map could store this information as prior knowledge for the vehicle’s reference.

The traffic speed and the traffic intensity were not high in Micron, which alleviated the constraints on sensors and computing devices. However, the drastic difference in traffic participants from typical urban streets had been recognized as the most challenging problem. Huge trucks tended to have different appearances, outer dimensions and features, which made the detection and tracking of such vehicles difficult. The decision-making process must be able to account for these vehicles as they tended to have very different driving behaviours from normal cars. Around the loading dock area or near certain facilities, it was sometimes the case that the road was entirely or partially block by huge parked trucks. These unusual scenarios that rarely seen on normal road posed challenges and risks for autonomous vehicles. The only way to handle these circumstances was to conduct more road tests and solve corner cases by adding additional features in the system.

3.2. Software Overview

With reference to several existing autonomous vehicle software architectures, a new software stack was developed for this application to achieve more advanced autonomous driving capabilities (Figure 3.1). The new software stack was designed and packaged into modules according to their primary functions for better modularity and easier future upgrades and maintenance. The arrows showed the flow of information between modules.

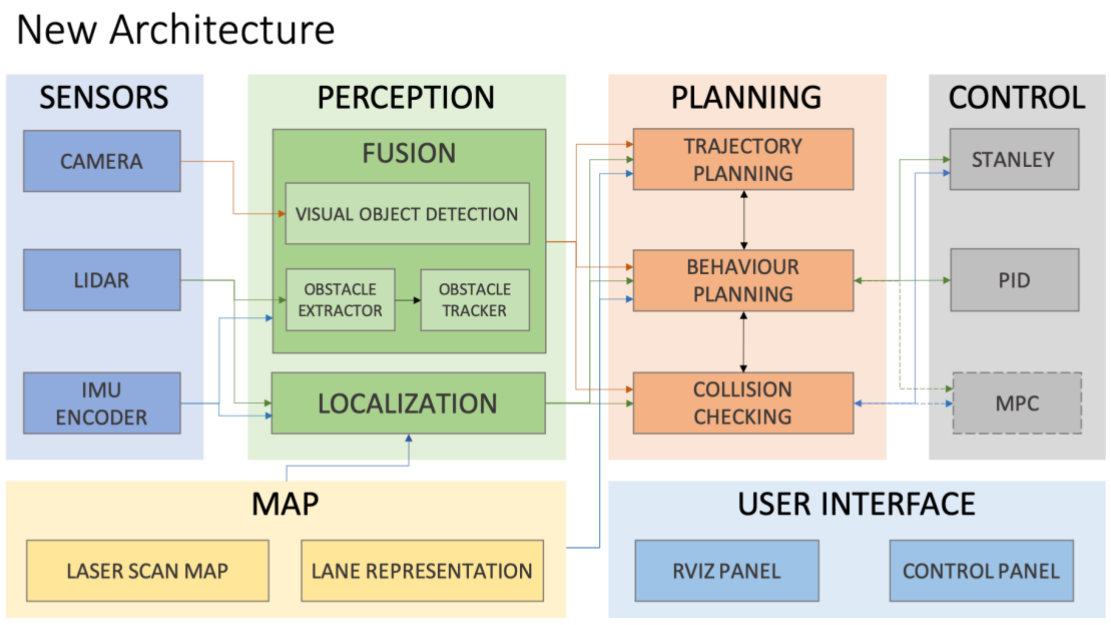

Figure 3.1: Overview of the new system architecture

In this new architecture design, the new perception module included the visual object detection algorithm and performed the task of fusing information from both LiDAR and camera. A behaviour planner and a trajectory planner had been added and packaged as the planning module. The previous control module had been upgraded, and an MPC had been newly developed as an alternative control option. Besides the original laser map, more information was integrated into the new map module to better support new autonomous functions. Additionally, an onboard GUI control panel was developed for the users to operate the vehicle as a separate software package.

4. Sensors

4.1. Sensor Layout

To upgrade the vehicle’s ability of perceiving the environment, the sensors FOV and their positions need to be redesigned to improve the sensor coverage. During the design stage, many sensor layouts have been proposed and compared.

Table 4.1: Selection of the LiDAR layout

Criteria (Weightage)	*Baseline (Previous version)*	*Design A*	*Design B*	*Design C*
3D LiDAR	One at the centre top of the buggy	One at the centre top of the buggy	One at the centre top of the buggy	Two at the two sides of the top
2D LiDAR	One at the front	One at the front left corner One at the rear right corner	One at the front Two at two rear corners	One at the front Two at two rear corners
Cost (25%)	0	-1	-2	-5
Coverage (50%)	0	+3	+3.5	+5
Aesthetic (25%)	0	-2	-1	-2
Total Score	0	0.75	**1**	0.75

As shown in Table 4.1, the comparison indicated that design B was the best option for the new LiDAR arrangement. According to design B, the 3D LiDAR position remained at the same place as before, while two additional 2D LiDARs with a range of 10 metres were mounted at two rear corners. Together with the original front 2D LiDAR positioned on the vehicle bumper, they were able to provide a 360-degree coverage for close-range obstacles surrounding the vehicle.

Table 4.2: Selection of the camera layout

Criteria (Weightage)	*Baseline (Previous version)*	*Design A*	*Design B*	*Design C*
Front	/	One at the centre	One large FOV and one small FOV	One at the centre
Each Side	/	One facing sideways tiled towards the back	One facing sideways tiled towards the back	One facing sideways, tiled towards the front
Rear	/	/	/	One at the centre
Cost (25%)	0	-3	-4	-4
Coverage (50%)	0	+3	+3.5	+4
Aesthetic (25%)	0	-1	-1	-1
Total Score	0	0.5	0.5	**0.75**

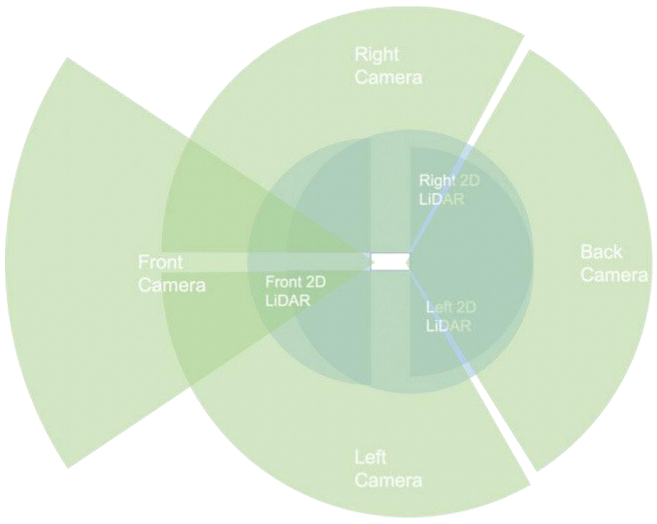

Figure 4.1: The final combined sensor layout

According to the best camera configuration shown in Table 4.2, the supplementary vision system comprised four cameras in total: one medium-range camera mounted at the front, two large FOV cameras mounted at each side of the vehicle, angled slightly towards the front, and one large FOV camera monitoring the back. The selected LiDAR and camera layouts have been combined and shown in Figure 4.1.

5. Perception

The perception module was designed to be responsible for two primary tasks, object detection and localization. To drive safely in a dynamic environment, the obstacle detection algorithms needed to retrieve information about obstacles such as their position, heading, speed, shapes, outer dimensions, even their possible trajectories from raw sensor data [5], [6]. Inside the fusion module, camera and LiDAR data were first processed separately to produce their own obstacles. Then, both results from the visual object detection and the LiDAR object detection algorithms were cross-checked and fused. The vehicle’s localization function was provided by the AMCL algorithm, [31], which was inherited from the previous version of the software stack.

5.1. LiDAR Obstacle Detection and Tracking

The obstacle detection and tracking algorithm used for LiDAR was introduced in the paper “Detection and tracking of 2D geometric obstacles from LRF data” [32]. This algorithm can be divided into two steps, obstacle extraction and obstacle tracking.

First, the raw laser scan data from LiDARs were merged to form a denser point cloud and broader coverage and sent to the obstacle extractor node, where the points were clustered into small groups based on their relative distance from each other. The algorithm then would transform these small point groups into two types of raw obstacles, circle obstacles and line segment obstacles, based on the shape and the size of their outlines. The raw obstacle results had already included several critical information about these obstacles, such as their positions in the 2D map and their radii.

Then, the raw obstacles were passed down to the obstacle tracker, which would filter the raw circle obstacle results with the use of Kalman filters. During this step, the obstacle’s velocity in *x* and *y* directions would be estimated to enable the tracking of obstacles. The final output of the LiDAR obstacle detection algorithm was a list of obstacles with each represented by the following attributes:

$$\{x, y, r_{true}, r_{with\ margin}, v_x, v_y\} \tag{5.1}$$

where x and y represented the obstacle's centre location, r_{true} and $r_{with\ margin}$ represented the true radius of the circle and the radius with a safety margin respectively. The v_x and v_y referred to the obstacle's estimated speed components in the x and y directions with respect to the map.

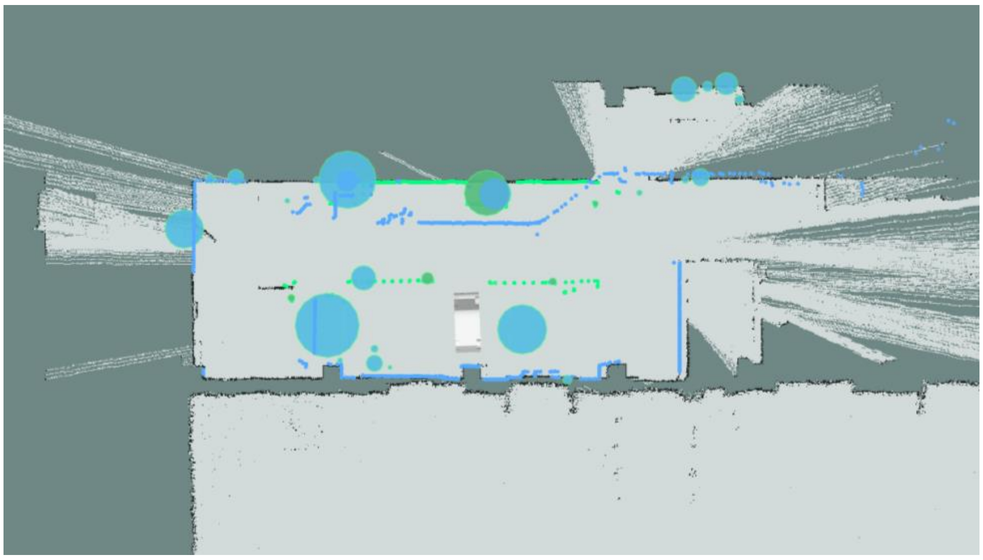
Figure 5.1: LiDAR obstacle detection results

5.2. Visual Object Detection

Although the LiDAR obstacle detection algorithm could provide the position, radius, and velocity of the obstacle, it was unable to identify the class of object that each obstacle belonged to. Differentiating a car from a person, movable versus static object was necessary for a more advanced decision-making process. As a supplementary, it was desirable to use cameras to classify the type of the object and combine the results with the LiDAR detection results.

As a proof-of-concept solution, one Nvidia Jetson TX2 computer and ZED stereo camera were setup on the vehicles to experiment with different visual object detection models. The aim was to select the most suitable object detection algorithm, in terms of the best trade-off between computation power requirement and the detection accuracy [33]. Due to the limited amount of computing power on the vehicle, the more advanced 3D object detection models were excluded from the list of candidates. After doing research on the existing 2D object detection models, six candidates models, SSD-MobileNet-v2 [34], [35], SSD-Inception-v2 [34], [36], RFCN-ResNet101 [37], [38], Faster-RCNN-Inception-ResNet-v2 [38]–[40], Faster-RCNN-ResNet101 [38], [39], and YOLOv3 [41], were selected for their competitive performances. All six models have been tested on a computer with Nvidia P5000 GPU with several pre-recorded real driving videos in Micron, and the testing results were shown in Table 3.5.

Table 5.1: A comparison of 2D visual object detection models

Performance	*Average FPS*	*COCO mAP*	*Qualitative Result*
SSD-MobileNet-v2	26.4	22	Bad
SSD-Inception-v2	24.4	24	Moderate
YOLOv3	15.9	33	Good
RFCN-ResNet101	9.5	30	Good
Faster-RCNN-ResNet101	8.3	32	Good
Faster-RCNN-Inception-ResNet-v2	1.5	37	Good

From the testing results, the winning models with the best trade-off between speed and accuracy were determined to be SSD-Inception-v2, YOLOv3. These two models were then implemented on the buggy for road testing. The models were tested both using a regular webcam, and a ZED stereo camera and the results were satisfactory. When using a ZED stereo camera, to take advantage of its depth-sensing capability, for every detection result, the depth of each object was calculated at the centre of each bounding box on the image. Hence, the output of the visual object detection was a list of objects with each represented by the following parameters:

$$\{class, confidence, x_{img}, y_{img}, w_{img}, h_{img}\} \tag{5.2}$$

where the *class* and *confidence* were the top-1 type of object the algorithm recognised and its corresponding confidence score. The following terms x_{img} and y_{img} were the centre location of the bounding box and w_{img} and h_{img} were the width and the height of the bounding box represented in number of pixels.

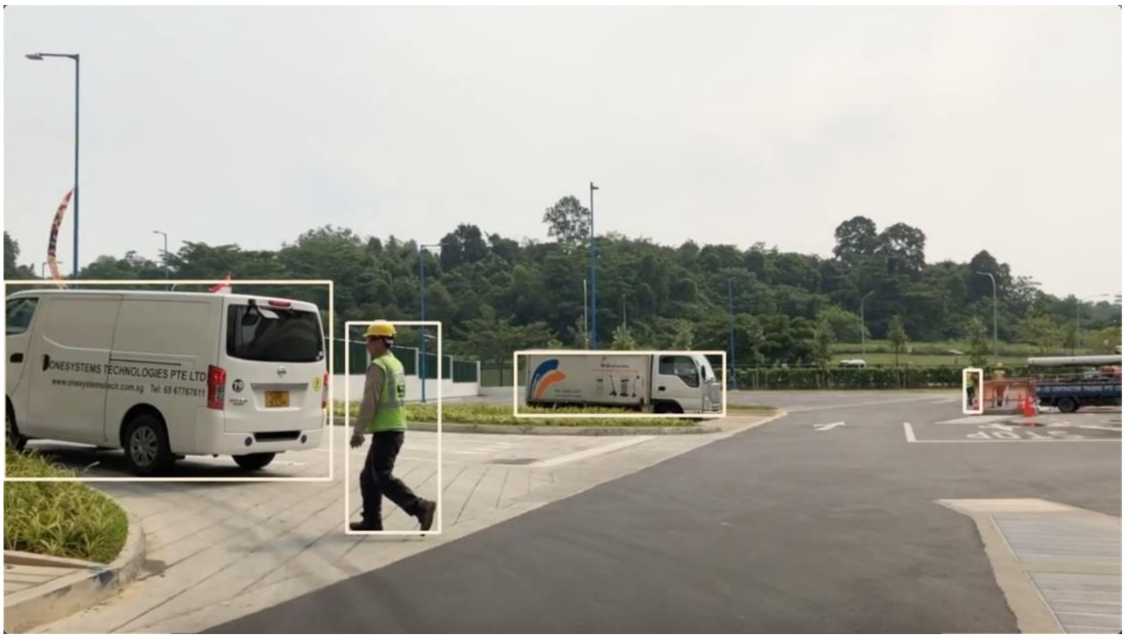

Figure 5.2: Qualitative results of visual object detection using SSD-Inception-v2

5.3. Fusion

After both LiDAR and camera obstacles were found, the last step of the fusion module was to combine this two information and produce the final fused obstacle detection results. This process was mainly a matching process which the obstacles results from both sources were cross-checking and matched based on the degree of overlapping.

As shown in Figure 5.3, based on the position of each bounding box appeared on the image, the angular position (β) and the distance of the object with respect to the camera could be derived from formulas:

$$\beta = \tan^{-1}\left(\frac{\left(\frac{W}{2} - x\right)\tan\left(\frac{\alpha}{2}\right)}{\frac{W}{2}}\right) \tag{5.3}$$

$$r = D \tag{5.4}$$

where α was the camera's FOV, and the camera image was of the size $W \times H$.

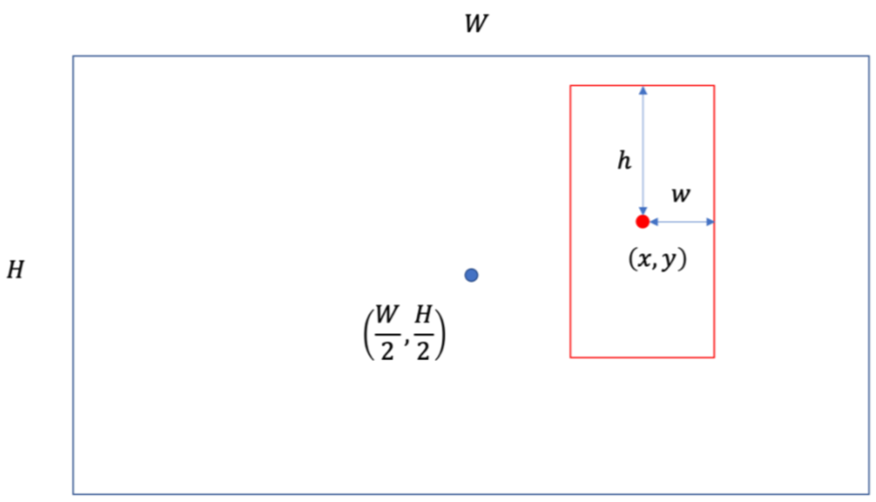

Figure 5.3: Illustration of the object bounding box on an image

Then, the position of each object in the vehicle's local frame was given by:

$$\begin{bmatrix} x_{object,car} \\ y_{object,car} \end{bmatrix} = \begin{bmatrix} x_{cam,car} \\ y_{cam,car} \end{bmatrix} + \begin{bmatrix} r\cos(\beta + \phi) \\ r\sin(\beta + \phi) \end{bmatrix} \tag{5.5}$$

where $[x_{cam,car} \quad y_{cam,car}]^T$ was the camera's position in the vehicle frame and ϕ was the heading of the camera in the vehicle frame. From the above relations, the object' real world position in the global map frame was given by:

$$\begin{bmatrix} x_{object} \\ y_{object} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x_{cam,car} + r\cos(\beta + \phi) \\ y_{cam,car} + r\sin(\beta + \phi) \end{bmatrix} \tag{5.6}$$

With both the camera and LiDAR obstacles transformed to the same map frame, they can be compared based on their position and radius. A simple matching algorithm was used which would assign the type of class to a LiDAR obstacle if it fell into the circle of a camera obstacle. Then, the output of the final fused obstacle information would be like:

$$\{class, confidence, x, y, r_{true}, r_{with\ margin}, v_x, v_y\} \tag{5.2}$$

6. HD Map

6.1. Lane Representation Format

On top of the original laser scan map, a customised lane representation file was added to the HD map to provide detailed lane information in support of driving in a structured lane environment. Previously, the map module only contains the laser scan map produced by the SLAM algorithm, which was only used for localisation purposes. In order to achieve more intelligent autonomous capabilities and let the vehicle plan its path online, more information such as the geometric and topological relations about lane was needed. Thus, a customised lane representation format was designed to support the downstream planning module.

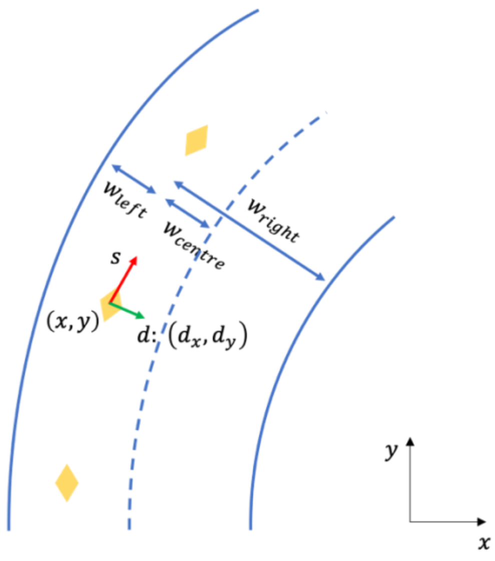

Figure 6.1: Illustration of customised lane representation format

With reference to the existing methods, the customised lane representation format was designed to be concise and easy to maintain. According to the design, each file would contain all the information required for one mission. For a piece of two-way two-lane road, the key features could be divided into four major components, the waypoints on the primary lane (usually the left lane in the obedience of the traffic rules), left boundary, centre dividing line, and the right boundary. Inside each file, all this information was compressed and represented by a list of sparsely separated waypoints that belonged to the primary lane (left lane). Each waypoint was assigned with several attributes:

$$\{x, y, d_x, d_y, s, w_{left}, w_{centre}, w_{right}, v_{max}\} \tag{6.1}$$

where *x* and *y* represented the location of this waypoint on the map, d_x and d_y were the *x* and *y* components of the waypoint's lateral unit vector in frenét frame, and s was the accumulated longitudinal distance from the starting point till this waypoint in frenét frame (Figure 6.1). The following three attributes, w_{left}, w_{centre}, and w_{right}, are the distances from the waypoint to the left, centre, and right boundaries, respectively. The last term, v_{max}, refers to the speed limit at this particular waypoint.

The representation of lanes could be visualised in Rviz as shown in Figure 6.2, where the small red arrows being the left lane waypoints, the blue lines representing the left and right lane boundaries, and the green line representing the centre dividing line.

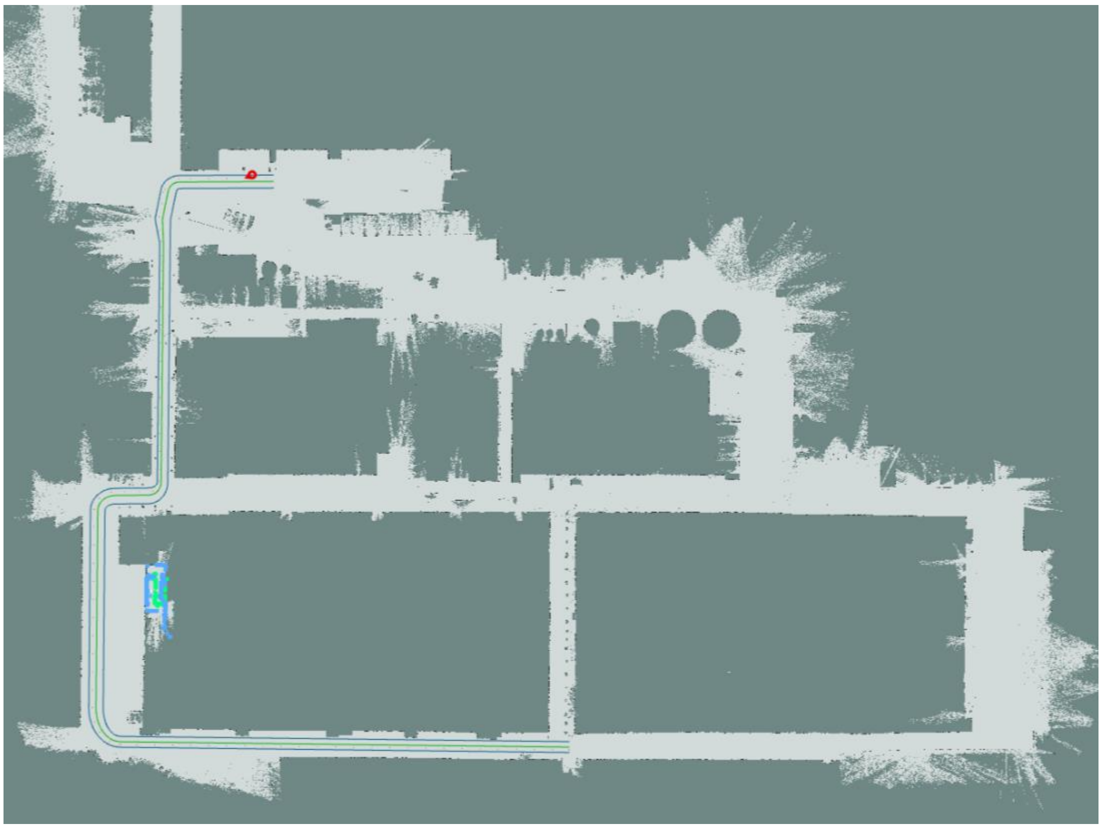

Figure 6.2: Visualization of lanes combined with the laser scan map

7. Planning

The planning module included three major components, the trajectory planner, the collision checker, and the behaviour planner. Unlike the other typical autonomous driving stack, there was no route planner as the mission was about operating on several relatively fixed route. The switching between different starting locations and different destinations could be done by simply switching to a different lane file.

7.1. Trajectory planner

The trajectory planner, often referred to as the local planner, was in charge of generating a trajectory that avoids all obstacles on the road, and the vehicle is able to execute smoothly without difficulties. Since the environment was mostly structured outdoor roads, planning algorithms for robots operating indoor or in an open space that disregard lanes and traffic rules would not be suitable for the job. Another consideration was that those algorithms did not take the vehicle's kinematics into account, and their solutions tend to be not optimal in the long term. After researching the existing autonomous vehicle trajectory planning algorithms, an algorithm called "*frenét optimal planning*" [17] was selected as the backbone of the trajectory planner. Based on this algorithm, with a series of modification and improvements, the vehicle was able to achieve autonomous path planning and obstacle avoidance capabilities.

7.1.1. Implementation of Frenét Optimal Planning Algorithm

The actual implementation of the frenét optimal planning algorithm can be generally divided into five steps, with each step performing a distinctive task. The detailed implementation may not be following the original algorithm exactly with some slight modifications to accommodate this particular application.

7.1.1.1. Interpolating Reference Curve

The first step was to interpolate a reference curve which would be used as the structure of the frenét frame by this algorithm. By using the waypoint information stored in the lane file, the reference curve could be generated easily. The interpolating tool selected was 2D spline, which was a combination of two cubic splines, S_x and S_y, with each establishing the relation between the longitudinal coordinate s of a point in frenét frame and the *x* or *y* coordinate of the same point in global map frame:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} S_x(s) \\ S_y(s) \end{bmatrix} \tag{7.1}$$

The coefficients of a cubic spline could be represented by (using cubic spline s_x as an example):

$$x_i \; = \; a_i \; + \; b_i s_i \; + c_i s_i^2 \; + d_i s_i^3 \tag{7.2}$$

where the a_i, b_i, c_i, and d_i were the corresponding coefficients for *i*-th waypoint. The *i*-th waypoint's *s* coordinate s_i is given by (using cubic spline s_x as an example):

$$s_i \; = \; \sum_{j=2}^{i} \sqrt{\left(x_j - x_{j-1}\right)^2 \; + \; \left(y_j - y_{j-1}\right)^2} \tag{7.3}$$

With *n* being the total number of waypoints to interpolate, and x_i being the *i*-th waypoint's *x* coordinate, the coefficients of each cubic splines was generated individually by solving the following matrices:

$$h = \; \{x_1 - x_0, x_2 - x_1, \cdots, x_{n-1} - x_{n-2}, x_n - x_{n-1}\} \tag{7.4}$$

$$A \; = \begin{bmatrix} 1 & 0 & 0 & & & & \\ h_0 & 2 \times (h_0 + h_1) & h_1 & \cdots & & & \\ 0 & h_1 & 2 \times (h_1 + h_2) & & & & \\ & \vdots & & \ddots & \vdots & & \\ & & & & 2 \times (h_{n-4} + h_{n-3}) & h_{n-3} & 0 \\ & & & \cdots & h_{n-3} & 0 & h_{n-2} \\ & & & & 0 & 0 & 1 \end{bmatrix} \tag{7.5}$$

$$B \; = \begin{bmatrix} 0 \\ 3 \times (a_2 - a_1)/h_1 \; - \; 3 \times (a_1 - a_0)/h_0 \\ 3 \times (a_3 - a_2)/h_1 \; - \; 3 \times (a_2 - a_1)/h_1 \\ \vdots \\ 3 \times (a_{n-1} - a_{n-2})/h_{n-2} \; - \; 3 \times (a_{n-2} - a_{n-3})/h_{n-3} \\ 3 \times (a_n - a_{n-1})/h_{n-1} \; - \; 3 \times (a_{n-1} - a_{n-2})/h_{n-2} \\ 0 \end{bmatrix} \tag{7.6}$$

$$C \; = \; A^{-1} \times B \tag{7.7}$$

After solving the above matrices A, B, and C, the coefficients of the cubic spline at the *i*-th place, a_i, b_i, c_i, d_i are given by:

$$a_i \; = \; x_i \qquad (1 \leq i \leq \; n) \tag{7.8}$$

$$c_i \; = \; C_i \qquad (1 \leq i \leq \; n) \tag{7.9}$$

$$b_i \; = \; \frac{a_{i+1} - a_i}{h_i} \; - \; \frac{h_i(c_{i+1} + 2c_i)}{3} \qquad (1 \leq i \leq \; n) \tag{7.10}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} \qquad (1 \le i \le n) \tag{7.11}$$

By interpolating the 2D spline with a series of densely spaced waypoints, an smooth axis would be formed for the frenét coordinate. The resulted reference spline could be visualised on the map, as shown in Figure 7.1.

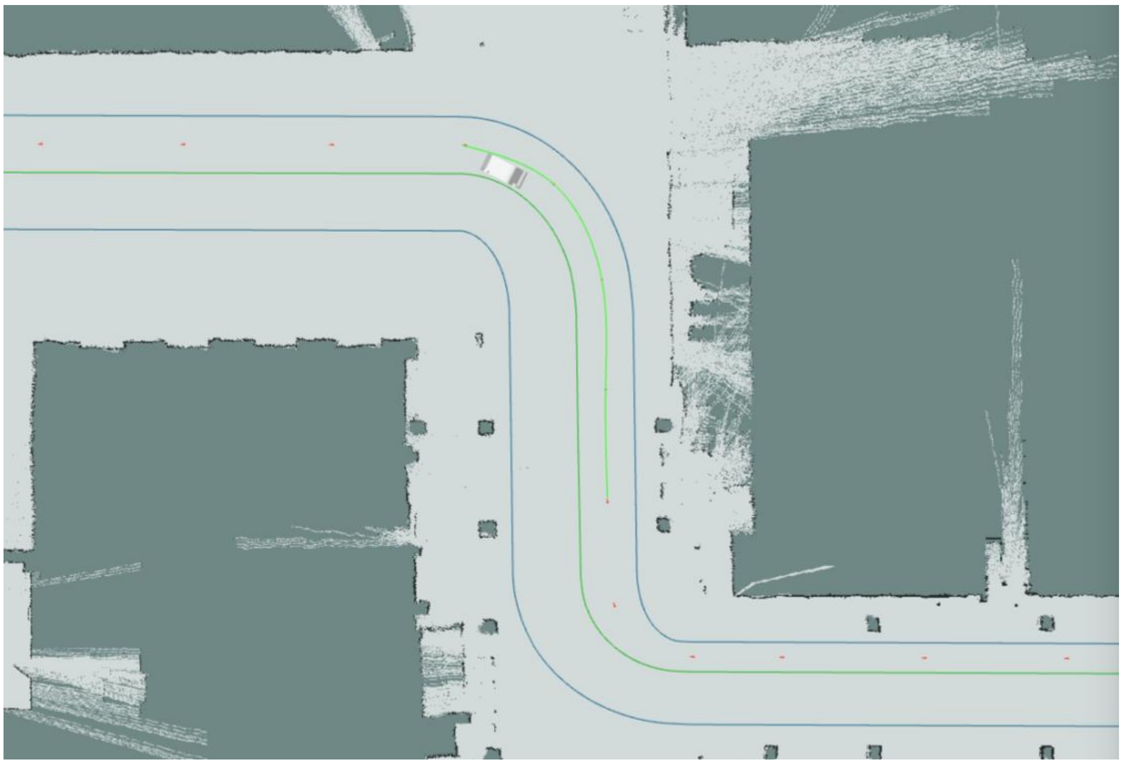

Figure 7.1: Visualization of the resulted reference spline on the map

7.1.1.2. Sampling Final States

After established a frenét frame based on a reference curve that was interpolated using waypoints, the vehicle's current pose could be represented in the new frenét frame. The vehicle's state usually was retrieved from the localization module in the form of a state vector with respect to the global map frame:

$$\begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix} \tag{7.12}$$

while the vehicle state in frenét frame could be represented as:

$$\begin{bmatrix} s \\ \dot{s} \\ d \\ \dot{d} \end{bmatrix} \tag{7.13}$$

The transformation of the vehicle's current pose from the global frame to the frenét frame could be done by first finding the closest next waypoint (x_{next}, y_{next}), and the closest previous waypoint (x_{prev}, y_{prev}), on the reference spline. As shown in Figure 7.2, the following relations could be established:

$$\vec{n}_s = \begin{bmatrix} x_{next} - x_{prev} \\ y_{next} - y_{prev} \end{bmatrix} \tag{7.14}$$

$$\vec{m} = \begin{bmatrix} x - x_{prev} \\ y - y_{prev} \end{bmatrix} \tag{7.15}$$

$$proj_{\vec{n}_s}\vec{m} = \frac{\vec{n}_s \cdot \vec{m}}{|\vec{n}_s|^2}\,\vec{n}_s \tag{7.16}$$

$$\vec{n}_d = \begin{bmatrix} d_{x,prev} \\ d_{y,prev} \end{bmatrix} \tag{7.17}$$

$$\Delta\theta = \theta - tan^{-1}\left(\frac{d_{y,prev}}{d_{x,prev}}\right) + \frac{\pi}{2} \tag{7.18}$$

$$proj_{\vec{n}_d}\vec{m} = \frac{\vec{n}_d \cdot \vec{m}}{|\vec{n}_d|^2}\,\vec{n}_d \tag{7.19}$$

$$k = \frac{proj_{\vec{n}_d}\vec{m}}{\left|proj_{\vec{n}_d}\vec{m}\right|} \tag{7.20}$$

where $\vec{n}_s$ was the vector from the previous waypoint to the next waypoint (frenét *s*), and $\vec{n}_d$ was the unit vector pointing in the lateral (frenét *d*) direction. Vector $\vec{m}$ is the vector from the previous waypoint pointing to the vehicle's current pose. The projection of vector $\vec{m}$ on vector $\vec{n}_s$ and vector $\vec{n}_d$ were denoted as $proj_{\vec{n}_s}\vec{m}$, and $proj_{\vec{n}_d}\vec{m}$ respectively.

Thus, the vehicle's initial state could be solved by evaluating:

$$\begin{bmatrix} s \\ \dot{s} \\ d \\ \dot{d} \end{bmatrix} = \begin{bmatrix} \sum_{i=2}^{prev} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \\ v\,cos(\Delta\theta) \\ k\left|\vec{m} - proj_{\vec{n}_s}\vec{m}\right| \\ v\,sin(\Delta\theta) \end{bmatrix} \tag{7.21}$$

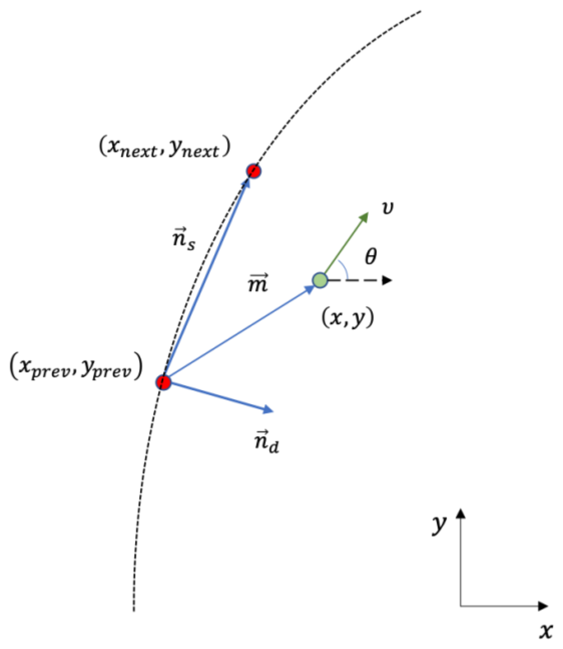

Figure 7.2: Illustration of transformation from global map frame to frenét frame

After the initial state had been found using the above formula, the algorithm would systematically sample a large number of possible final states in the same frenét frame. The sampling took place in a space with three dimensions, the longitudinal (*s*), lateral (*d*), and the time (*T*). The initial state and final state could be written as:

$$initial\ state\ = \begin{bmatrix} s_i \\ \dot{s}_i \\ \ddot{s}_i \\ d_i \\ \dot{d}_i \\ \ddot{d}_i \\ T_i \end{bmatrix} \qquad final\ state\ = \begin{bmatrix} s_f \\ \dot{s}_f \\ \ddot{s}_f \\ d_f \\ \dot{d}_f \\ \ddot{d}_f \\ T_f \end{bmatrix} \tag{7.22}$$

The final states were sampled uniformly in each degree of freedom. However, they have to be subjected to certain geometric constraints such as within the road boundary and a limited time extent:

$$v_{ref}T_{min}\ \leq\ s_f\ \leq\ v_{ref}T_{max} \tag{7.23}$$

$$w_{right}\ \leq\ s_d\ \leq\ w_{left} \tag{7.24}$$

where the v_{ref} was also an parameter that varying within the recommended speed range:

$$v_{min}\ \leq\ v_{ref}\ \leq\ v_{max} \tag{7.25}$$

7.1.1.3. Generate Candidate Trajectories

The next step of the algorithm was to connect each candidate final state to the initial state to form a trajectory. The trajectories were generated in the frenét frame and each was decoupled into two polynomials, with one longitudinal quartic polynomial $s(t)$ and one lateral quintic polynomial $d(t)$.

To solve the lateral quintic polynomial $d(t)$, the polynomial itself, together with its first derivative, and its second derivative could be written in the form:

$$d(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 + \alpha_4 t^4 + \alpha_5 t^5 \quad (7.26)$$

$$\dot{d}(t) = \alpha_1 + 2\alpha_2 t + 3\alpha_3 t^2 + 4\alpha_4 t^3 + 5\alpha_5 t^4 \quad (7.27)$$

$$\ddot{d}(t) = 2\alpha_2 + 6\alpha_3 t + 12\alpha_4 t^2 + 20\alpha_5 t^3 \quad (7.28)$$

By choosing $t_i = 0$:

$$\begin{bmatrix} s_i \\ \dot{s}_i \\ \ddot{s}_i \end{bmatrix} = \begin{bmatrix} s(0) \\ \dot{s}(0) \\ \ddot{s}(0) \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ 2\alpha_2 \end{bmatrix} \quad (7.29)$$

$$d(t) = s_i + \dot{s}_i t + \frac{\ddot{s}_i}{2} t^2 + \alpha_3 t^3 + \alpha_4 t^4 + \alpha_5 t^5 \quad (7.30)$$

$$\dot{d}(t) = \dot{s}_i + \ddot{s}_i t + 3\alpha_3 t^2 + 4\alpha_4 t^3 + 5\alpha_5 t^4 \quad (7.31)$$

$$\ddot{d}(t) = \ddot{s}_i + 6\alpha_3 t + 12\alpha_4 t^2 + 20\alpha_5 t^3 \quad (7.32)$$

Then, substitute t with t_f:

$$s_f = d(t_f) = \alpha_3 {t_f}^3 + \alpha_4 {t_f}^4 + \alpha_5 {t_f}^5 + s_i + \dot{s}_i t_f + \frac{\ddot{s}_i}{2} {t_f}^2 \quad (7.33)$$

$$\dot{s}_f = \dot{d}(t_f) = 3\alpha_3 {t_f}^2 + 4\alpha_4 {t_f}^3 + 5\alpha_5 {t_f}^4 + \dot{s}_i + \ddot{s}_i t_f \quad (7.34)$$

$$\ddot{s}_f = \ddot{d}(t_f) = 6\alpha_3 t_f + 12\alpha_4 {t_f}^2 + 20\alpha_5 {t_f}^3 + \ddot{s}_i \quad (7.35)$$

The above relations could be written in the following matrix form:

$$\begin{bmatrix} {t_f}^3 & {t_f}^4 & {t_f}^5 \\ 3{t_f}^2 & 4{t_f}^3 & 5{t_f}^4 \\ 6t_f & 12{t_f}^2 & 20{t_f}^3 \end{bmatrix} \times \begin{bmatrix} \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} s_f - (s_i + \dot{s}_i t_f + \frac{\ddot{s}_i}{2} {t_f}^2) \\ \dot{s}_f - (\dot{s}_i + \ddot{s}_i t_f) \\ \ddot{s}_f - (\ddot{s}_i) \end{bmatrix}$$

(7.36)

From equation 7.29 and equation 7.36, all the six coefficients of the quintic polynomial could be derived. Similarly, the coefficients of the longitudinal quartic polynomial $s(t)$ can be solved using the same method. After both polynomials had been solved, they were combined to form a three-dimensional trajectory. Once a large group of candidate trajectories had been sampled, their distribution in the 2D space would be similar to the pattern shown in Figure 7.3.

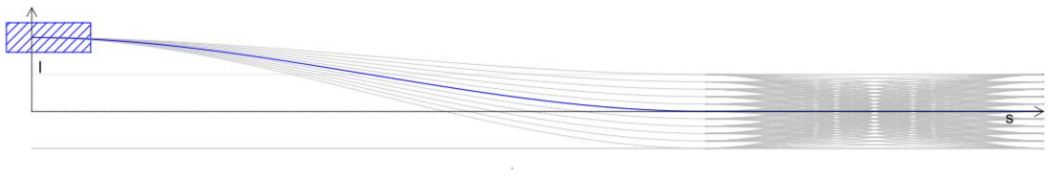

Figure 7.3: Trajectory sampling results, adapted from [23]

7.1.1.4. Assigning Costs

In order to quantitively evaluate the performance of each trajectory, several cost functions were implemented to compute the costs for each trajectory.

The first cost function was the total squared jerk, which is closely related to the human's perception of comfort. The jerk was defined as the derivative of the acceleration, making it the third derivative of s or d. The total square jerk was computed for both the longitudinal and the lateral polynomials. For the longitudinal trajectory $s(t)$, the total squared jerk was given by:

$$s(t) \in [0, t_f] \tag{7.37}$$

$$J_s = \int_0^{t_f} \dddot{s}(t)^2 \, dt \tag{7.38}$$

Similarly, the total squared jerk for the lateral trajectory $d(t)$ was given by:

$$d(t) \in [0, t_f] \tag{7.39}$$

$$J_d = \int_0^{t_f} \dddot{d}(t)^2 \, dt \tag{7.40}$$

In the longitudinal direction, to encourage driving at the desired speed, another cost function was designed to penalized speed differences at the final state:

$$\Delta v^2 = \left(\dot{s}_f - v_{ref}\right)^2 \tag{7.41}$$

Meanwhile, in the lateral direction, the offset from the centre of the lane would be penalized to encourage driving on the centre of the lane. This cost function was designed based on the difference between the lateral position of the final state and the desired centreline:

$$\Delta d^2 = \left(d_f - d_{ref}\right)^2 \tag{7.42}$$

The output value of each cost function was unified to $[0, 1]$ to ensure comparable results across different cost terms, and the emphases on particular cost terms were controlled by tuning the weights assigned for each cost term. The total cost for each trajectory equalled to the weighted sum of all the individual cost functions.

7.1.1.5. Checking for Hard Constraints

The last step of the algorithm was to convert the candidate trajectories from the frenét frame back to the global map frame and check for hard constraints. Any collisions detected or any violation of the predefined physical constraints will cause the corresponding trajectory to become invalid.

The collision model between the vehicle and obstacles was simplified using the method described in [42]. The rectangular vehicle footprint was modelled as three circles along the centreline of the vehicle in the longitudinal direction (as shown in Figure 7.4). The distances between the centre of each vehicle circle and the obstacle circles were compared with the sum of the two circles' radii to determine if there would be any collision:

$$margin = D - r_{obstacle} - r_{vehicle} \tag{7.43}$$

$$collision = \begin{cases} true, & margin < safety\,margin \\ false, & margin \geq safety\,margin \end{cases} \tag{7.44}$$

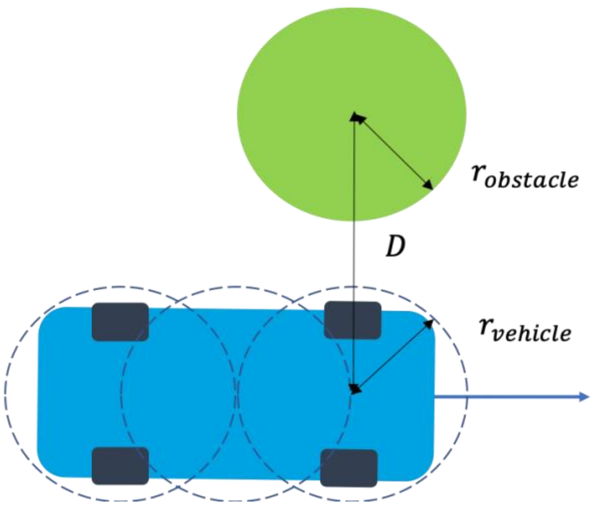

Figure 7.4: Collision model between the vehicle and obstacles

Besides collision checking, feasibility check is another necessary step to validate the candidature of the trajectory in advance. During this step, the vehicle's physical constraints and the manually defined hard constraints on speed, acceleration, and turning curvature would be check to ensure safety. Each trajectory must be able to satisfy the following criteria:

$$v_{min} \leq \dot{s}(t) \leq v_{max} \tag{7.45}$$

$$a_{min} \leq \ddot{s}(t) \leq a_{max} \tag{7.46}$$

$$\left|\ddot{d}(t)\right| \leq a_d \tag{7.47}$$

$$\kappa(t) \leq \kappa_{max} \tag{7.48}$$

where $\kappa(t)$ was the curvature of the trajectory at time step t, given by:

$$\kappa(t) = \frac{\Delta\varphi(t)}{\Delta s(t)} \tag{7.49}$$

Eventually, from the remaining candidate trajectories, the one with the minimum cost would be selected as the final trajectory for the vehicle to execute.

7.1.2. Practical Considerations and Improvements

Although the frenét optimal planning was a powerful planning algorithm, and its performance had been mostly satisfactory, several practical issues still occurred during the development and testing the vehicle. In order to advance towards a more robust and more intelligent obstacle avoidance capability, several improvements were made to the algorithm.

7.1.2.1. Trajectory Concatenation

The first improvement made to the algorithm was a post-processing step after the best trajectory had been successfully selected, called the trajectory concatenation. Initially, the best-selected trajectory was used directly by the control module as the current trajectory. During the testing, it was found that the vehicle tended to be unable to correct a small offset from the centre of the road when driving straight. The root cause was found to be that the polynomial trajectory generated was too smooth in nature, making the first one-third of the trajectory with little lateral displacement, which resulted in little motivation for the vehicle to correct that lateral offset.

Another adverse effect it brought was that the best trajectory might be unstable in certain scenarios. The planner might become indecisive sometimes, and arrive at a new best trajectory which varied drastically from its previous one, which could potentially pose threats and compromise comfort.

The idea was first inspired by the MPC algorithm, which would solve for a sequence of optimal control actions in the future and only perform the first action and discard the rests at each step. The solution was to plan an optimal trajectory in a relatively long time horizon (such as 10 seconds) and only the first few waypoints of this trajectory (the equivalent of the vehicle's motion in about 1 second) would be added into the current trajectory. The waypoints in the current trajectory list would be the actual trajectory that the vehicle was executing. Once a waypoint in this list had been executed by the vehicle, it will be deleted from the list. Thus the list was automatically shorting as the vehicle running. This list had both an upper limit (N_{max}) and a lower limit (N_{min}) for the number of waypoints (N) it contains. When the current trajectory contained enough waypoints ($N > N_{min}$) for the vehicle to operate, the trajectory planner would switch to use the last waypoint in the current trajectory list as the initial state to generate a trajectory for the next step. While the trajectory planner kept planning for the next step's motion, its latest results will be stored in another list, called the next trajectory. This list would be continuously updated by the planner until the current trajectory had not enough waypoints left in the list ($N \leq N_{min}$). The next trajectory list will pass down the first few waypoints to the current trajectory list to achieve a smooth and seamless transition. Then both the current trajectory list and the next trajectory list would repeat the same cycle again and again.

In case of special scenarios, such as emergency stops or when the vehicle deviated too much from the current trajectory, the trajectory planner will trigger a "regenerate" function which would erase both the current and the next trajectory lists and re-plan the trajectory entirely.

An illustration of the added trajectory concatenation was shown in Figure 7.5, where the blue curve represented the current trajectory, and the red one represented the next trajectory. By adding the trajectory concatenation, the planner was able to overcome the issues mentioned above,s and a safer and smoother ride was achieved.

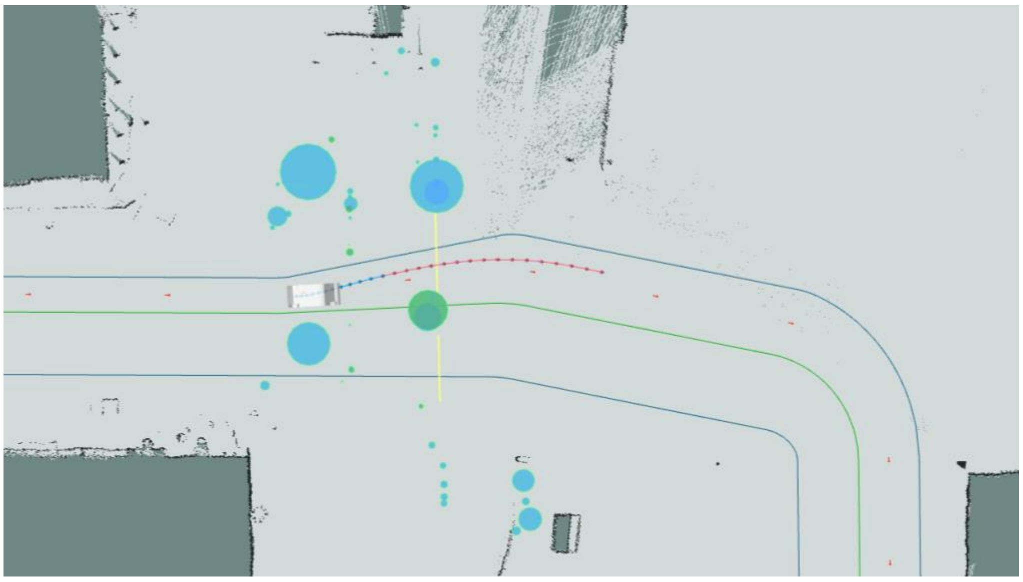
Figure 7.5: Illustration of the concatenated trajectory

7.1.2.2. Additional Cost Functions & Soft Constraints

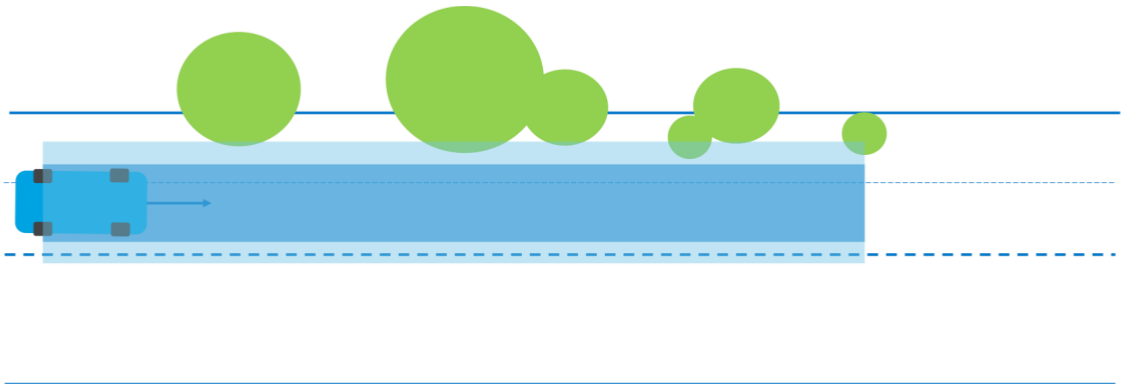
Figure 7.6: Illustration of the lateral offset and hard/soft safety margins

Due to the imperfection of the LiDAR obstacle detection results, the sizes of obstacles might shift drastically, especially when viewing an obstacle from different angles. When driving on the road, the trajectory planner might become sensitive to disturbances, and response with a sharp turn away from a false obstacle on the roadside. To improve the passengers' comfort, two measures were taken to solve this issue. The first solution was to add a small offset from the centre of the left lane, to encourage driving that was inclined towards the inner side of the road. Hence, the origin of the lateral axis had been shifted by a small amount during the sampling of the final states. Meanwhile, comparing to equation 7.42, the new lateral offset cost function was given by:

$$\Delta d^2 = \left(d_f - \left(d_{ref} + d_{offset}\right)\right)^2 \tag{7.50}$$

By making the above modifications, the vehicle became more preferred to drive on the inner side of the lane and stay further away from the disturbances.

The second measure was to add a soft constraint in the collision checking step of the algorithm, in addition to the original hard collision checking criteria. The new method of collision checking comprised two criteria, the hard safety margin and the soft safety margin. The new hard safety margin was functioning in the same way as the previous collision checking logic, except for the hard safety margin became smaller than before, as shown in Figure 7.6. Any obstacle that existed inside this hard safety margin would still be treated as collision, and the corresponding trajectory would be eliminated. However, outside the hard safety margin, there was a new layer of soft safety margin which would add a cost term to this trajectory if any obstacles were within this zone. The distance-to-obstacle cost function was designed as:

$$Cost_{obstacle} = \left(\frac{margin_{soft} - (D - r_{obstacle} - r_{vehicle} - margin_{hard})}{margin_{soft}}\right)^2 \tag{7.51}$$

7.1.2.3. Sampling ROI

In order to receive commands from the behaviour planner and generate the correct trajectory according to the decisions made by the behaviour planner, an ROI sampling feature was implemented. Since the behaviour planner expected the trajectory planner to be able to generate trajectories according to its decisions, it was necessary for the trajectory planner to restrict the sampling region within the desired lane instead of sampling over the entire space. Usually, the driving behaviour would imply a target ROI, which often could be taken as the target lane. As a result, based on the given target lane, the trajectory could guide the vehicle to switch between lanes effectively. Moreover, three predefined ROI: left lane, right lane, and both lanes, were implemented in the trajectory planner for the convenience of behaviour planner.

7.2. Behaviour planner

The primary function of the behaviour planner was to decide on the vehicle's next driving behaviour by analysing data from the perception module and the map module. In more details, the behaviour planner was designed in a finite state machine structure which allowed the vehicle to switch between desired states (driving behaviours).

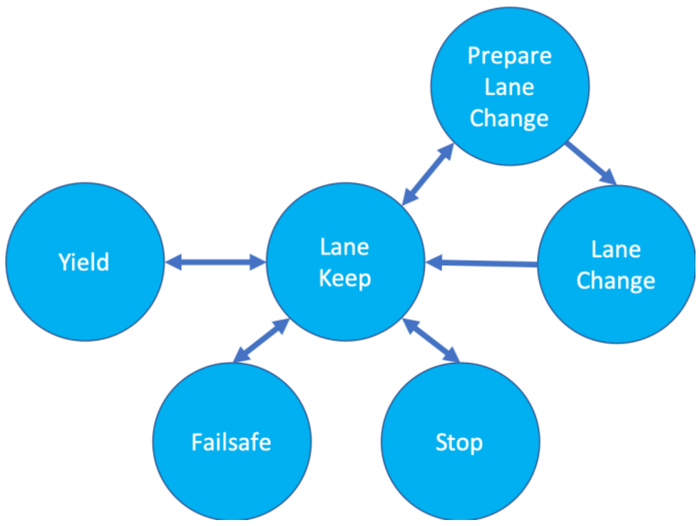

Figure 7.7: Vehicle states in the behaviour planner

As shown in Figure 7.7, the behaviour planner included six major vehicle states, namely: *lane keep*, *prepare lane change*, *lane change*, *yield*, *stop*, and *failsafe*. The arrows in the figure showed the allowed switching actions between states. Each state was allowed to switch to itself at each step as well.

The default state for the vehicle was the lane keep state which would request the trajectory planner to sample possible trajectories only on the current lane. When obstacles were encountered, based on the situation, the vehicle would decide if it was necessary to enter the stop state or yield state. If it were an emergency, the behaviour planner would choose to *stop*, and if it were a moving obstacle that blocked the vehicle's path, the behaviour planner would *yield* and wait.

In situations that are not so urgent, for example, the current lane was not entirely blocked, i.e., the trajectory planner was still able to find an available trajectory, the vehicle would remain in the *lane keep* state and follow the current lane. When the current lane was not available anymore, which meant that the trajectory planner was unable to find a safe trajectory in the current lane, the behaviour planner would enter the *prepare lane change* state. In *prepare lane* *change* state, the vehicle would consider the feasibility and safety of performing a lane change. The trajectory planner would be asked to try sampling trajectories on the new target lane, and the behaviour planner would start to analysis if there were any potential risks, especially the moving obstacles around the vehicle and in the target lane that might cause collisions. If it were determined to be unsafe, the behaviour planner would abort the lane change return to the *lane keep* state. Only when all the safety checks were passed, the vehicle would enter the *lane change* state and remain in this state until the lane change finished or the *stop*, *yield*, or *failsafe* state was triggered.

7.3. Collision Checker

In addition to the behaviour planner and the trajectory planner that are commonly used in typical autonomous driving architectures, a collision checker was developed as the final safeguard of the vehicle. The old method of using distance-to-obstacle as the gauge to decide whether the vehicle should slow down or stop was not flexible enough in many scenarios. After equipped with the obstacle avoidance capability, the vehicle became more intelligent in dealing with situations that were unable to solve before, and the limiting factor became the old collision checking function. Sometimes, in scenarios like the one illustrated in Figure 7.8, it would be impossible for the vehicle to continue with the planned trajectory as the old collision checker sensed obstacles within the stopping zone.

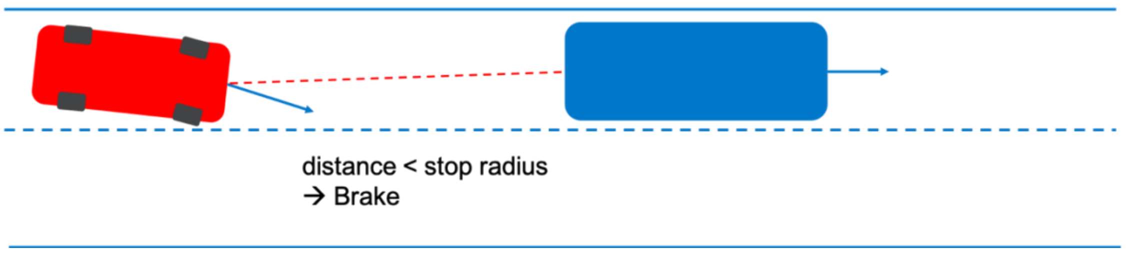

Figure 7.8: The limitation of using distance-to-obstacle in collision checker

To help the vehicle from getting stuck in such situations, a new method using time-to-collision and dynamic virtual bumper was proposed. Comparing to the old method, the new method utilised more information of the buggy to form a more accurate description of the vehicle's motion in the next a few seconds of time. Instead of using a static safety checking zone that was fixed in front of the vehicle, the new collision checking zone was able to deform based on the vehicle's kinematics and control outputs.

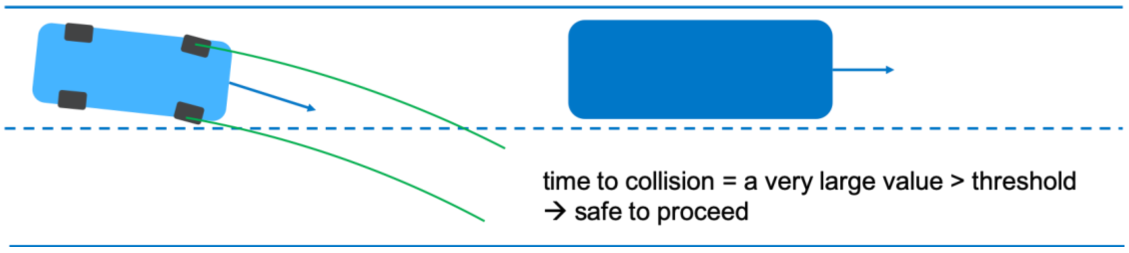

Figure 7.9: Illustration of using time-to-collision and dynamic virtual bumper

The dynamic virtual bumper was defined by a series of predicted vehicle position in the future. The initial state of the vehicle could be represented as:

$$\begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \\ v_0 \\ \delta_0 \end{bmatrix} \tag{7.52}$$

Assuming constant velocity v, the predicted vehicle position at step i was given by:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + \begin{bmatrix} v\ \cos\theta_i\ dt \\ v\ \sin\theta_i\ dt \end{bmatrix} \tag{7.53}$$

where T is the prediction horizon and dt is the time increment at each step.

Its length l and curvature κ purely depended on both the vehicle's velocity v and steering angle δ. The turn rate $\dot{\theta}$ of the vehicle could be determined based on the kinematic vehicle model. The above relations could be written as:

$$\dot{\theta}(t) = \frac{v(t)}{L_f}\,\delta(t) \tag{7.54}$$

where L_f was the distance front the vehicle's centre of gravity (CG) to the front axle. To better simulate the recovery of steering angle, a decay factor α was added into the equation, which would linearly decay from 1 to 0 within the prediction horizon T:

$$\delta(t) = \alpha(t)\,\delta_0 \tag{7.55}$$

$$\alpha(t) = \frac{T-t}{T} \tag{7.56}$$

From the above relations, the heading of the vehicle θ_i at step i could be derived as:

$$\theta_i = \theta_0 + \int_0^{i\,dt} \frac{T-t}{T} \frac{v(t)}{L_f} \delta_0 \tag{7.57}$$

Hence, the predicted vehicle position at i step :

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + \begin{bmatrix} v \cos\left(\theta_0 + \int_0^{i\,dt} \frac{T-t}{T} \frac{v(t)}{L_f} \delta_0\right) dt \\ v \sin\left(\theta_0 + \int_0^{i\,dt} \frac{T-t}{T} \frac{v(t)}{L_f} \delta_0\right) dt \end{bmatrix} \tag{7.58}$$

After the vehicle’s predicted positions at each time step have been generated, they were used to check for collisions with other obstacles. If any collision were detected within thresholds t_{slow} and t_{stop}, the vehicle would consider to slow down or stop, and the behaviour planner would be informed to make decisions accordingly.

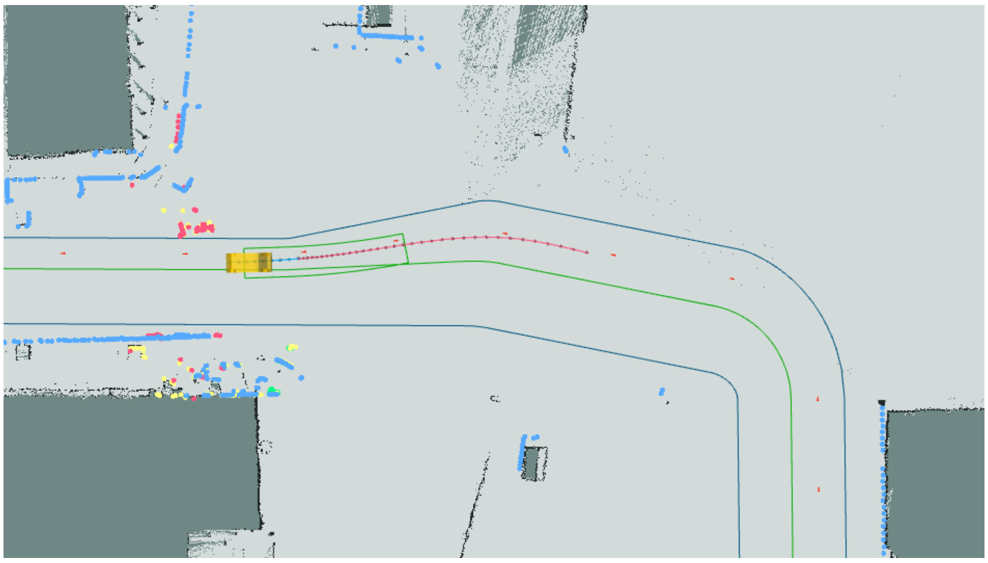

Figure 7.10: Visualization of the dynamic virtual bumper in action

The new method of applying time-to-collision and dynamic virtual bumper was implemented on the vehicle and tested. The green polygon shape shown in Figure 7.10 represented the dynamic virtual bumper. The performance of the collision checker was significantly improved compared to its predecessor, and the vehicle behaved more natural and more similar to a human driver.

8. Control

8.1. PID and Stanley

In the previous version of the autonomous driving software, the control module comprised two parts, a PID controller controlling the longitudinal motion of the vehicle and a Stanley controller controlling the lateral motion of the vehicle. The combination was able to follow a fixed path excellently during the first phase of the project. However, during the second phase, after the trajectory had been implemented, it was realised that this combination was no longer adequate. Compared to the handcrafted fix path in the first phase, the new polynomial trajectory generated by the planner tended to be more challenging to follow. Especially when turning and dodging obstacles, the vehicle tended to overshoot and deviate from the planned trajectory.

In order to solve the overshooting issue, several improvements were made to the control module. First, the steering angle output was lowered by adding an overall gain term in the Stanley formula:

$$\delta(t) = k_{overall}\left(\psi(t) + tan^{-1}\left(\frac{k\, x(t)}{v(t)}\right)\right) \tag{7.59}$$

Second, the speed limits v_{max}, at all turning points, slopes, and narrow passages were lowered to avoid aggressive motion. Additionally, when the steering angle was greater than a certain value, the speed limit would drop as well. After these minor improvements, the performance of the control module generally improved. It was able to complete the whole route with little overshooting and oscillation.

However, the combination of using PID and Stanley was still not satisfactory enough, and the root cause of the issues was not solved. It was determined that a better solution was to find an alternative solution.

8.2. MPC

8.2.1. MPC Controller Design

In order to improve the path tracking performance by switching to a new control algorithm, a linear MPC controller (as shown in Figure 8.1) was implemented on the vehicle.

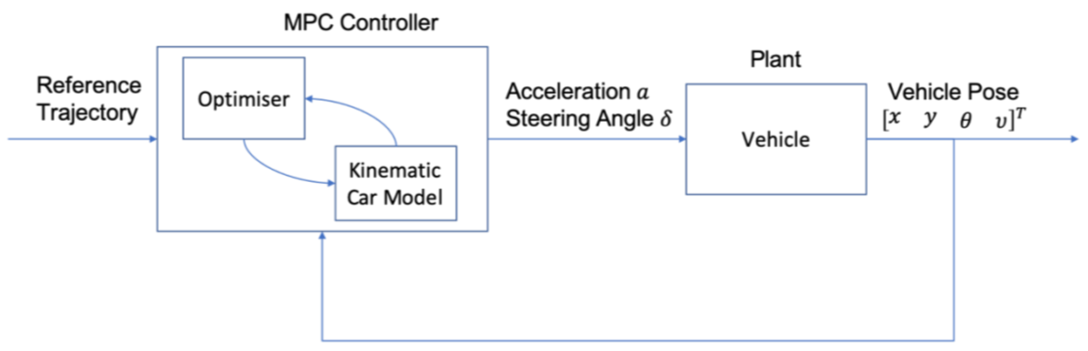

Figure 8.1: MPC controller design

The MPC controller took in two inputs, the reference trajectory from the trajectory planner, and the vehicle pose from the localisation module. Inside the MPC controller, an optimiser was iterating the optimisation process of minimising the designed cost functions with provided constraints. A kinematic vehicle model was used to help the optimiser predict the expected reactions of the vehicle. The optimiser was implemented using the existing functions in the optimisation software libraries, CppAD and Ipopt. The output of the MPC controller was the solved best sequence of actuations for a future time period T (prediction horizon). However, only the control actions of the next time step would be sent to the steering motor, the accelerator and the brake motor on the vehicle while the reset would be discarded and the controller would start another round of computation.

The target functions in the optimiser were designed based on three types of costs, including reference state error costs which were used to minimise the vehicle's error from the desired trajectory, actuator usage costs which aimed to reduce the uses of actuators, and the actuator change costs which would penalise huge change between actuations. The reference state error costs included the squared sum of the cross-track error (cte), the heading error (ψ), and the speed difference (Δv):

$$cost_{cte} = w_{cte} \sum cte^2 \tag{8.1}$$

$$cost_{\psi} = w_{\psi} \sum \psi^2 \tag{8.2}$$

$$cost_{\Delta v} = w_{\Delta v} \sum \Delta v^2 \tag{8.3}$$

The actuator usage costs were simply the squared sum of the steering angle (δ) and acceleration (a):

$$cost_{\delta} \ = \ w_{\delta} \sum \delta^2 \tag{8.4}$$

$$cost_{a} \ = \ w_{a} \sum a^2 \tag{8.5}$$

Similarly, the actuator change costs were given by:

$$cost_{\Delta\delta} \ = \ w_{\Delta\delta} \sum \Delta\delta^2 \tag{8.6}$$

$$cost_{\Delta a} \ = \ w_{\Delta a} \sum \Delta a^2 \tag{8.7}$$

The final total cost for each sequence of control outputs was the weighted sum of all the costs and in the above equations, w_{cte}, w_{ψ}, $w_{\Delta v}$, w_{δ}, w_{a}, $w_{\Delta\delta}$, and $w_{\Delta a}$ were the weights for each cost term.

The vehicle physical constraints were also modelled and taken into account in the MPC as lower or upper bounds for certain variables. The design started with two simple constraints: the maximum and minimal acceleration (deceleration) and the steering angle. The relation could be written as:

$$a_{min} \ \leq \ a \ \leq \ a_{max} \tag{8.8}$$

$$\delta_{min} \ \leq \ \delta \ \leq \ \delta_{max} \tag{8.9}$$

8.2.2. Test & Improvements

After the MPC had been implemented, it was first tested with simulated vehicle data to conduct a preliminary tuning of the weights and prove its functionality. A graph of the simulation outputs, as shown in Figure 8.2, showed the changes in cross-track error, steering angle, and speed over time after tuning of weights.

The vertical axis was each in metres, radians, and metres per second, while the horizontal axis was in time steps (0.2 seconds each). The trends showed that the error converged to zero quickly and smoothly, without overshooting or oscillation.

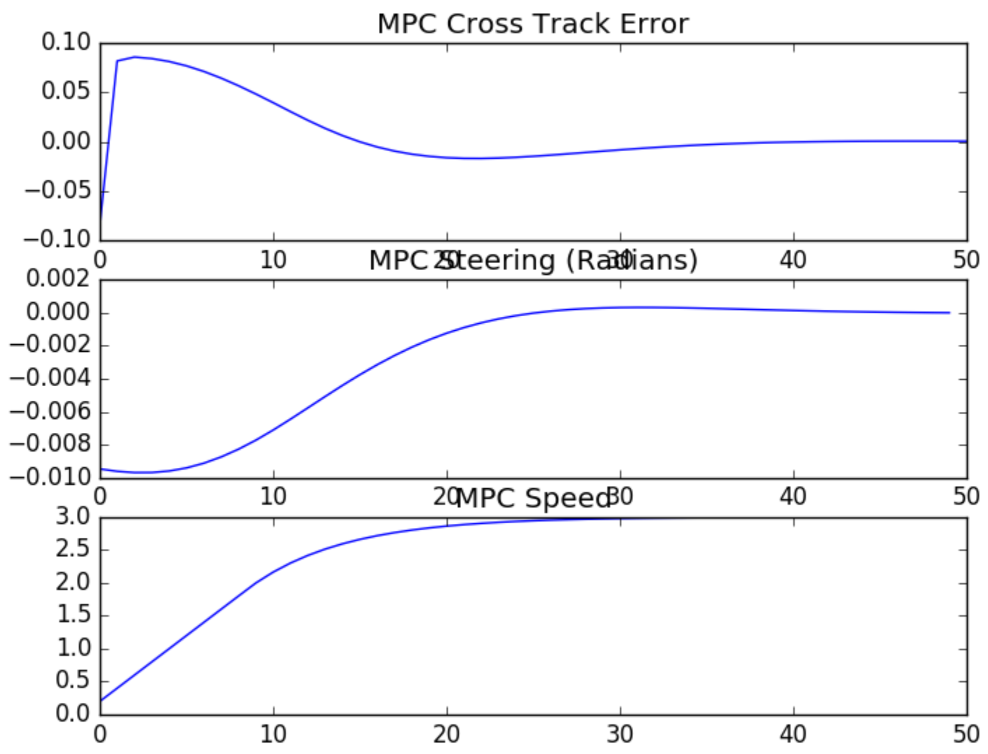

Figure 8.2: MPC controller tested in simulation

Then, the MPC controller was further testing on the vehicle. However, the results were not satisfactory. The vehicle tended to oscillate, and its path on a straight road was like an “S” shape. The vehicle was not able to follow the reference trajectory most of the time. Due to time constraints, the problem was not solved yet, and the vehicle was still using the improved PID controller and the Stanley controller as the control module. However, through observations, several possible causes of this issue were found.

First, due to the computing power limitation, this version of the MPC controller was only able to perform at 5Hz, while the improved PID controller and the Stanley controller were operating at above 10Hz. This difference in control rate may introduce a delay to the MPC controller. Together with the delay in sensor feedback, this might have caused the MPC controller to respond to changes slower and thus overshoot. The solution to this problem was to optimise the controller’s computational speed. One method was to add a control horizon in the current design of the optimiser, which would possibly reduce the computational load of the MPC controller significantly. In the current design, the optimiser would try out all the combinations of actuations in the entire prediction horizon. A control horizon is usually a time period shorter than the prediction horizon, and the optimiser could assume that no control actions would be made after this point of time. Thus, the dimension of the total possible control sequences would be significantly reduced.

The second issue was that there were no cost functions related to the smoothness of the ride or human comforts. The vehicle would not slow down during turning, and its behaviour was not as natural as human drivers. This problem could be solved easily by adding new cost functions that are associated with turning curvatures, longitudinal jerks and lateral jerks, which would penalise fast and tight turns.

9. Graphical User Interface

9.1. New Design

The first new upgrade is the new appearance design. The new GUI adopted a modern design, which makes it look much more concise and elegant than its previous version, as shown in Figure 9.1. All the visualizations of information were adjusted to provide a more intuitive user experience.

An info display banner was designed in the GUI to provide hints to users during the ride to help them operate the vehicle. When the vehicle is stationary and ready to move out, it will ask the users to select the destination and press the "START" button. When the user presses the "START" button, the GUI will perform a safety check. This new feature will ensure the vehicle will not be able to start while moving or without a destination. Once all the criteria have been met, the banner will display a 3-second countdown to notify the users that the vehicle will set out. During the mission, if the obstacle detector suggests that there are obstacles in the path the buggy is going, the vehicle will slow down, and the banner will show a warning to the users.

The appearances of destination buttons and the "START" and "STOP" buttons were improved, and a new safety check feature was embedded behind these buttons. During the mission, the destination buttons and the “START” button are disabled with some visual effects to notify the users these buttons currently being disabled.

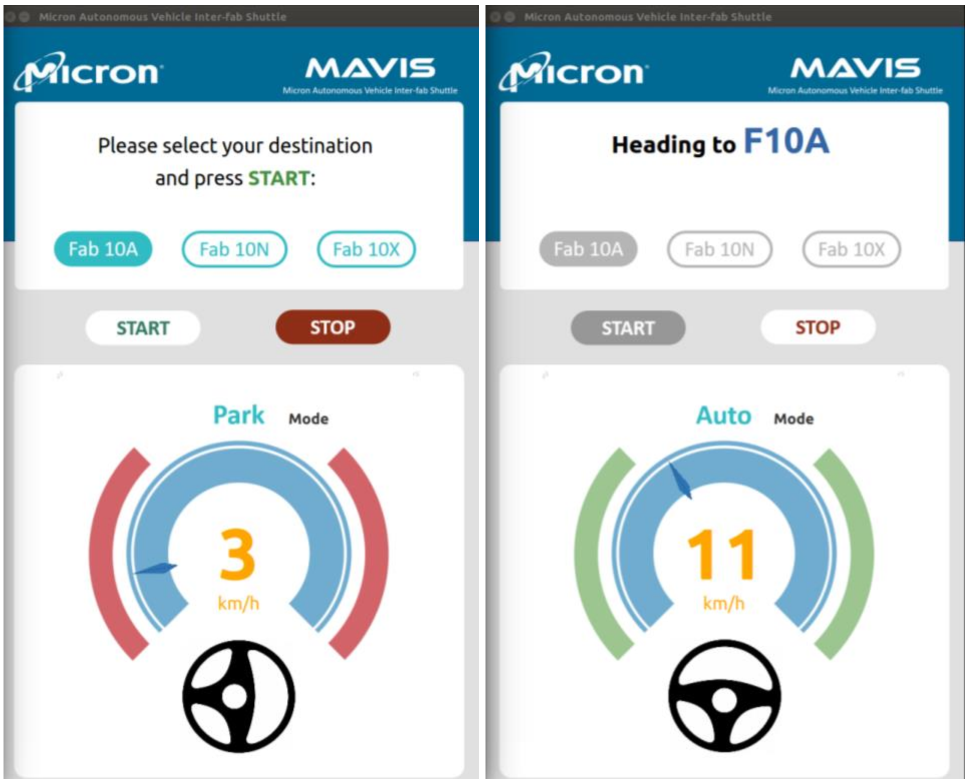
Micron Autonomous Vehicle Inter-fab Shuttle
Micron
MAVIS
Micron Autonomous Vehicle Inter-fab Shuttle
Please select your destination and press START:
Fab 10A
Fab 10N
Fab 10X
START
STOP
Park Mode
3
km/h
Micron Autonomous Vehicle Inter-fab Shuttle
Micron
MAVIS
Micron Autonomous Vehicle Inter-fab Shuttle
Heading to F10A
Fab 10A
Fab 10N
Fab 10X
START
STOP
Auto Mode
11
km/h

Figure 9.1: GUI in Park mode and in Auto mode

The bottom half of the new GUI is the vehicle status display area. In addition to the speed and steering status display, a vehicle mode display was added in the new design. To notify the user which mode the vehicle is currently at, the GUI displays one of the three vehicle-modes, park mode, auto mode, or manual mode on the screen.

In the new GUI, throttle and brake status are displayed through two bar surrounding the speedometer, similar to the function of ambient lights. The accelerator and brake status are received from the system, and the green or red colour displayed on the screen will vary according to the magnitude of the acceleration or brake. The harder the accelerator or brake is pressed, the darker these colours will appear.

Behind all these, the mainframe of the entire GUI was reconstructed to adapt to new changes. The new GUI frame was designed to be modular and friendly to new upgrade in the future. The functions are decoupled as much as possible and packed into modules that can be modified with minimal effects on the functions. The switches for several small features and some vehicles specific parameters that are different for each buggy are stored as parameters in a setting file, allowing the developers to change the settings and transfer the software between buggies quickly.

The new GUI has been tested on the vehicle extensively, and it has reached a stable version. In the future, if any new functions are developed in the central system, the GUI may need to be upgraded as well to incorporate new functions.

10. Conclusion and Future Work

10.1. Conclusion

All the new features mentioned in the previous chapters were integrated as one system on the vehicle and tested on the roads in Micron with a safety driver on board. By the end of the project, the new autonomous software on the vehicle was able to navigate through the roads safely and avoid obstacles that it encountered on its way. The visualisation of the new software system in real operations were shown in Figure 10.1, Figure 10.2, and Figure 10.3.

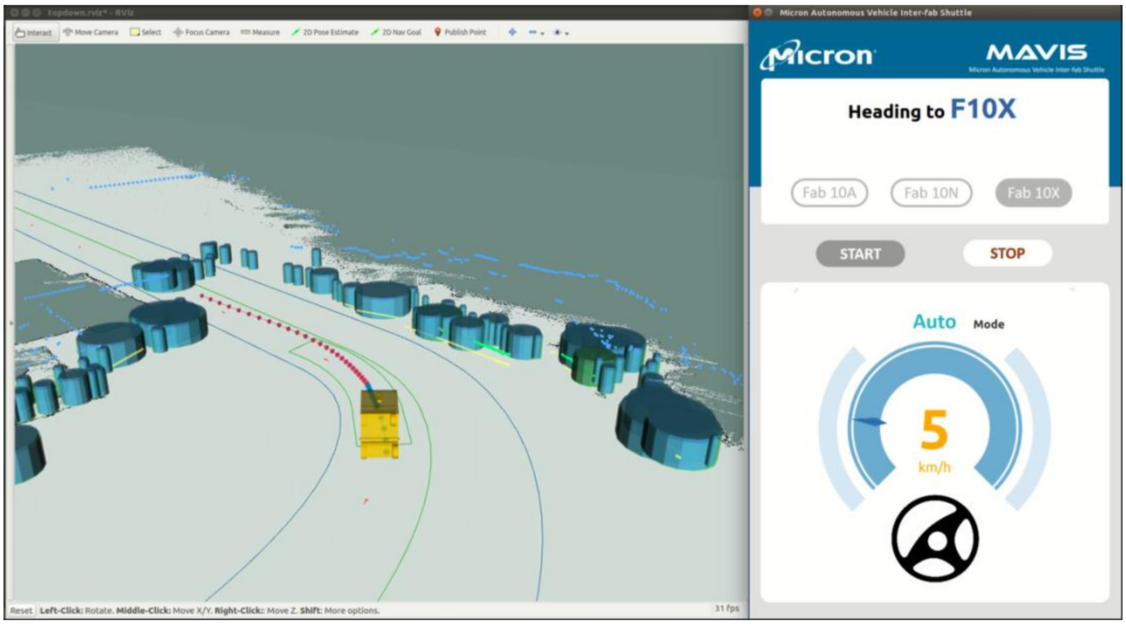

Figure 10.1: Trajectory planning at a left turn

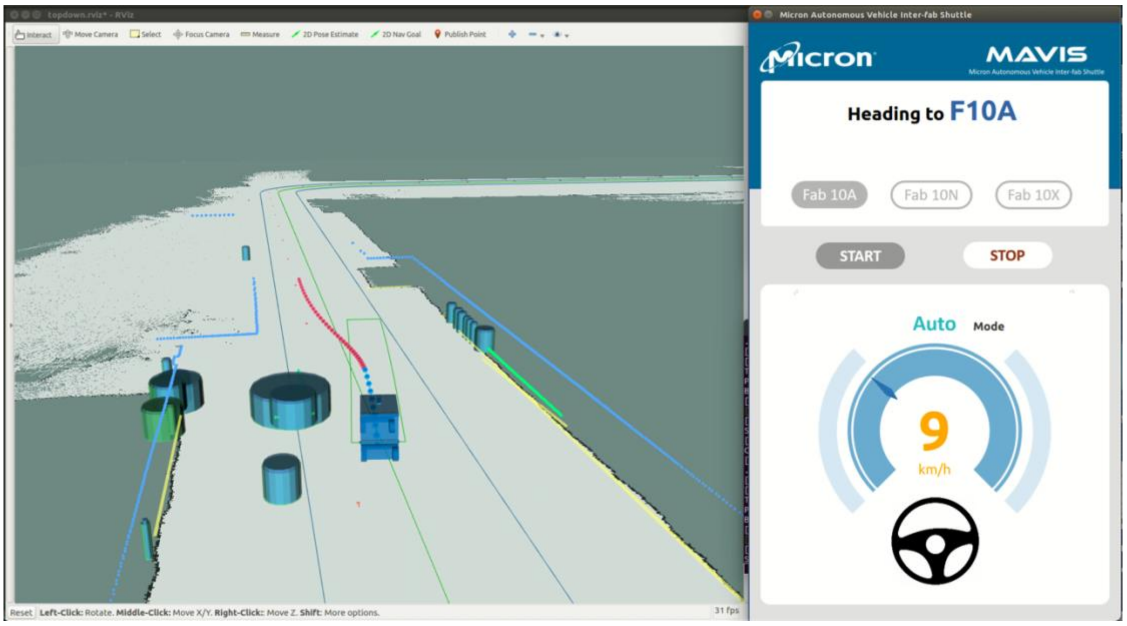

Figure 10.2: Avoiding a parked vehicle on the road side

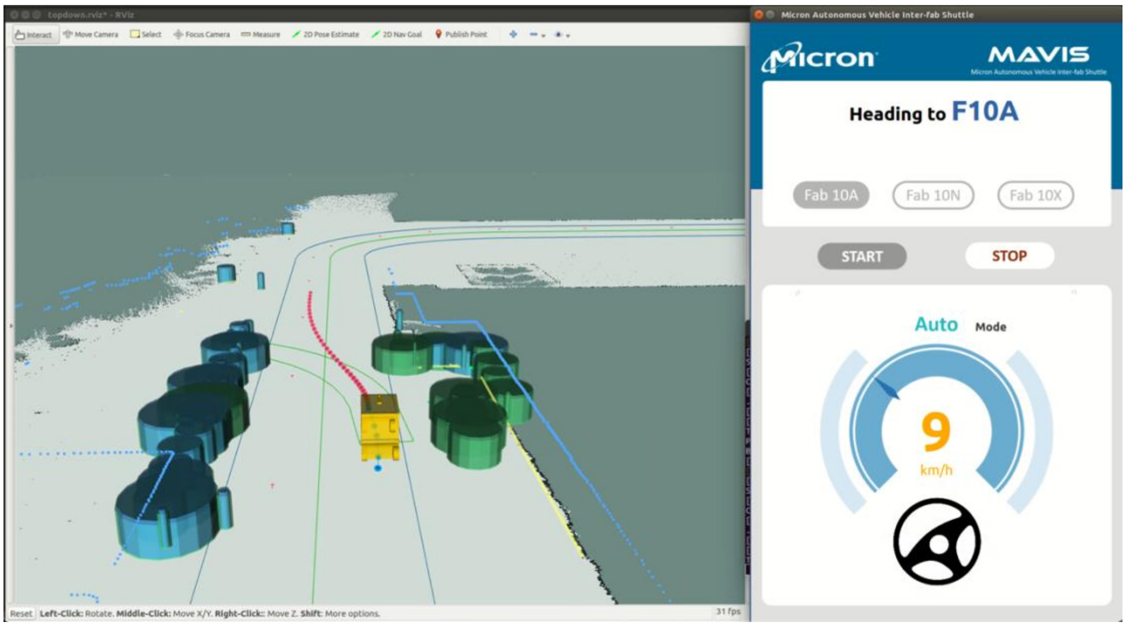

Figure 10.3: Switching back to the left lane

The performance and robustness of the new system were mostly satisfactory. However, there were still limitations and scenarios that the current software was unable to handle. There were still rooms for improvements inside every module.

10.2. Future Work

10.2.1. Perception

The perception module could be improved in terms of the accuracy of the obstacle detection and tracking results, and the robustness of the localization result.

The visual and LiDAR obstacle detection results described in this thesis was not sufficient in terms of the detection accuracy. The method treated every obstacle, no matter it was a car, a person or a part of the building, as one or multiple circles. However, in the real world, they were rarely circle. There were many state-of-the-art visual [10], [43]–[46] and LiDAR [8], [9], [13], [47]–[51] obstacle detection methods that could produce an accurate 3D bounding box around each obstacle, to provide a more accurate geometric estimation. The current obstacle tracking algorithm could also be improved by applying unscented Kalman filter with constant velocity and turn rate vehicle model. Visual object tracking [11] could be an alternative solution as well once the camera coverage became sufficient.

The localization result on the vehicle was not robust enough. In certain areas, the estimation of the vehicle position would start drifting due to magnetic field interference. This posed a danger to driving safety and required attention to it.

10.2.2. Map

Inside the map module, the lane information was packed in different files, with one file corresponded to one mission. However, as soon there would be many new possible routes and different missions would be very likely to share common road segments in their files, the maintenance of such a system would become tedious and erroneous. Thus, a better solution would be adopting a hierarchical structure and pack each road segment into one data structure. The switching between missions could be done by conducting a search among the road segments and load only the lane information that was on the selected route.

On the other hand, the content of the lane file could be richer by adding new attributes that would be useful for the mission.

10.2.3. Planning

The current behaviour planner had a simple structure which was only capable of basic autonomous functions and simple decision-making. It could be further improved by using a more advanced method such as online POMDP [29], [30] that would be capable of handling complex situations with uncertainties and interactions.

The current trajectory planner did not consider the motion of the obstacles, which made it only suitable for static and low-speed obstacles. It would become dangerous to drive with other fast-moving vehicles on the road. Thus, a simple solution could be adding the motion of the other obstacles into consideration. However, this would increase a new dimension to the algorithm's computational complexity. Another alternative was to switch to a new algorithm such as the lattice planner or EM planner [23] that were designed to deal with dynamic street scenarios.

After the perception module had been upgraded, and the detection and prediction of obstacles became more accurate, the collision checker would be able to utilise that information to achieve better safety.

10.2.4. Control

The MPC controller could be further improved based on the proposed improvements in Section 8.2.2, including optimising its computational speed and adding new cost functions. While on the other hand, to gain a more accurate description of the vehicle motion, a new non-linear vehicle model could be explored as well.

10.2.5. GUI

The current GUI only displayed very few information and the coordination between the Rviz panel and the control panel was not smooth. As more and more new features had been developed on the vehicle, it would be necessary to add more visualizations to the GUI to display that information crucial for the users to understand the vehicle's behaviours, such as camera images, obstacle bounding boxes, obstacle class name, and high threat targets. Another possible improvement in the future could be combining the Rviz panel and the control panel to form an integrated GUI.

References

[1] World Health Organization, “Global status report on road safety,” *Inj. Prev.*, 2015.

[2] S. Thrun *et al.*, “Stanley: The Robot that Won the DARPA Grand Challenge,” *J. F. Robot.*, vol. 23, no. 9, pp. 661–692, 2006.

[3] M. Montemerlo *et al.*, “Junior: The Stanford Entry in the Urban Challenge.”

[4] C. Urmson *et al.*, “Autonomous Driving in Urban Environments: Boss and the Urban Challenge,” *www.interscience.wiley.com). • J. F. Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[5] S. D. Pendleton *et al.*, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, 2017.

[6] J. Levinson *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2011, pp. 163–168.

[7] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012.

[8] S. Shi, Z. Wang, X. Wang, and H. Li, “Part-A^2 Net: 3D Part-Aware and Aggregation Neural Network for Object Detection from Point Cloud,” Jul. 2019.

[9] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, “STD: Sparse-to-Dense 3D Object Detector for Point Cloud,” Jul. 2019.

[10] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, “Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving,” Dec. 2018.

[11] X. Weng and K. Kitani, “A Baseline for 3D Multi-Object Tracking,” Jul. 2019.

[12] A. Buyval, A. Gabdullin, R. Mustafin, and I. Shimchik, “Realtime Vehicle and Pedestrian Tracking for Didi Udacity Self-Driving Car Challenge,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018, pp. 2064–2069.

[13] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, “Sensor

Fusion for Joint 3D Object Detection and Semantic Segmentation," Apr. 2019.

[14] K. Lewis, K. Hulme, E. Kasprzak, K. English, and D. Moore-Russo, "Experiential learning in vehicle dynamics education via motion simulation and interactive gaming," *Int. J. Comput. Games Technol.*, 2009.

[15] "Urban Challenge Route Network Definition File (RNDF) and Mission Data File (MDF) Formats," 2007.

[16] P. Bender, J. Ziegler, and C. Stiller, *Lanelets: Efficient Map Representation for Autonomous Driving.* .

[17] M. Werling BMW, S. Kammel, M. Werling, J. Ziegler, and S. Thrun, "Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenét Frame," 2010.

[18] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Jun. 2016.

[19] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annu. Rev. Control. Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, May 2018.

[20] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *4th Work. Algorithmic Comput. Robot. New Dir.*, pp. 293–308, 2000.

[21] J. J. Kuffner and S. M. La Valle, "RRT-connect: an efficient approach to single-query path planning," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2000, vol. 2, pp. 995–1001.

[22] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[23] H. Fan *et al.*, "Baidu Apollo EM Motion Planner," Jul. 2018.

[24] M. H. P. Bott and W. J. Hinze, "Potential field methods," *Dev. Geotecton.*, 2006.

[25] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for

mobile robot navigation," in *Proceedings - IEEE International Conference on Robotics and Automation*, 1991.

[26] L. Zhao, R. Ichise, Y. Sasaki, Z. Liu, and T. Yoshikawa, "Fast decision making using ontology-based knowledge base," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2016.

[27] S. Vacek, T. Gindele, J. M. Zöllner, and R. Dillmann, "Using case-based reasoning for autonomous vehicle guidance," in *IEEE International Conference on Intelligent Robots and Systems*, 2007.

[28] A. Furda and L. Vlacic, "Enabling safe autonomous driving in real-world city traffic using Multiple Criteria decision making," *IEEE Intell. Transp. Syst. Mag.*, 2011.

[29] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Springer Tracts in Advanced Robotics*, 2013.

[30] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2015.

[31] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell.*, vol. 128, no. 1–2, pp. 99–141, May 2001.

[32] M. Przybyla, "Detection and tracking of 2D geometric obstacles from LRF data," in *11th International Workshop on Robot Motion and Control, RoMoCo 2017 - Workshop Proceedings*, 2017, pp. 135–141.

[33] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors."

[34] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," Dec. 2015.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Jan. 2018.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[37] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully

Convolutional Networks," May 2016.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015.

[40] C. Szegedy *et al.*, "Going Deeper with Convolutions," Sep. 2014.

[41] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018.

[42] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2014, pp. 450–457.

[43] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. El Sallab, "YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud," Aug. 2018.

[44] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[45] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D Bounding Box Estimation Using Deep Learning and Geometry," Dec. 2016.

[46] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," Mar. 2019.

[47] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving," Mar. 2019.

[48] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection."

[49] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," Dec. 2018.

[50] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," Nov. 2018.

[51] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D Object Detection from Point

Clouds.”

Appendix A: User Survey

A small-scale user survey was conducted among some of the Micron team members to collect their opinions towards current shuttle services and autonomous vehicles. Twenty questionnaires were given out, and 17 valid responses were collected. The survey participants include engineers, executives, and contractors of Micron. After studying the result of all 18 questions in the questionnaire, some critical and instructive statistics are selected for discussion.

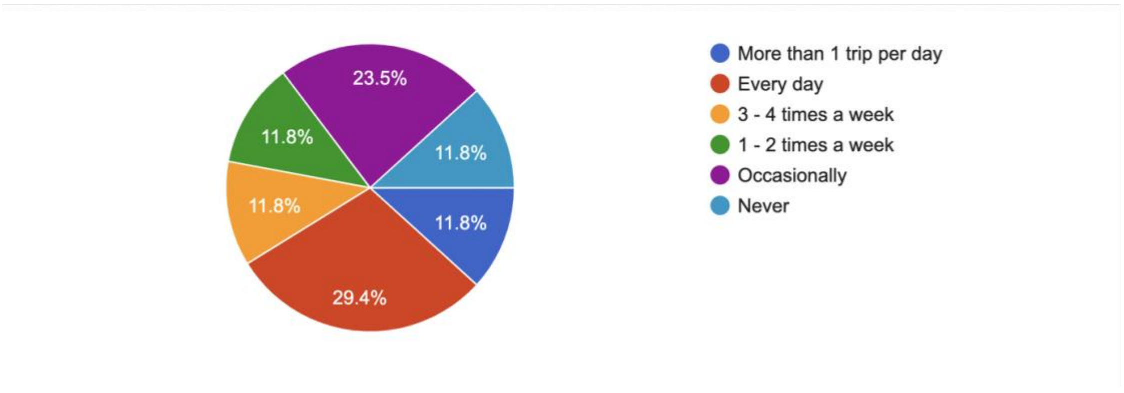

Figure A.1: Question No.2: How often do you need to commute within Micro

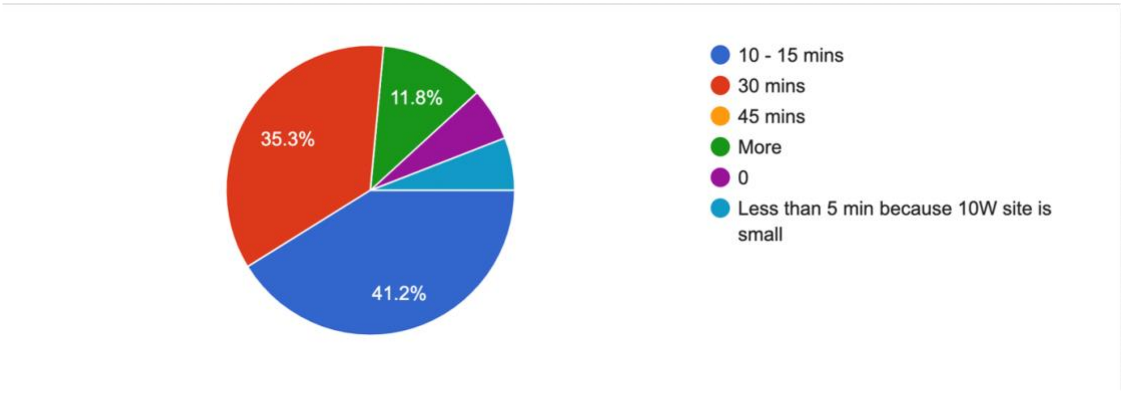

Figure A.2: Question No.4: How much time do you spend daily on commuting in Micron

First, the questionnaire asked participants questions regarding their behaviours and opinions towards current shuttle services. As shown in **Error! Reference source not found.** and Figure A.2, the frequency and intensity of travelling differs vastly from team member to team member. Different job scope and personal habits could result in this situation where some people's needs for shuttle service tend to be higher than others. Averagely, survey respondents travel 2 to 3 times per week and each time spends about 15 mins. In addition, question 3 shows that 82.4% of them usually travel by walking while only 17.6% of them take current internal shuttle buses. Question 6 further shows that 76.5% of the participants believe the campus is getting too big for walking. According to Figure A.3, the average estimate for the number of people commute in micron is about 500.

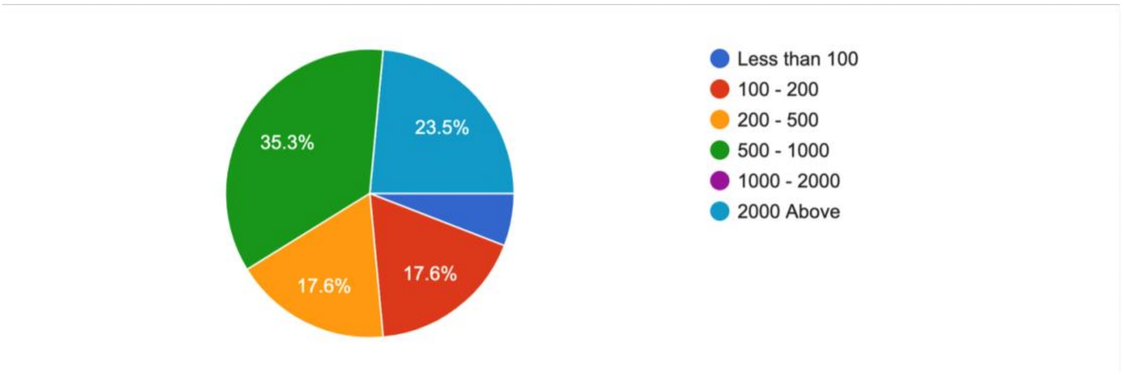

Figure A.3: Question No.5: How many people need to commute in Micron do you estimate

This survey also revealed that the top 2 causes of inconvenience to participants are the distance and the heat. When being asked what their expectations towards a new shuttle service was, timely, safety & comfort became their most popular options.