MONASH MOTORSPORT
FINAL YEAR THESIS COLLECTION

Simultaneous Localisation and Mapping using LiDAR for Autonomous Racing

Michael Mattiske -2018

The Final Year Thesis is a technical engineering assignment undertaken by students of Monash University. Monash Motorsport team members often choose to conduct this assignment in conjunction with the team.

The theses shared in the Monash Motorsport Final Year Thesis Collection are just some examples of those completed.

These theses have been the cornerstone for much of the team's success. We would like to thank those students that were not only part of the team while at university but also contributed to the team through their Final Year Thesis.

The purpose of the team releasing the Monash Motorsport Final Year Thesis Collection is to share knowledge and foster progress in the Formula Student and Formula-SAE community.

We ask that you please do not contact the authors or supervisors directly, instead for any related questions please email info@monashmotorsport.com

MONASH University
Engineering

Final Year Project

Simultaneous Localisation and Mapping using LiDAR for Autonomous Racing

Michael Mattiske

supervised by
Professor Tom Drummond

October 20, 2018

Significant Contributions

- Selected LiDAR unit to acquire.
- Designed mounting hardware for LiDAR unit.
- Designed and implemented a cone detection algorithm.
- Implemented a simultaneous localisation and mapping algorithm.
- Designed and implemented adaptations to SLAM algorithm to improve accuracy and reliability.

Executive Summary

Effective autonomous racing requires an accurate map of the race track and a precise and low-latency estimate of the vehicle's position within the track. This is known as the Simultaneous Localisation and Mapping (SLAM) problem. An end-to-end solution has been developed to solve this problem. The LiDAR sensor was chosen to provide environment perception, available models were investigated and an appropriate unit was selected. Mounting hardware was designed to mount the LiDAR unit to an existing racing car. With the platform this provided, a cone detection algorithm was designed and implemented to allow the system to perceive the cones that demarcate the track. These cones were used as landmarks in an Extended Kalman Filter (EKF) based SLAM solution that was implemented from scratch with adaptations to improve accuracy and reliability. Once the full system was shown to work effectively and provide the required functionality, it was implemented in optimised C++ code, ready to operate on an autonomous racing vehicle in real-time.

Acknowledgements

Special thanks to Georgia Ovenden and Bryce Ferenczi who both contributed to this project.

Thanks to Professor Tom Drummond for providing valuable guidance and feedback along the way.

Thanks to everyone at Monash Motorsport, both current members and alumni, who have offered help and advice throughout the year.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Monash Motorsport and Formula Student

Monash Motorsport is a student-run engineering team that designs, builds, tests and races open wheel formula style race cars as part of Formula SAE. The team has been competing in the combustion section since 2000 and competed with our first electric car in 2017.

Figure 1.1: Monash Motorsport's two cars. Combustion (left) and Electric (right)

Formula SAE/Formula Student is the world's largest student engineering design competition. The competition originated in the United States in 1978 and has grown to be a global competition with events throughout the world, one of which is FSAE Australasia where Monash Motorsport competes each year.

1.2 Formula Student Driverless

The new driverless competition ran for the first time at Formula Student Germany in 2017. At this event only one driverless car managed to complete all competition events. At the end of 2017 several students were recruited to work on developing the autonomous systems for the driverless vehicle and to undertake final year projects on relevant topics. The goal for the autonomous section is to have a car that can complete all events by 2019 with a plan to compete at FSG in 2020.

The competition is split into static and dynamic events. The static events test the team's business acumen and engineering knowledge, while the dynamic events test the performance of their race car. This FYP is focused on designing a system to contribute towards the dynamic events in the driverless category. These events are the following:

- Skidpad
- Acceleration
- Track drive
- Efficiency

The skidpad and acceleration events test the car's lateral and longitudinal acceleration respectively. The Track Drive involves navigating a previously unknown track for the duration of 10 laps and is the event with the highest points allocation. Points are scored in the efficiency category based on energy consumption during the Track Drive. All these events must be completed by the car without a driver.

1.2.1 Significant Rules Changes for 2019

Some significant rules changes have been made to the driverless competition for 2019 [1]. The most important of which is the introduction of the Autocross event. This event is a single lap of a previously unseen track. Importantly, data collected during this event can then be used during the Track Drive. This separates the previously combined challenge of navigating an unknown track and racing a known track at high speeds into two events. This serves to remove some complexity from the competition as manual tweaks can be made to the track map between Autocross and Track Drive to fix errors made by the SLAM algorithm. In the short-term this is a wise decision as it should lead to more teams successfully completing the events and results in more exciting spectating.

1.3 Previous Developments

In recent years significant developments have been made into the field of driverless vehicles. This has involved the continued development of existing proba-

bilistic methods for state estimation and research into the practicalities of implementing these solutions. The Formula Student competition has also recently branched into driverless vehicles and teams from around the world have developed driverless race cars [2–8].

Underpinning these recent developments is the Kalman filter which has been ubiquitous in robotics applications since its introduction. The Extended Kalman Filter adapts the original Kalman filter for nonlinear problems and has be-come the de facto standard for state estimation problems in robotics. The si-multaneous localisation and mapping (SLAM) problem builds upon this and allows a robot to autonomously navigate an unknown environment. A solution to the SLAM problem has long been considered a critical component of any truly autonomous system [9]. Since the SLAM problem has been solved [10] new techniques have been developed that provide benefits by the way of re-duced computational complexity or better performance for highly nonlinear sys-tems. These solutions include FastSLAM and Unscented Kalman Filter based approaches [11, 12].

The sensors that are used to perceive the environment are a critical component affecting the performance of the system. Sensors that can detect the environment with long range and low latency are beneficial. The most common approach for perception is LiDAR and cameras set up in a redundant system [2–4]. This is designed so that a single failure of a perception sensor is not catastrophic. The perception sensors are usually augmented by inertial sensors and odometry to provide accurate motion estimates between measurements [2–4, 6, 8].

A variety of techniques have been used to combine measurements from different perception sensors. The measurements can either be applied completely independently [2, 7] for maximum redundancy or a variety of techniques can be used to gain better performance or accuracy. One such method involves using rough estimates of cone positions from the LiDAR sensor to inform regions of interest for a camera based convolutional neural network. This provides the accuracy of camera based detection but with greatly reduced computational costs [8]. A multi-sensor SLAM solution as implemented by many of the Formula Student teams requires a way to manage differing sensor latencies. Various methods for solving this problem exist [13]. The simplest solution and one that is commonly used is to delay faster measurements to be in sync with longer latency measurements. Solutions that have been used by other Formula Student teams include running a simplified steady-state Kalman filter until the delayed measurements arrive [2].

Research into these previous developments has provided a strong foundation for Monash Motorsport's driverless vehicle and in particular the development of a perception pipeline.

1.4 Our Approach

Generally speaking the key challenges involved in converting an existing electric vehicle into a driverless vehicle for the Formula Student Driverless event can be split into three categories:

- **Perception:** Generate a map of an unknown track and determine the vehicle's location within this map and its state.
- **Planning and Control:** Given a full or partial map of the track determine and execute the optimal racing line to minimise time to complete the event.
- **Supporting Hardware:** Provide the compute hardware, software infrastructure and low voltage power and communications to enable the other categories to achieve their requirements safely and efficiently.

Monash Motorsport is tackling these challenges by assigning engineers into the following subsections which will be working in parallel to design the autonomous system of the driverless car while the rest of the team will be providing and maintaining the 2018 electric car as the platform for the design.

- Cameras
- LiDAR
- GPS/INS
- Path Planning
- Vehicle Actuation
- Low Voltage Systems
- Computing and ECU

1.5 Motivation for LiDAR

Redundancy is an important requirement in developing any autonomous system. This is even more the case for an autonomous vehicle capable of driving at high speeds and therefore of causing significant injury. For this reason, multiple sensors are required to ensure that in the case of a single failure the vehicle does not become unsafe. Many vehicle state sensors are available and in fact already exist on the Monash Motorsport cars. However, a key requirement for the event is to be able to detect the cones that act as landmarks demarcating the track. This significantly limits the applicable sensors. The obvious choice is a camera or set of cameras to provide depth information and our car will include such a system. Another excellent option for this use case, while less ubiquitous is LiDAR.

LiDAR is a 3D range finding technique that uses pulsed lasers to detect the distance to objects in the environment by timing the return of the laser after reflection off a surface. The most common method of providing 3D imagery from this is having multiple lasers arranged in a vertical array and swept across the

scene by rotating the device through 360 degrees. This provides some distinct benefits over other perception sensors such as cameras. Firstly, a much greater horizontal field of view is achievable than most sensors as the whole 360 degree range is visible. In addition, the depth accuracy of LiDAR is excellent and does not scale poorly with distance as per stereo cameras. Finally, due to several factors the total sensor pipeline introduces much less latency for LiDAR than with other sensors such as cameras. These factors include the lack of exposure time and lower data rates leading to much faster processing and object detection. Cameras do however offer their own benefits such as colour vision, improved range in our application and well developed and supported libraries for object detection. At Monash Motorsport for our autonomous car we have chosen to use both of these perception sensors to provide redundancy and to augment each other by their complementing strengths and weaknesses.

Chapter 2

System Overview

The system developed in this project consists of the full LiDAR pipeline required for SLAM. A system architecture diagram for the system is provided in Figure 2.1. The first element of the pipeline is the Velodyne Puck VLP-16 LiDAR (see Section 3.1 for details). The raw packet data is passed through the Velodyne ROS drivers [14,15] which provides a 3D pointcloud data structure. The pointcloud is processed by a cone detection filter that accurately extracts cones with very low latency (see Section 4.2 for details). The detected cones are published and received by the SLAM ROS node (see Section 5.3 for details). Currently the SLAM node has been developed to operate with LiDAR data but will be extended in the future to provide sensor fusion with cameras and GPS (see Section 6.2.1). Camera and GPS processing is being developed in parallel. The output of the SLAM node is the vehicle state and track map consisting of cone positions. This is provided to the path planning system which is also being developed in parallel.

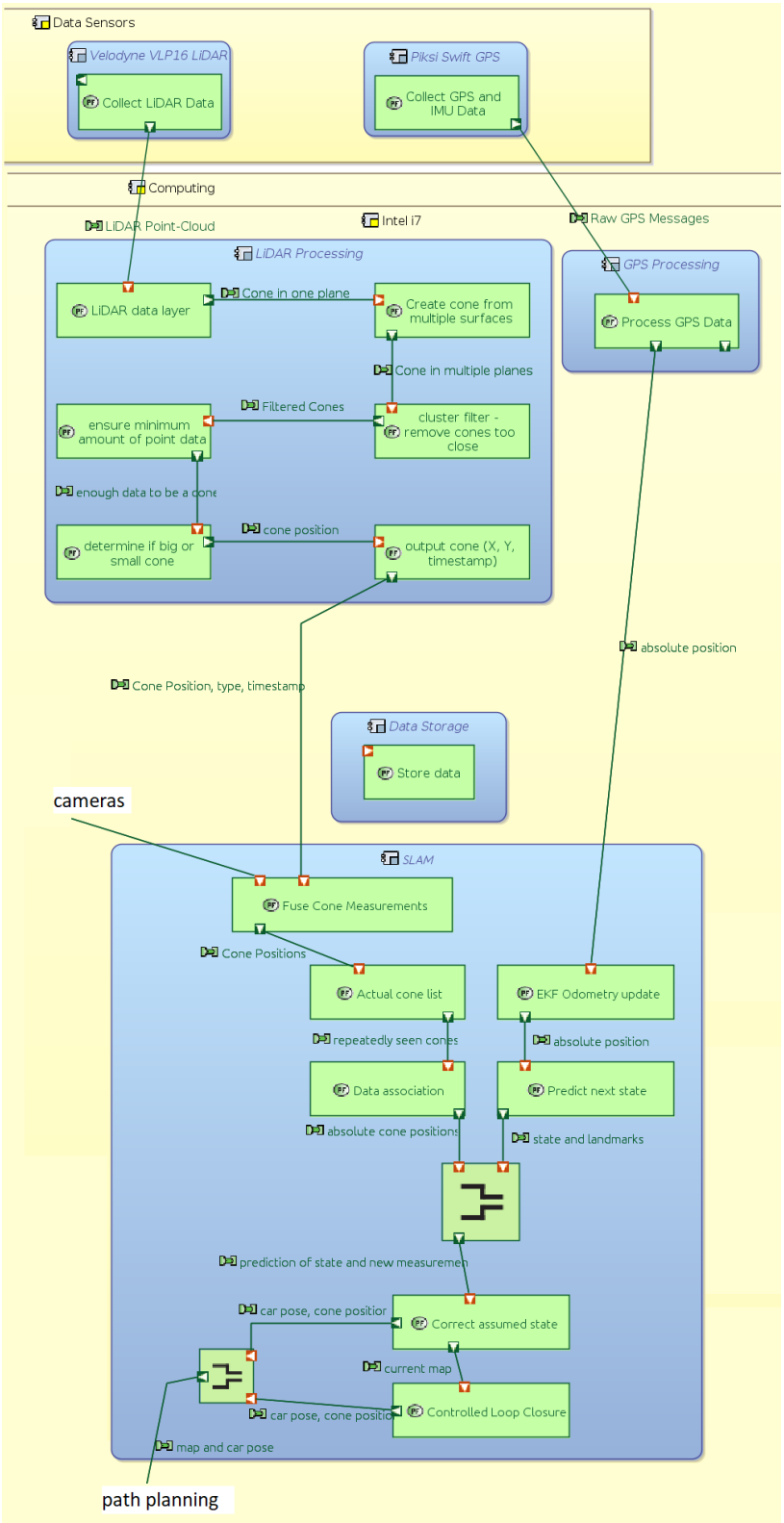

Figure 2.1: LiDAR and SLAM system architecture

Chapter 3

Hardware

While the major component of this project was developing algorithms for simultaneous localisation and mapping of the autonomous vehicle, a critical requirement is selecting the LiDAR device and designing its mounting method onto the vehicle.

3.1 LiDAR Selection

The primary function of the LiDAR sensor is to allow for perception of the environment and particularly the detection of cones. Early in the design process, a number of performance targets were developed to ensure this functionality could be met. These targets were based off performance achieved by competing Formula Student Driverless teams and are presented in Table 3.1.

Design Aspect	Target	Notes
Detection range	10m	Based on the method for cone detection this requirement constrains the vertical resolution
Measurement frequency	10Hz	This is calculated based on requiring 2 sightings of a cone with a range of 10m at 100km/h
Horizontal field of view (FOV)	180°	This is to maximise the number of cones that can be seen, particularly in tight corners

Table 3.1: LiDAR sensor performance targets

A variety of LiDAR units were assessed against these criteria to find the most suitable sensor.

As discussed in Section 4.2 two horizontal laser sweeps are required to reliably detect a cone. This meant that all planar LiDAR units could be removed from the candidate list. In addition, all LiDAR units that did not perform 360 degree

Rate of Revolution	Horizontal FOV	Horizontal Resolution	Vertical FOV	Vertical Resolution
5 - 20Hz	360°	0.1°- 0.4°	±15°or ±10°(Hi-Res)	2°or 1.33°(Hi-Res)

Table 3.2: Velodyne Puck VLP-16 specifications

sweeps could also be removed as they would not satisfy the horizontal field of view requirement. After these stages of filtering and at the time of writing the only two remaining units within our price range were from Velodyne. The two units were the Velodyne Puck VLP-16 [16] and the Velodyne Puck Hi-Res [17].

These two sensors have almost identical specifications (see Table 3.2) except for the vertical resolution and field of view which is the most important factor in determining the useful detection range of the sensor. Based on these differences the Velodyne Puck Hi-Res was evaluated as most suitable at the time of analysis due to its greater potential detection range. However, due to prohibitive cost and difficulty in acquiring a unit the Velodyne Puck VLP-16 was chosen as the best alternative.

3.2 Mounting

3.2.1 Mounting Location

Before a mounting design could be created, the optimal location for the LiDAR sensor on the vehicle was determined. The considerations made in this process were the Formula Student Rules [18], the available horizontal field of view and the detectable range of cones. Given the rule outlined in Figure 3.2 there are three obvious locations to consider.

- Near the top of the main roll-hoop (eliminated due to camera mount location)
- On, or in place of, the front wing (see Figure 3.1 right)
- Above the nosecone (see Figure 3.1 left)

The mounting position above the nosecone is limited by the vertical limit shown in Figure 3.2 of 500mm. Due to the existing design of the electric vehicle this makes it difficult to mount in this location. An additional advantage of mounting at the height of the front wing is this height bisects the cones and provides maximum likelihood to sense a cone with multiple laser sweeps from the LiDAR which leads to greater detection range. The front wing level mount is also popular with successful Formula Student Driverless teams from 2017 and this provides assurance that it is a workable design. This reasoning led to the selection of the mount at the height of the front wing concept.

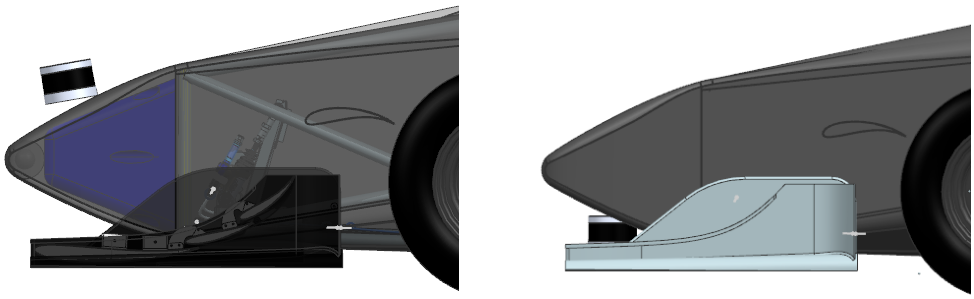

Figure 3.1: Mounting location concepts

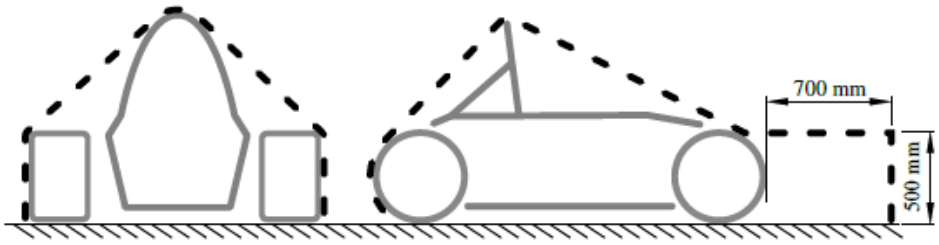

Figure 3.2: Envelope to mount sensor systems (DV 4.1.3)

3.2.2 Testing Mount

A testing mount was designed that could be interchangeably mounted on a current or previous car to allow for easy collection of data to aid in developing the cone detection and simultaneous localisation and mapping algorithms. This mount was designed for ease of manufacture and adjustability and did not need to meet all rules requirements. To simplify the design the Impact Attenuater (crash structure for the car) was removed, this limited us to pushing the vehicle or low speed driving for safety reasons. This would not affect our ability to collect useful data as, at least initially, the autonomous vehicle will be driving at similarly low speeds. The design of the mount is shown in Figure 3.3.

3.2.3 Final Mount

For the car to be competition ready, a competition compliant mount had to be designed that met all rules requirements and could stand up to the physical stress of high speed driving.

The decision was made to run the driverless car without the front wing. At the low speeds that the driverless car is expected to reach in the near future the downforce provided by the front wing is not significant for vehicle performance. Due to limited mounting space between the front wing position and the nosecone it is easier to design a mount with the front wing removed as this provides greater

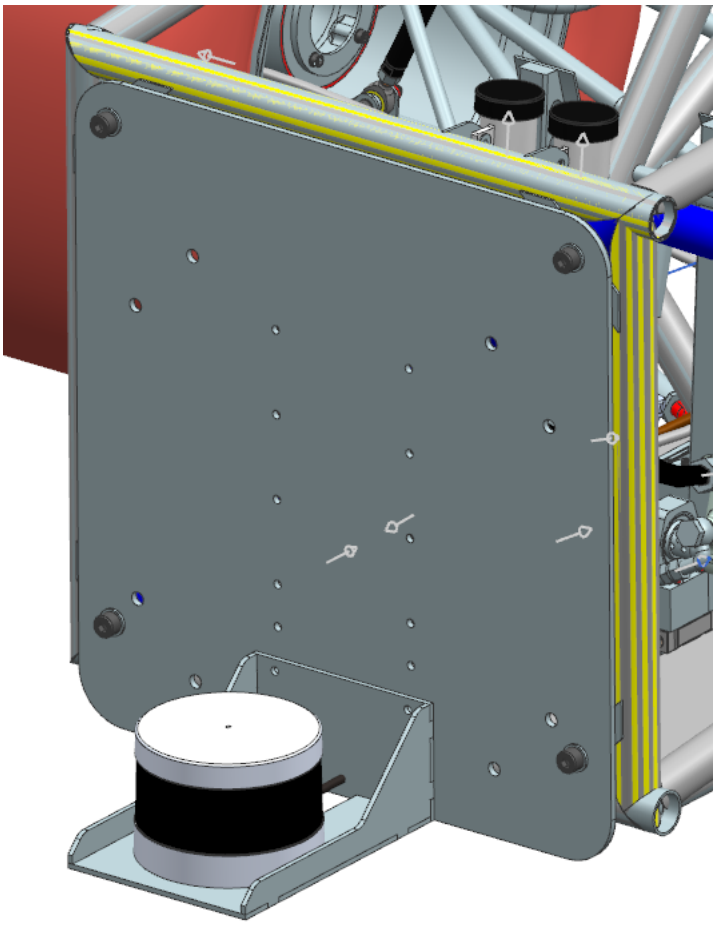

Figure 3.3: Mount developed for initial data collection

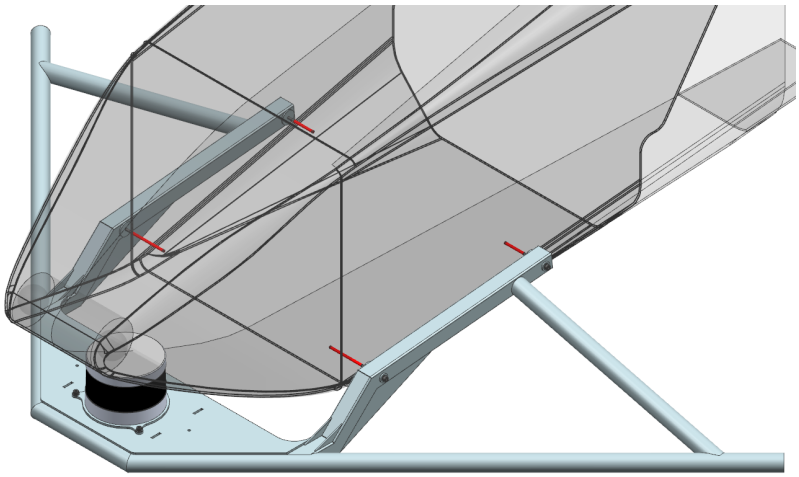

Figure 3.4: Final mount design

flexibility. Combining these factors led to the design to mount the LiDAR with the front wing removed.

The major rules requirements for the mount are the sensor envelope shown in Figure 3.2, a restriction on the minimum radius of forward facing edges and a requirement to meet certain crash structure maximum force limits. It was determined the most effective way to meet the crash structure rules was to mount the LiDAR mount off the existing front wing mounting points as this guarantees rules compliance.

While ensuring rules compliance the mount was optimised to allow for maximum horizontal field of view of the LiDAR. As per Table 3.1 the baseline goal for horizontal FOV is 180°, while maximising this value is always beneficial.

A cone deflection bar was designed into the mount. This helps to prevent cones becoming lodged within the suspension or aerodynamic structures. The bumper is designed to deflect cones to outside the track width. This is particularly important as in early testing we are expecting regular cone collisions.

A removable plate was designed to mount the LiDAR. This makes removing the LiDAR when not in use easy and minimises the use of the single embedded threaded insert in the base of the LiDAR unit to prevent accidental damage. This allows the key bolt to be positively locked and rarely or never removed. This plate is shown in Figure 3.5.

The mount was tested using Finite Element Analysis (FEA) to ensure that it could withstand the applied forces under event and testing conditions. This includes high speed collisions with cones and potentially scraping on the track surface. The cone collision load was calculated as follows

$$F = \frac{mv^2}{2d} \tag{3.1}$$

where m = 1 kg, v = 25 m/s and d = 0.2 m

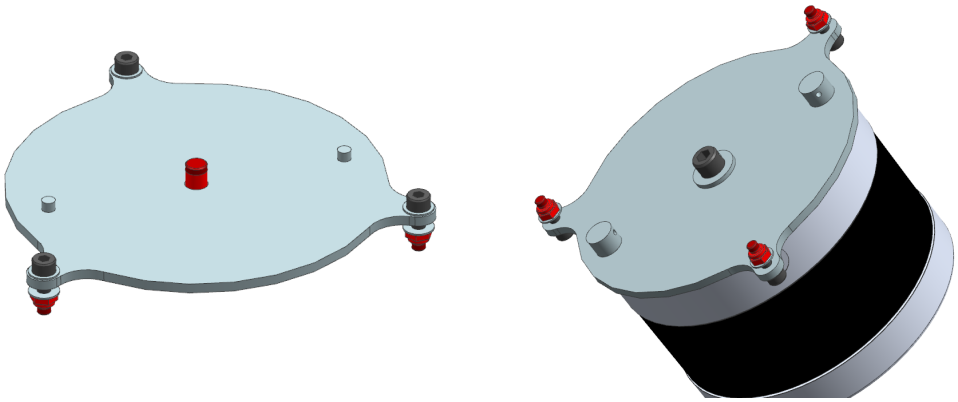

Figure 3.5: Removable LiDAR plate design

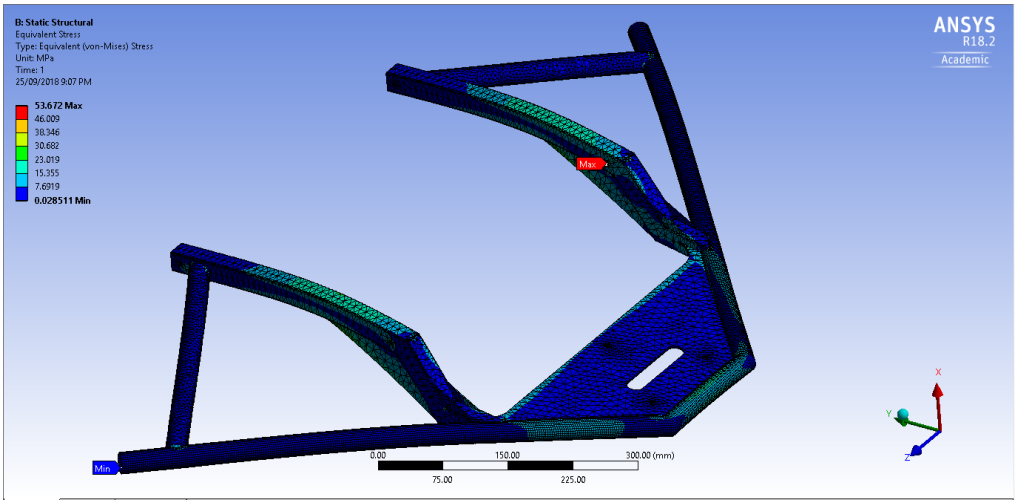

Figure 3.6: Finite element analysis of high-speed cone collision

This yields a load of 1600N which is used in the analysis shown in Figure 3.6.

Chapter 4

Environment Perception

To enable any form of autonomous behaviour the driverless vehicle must be able to perceive the environment around it. The environment perception problem is simplified in our case as the track is delineated by coloured cones. An algorithm is required that provides a picture of the scene that can be compared over time to estimate the motion of the vehicle. This can either be performed based on the full scene, for example using an Iterative Closest Point algorithm or based on discrete landmarks which in our case are the cones.

4.1 Iterative Closest Point

Tests were performed using the Iterative Closest Point algorithm (ICP) on successive LiDAR frames to determine if an accurate and reliable transformation between the two states of the vehicle could be developed. These tests were largely unsuccessful. The problems encountered stemmed from the nature of the pointcloud produced by the LiDAR sensor that we have access to and the environments used. As discussed in sensor selection (see Section 3.1), within the targeted price range the available LiDAR units at time of selection are quite low resolution in the vertical direction with few, relatively widely spaced laser sweeps, in our case 2 degree spacing. This is compounded by the sparse environments of race tracks designed to avoid collisions rather than provide rich environments for localisation. Combining these factors meant that the majority of LiDAR returns were reflections off the ground and much of the rest were the cones, as expected (Figure 4.1). This prevented ICP from working effectively. The ground is largely a plane parallel with the direction of travel and thus forms a strong circular structure when intersected by a conic laser sweep. Without influence from any motion of the car relative to the ground which may be caused by bumps or weight transfer due to acceleration this pattern will remain in a fixed location relative to the LiDAR. This means that the ICP algorithm perceives the scene as having not changed and hence the car as stationary as the ground outweighs the useful objects detected such as the cones. To overcome this issue the ground needs to be reliably removed from the scene. Assuming

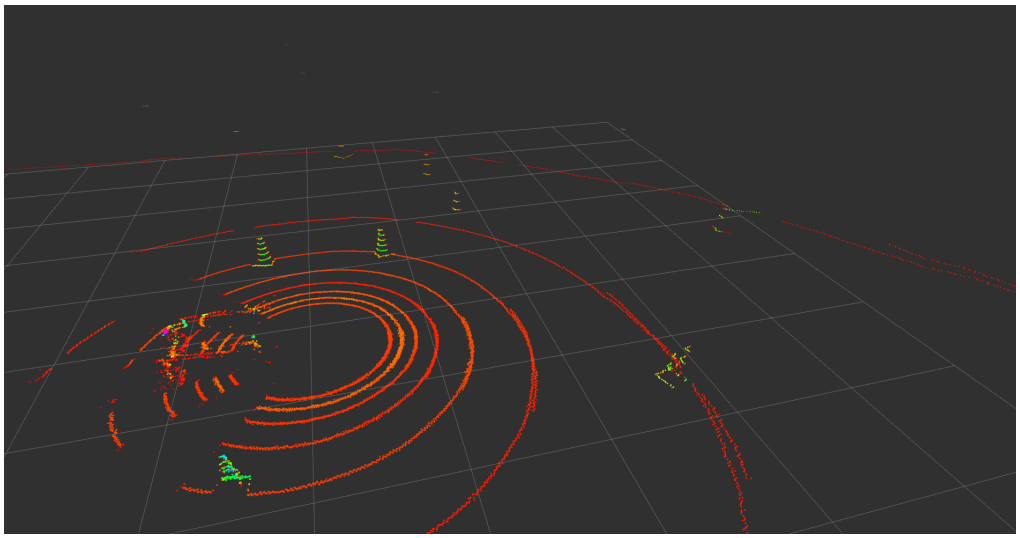

Figure 4.1: Raw LiDAR data showing unfiltered vehicle

this can be done effectively, once the ground has been removed, other than the occasional environmental feature (trees, posts, tire barriers etc), all that is left are the cones. This leads us to the conclusion that it is both easier and more effective to simply try to detect the cones and use them in a landmark based SLAM solution.

4.2 Cone Detection

To provide useful information to the path-planning and control systems, a track map needs to be built containing the locations of the cones which demarcate the track. To achieve this, an accurate and robust algorithm needs to be developed to detect the cones. There are a couple of different approaches to solving this problem. Either the cones need to be detected progressively as the LiDAR sweeps the scene or they can be detected holistically once a full 360°sweep has been completed. An algorithm that progressively detects cones is beneficial as it leads to less latency within the system and as such has been developed.

The developed algorithm works by applying successive layers of filtering to remove false positives from an initial, broad detection routine. The output of the initial detection routine is shown in Figure 4.5. This broad detection routine operates on packets of LiDAR data as they arrive. Points that are outside the usable field of view of the LiDAR are removed, this filters out points that are reflections from the vehicle (shown in Figure 4.1). After this the algorithm measures surfaces detected in each layer of the LiDAR sweep. A surface is defined as a string of successive points in a layer where the difference in range between consecutive points is small. Where a larger difference is detected this signifies the start or end of a surface. Once a surface has been completed (a transition has been detected at the start and end) the surface parameters are passed through a preliminary filter to see if they could be a section of a cone,

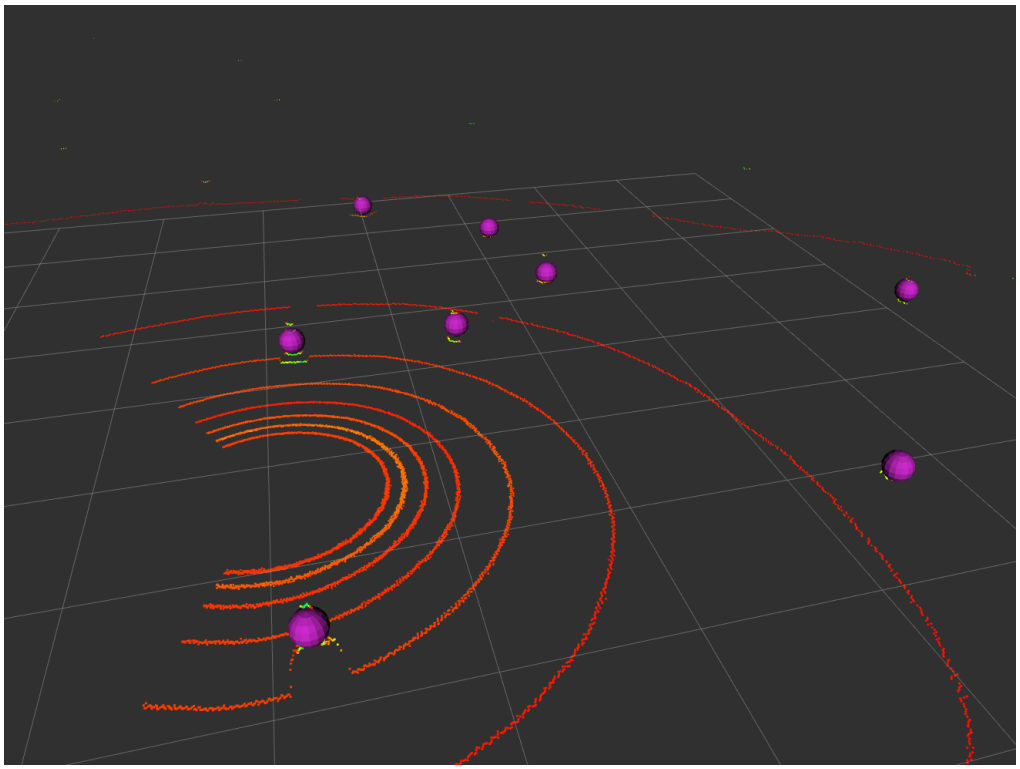

Figure 4.2: Visualisation of detected cones overlaid on LiDAR data

the checked conditions are

- **Surface Width:** Is the surface narrower than the width of a cone?
- **Range of Measurement:** Is the surface within the maximum theoretical distance for a cone to be detected in more than one layer?
- **Point Count:** Does the surface consist of more than one point?

Once this is passed, the surface is added to a preliminary list of cones. As more surfaces pass these conditions if these surfaces have a midpoint (X,Y) close to a previous "cone" they are combined else they are added as a new cone.

The test data we have used includes sections of track that have tall grass or bushes running alongside. This produces many surfaces which are accepted within the first basic filter. Therefore a cluster filter is also added. If more than two preliminary cones are close together they are removed since the actual track should not have cones in close proximity. It is very difficult to filter out all of the false positives as occasionally there are objects outside the track that are far from other objects and have all the features of a cone. While this means some false positives remain, reliable objects outside the track can act as landmarks for SLAM and do not cause issues.

The final filtering step before the results are output includes the following conditions

- **Layer Count:** Is the cone seen in more than one layer? This makes the

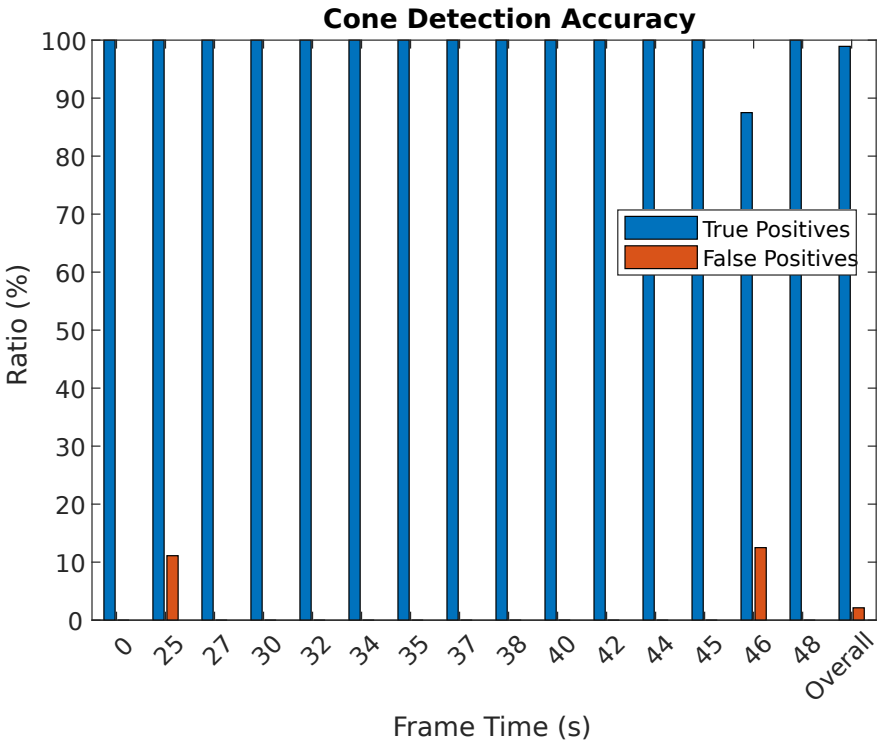

Figure 4.3: Accuracy of cone detection algorithm for various time steps

detection much more robust.

- **Point Count:** Does the cone have a minimum number of particles?
- **Cone Height:** If the height is <30cm then it is a small cone, else if the height is <50cm it is a large cone. If the height is greater than 50cm it is not a cone.

Once all of these conditions are met, the cone's location (X,Y) and detection time are passed into the SLAM algorithm.

The detection accuracy of this algorithm is quantified in Figure 4.3. Cones were manually labelled in the LiDAR data and the results of the algorithm were compared against this baseline. This shows that the algorithm is very reliable with minimal false positives and correctly detects over 95% of the cones. The ability for this cone detection algorithm to differentiate small and large cones is shown in Figure 4.4. This functionality is critical for loop closure as it means the large start cones can be differentiated from the other cones (see Section 5.3.2 for details).

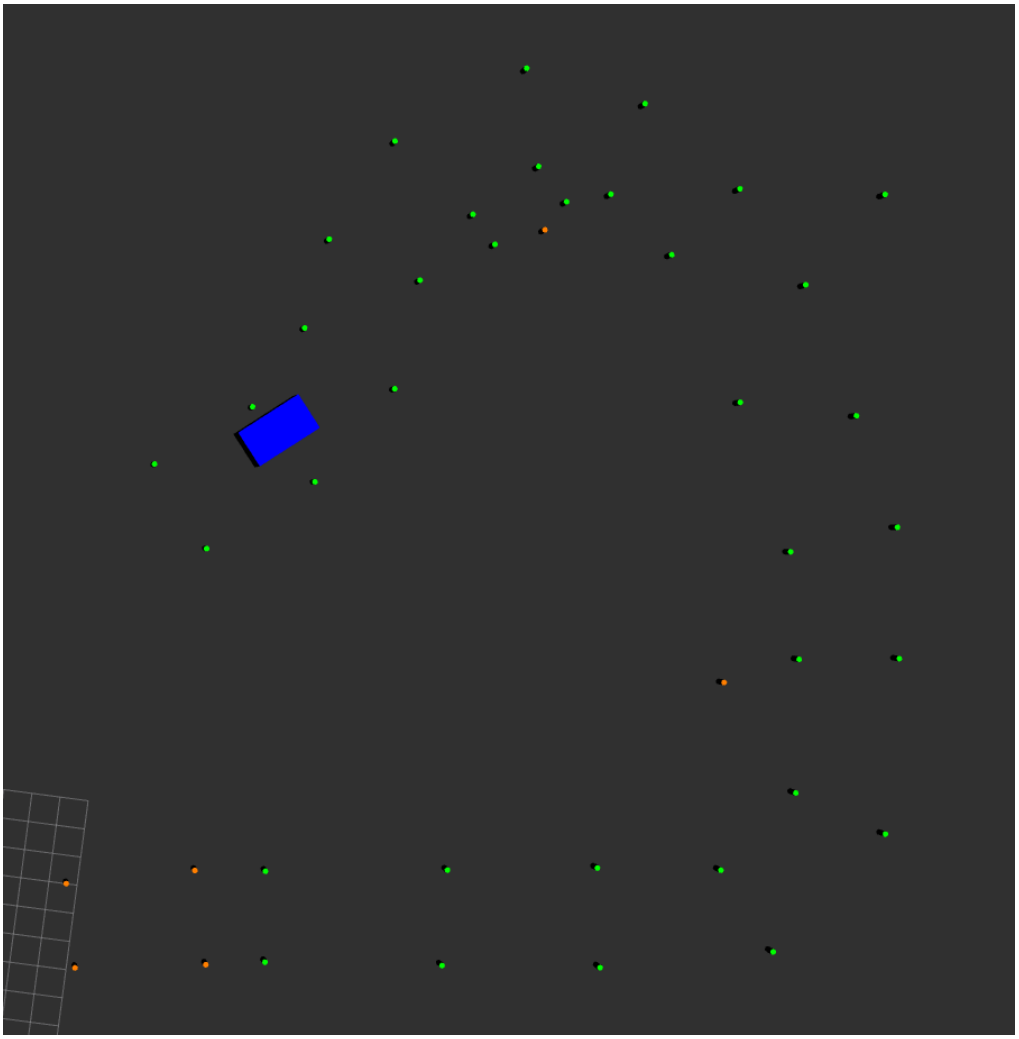

Figure 4.4: Demonstration of detection of small and large cones. Small cones are green. Large cones are orange. Some objects are incorrectly identified as large cones, this is caused by spurious objects such as light posts near the track.

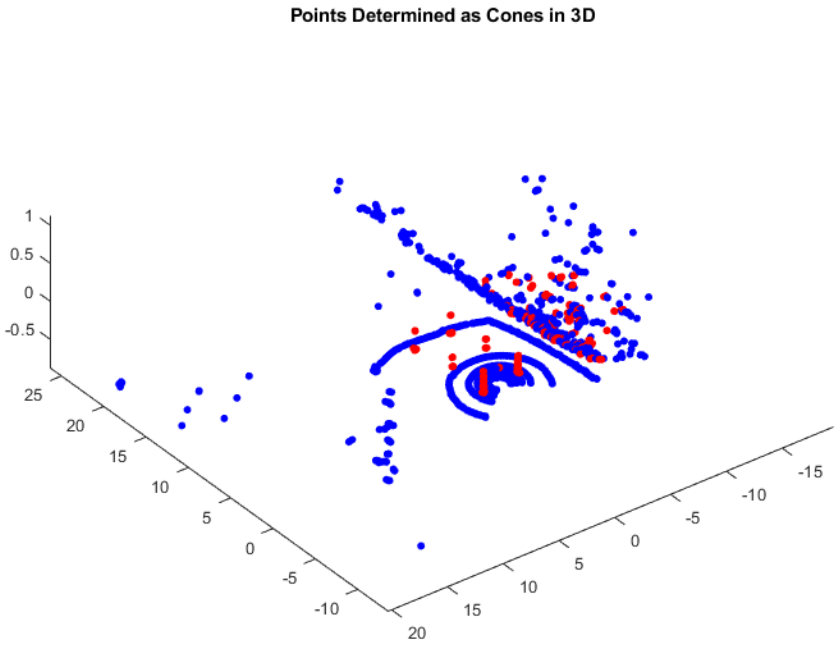

Figure 4.5: Cone detection in MATLAB before filtering

Chapter 5

Simultaneous Localisation and Mapping

In the Formula Student Driverless event the track is not known beforehand and there is no ground-truth for the location of the vehicle. Therefore, a simultaneous localisation and mapping (SLAM) algorithm is required. There are a wide variety of algorithms that can be used for this function and a few select options will be compared.

The SLAM problem involves generating a simultaneous estimate of both vehicle and landmark locations. The true locations are never known or measured directly. Observations are made between the true vehicle and landmark locations and are used to build correlations between these objects (see Figure 5.1) [9]. This requires having some form of model of the current scene and what has been seen previously (the map) and comparing the changes to estimate the motion of the vehicle during a time step.

There are two broad approaches to representing the scene and map. One option is to look at the whole environment and use many, but individually potentially unreliable features drawn from anywhere in the scene. The other option is to look for specific objects to be used as landmarks, generally this will result in a more sparse map but with each landmark being more reliable. Because of the nature of the Formula Student Driverless event we have convenient access to using the cones that demarcate the track as our landmarks for SLAM. A potential whole-scene based approach is discussed in Section 4.1 on iterative closest point while a landmark based approach will be assumed for this discussion.

There are two types of SLAM problem. Online SLAM keeps track of the current state of the vehicle and of the map and has sufficient performance to run in real time. Full SLAM also keeps track of the state of the vehicle at all previous time steps and is usually run after the fact [19]. For our application the previous states of the vehicle are irrelevant and as such an online solution is desirable.

One of the key limitations that needs to be considered is the continuous nature of the LiDAR sweep. Unlike a camera system where a single frame can accurately

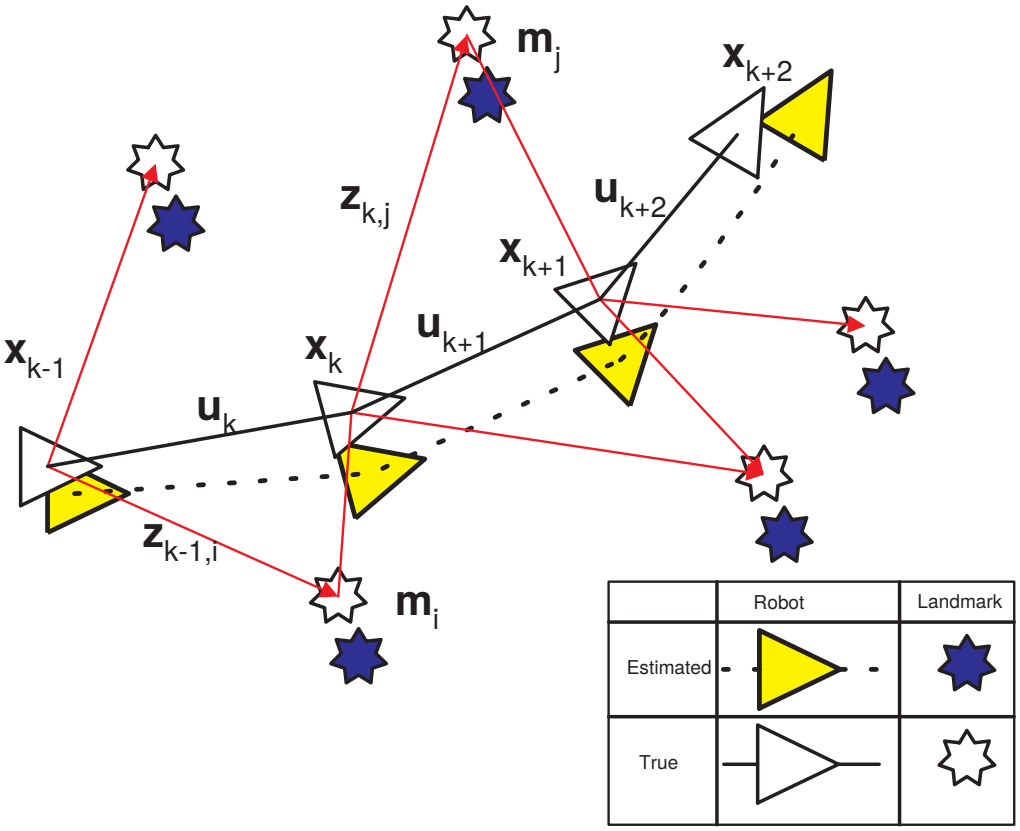

Figure 5.1: Diagrammatic formulation of the SLAM problem [9]

be described as being captured at a single point in time the LiDAR sweeps across its 360°field of view in 100ms. This means that given the maximum speeds achievable in Formula Student of roughly 100km/h the vehicle may have moved 2.8 metres during the duration of a single sweep. This is considerable and means that a single sweep of the LiDAR cannot be assumed to occur at a point in time.

There are two ways that this issue could be resolved. Either perform a transformation on the LiDAR pointcloud to undo the distortion caused by the car moving during the sweep. Or, treat each cone detection as a separate measurement at a distinct point in time. When this is considered in a probabilistic SLAM context these two approaches are equivalent if performed correctly. However, treating each measurement independently is simpler in practice. To apply all measurements together the cone detections have to be shifted to a single point in time. When this is modelled it would result in varying amounts of measurement noise being applied to each of the detections based on how far they have to be shifted through time to reach the common point. This is largely equivalent to applying the same measurement noise to each cone measurement and applying them separately with process noise of the car added between measurements according to the time elapsed. For accuracy and simplicity each measurement will be applied separately.

We will consider two common SLAM paradigms that apply to the online SLAM problem [19] these are Extended Kalman Filters and Particle Filters (an implementation of which is FastSLAM). Both of these paradigms will be investigated and a solution developed upon the Extended Kalman Filter approach.

5.1 Extended Kalman Filter SLAM

The Extended Kalman Filter (EKF) is an extension of the Kalman Filter to nonlinear systems. This extension is made by approximating nonlinearities as linear systems using a first order Taylor series expansion. Excellent explanations and derivations of the Kalman filter are provided by [20] and [21]. The EKF is broken into two steps. The predict step (Table 5.1) propagates the vehicle state forwards through time using a motion model. The update step (Table 5.2) takes this prediction and uses landmark measurements and an observation model to provide a corrected and more accurate estimate.

Predicted state estimate	$\hat{\boldsymbol{x}}_{k\|k-1} = f(\hat{\boldsymbol{x}}_{k-1\|k-1}, \boldsymbol{u}_k)$
Predicted covariance estimate	$\boldsymbol{P}_{k\|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1\|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k$

Table 5.1: The predict step of the extended kalman filter

Innovation	$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k\|k-1})$
Residual covariance	$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$
Kalman gain	$\boldsymbol{K}_k = \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$
Updated state estimate	$\hat{\boldsymbol{x}}_{k\|k} = \hat{\boldsymbol{x}}_{k\|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$
Updated covariance estimate	$\boldsymbol{P}_{k\|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k\|k-1}$

Table 5.2: The update step of the extended kalman filter

5.2 Comparison of EKF SLAM and FastSLAM

The two main approaches for online landmark-based SLAM are EKF SLAM and FastSLAM. The use of either of these algorithms represents a trade-off. EKF SLAM uses a single large Kalman filter to represent the location and uncertainty in the vehicle and all of the landmarks. FastSLAM, however, uses a particle filter to represent the distribution of states for the vehicle. Instead of representing this distribution parametrically as a Gaussian, it is represented with many particles which describe the distribution through sampling. Each of these samples, or particles, maintains its own list of landmarks, each of which has its distribution represented by a small 2x2 Kalman filter. No correlation is stored between any of these landmarks. This can be achieved because each particle represents a single 'perfect' estimate of the state and the combination of each of these 'perfect' estimates represents the distribution of uncertainty in the state [11].

This alternate method of representing the uncertainty leads to several advantages and disadvantages of FastSLAM over EKF SLAM [12, 22]. EKF SLAM

has been shown to generally perform better in closing long loops. This is because over time the particles that represent the uncertainty in the vehicle state can end up sharing common ancestors. If only a single common ancestor remains no corrections as a result of loop closure can occur before this fork in the particle ancestry. FastSLAM can successfully deal with more ambiguity in the data associations. Each particle makes its own data associations which leads to an overall robust data association and incorrect associations can be corrected in the future. By comparison, EKF SLAM only makes a single data association and if this is incorrect the filter can diverge with no recovery. Because the cones all appear identical to the LiDAR, being robust to ambiguity in data associations will likely be very important to achieving an accurate system. FastSLAM has lesser computational complexity for large numbers of landmarks with a complexity of $\mathcal{O}(M \log N)$ compared to $\mathcal{O}(N^2)$ for EKF SLAM where M is the number of particles used in FastSLAM and N is the number of landmarks [11].

After considering these factors it was decided to work on an EKF SLAM implementation first as it is simpler to implement and then develop a FastSLAM implementation later if EKF SLAM fails to meet performance requirements or if incorrect data associations are posing a major issue.

5.3 Implementation of EKF SLAM

Controlling an autonomous racing car requires accurate and low-latency estimates of the car's state and the surrounding track. This necessitates an optimised, platform targeted implementation to maximise computational performance. While MATLAB provides an excellent platform for development and testing of algorithms a different solution was required for implementation onto the car. C++ was chosen to fulfil this role due to it's ubiquity, performance and support within the Robot Operating System (ROS) [14]. In this section, details will be provided of our specific Extended Kalman Filter implementa-tion including adaptations to the standard algorithm and techniques to improve performance and ensure accuracy of the filter.

5.3.1 Models

The extended Kalman filter requires mathematical models of the sensor measurements and of the vehicle to provide predictions. These predictions are then compared to measured outcomes which forms the basis of Kalman filter update.

Motion Model

The Kalman filter state contains the position, orientation and linear velocities of the vehicle. These all exist in a global coordinate frame that is referenced to the starting location of the robot. Acceleration and angular velocities are

available from the IMU and exist in the vehicle's coordinate frame. Before the acceleration from the IMU can be applied it has to be transformed to the global coordinate frame as shown

$$\begin{bmatrix} \ddot{x}_{global} \\ \ddot{y}_{global} \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \ddot{x}_{vehicle} \\ \ddot{y}_{vehicle} \end{bmatrix} \tag{5.1}$$

where

$$\alpha = \theta_{t-T} + \frac{T^2}{2}\dot{\theta}_{meas} \tag{5.2}$$

The simple point-mass vehicle model is shown

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T & 0 \\ 0 & 1 & 0 & 0 & T \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-T} \\ y_{t-T} \\ \theta_{t-T} \\ \dot{x}_{t-T} \\ \dot{y}_{t-T} \end{bmatrix} + \begin{bmatrix} 0 & T^2/2 & 0 \\ 0 & 0 & T^2/2 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \begin{bmatrix} \dot{\theta}_{meas} \\ \ddot{x}_{global} \\ \ddot{y}_{global} \end{bmatrix} \tag{5.3}$$

These formulas propagate the state of the Kalman filter forward by T seconds using the simple vehicle model. This model can be easily extended to include linear acceleration and angular velocity within the Kalman filter state. This can provide benefit in filtering these derivatives but comes at the cost of having to apply measurements as an update which is computationally more expensive than a prediction.

Observation Model

A measurement model is required to convert a detected landmark in polar coordinates (ρ and ϕ) relative to the LiDAR to cartesian world coordinates. The LiDAR is positioned on the car with a single longitudinal offset along the centreline from the vehicle origin. This transformation allows the detected landmark to be compared directly to existing landmarks within the map.

$$\begin{bmatrix} x_{landmark} \\ y_{landmark} \end{bmatrix} = \begin{bmatrix} x_{vehicle} \\ y_{vehicle} \end{bmatrix} + \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} \rho + \begin{bmatrix} \cos\theta_{vehicle} \\ \sin\theta_{vehicle} \end{bmatrix} \text{offset} \tag{5.4}$$

where

$$\alpha = \theta + \phi \tag{5.5}$$

5.3.2 Controlled Loop Closure

Loop closure is a very important part of the SLAM algorithm. By default, EKF SLAM will match detected cones with what it thinks at the time to be the most likely match of previously seen cones. However, because all of the cones appear identical, if the estimated position of the car has drifted significantly since seeing a cone (such as in the case of long loop closure) a new cone could be incorrectly

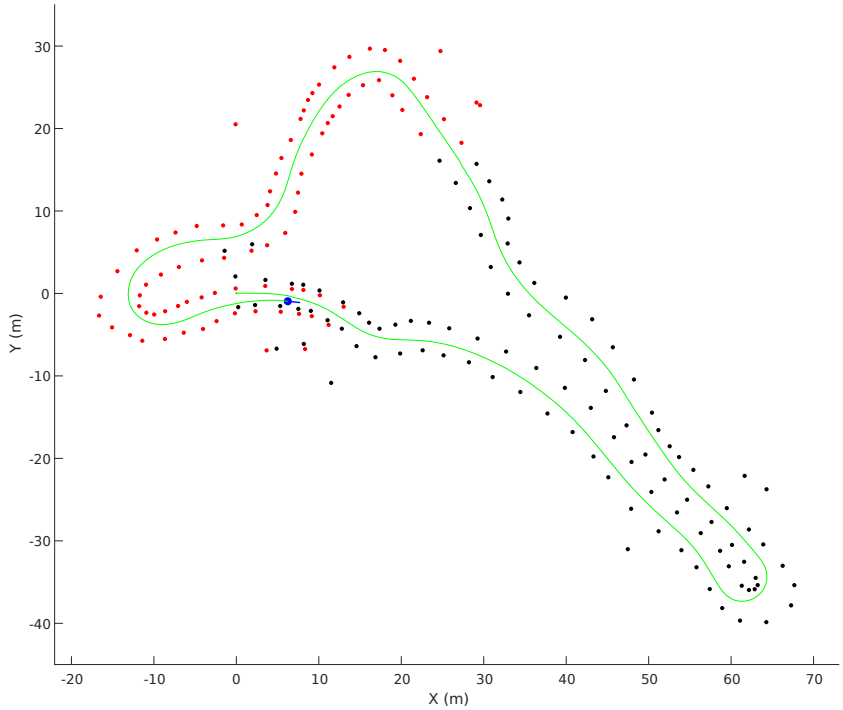

Figure 5.2: Map just before loop closure. New cones in red, old cones in black. New and old cones cannot be associated automatically.

matched. For this to occur drift of roughly 2-5m is needed which is possible for large tracks. Incorrect loop closure results in catastrophic failure of SLAM and needs to be avoided to create a reliable system.

To solve this issue the SLAM algorithm is prevented from associating detected cones with mapped cones that have not been seen for a certain travel distance, shown in Figure 5.2. Instead, manual loop closure is performed after the car reaches the end of the lap and detects the large orange cones which can be uniquely identified. After matching these, all other unmatched cones are rechecked in a sweep that propagates back through the lap. This prevents there ever being a large drift error when cones are matched and reduces the possibility of incorrect loop closure.

Landmarks are combined by applying a measurement that the two cones have no distance between them. This measurement is applied with no noise. After this measurement is performed the two landmarks have identical states and contributions to the covariance matrix. This means that one of these landmarks can be deleted from the Kalman filter with no loss of information. An example of a track map after this routine is run is shown in Figure 5.3, this can be compared to Figure 5.2 that shows the same map before loop closure.

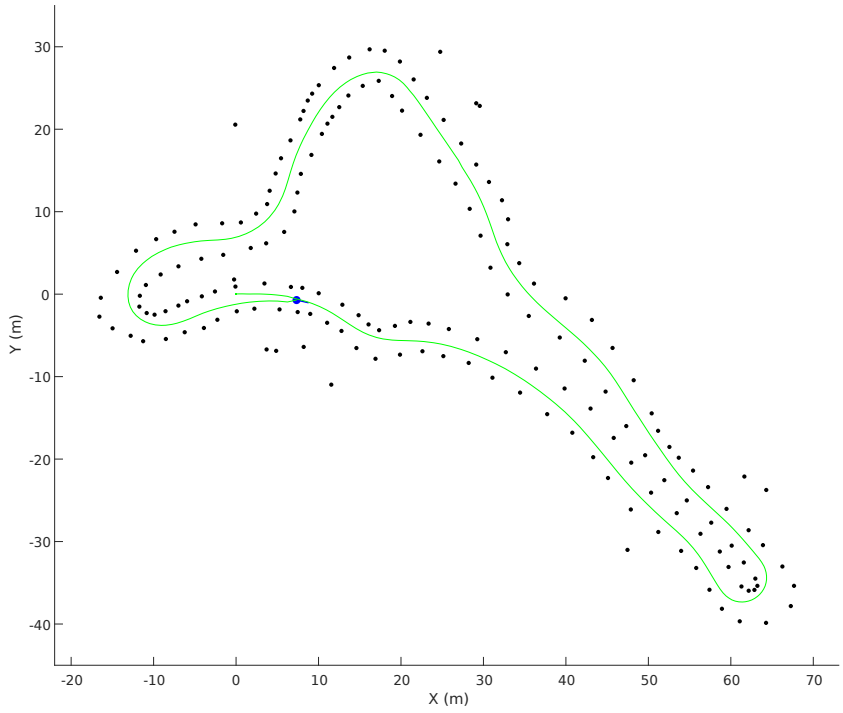

Figure 5.3: Map just after loop closure. Loop closure has been propagated back through the lap to combine new and old cones.

5.3.3 Preliminary Landmark List

A problem that was identified in test data both from another Formula Student team, AMZ [23] and our own testing is the presence of false positives in the detection of cones. These spu-rious detections were caused by foreign objects near the track. This might be long grass or bushes. The incorrectly detected cones cause issues with SLAM as they both increase the size of the state resulting in lower performance and cause inaccuracy as they are often not consistently detected resulting in incorrect information about the movement of the vehicle. Hopefully this will not be an issue at competition as there will be much more open space between the track and other objects that could be spuriously detected as cones.

However, as a precaution against distracting objects near the track a preliminary landmark list has been created within the SLAM algorithm. This means that the first time a cone candidate is detected by the LiDAR it is not immediately added to the main Kalman filter but rather added to a separate list. Only if these cones are detected again in the successive LiDAR sweep are they added to the main filter. This serves as an easy way to filter out false-positives in the cone detection. When comparing Figure 5.5 to Figure 5.4 it is clear that the preliminary landmark list dramatically reduces the number of false positives captured in the Kalman filter state which results in better performance and more accurate mapping. These figures are shown just before loop closure occurs and the amount of drift has been significantly reduced in Figure 5.5 where the preliminary landmark list has been enabled.

5.3.4 Optimisation of Implementation

The standard formulation of the update step for the extended kalman filter is shown [24]

Innovation	$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k\|k-1})$
Residual covariance	$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$
Kalman gain	$\boldsymbol{K}_k = \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$
Updated state estimate	$\hat{\boldsymbol{x}}_{k\|k} = \hat{\boldsymbol{x}}_{k\|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$
Updated covariance estimate	$\boldsymbol{P}_{k\|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k\|k-1}$

While this formulation provides a compact representation it can be rearranged to be computationally more efficient by optimising the order of matrix operations in updating the covariance estimate. This change is shown

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \tag{5.6}$$

The significant speed-up enabled by these changes is demonstrated in Table 5.3 showing a before and after comparison in MATLAB.

This formulation can be further improved by not explicitly calculating the Kalman gain and instead moving this calculation to being in-place. This allows for the use of standard matrix decompositions and linear solve routines that are more efficient and numerically stable than explicit matrix inverses.

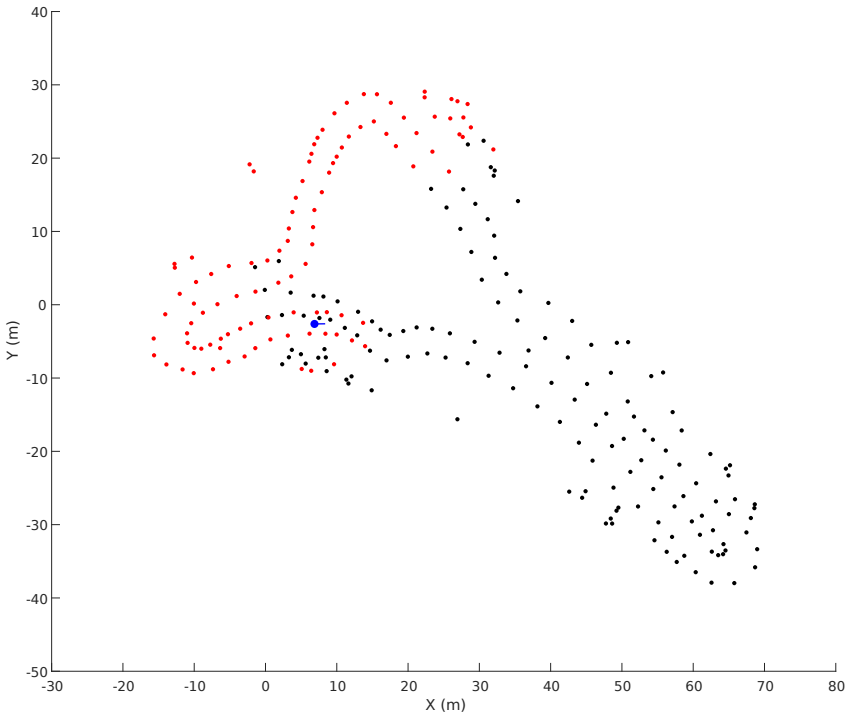

Figure 5.4: Map without Preliminary Landmark List enabled. There are 246 landmarks.

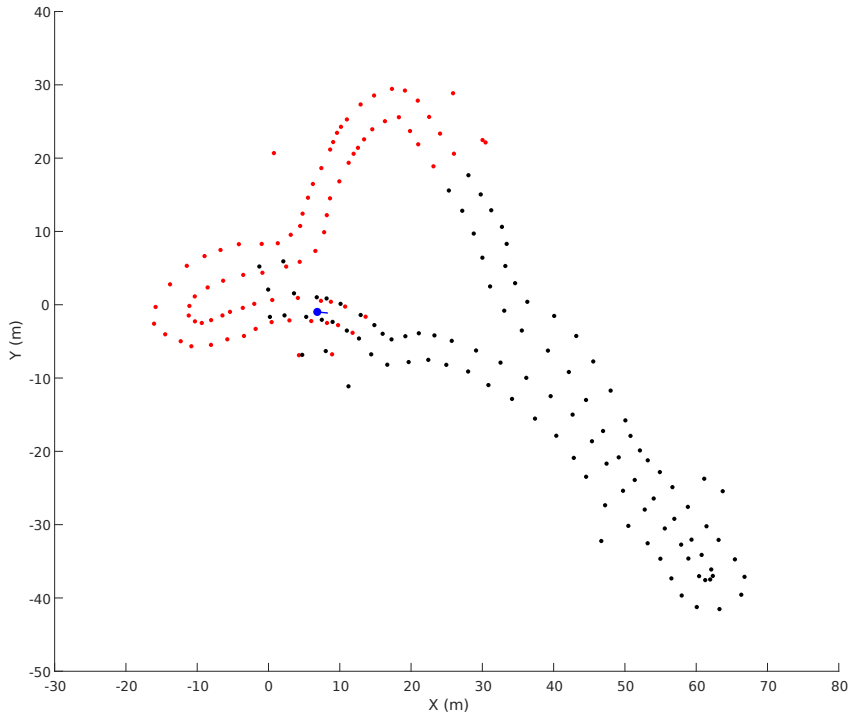

Figure 5.5: Map with Preliminary Landmark List enabled. There are 190 landmarks.

	Original	Improved
Covariance update time	7.978s	1.694s
Total time	16.18s	9.904s

Table 5.3: Performance improvements gained in MATLAB by formulating equations as per Equation 5.6 (compute time for 90 seconds of data)

As the $\boldsymbol{S}$ matrix is positive definite the Cholesky decomposition (LLT) can be used. To improve stability the LDLT decomposition was used instead which maintains comparable performance to the LLT decomposition [25].

Similarly we now focus on the standard predict step covariance estimate.

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k \tag{5.7}$$

The SLAM state vector has two parts; the pose, and the set of landmark locations within the map. The predict step, using the vehicle model only propagates the pose and leaves the landmark states unchanged. This allows Equation 5.7 to be computed in a block form that reduces the computational complexity in the number of landmarks from N^3 to linear [26]. This formulation is shown in Equation 5.8

$$\boldsymbol{P}_{k|k-1} = \begin{bmatrix} \boldsymbol{F}_k \boldsymbol{P}_{xx} \boldsymbol{F}_k^\top + \boldsymbol{Q} & \boldsymbol{F}_k \boldsymbol{P}_{xm} \\ \boldsymbol{P}_{xm}^\top \boldsymbol{F}_k^\top & \boldsymbol{P}_{mm} \end{bmatrix} \tag{5.8}$$

where

$$\boldsymbol{P}_{k-1|k-1} = \begin{bmatrix} \boldsymbol{P}_{xx} & \boldsymbol{P}_{xm} \\ \boldsymbol{P}_{xm}^\top & \boldsymbol{P}_{mm} \end{bmatrix} \tag{5.9}$$

5.3.5 Numerical Stability

The Kalman filter and particularly non-linear extensions of the Kalman filter (such as the EKF) are known to exhibit divergence caused by numerical instability [27, 28].

Our initial implementation in C++ was causing undefined behaviour that was determined to be due to numerical instability issues in calculating the inverse of the $\boldsymbol{S}$ matrix, see Table 5.2. This effect is carried through into computing the updated covariance matrix and results in a non-positive-definite covariance matrix.

When initially porting code from MATLAB to C++ single precision floating point numbers were chosen to represent all values within the EKF state and covariance. This was chosen as it leads to approximately a doubling of performance for operations which use vectorised instructions. This was determined to be one of the significant factors contributing to instability. However, after changing to

double precision floats some stability issues still remained. This was of particular note because a very similar implementation was stable in MATLAB.

Another way to minimise this problem is to calculate the covariance update in the Joseph Form [29]. This guarantees that the covariance matrix will remain positive-definite.

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\top + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k^\top \tag{5.10}$$

The downside of using this formulation is increased computational complexity. In addition, this is not solving the underlying issue but simply hiding it. For these reasons this form is not used in our implementation.

Based on testing it was concluded that two steps needed to be taken to prevent numerical stability issues. The first, outlined above, was simply to use double precision floating point numbers instead of single precision. The second is more application specific and requires ensuring that the measurement noise is not too small as this can lead to an ill-conditioned residual covariance matrix. By making these changes numerical instabilities issues have been alleviated.

5.3.6 Linear Algebra Libraries

To determine the most efficient method of implementing the extended Kalman filter in C++ for the compute hardware to be used on the final autonomous vehicle, a series of tests were performed comparing a range of linear algebra libraries. The tested libraries were Eigen [30], Eigen with Intel MKL [31], ArrayFire [32], and TooN [33].

	Eigen	Eigen MKL	Arrayfire	TooN
Single precision floating point matrix multiply 600x600	873μs	757μs	1263μs	88934μs

Table 5.4: Comparison of computation speed for selected libraries

For the test results shown in Table 5.4 a 600x600 matrix was used as this is approximately the maximum sized matrix that will be experienced during the driverless TrackDrive event. The track is up to 500m long with cones on both sides spaced at least every 5m. This corresponds to minimum 200 cones on a 500m track and considering the use of additional cones around corners it is estimated that a 500m track would use roughly 300 cones and this corresponds to a state with on the order of 600 elements (2 DoF for cones) and hence a covariance matrix of size 600x600.

The most important factor contributing to the performance of each of these libraries is its use of vectorized instructions and multithreading support. Eigen has explicit support for vectorisation and can use openMP to multithread matrix operations. The use of MKL expands upon this by using routines optimised

specifically for the Intel architecture that the CPU in our main compute node uses. TooN does not use many of these features and is instead optimised for very low overhead on small fixed sized matrices. This makes it very suitable, for example, for transformations on 6 DoF poses but is not suitable for our use case with dynamically sized matrices up to approximately 600x600. ArrayFire on the other hand is optimised for multithreading and large matrix operations but is primarily targeted towards use on GPUs. This opens an avenue of potential further optimisation in the future by performing SLAM computations on a GPU.

In addition to these performance considerations Eigen is the most feature rich and is the most widely supported of the compared libraries and so has excellent documentation both of the provided functionality but also supporting theory, implementation details and optimisation guides.

Following this testing and research Eigen was chosen as the linear algebra library used for implementing the Extended Kalman Filter SLAM. Intel MKL will be used in conjunction which provides additional performance with no extra burden on the programmer.

Chapter 6

Conclusions and Future Work

6.1 Concluding Remarks

This project has developed the LiDAR system for an autonomous race car and hence provided the framework for an integrated perception system. The full LiDAR pipeline and supporting hardware has been developed. The LiDAR system provides accurate and low-latency simultaneous localisation and mapping of the vehicle under racing conditions. The SLAM framework developed here will be built upon with measurements from the other perception sensors to form the complete perception system for the car. The progress made in this project contributes to Monash Motorsport being in an excellent position going forwards to continue the development of its first driverless race car.

6.2 Future Work

6.2.1 Sensor Fusion

The next step in developing a full perception system for the autonomous vehicle is sensor fusion. Combining the LiDAR sensor with stereo cameras, GPS/INS and odometry from the car will present great opportunities to increase the accuracy of the system and gain redundancy. Each of these sensors has its own set of strengths and weaknesses and combining them to create a single map and pose of the car allows the sensors to augment each other to improve the end result.

Firstly, focusing on sensors that can perceive the cones and hence the track. The LiDAR sensor has strengths in its wide horizontal field of view, very low sensor latency and highly accurate depth perception. However, it is limited in range and cannot perceive the colour of the cones. In these aspects a stereo camera system is of clear benefit. Stereo cameras provide excellent range and can detect

cone colour but are limited by latency and have a lesser field of view. We aim to combine these sensors in a common Extended Kalman Filter SLAM system.

In addition to the key sensors that can detect cones, the system can be augmented by GPS, INS (specifically accelerometers, gyroscopes and a magnetometer) and odometry from the car such as wheel speed sensors. These sensors, while unable to detect cones, operate at a much greater frequency than the cameras or LiDAR and will enable the system to always have a current estimate of the vehicle pose with very little latency, which is important in supporting control of the vehicle.

The major difficulty in combining each of these sensors is the varying sensor latency. This being the time from when the measurement occurs to when it can be used in sensor fusion after all required filtering has been applied, for example, cone detection algorithms for LiDAR and cameras that require substantial computation. Measurements must be applied to the filter in the correct order and substantial differences in latency will result in the order of measurements arriving at the SLAM function not being the same order as the measurements made of the environment. Somehow these out of order measurements need to be untangled and applied in the correct order. There are two fairly straightforward ways of dealing with this challenge.

Firstly, and most simply, the LiDAR and camera detections can be delayed to the slowest of the two. This can simply allow the measurements to be applied in the correct order but increases the overall latency of the system. For example, if the LiDAR has a latency of 10ms and the cameras 100ms the current estimate of where the car is relative to the cones will always be delayed by at least 100ms, the maximum of the two latencies. In order to provide an up-to-date estimate to path planning and vehicle control these 100ms can be filled in by simple integration of IMU and odometry data. While integration is not reliable for long periods of time due to drift, these sensors are sufficient for short periods of time as in this use case. Because multiple of these high-frequency, low-latency sensors will be available, an Extended Kalman Filter can be used to estimate the vehicle movement during this period. This EKF will only have to be computed for the vehicle pose which is significantly less computationally intensive than for the full state which includes many cone locations. When new measurements arrive we can easily roll back, apply the measurements and propagate forwards again.

The other option is to, for each measurement that arrives, roll back to the corresponding point in time, apply the measurement and then propagate forwards. The key difference to the previous method is this would involve undoing and redoing cone measurements rather than just measurements of vehicle pose. These measurements from the LiDAR and cameras are much more computationally intensive to apply as it requires updating the full Kalman filter state including that of all the landmarks rather than just the vehicle pose. Following the example given above of 10ms latency for the LiDAR and 100ms for the cameras, this solution would allow the estimated latency after applying a LiDAR measurement to be only 10ms rather than 100ms. However, this would require much more com-putational power to apply cone measurements multiple times and more memory

to store multiple full state covariance matrices to allow for time to be undone to apply delayed measurements.

Upon considering these two options it is evident that the first option of delaying cone measurements to that of the sensor with the greatest latency is the best solution to begin with and the second option can be developed if the increased latency is causing issues with vehicle control. Having high frequency, low latency sensors such as an IMU and wheel speed sensors means that the latency should be effectively hidden from path planning and vehicle control. This is especially true once the whole track is known and delayed cone measurements are not limiting the car's ability to see the upcoming track. The biggest potential issue is while the track is unknown, delaying cone measurements is equivalent to reducing the range at which cones can be detected and track boundaries estimated. However, on this first lap the car will be travelling at low speeds and given the range of our sensors we should have sufficient perception range.

6.2.2 Kalman Filter Tuning

One of the biggest challenges in developing a Kalman filter or extended Kalman filter system is the tuning of measurement and process noise parameters [34]. This is made more difficult by sensors whose noise does not follow a Gaussian distribution such as the GPS sensor. Once all of the sensors have been included in a sensor fusion system, significant work will be required to carefully test and analyse the sensors to tune these parameters and develop an accurate filter system.

6.2.3 Integrated Cone Detection

By closely integrating cone detection between the LiDAR and cameras, better performance can be achieved. As the cameras have greater range, by using existing camera measurements the LiDAR could detect cones outside its normal range. Having a good estimate of the expected location of a cone means that reliable detection could occur when only one laser sweep has intersected the cone, unlike the usual two. The LiDAR could also benefit camera cone detection. Camera cone detection is computationally expensive and having an estimate of cone locations could be used to narrow the search for cones within the camera's field of view.

6.2.4 Improved Data Association Algorithm

Currently data association is performed between measurements and existing landmarks by finding the minimum Euclidean distance error that is below a correspondence threshold. This works quite effectively for our simultaneous localisation and mapping problem because the cones that serve as landmarks are fairly regularly spaced which allows an assumption to be made of a lower bound on the distance between distinct landmarks. This assumption fails, however, for

the largest cones that mark the start/finish of the track. These cones are often placed immediately adjacent and cause the current data association algorithm to fail.

An improved data association algorithm could be designed that performs associations based on a likelihood threshold calculated from the covariance associated with the position of the landmark, the measurement noise and the covariance associated with the position of the vehicle. A value proportional to the probability of the sample within this distribution can be calculated using the Mahalanobis distance [35].

6.2.5 Deeper Driver Integration

A community developed ROS [14] driver exists for the Velodyne family of LiDAR units [15]. This communicates directly with the LiDAR and provides us with the calibrated 3D point cloud. The existence of this driver simplified the development of the LiDAR pipeline. However, the driver does not provide highly accurate timestamps in the published ROS messages. The included time represents the time when the message was sent rather than when the point was measured. This introduces small errors into the system. This could be rectified by modifying the published messages to include the highly accurate timestamps provided by the LiDAR.

References

[1] "Formula student rules 2019," https://www.formulastudent.de/fileadmin/user_upload/all/2019/rules/FS-Rules_2019_V1.1.pdf, 2018, version 1.1.

[2] M. de la Iglesia Valls, H. F. C. Hendrikx, V. J. F. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. Gawel, M. Bürki, and R. Siegwart, "Design of an autonomous racecar: Perception, state estimation and system integration," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2048–2055, 2018.

[3] N. B. Gosala, A. Bühler, M. Prajapat, C. Ehmke, M. Gupta, R. Sivanesan, A. Gawel, M. Pfeiffer, M. Bürki, I. Sa, R. Dubé, and R. Siegwart, "Redundant Perception and State Estimation for Reliable Autonomous Racing," *ArXiv e-prints*, 2018.

[4] M. Zeilinger, R. Hauk, M. Bader, and A. Hofmann, "Design of an autonomous race car for the formula student driverless (FSD)," in *Proceedings of the OAGM & ARW Joint Workshop, At Vienna*, 2017.

[5] T. H. Drage, J. Kalinowski, and T. Bräunl, "Development of an autonomous formula SAE car with laser scanner and GPS," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2652–2657, 2014, 19th IFAC World Congress.

[6] M. Poole, "Improved localization in uwa rev autonomous driving sae vehicle through imu, gps, wheel encoders and extended kalman filter," Master's thesis, Univeristy of Western Australia, 2017.

[7] T. D. Frysa, "Perception for an autonomous racecar," Master's thesis, NTNU, 2018.

[8] J. Cressell and I. Törnberg, "A combined approach for object recognition and localisation for an autonomous racecar," Master's thesis, KTH Royal Institute of Technology, 2018.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part i the essential algorithms," *IEEE Robotics and Automation Magazine*, June 2006.

[10] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, March 1987, pp. 850–850.

[11] M. Montemerlo and S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Springer Publishing Company, Incorporated, 2010.

[12] Z. Kurt-Yavuz and S. Yavuz, "A comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM algorithms," in *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, 2012, pp. 37–43.

[13] T. D. Larsen, N. A. Andersen, O. Ravn, and N. K. Poulsen, "Incorporation of time delayed measurements in a discrete-time kalman filter," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, Dec 1998, pp. 3972–3977 vol.4.

[14] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[15] J. O'Quin and J. Whitley, "ROS Velodyne," 2018. [Online]. Available: http://wiki.ros.org/velodyne

[16] "Velodyne Puck VLP-16." [Online]. Available: https://velodynelidar.com/vlp-16.html

[17] "Velodyne Puck Hi-Res." [Online]. Available: https://velodynelidar.com/vlp-16-hi-res.html

[18] "Formula student rules 2018," https://www.formulastudent.de/uploads/media/FS-Rules_2018_V1.1.pdf, 2017, version 1.1.

[19] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer-Verlag, 2007.

[20] P. S. Maybeck, *Stochastic models, estimation and control*. Academic Press, 1979.

[21] R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation [lecture notes]," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128–132, Sept 2012.

[22] M. Calonder, "EKF SLAM vs . FastSLAM a comparison," 2007.

[23] "Formula Student Driverless resources," 2018. [Online]. Available: https://github.com/AMZ-Driverless/fsd-resources

[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[25] "Benchmark of dense decompositions," 2018. [Online]. Available: https://eigen.tuxfamily.org/dox/group__DenseDecompositionBenchmark.html

[26] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part ii state of the art," *IEEE Robotics and Automation Magazine*, September 2006.

[27] T. Karvonen, "Stability of linear and non-linear Kalman filters," Master's thesis, University of Helsinki, 2014.

[28] M. Verhaegen and P. V. Dooren, "Numerical aspects of different Kalman filter implementations," *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 907–917, October 1986.

[29] R. Bucy and P. Joseph, *Filtering for stochastic processes with applications to guidance*. Interscience Publishers, 1968.

[30] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[31] "Using Intel MKL from Eigen," 2018. [Online]. Available: https://eigen.tuxfamily.org/dox/TopicUsingIntelMKL.html

[32] P. Yalamanchili, U. Arshad, Z. Mohammed, P. Garigipati, P. Entschev, B. Kloppenborg, J. Malcolm, and J. Melonakos, "ArrayFire - A high performance software library for parallel computing with an easy-to-use API," Atlanta, 2015. [Online]. Available: https://github.com/arrayfire/arrayfire

[33] E. Rosten, "TooN: Tom's object-oriented numerics library." [Online]. Available: https://www.edwardrosten.com/cvd/toon.html

[34] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of Kalman filters," in *in Proceedings of Robotics: Science and Systems*, 2005.

[35] P. C. Mahalanobis, "On the generalised distance in statistics," in *Proceedings National Institute of Science, India*, vol. 2, no. 1, Apr. 1936, pp. 49–55.