DESIGN AND DEPLOYMENT OF A SIMULTANEOUS LOCALISATION AND MAPPING SYSTEM FOR GNSS DEPRIVED LOCATIONS

Harley Pritchard

B. Eng. Mechatronic Engineering

Project Report

Department of Mechanical Engineering

Curtin University

2019

INTRODUCTORY LETTER

Unit 10, 7 Ventnor Street,
Scarborough,
WA, 6019,

26/11/2019

The Head Department of Mechanical Engineering,
Curtin University,
Kent Street,
Bentley,
WA 6102

I submit this report entitled “Design and deployment of a Simultaneous Localisation and Mapping system for GNSS deprived locations”, based on Mechatronic Engineering Research Project One and Two, undertaken by me as part-requirement for the degree of B.Eng. in Mechatronic Engineering.

Yours faithfully

Harley Pritchard

PREFACE

Some material contained in this thesis has been taken from the progress report submitted for Mechatronic Engineering Research Project 1, as Mechatronic Engineering Research Project 2 is a continuation of this project. Material has been used in: Abstract, Chapter 1 - Introduction, Chapter 2 - Background, Chapter 4 – Hardware Design, and Chapter 5 – Software Design.

ACKNOWLEDGEMENTS

There are many people who have supported this project and assisted throughout the past year. The author would like to thank their supervisor, Jonathan Paxman, for all his assistance and support throughout this project.

The author would also like to thank Robert Reid for his valuable knowledge and support when encountering difficulties with ROS and Cartographer and to Woodside for lending vital components for the duration of this project.

The author wishes to thank Hamish Thava and Amir Shahbazi of School of Civil and Mechanical Engineering, Curtin University for their technical support and assistance in the construction of this project.

The author would also like to thank Tony Snow and David Belton of the Department of Spatial Science, Curtin University for allowing access to survey data; John Blylevens for allowing access the Leighton Battery Tunnels.

Thanks to Srdjan Mirkovic, Tristan McKenzie and Vajra Kruger for their help in collecting data for this project and other friends and family that provided transport and support. Finally, my partner Louisa Godwin, without your support and patience this project would not be possible.

ABSTRACT

Robots operating in uncertain or unseen environments require the ability to solve two problems; what does the environment look like and where in the environment is the system located. This project is based around the design of a portable 3D Simultaneous Localisation and Mapping (SLAM) device to deploy and map indoor and outdoor areas of Curtin University. The handheld device design consists of a Velodyne PUCK LiDAR and Intel RealSense T265 tracking camera as sensor inputs and processed using Google's Cartographer on a Nvidia Xavier. To test the performance of the device, the system was deployed within Curtin University and the Leighton Battery Tunnels in Mosman Park, WA. The resultant point clouds were compared with survey data collected by the Spatial Science department and found to produce a mean point cloud to point cloud distance at Leighton Battery Tunnels of 0.17m, in significantly less time than traditional surveying practice. The completed design has been successful in creating a portable SLAM system for applications of 3D mapping or future autonomous navigation projects. The system was successful in completing SLAM at both Curtin University and Leighton Battery Tunnels in both real-time and in post-processing.

NOMENCLATURE

3D – 3-Dimensional

BE – Bayesian Estimation

EKF - Extend Kalman Filtering

IMU – Inertial Measurement Unit

ICP – Iterative Closest Point

KF – Kalman Filter

LiDAR – Light Detection and Ranging

GNSS – Global Navigation Satellite System

MAP - Maximum a Posteriori

OSRF – Open Source Robotics Foundation

PF – Particle Filtering

ROS – Robotics Operating System

SLAM – Simultaneous Localisation and Mapping

VO -Visual Odometry

VPU – Visual Processing Unit

USB – Universal Serial Bus

Table of Contents

INTRODUCTORY LETTER ..... **II**

PREFACE ..... **IV**

ACKNOWLEDGEMENTS ..... **VI**

ABSTRACT ..... **VIII**

NOMENCLATURE ..... **X**

TABLE OF FIGURES ..... **XVI**

LIST OF TABLES ..... **XX**

CHAPTER 1: INTRODUCTION **1**

1.1 Project definition 1

1.2 Aims and Objectives 2

1.2 Scope of work 2

1.5 Applications and Future Development 3

CHAPTER 2: BACKGROUND **5**

2.1 Theory and motivation 5

2.2 State of the Art 5

2.3 Robotics Operating System Overview 6

2.4 Mapping 8

2.4.1 Surveying *8*

2.4.2 Robotic Mapping *9*

2.5 Localisation 10

2.6 Simultaneous Localisation and Mapping 10

2.6.1 Design of SLAM *11*

2.6.2 Techniques to solve SLAM *11*

2.6.3 Results from SLAM *12*

2.7 Distance sensors 13

2.7.1 Laser scanning *13*

2.7.2 Sonic sensors *14*

2.7.3 Point clouds *14*

2.8 Odometry 14

2.8.1 Wheel Odometry *14*

2.8.2 Inertial Measurement Unit *15*
2.8.3 Visual Odometry *15*
2.8.4 GNSS *15*
2.8.5 Intel RealSense T265 ..... *16*
2.9 Mapping Comparison Techniques 16
2.10 SLAM Packages 17
2.10.1 Gmapping *17*
2.10.2 Hector SLAM *18*
2.10.3 Cartographer *19*
2.10.4 Karto-SLAM *20*
2.10.5 CoreSLAM *20*
2.10.6 LOAM *20*
2.10.7 Industry SLAM Packages *21*
2.11 Coordinate Transformations 21
CHAPTER 3: DESIGN REQUIREMENTS 22
3.1 Hardware Requirements 22
3.1.1 Processing Requirements *22*
3.1.2 Physical requirements *23*
3.1.3 Safety Requirements *25*
3.1.4 Power requirements *26*
3.2 Software Requirements 27
CHAPTER 4: HARDWARE DESIGN 28
4.1 Hardware Selection 28
4.1.1 RealSense T265 ..... *29*
4.1.2 Velodyne PUCK (VLP-16) *29*
4.1.3 NVIDIA Jetson-XAVIER *29*
4.1.4 Power Supply *30*
4.1.5 Additional *30*
4.2 Sketches 31
4.3 CAD Design 32
4.4 3D Printing 33
CHAPTER 5: SOFTWARE DESIGN 34
5.1 Coordinate Transformations 35

5.1 Transformation Tree 38
5.3 User Interface 40
5.4 Cartographer 41
CHAPTER 6: RESULTS AND DISCUSSIONS 42
6.1 Data Collection 42
6.1.1 Architecture Building – Curtin University 43
6.1.2 Leighton Battery Tunnels 43
6.1.3 Henderson Court – Curtin University 44
6.2 Scans and Maps 44
6.2.1 Curtin Architecture Building 45
6.2.2 Leighton Tunnels 46
6.2.3 Henderson Court 48
6.3 Comparison to truth 49
6.3.1 Architecture Building 50
6.3.2 Leighton Battery Tunnels 54
6.3.3 Henderson Court 55
6.4 Design 57
CHAPTER 8: CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK 64
REFERENCES 66
APPENDIX A 70
APPENDIX B- SLAM DEVICE USER GUIDE 72

TABLE OF FIGURES

Figure 1: Scope of work 2

Figure 2: ROS overview showing how the nodes publish and subscribe to topics (Yun 2015) 7

Figure 3: Traditional surveying set up (MacKinnon 2019) 9

Figure 4: PUCK Sensor Coordinate System (Velodynelidar.com, 2019) 13

Figure 5: RealSense T265 system overview (Intelrealsense.com 2019) 16

Figure 6: HectorSLAM occupancy grid example (Neves Dos Santos, 2014) 18

Figure 7: Cartographer system overview (The Cartographer Authors, 2018) 19

Figure 8: Cartographer occupancy grid with hits crossed out and misses as shaded (Hess et al. 2016) 20

Figure 9: Example of Euler angles (Sprague 2016) 21

Figure 10: Overview of System Diagram for deployable system 28

Figure 11: Initial sketches of design 31

Figure 12: CAD design 32

Figure 13: Results of 3D printed mount design 33

Figure 14: System overview 34

Figure 15: SLAM system overview (The Cartographer Authors, 2018) 35

Figure 16: RealSense T265 Sensor Coordinate System 36

Figure 17: PUCK coordinate frame described in manual (velodynelidar.com 2019) 37

Figure 18: Frame of reference from the velodyne_driver node (velodynelidar.com 2019) 37

Figure 19: Transformations between the system frames 37

Figure 20: Example of Cartographer required transforms tree (The Cartographer Authors, 2018) 38

Figure 21: Tf tree of system while recording.....38
Figure 22: Tf tree broadcasted in post processing by the cartographer_offline_node....39
Figure 23: iPad remotely connected to system via NoMachine.....40
Figure 24: Starting the system at the Leighton Battery Tunnels entrance. 42
Figure 25: Scanning at the Leighton Battery Tunnels 44
Figure 26: Architecture building map produced (above).....45
Figure 27: Architecture building map produced (side).....46
Figure 28: Architecture building map produced (Front).....46
Figure 29: Leighton tunnels map produced (Above) 46
Figure 30: Leighton tunnels map produced (Front) 47
Figure 31: Leighton tunnels map produced (Side).....47
Figure 32: Henderson Court map produced (Above).....48
Figure 33: Henderson Court map produced (Side) 49
Figure 34: Henderson Court map produced (Side) 49
Figure 35: Slice of Architecture building with heatmap of difference of points 50
Figure 36: Gaussian histogram of distance of points for slice of architecture building .50
Figure 37: Architecture ground truth (grey) against result (colour) 51
Figure 38: Architecture ground truth (grey) against result (colour) 52
Figure 39: Architecture result from difference of point clouds 53
Figure 40: Architecture results distribution of difference of points.....53
Figure 41: Leighton Battery Tunnels result from difference of points against ground truth 54
Figure 42: Leighton Battery Tunnels distribution of difference of points against ground truth 54
Figure 43: Henderson heatmap of distance between point clouds at sliced level.....55

Figure 44: Histogram of distance between point clouds 56
Figure 45: Henderson trees showing discrepancy of tree radius 56
Figure 46: Final design with components 57

LIST OF TABLES

Table 1: Project aims....2

Table 2: Processing Requirements....22

Table 3: Physical requirements....23

Table 4: Safety requirements....25

Table 5: Power requirements....26

Table 6: Software requirements....27

Table 7: Bill of materials....28

Table 8: Project aims achieved....58

Table 9: Explanation of requirements achieved....59

Table 10: Summary of project requirements achieved....62

CHAPTER 1: INTRODUCTION

1.1 Project definition

This project aims to design a system that is able to apply the Simultaneous localisation and mapping (SLAM) problem and map areas without Global Navigation Satellite System (GNSS). SLAM refers to the problem of trying to solve two problems simultaneously; what does the environment of the robot look like (mapping) and where is the robot in this environment (localisation)? SLAM attempts to solve mapping and localisation by slowly building up solutions to both by collecting data and applying probabilistic estimates. In many designs to help solve the SLAM problem, GNSS data is used to locate the position of the system. However, there are many applications where localisation and mapping need to be performed in environments where GNSS data is restricted.

The novelty of this project is the RealSense T265 tracking camera, which will be used as the odometry for the system. Odometry is tracking the movements over time to provide an estimate of the position of a device in respect to an origin. This project designs a payload that will apply the pose estimated by the T265 and distance measurements from a Light Detection and Ranging (LiDAR) sensor into the Robotics Operating System (ROS) SLAM package Cartographer to map indoor and outdoor locations where typically would require a solution without GNSS. An evaluation of the resulting point clouds has been performed by measuring the distance of the points against a dataset considered to be accurate.

The selected testing environments for this project include the Architecture building and Henderson Court at Curtin University, and the Leighton Battery Tunnels in Mosman Park, WA. These three environments have been selected as they each present different challenges for the system and all have previously been surveyed with high precision equipment that can be used as a ground truth reference for the system.

1.2 Aims and Objectives

The main aims for this project are:

Table 1: Project aims

ID	Description
A.1	Research feasibility of a portable SLAM system for use at Curtin University
A.2	Research the available SLAM packages available with ROS
A.3	Design of a portable SLAM system
A.4	Construction of portable SLAM system
A.5	Perform SLAM with system in GNSS deprived areas
A.6	Compare accuracy of created map and data against ground truth
A.7	Provide recommendations for future SLAM systems at Curtin University

1.2 Scope of work

The work completed was broken into three main sections: Hardware Design, Software Design, Implementation and Validation. This breakdown can be seen in the Figure 1 below.

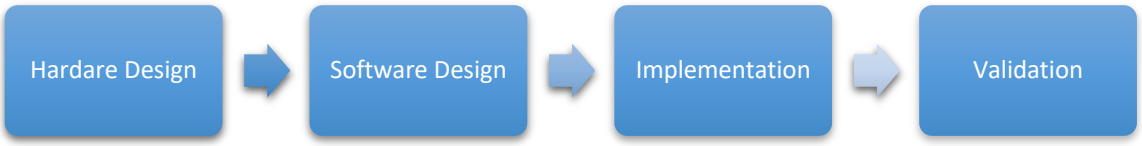


Figure 1: Scope of work

1.5 Applications and Future Development

This project has a wide range of applications within the robotics and 3D mapping fields. The principle application for this project has been to create a base for future development and applications for SLAM at Curtin University. The design presented contains both real-time and offline SLAM capabilities which can be implemented into autonomous navigation systems as well as real-time mapping of environments. Other alternate applications for this project include 3D mapping in mining or subsea environments where priori or GNSS data is not available.

CHAPTER 2: BACKGROUND

2.1 Theory and motivation

Simultaneous localisation and mapping (SLAM) is a particularly complex problem of trying to localise a robot's pose (position and orientation) with respect to its environment, whilst also trying to construct a model (the map) which best describe the features of its surroundings. It is often a series of computations and algorithms processing sensor data to actively solve the problem of building and updating a reference map whilst locating itself within the world. In robotics a priori is defined as the location of a set of landmarks. At the start of the problem both the surroundings and the position are not known or has no priori. Where priori such as a reference map or GPS is known the SLAM problem may not be required. Localising the position of a device is useful when exploring and navigating unknown environments where no prior map is available. Example areas where this is applicable is with autonomous vehicles navigating through a tunnel or dense high-rise buildings; a package delivery robot following a path underneath heavy foliage. Given recent technological advancements, development of the SLAM problem is at the forefront of robotics research as the industry develops more advanced technologies in sensors and processing power allowing for increasing use of robots to perform mapping missions in a wide array of fields such as vacuuming living room floors to mapping of hazardous areas such as underground mines and space exploration (Ueland 2017, Aloise et al 2019).

Traditional surveying practice for producing high-quality 3D point maps is often a cumbersome and requires expensive devices (>$50,000) (Allenprecision.com 2019). Often this practice requires a person carrying large pieces of specialised equipment into tight spaces and requires static targets that provide reference to the generated results, which can take days to perform. This project aims to provide a lower cost and faster alternative for producing these maps.

2.2 State of the Art

There have been several documented projects combining LiDAR and SLAM for mapping without GNNS data. Below are examples which closely relate to this project.

In Le Cras (2013) documented the deployment of a multi-sensor, large scale autonomous mapping system at Curtin University using LiDAR, multi camera and Inertial Measurement Unit (IMU) configuration. His aim was to create a device that was able to map out large scale underground mines. The system he created was a wheeled robotic device that was quite bulky, slow and required extensive post processing. However, he produced a highly accurate 3D point cloud of the Architecture building on campus.

There are several commercially available products which contain similar systems to the one proposed in this document. Example of these are ZEB (GeoSLAM 2019) and Hovermap (Emsent 2019). Both are expensive, contain proprietary software and ongoing subscription fees, which make them unrealistic for ongoing University projects. The ZEB system, Figure 6, by GeoSLAM is a moving head equipped with a LiDAR capturing 2D point profiles. The data is then logged onto a hard disk that can be later processed through their web application.

Hovermap, was created by Kaul et al (2015) at CSIRO with a custom SLAM algorithm and sensor setup. The system has a rotating Velodyne puck delivering a 360-degree scan of the environment and is used heavily in the mapping of underground mines where it is considered to be the state of the art. This product has since been commercialised under the license Emsent.

Grehl et al (2016) used a Clearpath Husky robot equipped with two SICK LiDAR devices, Inertial Measurement Units (IMU), wheel odometry and cameras to map the Reiche Zeche educational mine in Germany. The system used ROS libraries to run sensors and record data. The data collected was processed using the Hector SLAM package.

2.3 Robotics Operating System Overview

Robotics Operating System (ROS) is an open source and community driven framework for writing robot software. It is a middle wear operating system that operates on Linux (Ubuntu) and the system is built with modular architecture to provide hardware abstraction, low level device control and message sending amongst a wide variety of other services. It is a collection of tools, libraries and conventions that aim to simplify the task

of creating robots across a wide variety of platforms. (ros.org 2019). It is favourable for its efficiency in handling many jobs from different processes (Li et al. 2018).

Originally ROS was developed by Stanford Artificial Intelligence Laboratory in 2007 and continued at Willow Garage (Di Caro 2017). In 2013, ROS became managed by the Open Source Robotics Foundation (OSRF). Before ROS there was a lack of standards and code reusability within the robotics community. The core components of ROS are the nodes, topics and messages. The nodes are single executables programs that are individually compiled, executed and managed. They can either publish or subscribe to a topic or provide or use a service or action. Topics are a stream of data being sent, in the form of a message. The message will contain the data structures for the message. The ROS framework enables each node to communicate by either publishing or subscribing to topics. This process forms a graph like network of concurrent systems running, as seen below in Figure 2 (Yun 2015).

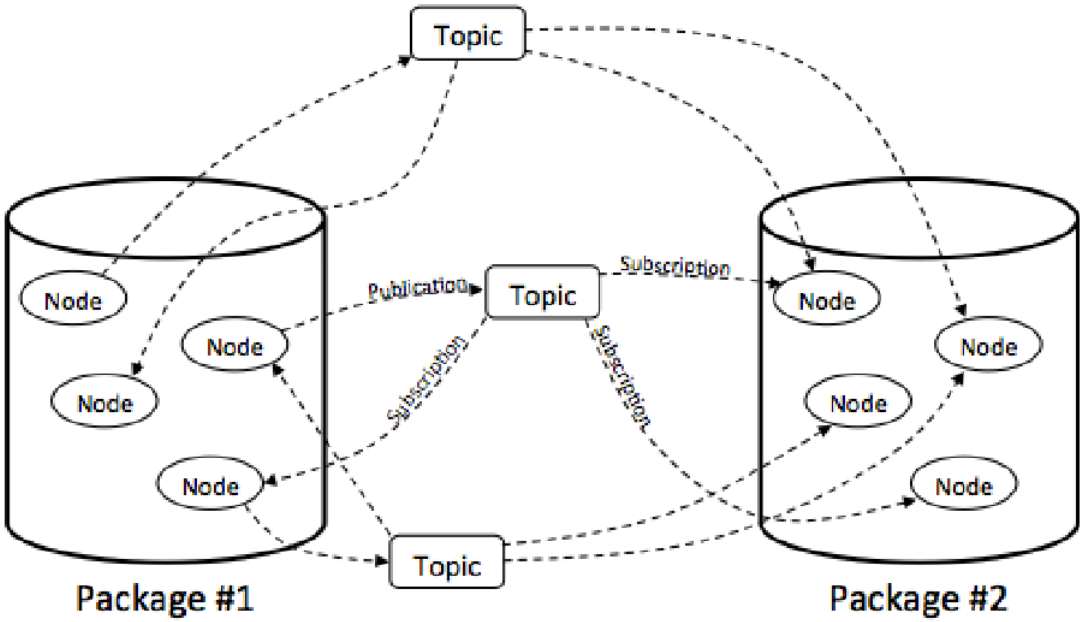


Figure 2: ROS overview showing how the nodes publish and subscribe to topics (Yun 2015)

2.4 Mapping

Mapping can have many definitions depending on the intended application. The definition that is applicable for this project is the process of making maps, and these maps may be used in robotics applications. The map making process is making a graphical representation of the world surrounding the device. The quality and type of mapping is dependent on the purpose, accuracy, range and resolution requirements of the result.

2.4.1 Surveying

Surveying is often required for high-quality results in applications for construction and mining. Traditional surveying practice is often a cumbersome approach to mapping where two people are required to lift heavy equipment and reference markers are stationed throughout the map to measure the accuracy of the network. Figure 3 shows an example of traditional surveying setup (MacKinnon 2019). Surveyors use equipment to perform a closed loop traverse, which enables them to measure the misclosure of the network. This error can be proportionally applied back through the network of points to minimise errors. However, if there are two gross errors cancelling each other, this may not be found.

Figure 3: Traditional surveying set up (MacKinnon 2019)

There are several approaches to surveying an area, the first requires a total-station seen in Figure 3. These stations cost anywhere between $4,000 and $50,000 (Allenprecision.com 2019). They incorporate high-quality lasers, that can measure distances up to 3.5km, and gyroscopes to measure the bearing of the device with 1 second precision (Allenprecision.com 2019). There is now equipment, like the Leica C-10, which incorporates the same principles to create 3D scans of the environment. This can also be traversed through the environment to create detailed 3D maps.

Mobile laser scanning involves mounting a laser scanner to the back of a vehicle and has become more prevalent with the recent advancements in technology. The resulting scans are positioned using odometry, usually GPS.

2.4.2 Robotic Mapping

Robotic mapping uses probabilistic approaches to building spatial models of the environments in which the robots operate. This is achieved by applying algorithms which incrementally build the map, and has the ability to be run in real-time, or require multiple passes through the environment. One of the earliest algorithms to solve this problem is Elves Occupancy grid-mapping algorithm (Elves 1989), which uses a fine

grid pattern to represent occupied and free space in the environment. Robotic mapping has several key challenges in which it faces. A key problem is the measurement noise and errors, as small errors in odometry can cause large errors in the position estimates and resulting map. Errors in odometry are often accumulative and as such increase over time.

2.5 Localisation

Localisation is a time dependant state approximation of the robots pose and velocity. It locates the position of the robot in the map developed in the mapping stage.
Localisation can be used to track the estimated position of the robot as it moves through its environment. IMUs are able to measure the maintained accelerations, velocities and orientations throughout operation. This can then be integrated to find the position and orientation at any given time. Similar to mapping, incorrect measurements may be integrated to produce large errors in the estimated position (Rahul Kala 2016). A map is often needed so that the localisation algorithms can compute the position based on the vision perception of the sensor readings.

2.6 Simultaneous Localisation and Mapping

Mapping requires localisation, and localisation requires mapping. This problem is called Simultaneous Localisation and Mapping (SLAM) problem and was first proposed in 1986 at the IEEE Robotics and Automation Conference in San Francisco when the suggestion of using estimation-theoretic models for robot localisation and mapping was discussed. However, it was not until 1995 when the International Symposium on Robotics Research that the acronym SLAM and the demonstration of convergence was presented. Early adoptions of this used odometry with laser or ultrasonic sensors as perception. There are three different ages of SLAM mapping, the classical age (1986-2004), the algorithmic-analysis age (2004-2015) and from 2015 – present (2019) being the robust perception age (Cedena 2016). During the classical age the main approaches to solving SLAM were based on Extended Kalman Filters (EKF), maximum likelihood estimation and Rao-blackwellised particle filters. Throughout the algorithmic-analysis age, the study of the observability, convergence and consistency were all key in the

roles of SLAM solvers being developed and open source libraries created such as the ones found on ROS.

2.6.1 Design of SLAM

The structural design of a SLAM system includes two main modules: the front-end and the back-end. The front-end abstracts sensor data into models that are suitable for estimation, while the back-end performs extrapolation on the abstracted data produced by the front-end. The back-end uses techniques to solve a maximum a posteriori (MAP) estimation to find a solution to the abstracted data provided by the front end. (Cedena 2016)

2.6.2 Techniques to solve SLAM

Common approaches to solve SLAM problems include Bayesian Estimation (BE), Kalman Filtering (KF), Extended Kalman Filtering (EKF), Particle Filtering (PF), GraphSLAM and Iterative Closest Point (ICP) (Le Cras 2013).

BE is an approach to optimum state estimation which attempts to find the minimum in a cost function. Each step evaluates the current state of the system based on its previous observation. However, it needs to integrate over the entire space collecting all landmarks and pose estimates. As the system may contain millions of landmarks and pose estimates the process becomes very costly.

KF is the representation of BE in a system where it produces estimates of the current state variables along with uncertainties and once the next observation is measured the uncertainties are updated by using a weighted average, favouring estimates with higher certainty. KF is effective as long as the system is linear and the noise follows a Gaussian Distribution (Le Cras 2013).

EKF uses a linear approximation of the system which linearises about an estimate of the current mean and covariance. EKFs were the first solution and are still one of the most popular solutions to non-linear SLAM due to their relatively low computational costs.

PF are another technique for approximating non-linear SLAM problems. They are a non-parametric recursive Bayes filters. The posterior is represented by a set of weighted samples; however, they may not be able to estimate posteriors for highly accurate

observations. This technique requires high computational costs for effective localisation and are subject to failing when subject to large state changes (Stachniss 2012, Le Cras 2013).

GraphSLAM was originally created by Thrum and Montermelo (2006) as a graphical network interpretation solution. It uses vehicle poses and landmarks as nodes and join via motion and measurement arcs, sometimes referred to as edges. GraphSLAM optimises the solution using the nodes as positions and the edges as constraints for the map.

ICP is the registration of point clouds by solving for the minimum difference between two clouds of points. This process requires sufficiently accurate initial pose estimates otherwise the algorithm may solve for the local minimum or completely fail (Le Cras 2013).

2.6.3 Results from SLAM

There are several results that can be computed from SLAM. These include 2D, 2D localisation for 3D mapping and 3D mapping and localisation. The most common implementation of SLAM mapping is the simplified version of a 2D map of a 3D environment. They are often used by wheeled robotics operating on a flat surface for indoor environments. They take a horizontal slice of the environment at a level above floor height.

2D localisation for 3D Mapping approach uses the optimisation of the position from the 2D SLAM solution and maps or overlays the 3D scan data. They often only use three degrees of freedom (X, Y and Yaw).

Bearing only 3D is an implementation of SLAM with bearing approximations to landmarks and features, removing the need for distances. It is popular with SLAM systems that have a single camera. They are not often used in real world examples.

Full SLAM or 3D mapping and localisation computes approximated 3 DOF localisation, measuring and solving for six degrees of freedom. These processes can consume significant processing capacity and often require simplifications to the mapping process. These may include reducing the mapping density, tracking key features or geometrically simplified maps. (Le Cras 2013)

2.7 Distance sensors

2.7.1 Laser scanning

LiDAR stands for Light Detection and Ranging and is the process of measuring distances by pulsing laser light at an object and measuring its reflectance. They are one or two axis scanning lasers to generate 2D or 3D information about its environment from which the reflectance can be used to either measure the distance or determine traits. Figure 4 shows how the data point is calculated via the rotation about the x and y axis of the PUCK LiDAR (Velodynelidar.com 2019). LiDAR used extensively throughout a large range of fields from agriculture, archelogy, video games and autonomous vehicles. Within the field of robotics, robots are able to use this data to navigate its surroundings as their accuracy and robustness make them an excellent for mapping and localisation (Blanco 2019, Cedena 2016, Ueland 2017).

Figure 9-1 VLP-16 Sensor Coordinate System

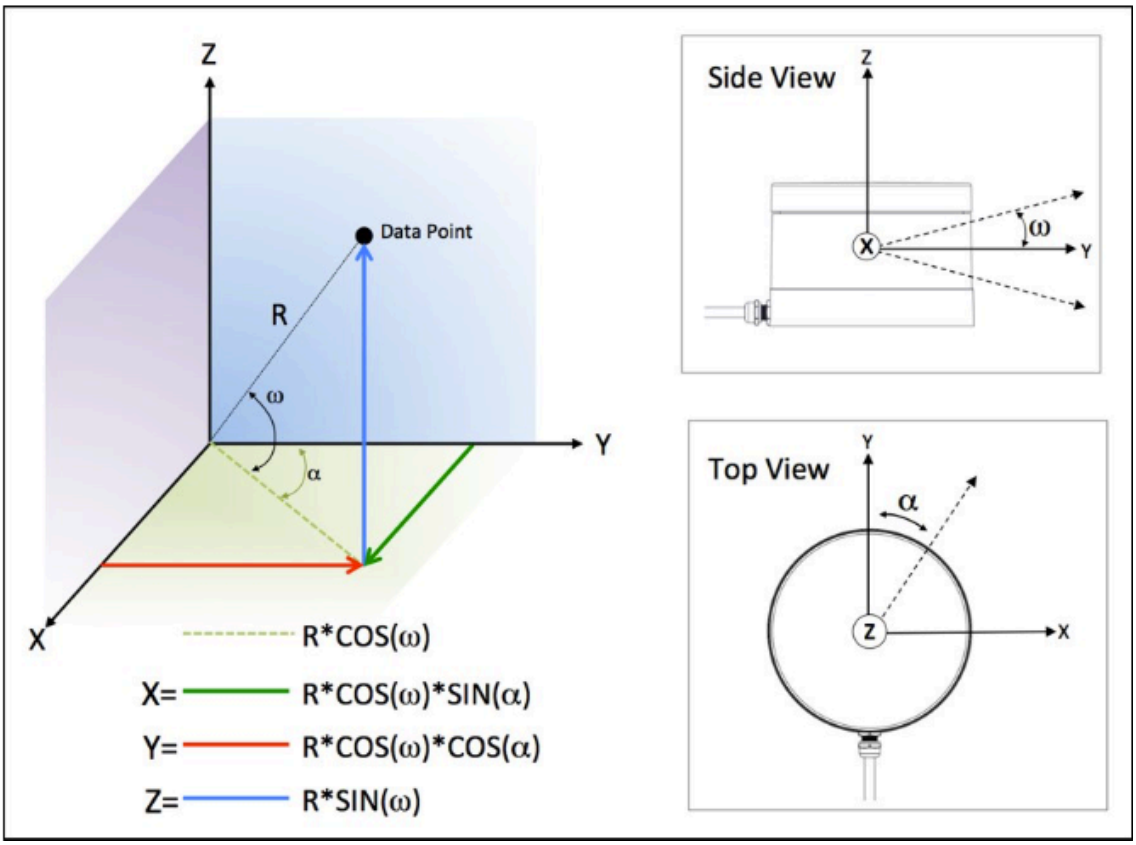


Figure 4: PUCK Sensor Coordinate System (Velodynelidar.com, 2019)

2.7.2 Sonic sensors

Ultrasonic sensors are another approach to collecting distance readings. These sensors produce ultrasonic waveforms and measure the return signal and time taken to receive to determine traits of the environment. This is often beneficial in areas where a low-cost solution is required for solutions to liquid level measurement and distance measurement. They are not affected by environmental factors such as light, dust, smoke, mist and vapor however only measure the highest reflectance in one direction and often contain a wide sensory lobe profile.

2.7.3 Point clouds

A point cloud is the combination of many data points in a 3D environment. They are often the result of a 3D scanning process from the distance sensors. The point clouds are able to visualise the data points in the 3D space.

2.8 Odometry

Odometry comes from the Greek words ‘hodos’, meaning journey and ‘metron’, meaning measure. It is the use of deriving an estimation of the change in a pose over time, such that it is the position in respect to the initial position (Dudek and Jenkin 2008). There are several types of odometry used to track a robot’s position to help solve the SLAM problem. These include wheel odometry, IMU and visual odometry.

2.8.1 Wheel Odometry

Wheel odometry is the simplest form of odometry. It uses encoders on the wheels of a robot to measure how many revolutions the wheel performs. This data is then converted into a linear displacement which can estimate the position of the robot. This form of odometry is very cheap, however suffers from positional drift and inaccuracies due to wheel slippage (Dudek and Jenkin 2008). It is also restricted to the robots designed with wheels and is not suited to rough terrains or to traverse obstacles such as stairs. This limits the applications as it cannot be applied to drones or hand-held devices.

2.8.2 Inertial Measurement Unit

The Inertial Measurement Units (IMU) is another relative positioning technique. It uses a microcontroller, motion sensor (accelerometer) and rotation sensor (rate gyroscope) to measure orientation and velocity, and estimate the position of the device relative to the initial point. These sensors do not require a reference and are fully self-contained. The device integrates two measurement readings to gain position and velocity estimates. Thus, they are highly prone to positional drift over time as small errors in measurements can result in large errors, which are accumulative. High quality components are often required to produce quality results, which often can be quite expensive. IMU's are often used with another sensor to perform loop closures to correct the position of the device by returning to the origin.

2.8.3 Visual Odometry

Vision based odometry (VO) is the pose estimation process in which the device uses an image stream from a camera to a track objects between the frames of images to estimate the change of the device. This process integrates the pixel displacements between image frames over time. In conditions with good lighting and features to track this can be a low-cost method of determining a pose of the device, and can operate with relative positional error as low as 0.1% (Scarramuzza and Fraundorfer 2011). The areas in which this device is less suited is in outdoor environments where the features do not contain enough texture, ground is not flat, or the lighting and features are dynamic due to wind or sunlight. Also due to large amounts of data collected by image streams, processing in real-time can become a factor as large amounts of computational power may be required.

2.8.4 GNSS

The Global Navigation Satellite System (GNSS) is the term used to describe the process of tracking movement by receiving radio communication from satellite systems. There are several satellite systems that currently operate around the world. The US system is called Global Positioning System (GPS), the Russian system is GLONASS and the European Union system is the GALILEO. In all these systems the process is much the

same. A receiver is required to receive communication from at least 4 satellites in direct line of sight. This requires the receiver to operate in an outdoor environment. With good signal the receiver is able to gain a local and global position of the device with a high confidence of the error, which does not accumulate with time.

2.8.5 Intel RealSense T265

The RealSense T265 is a low cost odometry unit. The device was released by Intel in January 2019 and incorporates two stereo fisheye cameras, each with approximately a 163-degree range of view (±5 degrees), and an IMU to provide 6-dof odometry of the system via processing on the onboard Intel Movidius Myriad 2 VPU, seen in Figure 5 (Intelrealsense.com 2019). The unit uses tracking of visual features to infer the position of the camera before applying sensor fusion with the IMU. The device's proposed applications were commercial robots, consumer robots, augmented reality and drone guidance.

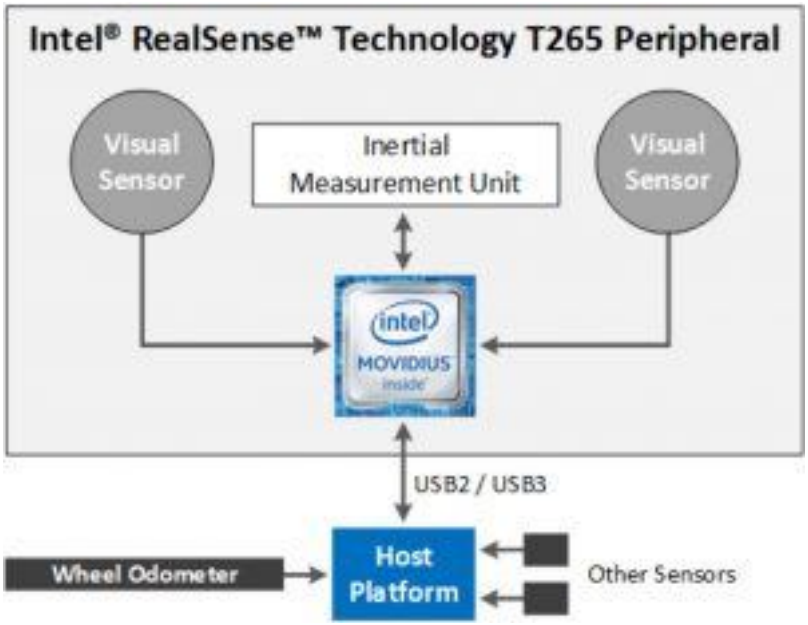


Figure 5: RealSense T265 system overview (Intelrealsense.com 2019)

2.9 Mapping Comparison Techniques

Testing SLAM maps can be done by comparing the position estimate from the SLAM result against ground truth positioning, typically a GPS, or by comparing the resulting point cloud data against a ground truth dataset. Fortunately, at the selected locations for

testing have had surveys, using calibrated, high precision surveying equipment, by the Department of Spatial Sciences at Curtin University. These point cloud datasets will be able to be compared against the 3D maps generated by the various SLAM attempts.

2.10 SLAM Packages

After running through the tutorials, a deep search into the SLAM packages available was performed to create a comprehensive list of packages which are available through ROS followed by a comprehensive analysis of the ROS Libraries for LiDAR based SLAM mapping. However, from further investigation this has been done thoroughly many times before. Below is a general overview of some of the more common and useful packages.

2.10.1 Gmapping

Gmapping is considered the most used SLAM library by robots worldwide (Santos, 2013). It was presented in 2007 and is a grid mapping approach to SLAM and produces 2D maps using a highly efficient Rao-Blackwellized PF SLAM (Grisetti 2007). The package selectively carries out re-sampling operations whilst it optimises the filtering stage by considering the movement of the robot and the most recent observation, drastically decreasing the uncertainty of the pose (Stachniss et al. 2019). The package is well supported through OpenSLAM.org a community page for SLAM algorithms and Github. It has been documented that it usually requires high number of points to produce good results and may struggle with down sampling (Santos 2013).

2.10.2 Hector SLAM

Focuses on 3D pose estimates to produce 2D maps, and does not require odometric information which enables it to be used for backpack devices or UAVs. It uses a Gaussian-Newton equation to scan match and EKF for 3d state estimation. It is considered one of the most used Lidar Based SLAM libraries. However, the results collected by Fang (2016), show the package to have limited use cases as it has been shown to be unable to produce maps with nominally perfect or noisy odometry data. In ROS it creates occupancy grid where in the map shown in Figure 6, black represents occupied, grey represents free and dark grey represents not measured (Neves Dos Santos 2014 and Li et al. 2018)

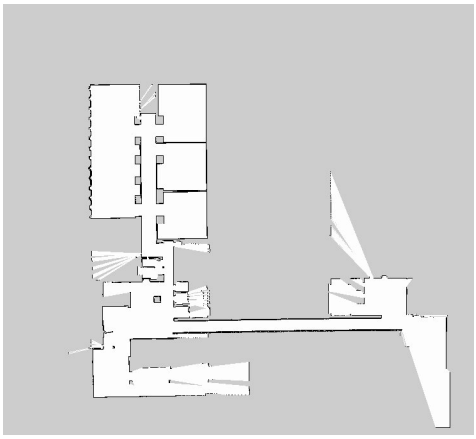


Figure 6: HectorSLAM occupancy grid example (Neves Dos Santos, 2014)

2.10.3 Cartographer

Cartographer is Googles Open Source SLAM library. It is considered one of the most used Lidar Based SLAM libraries with ROS wrapper and is highly regarded as a well-rounded approach to 2D and 3D SLAM problems. The package is very well documented and provide many publicly available datasets through their website (The Cartographer Authors, 2018). However, set-up and configuration with sensor packages is significantly more complicated than other mapping libraries. Figure 7 shows the system overview for how cartographer processes the sub-maps and global maps using sensor data.

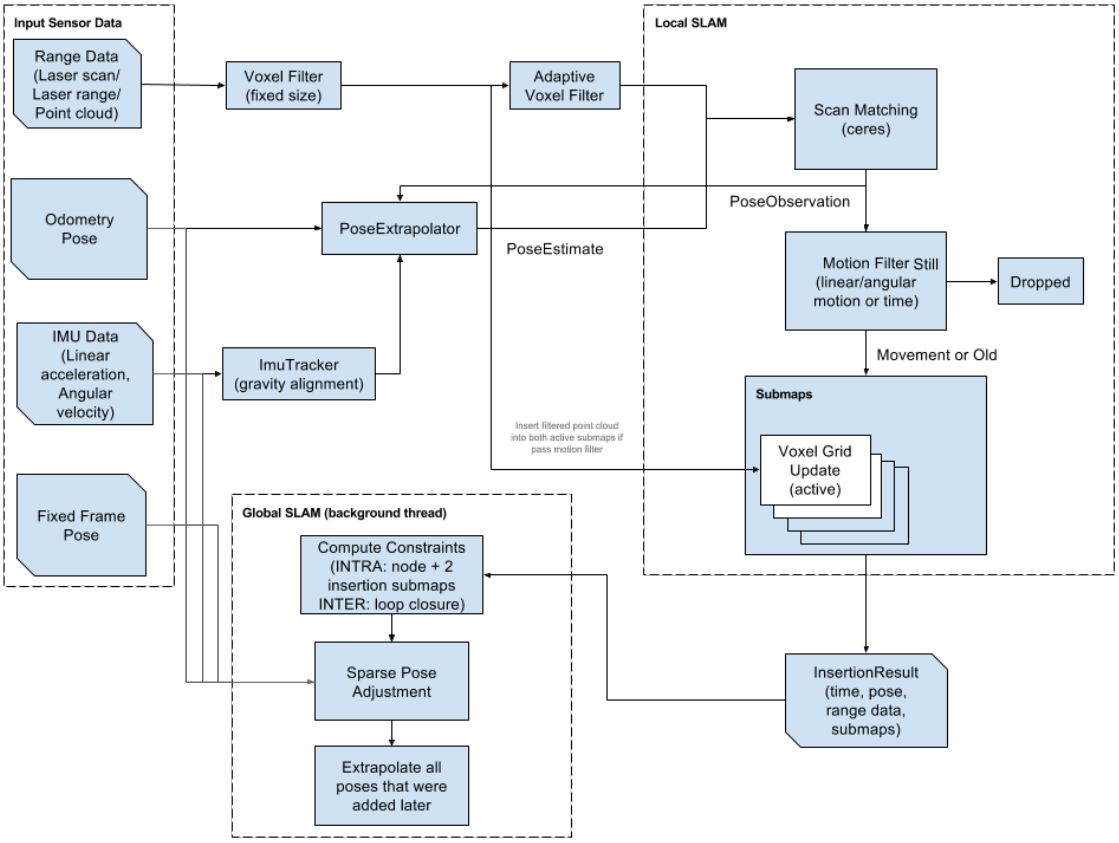


Figure 7: Cartographer system overview (The Cartographer Authors, 2018)

Occupancy grids in Cartographer are performed slightly different to other SLAM packages, it uses a miss and hit approach to building the maps. Where a hit is the data point off an object falling within a grid position and a miss is the cells passed by the laser from the robot to the data point seen in Figure 8 (Hess, et al. 2016).

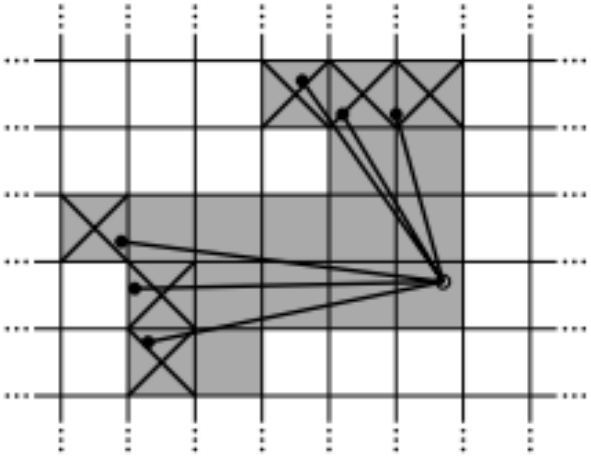


Figure 8: Cartographer occupancy grid with hits crossed out and misses as shaded (Hess et al. 2016)

2.10.4 Karto-SLAM

Karto-SLAM 2D SLAM is a graph-based SLAM approach developed by SRI International's Karto Robotics and uses Sparse Pose Adjustment (SPA) for both scan matching and loop-closures. This algorithm requires more memory with more landmarks, is usually more efficient when maintaining large scale maps environments, and is considered highly effiecient due to only maintaining pose graph (Fang 2016). Fang (2016), used the system to produce accurate results and suggests that it is able to handle noisy odometry data.

2.10.5 CoreSLAM

CoreSLAM is designed to be simple with minimal loss of performance. The algorithm is divided into two steps; performing a distance calculation for each scan using a simple PF and updating the map by matching each scan from the LRF to the map. Through particle filtering it calculates hypothesis and selects the best result (Fang 2016). Santos (2016), shows that the algorithm produces results with large error compared to Gmapping.

2.10.6 LOAM

LOAM was introduced by Zhang and Singh (2014) and contained three main development branches, loam_Velodyne, loam_Continuous and loam_Back_and_Forth.

The velodyne package was developed for the Velodyne LiDAR, whereas the continuous and back and forth was developed for a small Hokuyo LiDAR on a spinning head mount. Since 2017 the package has been unsupported, removed from Github and has become commercialised by Kaarta. However, the package produces high quality results and there still exists branches on Github where the open source community are continuing to make updates.

2.10.7 Industry SLAM Packages

There are several packages that have been commercialised and require either a paid licence to use the software or the software has become proprietary to the company's devices. These include CISRO SLAM (C-SLAM), GeoSLAM and HoverMap. As the licences require an expensive subscription fee they have not been used for this project.

2.11 Coordinate Transformations

The position and orientation of an object in a space is called the pose. The pose must always be referred to a coordinate frame which dictates how far away something is and the direction in which it is pointing. Translations refer to the displacement along each of the x, y and z axis of the coordinate frame and Euler angles are an intuitive representation of rotating the orientation of the coordinate frames. In Euler angles the rotations are performed in respect to the reference coordinate frame and order of rotations is significant. The orientation is expressed as a series of three rotations, roll, pitch and yaw, around each axis, seen in Figure 9 (Sprague 2016).

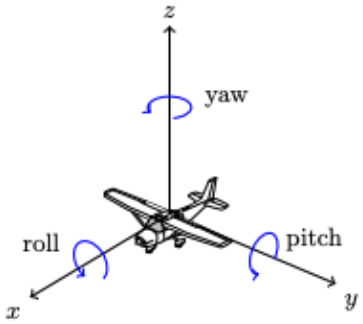


Figure 9: Example of Euler angles (Sprague 2016)

CHAPTER 3: DESIGN REQUIREMENTS

For the design to be successful there are several areas in which the design must meet criteria. The outline for these requirements has been broken down into three areas: Hardware, Software, Performance. These requirements have been set out so that there is a clear and qualitative path for the project to be evaluated and to achieve the main objectives outlined in Table 1.

3.1 Hardware Requirements

As the project aims to be a portable system which is capable of performing SLAM requirements there are several aspects in which requirements must be outlined. These include the processing, physical, safety and power. The IRC Center published a paper outlining the general recommendations for human factors of handheld computing devices. The recommendations applicable to this system have been included in the following sections.

3.1.1 Processing Requirements

For the system to be able to perform real-time SLAM processing using Cartographer, the recommended system requirements are as followed:

Table 2: Processing Requirements

ID	Description
R.1	64-bit, modern CPU (fast enough to compute real time SLAM)
R.2	16GB RAM
R.3	256GB storage
R.4	Graphics processing needs to be able to compute real time SLAM

Other requirements necessary for the system include enough storage space to capture and process the data. Between the PUCK and T265 the system is likely to be producing data of 3GB per minute, therefore the minimum storage requirement should be 256GB.

3.1.2 Physical requirements

The system is designed to be portable and this places some restrictions on the physical specifications of the device. For the device to be handheld the total weight of the system must be carriable by a human. The U.S Department of Defence (1995) and IRC Center (2005) determined the following recommendations for the physical design of a system.

Table 3: Physical requirements

ID	Description
R.5	Portable equipment should have rounded corners and edges (DoD, 1995).
R.6	Hand held equipment should be small (less than 100 x 255 x 125 millimeters), lightweight (less than 2.3kg) and conveniently shaped (DoD, 1995).
R.7	The equipment should have a non-slip surface and be shaped so as to prevent it from slipping out of the user's hand (DoD, 1995).
R.8	The equipment should be equipped with a means (such as a string, strap, or clip) to attach the device to the user's body or clothing when not in use so that the equipment does not interfere with the accomplishment of other tasks when not in use (DoD, 1995).

R.9	Devices should have good legibility and colour contrast, and be easy to learn. (IRC Center, 2005)
R.10	Devices should have sufficient screen size and that resolution for the task (IRC Center, 2005).
R.11	Devices should be sufficiently durable to withstand drops and knocks associated with normal use (IRC Center, 2005).
R.12	Devices must have an easy means of connecting to and transferring data to or from other systems (IRC Center, 2005).
R.13	If the device is used to transmit data over a wireless network, it should have consistent and available connectivity (IRC Center, 2005).
R.14	The display should accommodate expected operational lighting conditions, both high and low illumination (IRC Center, 2005).
R.15	Components must be available to source via purchase or loan.
R.16	Components must be able to connect with each other.

3.1.3 Safety Requirements

In order to create a device that is to be safely operated by the user, a standard of safety requirements must be met. The following criteria has been developed for this device:

Table 4: Safety requirements

ID	Description
R.17	Areas which the user may come in contact with on handheld or wearable devices should not produce so much heat as to be uncomfortable to the user (IRC Center, 2005).
R.18	All cables must be neat and of good condition for operation.
R.19	The device must be weather-proof or weather-resistant.
R.20	All lasers must be safe for contact with human eyes.

3.1.4 Power requirements

The system is required to power both the PUCK and Xavier via the battery. The T265 will be able to run from the USB-A power of the Xavier. The requirements for this system are:

Table 5: Power requirements

ID	Description
R.21	Handheld equipment should not require attachment to an electrical outlet (DoD, 1995).
R.22	Devices must have sufficient battery life for task completion (IRC Center, 2005). For this system we expect to be running for at least 1hr for each scan consuming ~23W at ~5V therefore the minimum battery size is 4600mAh.
R.23	Must be able to power Xavier (~15W) and Velodyne (~8W) at the same time.
R.24	Must be able to power Xavier via USB-C or DC-Jack
R.25	Must be able to power Velodyne PUCK via DC-Jack
R.26	Must be able to power RealSense T265 via USB-A connection.

3.2 Software Requirements

In order for the system to achieve the required goals, the software for the final project must complete a set of requirements. The requirements have been outlined in Table 6.

Table 6: Software requirements

ID	Description
R.27	The software must enable the user to remotely connect to the device, as the system does not have a screen attached when performing field work.
R.28	The software must enable the user to use a virtual keyboard or control the keys from the remote computer.
R.29	The system must run Ubuntu 18.04 as this is the recommended OS for Cartographer.
R.30	Must run ROS Melodic as this is the recommended software for Cartographer and RealSense packages
R.31	The system must be able to perform SLAM using Cartographer as the preferred SLAM package.
R.32	The system must be able to perform SLAM in post processing
R.33	The system must be able to perform SLAM in real-time

CHAPTER 4: HARDWARE DESIGN

The hardware design consisted of four areas: Hardware Selection, Sketches, CAD Design and Implementation. The overview of the system's signals and design can be seen in Figure 10.

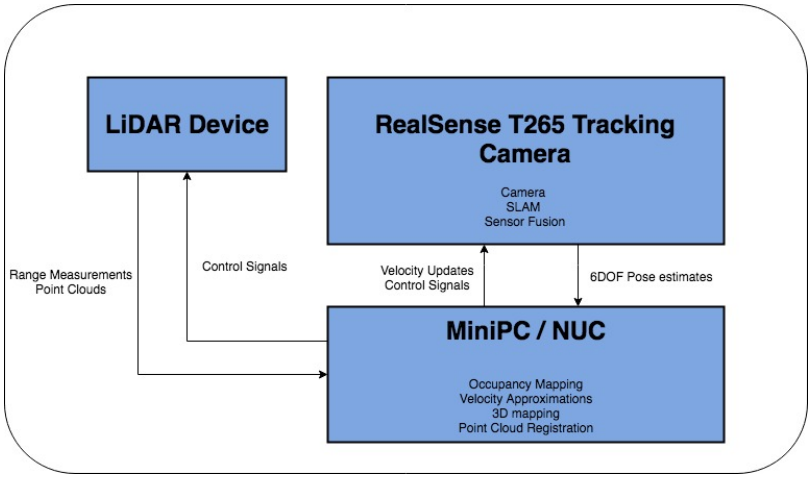


Figure 10: Overview of System Diagram for deployable system

4.1 Hardware Selection

The hardware was selected to meet the requirements outlined in chapter 3. Each component was selected to work for the overall system. A complete breakdown of components and their respective cost can be seen in Table 7.

Table 7: Bill of materials

Quantity	Description	Cost ($AUD)
1	Velodyne PUCK	~5000
1	RealSense T265	300
1	NVIDIA Jetson-Xavier	1200
1	Kogan 26800mAh Power-bank (75W)	70
1	3D printed sensors mount	~5

1	3D printed Xavier mount	~5
1	CAT-6e (Flat) Ethernet Cable	5
1	USB-C to USB-C Cable	0 (included with Xavier)
1	USB-A to DC-Jack Power Cable	3
1	Bottom-Half Fishing Tackle Box	7
1	1/8th Screw for LiDAR Mount	10
	TOTAL COST	**~$6,605**

4.1.1 RealSense T265

The RealSense T265 was selected to perform odometry for this system as it is a new, small form factor, low cost, reportedly accurate method and enables the system design to be more flexible as it does need any design constraints unlike wheel odometry.

4.1.2 Velodyne PUCK (VLP-16)

This sensor has been selected as it was available to loan over the period of this project. The sensor is a high-quality, industry standard for precision laser scanning.

4.1.3 NVIDIA Jetson-XAVIER

The Xavier was selected as it enabled the device to be powered by a USB-C connection on a lower power mode. It also has significantly higher graphics processing capacity with its NVIDIA Volta GPU containing 384 CUDA cores and 48 Tensor Cores, and 8GB of LPDDR4 RAM (NVIDIA 2019). This unit performs significantly better than most single board computers available on the market. It also contains GPIO pins enabling control of other devices. These pins were originally considered to power and control a light source at the front of the system. One aspect that was not considered before purchasing was the Xavier does not come standard with Bluetooth and Wi-Fi

connection. Therefore, an additional Intel 2635 M.2 Bluetooth and Wi-Fi adapter kit was required to be purchased.

4.1.4 Power Supply

The cost for the systems portable power supply was required to be as low as possible. The Kogan 26800mAh power-bank (75W) provided a good combination of size, power availability and duration. This power-bank enabled the Xavier to be powered directly through a USB-C connection and the PUCK through a customised USB-A to DC-Jack connection and 12V QC3.0 Trigger tester. The trigger tester board was required to inform the battery's microcontroller that 12V was desired rather than the standard 5V.

4.1.5 Additional

Due to the additional cabling required for the Velodyne, which in future applications may be possible to trim down, a container was required to carry the system. This container was once again selected to be as cheap as possible. After a search of various plastic boxes, a fishing tackle box was found to be the correct size and cost. This was dissembled to have the bottom half holding the device. It also contained a non-slip surface which was beneficial to carrying the device.

After initial testing it was found that the rotation of the PUCK's laser was causing noise to occur with the T265's IMU. To help restrict this movement the system was damped by applying foam underneath the device and pulling the system down into the box with an elastic band.

4.2 Sketches

There were numerous attempts at sketching designs that would incorporate all of the requirements into the smallest form factor. In Figure 11 shows a sketch of the final design. This was the first sketch that moved the mount point of the T265 above the rim of the PUCK as it became apparent the laser exited the device much higher than originally expected. This design also provided extra support at the back of the PUCK mount for the cable exiting the device. Another method for supporting the PUCK cable may have been to rotate the device 45 degrees so that the cable ran over the edge of the square base. However, this would have added unnecessary complexity when performing transformations of the sensor data.

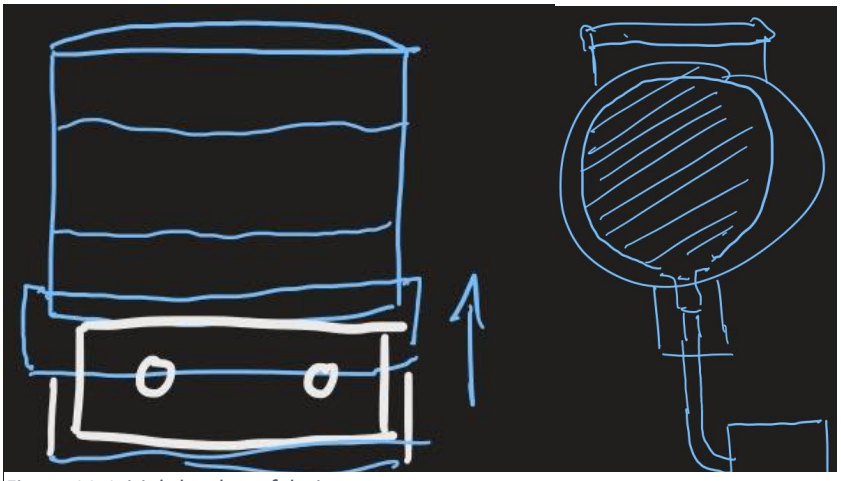


Figure 11: Initial sketches of design

4.3 CAD Design

The CAD design was formed by first downloading the design files for the selected hardware components and importing into Autodesk Fusion360. Then for the Xavier mount the existing mount brackets were modified to be extended to hold the sensor mount. Similarly, for the sensor mount the Velodyne PUCK design file was used to measure the base platform and locking tabs required and the T265 files were used to measure the screw offsets. This process can be seen in Figure 12.

In early development a trigger grip mount was designed to hold the device. However due to excess cabling, large battery and lack of time, this stage did not eventuate.

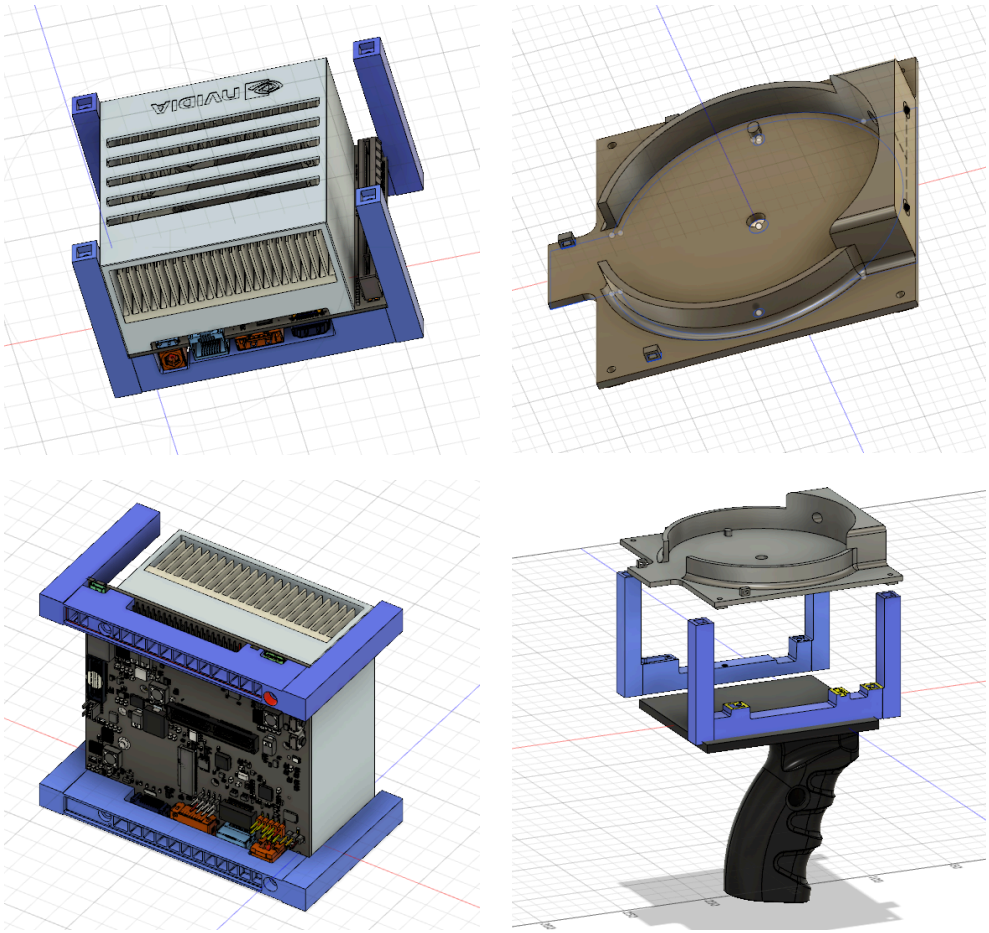


Figure 12: CAD design

4.4 3D Printing

The 3D printing was performed by first slicing the CAD design files in IdeaMaker before printing on the Raise3D Pro. This process enables a highly accurate and solid foundation for the device. The final printed mount can be seen in Figure 13.

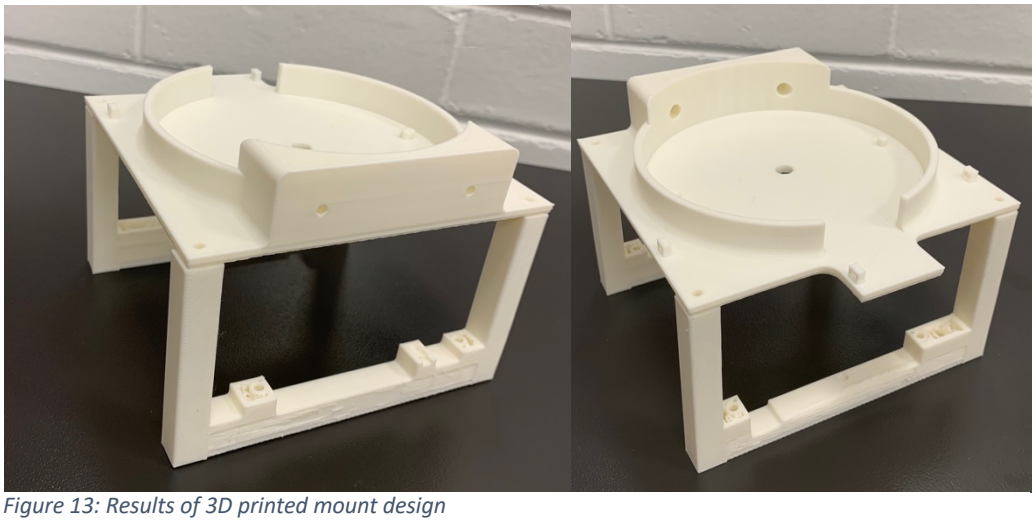


Figure 13: Results of 3D printed mount design

CHAPTER 5: SOFTWARE DESIGN

The software architecture was created in order to run Cartographer in both real-time and post-processing. Unfortunately, due to data buffering requirements the real-time SLAM could not be recorded for post processing and therefore whilst the system can perform real-time SLAM, the data collected is only valid for post-production. In order to run both real-time and post-processing (collecting) systems, two different software setups were created. These two setups perform very similar processes with one method running cartographer in real-time and the other running a rosbag record topic to collect data.

The overall system overview can be seen in Figure 14 and show how each process is embedded within the system. The basic system overview for SLAM can be seen in Figure 15, and follows the form of T265 sending pose data, to Cartographer, and PUCK sending point cloud data, via the velodyne_points topic, to Cartographer.

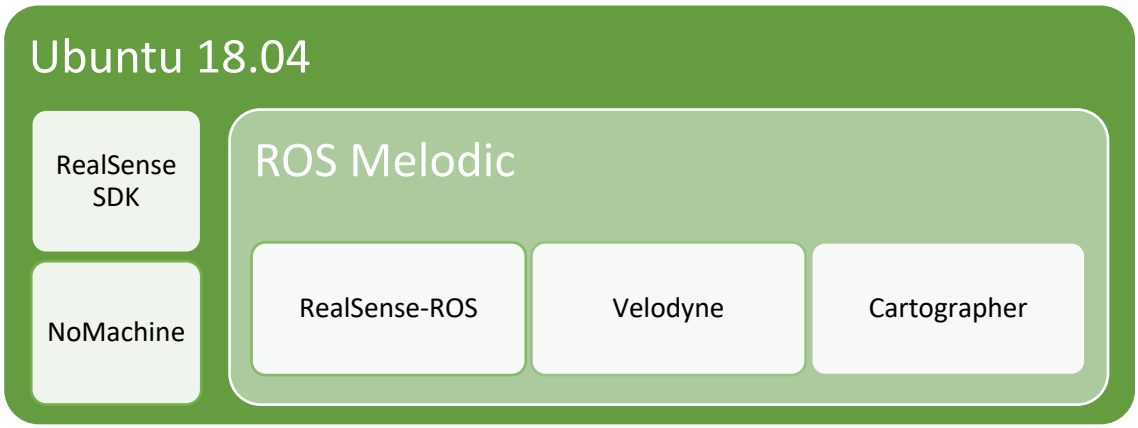


Figure 14: System overview

In order for the system to accurately combine the point cloud and pose data a series of coordinate transforms had to be performed. The transforms have been completed by two different methods, the static_transform_publisher and robot_description_publisher via a Universal Robot Description File (URDF) containing the description of the robot.

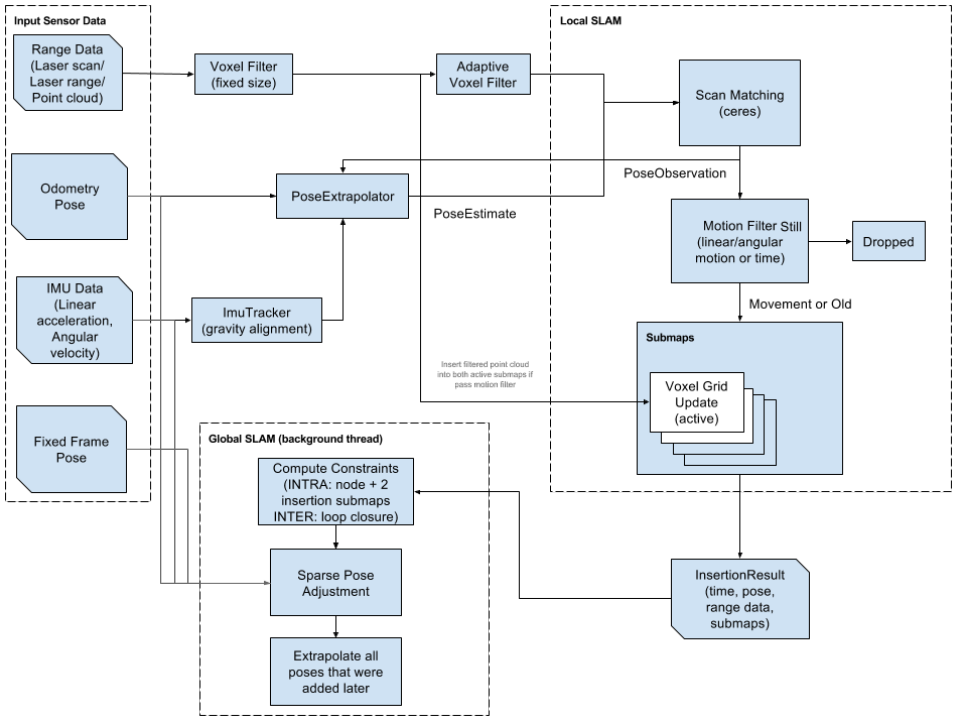


Figure 15: SLAM system overview (The Cartographer Authors, 2018)

5.1 Coordinate Transformations

In this system we have 3 main coordinate frames; pose frame, IMU frame and velodyne frame. Each frame has a different position and orientation, therefore a transformation must be done to accurately communicate the sensor data to Cartographer. Each of these frames were rigidly attached to each other via the mount meaning that the transformations were able to be static. In this system we have adopted base_link to be the same as the pose_frame. The way the transformations were done within this system was via URDF and static transform publisher, using translations and Euler angles.

The T265 contained many different coordinate frames, the two frames used by this system include the pose_frame and imu_frame, seen in Figure 16 and Figure 19. The resulting transformation between the two frames is:

Translation: [-0.000, 0.021, -0.000]

Rotation: in Quaternion [0.000, 0.000, 1.000, -0.000]

in RPY (radian) [-0.000, -0.000, -3.142]

in RPY (degree) [-0.000, -0.000, -180.000]

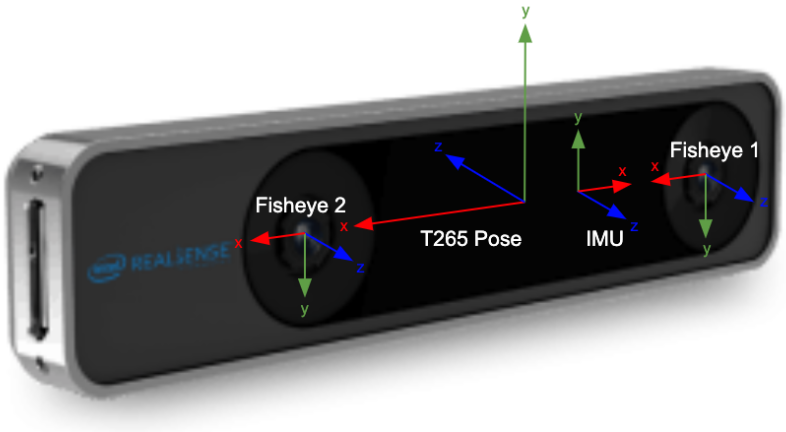


Figure 16: RealSense T265 Sensor Coordinate System

The velodyne frames are slightly different and at first confusing. The datasheet shows the coordinate frame as x to the left, y forward and z up, Figure 17 (velodynelidar.com 2019). However, the ROS package velodyne_points use the frame, shown in Figure 18, which has x forward, y to the right and z up. Therefore, the transformation needed between the pose frame and the velodyne frame is:

Translation: [-0.06345, -0.010, 0.000]

Rotation: in Quaternion [0.500, 0.500, 0.500, 0.500]

in RPY (radian) [1.5708, 1.5708, 0]

in RPY (degree) [90.000, 90.000, 0.000,]

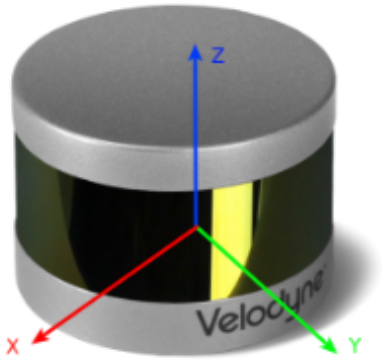


Figure 17: PUCK coordinate frame described in manual (velodynelidar.com 2019)

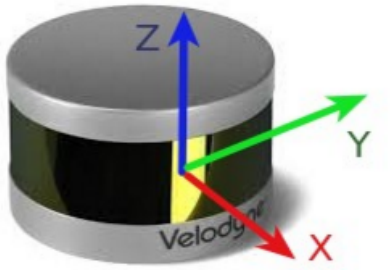


Figure 18: Frame of reference from the velodyne_driver node (velodynelidar.com 2019)

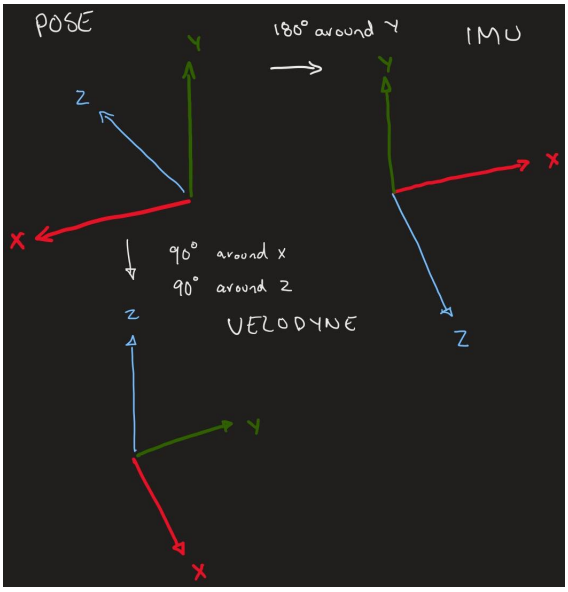


Figure 19: Transformations between the system frames

5.1 Transformation Tree

There is a standard robotic transform tree that applies to robotic systems. This follows the convention shown in Figure 20. and is the specified recommended format for use with Cartographer (The Cartographer Authors, 2018).

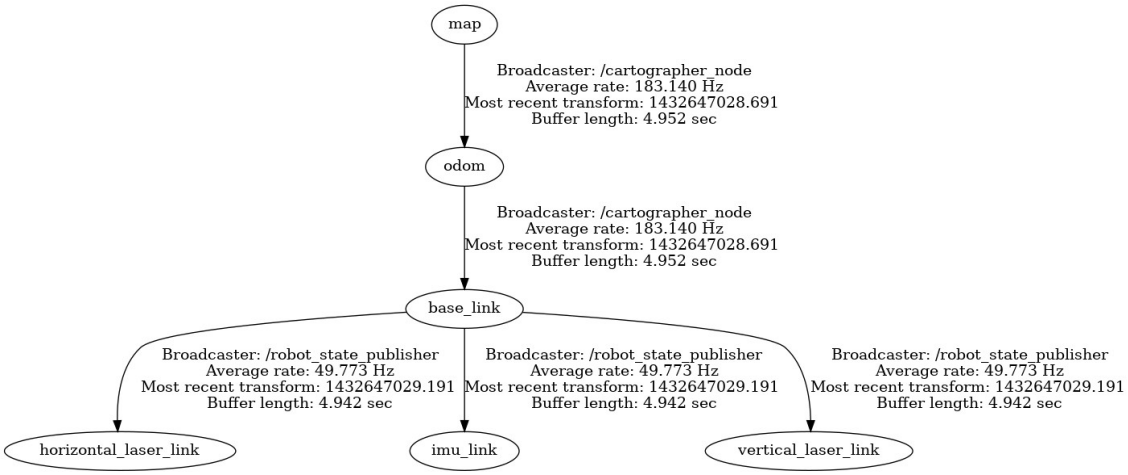


Figure 20: Example of Cartographer required transforms tree (The Cartographer Authors, 2018)

Unfortunately, the realsense-ros package effects this standard flow and breaks cartographer as the topics get re-published after launch. To counteract this process a static_transform_publisher was created to reassign the imu_optical_frame to link with the base_link. This transform tree can be seen in Figure 21.

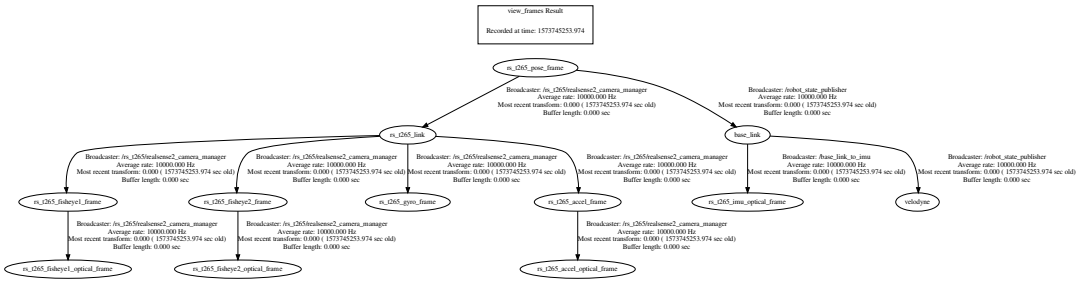


Figure 21: Tf tree of system while recording

When post-processing the recorded topics in cartographer the transform tree becomes Figure 22, which follows the standard form required for Cartographer.

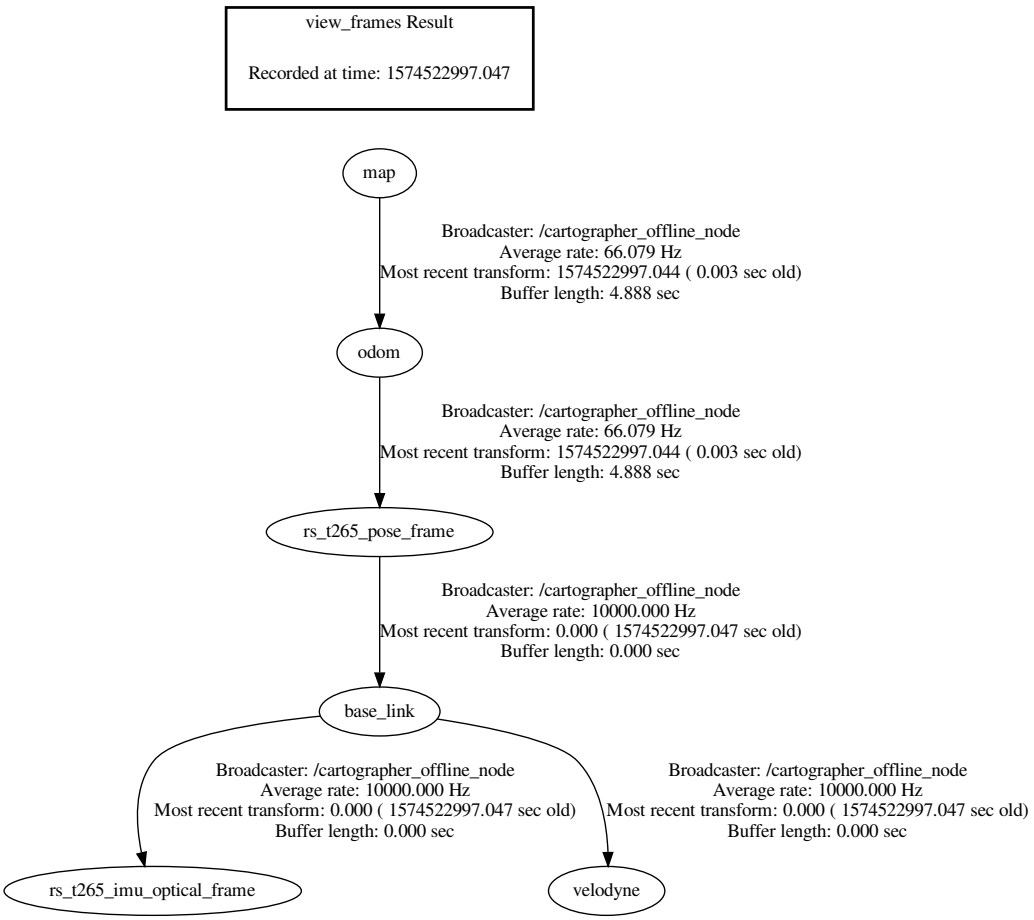


Figure 22: Tf tree broadcasted in post processing by the cartographer_offline_node

5.3 User Interface

NoMachine remote desktop software was selected for the controlling the machine remotely. This software was the only available remote desktop software that enabled the user to connect to the machine via desktop, iPad or phone. Running the remote desktop software enabled the device to not have to power a screen whilst running. Figure 23 shows NoMachine running on an iPad. This software also enables a virtual keyboard so that the user can operate the device as normal.

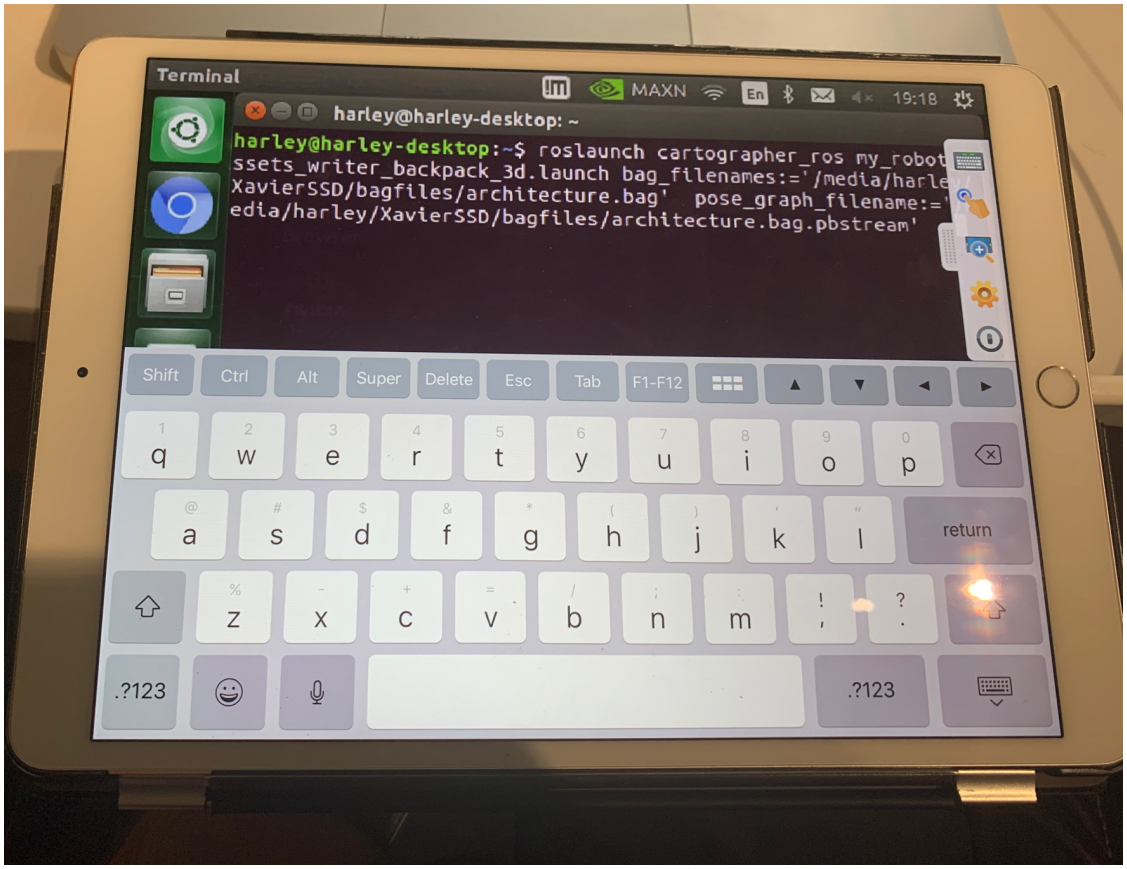


Figure 23: iPad remotely connected to system via NoMachine

5.4 Cartographer

Cartographer was implemented into this system by modifying the backpack example's launch, configuration and URDF files, provided on the cartographer-ros git repository. This process was an intensive and repetitive, as the cartographer algorithm is very particular about the sensor information it receives. Two sets of launch files were created to run cartographer, one real-time processing and one offline. Whilst the real-time processing works well, the sensor stream from the T265 was unable to record into a bagfile at the same time, so limited results have been recorded. For the offline_node, during launch a rosbag record command is started, which records all of the topics published except the raw camera feed of the T265. This recorded rosbag file could then be processed using the offline processor, and the cartographer settings tuned to for the system.

Tuning was performed via settings in .lua file which contain three stages, the map_builder, trajectory_builder and pose_graph. Whilst the settings were normally adjusted to for one purpose, it often had an adverse effect to other areas of the SLAM process. The map_builder settings contained processing settings for the overall building of the map and included settings like how many background threads to run and whether to process in 2D or 3D. The pose_graph settings were used for the local and global SLAM and contained settings such as how often to optimise the maps and how much weight to give the sensors and local slam results. Trajectory_builder settings mostly used for applying filtering and additional local SLAM parameters to Cartographer. All configuration and launch files can be found in the Appendix attached.

CHAPTER 6: RESULTS AND DISCUSSIONS

6.1 Data Collection

The system was set up and used to collect data at three locations, each containing their own challenges to for the system. The methodology for the setup can be seen in the instructions of use (Appendix A) and can be seen in Figure 24. The three areas mapped for this device were selected due to the availability of surveyed data for which this project will consider as ground truth. This dataset was collected by the Spatial Sciences Department at Curtin university with the use of a Leica C10, high quality 3D laser scanner. Each of these maps took several days to create. There may be some errors contained within these datasets however for the purpose of this study they act as a good baseline to measure the accuracy. The three areas mapped included the Architecture building at Curtin University, Leighton Battery Tunnels in Mosman Park, and Henderson Court at Curtin University.

Figure 24: Starting the system at the Leighton Battery Tunnels entrance.

6.1.1 Architecture Building – Curtin University

The Architecture building was the first map produced by the system. It was selected because its indoor corridor which had been surveyed previously and contained a long flat corridor, indoors which could test the systems reliability and drift over the length of the building. The system was started at one end of the corridor and traversed level 2 within the building following the left hand wall until the system returned to its starting location. This ensured that the floor had been scanned completely and no areas were missed. This environment was active during the scanning and had several people walk within the range of both the scanner and the camera. Fortunately the system was robust enough to process the data correctly and not lose tracking, however it may have introduced some error into the final maps.

6.1.2 Leighton Battery Tunnels

The Leighton Battery Tunnels are located in Mosman Park, Western Australia. They are historic artillery tunnels built in World War II. The tunnels are a network of dark, long and narrow corridors with a large staircase leading into and out of the tunnels. Figure 25 shows the scanner in use within the tunnels.

The tunnels had lights illuminating most of the paths, however they had to be mapped twice as the first time the camera lost tracking in the dark spots around corners and the long corridors. During the second attempt a torch was held underneath the device to illuminate the path as it progressed. This helped the scan significantly, however, upon reviewing the footage of the T265 it was apparent the torches spot light was often too bright and removed some of the detail from the image. An example of this can be seen in Figure 25

Figure 25: Scanning at the Leighton Battery Tunnels

6.1.3 Henderson Court – Curtin University

Henderson Court is a wide open, grassy area at the center of Curtin University. It contains many tall trees that block the sky and windy paths that traverse up the hill. Upon scanning the area was very active with many people moving around during scanning.

6.2 Scans and Maps

The maps were produced in post-processing using Cartographer's asset_writer function. This creates a point cloud from the pbstream and bagfiles and produces images from the different sides of the project. Within this map the darker outlines represent dense points collected, the green dot indicates a starting point, red indicates finishing location, and the blue line indicates path travelled throughout the scan. Whilst it wasn't intended the scanner was able to observe points outside of the building when passing doors and windows.

6.2.1 Curtin Architecture Building

The architecture building produced the maps seen below in Figures 26, 27 and 28. There are some areas such as near the start and end points where the map has not been aligned correctly. This misalignment may have occurred from sensor drift at the end of scanning, as the device was placed onto a table for several minutes before switching off and may have incorrectly tracked the operators movement. In future scans of this scale the building should be attempted to be traversed at least twice to improve the accuracy of the global SLAM. The maps also show some indicators that there may be a drift in the height of the device. This can be seen as the floor looks to drop away to the left in Figure 27 and by the shadowing of Figure 28. However, this may also have been cause by the T265 IMU starting from an orientation not level with the floor, or some other rotation to the SLAM. This map shows that the device was accurately able to perform SLAM over the length of the building, however does show room for improvement. A future challenge would be to traverse the staircases within this building, as an attempt to scan the entirety of the building.

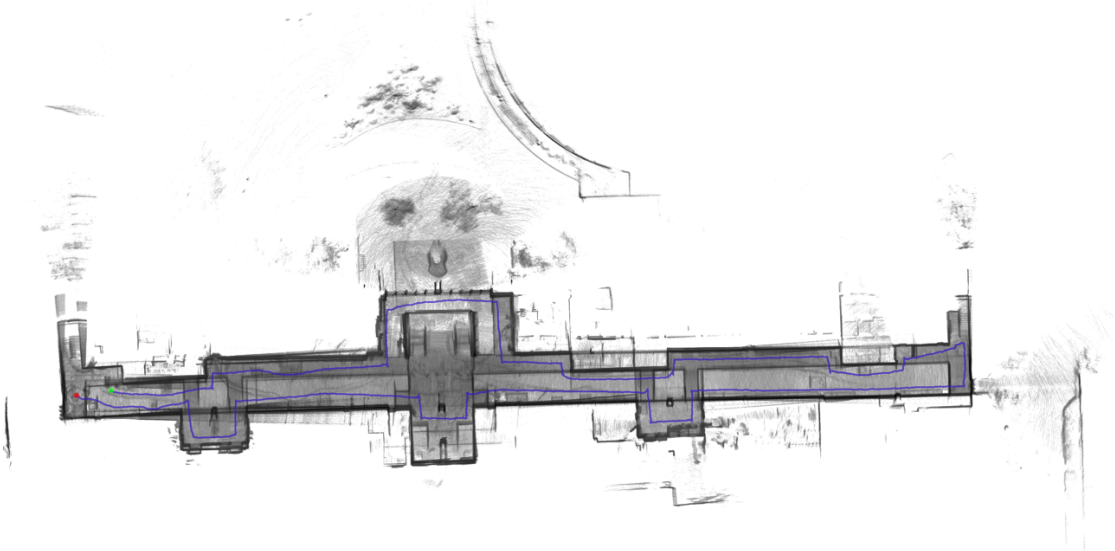


Figure 26: Architecture building map produced (above)

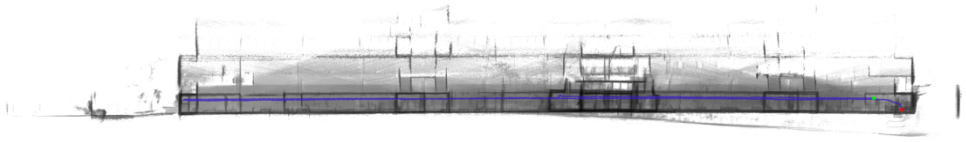


Figure 27: Architecture building map produced (side)

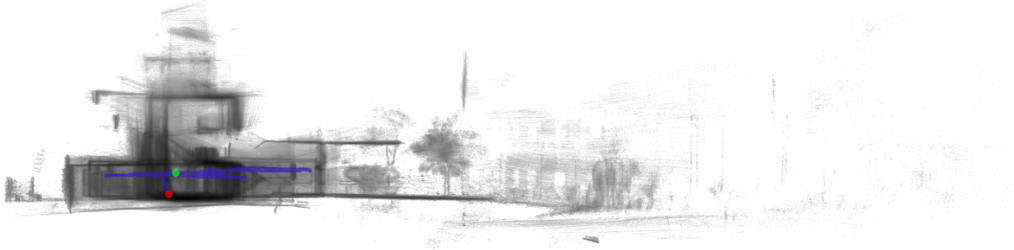


Figure 28: Architecture building map produced (Front)

6.2.2 Leighton Tunnels

The Leighton battery tunnels were difficult to navigate and map. The maps can be seen in Figures 29, 30 and 31. The maps show that the device was able to perform SLAM throughout the maze of tunnels. The map in Figure 29 also shows signs of overlap in the center of the scan. This likely would be able to be rectified with further tuning of Cartographers settings. The scatter off to the right of Figure 29 and Figure 31 show the start of the limestone tunnel that leads to a cliff-face. However, due to unstable grounds this section was not mapped during this project. There were also several long staircases which could be traversed in future-works. These were not attempted during this project, however could be used to close a loop of the system.

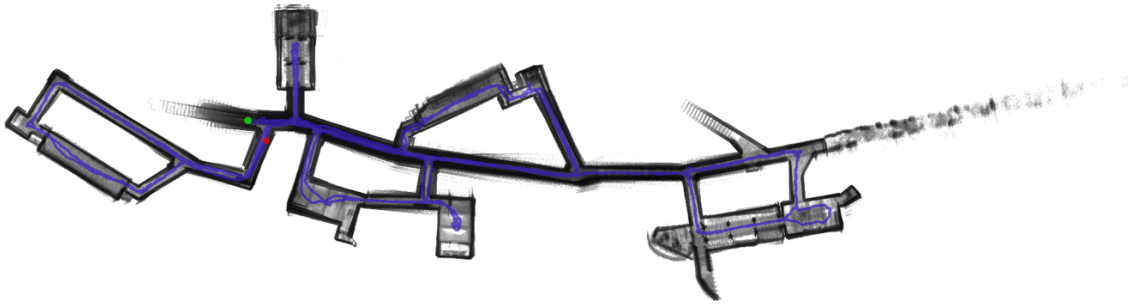


Figure 29: Leighton tunnels map produced (Above)

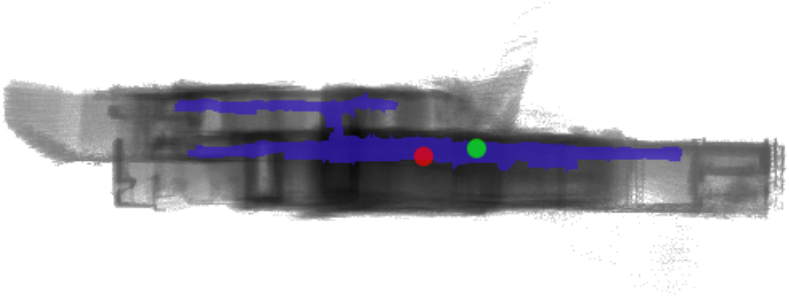


Figure 30: Leighton tunnels map produced (Front)

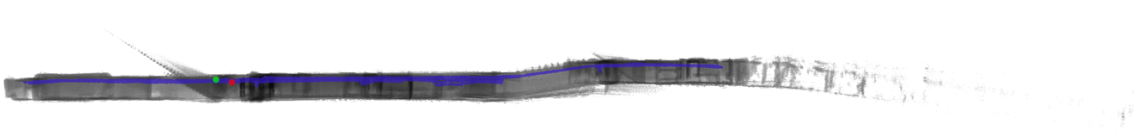


Figure 31: Leighton tunnels map produced (Side)

6.2.3 Henderson Court

The Henderson Court maps, shown in Figures 32, 33 and 34, was the largest map produced during this project. The area was traversed over 25 minutes following the blue line seen in Figure 32. There are some areas of the map which exhibit signs of double up, such as the bottom where there are two walls placed slightly offset. Once again this may be rectified via tuning of Cartographer parameters, however, the overall map produced good quality. There are several areas of the map that were only traversed once, in future work the quality may be increased by traversing the whole scene at least twice.

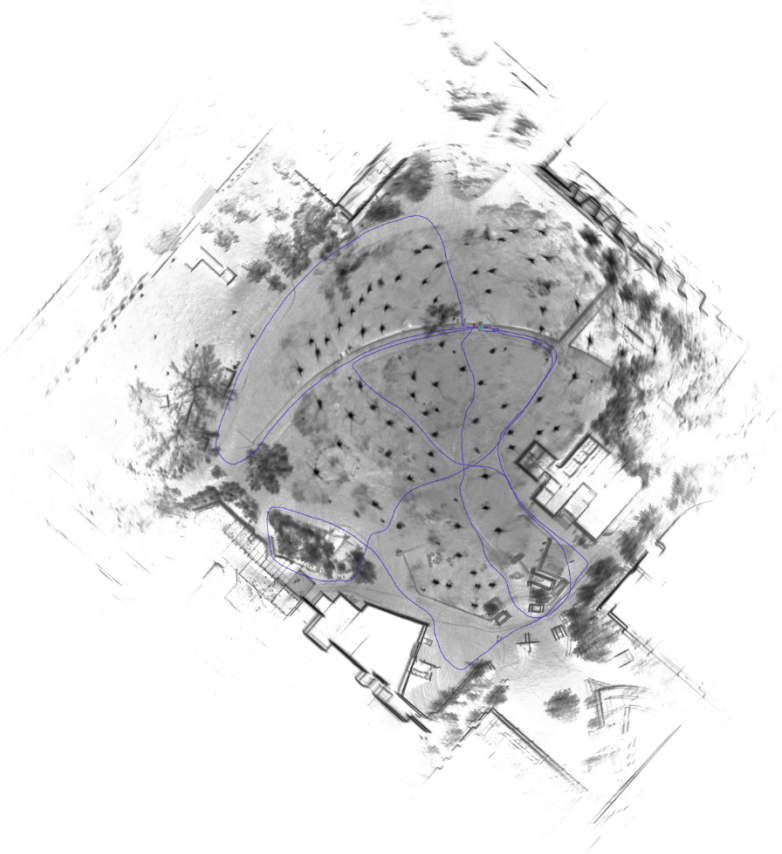


Figure 32: Henderson Court map produced (Above)

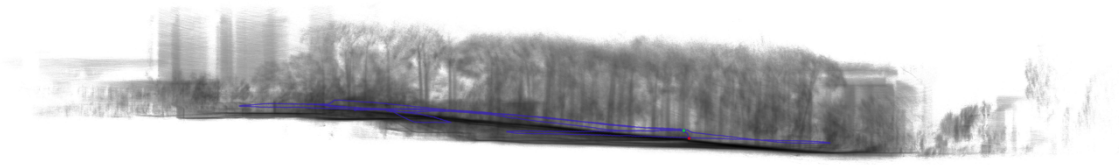


Figure 33: Henderson Court map produced (Side)

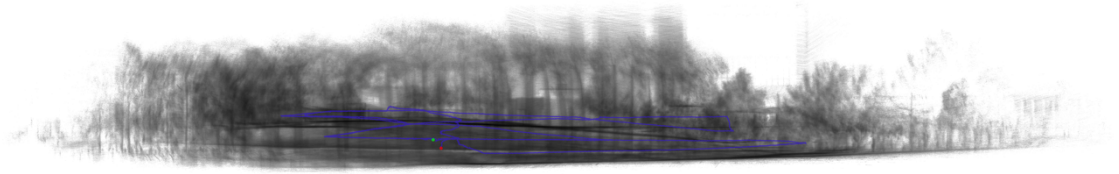


Figure 34: Henderson Court map produced (Side)

6.3 Comparison to truth

To compare the point clouds produced by Cartographer and the ground truth dataset, the cloud comparison software CloudCompare was used. This is an open source software that has been developed by Girardeau-Montaut (2003). Due to processing capabilities and the size of the datasets, before the point clouds could be compared, subsampling was performed on each dataset. This was performed at a minimum of 0.01m, therefore there could not be any datapoints within 0.01m of each other. This will reduce the overall reported accuracy of the system. The measured point cloud was aligned using ICP. Once aligned the distance between the points within the clouds were calculated using CloudCompare's Cloud/Cloud Distance calculator.

6.3.1 Architecture Building

The comparison to ground truth showed good fit over the entirety the building at a slice approximately 2m off the floor height. There are some areas which were mapped during this process that were not contained within the ground truth and show up as the warm (RED) outliers, the difference can be seen in Figure 35, containing the ground truth slice.

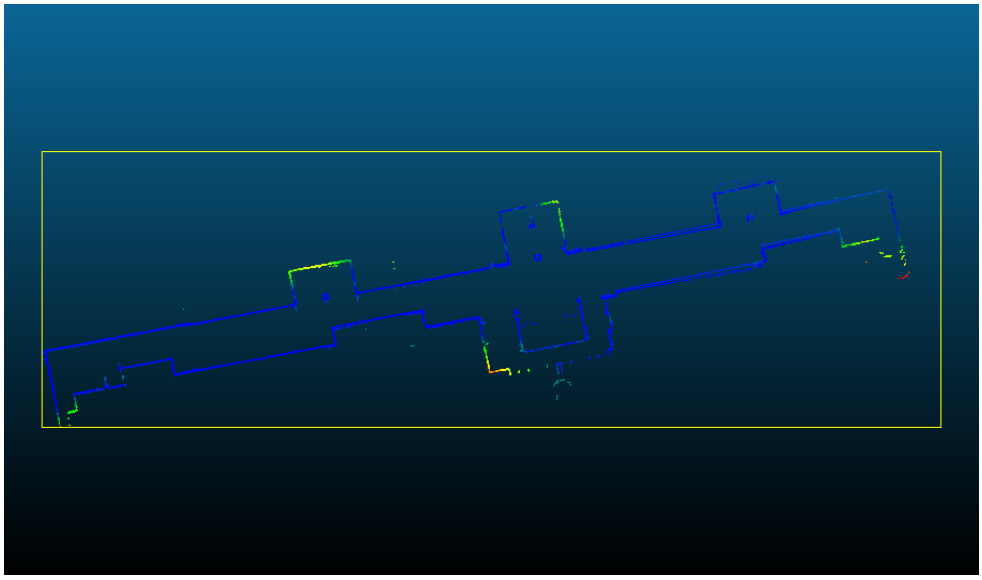


Figure 35: Slice of Architecture building with heatmap of difference of points

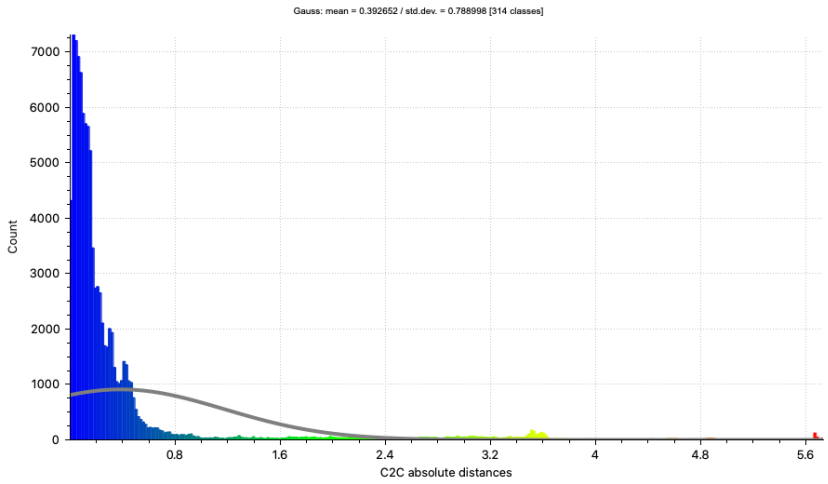


Figure 36: Gaussian histogram of distance of points for slice of architecture building

Figure 37 and 38 shows the difference of point clouds results (colour) against the ground truth data (grey). It can be seen that there are several locations along the building were points collected were not included in the survey data.

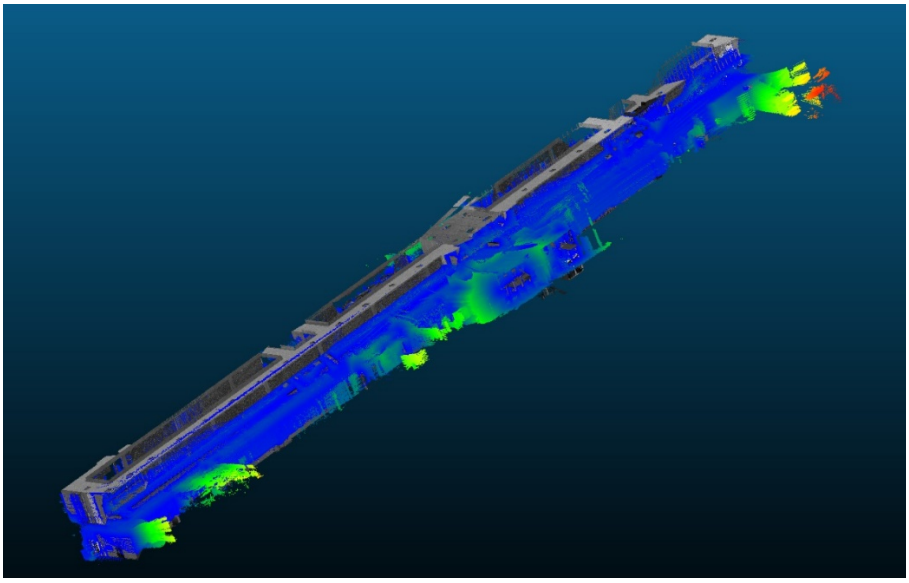


Figure 37: Architecture ground truth (grey) against result (colour)

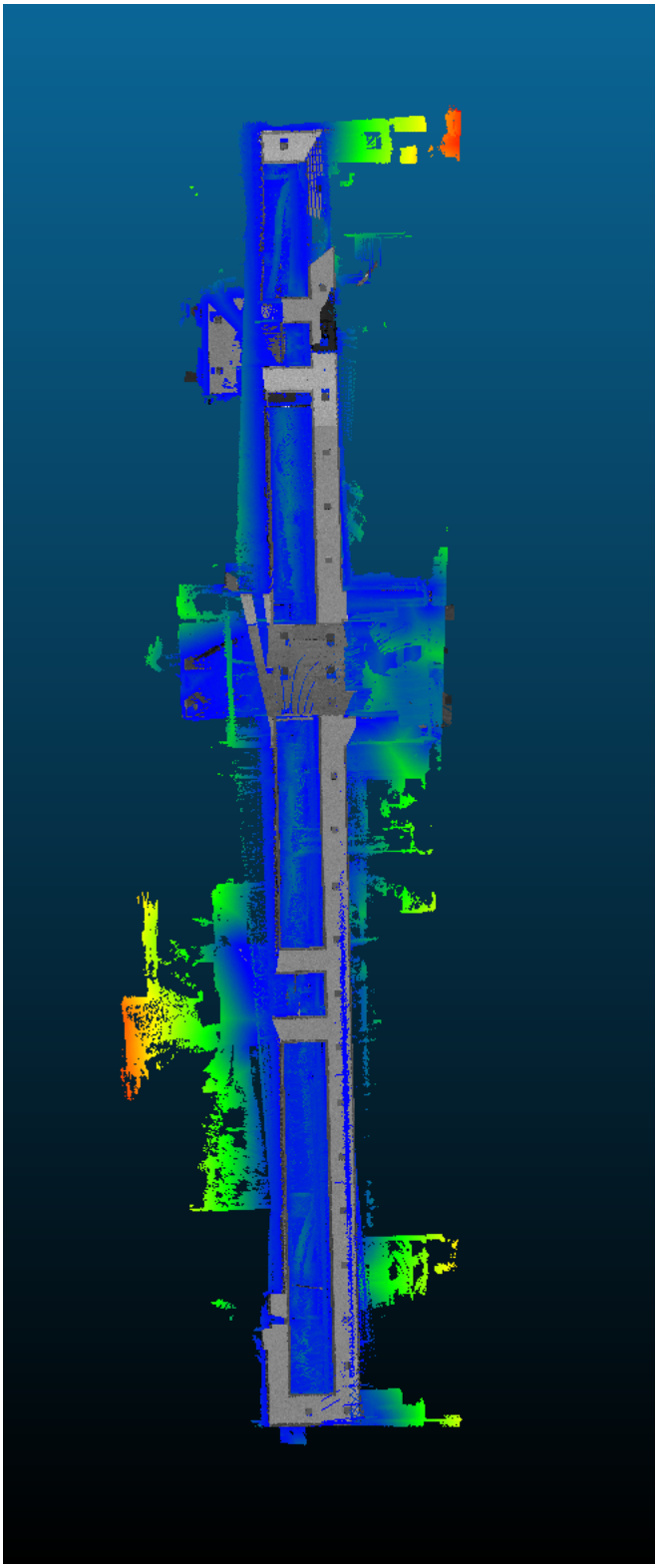


Figure 38: Architecture ground truth (grey) against result (colour)

The map in Figure 39 shows the distance between point clouds of the Architecture building taken at a 9m section from the floor upwards. The results show a good fit of the data with a mean of 0.33. There are sections shown that once again have been included in the results that were not collected in the survey data, these show as the bright spots.

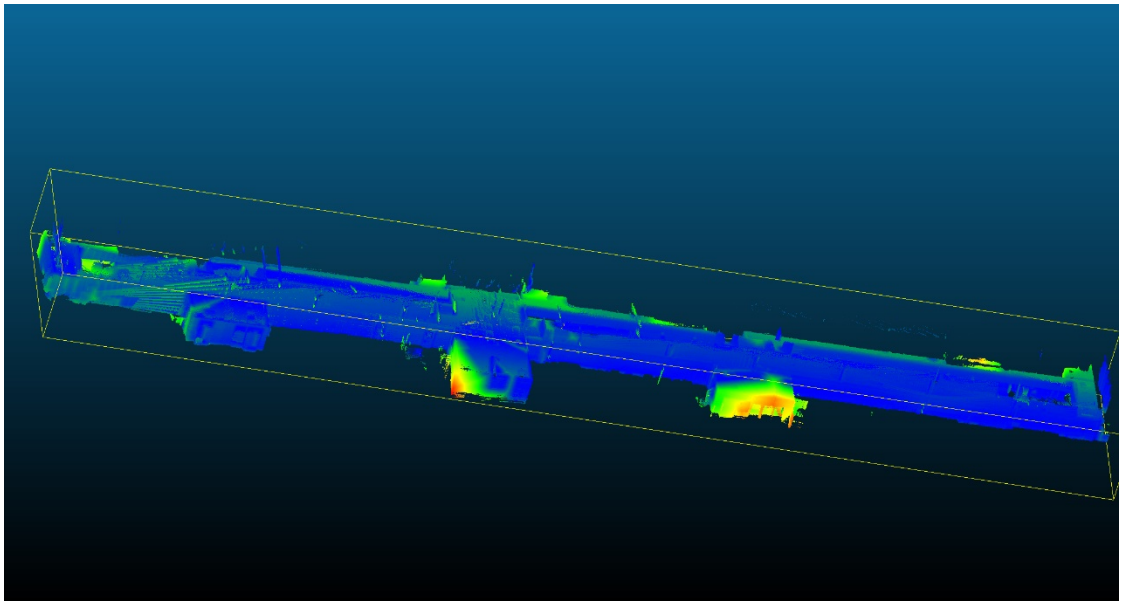


Figure 39: Architecture result from difference of point clouds

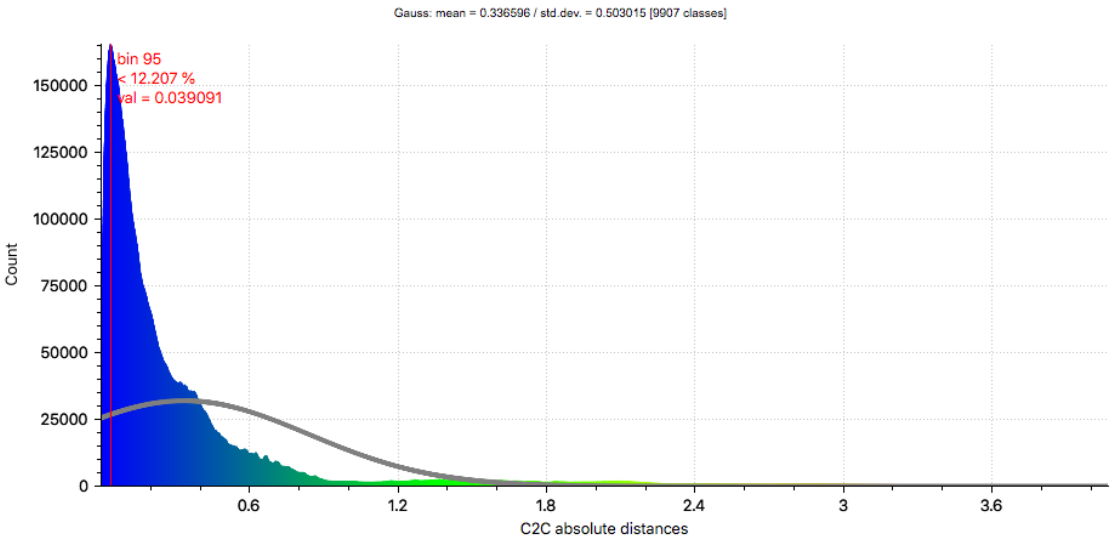


Figure 40: Architecture results distribution of difference of points

6.3.2 Leighton Battery Tunnels

The Leighton Battery tunnels produced accurate results with a mean point cloud to point cloud distance of 0.17m. There is an obvious outlier to the results with a red patch midway through the scans, Figure 41. This is likely an error during the mapping process, as the scans captured a person at this point of the map.

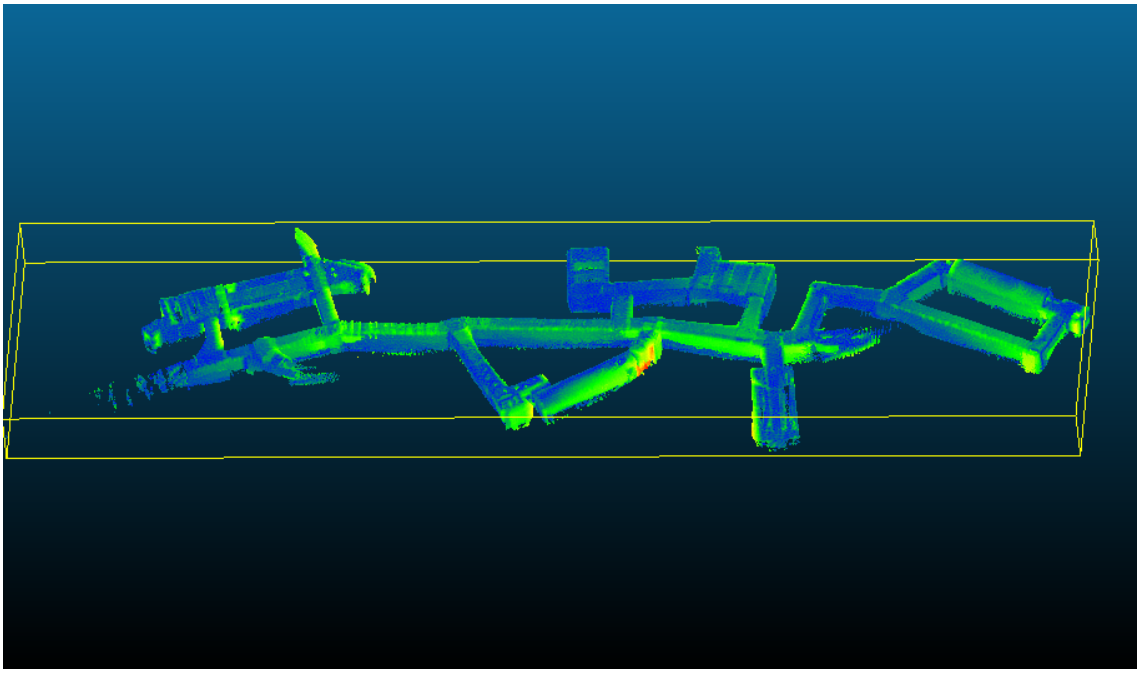


Figure 41: Leighton Battery Tunnels result from difference of points against ground truth

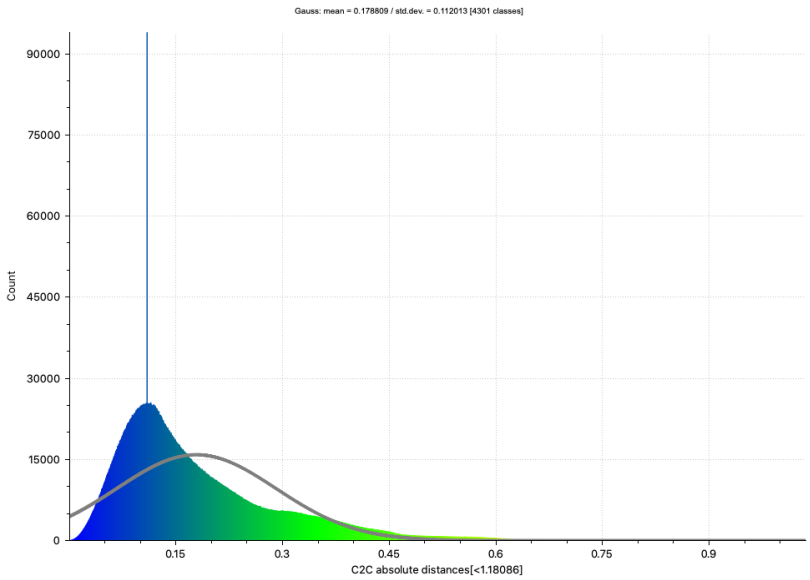


Figure 42: Leighton Battery Tunnels distribution of difference of points against ground truth

6.3.3 Henderson Court

Henderson Court showed that the system was capable of performing accurate SLAM on a large scale, with a mean point cloud to point cloud distance of 0.5m (Figure 44). However, there sections of the map showing in red that contain objects (food trucks) that are present in the data collected but were not present in the survey data. There are also signs of the noise in the map as can be seen by the light green outline to some of the objects in the map including trails of the trees.

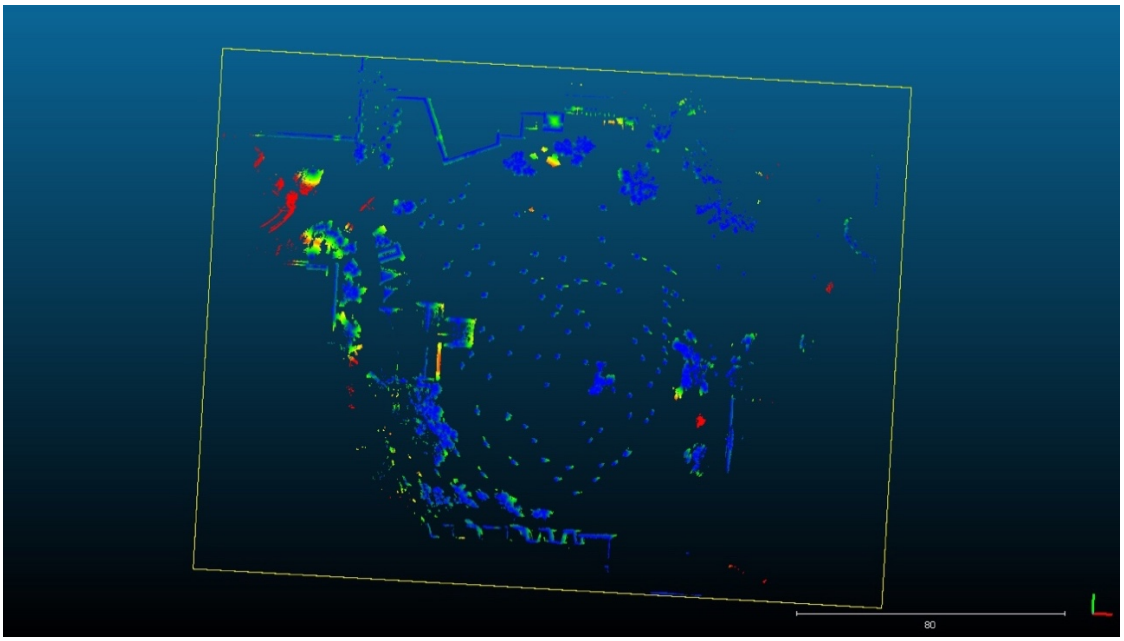


Figure 43: Henderson heatmap of distance between point clouds at sliced level

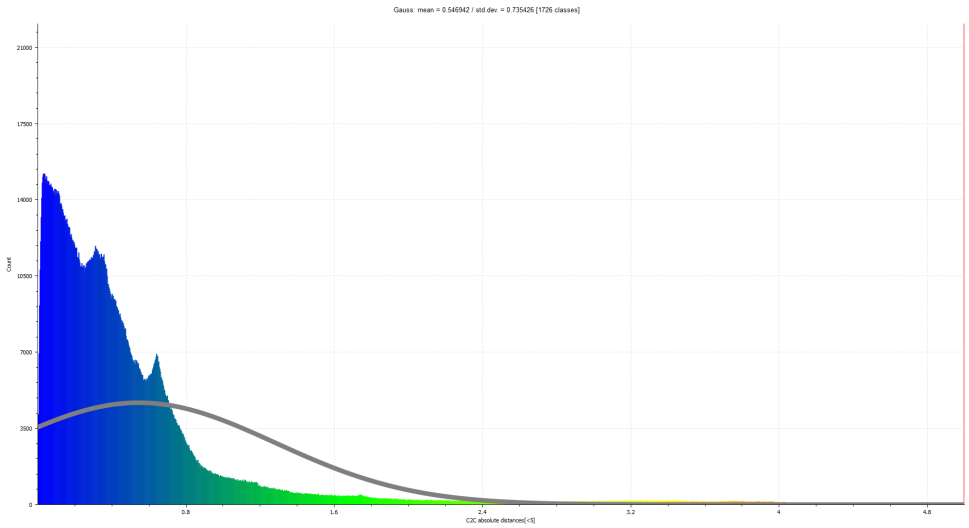


Figure 44: Histogram of distance between point clouds

However, in Figure 45, it can be seen that the centroid for all of the trees from the previous year's scans (red) fall within the centroid of the trees mapped this time. This is likely cause by the noise of the sensor producing the point clouds, it may be resolved by reducing the distance during post-processing as longer distances will likely contain more noise. Another potential cause for the discrepancy may be that the trees have grown, as the ground truth data was collected in 2016, over 3 years prior to this study.

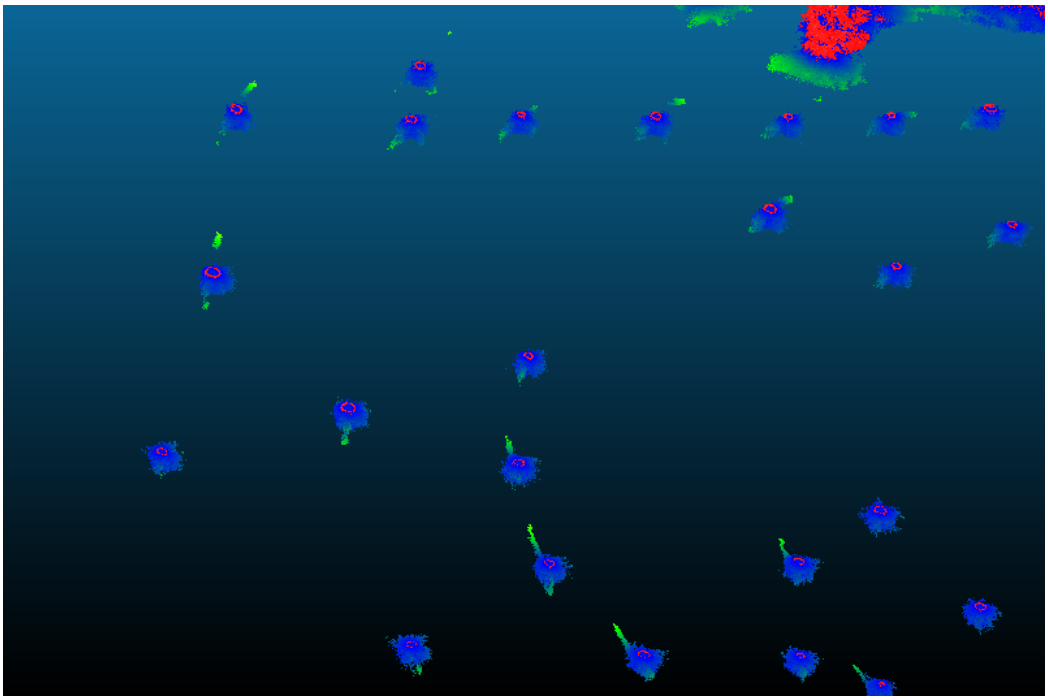


Figure 45: Henderson trees showing discrepancy of tree radius

6.4 Design

The final design of the project is shown in Figure 46. This design was capable of meeting all the aims set out, as can be seen in Table 8.

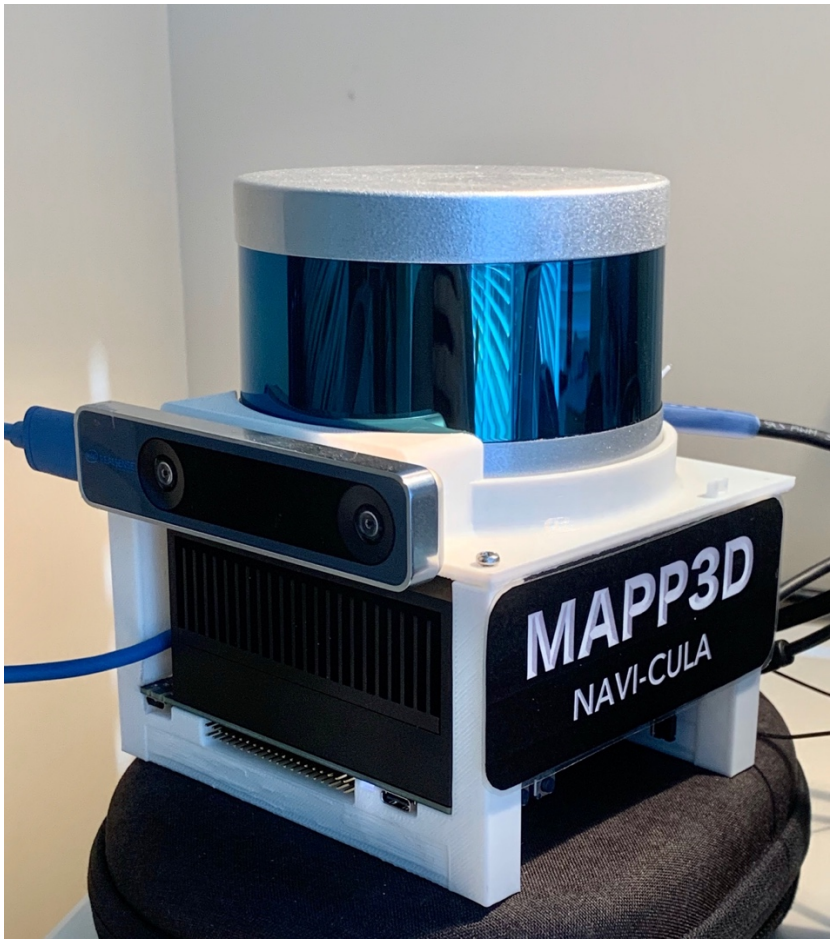


Figure 46: Final design with components

Table 8: Project aims achieved

ID	Explanation	Achieved (YES/NO)
A.1	Research was successfully undertaken which lead to a portable SLAM system	YES
A.2	After reviewing SLAM packages available on ROS, Cartographer was selected	YES
A.3	Design was successful in meeting the requirements for a portable SLAM system	YES
A.4	Construction was successful in meeting the requirements for a portable SLAM system	YES
A.5	The system was able to perform SLAM in three environments without GNSS	YES
A.6	All three areas had their results compared against the ground truth data	YES
A.7	Recommendations have been outlined in the conclusions for future SLAM systems at Curtin University	YES
	TOTAL	7 / 7

The design met 100% of the project aims and was able to perform both real time and post processed SLAM, with a portable system in GNNS deprived areas. These resulting maps were compared against the ground truth datasets provided by the Spatial Sciences department.

Table 9: Explanation of requirements achieved

ID	Explanation	Achieved
R.1	The Xavier contained a 64bit processor and was able to perform real-time SLAM	YES
R.2	The Xavier only had 8GB RAM	NO
R.3	The Xavier had sufficient storage for all scanning and processing (300GB of storage)	YES
R.4	Processing was able to compute real time SLAM	YES
R.5	The device had rounded corners	YES
R.6	The final system was 170x300x180mm and 2.8kg.	NO
R.7	The bottom of the holder was a non-slip surface	YES
R.8	No strap was attached to the device	NO
R.9	iPad connectivity was bright and readable	YES
R.10	iPad and Desktops connected had sufficient screen size	YES
R.11	The device was not drop tested, however assumed it would not be able to withstand a drop.	NO
R.12	Data was able to be transferred via both the remote desktop software and the USB ports on the Xavier	YES
R.13	Yes, the Intel Wi-Fi adapter came with antennas that provided strong connection	YES

ID	Explanation	Achieved
R.14	The display of the remote desktop was able to be adjusted for both bright and low lighting conditions	YES
R.15	All components selected were able to be either purchased or loaned	YES
R.16	All components connected well with each other. Although this sometimes meant purchasing additional parts.	YES
R.17	No areas on the device provided enough heat to cause discomfort.	YES
R.18	All cables were secured neatly with cable ties	YES
R.19	The device was not constructed to be weather-proof	NO
R.20	The lasers emitted from the PUCK are classified as safe for human eyes	YES
R.21	The device was powered via a portable battery	YES
R.22	The Kogan power-bank was capable of running the system for extended periods of time	YES
R.23	System was able to power Xavier (~15W) and PUCK (~8W) at the same time.	YES
R.24	The Kogan power-bank was able to provide USB-C power to the Xavier	YES
R.25	A cable was created to power the PUCK via DC-Jack	YES

R.26	The Xavier was able to power the T265 via USB connection	YES
R.27	NoMachine remote desktop allowed the user to connect remotely to the system	YES
R.28	NoMachine enabled the user to use a virtual keyboard or control the keys from the remote computer.	YES
R.29	A modified Version of Ubuntu 18.04 was installed on the Xavier	YES
R.30	ROS Melodic was able to be installed on the Xavier	YES
R.31	The system was able to perform SLAM using Cartographer.	YES
R.32	The system was able to perform SLAM post-processing with Cartographer	YES
R.33	The system was able to perform real-time SLAM	YES

The design also meets most of the requirements specified, as can be seen in Table 9 and 10. this shows that the design met 81.8% of the set requirements.

Table 10: Summary of project requirements achieved

ID	Achieved	ID	Achieved	ID	Achieved
R.1	YES	R.12	YES	R.23	YES
R.2	NO	R.13	YES	R.24	NO
R.3	YES	R.14	YES	R.25	YES
R.4	YES	R.15	YES	R.26	YES
R.5	YES	R.16	YES	R.27	YES
R.6	NO	R.17	YES	R.28	YES
R.7	YES	R.18	YES	R.29	YES
R.8	NO	R.19	NO	R.30	YES
R.9	YES	R.20	YES	R.31	YES
R.10	YES	R.21	YES	R.32	YES
R.11	NO	R.22	YES	R.33	YES
				TOTAL	**27/33**

Throughout this design process there were several issues encountered with setting up the Xavier. The main issue was that the Intel RealSense SDK would not natively run on the Xavier's ARM64 processor and additional work was required to create a modified kernel patch for Ubuntu to run the SDK. Also, the process of installing OS on the Xavier was very different as the method required using a host computer to flash the OS onto the target computer (Xavier).

Throughout the design process there were several issues with the coordinate transformations required within the system. These issues were encountered as the datasheet for the Velodyne reports the coordinate frames differently to the coordinate frame exported from the velodyne_points topic. Also, another issue was the RealSense IMU coordinate frame needed to be rotated to align with the RealSense pose frame.

CHAPTER 8: CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The designed system and work completed was a success. The system was a portable solution to being able to perform real-time and post-processing SLAM applications. The overall system was able to perform SLAM mapping to produce mean results of 0.33m of the map at the Architecture Building, 0.17m at Leighton Battery Tunnels and 0.5m at Henderson Court, whilst estimated taking ~3% of the time to collect the data (30mins vs 18 hours for Leighton Battery Tunnels). The design of the system is slightly larger and heavier than recommended requirements and future works should consider reducing the size and making the device more ergonomic for future hand-held mapping applications. For more precise data the system should be considered to be placed on a stable platform such as a trolley or vehicle, this will reduce the shaking of the T265 and should improve the pose tracking of the device.

This system was heavily reliant on the Cartographer package for producing SLAM solutions. However, further tuning of Cartographer parameters will lead to improved results of the SLAM processing and maps generated. Further work is required to record data for a comparison against the real-time against post processing SLAM results produced.

Future work should also consider performing the comparisons of the larger point clouds such as Henderson Court on a machine with significant processing power. This is due to the 200 million points per dataset that is trying to be compared.

After encountering many issues configuring the Xavier for this system and reviewing the design choices, the Intel NUC would have been a better fit for this project as the RealSense SDK runs natively on the Intel-based processors.

Work can also be done to change the input to Cartographer from the point cloud exported from the velodyne_points topic to the raw points information collected from the velodyne_node topic. This is due to the velodyne_points collecting a full revolution of points before it can be imported to cartographer, having the raw data would increase the accuracy of the device.

REFERENCES

Aloise, Irvin & Della Corte, Bartolomeo & Nardi, Federico & Grisetti, Giorgio. (2019). Systematic Handling of Heterogeneous Geometric Primitives in Graph-SLAM Optimization. IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2019.2918054.

Cadena. C and L. Carlone and H. Carrillo and Y. Latif and D. Scaramuzza and J. Neira and I. Reid and J.J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age", in IEEE Transactions on Robotics 32 (6) pp 1309-1332, 2016 https://arxiv.org/pdf/1606.05830.pdf

Center, I.R.C., 2005. Human Factors Guidance for the Use of Handheld, Portable, and Wearable Computing Devices.

Di Caro (2017). Lab 1 ROS Intro [online] Web2.qatar.cmu.edu. Available at: https://web2.qatar.cmu.edu/~gdicaro/16311-Fall17/slides/Lab-1-ROS-Intro.pdf [Accessed 31 May 2019].

Dudek G., Jenkin M. (2008) Inertial Sensors, GPS, and Odometry. In: Siciliano B., Khatib O. (eds) Springer Handbook of Robotics. Springer, Berlin, Heidelberg

Elfes, A., 1989. Using occupancy grids for mobile robot perception and navigation. *Computer*, *22*(6), pp.46-57.

Emesent. (2019). *Hovermap*. [online] Available at: https://emesent.io/products/hovermap/ [Accessed 28 Nov. 2019].

Fang, L, Fisher, A, Kiss, S, Kennedy, J, Nagahawatte, C, Clothier, R and Palmer, J 2016, 'Comparative evaluation of time-of-flight depth-imaging sensors for mapping and SLAM applications', in *Proceedings of the Australasian Conference on Robotics and Automation 2016*, Brisbane, Australia, 5-7 December 2016, pp. 1-7. http://www.araa.asn.au/acra/acra2016/papers/pap146s1.pdf

Filatov, A., Filatov, A., Krinkin, K., Chen, B. and Molodan, D., 2017, November. 2d slam quality evaluation methods. In *2017 21st Conference of Open Innovations Association (FRUCT)*(pp. 120-126). IEEE.

Fzheng.me. (2019). *Brief Review on Visual SLAM: A Historical Perspective*. [online] Available at: https://fzheng.me/2016/05/30/slam-review/ [Accessed 31 May 2019].

GeoSLAM. (2019). *ZEB Revo RT - GeoSLAM*. [online] Available at: https://geoslam.com/solutions/zeb-revo-rt/ [Accessed 28 Nov. 2019].

Girardeau-Montaut, D. (2006). Détection de changement sur des données géométriques tridimensionnelles.

Grisetti, G. Stachniss, C and Burgard, W "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34-46, Feb. 2007. doi: 10.1109/TRO.2006.889486

Hong. S, H. Ko, and J. Kim, "Vicp: Velocity updating iterative closest point algorithm," in 2010 IEEE International Conference on Robotics and Automation, May 2010, pp. 1893–1898. http://asrl.utias.utoronto.ca/~tdb/bib/dong_masc13.pdf

Intel® RealSense™ Depth and Tracking Cameras. (2019). *Intel® RealSense™ Depth and Tracking Cameras*. [online] Available at: https://www.intelrealsense.com/ [Accessed 28 Nov. 2019].

Kala, R., 2016. *On-road intelligent vehicles: Motion planning for intelligent transportation systems*. Butterworth-Heinemann.

Kaul, Lukas & Zlot, Robert & Bosse, Michael. (2015). Continuous-Time Three-Dimensional Mapping for Micro Aerial Vehicles with a Passively Actuated Rotating Laser Scanner. Journal of Field Robotics. 33. 10.1002/rob.21614.

Le Cras, J (2013). A Multisensor SLAM for Dense Maps of Large Scale Environments under Poor Lighting Conditions http://hdl.handle.net/20.500.11937/1041

Li, S.-A & Feng, H.-M & Chen, K.-H & Lin, J.-M & Chou, L.-H. (2018). Auto-maps-generation through Self-path-generation in ROS-based robot navigation. Journal of Applied Science and Engineering. 21. 351-360. 10.6180/jase.201809_21(3).0006.

MacKinnon, T. (2019). *Leica Total Station - Surveying from a Known Point - tmackinnon.com - Canadian Geomatics Portfolio*. [online] tmackinnon.com - Canadian Geomatics Portfolio. Available at: https://tmackinnon.com/leica-total-station-surveying-from-a-known-point.php [Accessed 28 Nov. 2019].

Neves Dos Santos, F. (2014). A collaborative, non-invasive hybrid semantic localization and mapping system (HySeLAM). 10.13140/RG.2.1.1862.1289.

NVIDIA. (2019). *Deploy AI-Powered Autonomous Machines at Scale*. [online] Available at: https://www.nvidia.com/en-au/autonomous-machines/embedded-systems/jetson-agx-xavier/ [Accessed 28 Nov. 2019].

Openslam-org.github.io. (2019). *OpenSLAM.org*. [online] Available at: https://openslam-org.github.io/ [Accessed 31 May 2019].

Sammartano, G. & Spanò, A. Appl Geomat (2018) 10: 317. https://doi.org/10.1007/s12518-018-0221-7

Shih An, Li & Feng, Hsuan-Ming & Chen, Kung-Han & Huang, Wen-Hung. (2018). Highly Autonomous Visualization Map-Generation Mobile Robot System Design Through the Robot Operating System Platform. Journal of Imaging Science and Technology. 62. 10.2352/J.ImagingSci.Technol.2018.62.3.030403.

Sprauge, N. (2016). *Coordinate Frames.* 1st ed. [ebook] Available at: https://w3.cs.jmu.edu/spragunr/CS354/handouts/frames.pdf [Accessed 28 Nov. 2019]

Stachniss, C. (2012). *Robot Mapping - WS 2012/13 - Arbeitsgruppe: Autonome Intelligente Systeme*. [online] Ais.informatik.uni-freiburg.de. Available at: http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/ [Accessed 31 May 2019].

Stachniss, C., Frese, U. and Grisetti, G. (2018). *OpenSLAM.org*. [online] Openslam-org.github.io. Available at: https://openslam-org.github.io/ [Accessed 28 Nov. 2019].

Ueland, E. & Skjetne, R. & Dahl, A. (2017). Marine Autonomous Exploration Using a Lidar and SLAM. V006T05A029. 10.1115/OMAE2017-61880.

Velodynelidar.com. (2019). Puck™. [online] Available at: https://velodynelidar.com/vlp-16.html [Accessed 28 Nov. 2019].

W. Hess, D. Kohler, H. Rapp, and D. Andor, Real-Time Loop Closure in 2D LIDAR SLAM, in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016. pp. 1271–1278.

Yun, X. (2015). Utilizing robot operating system (ROS) in robot vision and control.

Zhang, J and Singh, S. Loam: Lidar odometry and mapping in real-time. In Robotics: Science and Systems, volume 2, page 9, 2014

APPENDIX A

Completion of a 4th Year Practical Project

(When completed, students must attach this form with the final report)

Student Name - ----------

Student Number- ----------

4th Year Research Project has now been completed.

1. **All Chemicals have been returned.** **Yes / No**

2. **All Equipment has been cleaned and returned.** **Yes / No**

3. **All Samples and products from the project have been disposed of and/ or dealt with in an appropriate manner.**

   Yes / No

Mechanical / Mechatronics Engg. Technical Staff Member

Name – ----------

Signature- ----------

Date - ----------

APPENDIX B- SLAM DEVICE USER GUIDE

AM Portable Device

USER GUIDE

Harley James Benjamin Pritchard

Table of Contents

Requirements List ***75***

Set-up ***76***

Processing ***82***

IF running real-time cartographer **83**

IF recording for post processing **84**

Requirements List

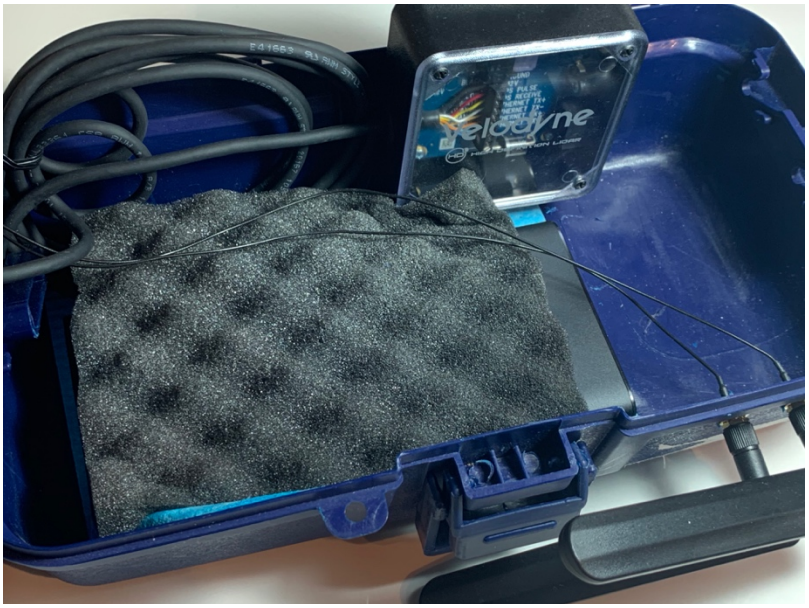
Quantity	Description
1	Velodyne VLP-16
1	RealSense T265
1	NVIDIA Jetson-Xavier
1	Kogan 26800mAh Power-bank (75W)
1	3D printed sensors mount
1	3D printed Xavier mount
1	CAT-6e (Flat) Ethernet Cable
1	USB-C to USB-C Cable
1	USB-A to DC-Jack Power Cable
1	Bottom-Half Fishing Tackle Box
1	$1/8^{th}$ Screw for LiDAR Mount
4	3x6mm Screws
Optional	Cable Ties
Optional	100x100x30mm Foam block

SET-UP

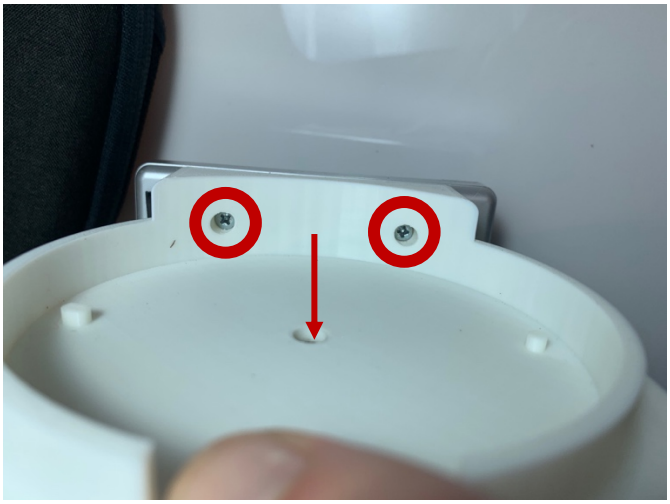
1. Attach Velodyne box and antennas

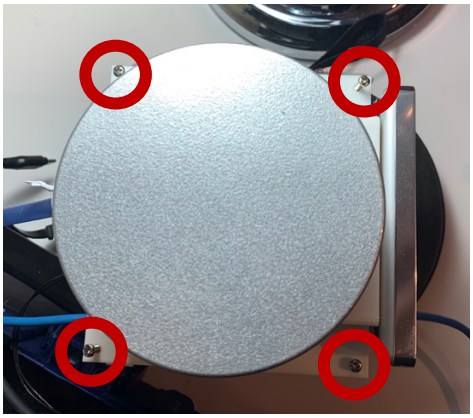
2. Place battery in container

3. Place foam over battery (optional)

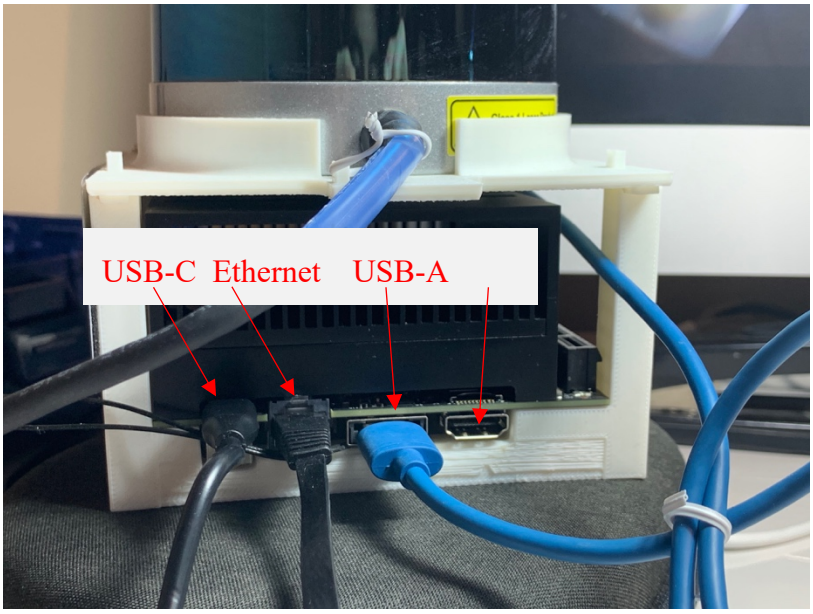


4. Attach sensors to 3D printed mount





5. Attach cables to back of Xavier

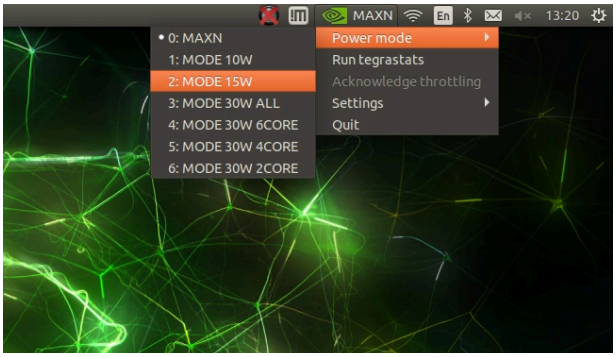


6. Attach power and ethernet cables to Velodyne box

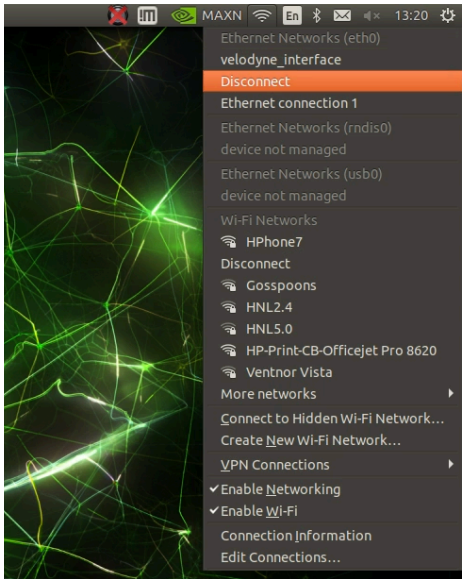
7. Boot system via power button

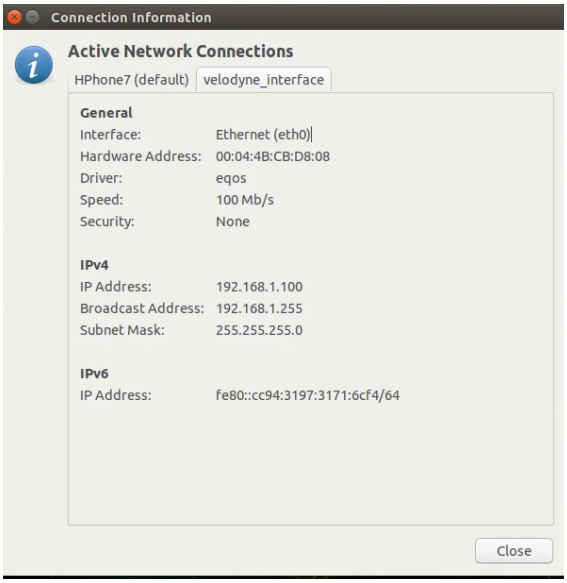
8. Power Velodyne via 12V trigger

9. Change power mode to 15V if running portable mode



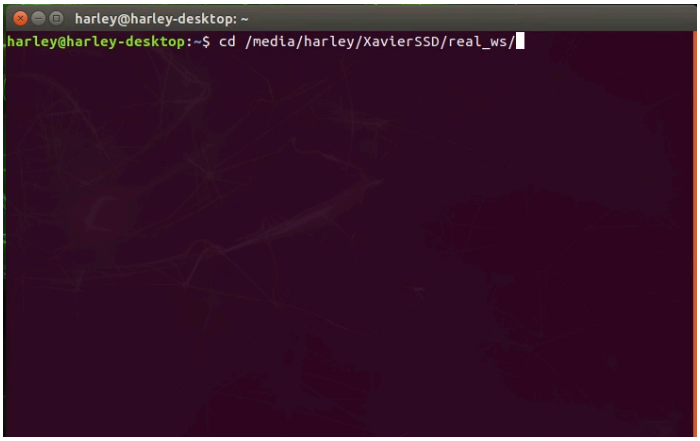
10. Connect Local-network to Velodyne and Wi-Fi network



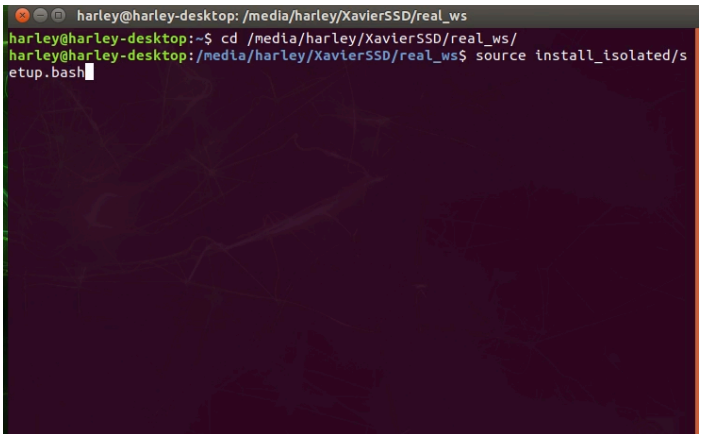


PROCESSING

11. Open terminal and change directory to real_ws

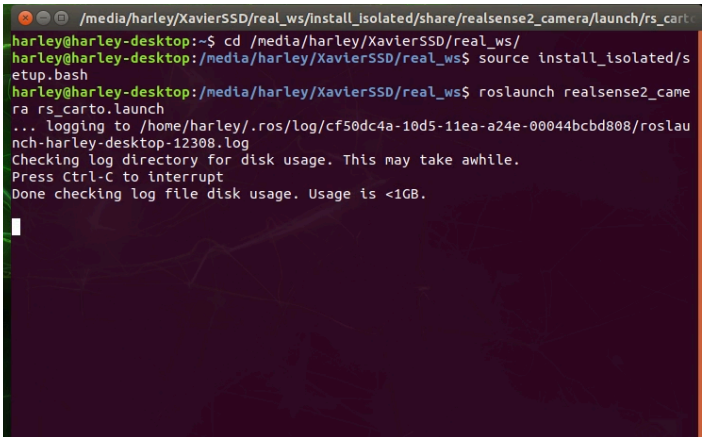


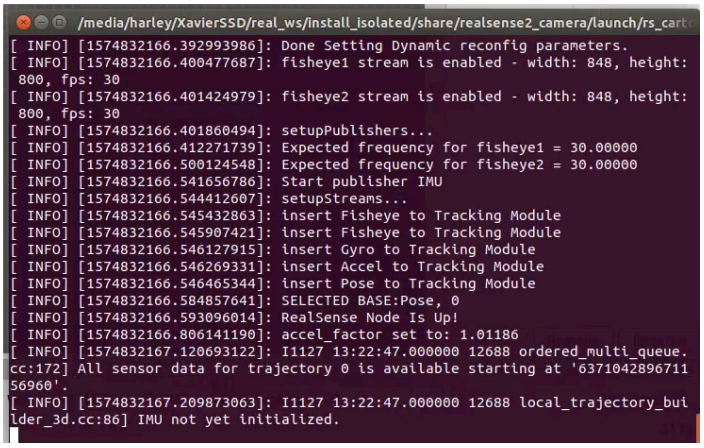
12. Run source install_isolated/setup.bash



IF RUNNING REAL-TIME CARTOGRAPHER

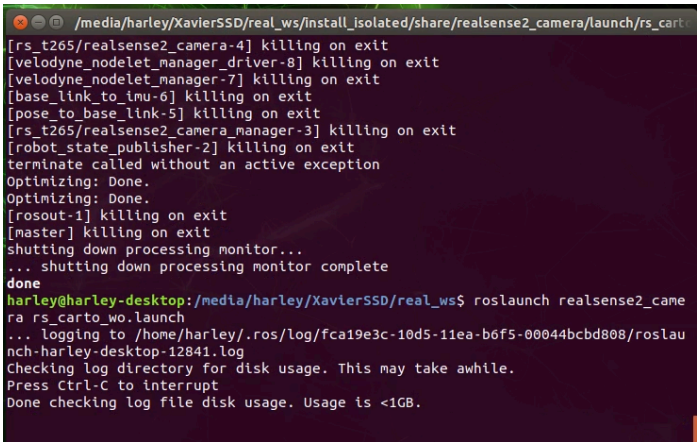
13a. roslaunch realsense2_camera rs_carto.launch



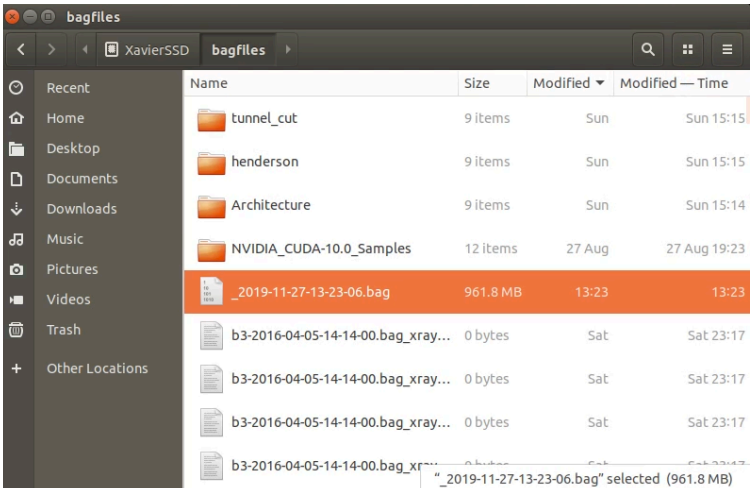


IF RECORDING FOR POST PROCESSING

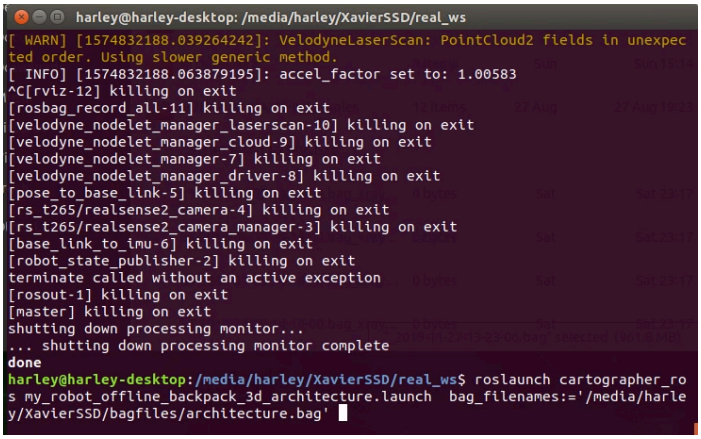
13b. roslaunch realsense2_camera rs_carto_wo.launch

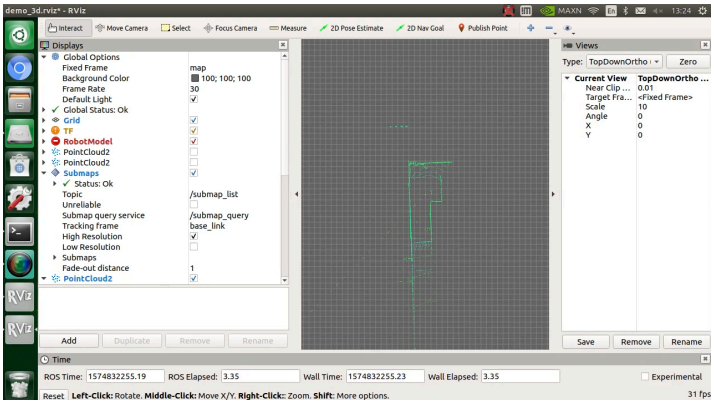


14. Find the rosbag contained here



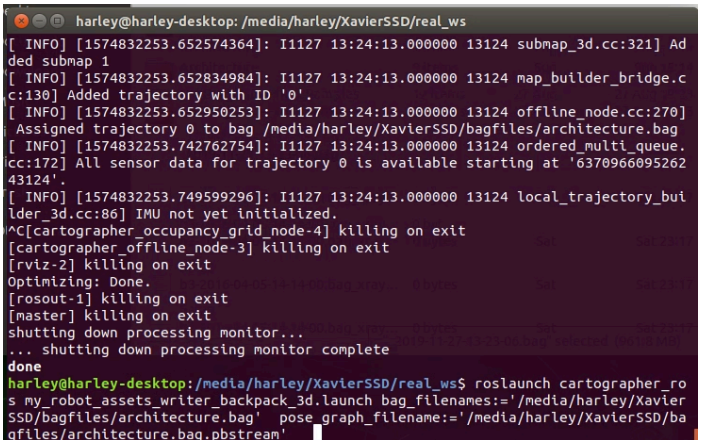
15. To post-process the bag
roslaunch cartographer-ros offline-3d-…launch
bag_filenames: " … "





16. roslaunch assets-writer bagfile:= “…” pbstream:= “…”

note: pbstream file will be “name as bagfile” .pbstream



17. Find results in same directory as rosbag file