GENG4411/GENG4412
Engineering Research Project

Autonomous Shuttle Bus:

Implementation and Management of an RTK-based Navigation System in GNSS-Degraded Areas

Hana Murray Allan
23063586
School of Engineering, University of Western Australia

Professor Thomas Bräunl
School of Engineering, University of Western Australia

School of Engineering
University of Western Australia

Word count: 6772
Date of Submission: October 15, 2025

Declaration of Contribution

My contribution

- Navigation manager system package.
- Extension of GNSS-waypoint navigation system.
- Results recorded, and code used for analysis.

Previous work used as foundation

- nUWAy shuttle bus platform, including:
  - Previous system setup - such as Docker, sensor setup, etc.
  - Previous GNSS-waypoint navigation node.
  - Previous 2D LiDAR SLAM integration.

Collaborative work with other students

- Extension of previous 2D LiDAR SLAM implementation onto new shuttle bus.
- Integration and configuration of sensors onto new shuttle bus.

Use of AI tools

I have used AI tools in the preparation of my report: Yes/No

Details of how AI tools were used:

- Formatting LaTeXdocument, e.g. formatting tables and recommending packages to achieve desired formatting.

Declaration

In accordance with University Policy, I certify that:

The above information is correct, and the attached work submitted for assessment is my own work and that all material drawn from other sources has been fully acknowledged and referenced.

Student signature H. Sullivan Date 13/10/2025

Supervisor confirmation

To the best of my knowledge, the student's contribution outlined above is correct.

Supervisor signature Date 13/10/25

Project Summary

Using Global Navigation Satellite System (GNSS)-based techniques for autonomous navigation and positioning has become a relatively mature and well-implemented area of research. Due to the low-cost nature of GNSS sensors nowadays, especially with the introduction of error-correction algorithms such as Real Time Kinematics (RTK) - which can allow for centimetre-level accuracy, and come built-in to many relatively low-cost sensors - this technique has become highly attractive to both academia and industry alike.

However, in many environments common for autonomous vehicles, both GNSS signals and internet connection, requirements for GNSS-RTK, can become too degraded to be meaningful, or even all together unavailable. For example, navigation systems based on GNSS unreliable in environments like in tunnels, underground, under dense vegetation, and/or within urban canyons created by tall buildings. Their performance can also degrade in regions with strong signal interference, like city centres or university campuses, or during poor weather conditions. These challenges underscore the need for the exploration of alternative methods to overcome this poor performance, including complementary fallback localisation methods to ensure reliable navigation across more diverse environments.

This project aimed to demonstrate that a GNSS based navigation system can have increase reliability with the implementation of a manager system that can switch between GNSS-RTK navigation, and other sensor navigation algorithms, such as Simultaneous Navigation and Mapping (SLAM). Using the University of Western Australia (UWA)'s nUWAy shuttle buses, this project aimed to provide real-world results for an implementation of this navigation manager system based on an RTK-enabled GNSS sensor, and 2D LiDAR SLAM.

While this project was not able to generate results for the navigation system in time for submission, the lessons learned during the development process provide valuable insights into the feasibility and challenges of implementing RTK-navigation on a campus environment, as well as trying to implement dynamic localisation algorithm switching. The results collected during the course of this project highlight how standalone RTK is not a one-stop fix to GNSS inaccuracy.

Additionally, the technical complexities encountered in sensor integration, coordinate frame transformations and computational power requirements highlight the complex nature of being able to dynamically switch localisation methods, in real time, on a limited platform. While theoretically possible, and successfully implemented against recorded data, successful real-world implementation requires careful consideration of computational overhead and package usage to enable smooth transitions without compromising navigation accuracy or system stability.

List of Publications

Submitted for publication:

L. Le, **H. Allan**, and T. Bräunl, "*Evaluation of 3D SLAM on a Shuttle Bus Equipped with Limited Sensors Using Sensor Fusion and Calibration Techniques*," presented at the Australasian Conference on Robotics and Automation (ACRA), 2025.

Acknowledgements

I want to acknowledge and express gratitude to my supervisor, Thomas Bräunl, for allowing me the opportunity to contribute to the nUWAy team. Seeing these shuttle buses on an open day were what drove a lost 17-year-old me to choose this degree, and how great does it feel to come full circle.

I'd like to extend my gratitude to the rest of the REV team - particularly Kieran Quirke-Brown and Lee Le, for their support and advice throughout the year, which has been absolutely invaluable to this project - as well as the other REV final year students, whose camaraderie has made this journey a fair bit more enjoyable.

Lastly, a massive thank you to my friends and family for their unwavering love and support – words cannot express how much it truly means to me.

Contents

Table of Contents

List of Figures

List of Tables

Nomenclature

Notation	Description
2-D	2-Dimensional
3-D	3-Dimensional
AMCL	Adaptive Monte Carlo Localisation
AV	Autonomous Vehicle
CPU	Central Processing Unit
EKF	Extended Kalman Filter
GLONASS	GLObalnaya NAvigatsionnaya Sputnikovaya Sistema
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
HTTP	Hyper Text Transfer Protocol
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LiDAR	Light Detection and Ranging
LIO-SAM	Lidar-Inertial Odometry via Smoothing and Mapping
LOAM	Lidar Odometry and Mapping
LTE	Long-Term Evolution (wireless standard)
Navmgr	Navigation Manager
NDT	Normal Distribution Transform
NRTK	Network Real Time Kinematics
NTRIP	Network Transport of RTCM via Internet Protocol
nUWAy	nUWAy autonomous shuttle buses
OS	Operating System
radar	Radio Detection and Ranging
RAM	Random Access Memory
REV	Renewable Energy Vehicle
ROS/ROS2	Robot Operating System
RTCM	Radio Technical Commission for Maritime Services
RTK	Real Time Kinematics
RTKLIB	Real Time Kinematic Library

Notation	Description
SLAM	Simultaneous Navigation and Mapping
UTM	Universal Transverse Mercator

1 Introduction

1.1 Overview

For an Autonomous Vehicle (AV) to operate effectively, one of the most critical requirements is its ability to localise within its environment [1]. Localisation refers to determining the vehicle's position relative to its surroundings, essential for key navigation tasks such as path planning, obstacle avoidance, and control [2]. To achieve this, autonomous vehicles are equipped with diverse sensors varying in cost, capability, and operational requirements. Commonly used sensors for localisation and mapping include cameras, Inertial Measurement Unit (IMU)/Inertial Navigation System (INS), Light Detection and Ranging (LiDAR), Radio Detection and Ranging (radar), and Global Navigation Satellite System (GNSS) receivers [3]. Each sensor type has distinct strengths and limitations, is prone to different errors, and performs differently across environmental conditions. By leveraging the complementary strengths of multiple sensors, an AV can enhance robustness and maintain functionality across diverse environments.

The use of GNSS for vehicle navigation and positioning is a well-established and widely implemented approach, adopted across various academic disciplines, industries, and consumer applications [4]. On a campus environment, GNSS waypoint navigation becomes attractive it is typically an open, fixed environment - with navigation often happening along fixed routes from common buildings such as lecture halls, libraries and cafes. However, GNSS based systems fall short in areas where GNSS signals becomes unreliable, or purely unavailable. For accurate geopositioning, many GNSS constellation systems require a minimum of five satellites visible [5]. In many environments, this is not feasible, due to signal obstruction or interference. For example, in areas known as "urban canyons", where tall buildings surround the AV, there are very small windows for satellites to make line-of-sight with the AV. For similar reasons, in tunnels, in underground car parks, and during poor weather conditions, GNSS can become too unreliable to use as a primary localisation source [6]. Consequently, this opens an opportunity to explore solutions to counteract the unreliability of GNSS-based systems in certain environments, and make GNSS-based navigation more feasible and robust in said environments.

Real Time Kinematics (RTK) and Precise Point Positioning (PPP) are error correction algorithms that have come about as solutions to the inaccuracy of GNSS-based positioning. With the increasing popularity of these algorithms, and the increased availability of off-the-shelf RTK/PPP-capable GNSS modules, users have the ability to easily increase the accuracy of GNSS modules – allowing for up to centimetre levels of positional accuracy [5]. Due to the increased availability and affordability of GNSS-RTK-capable modules, [4],[5] the adoption of GNSS-based localisation and navigation techniques has become highly appealing within the field of autonomous navigation - both in industry and academia alike. The "out-of-the-box nature" allows users to spend more time on areas such as control, perception and path-planning [7]. Consequently, there exist the opportunity to develop a GNSS-RTK enabled navigation system on the UWA Crawley campus. However, an RTK-enabled vehicle requires a stable internet connection in order to receive corrections from an RTK basestation, therefore in areas where GNSS is poor and/or internet signal is poor, there still exists a need for another localisation system to maintain navigation continuity.

There has been notable research on instead using other perception sensors for localisation – In par-

ticular, LiDAR and vision-based systems have emerged as promising alternatives to GNSS for localisation, each with unique strengths and limitations. LiDAR systems excel in generating accurate 2-D or 3-D spatial data - making them particularly good for SLAM applications - and are effective in low-light conditions and are mostly agnostic to varying weather conditions. However, they are costly, computationally intensive, and can struggle in adverse weather conditions like heavy rain or fog [8], [9]. Camera-based systems, on the other hand, are more affordable and provide rich visual data for tasks, like object recognition and Visual SLAM. Yet, they are highly dependent on lighting conditions and can perform poorly in low-light or featureless environments [10], [2]. These limitations underscore the importance of hybrid approaches that leverage the strengths of multiple sensor modalities for robust and reliable localisation.

Rather, taking a mix-modal approach to localisation could provide an effective method for the localisation of AVs in different environments without a pre-defined map. A system could utilise primarily GNSS-RTK-based localisation and navigation, and when moving into GNSS-denied environments, could dynamically transition to using a different perception sensor localisation algorithm until a good, stable signal is regained. Using the University of Western Australia's Renewable Energy Vehicle (REV) project's nUWAy autonomous shuttle buses, this project aims to show that a GNSS-waypoint navigation autonomous system could be improved using RTK integration and the addition of an intelligent management system and additional fallback localisation methods, based around other sensor algorithms, such as LiDAR SLAM.

1.2 Background

1.2.1 Localisation

Localisation is a fundamental capability for autonomous vehicles, requiring precise pose estimation within the operational environment. This necessitates integration of heterogeneous sensor modalities, each with distinct performance characteristics, costs, and operational constraints. Modern systems typically use multi-modal sensor suites, including cameras, IMU/INS, LiDAR, RADAR, and GNSS receivers. Accurate localisation serves as a prerequisite for higher-level autonomous functions including trajectory planning, obstacle avoidance, and vehicle control.

1.2.2 Pose and Odometry

The pose and odometry of a robot are important descriptors of a robot's current state. The pose of the robot refers to the current position and orientation of the robot. The pose of a 2-Dimensional (2-D) robot is pictured in Figure 1.1. For autonomous vehicles, the pose encompasses both position and orientation within its operational environment. In GNSS-based systems, position is typically represented by a set of latitude, longitude, and altitude coordinates - while orientation can be described using roll, pitch, and yaw angles. For ground vehicles like shuttle buses, that are constrained to planar movement, the pose can be simplified to X-Y position coordinates with a heading (yaw) orientation - which can be thought of as similar to the aforementioned figure.

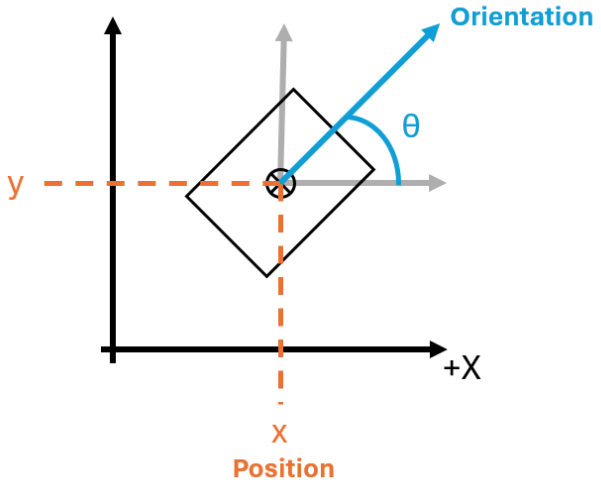


Figure 1.1: Pose

Similarly, a robot's odometry describes its change in pose over time. Odometry can be measured in a number of different ways, using different sensors. Typically, this information is obtained using wheel odometry sensors, which keep track of how far the vehicle has moved, or using IMU modules, which measure the linear and angular velocity and/or acceleration of a robot. Odometry sources tend to be subject to cumulative errors due to wheel slip, sensor noise or drift, and hence tend to be fused with other localisation sources in order to reduce probabilistic error.

1.2.3 GNSS-RTK Localisation

GNSS positioning utilises multiple satellites across different operators (the United States' Global Positioning System (GPS), Russia's GLObalnaya NAvigatsionnaya Sputnikovaya Sistema (GLONASS), Europe's Galileo, China's Beidou, and/or possibly more) in order to triangulate the current position of an object. However, GNSS positioning is affected by numerous sources of error and noise, such as satellite or receiver clock bias and clock error, satellite ephemerides and atmospheric interference [11].

GNSS-based localisation methods are the most common choice, due to the global coverage available, and as several low-cost, accurate mass-market options are available to the average consumer [4]. This makes utilising GNSS as the main form of localisation very appealing to academics, industry and consumers alike. However, as previously discussed, GNSS-based localisation systems can be unreliable in GNSS-degraded environments, making them not applicable to all scenarios.

Real Time Kinematics (RTK), or RTK-GNSS, is a precise differential GNSS positioning system that uses carrier-phase measurements to deliver more accurate positional data to a rover (mobile receiver) [12]. This is typically done using a basestation with a known position, which sends real-time error corrections to an AV about its position [11]. This allows for improved positional accuracy in open-sky environments over normal GNSS positioning, with up to centimetre-level accuracy [13]. RTK requires a stable channel for communication, commonly done using an Network Transport of RTCM via Internet Protocol (NTRIP) server – which uses the Hyper Text Transfer Protocol (HTTP) to send corrections over the internet, in the form of messages using the Radio Technical Commission for Maritime Services (RTCM) protocol [14]. RTK-enabled GNSS receivers have become increasingly popular inclusions into AVs, as they provide a relatively simple way to improve the quality and robustness of AV positioning.

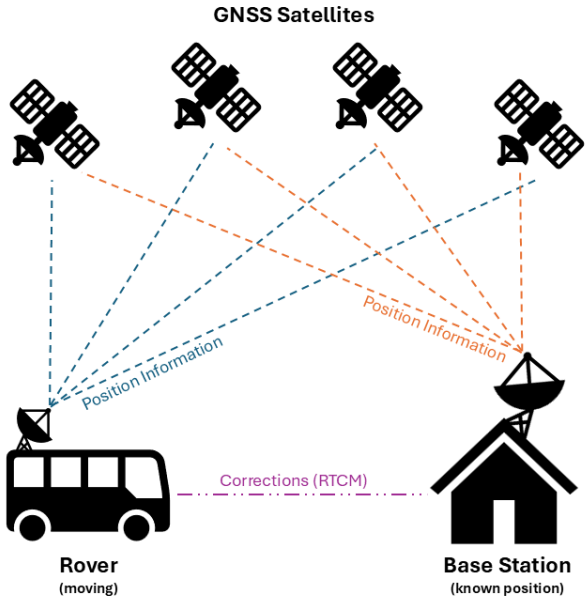


Figure 1.2: Real Time Kinematics

Despite its advantages, RTK-GNSS systems are not without limitations. The accuracy of RTK positioning is highly dependent on the quality of the communication link between the rover and the basestation, as well as the distance between them. As the distance increases, the effectiveness of the error corrections diminishes, with RTK correction data being most effective from within 10–20-km radius from the reference basestation [15]. Furthermore, RTK systems are susceptible to signal obstructions caused by buildings, trees, or other environmental factors, which can lead to degraded performance or complete loss of positioning. To mitigate these challenges, multi-constellation and multi-frequency GNSS receivers are often employed, as they can improve signal availability and reduce the impact of interference. Additionally, Network Real Time Kinematics (NRTK) technology makes RTK more feasible for larger, more expansive environments [16], [17] - However this is out of the scope of this project.

1.2.3.1 Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) is a probabilistic state estimation algorithm that extends a normal Kalman filter to handle non-linear systems by linearising around the current state estimate [18]. Commonly in autonomous navigation and localisation, EKF serves as a sensor fusion framework that combines multiple sources of positioning and odometry data to produce a single, more accurate and robust pose estimate. The filter operates by maintaining a state vector which typically includes the vehicle's position, orientation, and velocities, along with an associated covariance matrix which represents the uncertainty in these estimates.

By fusing complementary sensor inputs, such as GNSS coordinates, IMU measurements, wheel odometry, and visual or LiDAR odometry, the EKF can leverage the strengths of each sensor while mitigating their individual weaknesses. For instance, while GNSS provides absolute positioning but may suffer from signal degradation, IMU data offers high-frequency motion updates but is subject to drift over time [19, **?**].

1.2.4 Simultaneous Localisation and Mapping (SLAM)

Simultaneous Navigation and Mapping (SLAM) is a technique used by autonomous systems to build a map of an unknown environment while simultaneously keeping track of the robot's position within it [20]. SLAM is pivotal for autonomous navigation, enabling robots to operate where pre-existing maps are unavailable or unreliable, as well as to localise to a pre-defined map – However, this project focuses on implementing SLAM without a pre-defined map. The process integrates data from various sensors to create a consistent representation of the environment and the robot's trajectory [20]. SLAM can use various sensors, each with unique strengths and limitations. Common sensors include cameras, LiDARs, and radar systems.

LiDAR-based SLAM is common for AV localisation in GNSS-degraded environments and improves performance in GNSS maps. It is favoured for its ability to generate highly accurate 2-D or 3-D spatial data [21]. Algorithms such as Lidar Odometry and Mapping (LOAM) and Lidar-Inertial Odometry via Smoothing and Mapping (LIO-SAM) process LiDAR data for real-time mapping and localization [21, 22]. LiDAR SLAM performs well in low-light conditions and is less affected by lighting changes than Visual SLAM, though it can be computationally intensive and struggle in sparse or repetitive environments.

1.3 Previous Multi-Localisation Manager Implementations

[7] describes a decision-making algorithm for switching between GNSS-based positioning and cellular (Long-Term Evolution (wireless standard) (LTE)/5G) based positioning for autonomous driving systems, based on simulated sensor data. The paper noted good results, and after 5,000 simulations, the system selected the correct mode 98.2% of the time and was able to maintain a continuous navigation time of 99.7%, compared to 78.3% of the time with GPS alone.

Similarly, [6] discusses a method to switch between GNSS-based and 3-Dimensional (3-D) LiDAR-based self-localisation using Normal Distribution Transform (NDT) scan matching, with an autonomous vehicle for the Tsukuba Challenge 2022. The system self-localises using concurrently using RTK-GNSS, 3-D LiDAR NDT scan matching and gyro-odometry – then a switcher node is used to determine which is the best localisation method and sends it to a localiser. While this method showed improvement over GNSS-RTK localisation alone, the system experienced poor performance due to lack of computing power.

[23] describes a navigation decision-making system based around behaviour trees, which is able to switch between Visual SLAM, LiDAR landmark recognition and GNSS localisation for autonomous navigation around the University of Sydney campus – particularly through tunnels. From provided mapped data, the system resulted in a path very similar to a global reference, and showed improvement over dead-reckoning of GNSS positioning alone.

1.4 Objectives

Table 1.1: Project Objectives

#	Description
O1	Develop an GNSS-RTK waypoint navigator for the nUWAy autonomous shuttle buses to work with developed navigation system. 2.a. Setup RTK basestation on campus. 2.b. Setup nUWAy4 autonomous shuttle buses to enable GNSS-RTK. 2.c. Evaluate feasibility of RTK navigation on campus. 2.d. Evaluate feasibility of extending existing waypoint navigation system on nUWAy shuttle buses to take GNSS-RTK waypoints.
O2	Extend pre-existing 2-D LiDAR SLAM system on nUWAy shuttle buses to work with GNSS-RTK coordinate waypoints and developed navigation system.
O3	Develop a deterministic navigation manager system which allows a primarily GNSS-RTK localising autonomous shuttle bus to dynamically fallback to LiDAR SLAM localisation during areas with degraded GNSS signals.

2 Design Process

2.1 Constraints, Requirements and Evaluation Criteria

2.1.1 Constraints

Table 2.1: Design Constraints

#	Constraint
C1	The system was restricted to the nUWAy4 shuttle bus, due to required sensor suite - more in section 2.2: Design Tools.
C2	Testing was restricted to areas depicted in Figure 2.1, due to physical limitations of the nUWAy shuttle buses. Evaluation and testing of localisation systems would be restricted to the path in Figure 2.2 after evaluation of the available campus map in order to test both GNSS-RTK favourable conditions, and GNSS-degraded conditions.
C3	Shuttle buses could not be operated during poor weather conditions and night-time operation was kept to a minimum, reducing testing opportunities and repeatability.
C4	Onboard computational resources on the nUWAy4 shuttle constrained ability to run sensor, localisation, mapping and managerial nodes at the same time - more in section 2.2: Design Tools.
C5	The bus requires a stable internet connection at all times in order to receive an RTK corrections stream.
C6	Physical access to the RTK basestation infrastructure was constrained by university campus management protocols, limiting configuration flexibility, as well as requiring university IT services approval for network integration - see section 2.2.2: Real Time Kinematics (RTK) Basestation.
C7	The system was limited to ROS2 version Humble Hawksbill, restricting compatibility with some open-source packages and influencing design decisions.
C8	System feedback relied primarily on ROS2 terminal output, and RQT and RVIZ visualisation, which provided limited clarity for real-time debugging and situational awareness.

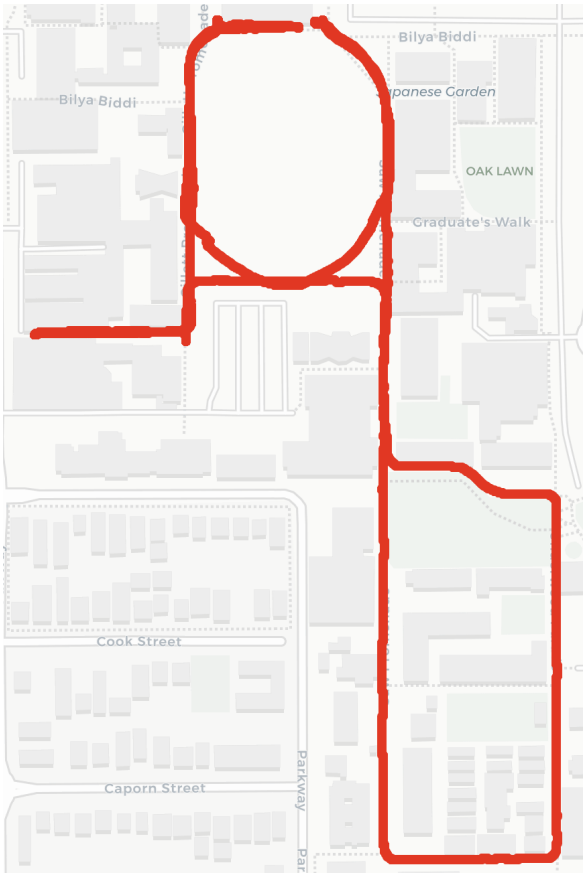


Figure 2.1: Available Driving Path on UWA Crawley Campus

Figure 2.2: Test Path

2.1.2 Requirements

Table 2.2: Design Requirements

#	Requirement
R1	The navigation system will take GNSS latitude and longitude coordinates to navigate through, both in GNSS-RTK mode and SLAM mode.
R2	The navigation system will not require a fixed, pre-defined map for navigation outside of GNSS coordinates.
R3	The navigation system will be able to avoid obstacles and navigate safety through a fixed path.
R4	The navigation system should re-localise itself correctly and continue navigation seamlessly without human intervention.
R5	The navigation system should be able to operate on the bus's main PC alone.

2.1.3 Evaluation Criteria

Table 2.3: Evaluation Criteria

#	Criteria
E1	Feasibility of GNSS-RTK based navigation will be evaluated against values such as: 1. Average and median positional accuracy, in form of covariance data. 2. Maximum positional accuracy. 3. Minimum positional accuracy. 4. Mapping of covariance data to find areas where GNSS-RTK performs best.
E2	Autonomous navigation system will be evaluated against qualitative values such as: 1. Positional accuracy of waypoint path following. 2. Number of disengagements and/or interventions. 3. Latency during localisation method transitions. 4. Continuity of localisation.
E3	Navigation system will be evaluated for usability by the end user, including: 1. Ease of deployment and implementation on other nUWAy shuttle buses. 2. Observable and transparent system monitoring, including current position, waypoint, positional covariance and node states. 3. Runtime stability.

2.2 Design Tools

2.2.1 nUWAy Shuttle Buses

The nUWAy autonomous shuttle buses (nUWAy) are a set of fully electric, drive-by-wire EasyMile EZ10 shuttle buses capable of autonomous driving. The shuttles belong to University of Western Australia's (UWA) Renewable Energy Vehicle (REV) project. Currently, there are four shuttles, three on campus, and one in Perth's Northern suburb of Eglinton.

Figure 2.3: nUWAy4 Autonomous Shuttle Bus

The shuttles differ between them as nUWAy1 and nUWAy2 are Gen1 EZ10 shuttles, while nUWAy3 and nUWAy4 are Gen2 - with markedly different internal wiring and layout. Additionally, between shuttles of the same generation, there are differences between their sensor configurations. Due to the differences between shuttles, and a limitation in resources, it was noted that focusing the design and testing to one shuttle (nUWAy4) would be important to meet the scope of this project - as mentioned in section 2.1.1: Constraints. Over the course of this project, nUWAy4 was outfitted with sensors that better suited the requirements of this project, with more on this later in this chapter. Additionally, the different shuttle buses are outfitted with different main PCs. The details of the relevant sensor suite and PC specification have been detailed in Tables 2.4 and 2.5 respectively.

Table 2.4: Relevant nUWAy4 Sensor Details

Sensor	Details	Purpose
LiDARs	2x 3-D Velodyne VLP-16 (16-layer) 2x SiCK LD-MRS (4-layer) 4x SiCK LMS-151-10100 (single-layer)	Localisation and 3-D Vision Localisation Safety
GNSS *INS*	SBG Ellipse D dual antenna	Positioning and Localisation Orientation and Odometry

Table 2.5: nUWAy4 Onboard PC Specifications

Component	Specification
CPU	Intel Core i7-6700TE @ 2.4GHz
GPU	Integrated Intel HD Graphics 530
RAM	16 GB DDR3
OS	Ubuntu 22.04

The nUWAy4 shuttle bus system architecture and sensor pipeline is illustrated in Figure 2.4. Sensor data from the previously described hardware is processed by the shuttle's main PC, which interfaces

with the vehicle's control systems via CAN bus communication. During autonomous operation, a PlayStation 4 controller provides deadman-switch functionality that enables or disables autonomous driving functionality. The system pre-processes the LiDAR data, combining pointcloud data from the front and rear LiDARs into a unified 3-D representation before converting it to a 2-D laser scan format for subsequent perception and navigation tasks.

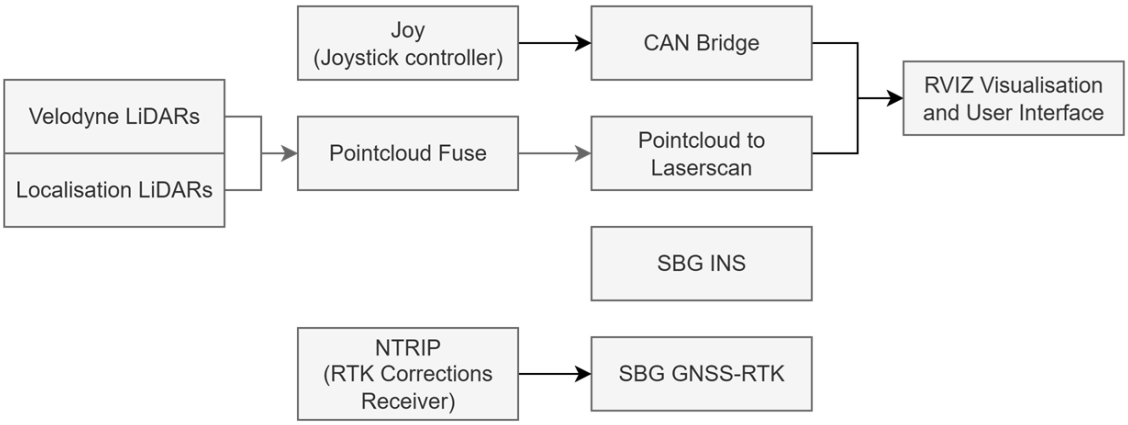


Figure 2.4: nUWAy4 shuttle bus system architecture

2.2.2 *Real Time Kinematics (RTK) Basestation*

A basestation with a known position is required to enable GNSS-RTK positioning, as RTK corrections require a . As RTK errors are dependent on an AVs distance from its base, with single-base RTK, a rover should not exceed a 20-km radius away from the basestation [15]. Thus it was concluded that a basestation would need to be set up on campus, created specifically for the nUWAy autonomous shuttle buses. The basestation was built and set up on the roof of EZONE engineering building on UWA's Crawley campus - shown in Figure 2.5.

Figure 2.5: Location of RTK Basestation

The basestation is built around a Raspberry Pi 4B running Ubuntu Server 22.04, equipped with a RTK-capable Sparkfun GPS-RTK module featuring a u-blox ZED F9P receiver. Initially, the receiver was configured through u-blox's software suite to enable RTCM error correction message transmission. A 24-hour survey period was conducted to determine the precise known position of the receiver, which is essential for accurate RTK corrections. Following identification of its fixed position, the basestation was set up using RTKExplorer's fork of Real Time Kinematic Library (RTKLIB) [24] to process the RTCM messages calculated by the u-blox receiver.

Table 2.6: Campus RTK Basestation Details

Component	**Details**
Mainboard	Raspberry Pi 4B 2-GB
GNSS-RTK Receiver	Sparkfun GPS-RTK-SMA with u-blox ZED-F9P
RTK Publisher	RTKExplorer RTKLIB
OS	Ubuntu Server 22.04
Specification	**Details**
Available Satellites	GPS, GLONASS, Galileo, BeiDou
Number of Concurrent GNSS	4
Horizontal Position Accuracy	0.010-m (with RTK)

The processed correction data is subsequently transmitted to an NTRIP server mountpoint provided by RTK2GO, creating an RTK corrections service accessible to compatible GNSS receivers. This configuration enables the basestation to provide real-time RTCM correction messages to any RTK-capable robot within its operational range, including the nUWAy shuttle buses, hence establishing the foundational infrastructure for precise GNSS-RTK positioning across the Crawley campus.

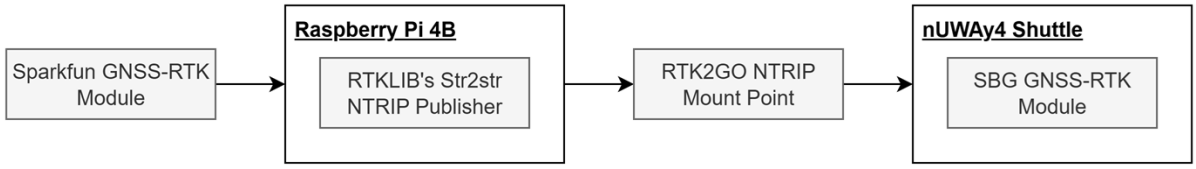


Figure 2.6: RTK Basestation Corrections Pipeline

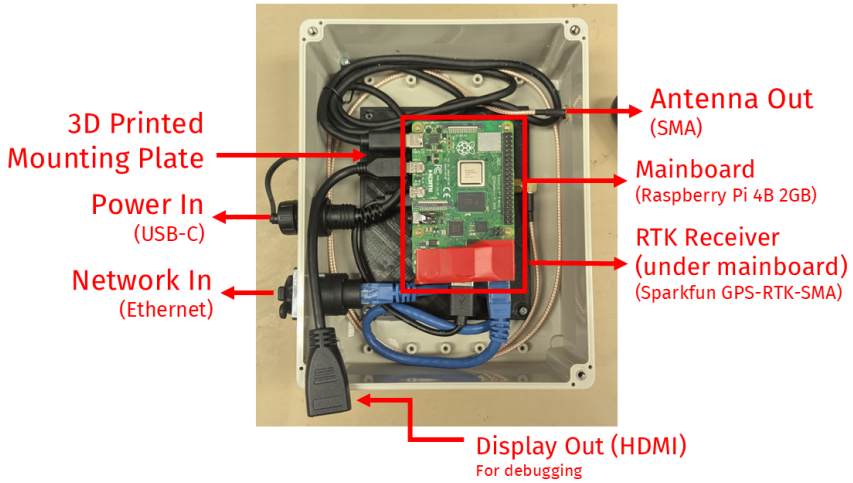


Figure 2.7: RTK Basestation Internals

2.2.3 GNSS Module

To evaluate the feasibility of GNSS-RTK navigation on campus, it was essential to assess the accuracy of available GNSS modules aboard the nUWAy4 shuttle bus. Initially, the shuttle was equipped with a Novatel FlexPak 6 RTK-compatible GNSS receiver. However, preliminary testing revealed that this module could not provide sufficient accuracy for reliable autonomous navigation, even with RTK corrections enabled. Given the positive performance of an SBG Ellipse D GNSS/INS module in previous applications, this unit was subsequently installed on nUWAy4 for more detailed evaluation. The relevant specifications of each unit are listed in section B.0: GNSS Module Specifications

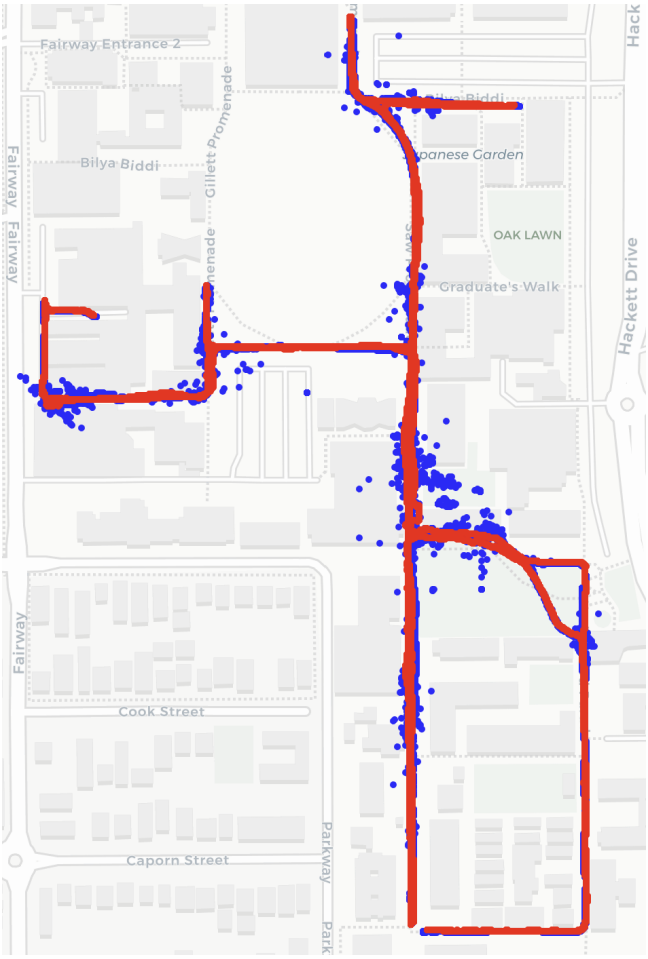


Figure 2.8: Comparison of Novatel FlexPak 6 (blue) and SBG Ellipse-D (red) GNSS module positional accuracy.

Table 2.7: Novatel and SBG GNSS-RTK Results

Combined Covariance[a] [m]	**Novatel Flexpak 6**	**SBG Ellipse-D**
Average	1087	0.4291
Median	0.3756	0.0635
Standard Deviation	5829	0.0635
Mode	27190	0.0002
Minimum	0.0001	0.0002
Maximum	136200	126.4

[a] Combined covariance was calculated as the average between the latitudinal and longitudinal covariances

Comparative testing was conducted with both sensors operating simultaneously along identical routes, and utilising RTK error corrections. Both modules were run using their respective ROS2 drivers. The results, illustrated in Figure 2.8, demonstrate the relative performance characteristics of each system. The positional covariances of each module along the test route were analysed, with Figures 2.9a and 2.9b presenting spatial distributions of positional uncertainty.

(a) Novatel FlexPak 6 GNSS-RTK module positional covariance

(b) SBG Ellipse-D GNSS-RTK module positional covariance

Figure 2.9: Novatel and SBG GNSS-RTK module comparison

Analysis of the Novatel data revealed significant reliability issues. The unit exhibited covariance spikes exceeding 1000-metres across extended sections of the test route, with some readings displaying negative covariance values – mathematically impossible given that covariance represents squared measurements. These anomalies suggest underlying sensor problems, potentially including sensor noise, internal onboard error correction failures, or compatibility issues with the ROS2 driver.

When outlier values above 1000-metres were filtered from the dataset (Table 2.8), the Novatel demonstrated improved performance metrics approaching those of the SBG module. However, this data cleaning process does not address the reliability concerns observed during operation, and additional pre-processing would have been required to filter data in real time. The frequent occurrence of extreme outliers renders the sensor unsuitable for the GNSS-RTK-based navigation requirements this project demands.

In contrast, the SBG GNSS-RTK provided the expected sub-centimetre accuracy of an RTK-enabled receiver, alongside reliable, consistent results indicated by the mode combined covariance. Given these limitations and reliability concerns, the SBG Ellipse-D was selected as the primary GNSS-RTK receiver for this project.

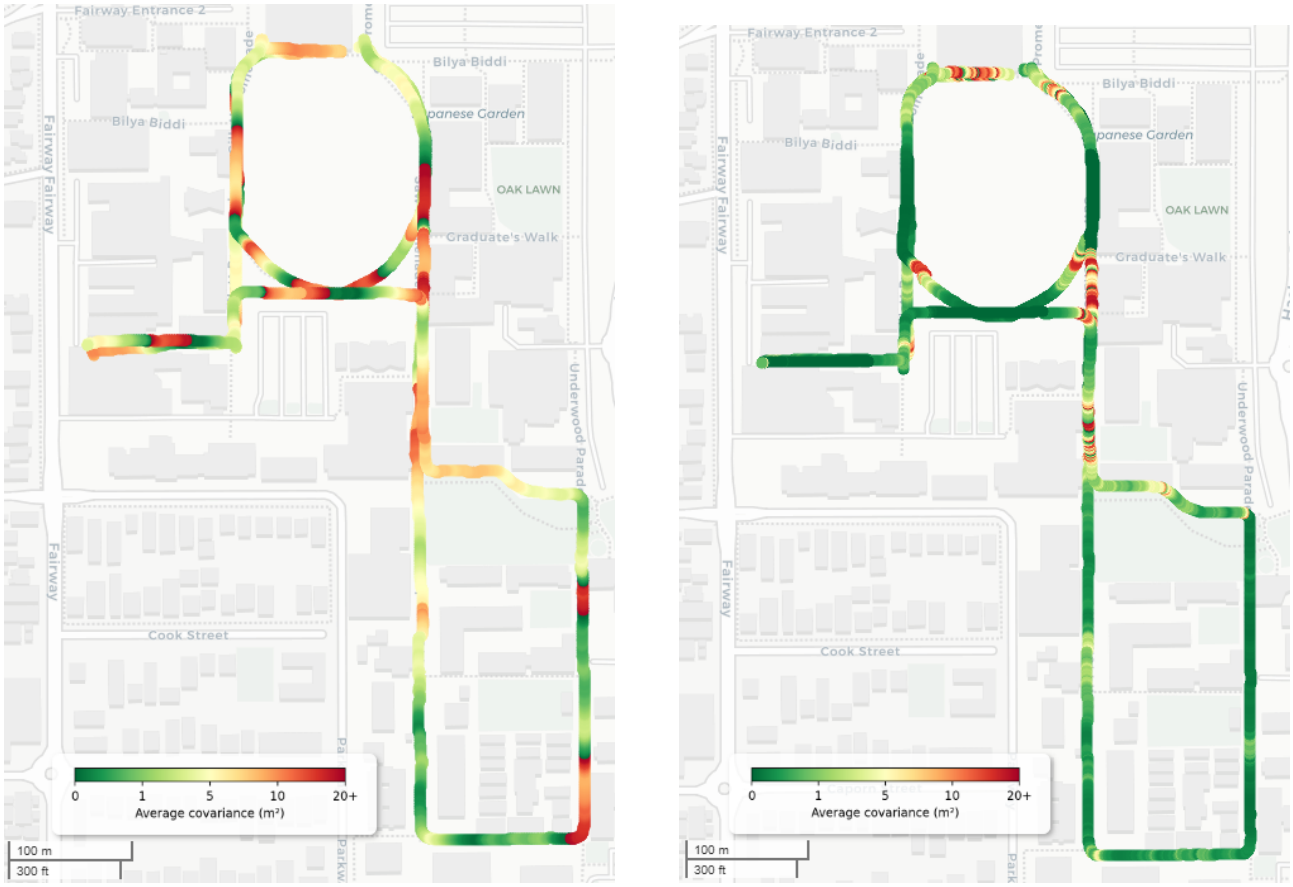
Table 2.8: Novatel, Filtered Novatel, and SBG GNSS-RTK Results

Combined Covariance[a] [m]	Novatel Flexpak 6	Filtered Novatel Flexpak 6	SBG Ellipse-D
Average	1087	13.31	0.4291
Median	0.3756	0.3146	0.0635
Standard Deviation	5829	69.10	0.0635
Mode	27190	2.591	0.0002
Minimum	0.0001	0.00006	0.0002
Maximum	136200	998.16884	126.4

[a] Combined covariance was calculated as the average between the latitudinal and longitudinal covariances

2.3 SBG GNSS vs. GNSS-RTK

Once it was decided that the SBG was the better option for a RTK-enabled GNSS receiver for this project, an evaluation of regular GNSS positioning and GNSS-RTK positioning was done to evaluate the feasibility of GNSS-RTK waypoint navigation as a worthwhile approach. The covariance map along a fixed route on campus can be seen in Figure 2.10, covering the whole drivable path by the nUWAy shuttles.



(a) SBG GNSS-only mapped positional covariance

(b) SBG GNSS-RTK mapped positional covariance

Figure 2.10: SBG GNSS-RTK Comparison

Table 2.9: SBG GNSS and GNSS-RTK Results

Combined Covariance[a] [m]	**GNSS**	**GNSS-RTK**
Average	1.652	0.4855
Median	1.195	0.0642
Standard Deviation	1.853	2.725
Mode	0.8152	0.0002
Minimum	0.3842	0.0002
Maximum	55.10	89.91

[a] Combined covariance was calculated as the average between the latitudinal and longitudinal covariances

The implementation of GNSS-RTK demonstrated overall improvements in positioning accuracy compared to standard GNSS positioning - as seen in Figure 2.9 and Table 2.9. The average and median positional covariance showed approximately 70% and 95% improvements respectively, while the mode covariance indicated enhanced consistency and reliability in the GNSS-RTK data. These improvements provide strong evidence that RTK corrections significantly enhance positioning performance.

However, both the standard deviation and maximum covariance values increased when using RTK, suggesting greater variability in the measurements. This increase can be attributed to the data collection methodology, as the SBG internally incorporates the RTK correction stream into its output so GNSS and GNSS-RTK datasets had to be collected at different times - thus it is possible the datasets experienced different satellite visibility conditions, network stability, and atmospheric variations, which could have contributed to these variations.

Despite the increased variability, the substantial improvements in mean, median, and mode covariance values demonstrate that GNSS-RTK positioning offers significant advantages for autonomous navigation systems. These results provide compelling evidence that RTK-enabled positioning systems are well-suited for deployment in open campus environments such as UWA's Crawley campus, where reliable centimetre-level accuracy is essential for autonomous vehicle operations.

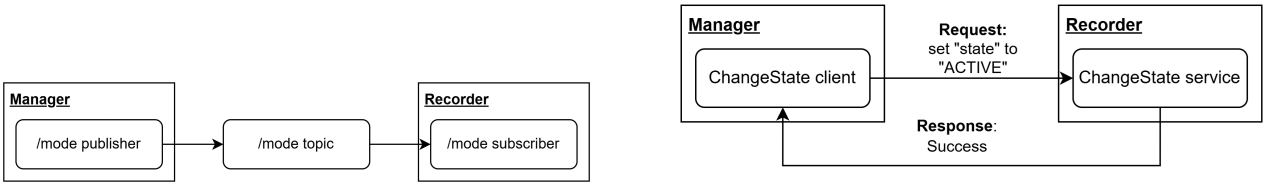
These results also informed the selection of the test route for evaluating the navigation system (Figure 2.2). The chosen path from the Robotics Lab to the Social Sciences building allows testing within diverse GNSS conditions: an urban canyon near the Robotics Lab with moderate positional covariance, open-sky environments along James Oval carpark and Social Sciences with optimal covariance, and areas with dense foliage causing signal degradation at the turns between the Robotics Lab and James Oval, and near Guild Village. This varied environment was expected to provide an ideal testbed for evaluating the system's ability to transition from GNSS-RTK mode during signal degradation and assess recovery performance upon returning to areas with reliable GNSS reception.

2.3.1 ROS2

Robot Operating System (ROS/ROS2) is an open-source robotics software toolkit for designing and implementing complex robotic systems. It provides a large repository of modules such as hardware drivers and algorithms. ROS2 Humble Hawksbill is used as the framework behind the autonomous functionality of the nUWAy shuttle buses at the time of writing.

In ROS2, software is organised into modular components called nodes, which perform specific tasks such as sensing, control, or decision-making. ROS2 provides several interfaces for node commu-

nication, including topics and services. A topic uses a publisher-subscriber model where one node publishes data (messages) and others subscribe to receive it - useful for long-term, asynchronous data readings like sensor or mode readings. On the other hand, a service enables request-response communication between nodes, where a client sends a request to a service server and waits for a reply. This is ideal for short, discrete interactions such as triggering actions, changing parameters, or requesting function calls.



(a) ROS2 publisher-subscriber model *(b) ROS2 service-client model*

Figure 2.11: ROS2 Interfaces

2.3.1.1 Lifecycle Nodes

ROS2 provides a type of node known as Lifecycle Nodes [25]. These allow fine control over node states, including configuration, activation, deactivation, cleanup, and shutdown. There are four primary states: `UNCONFIGURED`, `INACTIVE`, `ACTIVE`, and `FINALIZED`, which transition through six different states. Transitions can be triggered by other nodes using the `ChangeState` service. Regular ROS2 nodes cannot be converted into lifecycle nodes, so it was important to find compatible packages or build nodes with this architecture from the ground up.

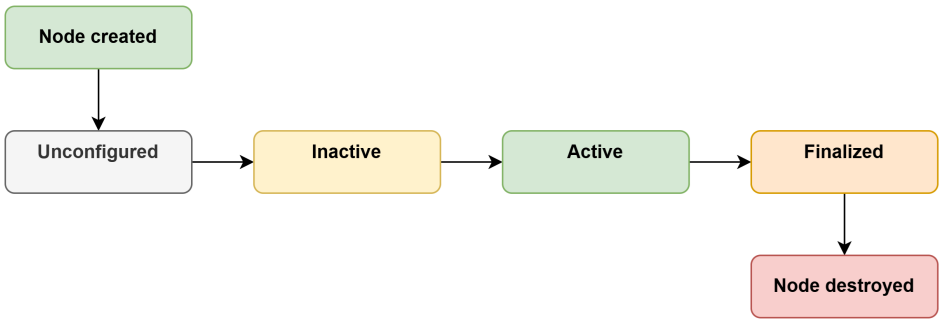


Figure 2.12: ROS2 Lifecycle Nodes

2.3.1.2 RVIZ

RVIZ is a ROS2 package for visualising sensor data, robot models, and navigation processes. It provides an interactive interface to observe how components of a robot's software stack interact during operation. Users can also create custom panels for direct interaction and control. For instance, the nUWAy shuttle buses use a custom panel to configure and activate the lifecycle node that manages the CAN bridge driver.

2.3.1.3 SLAM Toolbox

`Slam_toolbox` is a ROS2 package that provides tools for 2-D SLAM localisation and mapping [26], using an improved version of the Karto SLAM algorithm. It supports lifecycle nodes for dynamic activation and deactivation, making it suitable for this project where localisation methods are

switched dynamically. However, the ROS2 Humble version required a forked build for compatibility with the nUWAy shuttles, which created additional complexity and had to be managed correctly in order to not conflict with other `slam_toolbox` installations on the nUWAy shuttles, as to not effect other students' projects.

2.3.1.4 Robot Localization

Robot Localization [27] is a ROS2 package that provides state estimation through Extended and Unscented Kalman Filters. It enables fusion of multiple positioning and odometry sources for robust pose estimation. The `ekf_node` fuses data from IMUs, GNSS receivers, and various odometry inputs.

The `navsat_transform_node` integrates GNSS data into the robot's local frame by converting global coordinates (latitude, longitude, altitude) into the Universal Transverse Mercator (UTM) system, with reference to a specific datum. This transformation provides the crucial `[map]->[odom]` link anchoring local odometry to a global frame. Combined with the EKF node, it fuses GNSS accuracy with high-frequency sensor data for reliable localisation.

2.3.1.5 Navigation2 Stack

The Navigation2 (Nav2) stack is a ROS2 platform providing a comprehensive framework for autonomous navigation [28]. It integrates perception, planning, control, localisation, and visualisation into modular servers that communicate via ROS2 interfaces. Nav2 was chosen for this project for its flexibility and ability to simplify low-level setup tasks such as control and path planning.

2.3.1.6 Rf2o Laser Odometry

Continuous odometry data is required to maintain position estimates during localisation transitions. When switching between GNSS-RTK and SLAM, GNSS odometry briefly becomes unreliable due to high covariance, requiring a secondary odometry source. As `slam_toolbox` does not provide standalone odometry, alternatives were explored. Wheel odometry proved too computationally expensive for the nUWAy4 onboard PC.

The `rf2o_laser_odometry` package was selected to provide laser-based odometry by comparing consecutive LiDAR scans [29]. It offered low computational overhead, compatibility with existing sensors, and allows the 2-D LiDAR SLAM to be better coupled with the GNSS data. The laser odometry was fused with GNSS and IMU data using the EKF node. During localisation switching, continuous laser odometry maintained positioning estimates, improving robustness and system stability.

2.3.2 Docker

Docker is a containerisation platform used to deploy modular, reproducible environments with defined operating systems, packages, and configurations. On the nUWAy shuttle buses, it runs separate containers for sensors and functionalities, managed through Docker Compose. When configured correctly, these containers and their ROS2 nodes communicate seamlessly as if part of one system. The pre-existing setup, created by previous students, was extended to meet this project's requirements.

2.4 Previous Work

Some system components were inherited from previous nUWAy teams, including parts of the localisation pipeline. These modules were assumed functional based on early discussions and demonstrations, but were later found to contain configurations that were incompatible with this project's requirements. Additional validation and adjustment were needed before integrating them with the new localisation-switching framework.

2.4.1 Waypoint Navigation

The Navigation Manager incorporates a waypoint navigation system built on the Navigation2 `BasicNavigator` API. A pre-existing node by a previous student, `bus_follow_path`, which was operational on nUWAy1 and nUWAy2, processed waypoint files containing global odometry coordinates and headings, converting them into goal poses for Navigation2 path planning and control. To integrate with the Navigation Manager, it had to be refactored to support lifecycle functionality and dynamic activation. The input format was modified to accept GNSS coordinates and ensure compatibility between the two localisation methods.

2.4.2 SLAM Navigation

A working 2-D LiDAR SLAM navigation system was successfully implemented on the first generation nUWAy1 and nUWAy2 shuttles. It was assumed to be compatible with nUWAy4's sensor suite but would likely require additional validation and extension to support GNSS waypoint input before integration with the Navigation Manager system.

2.5 Considered Designs

2.5.1 RTK Waypoint Navigation

It was initially expected that only small modifications were needed to adapt the existing system for GNSS waypoint navigation. A simple `(latitude, longitude, heading)` to `(x, y, heading)` conversion was assumed sufficient. However, the original system relied on Nav2's Adaptive Monte Carlo Localisation (AMCL) server (see Appendix A.0), which requires a predefined occupancy map and was therefore incompatible. The navigation architecture was redesigned to work without a predefined map, using a pose-to-pose approach instead of path following.

2.5.2 Localisation Algorithm Switching

As discussed in section 1.2.1: Localisation, localisation anchors the vehicle's pose to a map, enabling movement tracking. In ROS2, this is represented through transform trees:

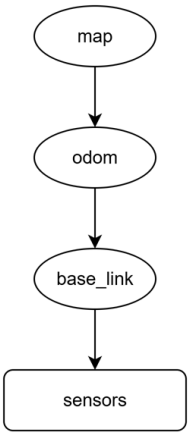


Figure 2.13: ROS2 transform tree example

The `[map]->[odom]` transform is fundamental to many localisation and mapping systems. SLAM algorithms like `slam_toolbox` and AMCL provide this transform, while GNSS-based localisation typically uses the `navsat_transform` and EKF nodes.

The challenge in switching localisation sources lies in changing the map reference, as each defines a different coordinate frame. Since ROS2 does not support multiple publishers for the same transform, one must be disabled before another is enabled. The Navigation Manager was therefore designed to handle lifecycle nodes and manage this switching process dynamically.

The localisation switching system went through several iterations, as early designs failed to meet requirements. Each version was tested using real-world ROS2 data from full campus runs, allowing realistic simulation without bus access. This proved one of the most difficult aspects of the project, requiring repeated refinement to ensure compatibility across localisation systems and the Navigation Manager. The following approaches were trialled, with implementation details and rejection reasons in Appendix C.0:

1. Transform multiplexer
2. Odometry multiplexer
3. Buffer transform frames

3 Final Design

3.1 Navigation Manager (Navmgr)

The full nUWAy4 system architecture and sensor pipeline utilised for the Navigation Manager system can be seen in Figure 3.1

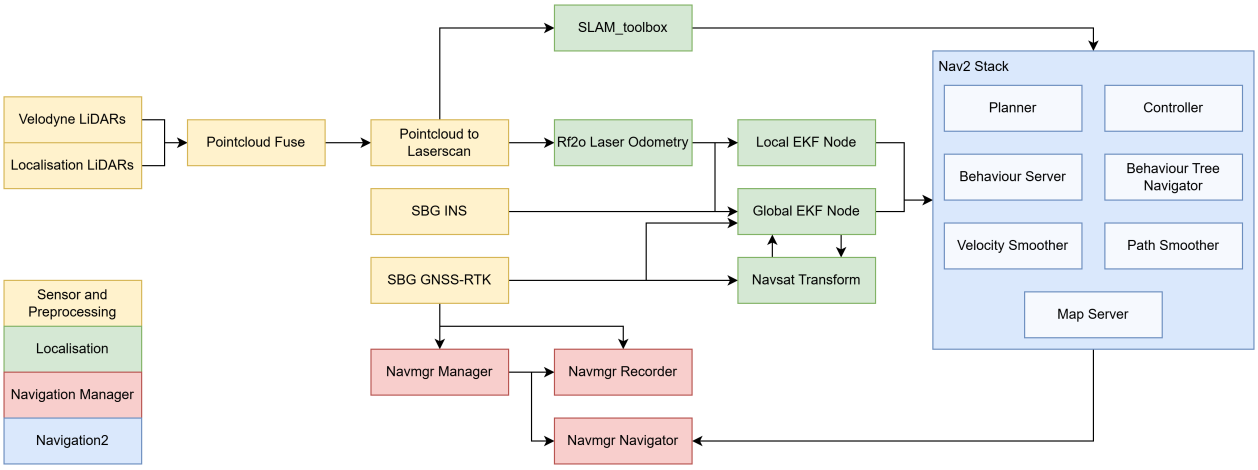


Figure 3.1: Final nUWAy4 shuttle bus system architecture

The Navigation Manager (Navmgr) works based off ROS2 lifecycle nodes. This form of node allow for nodes to be dynamically configured, activated, and deactivated over the course of its lifecycle. This was key to getting the Navigation Manager working, and any surrounding developed nodes for the system were developed using this framework. Additionally, the manager utilises ROS2 interfaces to subscribe and publish information to topics, and interact with services and clients that allow the system to manage other lifecycle nodes.

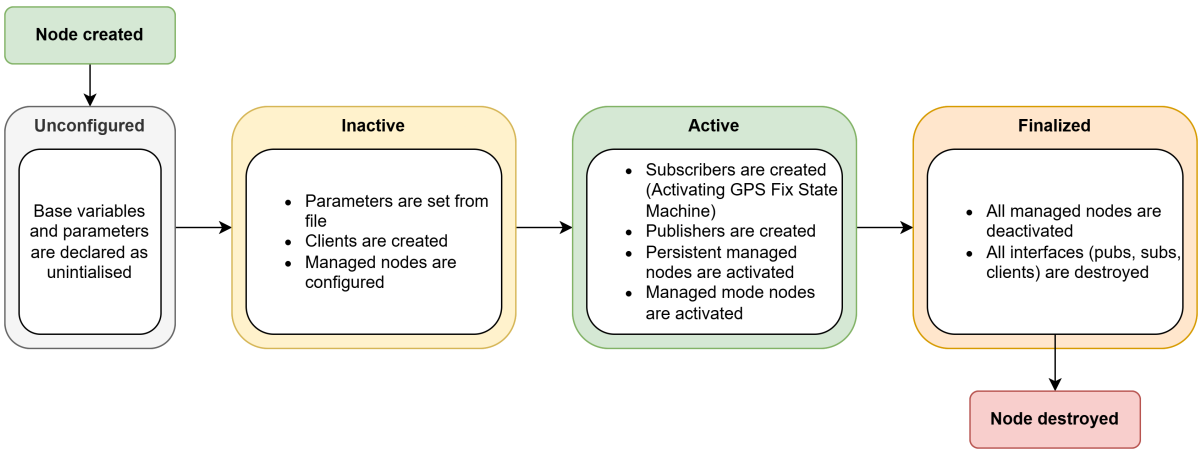


Figure 3.2: Navmgr ROS2 Lifecycle Setup

The full Navigation Manager system utilises several ROS2 nodes for localisation, mapping, path planning and operation. Robot Localisation was utilised for the global and local localisation of the robot to GNSS UTM coordinates, utilising `navsat_transform` for the `[utm]->[map]`

transformation and two `ekf_nodes` for the global and local localisation of the robot, forming the `[map]->[odom]` transformation and the `[odom]->[base_link]` transforms respectively. These nodes were also used to fuse sensor data together, in particular IMU information, GNSS odometry and LiDAR odometry data for the global localiser, and IMU information and LiDAR odometry data for the local localiser.

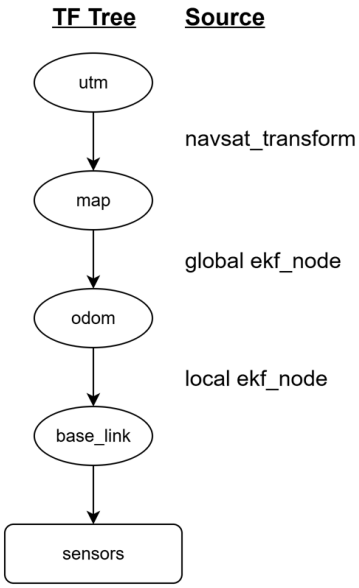


Figure 3.3: Navigation Manager transform tree

Three purpose-built ROS2 lifecycle nodes were developed over the course of this project, the Manager, the Navigator and the Recorder.

3.1.1 Manager

The manager is built off of a state machine based on the current GPS fix. Based on the covariance of the GPS fix, and prescribed timeout periods, the manager is able to switch between the different navigation modes - activating and deactivating mode modes, and calling services as required. The GNSS covariance threshold, the timeout periods, modes, and mode-specific nodes are able to be provided and tuned via ROS2 parameter config files. The two main functions, the state machine and mode changer, have been described in Figure 3.4 and Algorithms 1 and 2.

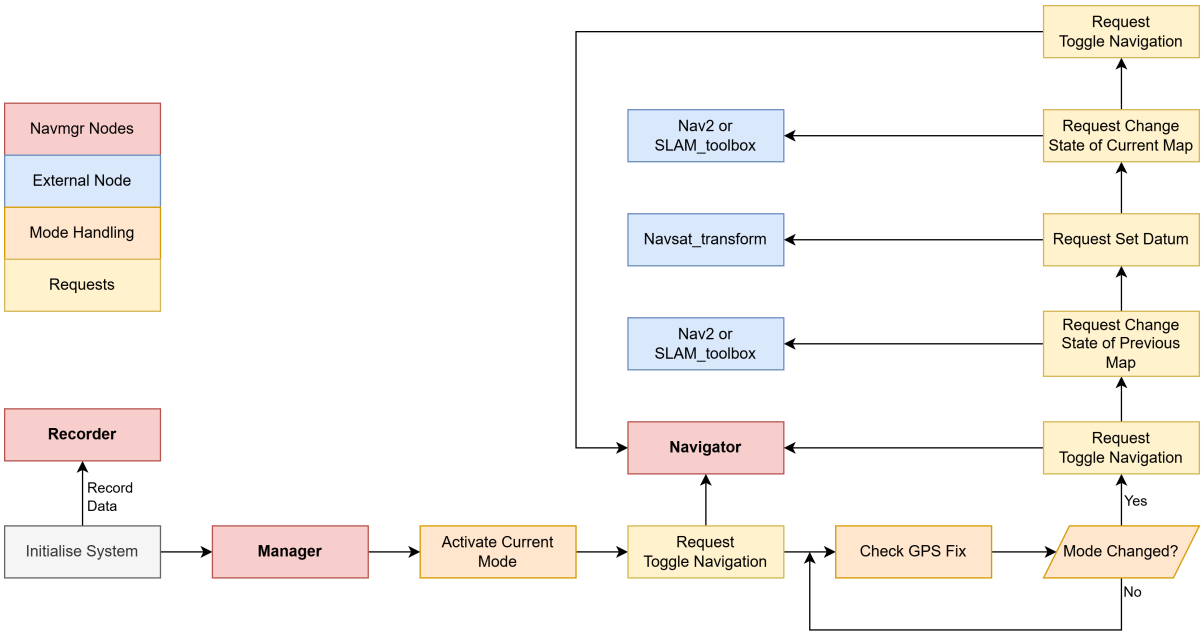


Figure 3.4: Navmgr System Flowchart

A notable aspect of the Manager is the localisation switching method – Since the SLAM map and GNSS coordinate frame had to be aligned when `slam_toolbox` was launched. This is as `slam_toolbox` starts its map at the GNSS datum. This was achieved by storing the most recent GNSS coordinate and transmitting it to the `navsat_transform` node via a service request to update the datum. This effectively re-centres the coordinate system around the vehicle's current position. The trade-off, however, is that each time this occurs, all waypoints must be re-projected into the (x,y) frame, and a new path plan must be generated.

Algorithm 1 Navigation Manager GPS Fix State Machine Logic

Inputs: GPS message with positional covariance
Parameters: $gpsCovarianceThreshold$, $gpsTimeout$, $slamWait$
State: $mode \in \{\text{GPS}, \text{SLAM}\}$, $hasFix$

procedure GETGPSFIX(gpsMsg)
    Read gpsMsg
    $hasFix \leftarrow (accuracy < gpsCovarianceThreshold)$

    if $hasFix$ **then**
        if $mode = \text{SLAM}$ and fix stable for $> gpsTimeout$ **then**
            Switch to GPS mode
        end if

    else
        if $mode = \text{GPS}$ and loss persists for $> gpsTimeout$ **then**
            Switch to SLAM mode
        end if
    end if
end procedure

Algorithm 2 Navigation Manager Mode Change Procedure

Input: newMode
State: currentMode, slamStart

procedure CHANGEMODE(newMode)
  if $newMode = currentMode$ **then**
    return
  end if

  Deactivate all nodes of currentMode
  $currentMode \leftarrow newMode$
  Activate all nodes of currentMode

  if $currentMode =$ GPS **then**
    $slamStart \leftarrow$ None
    Deactivate $slam\_toolbox$
    $navsat\_transform$'s $datum \leftarrow$ initial GPS position
    Activate map_server
  else if $currentMode =$ SLAM **then**
    Record current time as $slamStart$
    Deactivate Nav2's map_server
    $navsat\_transform$'s $datum \leftarrow$ last good GPS position
    Activate $slam\_toolbox$
  end if
end procedure

3.1.2 Navigator

The original `bus_follow_path` navigator was subsequently extended to meet the new system requirements. This involved replacing the `BasicNavigator` API with a direct interface to Nav2's `NavigateToPose` action service. The total path results are published at the start of navigation (Figure 3.5) and once navigation is toggled, the path to the next waypoint is published (Figure 3.6). The navigation pipeline can be seen in Figure 3.7

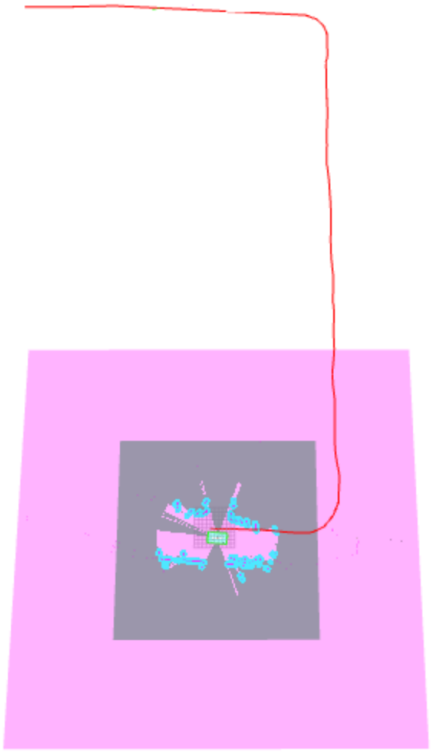


Figure 3.5: Planned path by Navigator (red)
vehicle footprint is green, obstacles identified are blue,
global costmap is pink, and local costmap is grey.

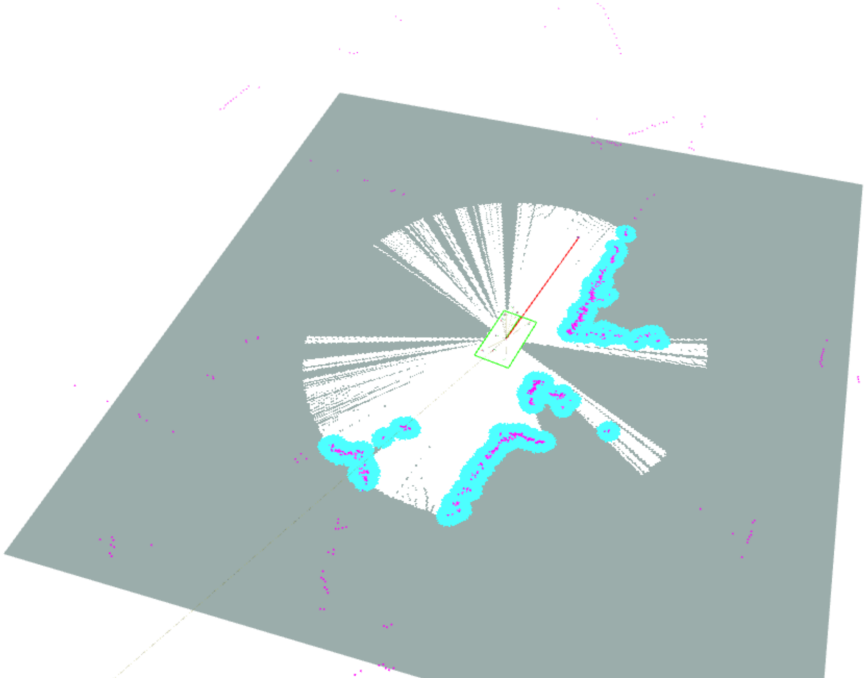


Figure 3.6: Local waypoint path planning results

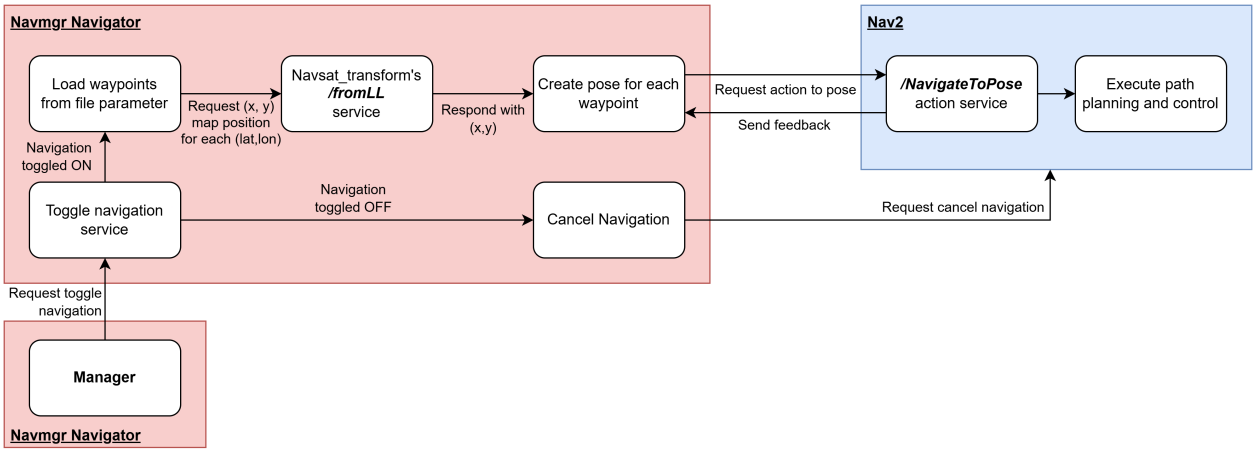


Figure 3.7: Navigation Manager Navigator

3.1.3 Recorder

The recorder subscribes to important topics and saves them into a file for easier processing and analysis, and abstraction over ROS2 bagfiles. The current latitude, longitude, their respective covariances, and heading values are recorded. Additionally, the node listens to the Navigation Manager's current mode and status, recording when these change, and what they changed to. Finally, the node listens to button inputs on the controller, detecting when the deadman switch is pressed or released, and when the toggle navigation button is pressed.

3.1.4 User Interface

An RVIZ2 plugin was developed to provide a user interface for the Navmgr, in order to fulfil the transparency and usability criteria set out in section 2.1.3: Evaluation Criteria. The user interface consists of three main panels: the control panel, which allows for the configuration and activation of the Navmgr lifecycle node; the status panel, which provides the current lifecycle state of each managed node; and the monitor panel, which provides the current positioning and accuracy data of the shuttle bus, as well as the current mode and status of the Navmgr.

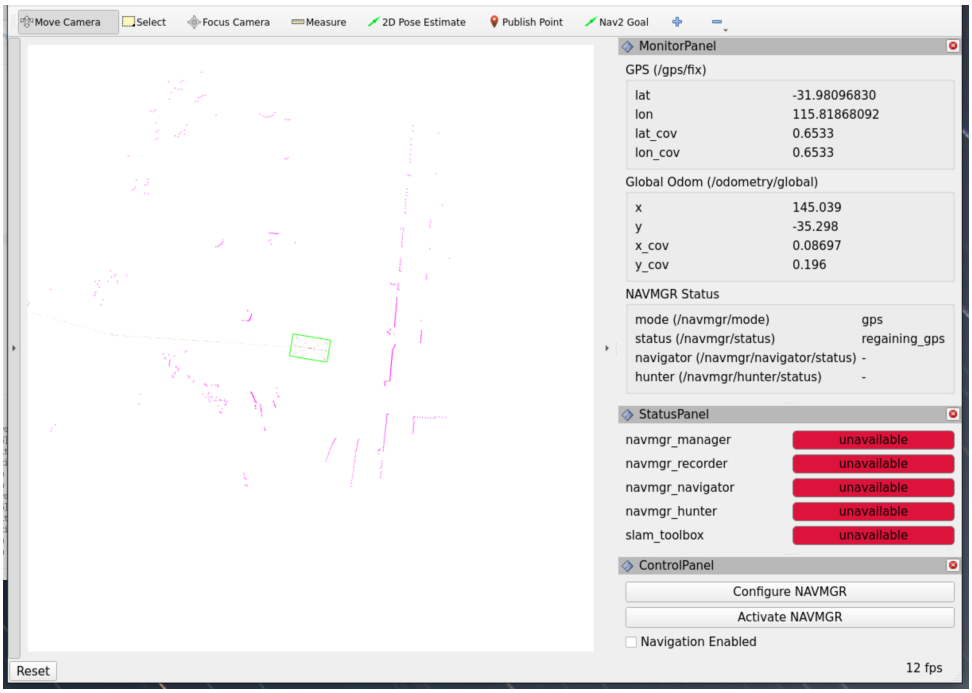
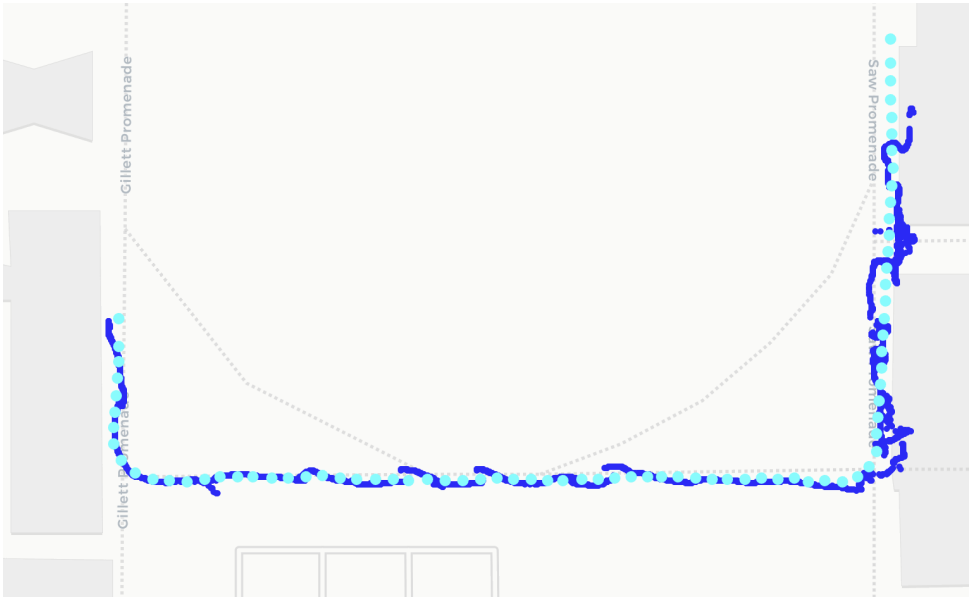


Figure 3.8: Navigation Manager User Interface

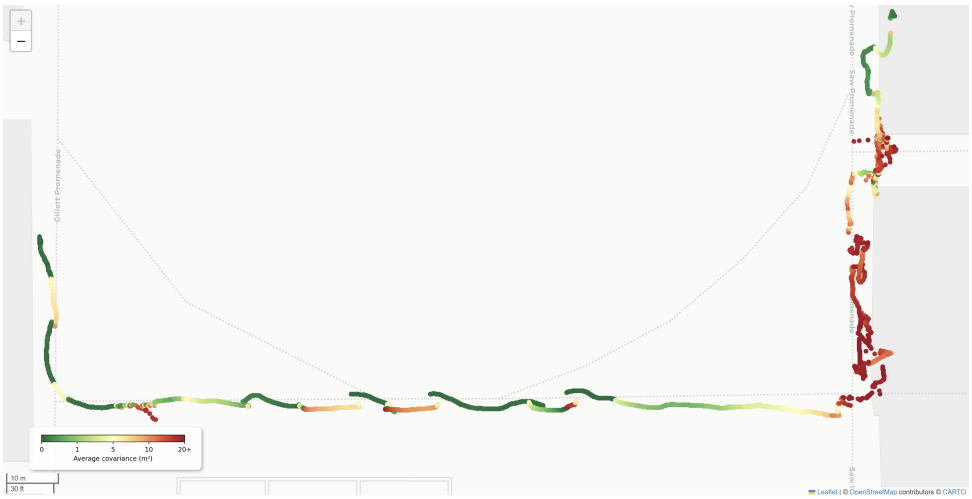
4 Results and Discussion

4.1 RTK Driving

The RTK driving results can be see in Figure 4.1a, where the GNSS coordinate waypoints given to the navigator are coloured in cyan, and the actual path of the shuttle bus is coloured in blue. The GNSS covariance values have been provided in Table 4.1, and have been pictured in Figure 4.1b.



(a) RTK Autonomous Navigation Results versus Waypoints



(b) Positional covariance during autonomous RTK navigation

Figure 4.1: Novatel and SBG GNSS-RTK module comparison

The covariance data revealed significant variation throughout the navigation path. As anticipated, the worst covariance values (exceeding 900,000-metres) occurred at the turn between James Oval car park and Guild Village, where dense foliage and building obstruction degraded GNSS signal quality.

However, unexpected covariance fluctuations were also observed along the supposedly open James Oval car park section.

These accuracy variations directly impacted navigation performance. During the path along the car park, positional uncertainty caused the bus to deviate toward the flower bed rather than maintaining the intended straight trajectory. Operator intervention was required to correct the course, as evidenced by the erratic path shown in the data. Similarly, after the challenging turn between James Oval and Guild Village, the system failed to recover stable localisation, causing the bus to drift toward nearby buildings and again requiring manual override.

These results demonstrate that even in predominantly open-sky environments, localised GNSS degradation can severely compromise autonomous navigation performance, validating the need for the proposed multi-modal localisation approach. However, this also indicates that GNSS-RTK signals cannot be fully predicted, as there is still a large amount of variability, even along clear-sky paths - which could be due to changes in network availability, weather conditions, or sensor noise.

Table 4.1: RTK Waypoint Navigation Covariance

Combined Covariance[a] [m]	**GNSS**
Average	6,559
Median	1.208
Standard Deviation	59,450
Mode	0.00951
Minimum	0.00828
Maximum	949,400

[a] Combined covariance was calculated as the average between the latitudinal and longitudinal covariances

4.2 SLAM Driving

The implementation of 2-D LiDAR SLAM navigation followed a two-stage process. First, the existing SLAM system was adapted to interface with the new nUWAy bus sensor suite. Second, it was extended for GNSS waypoint compatibility.

The first stage involved configuring the transform tree, EKF parameters, and tuning `slam_toolbox` for stable localisation. After several weeks of testing, a functional system capable of receiving Nav2 goals through RVIZ was achieved. The results, shown in Figure 4.2, demonstrated accurate real-time mapping and navigation in controlled conditions.

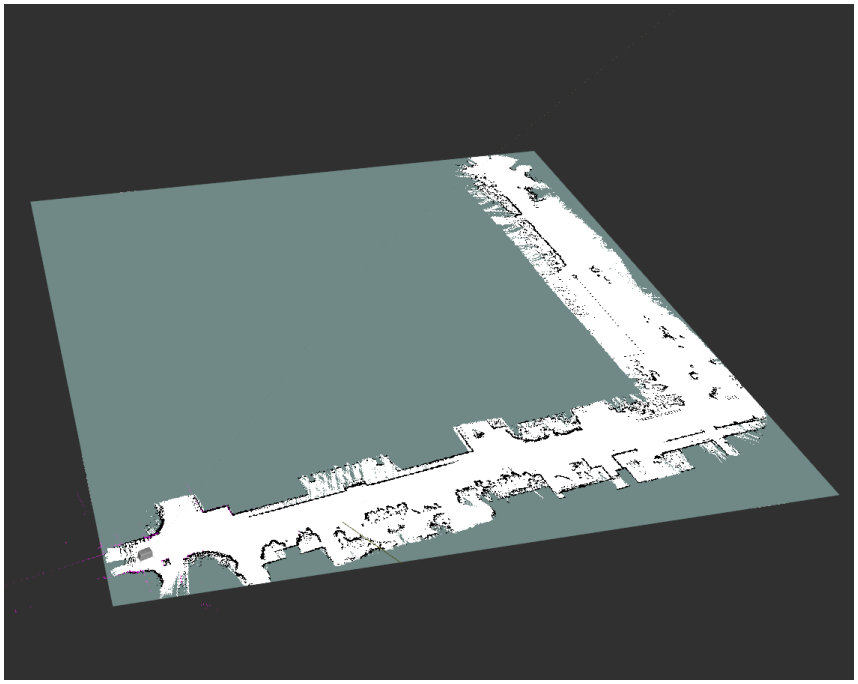


Figure 4.2: Preliminary SLAM results

The second stage, extending SLAM for GNSS waypoint navigation, proved significantly more complex than expected. The localisation and transform management between SLAM and GNSS introduced several frame alignment issues, requiring major redevelopment of the node interfaces. Additional complexity arose from the behaviour of `slam_toolbox`, which always initializes its map at the map datum. This offset map can be seen in Figure 4.3.

Additionally, testing showed that concurrent SLAM and Nav2 processes heavily exceeded the onboard PC's processing capacity. Even with partial offloading to an external laptop, CPU utilisation reached 100% on both systems, preventing real-time operation. To minimise computational load, SLAM was designed to activate only when necessary. As a result, each activation caused the map to reset, requiring the system to re-establish its local coordinate frame and recompute all waypoints relative to SLAM's new reference frame. However, due to these changes, the transition times increased significantly, as the manager has to wait for the respective maps to deactivate or reactivate, and the datums to be set.

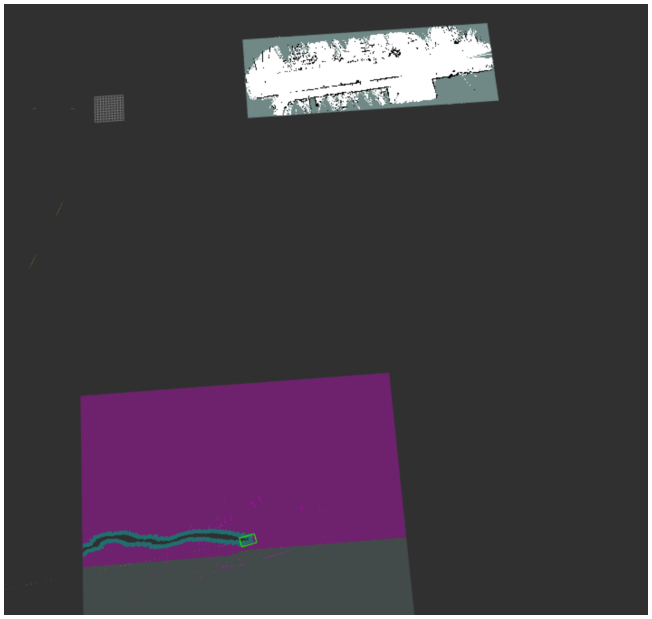


Figure 4.3: SLAM map offset to current vehicle location
SLAM map is at the top, vehicle map is below

4.3 Navigation Manager

The Navigation Manager could not be tested in live driving scenarios due to computational and system complexity limitations. Exploratory trials from over the course of this project, using recorded ROS2 datasets, showed the manager correctly handled node state transitions and GNSS/SLAM switching, but full navigation behaviour was limited by processing constraints.

Requirement R5 (real-time operation) could not be achieved, as the onboard computer was quickly overloaded. This also prevented in-person validation of requirement R4 (dynamic localisation switching). Testing also revealed that localisation transitions caused navigation interruptions: Nav2's map server had to shut down before SLAM could initialise, resulting in several seconds without valid pose data. This confirmed that the system did not yet achieve the seamless switching required by the design objectives. Overall, requirements R1–R3 were achieved, but the system fell short on R4 and R5 due to computational bottlenecks. Requirement R2 - operation without a predefined map - may need re-evaluation, as using even a poor, rough static map could significantly improve stability and efficiency.

4.4 Limitations

4.4.1 nUWAy4 Main PC

Initial tests indicated the main PC could handle sensor input, Nav2, and waypoint navigation. However, once SLAM and EKF processing were added, the hardware quickly became a limiting factor. Upgrading was not feasible as it would require additional setup and validation time, which would have impacted other student's work on their projects as well. Instead, a compromise was devised such that the main PC handled sensor input, while localisation and navigation ran on an external laptop. Although functional, the system was bottlenecked by Ethernet bandwidth, and both computers remained near full CPU load, confirming that the navigation stack exceeded available processing capacity.

4.4.2 Internet Connection and Network Bandwidth Constraints

GNSS-RTK localisation required a stable internet connection for correction data. In several campus areas, including near the Robotics Lab, connection dropouts caused covariance spikes despite good GNSS visibility. During testing, it was also noted that heavy sensor data traffic also saturated the shuttle's internal network, further disrupting RTK communication. These issues showed that the system could not maintain reliable operation without consistent network quality – which is a known limitation of RTK-based navigation systems, however if this system were adapted to two different localisation systems, it could also pose an issue if sensor input load was high.

4.4.3 Localisation Switching

Since ROS2 does not support multiple publishers for a single transform, the Navigation Manager had to explicitly manage transitions between localisation sources. In testing, switching between GNSS and SLAM caused temporary loss of the `[map]->[odom]` transform, leading to navigation failures that required a full stack restart. The implemented process – deactivating Nav2's `map_server`, updating the GNSS datum, and restarting `slam_toolbox` – was somewhat reliable in preliminary testing, but extremely slow. Each transition created several seconds of downtime where the vehicle could not localise, highlighting the need for a more efficient switching mechanism.

4.4.4 No Persistent Map

Because `slam_toolbox` was repeatedly activated and deactivated, its generated maps were discarded after each session. This meant the system had to rebuild the environment map every time it revisited the same area, wasting computation and time. For a fixed shuttle route, this approach is inefficient. Persistent mapping or partial map caching would allow reuse of previously mapped regions and reduce redundant processing.

5 Future Work

5.1 nUWAy4 Main PC Upgrades

The main limitation of this project was the outdated hardware. The onboard PC, equipped with a decade-old CPU and 16 GB of RAM, struggled with concurrent tasks such as SLAM mapping, Nav2 path planning, and RVIZ visualisation. Even with partial offloading to a laptop, real-time performance was not achievable. Future work should explore migrating SLAM and Nav2 to the Orin PC used for AI tasks, which offers a significantly faster processor and dedicated GPU, improving real-time reliability and sensor synchronisation.

5.2 Navigation System Optimisation

As discussed in section 4.4: Limitations, computational overhead remains the primary barrier to overall system performance and thus the testing and validation of the system. Tuning `ekf_node` parameters, `slam_toolbox` settings, and Nav2 costmap configurations could yield substantial efficiency gains. The current configuration uses a 7000-by-7000-pixel map and large (50-metre and 100-metre) costmaps, which are computationally expensive. Lowering resolution, dynamically scaling maps, or incorporating persistent mapping (Section 4.4.4) could enable smoother, lower-latency operation - which could make this system feasible for real-world operations. However, it is likely that a different combination of GNSS localisation, sensor fusion and/or SLAM packages may have to be further explored to find the most performant implementation.

5.3 Field Testing and Validation

As the system was not able to be fully tested over the course of this project - comprehensive validation trials would still be needed to evaluate performance across varied environments such as open fields, semi-urban roads, and GNSS-degraded areas. Future evaluations should include quantitative metrics – position drift, recovery time, and uptime – rather than qualitative observations alone. Incorporating adaptive thresholds for covariance-based switching or a memory-based localisation mechanism, like from [7], could improve transition reliability and responsiveness.

5.4 Low-Level Sensor Improvements

Instead of switching between full localisation frameworks, future research could focus on integrating lightweight, low-level sensor systems to improve GNSS navigation in poor-signal regions - rather than trying to switch between two fully-fledged localisation systems. Early in this project, a vision-based obstacle avoidance prototype was developed - with the end goal of using machine learning to segment road images into "drivable" and "non-drivable" regions. From this segmentation, a line-of-best-fit was calculated to determine the optimal steering trajectory for obstacle avoidance. The segmentation pipeline was completed during this project, while a collaborating team member labelled, trained and validated the model. The resulting system demonstrated the potential for sensor-level

perception improvements that could support or even replace complex localisation switching in simpler environments.

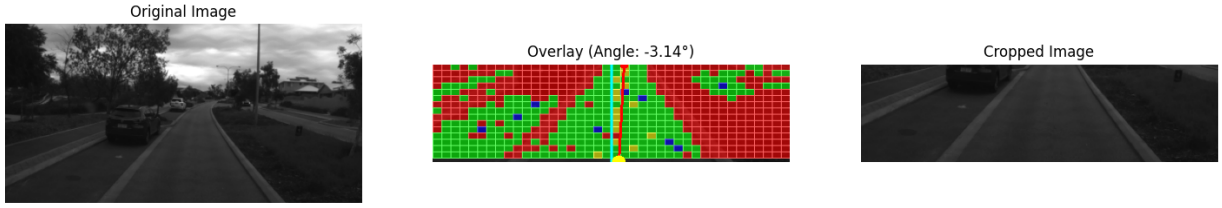


Figure 5.1: Obstacle avoidance segmentation data

6 Conclusion

This thesis presented the design, implementation, and evaluation of RTK-based autonomous navigation on the UWA Crawley campus, as well as of a multi-sensor localisation system for operation in GNSS-degraded environments. The system was developed and deployed on the UWA REV's nUWAy autonomous shuttle buses, aiming to address the key challenge of maintaining accurate, and continuous, localisation in areas where GNSS signals become unreliable or unavailable.

The primary contribution of this work is the development of an RTK basestation on campus, and the evaluation of the feasibility of RTK-based waypoint navigation for on-campus use. Additionally, the development of a multi-modal localisation manager that can switch from GNSS-RTK positioning and LiDAR-based SLAM techniques was developed, however it could not be fully tested. This thesis aimed to address that by implementing RTK error correction, sensor fusion, and a dynamic switching mechanism between different localisation sources, a primarily GNSS-based navigation system can maintain navigation continuity and be made more robust across diverse campus environments – such as areas with poor GNSS coverage such as building overhangs, tree-lined paths, and urban canyon-like settings.

The hardware implementation successfully demonstrated the feasibility of retrofitting existing autonomous platforms with improved RTK-GNSS capabilities. The custom RTK basestation, integrating a SparkFun u-blox RTK-enabled GNSS receiver, with RTKExplorer's fork of RTKLIB, allowed the nUWAy shuttle buses to achieve centimetre-level accuracy when operating in RTK-mode. The limitations of RTK-based navigation was confirmed through on-campus field testing over the course of a diverse test path. Likewise, several limitations were identified during development and testing. The reliance on stable internet connectivity for RTK corrections remains a constraint in certain campus locations. Additionally, the system suffered greatly due to poor computational power on the nUWAy shuttle buses - heavily bottlenecking sensor fusion, SLAM navigation and the collection of results. Similarly, the switching logic between localisation modes, while functional, could benefit from more sophisticated sensor fusion techniques, different localisation methods, or a persistent map to provide smoother transitions and improved accuracy during mode changes.

Future work should focus on implementing different sensor fusion algorithms, such as particle filters or different EKF filters, the exploration of the usage of different localisation methods other than SLAM, or the implementation of low-level obstacle avoidance techniques in order to improve GNSS-RTK waypoint navigation. While the final navigation manager system could not be fully tested on the nUWAy shuttle buses, this project hopes to demonstrate feasibility (or lack thereof) of combining GNSS-RTK with SLAM techniques for autonomous vehicle navigation in mixed environments. As autonomous vehicles become more commonplace in more diverse and variable environments, such integrated localisation approaches could become essential for ensuring safe and reliable navigation.

References

[1] D. Bastos, P. P. Monteiro, A. S. R. Oliveira, and M. V. Drummond, "An Overview of LiDAR Requirements and Techniques for Autonomous Driving," in *2021 Telecoms Conference (ConfTELE)*, Feb. 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9435580

[2] Y. Alkendi, L. Seneviratne, and Y. Zweiri, "State of the Art in Vision-Based Localization Techniques for Autonomous Navigation Systems," *IEEE Access*, vol. 9, pp. 76 847–76 874, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9438708

[3] Y. Zhang, P. Shi, and J. Li, "LiDAR-Based Place Recognition For Autonomous Driving: A Survey," *ACM Computing Surveys*, vol. 57, no. 4, pp. 1–36, Apr. 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3707446

[4] N. Joubert, T. G. R. Reid, and F. Noble, "Developments in Modern GNSS and Its Impact on Autonomous Vehicle Architectures," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020, pp. 2029–2036. [Online]. Available: https://ieeexplore.ieee.org/document/9304840/

[5] Y. He, J. Li, and J. Liu, "Research on GNSS INS & GNSS/INS Integrated Navigation Method for Autonomous Vehicles: A Survey," *IEEE Access*, vol. 11, pp. 79 033–79 055, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10195949/

[6] T. Hasegawa, H. Miyoshi, and S. Yuta, "Experimental Study of Seamless Switch Between GNSS- and LiDAR-Based Self-Localization," *Journal of Robotics and Mechatronics*, vol. 35, no. 6, pp. 1514–1523, Dec. 2023. [Online]. Available: https://www.fujipress.jp/jrm/rb/robot003500061514

[7] S. Bhat and A. Kavasseri, "Multi-Source Data Integration for Navigation in GPS-Denied Autonomous Driving Environments," Jul. 2024. [Online]. Available: https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120317.html

[8] R. Yuan, B. Ji, Y. Gao, and H. Tao, "A review of LiDAR simultaneous localization and mapping techniques for multi-robot," *Robotica*, pp. 1–41, Sep. 2025. [Online]. Available: https://www.cambridge.org/core/journals/robotica/article/review-of-lidar-simultaneous-localization-and-mapping-techniques-for-multirobot/4F66EFD9B1E153198D60057E12CDCA12

[9] X. Yue, Y. Zhang, and M. He, "LiDAR-based SLAM for robotic mapping: State of the art and new frontiers," Nov. 2023. [Online]. Available: http://arxiv.org/abs/2311.00276

[10] M. Horst and R. Möller, "Visual Place Recognition for Autonomous Mobile Robots," *Robotics*, vol. 6, no. 2, p. 9, Jun. 2017. [Online]. Available: https://www.mdpi.com/2218-6581/6/2/9

[11] R. Schmalzried, "An Examination of Moving Baseline RTK, RTK-Based Heading Technology and How RTK-Based Solutions Support Autonomous Vehicle Sensor Edge Cases," Swift Navigation, Whitepaper, Sep. 2017. [Online]. Available: https://www.swiftnav.com/sites/default/files/whitepapers/moving_baseline_white_paper_092017.pdf

[12] T. Takasu and A. Yasuda, "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB," Laboratory of Satellite Navigation, Tokyo University of Marine Science and Technology, Tech. Rep., Jan. 2009. [Online]. Available: https://gpspp.sakura.ne.jp/paper2005/isgps_2009_rtklib_revA.pdf

[13] C. Park, S. Goh, H. Jang, and S. Park, "Development of Accurate Position Estimation Device Using Rtk Module Based on GNSS," in *2023 8th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*. Beijing, China: IEEE, Nov. 2023, pp. 315–319. [Online]. Available: https://ieeexplore.ieee.org/document/10390804/

[14] S. Mahato, A. Ghosh, P. Mishra, S. Dan, and A. Bose, "Dynamic RTK using Compact GNSS Modules within an Indian Urban Environment," in *2023 3rd International Conference on Range Technology (ICORT)*, Feb. 2023, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10249099/

[15] A. El-Mowafy, "Precise Real-Time Positioning Using Network RTK," in *Global Navigation Satellite Systems: Signal, Theory and Applications*, S. Jin, Ed. InTech, Feb. 2012. [Online]. Available: http://www.intechopen.com/books/global-navigation-satellite-systems-signal-theory-and-applications/precise-real-time-positioning-using-network-rtk

[16] H. Koivula, J. Kuokkanen, S. Marila, S. Lahtinen, and T. Mattila, "Assessment of sparse GNSS network for network RTK," *Journal of Geodetic Science*, vol. 8, no. 1, pp. 136–144, Dec. 2018. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/jogs-2018-0014/html

[17] C. Rizos and S. Han, "Reference station network based RTK systems-concepts and progress," *Wuhan University Journal of Natural Sciences*, vol. 8, no. 2, pp. 566–574, Jun. 2003. [Online]. Available: http://link.springer.com/10.1007/BF02899820

[18] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," 2006.

[19] Y. Yin, J. Zhang, M. Guo, X. Ning, Y. Wang, and J. Lu, "Sensor Fusion of GNSS and IMU Data for Robust Localization via Smoothed Error State Kalman Filter," *Sensors (Basel, Switzerland)*, vol. 23, no. 7, p. 3676, Apr. 2023. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10099052/

[20] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1638022/

[21] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, Jul. 2014. [Online]. Available: http://www.roboticsproceedings.org/rss10/p07.pdf

[22] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," Jul. 2020. [Online]. Available: http://arxiv.org/abs/2007.00258

[23] S. Yi, S. Worrall, and E. Nebot, "Integrating Vision, Lidar and GPS Localization in a Behavior Tree Framework for Urban Autonomous Driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, Sep. 2021, pp. 3774–3780. [Online]. Available: https://ieeexplore.ieee.org/document/9564875/

[24] T. Everett, "Releases · rtklibexplorer/RTKLIB." [Online]. Available: https://rtkexplorer.com/

[25] G. Biggs and T. Foote, "Managed nodes," Feb. 2021. [Online]. Available: https://design.ros2.org/articles/node_lifecycle.html

[26] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, May 2021. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.02783

[27] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in *Intelligent Autonomous Systems 13*, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds. Cham: Springer International Publishing, 2016, vol. 302, pp. 335–348. [Online]. Available: https://link.springer.com/10.1007/978-3-319-08338-4_25

[28] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The Marathon 2: A Navigation System," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 2718–2725. [Online]. Available: http://arxiv.org/abs/2003.00368

[29] M. Jaimez, J. G. Monroy, and J. Gonzalez-Jimenez, "Planar odometry from a radial laser scanner. A range flow-based approach," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, May 2016, pp. 4479–4485. [Online]. Available: https://ieeexplore.ieee.org/document/7487647/

[30] A. Kaczmarek, W. Rohm, L. Klingbeil, and J. Tchórzewski, "Experimental 2D extended Kalman filter sensor fusion for low-cost GNSS/IMU/Odometers precise positioning system," *Measurement*, vol. 193, p. 110963, Apr. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026322412200238X

[31] M. I. Ribeiro, "Kalman and Extended Kalman Filters: Concept, Derivation and Properties," Feb. 2004.

[32] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking," *Journal of Sensors*, vol. 2021, no. 1, p. 2054828, Jan. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2021/2054828

[33] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual SLAM," *Expert Systems with Applications*, vol. 205, p. 117734, Nov. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417422010156

[34] Z. Hong, Y. Petillot, A. Wallace, and S. Wang, "Radar SLAM: A Robust SLAM System for All Weather Conditions," 2021. [Online]. Available: https://arxiv.org/abs/2104.05347

[35] J. Zhu, H. Li, and T. Zhang, "Camera, LiDAR, and IMU Based Multi-Sensor Fusion SLAM: A Survey," *Tsinghua Science and Technology*, vol. 29, no. 2, pp. 415–429, Apr. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10258154/

[36] P. Chen, X. Zhao, L. Zeng, L. Liu, S. Liu, L. Sun, Z. Li, H. Chen, G. Liu, Z. Qiao, Y. Qu, D. Xu, L. Li, and L. Li, "A Review of Research on SLAM Technology Based on the Fusion of LiDAR and Vision," *Sensors (Basel, Switzerland)*, vol. 25, no. 5, p. 1447, Feb. 2025. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11902412/

[37] Y. Kato and K. Morioka, "Autonomous Robot Navigation System Without Grid Maps Based on Double Deep Q-Network and RTK-GNSS Localization in Outdoor Environments," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, Jan. 2019, pp. 346–351. [Online]. Available: https://ieeexplore.ieee.org/document/8700426

[38] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo localization for mobile robots,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2. Detroit, MI, USA: IEEE, 1999, pp. 1322–1328. [Online]. Available: http://ieeexplore.ieee.org/document/772544/

Appendices

A.0 Literature Review

A.1.0 Localisation

Localisation is a fundamental capability for autonomous vehicles, requiring precise pose estimation within the operational environment. This necessitates integration of heterogeneous sensor modalities, each with distinct performance characteristics, costs, and operational constraints. Contemporary systems typically employ multi-modal sensor suites including cameras, inertial measurement units (IMU), Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR), and Global Navigation Satellite System (GNSS) receivers. Accurate localisation serves as a prerequisite for higher-level autonomous functions including trajectory planning, obstacle avoidance, and vehicle control.

A.1.1.1 Pose and Odometry

The pose and odometry of a robot are important descriptors of a robot's current state. The pose of the robot refers to the current position and orientation of the robot, as pictured in Figure X below. For example, for an autonomous vehicle that utilises GNSS localisation, the pose can be described as its position along latitude, longitude and altitude, and its orientation can be described as its current heading or roll, pitch and yaw.

Likewise, a robot's odometry describes its change of pose over time. Odometry can be measured in a number of different ways, using different sensors. Typically, this information is obtained using wheel sensors, or using IMU modules, which measure the linear and angular velocity and/or acceleration of a robot. Odometry sources tend to be subject to cumulative errors due to wheel slip, sensor noise or drift, and hence tend to be fused with other localisation sources in order to reduce probabilistic error.

A.1.2.2 GNSS/RTK-Based

GNSS positioning utilises multiple satellites across different operators (the United State's GPS, Russia's GLONASS, Europe's Galileo, China's Beidou, and/or possibly more) in order to triangulate the current position of an object. However, GNSS positioning is affected by numerous sources of error and noise, such as satellite or receiver clock bias and clock error, satellite ephemerides and atmospheric interference [11].

GNSS-based localisation methods the most common choice, due to the global coverage available and as several low-cost, accurate mass-market options are available to the average consumer [3], [4]. This makes utilising GNSS as the main form of localisation is a cost-effective and efficient way to implement localisation and mapping on AVs. However, as previously discussed, GNSS-based localisation systems can be unreliable in common GNSS-degraded environments, making them not applicable to all scenarios.

RTK, or RTK-GNSS, is a precise differential GNSS positioning system that uses carrier-phase measurements to deliver more accurate positional data to a rover (mobile receiver) [12]. This is typically done using a basestation with a known position, which sends real-time error corrections to an AV about its position [11]. This allows for improved positional accuracy in open-sky environments over normal GNSS positioning, with up to centimetre-level accuracy [13]. RTK requires a stable channel for communication, commonly done using an NTRIP server – which uses the HTTP to send corrections over the internet, in the form of messages using the RTCM protocol [14]. RTK-enabled GNSS receivers have become increasingly popular inclusions into AVs, as they provide a relatively simple way to improve the quality and robustness of AV positioning.

Despite its advantages, RTK-GNSS systems are not without limitations. The accuracy of RTK positioning is highly dependent on the quality of the communication link between the rover and the basestation, as well as the distance between them. As the distance increases, the effectiveness of the error corrections diminishes, with RTK correction data being most effective from within 10–20-km radius from the reference basestation [15]. Furthermore, RTK systems are susceptible to signal obstructions caused by buildings, trees, or other environmental factors, which can lead to degraded performance or complete loss of positioning. To mitigate these challenges, multi-constellation and multi-frequency GNSS receivers are often employed, as they can improve signal availability and reduce the impact of interference. Additionally, NRTK technology makes RTK more feasible for larger, more expansive environments [16], [17] - However this is out of the scope of this project.

A.1.3.3 Extended Kalman Filter (EKF)

The EKF is a probabilistic state estimation algorithm that extends a normal Kalman filter to handle non-linear systems by linearising around the current state estimate [18]. Commonly in autonomous navigation and localisation, EKF serves as a sensor fusion framework that combines multiple sources of positioning and odometry data to produce a single, more accurate and robust pose estimate. The filter operates by maintaining a state vector that typically includes the vehicle's position, orientation, and velocities, along with an associated covariance matrix that represents the uncertainty in these estimates.

By fusing complementary sensor inputs such as GNSS coordinates, IMU measurements, wheel odometry, and visual or LiDAR odometry, the EKF can leverage the strengths of each sensor while mitigating their individual weaknesses. For instance, while GNSS provides absolute positioning but may suffer from signal degradation, IMU data offers high-frequency motion updates but is subject to drift over time [19, 30]. The EKF's prediction and update cycle continuously estimates the vehicle's state by predicting the next state based on motion models and then correcting this prediction using incoming sensor measurements, weighted according to their respective covariances and reliability [31].

A.1.4.4 Simultaneous Localisation and Mapping (SLAM)

SLAM is a technique used autonomous systems to build a map of an unknown environment while, at the same time, keeping track of the robot or AV's position within it [20]. SLAM is a pivotal technology for autonomous navigation. It enables AVs and other robots to operate in environments where pre-existing maps are not available, or unreliable. The process involves integrating data from various sensors to create a consistent representation of the environment and the agent's trajectory [20].

SLAM can be implemented using a variety of sensors, each with its own strengths and limitations. Commonly used sensors include cameras, LiDARs, and radar systems. Visual SLAM, for instance,

relies on cameras to capture images of the environment and uses computer vision techniques to extract features and track their motion over time. This approach is relatively low cost and yields rich visual and semantic information, but it is sensitive to illumination changes and tends to degrade in low-light or feature-poor environments [32, 33]. Radar-based SLAM, in contrast, is often more robust under adverse weather conditions (for example, rain, fog, snow) because radar signals are less affected by atmospheric conditions [34]. However, radar typically produces lower spatial resolution data and presents challenges in feature extraction and association compared to cameras or LiDARs.

A.1.5.5 LiDAR-Based

Often when talking about LiDAR-based localisation methods, some form of SLAM is meant. LiDAR-based SLAM is a common method for the localisation of AVs in GNSS-degraded environments, and has shown to improve the functionality of a vehicle when driving in a GNSS map [7]. LiDAR SLAM is a popular choice for autonomous vehicles due to its ability to generate highly accurate 2D or 3D spatial data [21]. LiDAR sensors work by emitting laser beams and measuring the time it takes for the beams to return after hitting an object, which allows for precise distance measurements. This data is then used to create detailed point clouds of the environment. LiDAR SLAM algorithms, such as LOAM (Lidar Odometry and Mapping) and LIO-SAM (Lidar-Inertial Odometry via Smoothing and Mapping), are designed to process this data efficiently, enabling real-time mapping and localization [21, 22]. LiDAR SLAM is particularly effective in low-light conditions and is less affected by changes in lighting compared to Visual SLAM. However, it can be computationally intensive and may face challenges in environments with sparse or repetitive features. Additionally, the integration and fusion of LiDAR data with other sensors, such as cameras and IMUs, has been shown to enhance the robustness and accuracy of SLAM systems in challenging environments [**?**, 35, 36].

Other proposed LiDAR based methods include sensor fusion with an INS system, much like with the GNSS, as well as fusion with a GNSS system [37] – which like GNSS-INS fused systems, tend to suffer from error accumulation. Likewise, LiDAR scan matching systems have been proposed in regards to improving a GNSS-based navigation system [7].

A.1.6.6 Visual-Based

SLAM can also be applied using vision-based systems, and models exist such as ORB-SLAM3 and SVO2, however LiDAR-based SLAM was found to perform much better in most situations [18]. Alternatively, vision systems can be fused with INS systems and GNSS-RTK positions in order to form a better view of ground truth [19], [20], however once again these methods tend to greatly accumulate errors.

Visual-based systems tend to be far less computationally heavy and have a lower cost to implement that LiDAR based systems. However, these systems are inherently more fragile as they depend on the right lighting conditions and scenes with texture. In poor lighting and low-texture environments, or under rapid motion, these systems tend to fail [21].

A.2.0 Adaptive Monte Carlo Localisation (AMCL)

The AMCL algorithm is a probabilistic localisation technique that employs a particle filter to estimate a robot's 2-D pose within a known environment [38]. This method utilises sensor data to compare observed environmental features against a pre-existing map, weighting a set of potential poses (particles)

according to their consistency with sensor measurements.

A principal limitation of AMCL is its dependence on a pre-constructed and well-detailed map, as localisation accuracy directly correlates with the quality of this map [38]. This requirement, however, makes it unideal for use in this project - which aims to use primarily GNSS-based mapping and navigation, with supplementary fallback methods in areas where GNSS signal is degraded. This raises an important consideration regarding the relative benefits of a purely GNSS-based localisation approach compared with an AMCL or AMCL-GNSS hybrid system, particularly in scenarios where high-quality maps are available and AMCL can be effectively tuned.

A.3.0 Decision-Making

Decision-making is a critical component of autonomous driving as well, responsible for interpreting perception data and determining the next appropriate course of action while operating. Using decision-making methods is integral to many path-planning and behaviour modelling of autonomous vehicles, and many different methods have been trialled, including: Probabilistic methods, statistical learning methods, deep learning methods and reinforcement learning methods [11].

A.3.1.1 Previous Localisation Switching Implementations

[7] describes a decision-making algorithm for switching between GNSS-based positioning and cellular (LTE/5G) based positioning for autonomous driving systems, based on simulated sensor data. The paper noted good results, and after 5,000 simulations, the system selected the correct mode 98.2% of the time and was able to maintain a continuous navigation time of 99.7%, compared to 78.3% of the time with GPS alone.

Similarly, [6] discusses a method to switch between GNSS-based and 3-D LiDAR-based self-localisation using NDT scan matching, with an autonomous vehicle for the Tsukuba Challenge 2022. The system self-localises using concurrently using RTK-GNSS, 3D LiDAR NDT scan matching and gyro-odometry – then a switcher node is used to determine which is the best localisation method and sends it to a localiser. While this method showed improvement over GNSS-RTK localisation alone, the system experienced poor performance due to lack of computing power.

[23] describes a navigation decision-making system based around behaviour trees, which is able to switch between Visual SLAM, LiDAR landmark recognition and GNSS localisation for autonomous navigation around the University of Sydney campus – particularly through tunnels. From provided mapped data, the system resulted in a path very similar to a global reference, and showed improvement over dead-reckoning of GNSS positioning alone.

B.0 GNSS Module Specifications

Table 1: GNSS Module Specification Comparisons

Specification	**Novatel FlexPak 6**	**SBG Ellipse-D**
Single Point Horizontal Performance	1.2 – 1.4-metres	1.2-metres
RTK Horizontal Performance	0.01-metres	0.01-metres
Maximum Data Rate	100-Hz	200-Hz (GPS)
Time to First Fix	<35 – 50-seconds	<24-seconds
GNSS Satellites	GPS, Galileo, BeiDou, GLONASS	GPS, Galileo, Beidou, GLONASS

C.0 Localisation Algorithm Manager Design Iteration

C.1.0 Enabled and Disabling Transform Publishing

Both `slam_toolbox` and `ekf_node` provide parameters for enabling and disabling the `[map]->[odom]` transform. Initially, it was devised, the manager send requests to change parameters to both of these nodes, in order to dynamically change which localisation method was being used. That way, no additional nodes would have to be created in order to facilitate this change.

This was done ROS2's Service and Client interfaces. `ekf_node` and `slam_toolbox` expose a `"SetParameter"` service, which allows for users, and other nodes to send requests via a `SetParameter` client to change the currently used parameters by the given node. Figure 1 illustrates an example of this using the `ekf_node`.

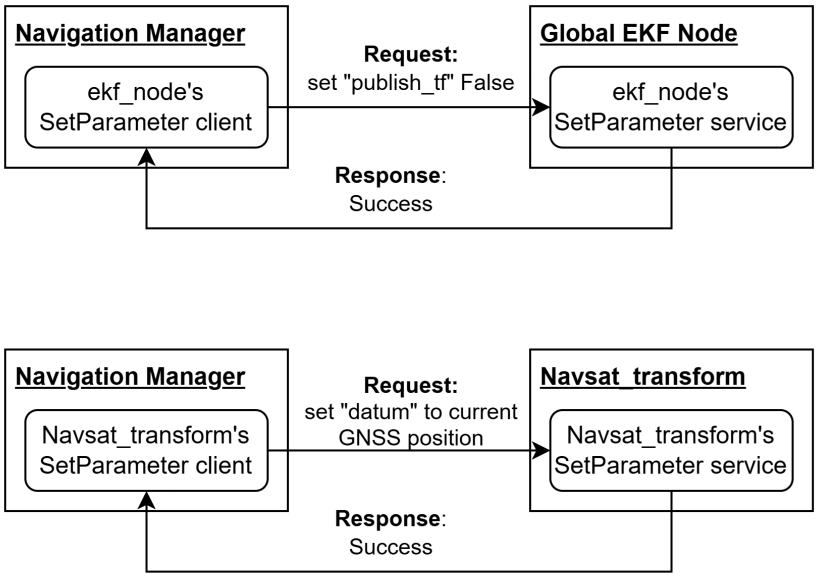


Figure 1: Set Parameter Request Call

However, during testing it was found that `ekf_node` does not dynamically update its parameters. When a request to change the `publish_odom` parameter was send, it would be received and accepted, however nothing would change. The node instead takes a "set-and-forget" approach to its parameters, getting its parameters during initialisation, saving them to variables and then using the variable for the rest of its runtime. Therefore, this approach had to be pivoted away from.

C.2.0 Transform Multiplexing

The next approach taken was developing a transform multiplexer, which could take in the current mode, each mode's respective `[map]->[odom]`, and switch to the required output transform.

This was done using ROS2 transform listeners, and a transform broadcaster. The global `ekf_node` and `slam_toolbox` nodes were setup to have their own `[map]->[odom]` transforms, named `[gps_map]->[gps_odom]` and `[slam_map]->[slam_odom]` respectively. The transform multiplexer would then listen to these transforms, and broadcast the `[map]->[odom]` required to build the correct tree.

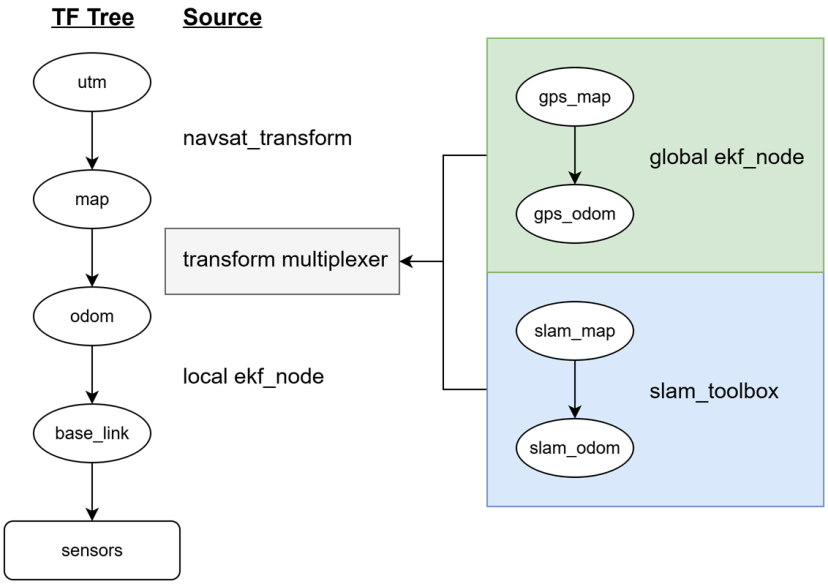


Figure 2: Transform Multiplexer

However, this approach did not work as `slam_toolbox` and `ekf_node` look for the whole `[map]->[odom]->[base_link]` tree, and trying to breakdown the tree into smaller subsets resulted in "`[<mode>_map]->[base_link] transform not found`" errors. Therefore the listener could not find the required transforms, and the multiplexer did not work.

C.3.0 Odometry Multiplexing

Following the transform multiplexer's failure, the approach was adapted to focus on odometry multiplexing instead. In this configuration, the global `ekf_node` was configured to subscribe to a `/odometry/muxed` topic. The multiplexer node would then monitor each localisation mode's respective odometry topic and republish the current mode's odometry data to the multiplexed topic. During GNSS operation, the `/odometry/gps` data would be forwarded to `/odometry/muxed`, while SLAM operation would redirect `/odometry/lidar` data to the same topic.

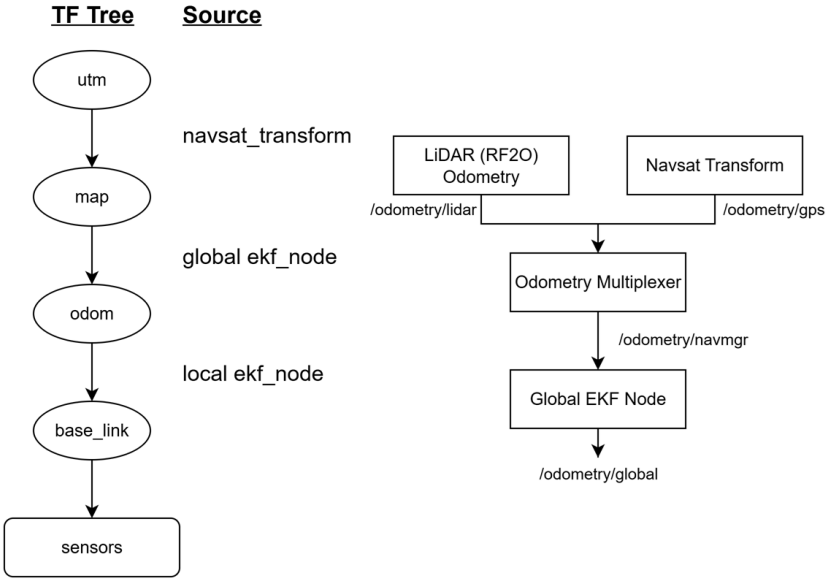


Figure 3: Odometry Multiplexer

Similar to the transform multiplexer, this approach did not work as the LiDAR and GNSS odometry did not share the same reference frame, were not continuous (that is, had different timestamps), and therefore did not produce similar values. Consequently, this resulted in a large spike in the `/odometry/global` output by the global `ekf_node` - spiking the positional values into the billions. Once this happened, the global odometry values could not be restored back to its original state, and the global EKF node had to be restarted.

C.4.0 Temporary Buffer Transform Frames

l After the last two failed multiplexing attempts, a step was taken back away from directly interacting with the transforms or odometry sources. Instead, the transform output of the `navsat_transform` and global `ekf_node` was set to form the `[utm]->[world]->[map]` transform, and a static 1-to-1 tranform was used with a buffer frame to connect this to the local `ekf_node`'s output when SLAM was not active. Then, when SLAM had to be activated, the static transform was disabled, and SLAM was activated and allowed to publish the `[map]->[odom]` transform.

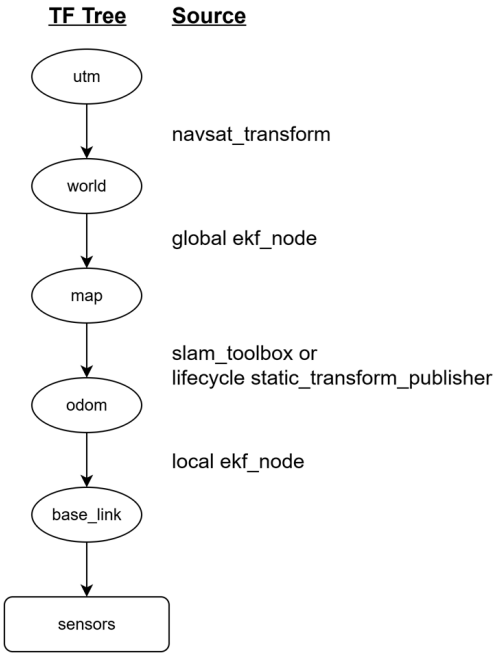


Figure 4: Buffer Transform Frames

This approach showed promising results, as the transform tree was able to be maintained through the life of the navigation manager. However, this approach required the creation of a lifecycle node-compatible static transform publisher, as ROS2 does not provide this functionality natively. While simple to construct, the additional creation of a node was determined to be non-compliant with the evaluation criteria of this project - which underlined ease of use and implementation by the end user. Therefore, a method that did not require the creation of any additional nodes, and could work with the packages selected "out-of-the-box" was prioritised.