School of Electrical, Electronic and Computer Engineering

Final Year Research Project Paper

NUWAY Shuttle Bus Project

Autonomous Vehicle Waypoint Path Planning

Kyle Carvalho

21881544

Supervised by Professor Dr.

Thomas Bräunl

Submitted on: 19/12/2020

Word Count: 7905

Table of Contents

Table of Figures

List of Tables

Abstract

At the start of 2020, the University of Western Australia acquired an electric vehicle (termed nUWAy) with the hardware capability for autonomous movement but without the software to achieve this. This vehicle was planned to be used on campus pathways with students, bicycles, and vehicles in its vicinity. Due to the constant changing and unstructured environment this vehicle will be operating in, nUWAy will need to actively react to its ever-changing environment. The purpose of this paper is to research and alter current path planning algorithms to fit the niche between small robotic autonomous path planning and structured road vehicle path planning by merging both behaviours. The paths will have to be created on-demand between selectable waypoints which lie on a pre-recorded map. Using the Robot Operating System (ROS) to run and test out different open-source planners was key to choosing a path planner that would best suit nUWAy. Search-based Planning Library (SBPL) lattice planner package, provided in ROS, is an open-source path planner which was chosen for its ability to take into account the kinematic movements and limitations of the vehicle. Testing was carried out on a ROS simulation environment termed StageROS whilst then being used on the actual vehicle to test out the paths created with live readings. The movement of the bus was optimised by refining the motion primitives and tuning the planner's parameters to create smooth paths. With the current software infrastructure, the user can select a waypoint through a UI, and a safe path connecting nUWAy to its destination will be created and displayed.

ACKNOWLEDGEMENTS

Firstly I would like to thank Professor Thomas Bräunl for his role as my supervisor. He provided feedback, advise, and direction during this project. I am much obliged to him for the time he took to supervise this project. Next I would like to thank all the other students working on nUWAy for their time and help specially Farhad Ahmed, Yuchen Du, and David Gregory. Finally I would like to thank the other PHD students for volunteering their time to help trouble shoot and design new components for the bus specifically Thomas Drage, Chao Zhang, and Craig Brogle.

Nomenclature

SAE	Society of Automotive Engineers
ROS	Robot Operating System
nUWAy	UWA's Autonomous Shuttle Bus
SBPL	Search Based Planning Algorithm
TEB	Timed-Elastic-Band
DWA	Dynamic Window Approach
PRM	Probabilistic Roadmap
CAN	Controller Area Network
UAV	Unmanned Autonomous Vehicle
UWA	University of Western Australia
REV	Renewable Energy Vehicle
RViz	ROS Visualization
SLAM	Simultaneous Localization and Mapping
TF	Transform
CARLA	Car Learning to Act
LiDAR	Light Detection and Ranging
AV	Autonomous Vehicle
3D	Three Dimensional
GPS	Global Positioning System
IMU	Inertial Measurement Unit
DOF	Degrees of Freedom

1 Introduction

1.1 Background

Autonomous vehicles have recently gained market penetration in different industries. Transitioning from Tesla’s Autopilot mode to the current interest in contactless delivery using autonomous delivery during the Coronavirus epidemic. This increased interest in autonomous vehicles has allowed for more widespread research on the topic resulting in multiple new ways to approach the topic and apply it to different scenarios. Using current navigation technology and combining it with known LiDAR object avoidance methods will be key to the navigation of autonomous vehicles in pedestrian areas.

Autonomous road vehicles can navigate in a structured environment with predictable environmental objects that are clearly labelled such as lane markings, cars, and signs. When operating in a pedestrian area, these vehicles don’t have identifiers like these and thus navigation isn’t as simple as following the known rules between two waypoints. Robotic platforms currently have planners for unstructured environment navigation using LiDAR.

Finding the best selection of currently designed navigation planners for vehicles sized robots will be key to using vehicles in unstructured environments such as sidewalks with cyclists and pedestrians. These planners are key to adopting autonomous vehicles near humans since these environments are too chaotic/unpredictable for current autonomous navigation methods.

After research, choosing a planner , and getting it functional on the vehicle, the next process would be to tune the planner to not only take into account the vehicle’s size and capabilities but to adjust the behaviour. This includes designing a behaviour for the vehicle which is suitable for pathways and ensuring safe operations no matter what.

1.1.1 EasyMile Cybercar

The REV Project at UWA acquired an autonomous capable EasyMile v2 Cybercar Shuttle bus termed nUWAy. This vehicle was delivered to UWA in late February of 2020 with all of the software for autonomy wiped but still containing most of the hardware which is required for said autonomy. Students were then given projects to program the software and other tasks for their research projects to get the vehicle moving on campus by the end of the year.

Figure 1: nUWAy Shuttle Bus

Table 1: Important nUWAy Navigation Components [1]

Sensor/Electronic Device	Importance
Rear and Forward Steering (Rack and Pinion)	The vehicle is a non-holonomic dual-drive Ackerman system which enables a lot more flexibility in terms of navigation and symmetrical forward and reverse driving capability thus can be operated in both directions without hindrances in functionality.
IMU	Important for dead reckoning (predicting the vehicle's position when GPS isn't available)
LiDAR	Important for these reasons: • Creating a map to help the vehicle locate itself • Use for a live view of moving objects in the vehicle's way • Map created from LiDAR will be used to generate the global path the vehicle will navigate on.
GPS	Global positioner which is very important when locating the vehicle on the larger MAP
CAN bus controlled motors and joystick controller [1]	Due to the inability to communicate with the CAN controlled motors, a secondary board and accompanying driver is used to communicate with the vehicle to give it drive commands

1.3 Problem Statement

The major problem is to develop a safe and predictable navigation system for nUWAy such that it can navigate to a waypoint, suggested by the user, on a LiDAR map whilst avoiding new obstacles to the map which were not previously recorded thus actively avoiding obstacles. This navigation system will be compromised of ROS packages which in turn are based on basic motion planning robotic methods. Testing will occur on optimising the speed and behaviour of the packages on the vehicle.

Navigation of any AV relies on its current location, current and past free space, and the destination. Using a prebuilt global map of free space and known obstacles, nUWAy will need to be able to form a path to a given GPS/map coordinate destination. The navigation of the bus will work in a waypoint based system where the bus will navigate through multiple known waypoints before its destination to keep it centred inside the free area and away from the edges. This means the software will make multiple local (smaller paths) paths using spline functions or local costed trajectories to connect these waypoints.

The GPS system will be used both to localise the vehicle and as an updating feature of where the current vehicle is. Dead reckoning (the process of calculating the estimated location by using the last known location and advancing it with the known speed over the course of time) will then be used with the IMU for slow GPS responses, due to poor connections, to make sure the device always has an estimation of where it is.

The following assumptions were made for the navigation of the vehicle:

- The vehicle will have a LiDAR map for global path planning
- The vehicle will always have an active LiDAR view and IMU data
- nUWAy will correctly localised using another SLAM localisation package
- Obstacle avoidance will not include actively moving obstacles since it is out of scope of this report but instead will come to a safe stop distance and will try to navigate around the object instead of predicting the item’s path
- The vehicle can be represented as a unicycle in terms of motion primitives since both sets of wheels can turn independently

2 Literature Review

2.1 Path Creation and Vehicle Control

Before approaching and deconstructing the navigation packages available it was important for me to look at the current method for robotic navigation and truly understand it's technique. The following are the basics about how the paths are generated.

2.1.1 Hermite Spline

Creating a path between waypoints is tricky as you must allow for no erratic movements. The best method to connect all points in a smooth manner is to use polynomials to connect each point. Hermite Spline can be used to create a spline which passes through every control point. The equation for creating a Hermite Spline is as follows: [2][3]

$$h_1 = 2s^3 - 3s^2 + 1$$
$$h_2 = -2s^3 + 3s^2$$
$$h_3 = s^3 - 2s^2 + s$$
$$h_4 = s^3 - s^2$$
$$\text{for } 0 \leq s \leq 1 \qquad f(s) = h_1 p_1 + h_2 p_2 + h_3 t_1 + h_4 t_2$$

Where p1 and p2 are your starting and ending points and t1 and t2 are your tangents and s is your interpolation factor

2.1.2 Steering Angle

After knowing the path, you will be taking figuring out where to point to stay on the path is your next step. It's a simple matter of figuring the angle with the direction you are going versus the angle you need to head. The hard part will be the conversion from GPS coordinates to a global base frame. V is the vehicle's direction vector whilst W is the desired direction given by the direction you need to head to be on the predetermined path. [5]

$$\vec{V}\vec{W} = |\vec{V}||\vec{W}|\cos\theta$$

$$\cos\theta = \frac{\vec{V}\cdot\vec{W}}{|\vec{V}|\cdot|\vec{W}|}$$

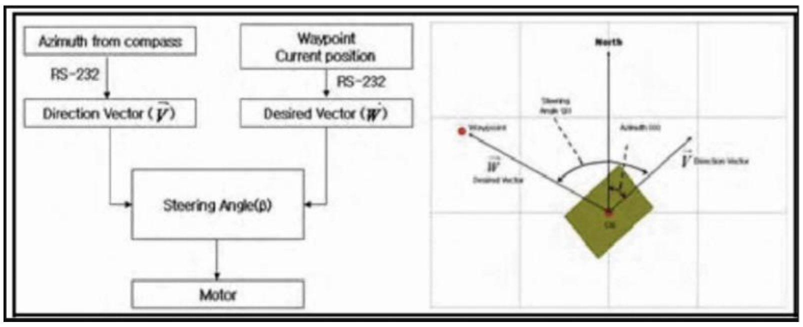


Figure 2: Steering Angle Using GPS [6]

2.2 Object Avoidance

2.2.1 Virtual Points

There are multiple ways to approach object avoidance. The main way is to when an object is detected and a collision is detected the program needs to run multiple different potential alternate paths through alternate or virtual waypoints to get around the object. This new virtual waypoint will need its own generated path calculated and then it needs to be checked if it will still collide with the object. (See Figure 4)

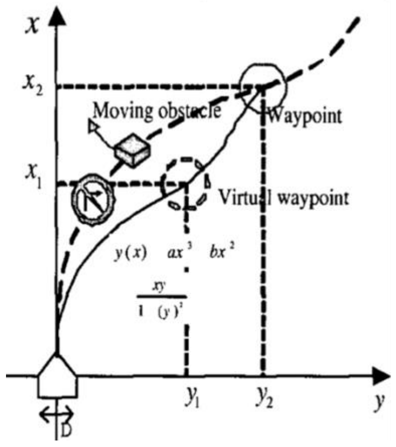


Figure 3: Virtual Point Explanation [8]

There are two methods of generating paths the Voronoi Cell and Cubic Polynomial method. The Voronoi Cell method prevents long detours when multiple obstacles are nearby causing the vehicle to continuously go further from the base frame. Due to the time taken to calculate the Voronoi Cell path it is not feasible to do in the nUWAy Scenario. A simple Cubic Polynomial method will work.

2.2.2 Path Planning According to distance to Base Frame

To prevent the vehicle from veering to one side of the road it is important that the base frame (the initial path calculated at the start of the journey) be located at the centre of the free space area that the vehicle navigates through. This gives it the most room in case of bigger obstacles later thus allowing for more area for manoeuvring. The best example is illustrated in Figure 4 where the best path is chosen because it passes through no obstacles but also because it is the closes to the originally base frame bath in black. This path selection also accounts for smoothness of the curve. These variables are weighted and the most ideal path is selected [4].

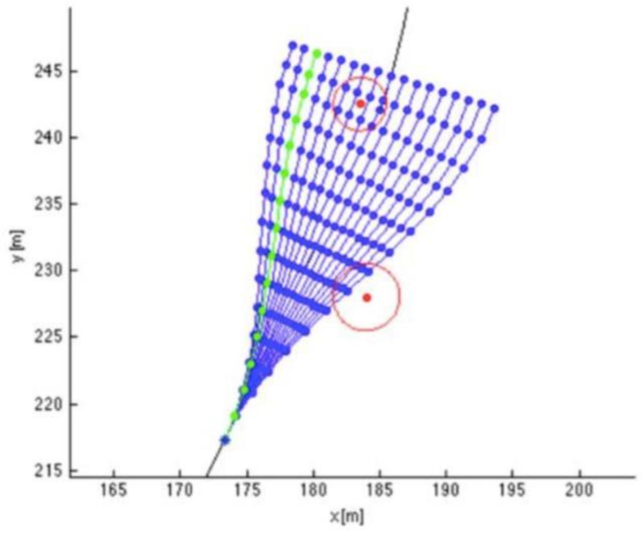


Figure 4: Path Selection Considering Object (red) Avoidance[4]

2.3 Move Base

Move base is the process by which robots on ROS's navigation program operates in terms of information flow (inputs and outputs). This navigation standard is core to all ROS motion planning and will be required software architecture for the navigation stack on nUWAy[13]. When looking at move base you can see after a goal is given as an input the process is as such:

1. Input goal/pose has come as a geometry message (X, Y, Z, theta) on the map
2. Global planner receives destination information and access costmap information. Costmap information is the LiDAR map information with an additional safety factor

added increasing the size of the objects to prevent the vehicle from operating too close to any object.

3. The global planner generates a generic path using the map and destination
4. The path is then passed to the local planner to break down into achievable motions which the vehicle will receive as a command velocity messages. The local planner will also use the local costmap to avoid active moving obstacles whilst still trying to stick as close to the generated global path as possible.

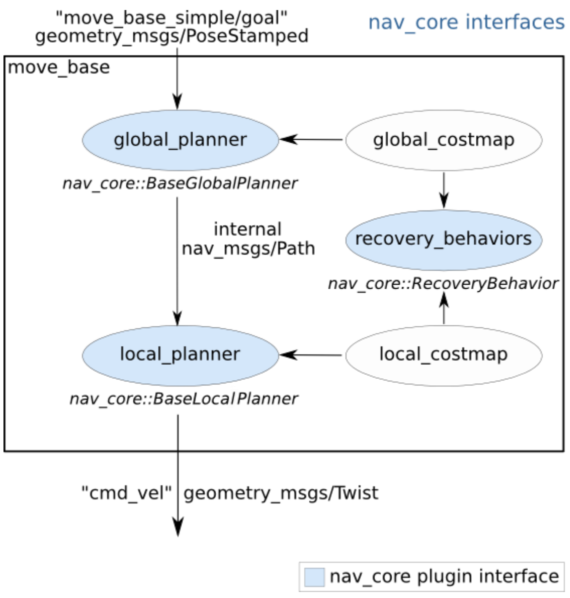


Figure 5: Move Base Break Down[13]

The global planner works with the total map information but the local planner works with what the vehicle currently sees and interpolates that with the direction given by the global planner's path to give the vehicle drive commands[13].

2.4 Planner Types Literature Research

Sampling-based planners work by randomly exploring the space available to try and find the best path. It will generate a random tree that explores the space and produces a varied but still poor quality path since there the main metric is to measure is the absence of collision. When run for shorter intervals the problems gets worse since it can not fully explore the free space area thus it will not be the planner of choice for a vehicle which knows where it wants to go. [15]

Variation planners work by attaching a cost to the to a path and velocity function and functions successfully only when there is a finite small amount of possible outcomes otherwise the computational load becomes too high. The path created is judged according to multiple different variables including, vehicles possible kinematics, closeness to a collision, etc… This type of planner will be used for the local planner to decide what immediate motions are needed either when turning or avoiding a smaller obstacle[12][9].

Search-Based Lattice Planner starts by discretising the total free space area and creates a list of possible moves that are feasible which are called motion primitives. From then onwards the paths created will consist of multiple motion primitives connected. By simplifying the system down to a set of possible moves the searching algorithm can work much faster than the Sampling method. A searching method will be needed to speed up the time needed for the planning and each path will still best costed to find the most cost-effective path. [15]

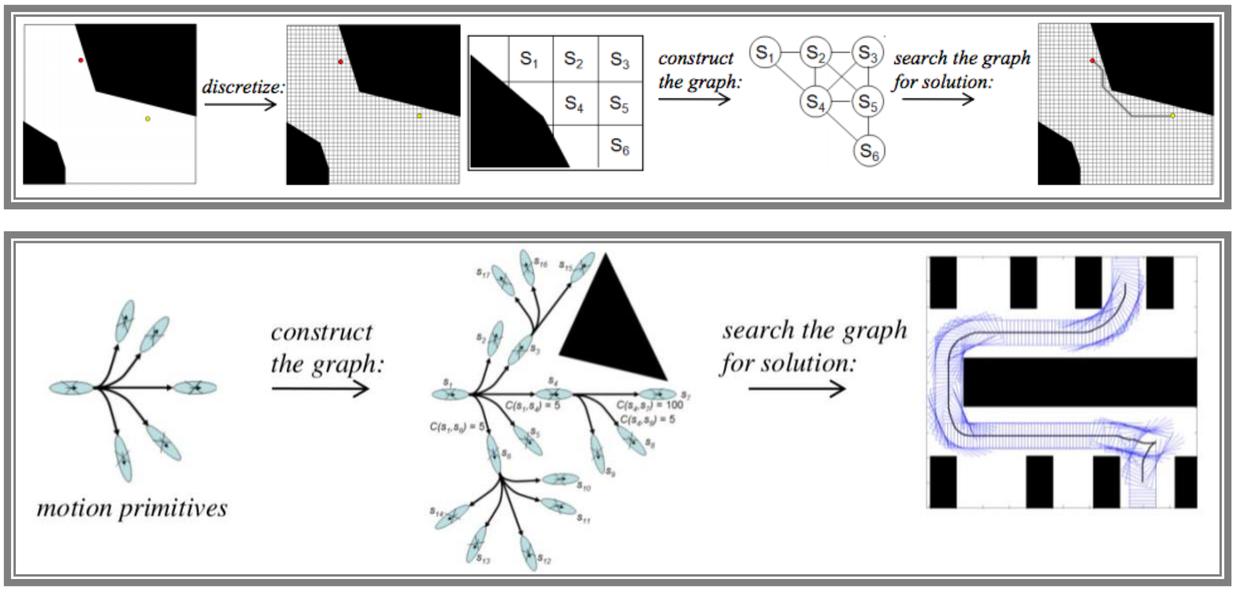


Figure 6: Discretising and Constructing a Primitive Path for Search-Based Lattice planners[19][15]

3 Design Process

3.1 Requirements

The program created will need to be able to take in a final destination, map of the navigable area, live LiDAR reading and create a path which can is kinematically feasible, time-efficient, and has a predictable safe behaviour. The programmed system needs to output ROS Twist messages for velocity and it needs to work as part of the Move Base ROS package's Navigation Stack. It needs to be self-contained such that it can easily be replaced and altered such that during agile development.

3.2 Constraints

The following limitation controlled the final design of the program created:

- The computer available on the bus was not up to the processing required for Navigation and Simultaneous Localisations and Mapping (SLAM). Moving other processes to the Xavier (secondary computer) helped speed up the problem. A larger resolution map was required to speed up the path generation causing less accurate paths to be created.
- The X-Sens IMU produced a lot of noise thus producing a lot of error in the dead reckoning. The solution was to disregard the IMU readings and rely purely on the RTK GPS unit due to its high accuracy. This is only a solution due to the vehicles very slow speed and will eventually require a heartbeat safety connection to prevent the vehicle from driving when GPS connection is lost.
- The fusion of GPS to waypoint was not yet possible. It is still in the works but the current system works purely off pre-registered map coordinates.
- The LiDAR on the vehicle prevents scans of items below 30 cm which prevents accurate maps of drivable terrain to be

3.3 External Effects on the Project

There were a few major external factors which effected the completion of this project and due to time constraints narrowed the scope of the program.

- The COVID-19 Pandemic: Straight after the onset of acquiring the vehicle the pandemic and it's lockdown hit. This prevented any physical work on the bus for a long time. This slow down not only effected my thesis (since I had less time to get the sensors operational) but also due to other students not being able to access the vehicle thus delaying their work too. Other work including the SLAM, Pose Determination, and Mapping was part of the critical path for my thesis thus delays to all of these research projects cause delays to my testing.

- Lack of control over motors: Due to the motors being controlled by a non-decipherable CAN network, control of the vehicle directly to the motors was not possible this year. Instead, another student-designed, developed, tested, and produced a communication board which could send messages from the computer straight through the joystick controller input to the vehicle. Developing testing and creating a ROS driver for controlling this board took time and was only completed in November of 2020 thus not allowing for autonomous testing till December of 2020.

- Since this project relies on multiple student's projects, multiple delays were incurred due to sharing the vehicles. Delays such as the vehicle being used on other tests, errors causing loss of functionality in the bus, and the continuous changing of settings and programs in the bus. This is natural in a group project but this in combination with the COVID pandemic cost months of progress at the start of the year since the vehicle wasn't operational for navigation till the end of the year.

3.4 Basics of ROS Navigation Explained

3.4.1 ROS

Robot Operating System (ROS) 1.0 is the system by which communication between ethernet-based electronic components is hosted on nUWAy. Even thou it is termed an operating system it is a misnomer since the system runs on a Linux operating system. nUWAy itself is running ROS Kinetic Kame on Ubuntu 16.04 Xenial on two different computers which is an older version of ROS. One being the main nUWAy based computer and the other being a Xavier external computer. Every sensor’s output (LiDAR, GPS, Camera and IMU) can then be connected to a computer network to allow the main computer to run ROS base packages to access said data and manipulate it. These packages range from drivers to pre-built manipulation programs which accomplish a range of different tasks.[25]

ROS has an open-source library consisting of many previously worked on packages. These packages are specialised for robotic vehicles and ranges from reading in and interpreting data to movement and mapping. ROS’s libraries were heavily used to interact with nUWAy for both readings in data and controlling the vehicle.

3.4.5 ROS Navigation Stack Move Base

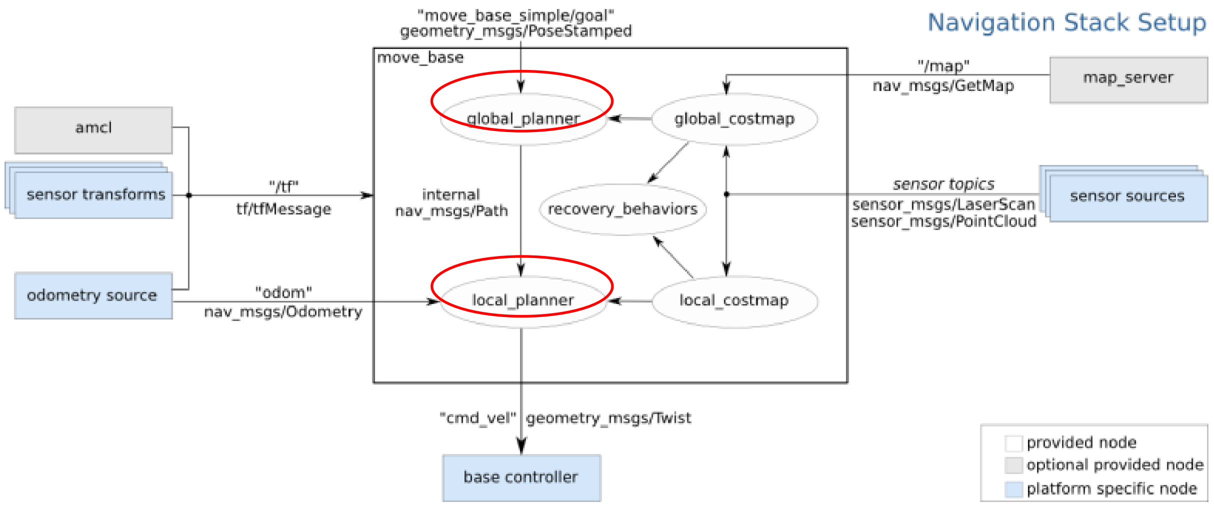


Figure 7: ROS's Navigation Block Diagram (Highlighted in Red are the planners) [25]

Move base is part of the navigation stack set up by the ROS environment. The navigation stack is a term that describes all the programs that determine the motion planning of the vehicle. The base layout, see figure above for illustration, requires 3 major inputs which are sensory inputs, a map, and the transfer of the vehicles which in combination can be used to design a path and output velocity commands (in the translational and rotational directions). The parts this paper will cover is the selection and alteration of the global planner and local planner whilst using all the other components from the Move Base package unaltered. [16]

Move base requires both the current position of the robot and the map of the surroundings to actively track the position of the vehicle compared to its path on a Lidar Map. These two functions (pose determination and mapping) will need to be fully developed before vehicle testing can occur and thus are part of the critical path for testing this system on the vehicle. Both of these programs are being developed by another student and thus working closely with him was core to developing my system. Until SLAM and pose estimation is fully developed rigorous testing will occur in simulation using prototype maps and test environments to source the best planners.

4 Final Design

This section goes through the information required and tools used to test and integrate ROS's navigation stack to the vehicle.

4.1 Design Process

After the research phase of the project, it was clear that it would be inefficient to design program a navigation system using robotic motion planning programs from scratch. Instead, the project was turned into a research and application study on how to optimise current motion planning methods on smaller robots for actively changing and unstructured pathways on a shuttle vehicle sized robotic platform.

4.2 Design Tools: ROS Packages

The selection process took into account a range of motion planning methods and weighed their positives and negatives and the final tests in the results selection measure their efficiency and reliability.

4.2.1 Planner Selection

The measurements for a successful planner setup are the following:

- Quickest path generation time
- The fewest amount of recalculations of path
- Customisability of behaviour for safety purposes
- The overall time that is taken to complete path
- Reaction time and behaviour to new obstacles
- The smoothness of the curve to prevent harsh turns
- Distance from objects and their costmap

Two planners needed to be selected. The global planner for planning the whole route and a local planner will act to reach points on that global path by sending it commands for motion and avoiding new obstacles. The path generated will consist of X-Y coordinates and a final direction constrained to Ackermann (non-holonomic) kinematics. Notice that the path created by the global planner does not consider the direction or pose of the robot and purely determines the points the vehicle should navigate through however the local planner's path will have the desired direction as well as the X, Y coordinates to navigate through.

4.2.2 Global Planner

The global planner works by calculating the best route off the map information alone. There are multiple methods for achieving this. The major difference when selecting them is in terms of computational speed. The majority of the considered global planners work in the same overall process[15][18]:

1. Break the maps down into multiple different cells
2. Generates trees which are paths that will explore the available area either randomly or using a searching algorithm (A*, Djikstra, or Weighted A*) creating nodes which will have the cost of navigating through that area in it.
3. Each node's cost on a successful goal path will be totalled to obtain a complete path cost
4. Finally, the path with the lowest cost will be selected either after a time out or when the first viable path is found.

The carrot planner is the simplest form of a global planner since all it does is directs itself towards the goal without much regard to the map as a whole. The planner essentially finds free space on the way to the goal and places a waypoint there. This form of waypoint management works well to speed up the processing as it doesn't consider kinematics, map layout as a whole, or multiple paths. The local planner will then solely direct the vehicle when close to obstacles and manoeuvring. It passes all the computation to the local planner to handle whilst making the vehicle essentially start searching randomly for the final location. The Search-Based Planning Library package works in the following method is a much more intensive planner in terms of computational load. To see its process see Figure Discretising and Constructing a Primitive Path.

4.2.3 TEB Planner Local Planner

Time Elastic Bands (TEB) will create a new global path with vehicle poses making up the list of intermediate nodes on the path. It uses the kinematics of the vehicles by using the full range of both the vehicle's velocity and acceleration to give the vehicle command velocity in both the forward translation and yaw rotational axis. This method of local planning will encounter problems when a moving object comes into view and keeps getting in the way of the newly created collision avoidance path. This newly generated path will keep recalculating whenever the same object moves into the path of the vehicle. The problem is solved by generating multiple paths simultaneously. The first using the newly created avoidance path, the second as an offshoot of the original path and finally a new path not taking into account any of the previously generated maps. These created paths are termed elastic bands which occur at certain time intervals[14]

4.2.4 DWA Planner

The Dynamic Window Approach(DWA) works in much the same way in terms of calculating the kinematic trajectory of the vehicle. Instead of finding the first available path of three choices the planner runs multiple trajectory calculations and scores each trajectory according to multiple different variables. The highest scoring path would then be used to control the vehicle[17].

4.2.5 Goal Passer Planner

The goal passer planner is the simplest form of a local planner. It estimates the pose needed for the vehicle to stay on the path. Any deviation or interruptions in the path will send out a call to recreate the global path. The slowest and least adaptable planner and should only be used when computational power is limited[11].

4.4 Simulation

Stage ROS is a very simple simulation environment used to input a custom environment's LiDAR readings and location reading. It was used when testing the code in simulation as it allows us to input test environments and see how the vehicle reacts to different kinds of situations. It even has the ability to add mobile obstacles and actively move it. It was core to developing the brains and tuning the vehicles navigation stack[16].

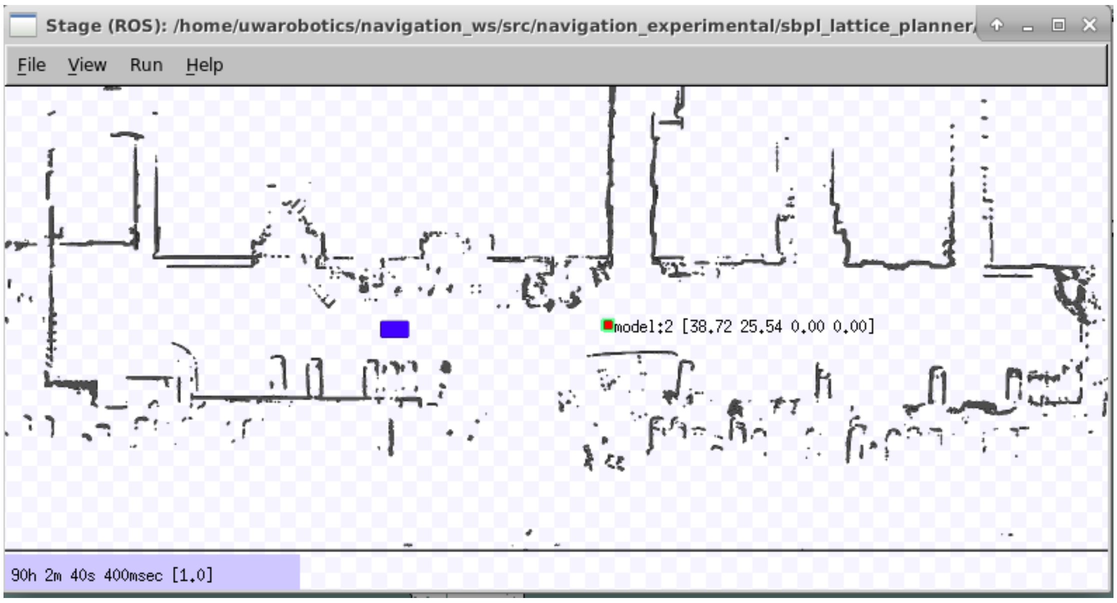

Stage (ROS): /home/uwarobotics/navigation_ws/src/navigation_experimental/sbpl_lattice_planner/
File View Run Help
model:2 [38.72 25.54 0.00 0.00]
90h 2m 40s 400msec [1.0]

Figure 8: Stage ROS Simulation Environment

CARLA is a system by which autonomous vehicles are trained to operate in cities and urban environments. This system would be great in designing a vehicle behavioural algorithm. Unfortunately creating a CARLA simulation uses a unity based engine and time to create a bridging communication program to use ROS programs. This task was undertaken by a group of students this year but due to the difficulty of the project, it wouldn't be fully functioning until next year thus disabling testing for this paper.[16]

4.5 Real Life Board

A communication board was created to take in ROS's vehicle command messages and translate them into voltage inputs for the joystick controller to then send CAN bus messages to the vehicle's motors. This was done because the team didn't have access to the decoding method of the CAN Bus messages used by EasyMile.

4.6 Project Management (Semi-Agile and Waterfall)

This project will range from a research phase, simulation phase, to implementation on nUWAy phase. A waterfall method of full completion will be required before transitioning between each phase since the information from the previous section is core to proceeding into the next phase of the project. Without research into the availability and usage of planners, a proper simulation cannot be developed and tuned. Without a proper working simulation, the vehicle's behaviour cannot be trusted. An agile approach was undertaken in the latter two phases since in each phase since new problems would be discovered causing the whole simulating/vehicle testing phase to restart with the newly altered program.

4.7 Other nUWAy Work

NUWAY was a very large project with a lot of active components needing replacement, calibration, or setup. Due to the scale of the project, a lot of time was invested in parts outside of this paper that will be documented in this section.

4.7.1 GPS and IMU

Originally starting with the vehicle's OEM Novatel GPS, a lot of time was invested in calibrating the GPS and trying to read the out. The GPS was an older model NOVATEL PwrPck 6 prevented the sue of ROS drivers. I started by creating a custom driver and tried deciphering the communication protocol. Eventually, after consultation with a PhD student who specialises in autonomous vehicles, another driver was suggested. In the end, the system was working using an NMEA NAVSAT ROS driver as was much more reliable than the current driver in development.

After obtaining an Xsens IMU device to add to the vehicle, testing occurred to obtain the IMU's base setting. Integration to enables its input in Move Base took time due to the inaccurate data it was producing. After testing and calibration, it was found to be caused by a combination of an improper mounting axis and due to the fact it was in close vicinity to a metallic object. After creating a custom mount the Xsens produced much more stable results.

4.7.2 SAE Brake System

Part of my duties as a member of the REV team was to acquire and test a new pneumatic braking system for the SAE self-driving car. The current electric power braking system would not run if power was cut thus a spring-loaded pneumatic breaking system was designed. I was in charge of acquiring the parts testing the pressure and making sure the design would meet the specification set out by the design.

4.7 Design Process Evaluation

After research and creating the motion primitives and specify the motions costing for the vehicle, the design process will go as follows:

1. Initial phase on getting the simulation functioning with a basic carrot planner and the simple goal passer local planner in a basic environment
2. Test the grading criteria, stated in section 4.2.1, of the basic layout whenever a new planner is added the criteria must be tested
3. Change the major control variables as per the tuning guide specified earlier in the paper
4. Change the map to that of the parking lot behind engineering at UWA where the bus will be tested
5. Change the searching algorithm to see what would best work in a defined area that is not very large
6. Install and configure the SBPL planner with a basic Goal Passer local planner and test accordingly
7. Install the remaining DWA and TEB Local Planners and test out their speed and functionalities.
8. After final testing, select the best option for the planner and proceed to install packages on the bus and run the navigation stack on the bus and measure changes in speed and see if there are any changes in characteristics in path generation.
9. Test in an open area with the custom communication board and limit the speed
10. Gradually bring the vehicle to a more diverse environment in the parking lot and slowly allow it to navigate around with two safety operators always looking out to make sure they can stop the vehicle remotely.

5 Final Design, Results, and Analysis

5.1 Final Simulation Results

5.1.1 Initial Testing

The figure below illustrates the base SBPL planning algorithm's path with basic customisation given a simple straight navigation command.

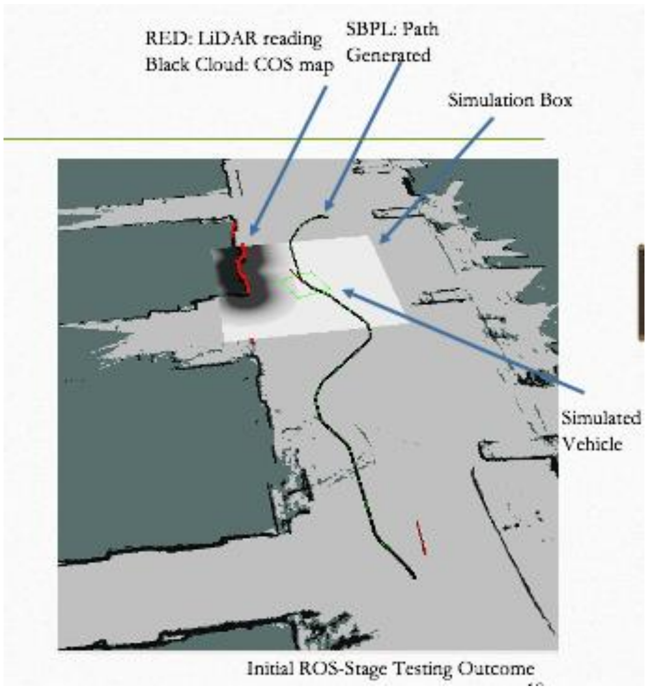


Figure 9: R-Viz view of Initial testing with Little Customisation

The major problems with the programs are as follows:

- Erroneous errors in a very simple straight-line navigation task. Unexpected turns placed the vehicle in contact with the sidewalk and its behaviours are not understandable or predictable.
- Small errors in mapping caused the vehicle to overreact and cut off the program early.
- Instead of turning in place or using a small amount of space, the vehicle used all available area to make a small turn which is not ideal for navigation in pedestrian areas.

5.1.2 Motion Primitives and Behavioural choices

Motion primitives are customised precalculated motions that are kinematically feasible for the vehicle to take which have costs associated with them. These must be generated for use in the SBPL planning algorithm such that the planner can attach theses kinematic motions together to create a path [15].

When designing the motion primitives of the system the costing and biasing of certain motions was a key design decision to the behaviour of the vehicle. The motion primitives are generated with the use of MATLAB and are stored as a table for quick look up by the planner when required.

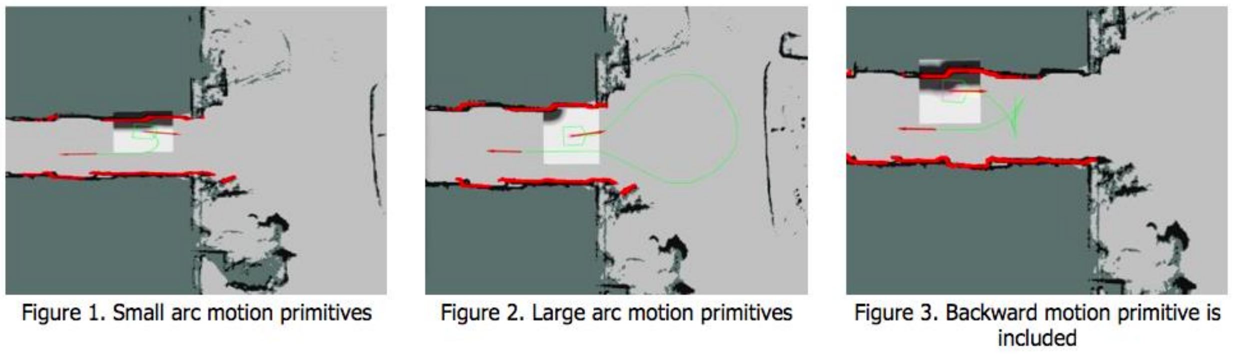


Figure 1. Small arc motion primitives
Figure 2. Large arc motion primitives
Figure 3. Backward motion primitive is included

Figure 10: Example of behavioural choices made

The following decision was made when designing the motion primitives:

- Assume that the vehicle's kinematics can be simulated as a non-holonomic unicycle. This is due to dual-axis Ackermann steering that is active when the vehicle is in operation
- Cost reversing motions much higher and try biasing forward motions since the vehicle shouldn't be reversing in populated areas as it will put pedestrians in a state of unease. This mean reversing will only occur as a last case scenario and at failure states.
- Bias left-hand side drive motions by costing it lower. This is to mimic current Australian road vehicles thus making sure if the vehicle has a new obstacle car or human it will reliable stick to the left-hand side when navigating around it.
- Cost turn in place much lower than what is recommended. The vehicle is much larger than planned for this program thus it will not have a lot of places to turn. So the vehicle should bias turning in place rather than movement. This is also a much safe behaviour since it will take up less space doing it. For example, see figure 10.

An example of the final motion primitives are listed below accompanied with a combination of how all the motion would look overlayed:

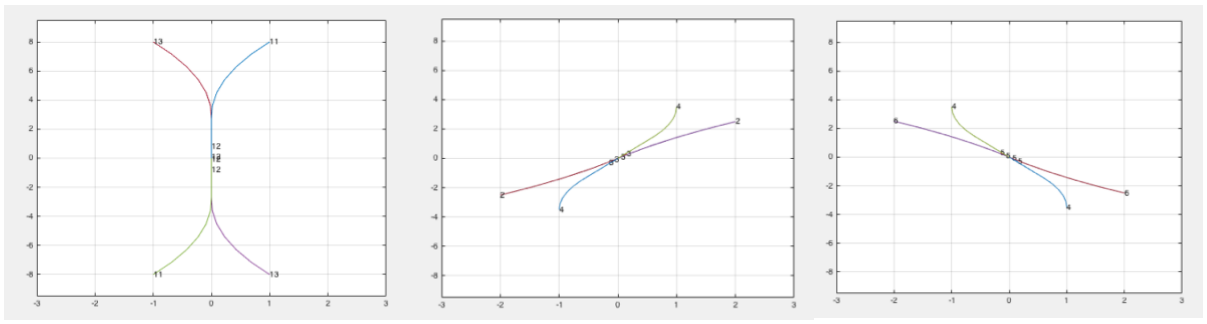


Figure 11: 3 of the 8 Motion Primitives generated by MATLAB Graphed

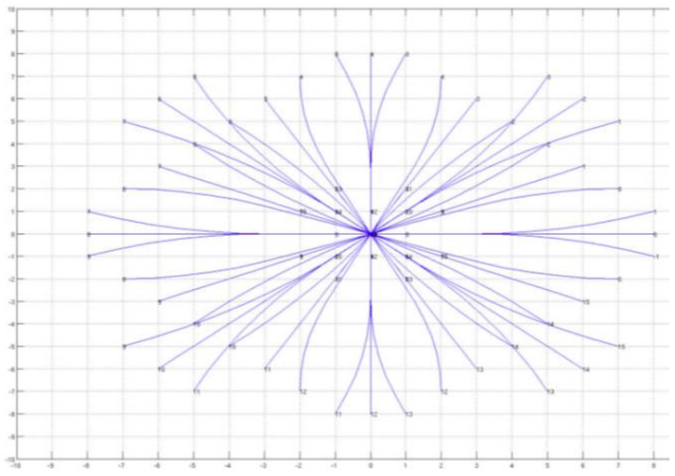


Figure 12: Combined Motion Primitives achieved with a unicycle model [15]

5.1.3 SBPL and Carrot Planner

Then deciding which global planner to use the following data was collected to confirm that SBPL was the correct choice. Note: These tests are the average of 10 runs each. The map-wide metric contains hallway and short tight turns whilst the short steps consist of simple turns or movement from one room to another. All testing was done in ROS Stage using a standard street layout out the map with walls surrounding the vehicle’s free space.

Table 2: Global Planner Evaluation

Evaluation Metric	Base SBPL		Carrot Planner	
	Map-Wide Navigation(s)	Short Navigation Steps	Map Wide Navigation	Short Navigation Steps
Time For Generation	6.42	1.28	6.12	1.22
Path Accuracy	High		Low (due to poor kinematic planning)	
Recalculations Necessary	4	0	12	2
Turn Radius	Relatively small and more feasible turns		Very high due to imprecise path connection	
Smoothness off Path	Path generated with quadratic approximation thus it was smooth		Jagged and imprecise at discontinuation points	
Customisability	Well documented and thus much easier		Documented but doesn't give much option for expansion	

It is evident with regards to almost every metric expect time that the SBPL planner is the obvious choice. The lack of complexity in the carrot planner assists with the initial planning phase but due to the lack of kinematic planning, the SBPL planner creates much more feasible paths which requires fewer path recalculations.

5.1.4 SBPL Optimisation and tuning

There are three major available searching algorithms. Without going into the specificity of each algorithm the breakdown is as such. Dijkstra spreads out and tests all possible nodes in the discretised area starting at the base point. It keeps searching all free space nodes and evaluating them in an orderly emitting method. Eventually getting to the goal. A* uses the known location of the goal as a sense of direction to search in that direction. Weighted A* goes one farther by costing nodes closer to the goal on a gradient such that the nodes closer to the goal is not only favoured but also fully explored. The Figure below shows how ineffective Dijkstra is and how much more effective (measured in terms of total nodes counted) the Weighted A* method is thus it was the chose algorithm.[23]

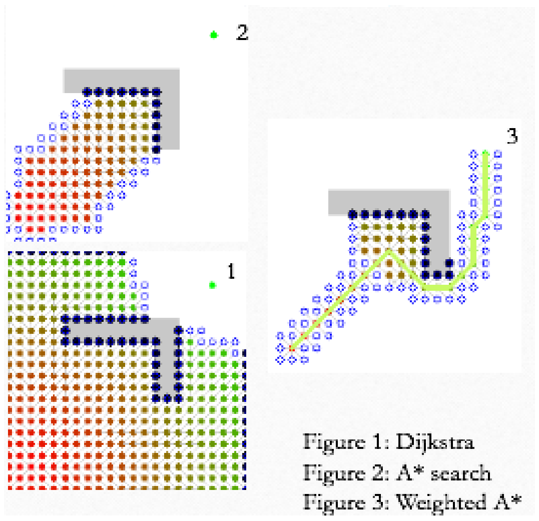


Figure 13: Different Searching Algorithms [15]

When tuning the simulation settings for this vehicle the following were chosen:

- Allowing quadratic approximation to increase the speed of the searching algorithm
- Turn off-grid path to allow for softer turns and smoother paths as this enables curve smoothening.
- Use the rated acceleration and velocity as the input maximum and minimums since in simulation the vehicle must be fully tested
- Restructured the behavioural timer to prevent too long planning times

5.1.5 SBPL Cost Tuning

The costing factor for vehicles is an important part of behavioural analysis since it determines how the vehicle approaches obstacles. A range of 1-233 for the cost factor is the given range available. Post testing it was discovered a cost of 66 allowed for near-wall operations without the approach to the said wall being disturbed as illustrated in the Initial Testing Section above.

The costmap is a bubble which surrounds objects in a simulated environment which essentially inflates the size of objects as a form of additional security to prevent vehicles from operating too close to objects or turning into them. The initial simulation had very small inflation radius which caused the vehicle to operate dangerously close to structures. However too high inflation radius will prevent finite motions. The decay rate determines by which the costing scales down when close to objects. When ideally tuned, the vehicle will cost when navigating tight spaces correctly and always stick to the middle of said space. Finally, the resolution of said costmap will be determined by either your map's resolution or the LiDAR's resolution. The following variables were tested and chosen[11]:

Costmap resolution: 0.03 (determined by the map)
Cost scaling factor: 2.2
Inflation Radius: 1.2

5.1.6 Local Planner Tuning

Using the DWA local planner the following tolerances were required to prevent the vehicle over adjusting itself to the path and instead uses the global path as a guide. The following parameters were set[10][11]

- Translation Tolerance: 0.1
- Latch Tolerance: False (leads to the robot trying to match the goal pose to closely leading to delays and unpredictable pactions)
- Yaw Tolerance: 0.05

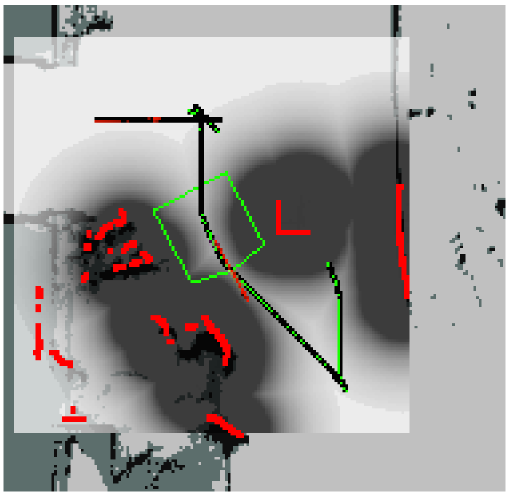
Sim-Time is the given time for an operation to occur at a max set velocity

- Sim-Time: 3.4 (set between 2 to 5 for feasible results)

Simulation granularity: Step size for each path trajectory. This can create a lot of computational loads thus not minimising this variable is key to decreasing computational run time

- Simulation granularity: 0.04

Figure 14: Rviz view demonstrating Obstacle avoidance and tight maneuvering working correctly



The figure above illustrates the obstacle avoidance and created by the local planner. It functions slower and with only slight adjustments to make sure the vehicle narrowly navigates the free space.

5.2 Final Simulation Results Discussion

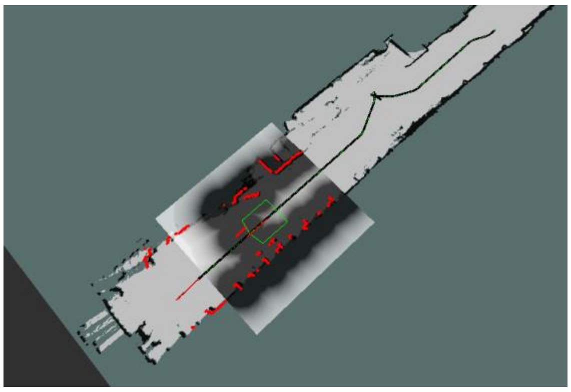
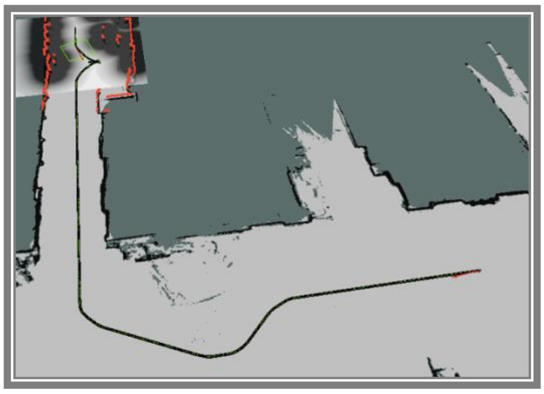


Figure 15: Path generated in a sensible predictable manner with a safe distance maintained



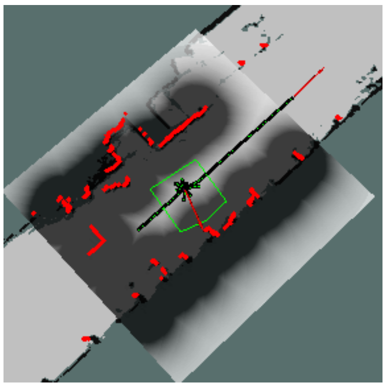


Figure 16: Final Prototype Simulation Path Generation

Figure 17: Error Recovery Behaviour in Action

When looking at the images above and comparing it to the initial testing images at the start of section 5.1 its is evident that tuning and behaviour alteration testing worked. The following are the outcomes from this Simulation Design phase:

- Paths formed have very large turning radii thus creating smoother paths
- Paths formed are logical and no longer overreact to stimuli
- Vehicle biases the left side of the Road
- Reliable distance between objects and the vehicle is held unless in an error state
- In error states, the vehicle operates slowly and does not go too close to any walls
- At goal, vehicles terminate predictably and in the correct pose

- Recalculates according to effectively if the surrounding changes as compared to that of the vehicle's recorded map

The simulation stage of the project was a rounding success as it produced a solution which should be able integrated on to the vehicle through ROS. The only major concerns are that fact that the system did not operate on a low-end computer when simulating thus computational power might be a problem whilst on nUWAy.

5.2 Vehicle Testing Results

5.2.1 Initial Vehicle Path Generation Testing

The vehicle was initially tested by displaying the path it would hypothetically follow on R-Viz. ROS Vision is used to display what the vehicle sees concerning the vehicle's placement in 3D space thus it is perfect for displaying the path created. Upon first software migration from simulation to hardware, there were a lot of integration problems. When on the simulator the there was little to the limitation in terms of computational power. Unfortunately, after the transition from simulation to the vehicle, a lot of processing and unpredictable errors occurred. Errors/Loss of functionality include the following:

Table 3: Initial Testing Failure Analysis

Loss in Functionality	Causes	Consequences
Time taken to generate path varied from 40 seconds for simpler paths to 2 minutes for complex paths	Slower computer in combination with more sensors slowed down the processing speed of the planner. The planner was also initially set to a very low resolution	The vehicle was non-operable or testable due to long wait times preventing a proper localisation to initialise a full path
Loss of continuous R-Viz view due to the computer freezing and crashing		User feedback was completely removed thus all testing information would be lost

Vehicle’s behaviour didn’t resemble what was predicted recorded in simulation	Simulation made assumptions that were not true in real life such as using a unicycle model for the motion primitives	Programs would need to be individually tested to find out the failure point.
When a path was generated the vehicle’s sensors wouldn’t allow for the vehicle to remain stationary for long enough to test if the local planner was functioning adequately	Sensors were not calibrated correctly giving erroneous data preventing SLAM from outputting a proper initial pose.	Navigation couldn’t even start processing or begin the Search-Based Planning process without an initial location and pose estimate
Navigation stack not recalculating after the initial path is generated	Vehicle is not stationary for long enough to allow for a recalculation and thus the whole path must be recreated	Excessively slow local planning and remapping

5.2.2 Fixes from Initial Tests

Majority of the problems caused were due to the speed of the vehicle’s processor being to slow thus fixing this was the primary issue. The first part to change was the planners themselves. The local planner was downgraded to the simple goal passer local planner. The SLAM algorithm was changed to a more widely supported and faster cartographer package then the base G-mapping program. The SBPL and SLAM mapping resolution was changed from an original 5 cm to 30cm. This includes recalculating the motion primitives for the current resolution map.

All other Programs were moved over to the secondary computer leaving the base computer only for communication, navigation, and SLAM. The SBPL settings were tuned for speed.

Grid Path behaviour, reduced inflation size, and use of A* algorithm all sped up the processing of the navigation stack.

New RTK GPS antennae were ordered to replace the indoor Xsens GPS antennae and the IMU readings from said Xsens were disregarded. The vehicle would use the LiDAR's odometer for detecting motion and predicting the vehicle's pose instead of the IMU.

5.6.3 Subsequent Vehicle Path Generation Testing

With the new system path generating occurred at much faster speeds. Sometimes taking up to a maximum of 4 seconds. The reasons the speeds were so high were because the complexity of the system was reduced. This includes using a simpler local planner, reducing the sensory inputs, and running the system a high-resolution system. Increasing the resolution reduced the amount and type of movements available to the vehicle.

As seen in the figure below the paths generated with the simplified system were much more ridged. This was mainly due to the tuning and resolution change of the base SBPL package. These paths remained feasibly possible but started to suffer when close to objects. The local planner would purely keep the vehicle on track and refrain from recalculating the whole path when a new object came into view thus effectively losing its function.

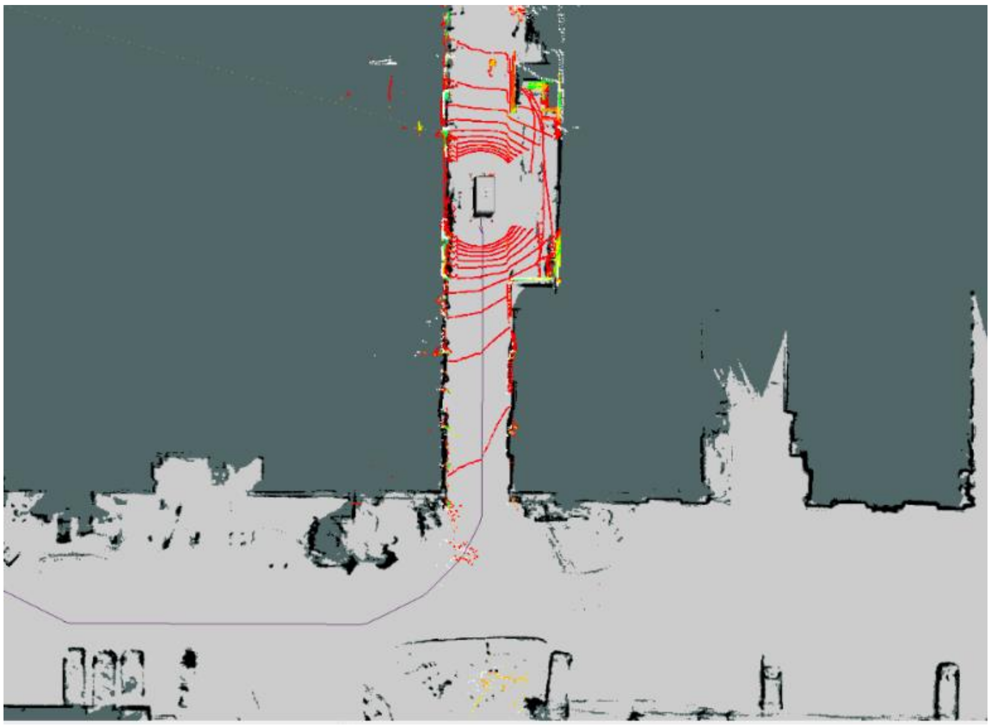


Figure 18: RVIZ view of Secondary Path Generation Testing in Working Condition

5.2.4 Fixes from Subsequent Testing

The following trade-offs occurred to get the vehicle ready for actual autonomous driving:

- The basic local planner, Goal Passer, was replaced by a custom planner named the desired direction. This custom planner was used by another student's thesis to help assist with localisation. It would replace all the other function of the local planner and purely remain on the path. This eliminated the active obstacle avoidance from the navigation stack.
- The maximum speed was capped to half the simulations speed to make sure all manoeuvres can be seen by riders in case the autonomy fails and requires human intervention.

5.2.5 Autonomous Driving Final Testing

The final section of testing occurred just before this paper was written. It consisted of the vehicle operating purely off the navigation commands created by ROS' navigation stack. After final testing occurred, no other changes to the vehicle's behaviour occurred due to the need for all other students to record data thus the project reached its completed stage

Table 4: Autonomy Testing Results

Flaws in Function	Causes
Paths generated did not function when in tight environments	Resolution of the generated motion primitives was too high
Vehicle would navigate into off road areas	Poor mapping in high resolution missed some low lying barriers causing undrivable areas to come up as free space.
Speed varied highly and wasn't as continuous as the simulation	The custom local planner requires a proper speed controller with a PID system
Vehicle would sway on straights	Higher tolerance for goal acceptance required to stop the vehicle over correcting its direction on straights
Limited to no recalculation occurred when the vehicle fell off its path	The local planner did not communicate error states to the global planner. A separate function needs to be created to notify the global planner a recalculation is needed.

6 Discussion and Conclusion

6.1 Improvements

Improvements to the project can be narrowed down to increasing the speed of the path generated whilst bringing over the complexity of the simulation's navigation stack to the vehicle. The major suggested improvement would be the upgrade of the computer in the vehicle or the transition of the navigation stack on to a more capable device such as the secondary computer. This upgrade would remove half of the issues that were created when transitioning the package from simulation to the vehicle as listed in section 5.6.1. This new computer would generate much more accurate path and would be able to run the TEB and DWA customised local path planner advised earlier in this report to see which will run better.

Mapping of the area needs to be done at a lower but more precise resolution. This will enable more complex and finite motion by the navigation stack and remove a lot of the overcorrection errors seen in Section 5.6.5. This would also enable the use of a larger array of error recovery motions when the vehicle is stuck in an error state. These maps need to also be cordoned off by hand to prevent the vehicle from driving paths that it sees as flat but we know is not advisable to drive on such as dirt sidewalks.

Testing in a CARLA simulator is a very important step which was skipped in this paper due to the simulation not being ready in time for this project. The simulation should test all the behaviours in a 3D space with speed taken into account. It can also test the active object avoidance such as humans and cars moving in the vehicle's way.

The time difference spent simulating the system versus the testing done on the is the vehicle was very high. This was mainly caused by external sources such as COVID—19 and the nature of a group project causing unpredictable delays but a lot more testing needed to occur on the vehicle to fix a lot of the current glaring issues with the project.

6.2 Future Work

The following future work is advised as an outcome of this report:

- A rigorous study into the range of local planners available. This was briefly covered by this report, but this report mainly focussed on perfecting the global planner. Thus there needs to be better testing in terms of speed, behaviour, and safety on the local planner. An exploratory and design research paper on the use of a TEB or DWA local planner is advised. The outcome of the report should determine the planner that needs to replace the current desire direction planner on the vehicle and the optimal tuning for computational time.

- The vehicle has no formal method of recording and discovering risks and mitigating risks. A proper risk registry to an industry ISO 31000 standard is highly recommended due to the complexity of the vehicle. A formal process can analyse all available risks and can help design future projects to develop risk mitigation and safety schemes for the vehicle.

- Update the motion primitives of the vehicle such that we use bicycle simplification of the vehicle. The current unicycle assumption of the vehicle isn't quite right since vehicles dual axis steering doesn't quite allow for unicycle movement thus a kinematic study into what motion primitive model best represent the bus must be undertaken.
- Testing must occur on CARLA to refine the behaviour of the vehicle in active environments since the local planner originally tested in the basic ROS stage simulation wasn't tested with actively moving object which is highly recommended if this bus should operate around people.

- Move the current navigation and SLAM system to the other computer to allow for much better computational loads followed by installation of the planned SBPL Low Resolution and TEB local planner package to be run on the vehicle.

- A psychological study on human comfort and awareness would be a great study to determine whether the current behaviour state of the vehicle would be comfortable around pedestrians or is still too foreign for people to be comfortable with it in pedestrian areas.

6.3 Conclusion

The project itself did design a Navigation Stack which worked behaviourally well in simulation and designed a navigation stack which worked on the vehicle thus it did succeed in proving that ROS motion planning can be used for complex large vehicle navigation in urban unstructured environments. Unfortunately the due to hardware limitations the perfected simulation solution wasn't able to be installed in its full form thus it couldn't be proven currently. The current state of the vehicle very basic autonomy is available, but it isn't up to the complexity and results achieved in simulation. Migration of the original software on to another computer will solve this issue and fully prove that ROS's navigation stack can be used for true navigational autonomy.

7 References

[1] EasyMile, “EasyMile EZ10 v2 Cybercar User Manual”, Toulouse, France, 2015

[2] T. Bräunl, "Localization and Navigation" in Embedded Robotics. Berlin, Germany: Springer, 2006, ch.14, pp.249-250

[3] T.Bräunl, "Localization and Navigation" in Embedded Robotics. Berlin, Germany: Springer, 2006, ch.25, pp.404.

[4] T. Churack, "Improved Road-Edge Detection and Path-Planning for an Autonomous SAE Electric Race Car," UWA, Perth, 2015.

[5] B. Yoon, M. Park and J. Kim, "UGV(Unmanned Ground Vehicle) Navigation Method using GPS and Compass," 2006 SICE-ICASE International Joint Conference, Busan, 2006, pp. 3621-3625.

[6] Hoon-Je Woo, B. Yoon, Bong-Geun Cho and J. Kim, "Research into navigation Algorithm for unmanned ground vehicle using Real Time Kinemtatic (RTK)- GPS," *2009 ICCAS-SICE*, Fukuoka, 2009, pp. 2425-2428.

[7] H. Elsayed, B. A. Abdullah and G. Aly, "Fuzzy Logic Based Collision Avoidance System for Autonomous Navigation Vehicle," 2018 13th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2018, pp. 469-474.

[8] U. Lee, S. Yoon, H. Shim, P. Vasseur and C. Demonceaux, "Local path planning in a complex environment for self-driving car," The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent, Hong Kong, 2014, pp. 445-450.

[9] Ulrich G. Leuthaeusser "Robot path planning based on variational methods", Proc. SPIE 1613, Mobile Robots VI, (14 February 1992); https://doi.org/10.1117/12.135177

[10] Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In Proceedings 1999 IEEE Interna- tional Conference on Robotics and Automation (Cat. No. 99CH36288C), volume 1, pages 341–346. IEEE.

[11] Zheng, K., 2020. ROS Navigation Tuning Guide. [online] arXiv.org. Available at: https://arxiv.org/abs/1706.09068.

[12]K. Majd, M. Razeghi-Jahromi and A. Homaifar, "Optimal Kinematic-based Trajectory Planning and Tracking Control of Autonomous Ground Vehicle Using the Variational Approach," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018, pp. 562-566, doi: 10.1109/IVS.2018.8500609.

[13] LU, D., 2020. Global_Planner - ROS Wiki. [online] Wiki.ros.org. Available at: http://wiki.ros.org/global_planner?distro=melodic.

[14] Pablo Marin-Plaza, Ahmed Hussein, David Martin, Arturo de la Escalera, "Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles", *Journal of Advanced Transportation*, vol. 2018, Article
ID 6392697, 10 pages, 2018. https://doi.org/10.1155/2018/6392697

[15] Limpert, Nicolas & Schiffer, Stefan & Ferrein, Alexander. (2015). A local planner for Ackermann-driven vehicles in ROS SBPL. 10.1109/RoboMech.2015.7359518.

M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs,
R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating
System,” in ICRA Workshop on Open Source Software, 2009
[16] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs,R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot OperatingSystem,” in ICRA Workshop on Open Source Software, 2009

[17] B. Cybulski, A. Wegierska and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, Poznań, Poland, 2019, pp. 104-111, doi: 10.1109/RoMoCo.2019.8787346.ther Global

[18] Filotheou, A., Tsardoulias, E., Dimitriou, A. et al. Quantitative and Qualitative Evaluation of ROS-Enabled Local and Global Planners in 2D Static Environments. J Intell Robot Syst 98, 567–601 (2020). https://doi.org/10.1007/s10846-019-01086-y

[19] Search-Based Planning Lab, Primer to Search-Based Motion Planning (2020). http://sbpl.net/node/50

[20] X. Zhang, M. Chen and X. Zhan, "A combined approach to single-camera-based lane detection in driverless navigation," 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, CA, 2018, pp. 1042-1046.

[21] U. Lee, S. Yoon, H. Shim, P. Vasseur and C. Demonceaux, "Local path planning in a complex environment for self-driving car," The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent, Hong Kong, 2014, pp. 445-450.

[22] Q. Luo, Y. Cao, J. Liu and A. Benslimane, "Localization and Navigation in Autonomous Driving: Threats and Countermeasures," in IEEE Wireless Communications, vol. 26, no. 4, pp. 38-45, August 2019.
[23] E. Coronel, A. Pojomovsky and F. Gaona, "Reliable navigation-path extraction system for an autonomous mobile vehicle," 2015 Tenth International Conference on Digital Information Management (ICDIM), Jeju, 2015, pp. 175-181.

[24] Varas, N., 2020. Move_Base - ROS Wiki. [online] Wiki.ros.org. Available at: <http://wiki.ros.org/move_base> [Accessed 27 March 2020].