**IMPLEMENTATION OF SLAM, NAVIGATION, OBSTACLE AVOIDANCE, AND
PATH PLANNING OF A ROBUST MOBILE ROBOT USING 2D LASER SCANNER**

by

Omid Karimpour

Bachelor of Science in Electrical Engineering, University of Shahid Beheshti, 2015

A project presented to Ryerson University
in partial fulfillment of the requirements for the degree of
Master of Engineering
in the program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

©Omid Karimpour, 2019

Author's Declaration

I hereby declare that I am the sole author of this project. This is a true copy of the project, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

IMPLEMENTATION OF SLAM, NAVIGATION, OBSTACLE AVOIDANCE, AND PATH PLANNING OF A ROBUST MOBILE ROBOT USING 2D LASER SCANNER

2019

Omid Karimpour

Master of Engineering

Electrical and Computer Engineering

Ryerson University

Over the last decade, navigation and Simultaneous Localization and Mapping (SLAM) have become key players in developing robust mobile robots. Several SLAM approaches utilizing camera, laser scan, sonar and fusion of sensors were developed and improved by a number of researchers. In this thesis, comparisons of these methods were evaluated, especially those offering low cost benefits, and low computation and memory consumption. The aim of this thesis was to select the most reliable and cost-efficient approach for indoor autonomous robotic applications. Currently, there are numerous studies that have optimized these SLAM methods; however, they still suffer from various complications such as scale drifting and excessive computation. This study performed different experiments to observe these challenges in real-world environments. A modified Pioneer robot was used to implement the selected SLAM system and furthermore, perform obstacle avoidance and path planning in indoor office environments. The results and tests show the reliable performance of Gmapping after tuning its parameter and set right configurations.

Acknowledgements

I'm thankful to many people for their continuing support of this work. Above all, I'd like to acknowledge my professor Dr. Alexander Ferworn, who always believed in my abilities and never hesitated with inspirational support. His trust made me believe in myself more than ever and gave the freedom to explore this field and pursue my passion. The collaboration and discussions throughout my time in Network-Centric Applied Research Team (NCART) Lab, paved the way for my future works. I'd like to acknowledge my friend Mohammadreza Mousaei, without his help and support I would never take this path, his uninterrupted guidance is notable during this work and my life. Last but not least, thank all lab mates.

Dedication

Dedicated to my parents who've supported me and trusted me, in academy and life.

My mother, her infinite love and support can hardly be expressed by words.

My father with spending all his life to provide us with a brighter future.

Table of Contents

ABSTRACT .	III
LIST OF FIGURES	VIII
GLOSSARY	X
SECTION 1. INTRODUCTION	1
1.1. OVERVIEW OF THE PROJECT	1
SECTION 2. LITERATURE REVIEW AND PREVIOUS WORKS	4
2.1. SIMULTANEOUS LOCALIZATION AND MAPPING	4
2.1.1. <i>Introduction to SLAM</i>	5
2.1.2. <i>Vision-Based SLAM Algorithms</i>	7
2.1.2.1. Semi-direct Visual Odometry (SVO)	8
2.1.2.2. Dense Piecewise Planar Tracking and Mapping SLAM	8
2.1.2.3. Large Scale Direct SLAM	8
2.1.2.4. Direct Sparse Odometry	9
2.1.2.5. ORB SLAM	10
2.1.2.6. Stereo Parallel Tracking and Mapping	10
2.1.3. <i>LIDAR-based SLAM Algorithms</i>	10
2.1.3.1. KartoSLAM	11
2.1.3.2. Core SLAM	12
2.1.3.3. Lago SLAM	12
2.1.3.4. Cartographer	13
2.1.3.5. HectorSLAM	13
2.1.3.6. Gmapping	14
2.2. PRE-MAPPING	14
2.3. LOOP CLOSURE	16
SECTION 3. IMPLEMENTATION TECHNIQUES	17
3.1. HARDWARE SYSTEM	18

3.2.	SOFTWARE SYSTEM	22
3.2.1.	ROS.....	23
3.2.2.	RPLIDAR Setup on ROS	25
3.2.3.	Pioneer 2DX Setup on ROS	27
3.2.4.	SLAM.....	28
3.2.5.	Localization	31
3.2.6.	Navigation Stack	32
SECTION 4.	CONDUCTED TESTS AND RESULTS.....	33
4.1.	CHALLENGES AND OBSERVATIONS.....	37
4.1.1.	Small and Narrow Obstacles and Different Surfaces	38
4.1.2.	Glass Detection Challenge	39
4.1.3.	Stairs Detection and Mapping Challenge	41
4.1.4.	Loop Closure Challenge.....	44
4.1.5.	Disaster Area's Pre-mapping Practical Objection.....	44
4.1.6.	Dynamic Environments	45
4.2.	SENSOR FUSION	45
SECTION 5.	CONCLUSION AND FUTURE WORK.....	47
5.1.	FUTURE WORK	47
APPENDICES		49
REFERENCES		59

List of Figures

Figure 1: Table and glass manual adjustment of the map prior to the localization	15
Figure 2: Pioneer 2DX physical dimensions [24]	18
Figure 3: Robot set up using pioneer 2DX, laptop and serial port [24]	19
Figure 4: Supposed to be a RPDIDAR photo	20
Figure 5: RPLIDAR dimensions [25]	21
Figure 6: Regulator DE-SWADJ	22
Figure 7: System overview	24
Figure 8: Visualization of ROS a) Performing Gmapping in real world b) Performing Gmapping using Gazebo simulation	25
Figure 9: RPLIDAR placement	26
Figure 10: 2 nd floor of engineering building's map (Large loop)	30
Figure 11: NCART lab's map (Small loop)	30
Figure 12: Initial localization process in NCART lab	31
Figure 13: Path being plan by this system in NCART Lab	32
Figure 14:: NCART map with velocity of 1 m/s	35
Figure 15: NCART map with velocity of 1.5 m/s	35
Figure 16: NCART map with velocity of 2 m/s	35
Figure 17: NCART map with velocity of 3 m/s	35
Figure 18: Loop closure test in 2 nd floor Engineering building Ryerson University (The break on top of the map shows he mismatch h)	35
Figure 19: NCART loop closure test (Small loop)	36

Figure 20: Bumpy areas were substituted with books, metal and wood plates	39
Figure 21: Glass detection test in front of the glass.....	40
Figure 22: The top part shows glass scans and how they fail to detect the glass	41
Figure 23: Stairs in engineering building.....	42
Figure 24: Mapping stairs using Gmapping as the robot is going up	43
Figure 25: Mapping stairs using Gmapping as the robot is going down	43

Glossary

<i>AMR:</i>	<i>Autonomous Mobile Robot</i>
<i>ARIA:</i>	<i>ActiveMedia Robotics Interface for Applications</i>
<i>DPP-TAM:</i>	<i>Dense Piecewise Planar Tracking & Mapping</i>
<i>DSO:</i>	<i>Direct Sparse Odometry</i>
<i>LIDAR:</i>	<i>Laser Scanner</i>
<i>LSM:</i>	<i>Large Scale Monocular</i>
<i>NCART:</i>	<i>Network Centric Applied Research Team</i>
<i>PC:</i>	<i>Personal Computer</i>
<i>PF:</i>	<i>Particle Filter</i>
<i>RBPF:</i>	<i>Rao-Blackwellized particle filter</i>
<i>RMSE:</i>	<i>Root Mean Square Error</i>
<i>ROS:</i>	<i>Robot Operating System</i>
<i>RTAB-Map:</i>	<i>Real-Time Appearance-Based Mapping</i>
<i>SKF:</i>	<i>Extended Kalman Filter</i>
<i>SLAM:</i>	<i>Simultaneous Localization & Mapping</i>
<i>SP-TAM:</i>	<i>Stereo Parallel Tracking and Mapping</i>
<i>SPA:</i>	<i>Sparse Pose Adjustment</i>
<i>SVO:</i>	<i>Semi-direct Visual Odometry</i>
<i>UAV:</i>	<i>Unmanned Aerial Vehicle</i>
<i>UGV:</i>	<i>Unmanned Ground Vehicle</i>
<i>URDF:</i>	<i>Robot Model Description</i>
<i>USAR:</i>	<i>Urban Search and Rescue</i>

Section 1. Introduction

Over the past decades, mobile robot applications have been a source of attention. There are several major areas of interests such as mapping of indoor and outdoor environments, localizing the robot in a pre-mapped environment, and performing both tasks simultaneously which is called Simultaneous Localization and Mapping (SLAM) [1]. Nowadays, there is a dramatic increase in mobile robotic research where new algorithms are being proposed to address SLAM and navigation challenges. Moreover, many investigations and experience have improved these algorithms to reduce the amount of redundant data processing resulted in more efficient computation and the ability to utilize multi-sensors.

Recent studies in the Robot Operating System (ROS), SLAM, Navigation stacks, layered Costmap, obstacle avoidance, and other optimization techniques, make it possible to process more data with limited processing power and memory. These findings improve the overall result by developing other approaches such as utilizing sensor fusion. It is important however, not to assume that these techniques will work in all cases.

1.1. Overview of the Project

The objective of this project is to determine a robust and reliable SLAM system that could perform in an indoor environment utilizing low-cost sensors and processing power. First and foremost a survey of various SLAM systems was conducted. We avoided approaches that rely on highly accurate sensors and high computation hardware as they come at a high cost. These methods often utilize more than one type of sensor. We outlined the advantages and disadvantages of popular vision and Light Imaging Detection and Ranging (LIDAR)-based algorithms and among

them, Gmapping provided the most reliable result. Gmapping SLAM algorithm has been implemented on a customized Pioneer 2dx platform to perform SLAM, obstacle avoidance, and path planning using Navigation Stack. Thereafter, we investigated several optimization methods such as Layered Costmap. This method enhanced the map by layering different features and landmarks especially in dynamic areas. This dynamic layer is unstable and causes a constant change in the robot navigation. The Layered Costmap also sets other layers such as sensor fusion, sonar and LIDAR to provide faster and reliable navigation. We dedicate part of the second section explaining the limitations and challenges of layered Costmap in different mobile robot applications.

In general, different layers of the Costmap are configured to contain selected landmarks. These layers are being established so that the algorithm could detect different areas with their corresponding features. This allows the navigation to plan out the fastest path by making adjustments such as avoiding dynamic areas. These navigation decisions are being made by a path planner; therefore, an accurate scan measurement is critical. Another navigation challenge that can occur is detecting glass in new office buildings. Low-cost laser scanners such as RPLIDAR are not able to fully recognize glass resulting in a misguided path. Currently, there is only a few studies on this complication which has been reviewed in the literature review section, particularly pre-mapping which offers a low-cost solution. Other challenges to the navigation system that are briefly outlined are freer zone likelihood, and personal space.

In the Implementation and Conducted Tests and Results sections we disclose our experiments where a detailed step-by-step implementation of SLAM, Navigation Stack, Obstacle avoidance, Path planning and other methodologies using RPLIDAR, and Pioneer mobile robot are

discussed. Some experiments on avoiding stairs, mapping stairs and practical glass detection. Thereafter, we proposed some possible solutions to resolve mentioned challenges such as stairs detection. Overall in this study, we utilized the ROS, and all configurations, transforms, and Launch files have been added to the system and discussed in detail. The implementations of this project are performed by the ROS Kinetic and C++ Programming language. RVIZ package has been used for visualization of the robot model, movement and other ROS visualizations. All simulations have been performed in Gazebo (9.0.0).

Section 2. Literature Review and Previous Works

The Robot Operating System is the most popular robotics framework used in robotic research and development around the world. ROS provides a platform which focuses on collaborative and rapid software development in the robotic field. The addition of libraries, packages, and tools to the ROS have simplified roboticists endeavors to build robust robot platforms such as autonomous mobile robot (AMR) [2].

All non-trivial mobile robots are expected to move effectively in a specified environment, localize themselves within some form of mapped representation of the environment, and would typically use the maps to avoid obstacles. Having a map such as this and being able to improve this map is essential for continuous operation in the environment. To make competent decisions about where to move, a path planner is added to the system. The path is being chosen based on the map that has been generated. As localization and mapping are extremely important to mobile robot applications, this section provides an overview of various SLAM systems.

2.1. Simultaneous Localization and Mapping

Many AMRs localize themselves with respect to an internal map. The map is devised by various means and is derived through the interpretation of sensor data. This map is never complete and is constantly being augmented and corrected as new sensor data is being measured simultaneously. This concept is a classic example of the “chicken and egg” problem, where the robot does not know where it is, so it attempts to localize itself on a map it creates or tries to build a map so it can determine its location within it.

This apparent paradox can be explained through an example. Imagine you are on vacation and parked your car to reach a tourist destination by walking. You know that you eventually have to come back to where you parked. At first, you might start collecting information such as landmarks to remind yourself later where your car is. A café, restaurant, street number and other features or landmarks are some of the parts of a map you build inside your mind and is intended to remind you of the path you have taken and would lead you back to your car. SLAM schemes use a similar approach with the expectation that the location of the robot is accurate in the map since critical mobility functions such as successful obstacle avoidance, path planning and exploration are dependent on the accuracy of SLAM.

From this point, several SLAM algorithms have been proposed, but established SLAM algorithms have been thoroughly investigated and will be discussed here. Among all these SLAM algorithms, Gmapping, Hector SLAM and cartographer are found to be more robust and accurate. The objective of all SLAM systems is to read the sensor measurements, send series of controls during a period of time to compute the pose estimation of the Robot, and map the environment simultaneously. These arguments are usually probabilistic, based on a given map, sensor measurements, controls, and the pose. The algorithm calculates the belief which is the knowledge of the robot from its state in the environment. The most general algorithm to calculate the robot's state is the Bayes filter [3].

2.1.1. Introduction to SLAM

While there is a plethora of SLAM systems that are used for indoor and outdoor mobile robot applications, this project is focused on indoor environments. Furthermore, this section will discuss the number of SLAM systems that exist in ROS. The state-of-the-art algorithms and ease

of implementation are some of ROS strengths. This section discusses the advantages and disadvantages of several SLAM systems and how it relates to the proposed system.

The proposed system is a LIDAR-based SLAM system. Further details of this selection are presented in the next section along with the technical approach. A survey of other works that used this approach will be explained here. In contrast, we also discussed visual SLAM systems and why they are unsuitable for the proposed system. We also expanded some of the experiments conducted by Maksim Filipenko and Ilya Afanasyev in some of these vision-based and LIDAR-based SLAM methods [4].

While there are a vast variety of sensors, camera and LIDAR-based approaches, this thesis examined monocular, stereo camera, and laser scanners because they are commonly used for this type of system. A technique that combines these sensor data from different sources is called sensor fusion which results in less uncertainty and more dependable data. Undoubtedly, the idea of sensor fusion was impossible up until the last decade due to processing and memory limitations. However, several algorithms and developments have contributed to making sensor fusion possible. One of these developments have utilized sensor fusion to accurately map a glass office environment without pre-mapping or adding extra adjustments. Explicit discussion in the sensor fusion utilizing LIDAR, sonar/advanced sonar [5] and cameras; tuning SLAM parameters [6]; and their effects on SLAM accuracy, have been conducted in the next section to search for an optimal solution for indoor AMR applications and their challenges.

These Approaches are discussed in the following two different sections: vision-based SLAM algorithms and LIDAR-based SLAM Algorithms including the challenges of these approaches. Overall performance of vision-based SLAM systems depends on the computation resource. The

focus of this project is on a limited hardware; therefore, visual SLAM approaches are not an optimal choice. However, we will go through these vision-based methods only to analyze their performances in different environments. A discussion on the comparison of LIDAR-based SLAM performances shows that mapping Cartographer, and Hector SLAM provide more accurate results compared to KartoSLAM, Core SLAM, and Lago SLAM algorithms. However, issues with these approaches are that Cartographer suffers from the blurry effects, Hector SLAM does not provide an accurate result when low rate laser scans are available and Gmapping increases computation and has other challenges such as depletion problem. For the purpose of this project, Gmapping is preferred over other LIDAR systems as an adaptive resampling technique. This technique has been developed by [7] to optimize Gmapping performance.

2.1.2. Vision-Based SLAM Algorithms

Vision-based SLAM algorithms are based on either monocular or stereo cameras. Monocular-based approaches require extra modules in order to perform the scale recovery. As the distance between features and the camera cannot be measured in real scale, recovery of these scales is necessary to eliminate this drift of scale. In contrast, stereo cameras achieve reliable results over monocular algorithms as they perform flawlessly without the scale recovery. However, they suffer from the same problem when the distance between the camera and feature is high. Both of these methods could be used as an additional source for pose estimation and mapping. Approaches such as Semi-Direct Visual Odometry (SVO) and Dense Piecewise Planar Tracking and Mapping (DPPTAM) suffer from losing track of robot turns when there is a lack of features in the environment. Other monocular approaches such as Large Scale Direct (LSD) SLAM, Direct Sparse Odometry (DSO) and ORB SLAM are accurate enough if the scale recovery is being used as an additional module. ORB SLAM is the most stable and robust algorithm among other visual

SLAMs. We will briefly discuss each of these SLAM approaches and point their advantages and disadvantages.

2.1.2.1. Semi-direct Visual Odometry (SVO)

SVO is a popular monocular algorithm which provides a decent accuracy and speed with limited computation in most applications. In [8, 9], this approach has been developed by analyzing the intensity of each pixel using a probabilistic mapping method. This method includes more dependable points than any other monocular approach. This dependable point has been tested by Maksim Filipenko and Ilya Afanasyev [4] in areas and environments without access to GPS. The result is a high frame-rate motion estimation which increases the accuracy and robustness of this method and therefore, makes a great addition for mapping and pose estimation.

2.1.2.2. Dense Piecewise Planar Tracking and Mapping SLAM

DDPTAM method is a direct system for pose estimation and uses a similar approach as SVO, but with more accuracy. This direct monocular SLAM algorithm was developed in [9] and resulted in an advanced algorithm containing an uncluttered output with fewer errors in a variety of conditions. There were several comparisons between this method and other monocular methods and overall, higher accuracy and lower processing time was the result of DDPTAM. It is important to mention, that this method is not robust enough and fails on tracking all robot turns.

2.1.2.3. Large Scale Direct SLAM

LSD is another monocular SLAM algorithm which was proposed by [10] and uses a similar approach to the last two methods. However, the advantage of this method is that it provides an accurate pose estimation compared to other methods. The result of this featureless monocular

algorithm is a real-time output that filters over a large number of pixelwise stereo comparisons. LSD is a direct tracking method that contemplates the depth value and it is also less susceptible to noise. Some of the novelties of this approach include its ability to detect scale-drift and incorporate noise on the estimated depth using its probabilistic approach. This system also provides a loop closure option to perform flawlessly when any drift error occurs, however, the system has problems with identifying the robot's initial pose shift and has a poor ROS integration [4].

2.1.2.4. Direct Sparse Odometry

DSO SLAM algorithm is a monocular visual odometry. The main features of this method are calibration, motion bias and shutter effect. In [11], a comparison was conducted on both feature-based and semi-supervised methods. It was found that photometric calibration results in an improvement of direct methods. Moreover, feature-based methods have a large performance bias on certain movements compared to direct methods, whereas direct and semi-direct methods are more sensitive to rolling shutter effect. Therefore, feature-based methods need a higher resolution in order to perform accurately. DSO, like many monocular SLAM systems, has a robust pose tracking, however, it suffers from the lack of a loop closure resulting in a noisy map. The dependency of this method to scale recovery is also a general criticism of this technique [4].

The next two methods are based on Stereo SLAM algorithms. The major advantages of stereo compared to monocular is eliminating the scale recovery step, using the collected databases to process them offline, and delivers an accurate state estimation [12].

2.1.2.5. ORB SLAM

The ORB is a powerful SLAM algorithm, suitable for different environments. Reliable loop closure and robust initializations are other advantages of this feature-based method. Findings from ORB SLAM [13] show that the system produces robust and real-time results because it uses a novel strategy of creating sparse point cloud as the map. This approach has an accurate pose estimation and creates a 3D sparse map [4]. Automatic initialization, bundle adjustment and depth estimation are a number of optimizations that result in a more robust approach. This method is mostly being used in indoor robotics applications and the map is a good approximation of the real world. The only disadvantage of this approach is that scale recovery is not solved which result in a weak trajectory approximation [13].

2.1.2.6. Stereo Parallel Tracking and Mapping

The Stereo Parallel Tracking and Mapping (SP-TAM) SLAM is a real-time and feature-based approach suitable for both indoor, and outdoor environments. It also includes a loop closure with a large range of dynamic areas. This method avoids the boost mapping problem in monocular methods by dividing the tasks into two parts. The first part is the tracking that deals with the camera pose and the second part is mapping that updates a submap with a lower frequency [14]. Overall this method is reliable and robust but has inaccurate localization.

2.1.3. LIDAR-based SLAM Algorithms

LIDAR measures the duration from sending a laser scan to receiving the reflection to detect the distance and objects around it. LIDARs are not only robust against interference, but also are independent from ambient lights. Some of these SLAM algorithms are more dependent to an

accurate LIDAR measurement than others. Among SLAM algorithms that we studied, Gmapping, Hector SLAM, and Cartographer are more robust and reliable than Lago, Karto and CoreSLAM. These algorithms will be outlined in detailed in the sections below.

Some of the general comparisons between these algorithms is that Lago SLAM has the highest CPU usage. Another comparison is that Karto results are more accurate than Lago SLAM and are less affected by noise in the real world. However, Karto SLAM provides a less reliable map compared to Gmapping, Hector SLAM and Cartographer. On the other hand, Core SLAM is a simple Particle Filter (PF) approach that requires more particles which lead to a more complex computation. PFs are a set of Monte Carlo algorithms used to solve the filtering problem by estimating the posterior density of the state variable based on given sensor observations [3]. Between the three most robust and reliable algorithms, Cartographer suffers from blurry effects in some cases whereas Hector SLAM only uses scan matching and does not use the odometry data which could be a disadvantage in some environments. Scan matching is a technique that finds the robot's pose by comparing its positions from two different scans. Relying on only scan matching increases the complexity of computation, and therefore the CPU load. As the industry tries to optimize the cost, keeping the CPU loads low is critical. Gmapping approach is less complex than Hector SLAM and therefore, result in less computation. It also remains a robust and accurate approach among many studies since it utilizes both scan matching and odometry to minimize the use of particles.

2.1.3.1. KartoSLAM

The Karto SLAM algorithm is a graph-based SLAM method that was developed by Karto Robotics [15]. Overall, graph-based SLAMs require a loop closure improvement method to be

more accurate on large scales. Also, they generate higher efficiency when a higher number of landmarks is being used which evidently requires higher computation and memory usage. Karto SLAM is not suitable for our project not only because adding the loop closure improvement to this method would increase the computation, but also there are other approaches such as Gmapping that would provide a better result with the same CPU load.

2.1.3.2. Core SLAM

Core SLAM is based on tiny SLAM algorithm and has been designed to have less computation compared to other approaches [16]. Core SLAM calculates the distance with a variety of sensor measurements and matches these scans with the map to find the optimized pose. Each particle is being assigned a specific weight based on the final pose decision. The lowest weight particle is the least accurate estimation and is being removed. The highest weight particle is the most accurate pose estimation and is the robot's pose. Finally, an optimization technique maps the obstacles using points around each obstacle [16]. This approach has a more complex computation and is not desirable for projects with limited resources such as this thesis.

2.1.3.3. Lago SLAM

Lago SLAM requires an initial guess of the robot's pose [17] similar to any other graph-based SLAM methods. Lago SLAM solves a local convex approximation in each step. Then, updates the graph until it reaches a minimum local cost function. This method can take advantage of any optimizer. The results of this method, even after using optimizers, are not as accurate as Karto SLAM and it is more affected by noise than other systems [4].

2.1.3.4. Cartographer

The cartographer approach focuses on the backpack mapping platform, enhancing the loop closure in real time, as well as, other improvements such as branch and bound [18]. This system was developed recently in 2016 and it represents the closest fitting from the 2D LIDAR map to the real world. The tests from [19] show that the difference between trajectories of Cartographer and Hector SLAM are less than 3cm in the root mean square error (RMSE) which means they both demonstrate a similar result. Cartographer also uses pose estimation continuously and includes real-time loop closure. The only downside of Cartographer is the blurry effect which was evaluated by Anton et al. [20] based on analyzing the following metrics: number of corners in a map, enclosed areas, and the proportion of occupied and free cells.

2.1.3.5. HectorSLAM

Hector SLAM is a flexible and scalable SLAM system, superior in its class, which provides pose estimation with full 6 degrees of freedom [19]. It utilizes limited computation resources resulting in a reduction in both the cost and weight of the robot. Hector SLAM uses an occupancy grid map in order to represent an arbitrary environment, the LIDAR scans are being transformed into a local frame to be stable in event of scanning across 6 degrees of freedom motion. The scans are being converted to the point cloud of LIDAR scan endpoints. These point clouds need to be processed to perform down-sampling. The endpoints are being sorted within the threshold of the intended scan to limit the processing and memory [19]. Compared to Gmapping, this approach has been successful in small case scenarios with unstructured environments such as most disaster areas, however, it is not suitable for large loop closures as it only relies on scan matching. Therefore, Hector mapping is a desirable solution for applications that can't rely on accurate

odometry. As long as high rate scans are being used, Hector SLAM is a reliable and accurate approach.

2.1.3.6. Gmapping

Rao-Blackwellized particle filter (RBPF) based methods effectively proposed a solution to the SLAM problem by using PF and creating a grid-based map [7]. Gmapping, a RBPT-based SLAM approach, is an efficient indoor AMR algorithm that is also being utilized particularly in the Search and Rescue (SAR) field. In this method, landmarks are constantly being captured in order to estimate the robot's pose. Low efficiency in capturing landmarks could result in poor mapping and high efficiency landmark capturing results in lags due to low processing speed. In the study [6], the researchers tuned Gmapping parameters to optimize the number of landmarks with respect to the processing power. However, one of the disadvantages of this algorithm is that a high number of particles are required to achieve an accurate result. Gmapping uses a set of particles to approximate the probability distortion in order to solve the SLAM [7]. An improvement approach called adaptive resampling technique [21], was created to deal with these high volumes of particles. Considering both present and past movements of the robot, the author came up with a distribution accuracy computation to limit the required particles.

2.2. Pre-mapping

In USAR applications, having the floor map, stairs and exits of a building prior to the disaster could help the rescue team outline the areas with more probability of possible survivors. However, glass doors/windows and stairs pose great challenges and may require manual adjustments to modify these areas on the map. This process of mapping is called pre-mapping [22], which serves an important role in better understanding a disaster area. In [22], LIDAR was used to outline this

idea, although they faced several problems such as floor plan distortion by a catastrophic event. We will review some of these challenges in more detail in “Problem and Solution” section.

It is important to note that current SLAM methods present some shortages such as failing to detect slim objects, glasses, and obstacles that are out of the sensor’s range. A low-cost solution to this issue is to manually adjust the map. The evidence in support of this could be found in some of the tests that we performed on mapping the environment prior to localization shown in Figure 1. These tests result in a better understanding of certain obstacles which are not easily detectable by most sensors and SLAM algorithms.

Figure 1: Table and glass manual adjustment of the map prior to the localization

In one particular study [22], a classroom with narrow tables has been chosen to test the Gmapping. The first test involved an ideal classroom with wider objects and walls that were completely covered by cardboard. This allowed them to evaluate the robot’s Gmapping

performance in an ideal environment. For the second test, a similar classroom with narrower spaces and smaller objects resulted in a lack of data as expected. A third and final test was performed in a dynamic classroom with elements from the first two tests. A major evaluation was conducted after all experiments with respect to different SLAM methods in an effort to find the optimal SLAM method. The results were compiled in [22] and their work has proposed pre-mapping as a reliable solution for lack of reliable results in such environments.

2.3. Loop Closure

For each SLAM algorithm test, mapping a large loop is an essential part of the process. Without a loop closure optimization, most of these approaches fail to close this loop because of unreliable sensor measurements. In [23], an improved loop closure method was presented and implemented by utilizing a low-cost LIDAR similar to the scanner used in this project. Gmapping with RPLIDAR does not provide an optimal result in small indoor environments since many landmarks need to be captured. Since RPLIDAR measurements are reliable on large scales but not accurate enough on small scales, the loop closure improvement is still an attractive idea in both small and large loop corrections. Loop closure consists of two areas of improvement which are loop correction and loop detection. There are several optimizations in the loop correction, however, loop detection still remains a major challenge. Gmapping registers several different laser measurements including when a robot moves to a certain distance, change direction or in specific time periods, which are all hard to control. To control these registrations, a study developed a Kalman filter-based loop closure improvement method was developed to register the laser scans only when the robot moves to a specific distance [23]. This method assumes that the robot detects the loop accurately, and when the robot revisits the starting point, the loop correction procedure starts and changes the map to result in the closure of the loop.

Section 3. Implementation Techniques

In this section, a detailed explanation of implementation techniques and the objective behind hardware and software used will be outlined. The types of hardware components reviewed are the Pioneer 2DX, motor controller, RPLIDAR, batteries, host computer and connections. An overview of software will include the ROS, ARIA client program, RPILIDAR, Gazebo simulation, SLAM system, costmaps, obstacle avoidance and path planning.

In this project, Gmapping has been utilized and tested in variety of environments such as offices and rough surface environments. The tests and experiments showed that Gmapping is robust and reliable in such environments. Evidence that supports this finding can be found in the “Conducted Test and Results” section. Since all tests are being performed in the ROS platform, the first step is to interact with Pioneer 2DX using a host computer and Serial-USB port. A connection is established between ARIA and ROS using ROSARIA package and then the system is tested by publishing a velocity command through a ROS topic. Secondly, establish the connection between Rplidar and host computer using Micro USB-USB port. In this host computer the RPLIDAR ROS package and RVIZ simulator is being used to send the scans’ measurements to RVIZ. Before testing the system in the real world, simulations of the EKF and Gmapping SLAMs were run to compare the results and establish a baseline by referencing this map as an ideal map. This system was then tested in a real-world environment by using both small and large loop closure to compare the results. This test was conducted several times at different velocities and scanned frequencies to find the optimal set up for our environment. After generating a map with an optimal set up, it was saved and used in the next step so that the robot can localize itself

on this map and perform obstacle avoidance and path planning. In this section, hardware and software components and their implementations are reviewed.

3.1. Hardware System

Pioneer 2DX: This mobile robot has been chosen as the base because the dimensions were ideal compared to other available platforms. In consideration that a host computer has been used in this project, there was no minimum requirement of computation component. Also, the weight of this robot was decreased by applying some modifications which will be discussed in this section. The client-server architecture for this robot was originally developed by Kurt Konolige, Ph.D. Pioneer 2 DX. An ActiveMedia's robot base has a three-wheel mobile platform including two powered wheels and one castor wheel which allows it to freely rotate. To run this robot, it is essential to either use the onboard microcontroller or host computer with a client server software. ARIA client server was selected because ROS packages for this client server is available and it was suggested by the manufacturer. Physical dimensions of this robot are shown in Figure 4.

Figure 2: Pioneer 2DX physical dimensions [24]

Several expansion power and I/O ports are available for additional sensors and accessories. RS-232 serial port, digital I/O, A/D ports and other controllers are accessible for clients to communicate with this base and made this robot an ideal choice for our project. Moreover, the motor-power board supplies both 12 and 5 VDC for external accessories. The Pioneer 2DX microcontroller has a 20 MHz Siemens 88C166 with 32k ROM and dynamic RAM, and other I/O ports. A serial port is available on the console to connect this microcontroller to the client computer (Host) for data exchange [24].

Figure 3: Robot set up using pioneer 2DX, laptop and serial port [24]

The first step before using this base is performing a self-test by pressing the MOTORS button twice after start-up. The self-test consists of two tests which are Motors Test and Sonars test. Motors test start by engaging the left and right drive wheel separately by forward and backward

movement. Lastly, it tests each sonar sensor range and connection separately. To confirm the data, distance measurements were retrieved. This self-test was performed to ensure the drivers, motor-power board and connections were working properly.

Figure 4: Supposed to be a RPDIDAR photo

RPLIDAR: The RPLIDAR A1M8 was used in this project as this sensor has a reliable scan measurement compared to other LIDARs with similar price range. Other models such as RPLIDAR A3M1 has a more reliable scan but is susceptible to noise. However, as the industry tries to keep the cost low. A1M8 provides a comparable result in our test environments to its costly competitors. To process the distance and angle between object and RPLIDAR, the object is

sampled by reflecting a laser signal from the object. This distance is being captured more than 2000 times per second. In terms of hardware, a LIDAR typically consists of a range scanner system and a motor system. These two systems are powered separately and that's why a regulator has been used to power the RPLIDAR motor system. This 360° scanner uses TTL serial port to communicate with the server and power the range scanner. Figure 4 shows the regulator DE-SWADJ that was used to power the RPLIDAR's motor system. The dimensions of this sensor are shown in Figure 5 [25].

Figure 5: RPLIDAR dimensions [25]

Regulator: As mentioned a DE-SWADJ regulator has been used to power the RPLIDAR motor separately. This regulator plays an important role as it allows the ability to adjust the speed of the RPLIDAR's motor and therefore scans using this module.

Figure 6: Regulator DE-SWADJ

Power Supply: A LiPo battery replaced the old batteries of this robot to lower the weight from 9KG to 7.5KG. This battery is a DXF Li-polymer with 32SP cells, 11.1V Voltage, Capacity of 7000 mAh, Discharge of 35C and weight of 595g.

3.2. Software System

The types of software discussed in this section are the ROS configuration full software overview and configuring Pioneer and RPLIDAR with ROS. Detailed instructions of the SLAM, navigation, Costmap, obstacle avoidance and other implementations will be thoroughly explained.

3.2.1. ROS

ROS [15] is a Linux based software framework that uses packages, nodes, topics, messages and services to operate a robot. A *node* receives sensor measurements and passes it to other nodes in order to process data. The information that moves from node to node is called a *message*. These messages travel through a portal called a *topic*. A *Publisher* is publishing data to this topic and *subscriber* is subscribing to this topic to receive these messages. The *package* consists of all related nodes and is essential for building a control system for any robot through ROS.

In this thesis, ROS Kinetic was installed and the ROS environment was configured. To create, build, and modify Catkin packages, a directory called Workspace was used. The ROS workspace was created using the commands in A.2.

The command in A.3 was added to the *bashrc* file so that it could automatically occur every time the terminal is opened as only one catkin workspace was created for this thesis.

Figure 7: System overview

The Figure 7 shows all connections of the ROS systems including all transforms. In the figure 8, one could find the visualization of our system computation graph in ROS which consist of the robot, simulation, LIDAR and other components.

Figure 8: Visualization of ROS a) Performing Gmapping in real world b) Performing Gmapping using Gazebo simulation

3.2.2. RPLIDAR Setup on ROS

RPLIDAR A1M8 is a 360° 2D LIDAR that was used in this thesis. This LIDAR has been placed centred on top of the robot with a height of 245 mm as shown in Figure 9. A plastic plate was used to fix this laser scanner on top of the chassis. Maximum scanning frequency of this LIDAR with 360 points samples is 5.5 Hz. During testing, the nominal speed of this laser scanner was maintained at 35-40% of its max speed which is the optimal speed for this environment. A USB connection was used to transfer the data to the LIDAR and a power supply of 5-10V was required for the regulator to supply power for RPLIDAR motor's system. It is important to keep in mind that changing the value of this regulator affects our scanning speed. For any further information, please refer to the RPLIDAR A1M8 data sheet [25].

Important factors need to be considered when setting up the LIDAR on the robot. If the LIDAR is set too low, it would affect the mapping as well as localization since it is unable to capture enough landmarks. On the other hand, setting this laser scanner too high could result in not detecting the obstacles that the robot is supposed to avoid. After conducting different height tests, we placed the RPLIDAR at the centre of the robot with a height of 245 mm. This gave an optimal navigation by easily identifying most obstacles in our test environment (2nd floor of Ryerson Engineering Building)

Figure 9: RPLIDAR placement

Powering RPLIDAR with only an USB is not sufficient for long distances and therefore, this laser scanner needs a separate port for its power consumption. A regulator was added to not only control the RPLIDAR motor speed, but also was a separate power source for this motor

system. The motor speed directly affected the map as the number of landmarks captured were changing respectfully.

3.2.3. Pioneer 2DX Setup on ROS

In order to setup the Pioneer 2DX, an interface that connects to the ActiveMedia controller was required. This interface is called ActiveMedia Robotics Interface (ARIA) and is an open source software based on C++. Any low-level details such as commands and serial communication could be accessible through ARIA. The ARIA package was installed on the Linux Ubuntu 16.04, using the commands in A.1. At this point, ARIA and ROS are installed and ROSARIA package is required to communicate between the ARIA library and ROS. This platform automatically updates the ARIA version and provides both control and position estimations by using ROS nodes to connect to this library.

The ROSARIA package has been retrieved to our catkin workspace and the package was built using commands in A.5. This process had only needed to be completed once. However, *roscore* must always be running so that the ROS nodes can communicate in their network. At last, the ROS nodes are implemented as shown in Appendix A.6. The robot was connected to a host via serial to a USB port. In such case, the command was modified in order to connect via USB0. After getting a connection to the robot using ROSARIA, the pose estimation was echoed as the ARIA publishes the odometry data. This information is being published in `RosAria/pose` topic every 100 ms. This frequency is by default and is adjustable. Two other parameters that refer to the port and frame of the robot are *base_link_frame*, *port*. They are set to `dev/ttyUSB0` and `base_link` by default.

In the first test, different velocities were observed on the robot in order to test the velocity commands in A.9. This velocity state is being published to the ARIA using `/RosAria/cmd_vel`. This command needs to be published every 100 ms in order to maintain this velocity. If the robot does not receive the `cmd_vel` message for 600 ms, timeout will occur, and the robot stops for safety reasons. In order to eliminate this, the command needs to be modified to publish a more frequent velocity greater than 600 ms. Overall, all commands are shown in the Appendix including the laser scanner in RVIZ, the connection of the `base_link` to `laser_frame` and launching the pioneer 2DX robot model description (URDF).

3.2.4. SLAM

SLAM algorithms are developing rapidly and high demands for smaller and lighter robots have challenged researchers to simplify their methods and decrease the computation parts. The rapid development in indoor 2D SLAM applications has been heavily studied, which led to several improvements such as [7] by Grisetti et al. These approaches improved the RBPT by considering both the recent observation of the robot and updating its motion model. The number of particles decreased using the scan matching technique (lower computation) after reducing the resampling step frequency.

Gmapping SLAM algorithm has been used in this project which is one of the most studied algorithms with several optimization methods. A thorough implementation and evaluation study of Gmapping was conducted by Abdelrasoul et al. which proposed an optimization technique that involved tuning its parameters [6]. To lower computation, they modified the number of particles and resampled the threshold. This created a reliable map with less CPU load and memory requirements. Our work supports these findings and will be further discussed in the “Conducted

Tests and Results” section where implementing this method resulted in low memory usage. This involved conducting different speeds of the robot using Gmapping SLAM using low-cost 2D LIDAR which resulted in different maps with different speeds.

In this test, the scan measurements from RPLIDAR and odometry of the Pioneer are used by SLAM Gmapping to create a map using the Rao-Blackwellized particle filter base [Gmapping] approach. Firstly, the test was conducted using Gazebo and RVIZ simulation, and thereafter, the robot was tested in a small and large loop in a real-world environment. After the Gmapping process, the map is saved into PGM and YAML files. These two files are respectfully an image of the map (PGM) and a description of this map (YAML). This saved map is used later on to navigate and localize the robot. The instructions for the simulation and practical tests are found in the Appendix A.8, A.10 and A.11. Figure 11 and 12 also shows the final map of the NCART lab and a loop closure of the 2nd floor, Engineering building, Ryerson University. The visualization of the system computation graph of both simulation and real tests are shown in Figure 11 and 12.

Figure 10: 2nd floor of engineering building's map (Large loop)

Figure 11: NCART lab's map (Small loop)

3.2.5. Localization

A localization algorithm uses odometry data, LIDAR scans, and a saved map to visualize the map and location of the robot. To deal with different states of localization, launch files are written and configured. Launch and configurations files are critical for the robot localization and can be found in Appendix A.13 and A.14. During localization, the map is retrieved from the saved map using a Map Server. Thereafter, AMCL is used to localize the robot in the map which also requires an initial guess by the user. In Figure 13, the location of the robot is identified and shown. As the robot moves, the algorithm captures more landmarks which increases the certainty of localization.

Figure 12: Initial localization process in NCART lab

3.2.6. Navigation Stack

After mapping and localizing the robot, navigation is the main objective of an AMR. A move base package from the navigation stack [26] was used where this package provides a velocity command and uses localization data to navigate the robot in the environment. The details of this approach are mentioned in [26]. RVIZ tools were then configured and provided a destination for the robot to plan a safe path, accordingly. Two launch files were set up to launch all necessary transform connections such as roscore, ROSARIA, RPLIDAR, map server and RVIZ. The YAML file was used and saved from the SLAM section. For further details, the commands, configurations and launch files are found in Appendix A.13 and A.14. Figure 14 shows the planned path by this system after manually setting a goal for the robot to reach by avoiding obstacles on its way.

Figure 13: Path being plan by this system in NCART Lab

Section 4. Conducted Tests and Results

The tests in this section focus on providing an optimized robot set up and reveal the challenges faced and potential solutions. The goal is to simplify some of these challenges by creating a more reliable map and pose estimation. Some of these findings address the challenges faced by past studies and gives future studies the foundation to further explore in this field. The “Challenges and Observation” section offers an in-depth discussion on results of these tests.

Robot speed’s modification: Mapping the environment is one of the main tasks which is performed by capturing a sufficient number of landmarks from the environment. One factor that could affect this process is the speed of the robot. To evaluate the effects speed has on mapping, an experiment was conducted in the NCART lab at Ryerson University where changes in the velocity of the robot were evaluated and SLAM was performed with each of these speeds. Additionally, this experiment involved loop closure, analyzing the SLAM accuracy and tuning parameters. The robot speed and direction were controlled using the command line in Appendix A.9. We started off with 1 m/s and increased this speed by 0.5m/s steps to reach 3 m/s. Figures 15-18 shows that loop closure was completed on all scenarios. It was expected that there would be a less reliable map when the speed was increased. However, the difference between these maps created by various speeds were not significantly noticeable and it was concluded that the performance of Gmapping is independent to various speeds of Pioneer 2DX.

Figure 14:: NCART map with velocity of 1 m/s

Figure 15: NCART map with velocity of 1.5 m/s

Figure 16: NCART map with velocity of 2 m/s

Figure 17: NCART map with velocity of 3 m/s

Altering the LIDAR heights: Setting the LIDAR sensor at an appropriate height is critical as it changes the range of obstacles and landmarks that is being captured. This height could be set differently depending on the environment and its landmarks. For the purpose of this thesis, most of the obstacles the robot needs to avoid on the second floor of the Ryerson Engineering building are either not dependent to height such as doors and walls, or has obstacles that are the same height as the robot. Obstacles such as tables are not necessarily an obstacle for the robot if it can find a better path using the space under the table. Therefore, the LIDAR was installed at the center and top of the Pioneer with height of 245 mm which was shown in Figure 9. This setup allows the

robot to easily navigate the lab and office areas including passing under the table, if there is a possible path.

Long and short loop closure test: Firstly, the loop closure has been tested in the NCART lab, which is a small area compared to the Ryerson Engineering building 2nd floor. Thereafter, the robot was tested in a larger loop on the 2nd floor. During this test, it was discovered that the Gmapping could fail to detect this loop. In this case, the final map is not reliable as the loop correction depends on the loop detection to happen. We hope to optimize the loop closure process of Gmapping by manually adjusting the map in our future work.

Figure 18: Loop closure test in 2nd floor Engineering building Ryerson University (The break on top of the map shows the drift)

Figure 19: NCART loop closure test (Small loop)

Stairs mapping: A robust mobile robot should be able to autonomously navigate through a building and move to other floors either using elevator or stairs. In case of using the stairs, the robot should be able to detect these stairs. To evaluate stair navigation, LIDAR was used for mapping and these results are discussed in the “Challenges and Observation” section.

Destroyed environment vs flat surface: During navigation, there may be some cases where an AMR is required to navigate through different types of surfaces. These surfaces could be a simulation of toys in a house, stack of papers on the ground of an office floor, or maybe a damaged area after an explosion. As one can see, different applications are required for the robot to move in various situations. We tested some of these less flat areas in our application. We extended this section in the “challenges and observations” subsection.

RPLIDAR speed's modification: Not only does the speed and height of the robot could influence the scans, but also RPLIDAR's frequency directly affects the number of landmarks that

are being captured at a given period of time. Therefore, we dedicated a section on the overview of RPLIDAR power and nominal speed.

The implementations of this project were performed by utilizing Robot Operating System (ROS Kinetic) and C++ Programming language. RVIZ package has been used for visualization of the robot model, movement and other ROS visualizations. All simulations of have been performed in Gazebo (9.0.0).

4.1. Challenges and Observations

An AMR could face several challenges such as not detecting certain obstacles (tables or glasses) due to the height of the sensors or their capabilities. There were a few optimizations that aimed to solve these challenges. Particularly, AMR applications have been perfected in the past decade. Being able to process the data with less computation and memory requirements is one among many optimizations that are possible. Studies have advanced these approaches and have paved the way to achieve more accurate results in areas such as localization, mapping, and path planning. However, problems still remain with these approaches. In practice, mapping an indoor environment with glass walls and stairs is one among many. The goal of this thesis is to investigate the challenges during the mapping of glass environments and based on the findings, provide potential solutions. One of the many solutions to this challenge had utilized the advantages of sensor fusion and has been outlined in the introduction of sensor fusion. During the investigation of real-world challenges, another critical challenge especially in the exploration of real buildings is detecting and mapping stairs. This particular situation has brought us to investigate several approaches. For instance, in uneven and unpredictable terrain, locating and mapping uneven terrain is problematic and extremely important as seconds could change the lives of people in planning a

path to a trapped victim in a damaged building. The rescue team would need to be highly accurate on their calculations beforehand. Supporting evidence on this matter can be found in situations where the rescue team does not have access to a floorplan. Therefore, an AMR may be required to map the uneven terrain and the rest of the environment. However, Robots utilizing 2D LIDAR are not capable of fully mapping stairs and other similar features. Altogether, AMR applications suffer from limitless number of challenges which requires many optimizations and developments; however, only some significant challenges have been outlined in this section.

4.1.1. Small and Narrow Obstacles and Different Surfaces

After conducting tests at the NCART lab, small and narrow obstacles were mainly observed such as narrow tables and chairs. These types of obstacles are not detectable by the LIDAR and result in a less accurate map. During the navigation process, the robot uses the map to plan its path through these tables which evidently resulted in the robot crashing. In addition to these narrow obstacles, small objects that are not detectable by the sensors create a bumpy surface. To overcome this issue, these bumpy areas were substituted with books, metal and wood plates as shown in Figure 21. In this type of situation, the robot either has to detect these obstacles and change its path, or pass these gravel-like surfaces which requires different robot bases. Changing the base of the robot is not only costly, but also not effective enough as it creates other challenges such as requiring wider paths to choose from. Subsequently, detecting these obstacles become essential for the navigation process. It would be beneficial to use fusion sensors such as camera and laser scanner as it is a cost-effective approach and will be discussed more in the “Future Works” section.

Figure 20: Bumpy areas were substituted with books, metal and wood plates

4.1.2. Glass Detection Challenge

Glass detection is one of the well-known challenges that hasn't been extensively studied by researches until recently. However, there have been several attempts as mentioned to solve this challenge. Despite some expensive sensors that are able to detect glass, other approaches use at least two sensors, which require more processing power and memory. In this thesis, several tests were conducted in glass areas to investigate this problem further. These tests involved using a RPLIDAR that is not able to detect glass surfaces due to the reflection beam is either coming back from the wall behind the glass or no reflection is being detected at all. Sensor fusion could be a solution to this issue according to studies of T. Chang et al. in [27], which used both advanced sonar and LIDAR. Despite their high computation, these methods certainly provide a solution

model to this current high-profile challenge. Sonar sensors are not the best choice in terms of mapping as the sensor measurements are not reliable and accurate, however they are being used as an additional sensor since they could detect glass surface. Using LIDAR and sonar together not only could be more accurate, but also is able to detect these surfaces. When sonar sensor measurements provide a significant difference from the LIDAR measurements, the surface is most likely a glass because LIDAR is not detecting it. Figure 21 shows the LIDAR scan measurements against glass surface. We will discuss these methods further in depth in the “Fusion Sensors” section.

Figure 21: Glass detection test in front of the glass

Figure 22: The top part shows glass scans and how they fail to detect the glass

4.1.3. Stairs Detection and Mapping Challenge

One of the main challenges in robot navigation is stair detection and mapping. An advantage to having these maps is that it could be helpful for a SAR team to use after incidents such as bombing and certain tasks that require a immediate response in limited time manner. The number and height of stairs could be captured and mapped by using either Unmanned Ground Vehicle (UGV) or Unmanned Aerial Vehicle (UAV). In this section, we tried to show how this attempt failed for 2D laser scanners going up the stairs and what are the next steps to solve this challenge utilizing low-cost sensors.

Figure number 24 and 25 are demonstrating the stairs maps which has been gathered by Gmapping and RPLIDAR A1M8. Figure 24 shows the map after the robot climbed up the stairs

and Figure 25 illustrates the map after the robot climbed down the stairs. Evidently, these maps are not practicable as the scan measurements are stacked on top of each other to create a meaningless map. Based on these tests and mentioned challenges, 3D LIDAR is preferable for such applications. Also, separate Costmap layer could be assigned to maps of the stairs in order to improve the behaviour of the robot.

To provide a more stable platform for the robot to go up and down the stairs, the robot was centered on top of a wooden board. Certainly, an alternative platform such as a tracked robot is more practical for future works.

Figure 23: Stairs in engineering building

Figure 24: Mapping stairs using Gmapping as the robot is going up

Figure 25: Mapping stairs using Gmapping as the robot is going down

4.1.4. Loop Closure Challenge

Several studies have attempted to optimize the loop closure in a number of approaches such as Gmapping. Some of these studies have led to the development of new loop closure methods and others have added optimizations on current approaches. However, most of these studies only focused on the loop correction and relied on an accurate loop detection. In this thesis, mismatches on the loop detection were observed when the robot was required to close a longer loop. It was the Gmapping loop detection that failed to detect the loop in a long loop in 2nd floor Ryerson Engineering building. This detection is more challenging when there are similar features and landmarks in environments such as office and dynamic areas.

4.1.5. Disaster Area's Pre-mapping Practical Objection

Pre-mapping an indoor office or public environment are necessary, in some cases. One particular pre-mapping approach that is common among studies is mapping these environments for emergency rescue teams to use in future natural disasters such as an earthquake. This approach is called Disaster Areas Pre-mapping and has found to be helpful for the USAR team to approximate the location of survivors [22]. It is critical to emphasize that currently, the number of places which have been mapped are few which make this approach more theoretical than practical. There is also the problem of the chaotic effects of forces capable of producing a disaster. For example, a severe earthquake or bomb blast might cause of multi-story structure to collapse and deform in ways that make any pre-mapping irrelevant. Locating the critical part of a building such as stairs, and hallways is almost impossible in the case of a severe destruction. Disaster Area's Pre-mapping still remains a popular approach among current studies and we hope to provide recommendations that addresses this issue based on improved and more complete simulations.

4.1.6. Dynamic Environments

AMRs face dynamic areas on a day-to-day basis as most applications require robots to robustly navigate in such environments like in houses or offices. In the literature review section, we discussed how indoor navigation and SLAM demand a robust navigation in dynamic areas. It is also noteworthy to mention that the boundary of each person is their privacy and should be set so that the robot does not get too close to a person. To address this issue, these boundaries need to be adaptable as the environment changes. In any case, detecting a dynamic object from a static obstacle is critical to meet these expectations. These dynamic areas could be saved in a separate Costmap layer to simplify robot's navigation. This argument has been supported by evidence from Layered Costmap for Context-Sensitive Navigation in [28]. In this study, the robot could avoid driving too close to people by navigating through paths with less dynamic areas. It was suggested that it is more reliable to choose a path with less hazard to lower the chance of collision by assigning a Costmap to caution zones. It is our intent to implement and improve this layer in our future work.

4.2. Sensor Fusion

Having considered sensor fusion, it is also reasonable to look at limitations of single-sensor-based applications. Main sensors that are being used in indoor AMR applications are LIDAR, camera and sonar. As we already compared vision-based and LIDAR-based advantages and disadvantages, we will briefly go through sonar and LIDAR sensors and sensor fusion using these two sensors.

Sonar Sensor: This sensor is being used in a number of indoor robotic applications and is one of the most useful and cost-efficient sensors compared to other methods. This sensor is able

to detect any transparent surface such as glass, which is commonly used in most offices and indoor environments. However, high range of errors and the low-resolution readings of this sensor makes it unreliable compared to other alternatives such as LIDAR and camera.

LIDAR Sensor: Laser scans can provide a high resolution and highly accurate detection. This accurate scan data is the main reason LIDAR is dramatically being used in indoor mapping, localization and navigation. However, transparent surfaces such as glass are not fully detectable by LIDAR. Therefore, it is not suggested to use these sensors without pre-mapping and further map adjustment.

These two sensors can complement each other where sonar results are unreliable but able to detect transparent areas. LIDAR scans are reliable, despite not being able to detect transparent areas. To compensate for this lack of detection, [27] developed an algorithm using both sensors. This algorithm compares both measurements of sonar and LIDAR, and if there is a significant difference between these measurements, the algorithm assumes it detects a glass object if LIDAR did not detect an obstacle, but sonar did. Although, significant errors in sonar sensors, especially in edge situations, could dramatically affect the accuracy of this solution [27]. Using sensor fusion in our future work could eliminate the number of failures that we faced in our tests and could potentially result in a more accurate navigation in such indoor environments.

Section 5. Conclusion and Future Work

In this thesis, the advantages and disadvantages of different LIDAR and vision-based SLAM systems were investigated and compared. The main conclusion derived was that vision-based SLAM approaches suffered from scale drifting, lost track of the robot and relied on a high computation in comparison to LIDAR-based approaches. However, LIDAR-based SLAMs are preferable as it met our requirement of limited hardware for this project. Among these LIDAR-based approaches, Gmapping provides a reliable and cost-efficient result. This finding was based on the modified mobile robot used to evaluate Gmapping performance in a typical office environment. Different experiments were conducted by launching the robot in various closed-loop trajectories. Simulation of a dynamic office environment with a bumpy area, and glass detection areas are some of these experiments that required manual map adjustments. Sensor measurements such as LIDAR were collected to perform Gmapping SLAM, and AMCL Navigation. This study shows the reliability of Gmapping in indoor office environments.

5.1. Future Work

In this thesis we explored several different techniques for exploring closed loop navigation and traversal. We examined solutions for dealing with glass (windows) in offices and stairs. The main challenges encountered during testing were discussed and gave us insights to the shortcomings of dealing with static real-world environments. We assume that any AMR should be able to navigate in any typical office environment that generally consists of stairs, glass walls, bumpy areas and large closed loops. Based on our findings, we suggest the manual adjustment of obstacles in the map as most low-cost sensors are not capable of accurately detecting some of these obstacles, dynamic areas, and bumpy areas. However, the true definition of indoor autonomous

mobile robot requires a robot to autonomously navigate in any indoor environment without human interaction. As this is beyond the scope of this thesis, we hope more realistic solutions will be developed over time. Our work suggests that this will inevitably involve sensor fusion and areas additional forms of maps used for specific purposes (unstructured area maps for example).

Appendices

A.1. Installations and setups

```
sudo dpkg -i libaria_2.9.1+ubuntu16_i368.deb
```

A.2. ROS workspace

```
mkdir -p ~/catkin_ws/src  
  
cd ~/catkin_ws/src  
  
catkin_init_workspace  
  
cd ~/catkin_ws  
  
catkin_make
```

A.3. bashrc modification

```
. ~/catkin_ws/devel/setup.bash
```

A.4. Launching RPLIDAR

```
roscore  
  
roslaunch rplidar_ros view_rplidar.launch
```

A.5. ROSARIA

```
cd ~/catkin_ws/src  
  
git clone https://github.com/amor-ros-pkg/rosaria.git  
  
make -j4  
  
catkin_make
```

```
roscore
```

A.6. Launch ARIA using ROSARIA

```
roslaunch rosaria RosAria _port:=/dev/ttyUSB1
```

A.7. Echo the pose using ARIA library

```
rostopic echo /RosAria/pose
```

A.8. gmapping Gazebo simulation

```
Roscore

roslaunch p3dx_gazebo gazebo.launch

roslaunch p3dx_description rviz.launch

roslaunch Rplidar_ros view Rplidar.launch

roslaunch gmapping slam_gmapping scan:=/p3dx/laser/scan

roslaunch tf static_transform_publisher 0 0 0.1 0 0 0 odom base_link 100

rqt_graph

#different terminal

Rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

And add a display with a map to the RVIZ, then set it to the topic /map.

A.9. Publishing Velocity through ROSARIA

```
rostopic pub /RosAria/cmd_vel geometry_msgs/Twist -r 3 -- '[0.0, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```

A.10. Gmapping using RPLIDAR

```
roscore

roslaunch rplidar_ros view_rplidar.launch (roslaunch rplidar_ros rplidarNode)

rostopic echo scan

dmesg | grep tty

roslaunch rosaria RosAria _port:=/dev/ttyUSB1

roslaunch p3dx_description rviz.launch

roslaunch tf static_transform_publisher 0 0 0.1 0 0 0 base_link laser_frame 100
```

The code below was used to move the robot forward

```
rostopic pub /RosAria/cmd_vel geometry_msgs/Twist -r 3 -- '[0.1, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```

A.11. Saving the map after SLAM

```
Roscore

Rosrun rosaria RosAria port:/dev/ttyUSB0

Transforms?

Rosrun mapping slam gmapping

Rosrun map_server map_saver -f mymap
```

A.12. Localization

```
roslaunch amcl amcl diff.launch

roslaunch map_server map_server mymap.yaml
```

```
roslaunch rviz rviz
```

A.13. Move Base Launch File

```
<launch>

<master auto="start"/>

<!-- Run the map server -->

    <node name="map_server" pkg="map_server" type="map_server" args="$(find
my_map_package)/my_map4.pgm 4000 X 4000"/>

<!-- Run AMCL -->

    <include file="$(find amcl)/examples/amcl_omni.launch" />

    <node pkg="move_base" type="move_base" respawn="false" name="move_base
" output="screen">

        <remap from="/odom" to="/pose" />

        <remap from="cmd_vel" to="/RosAria/cmd_vel"/>

        <rosparam file="$(find p3dx_2dnav)/costmap_common_params.yaml" command
="load" ns="global_costmap" />

        <rosparam file="$(find p3dx_2dnav)/costmap_common_params.yaml" command
="load" ns="local_costmap" />

        <rosparam file="$(find p3dx_2dnav)/local_costmap_params.yaml" command=
"load" />

        <rosparam file="$(find p3dx_2dnav)/global_costmap_params.yaml" command
="load" />

        <rosparam file="$(find p3dx_2dnav)/base_local_planner_params.yaml" com
mand="load" />
```

```
</node>

</launch>
```

A.14. Configuration Files

```
<launch>

<!-- Launch rplidar -->

<node pkg="rplidar_ros" type="rplidarNode" name="rplidarNode" output="screen"
>

    <param name="serial_port"          type="string" value="/dev/ttyUSB0"/>

    <param name="serial_baudrate"      type="int"    value="115200"/><!--A1
/A2 -->

    <!--param name="serial_baudrate"    type="int"    value="256000"--><!--A3 -->

    <param name="frame_id"             type="string" value="laser"/>

    <param name="inverted"              type="bool"   value="false"/>

    <param name="angle_compensate"      type="bool"   value="true"/>

    </node>

<!-- Connect to Pioneer -->

<node pkg = "rosaria" type = "RosAria" name = "RosAria">

    <param name="port" value="/dev/ttyUSB1" type="string"/>

</node>

<!-- Launch p3dx description -->
```

```

<arg name="model" default="$(find p3dx_description)/urdf/pioneer3dx.xml"/>

<arg name="gui" default="False" />

    <param name="robot_description" textfile="$(arg model)" />

    <param name="use_gui" value="$(arg gui)"/>

<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />

<node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

<node name="rviz" pkg="rviz" type="rviz" args="-d $(find urdf_tutorial)/urdf.rviz" />

<!-- transform configurations -->

<node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0.2659 0 0 0 base_link laser_frame 100"/>

<node pkg="tf" type="static_transform_publisher" name="link2_broadcaster" args="0 0 0.1 0 0 0 map odom 100"/>

<node pkg="tf" type="static_transform_publisher" name="link3_broadcaster" args="0 0 0.2659 0 0 0 base_link laser 100"/>

</launch>

```


A.15. Navigation Launch File

```
<launch>

<!-- Launch rplidar -->

<node pkg="rplidar_ros" type="rplidarNode" name="rplidarNode" output="screen"
>

    <param name="serial_port"          type="string" value="/dev/ttyUSB0"/>

    <param name="serial_baudrate"      type="int"    value="115200"/><!--A1
/A2 -->

    <!--param name="serial_baudrate"    type="int"    value="256000"--><!--A3 -->

    <param name="frame_id"             type="string" value="laser"/>

    <param name="inverted"              type="bool"   value="false"/>

    <param name="angle_compensate"     type="bool"   value="true"/>

</node>


<!-- Connect to Pioneer -->

<node pkg = "rosaria" type = "RosAria" name = "RosAria">

    <param name="port" value="/dev/ttyUSB1" type="string"/>

</node>


<!-- Launch p3dx description -->

<arg name="model" default="$(find p3dx_description)/urdf/pioneer3dx.xml"/>
```

```

<arg name="gui" default="False" />

    <param name="robot_description" textfile="$(arg model)" />

    <param name="use_gui" value="$(arg gui)"/>

<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" >

</node>

<node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

<node name="rviz" pkg="rviz" type="rviz" args="-d $(find urdf_tutorial)/urdf.rviz" />

<!-- transform configurations -->

<node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0.1 0 0 0 odom base_link 100"/>

<!-- <node pkg="tf" type="static_transform_publisher" name="link2_broadcaster" args="0 0 0.1 0 0 0 map odom 100"/> -->

<node pkg="tf" type="static_transform_publisher" name="link3_broadcaster" args="0 0 0.1 0 0 0 base_link laser 100"/>

<!-- Launch Gmapping -->

<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">

    <param name="map_udpate_interval" value="1.0"/>

    <param name="maxUrange" value="5.0"/>

    <param name="sigma" value="0.1"/>

```

```
<param name="kernelSize" value="1"/>

<param name="lstep" value="0.15"/>

<param name="astep" value="0.15"/>

<param name="iterations" value="1"/>

<param name="lsigma" value="0.1"/>

<param name="ogain" value="3.0"/>

<param name="lskip" value="1"/>

<param name="srr" value="0.1"/>

<param name="srt" value="0.2"/>

<param name="str" value="0.1"/>

<param name="stt" value="0.2"/>

<param name="linearUpdate" value="1.0"/>

<param name="angularUpdate" value="0.5"/>

<param name="temporalUpdate" value="0.4"/>

<param name="resampleThreshold" value="0.5"/>

<param name="particles" value="10"/>

<param name="xmin" value="-5.0"/>

<param name="ymin" value="-5.0"/>

<param name="xmax" value="5.0"/>

<param name="ymax" value="5.0"/>

<param name="delta" value="0.02"/>
```

```
<param name="l1samplerange" value="0.01"/>

    <param name="l1samplestep" value="0.05"/>

    <param name="lasamplerange" value="0.05"/>

    <param name="lasamplestep" value="0.05"/>

</node>

</launch>
```

References

- [1] H. F. Durrant-Whyte and J. J. Leonard, "Simultaneous map building and localization for an autonomous mobile robot,," *IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, vol. 3, pp. 1442-1447, 1991.
- [2] 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," *ICRA Workshop on Open Source Software*, 2009.
- [3] S. Thrun, W. Burgard and F. D., Probabilistic ROBOTICS, London: The MIT Press, 2006.
- [4] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," in *9th IEEE International Conference on Intelligent Systems*, Madeira, Portugal, 2018.
- [5] A. Diosi, G. Taylor and L. Kleeman, "Interactive SLAM using Laser and Advanced Sonar,," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 1103-1108, 2005.
- [6] Y. Abdelrasoul, A. B. S. H. Saman and P. Sebastian, "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM,," *2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, pp. 1-6, 2016.
- [7] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34-4, 2007.

- [8] C. Forster, M. Pizzoli and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry,," *IEEE International Conference on Robotics and Automation*, p. 15–22, 2014.
- [9] J. Civera and A. Concha, "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence," *IEEE International Conference on Intelligent Robots and Systems*, p. 5686–5693, 2015.
- [10] J. Engel, T. Schoeps and D. Cremers, "Lsd-slam: Large-scale direct monocular slam,," *European Conference on Computer Vision*, p. 834– 849, 2014.
- [11] N. Yang, R. Wang, X. Gao and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect,," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, p. 2878–2885, 2018.
- [12] M. Labbe' and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," *IEEE Int. Conference on Intelligent Robots and Systems*, p. 2661–2666, 2014.
- [13] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, "Orb-slam:aversatile and accurate monocular slam system,," *IEEE Trans. on Robotics*, vol. 31, no. 5, p. 1147–1163, 2015.
- [14] T. Pire, T. Fischer, G. Castro, P. DeCristo' foris, J. Civera and J. JacoboBerlles, "S-PTAM: Stereo Parallel Tracking and Mapping,," *Robotics and Autonomous Systems*, vol. 93, p. 27–42, 2017.
- [15] B. Gerkey, "ROS WiKi," [Online]. Available: http://wiki.ros.org/slam_karto. [Accessed 15 12 2010].

- [16] B. Steux and O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program,," *11th International Conference on Control Automation Robotics & Vision*, pp. 1975-1979, 2010.
- [17] L. Carlone, R. Aragues, J. Castellanos and B. Bona, "A linear approximation for graph-based simultaneous localization and mapping," *In Proc. of the Int. Conf. Robotics: Science and Systems*, 2011.
- [18] W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2d lidar slam," *Robotics and Automation (ICRA)*, p. 1271–1278, 2016.
- [19] S. Kohlbrecher, O. Stryk, J. Meyer and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155-160, 2011.
- [20] A. Filatov, A. Filatov and K. Krinkin, "2D SLAM Quality Evaluation Methods," *Conference of Open Innovations Association (FRUCT)*, no. 21st, pp. 120 - 126, 2017.
- [21] J. M. Santos, D. Portugal and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System,," *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1-6, 2013.
- [22] J. Esenkanova, H. Ilhan and S. Yavuz, "Pre-Mapping System with Single Laser Sensor Based on Gmapping Algorithm," *International Journal of Electrical Energy*, pp. 97-101, 2013.
- [23] P. Wang, Z. Chen, Q. Zhang and J. Sun, "A loop closure improvement method of gmapping for low cost and resolution laser scanner," *IFAC (International Federation of Automatic Control)*, 2016.

- [24] ActivMedia, *ActivMedia Robotics Pioneer 2/PeopleBot Operations Manual v9*, ActivMedia, 2001.
- [25] S. Slamtec.Co., "Slamtec," 13 03 2013. [Online]. Available: <http://www.slamtec.com/en/lidar/a1>. [Accessed 04 07 2016].
- [26] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," *IEEE international conference on robotics and automation*, pp. 300-307, 2010.
- [27] T. Zhang, Z. J. Chong, B. Qin, J. G. M. Fu, S. Pendleton and M. H. Ang Jr., "Sensor fusion for localization, mapping and navigation in an indoor environment," *IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM)*, no. 7th, 2014.
- [28] D. V. Lu, D. Hershberger and W. D. Smart, "Layered costmaps for context-sensitive navigation,," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 709-715, 2014.
- [29] S. Thrun, "An online mapping algorithm for teams of mobile robots," *Int. J. Robot*, vol. 20, no. 5, p. 335–36, 2001.
- [30] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai and B. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping,," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 22-29, 2010.
- [31] G. Dissanayake, H. Durrant-Whyte and T. Bailey, "A computation- ally efficient solution to the simultaneous localisation and map building (SLAM) problem,," *Proc. IEEE Int. Conf. Robot. Autom*, p. 1009–1014, 2000.