Technical University of Denmark

Jim Aldon D'Souza (s141380)

Mapping and Navigation for Space Robot System

Master's Thesis, July 2016

DTU Electrical Engineering
Department of Electrical Engineering

German Research Center for Artificial Intelligence

JIM ALDON D'SOUZA (S141380)

Mapping and Navigation for Space Robot System

Master's Thesis, July 2016

Supervisors:

Nils Axel Andersen, Associate Professor at the Electrical Engineering Department of DTU

Ole Ravn, Associate Professor at the Electrical Engineering Department of DTU

Daniel Hennes (Dr.), Senior Researcher at DFKI Robotics Innovation Center, Bremen

DTU - Technical University of Denmark, Kgs. Lyngby - 2016

Mapping and Navigation for Space Robot System

This report was prepared by:
Jim Aldon D'Souza (s141380)

Advisors:
Nils Axel Andersen, Associate Professor at the Electrical Engineering Department of DTU
Ole Ravn, Associate Professor at the Electrical Engineering Department of DTU
Daniel Hennes (Dr.), Senior Researcher at DFKI Robotics Innovation Center, Bremen

DTU Electrical Engineering
Automation and Control
Technical University of Denmark
Elektrovej, Building 326
2800 Kgs. Lyngby
Denmark
Tel: +45 4525 3576

s141380@student.dtu.dk

Project period: February 2016- July 2016

ECTS: 32.5

Education: MSc

Field: Electrical Engineering

Class: Confidential until October 1st, 2016

Remarks: This report is submitted as partial fulfillment of the requirements for graduation in the above education at the Technical University of Denmark.

Copyrights: ©Jim Aldon D'Souza, 2016

Table of Contents

List of Figures

Abstract

Space exploration rovers are tasked with navigating difficult, featureless and often unknown terrain. Existing rover systems rely on regular human input, orbital reconnaissance, constant lighting and feature rich patches of terrain for navigation. These limitations restrict the operational scope of a space mission, which makes reliable autonomous navigation and mapping an attractive prospect.

This thesis proposes a laser based navigation solution which uses autonomously deployable artificial landmarks to introduce features into the environment. A decoupled Simultaneous Localization and Mapping (SLAM) system with a scan matching front-end and a graph optimisation back-end is implemented using a laser scanner. The SLAM system is facilitated by a proposed method which uses indistinguishable environment modifiers to reduce misalignment errors and improve data association in ambiguous terrain.

Our solution is tested on a robot system with real world data. Experiments in approximated featureless environments show promising results across a variety of navigation tasks.

Acknowledgements

This thesis was written and realised at the Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI), Bremen, Germany during the fourth semester of my Masters in Electrical Engineering at the Technical University of Denmark (DTU) in Lyngby, Denmark.

I would like to express my sincere gratitude to my advisors Nils Axel Andersen and Ole Ravn for their constant support and guidance. I also wish to thank my DFKI advisor Daniel Hennes. The door to his office was always open whenever I ran into a tight spot and this thesis wouldn't have come to fruition without him. A special thanks to Sebastian Kasperski for his expertise and invaluable input on a host of topics.

I'm grateful to Janosch Machowinski, Javier Hidalgo Carrio and the entire Envire team at DFKI for their immense help and insight which made experiments on a robot with real world data possible. I also wish to thank my friends for the stimulating discussions, and their help in reviewing this thesis.

I am indebted to my parents and my sister supporting for encouraging me every step of the way. Their unwavering belief in me is a big part of the reason I am here today.

- Jim Aldon D'Souza

CHAPTER 1

Introduction

"Our universe is a sorry little affair unless it has in it something for every age to investigate."

–Seneca

Privatization of the space industry (Solomon, 2011), reusable launch vehicles (Young, 2015) and advances in robotics leave little doubt that space exploration is gaining mainstream focus. Semi and fully autonomous robotic systems are an essential part of the international space exploration effort, with four planned lunar rover missions in 2017 (2 by Google Lunar XPRIZE, *Chandrayan-2* by India and *SELENE-2* by Japan), and several Martian rover missions planned by the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the Canadian Space Agency (CSA) and China National Space Administration (CNSA) before 2020.

Although robots enjoy the obvious edge of cost-effectiveness, accessibility and overall practicality, they lack the operational flexibility and native intelligence of humans. Compounding this are factors like slow navigation due to transmission delay, restricted communication windows, limited bandwidth and constraints due to a small perception range; thus making autonomous navigation pertinent for bolstering space exploration missions.

Plans for a research facility on the Moon, or a lunar base are in the works, and the international space community hopes to have it ready by 2050. Akin to construction activity on Earth, operations like site selection, site preparation and base construction will require detailed 3D maps. In contrast to exploratory tasks on Mars, lunar rovers would perform extended operations in local work site environments and would be required to visit the same location multiple times. Ultimately, creating globally consistent, accurate 3D maps in the absence of infrastructure based positioning systems (like GNSS) will be one of the first steps in the lunar base undertaking. As explained in the subsequent sections, mapping an unknown environment necessitates accurate localization of rover poses.

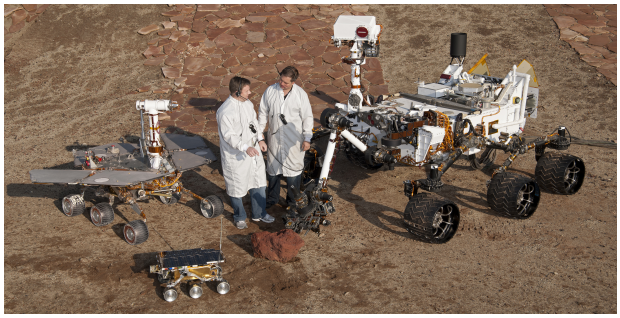
The focus of this work is to improve autonomous capabilities, specifically navigation, of independent rovers in planetary environments.

1.1 The Martian Rovers/State of the Art

NASA's twin Mars Exploration Rovers (MER) Spirit and Opportunity (operational since early 2004), and the 900kg Mars Science Laboratory (MSL) or Curiosity rover (operational since August 2012), have logged almost $64kms$ between them, providing a *benchmark* for state of the art exploration on Mars. While neither of them is fully-autonomous, they do implement and make use of localization techniques in the form of ego-motion or Visual Odometry **(VO)** (Olson et al., 2003)(Maimone et al., 2007) and Wheel Odometry **(WO)**, via a system called Autonav. While VO and WO result in relative localization of the rovers (dead-reckoning), the global localization has to be done manually by the control centre using maps generated by orbiter reconnaissance. The ExoMars rover, part of the first flagship mission of the Aurora programme due to launch in 2020, initiated by ESA also features similar sensors and navigation techniques as Curiosity. In essence the extent of autonomous driving in the rovers is navigating between manually declared waypoints, while localizing itself on an *a-priori* map, keeping drifts in position estimates to a minimum (*eg.* 10% during a $100m$ drive on the MER (Maimone et al., 2007)).

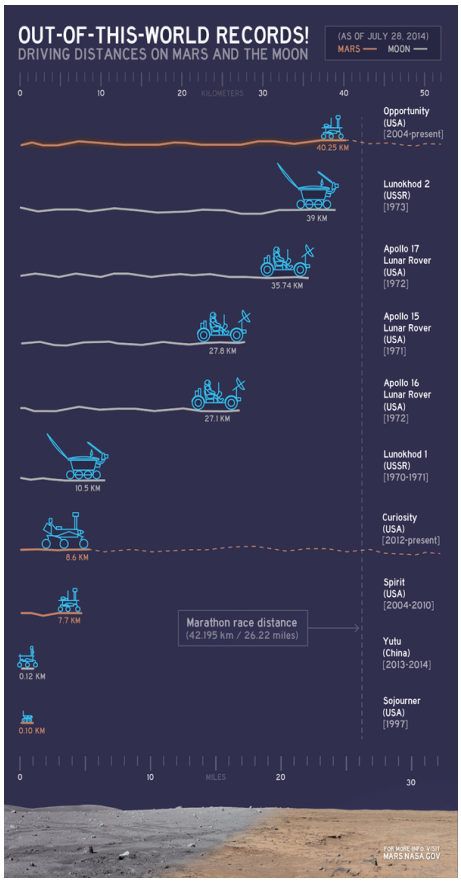
The rovers use VO along with the on-board Inertial Measurement Unit (IMU) to compute the translation and rotation between two poses by tracking movement of ground features like rocks using stereo cameras. WO helps calculate distance travelled by rover using encoders on wheels, and Sun sensors provide absolute orientation. However, the localization accuracy depends highly on the available features and the terrain type. While VO fails when there are few features to track, WO becomes unreliable when the rover traverses non-cohesive surfaces like sand which causes slippage. The problem compounds when the rover has to travel through a sandy terrain with no rocks on the surface, as both VO and WO then become unreliable. To counter the above limitations, NASA uses approaches like the D* algorithm (Stentz, 1995) for long-range path planning which accounts for localization errors by adding uniform margins to obstacles. Other approaches like path planning to ensure superior localization, and minimize entropy, called active localization are proposed (Inoue et al., 2016). Some other proposed methods mediate between localization accuracy and path cost using particle filters (Correa and Soto, 2010) or planning in belief states (Prentice and Roy, 2010). However, the underlying problem persists, which necessitates regular human intervention to ensure the rover is on the right path.

We will use the Martian scenario as a motivation for this thesis, although the design decisions made will be applicable for general planetary rover operations.



(a) Sizes of Soujourner, Opportunity and Curiosity

(b) Wheel sizes of Martian rovers



(c) Distance driven by rovers on Mars and Moon

Figure 1.1: Planetary Rovers (Photograph courtesy of NASA/JPL)

1.2 Robotic Mapping and Navigation

Effective navigation in diverse and uncharted environments is a key concern in robotic exploration. Navigation in unknown environments depends on robust mapping and localization by the robot. Mapping is the problem of assimilating perceptual information gathered with sensors into a representation and localization is estimating the sensors' pose within that representation. Since these two are co-dependent, the problem of Simultaneous Localization and Mapping (SLAM) or Concurrent Mapping and Localization (CML) deals with the task of concurrently solving the mapping problem and the induced localization problem.

The problem of SLAM in 3D environments or 3D SLAM is tackled by a variety of approaches which are tailored for application compliance and available resources. Methods which can be classified by their map parametrization like spatially located landmarks (using Extended Kalman Filters (Weingarten and Siegwart, 2005)), 3D occupancy grid maps (Endres et al., 2014) or raw range data (Nuchter et al., 2007)(Alismail et al., 2014) aim to solve the 3D SLAM problem. For reasons discussed later, we will concern ourselves with the third approach.

1.3 Motivation

To reiterate, below are some navigational limitations of the current state-of-the -art deployed semi-autonomous robotic systems

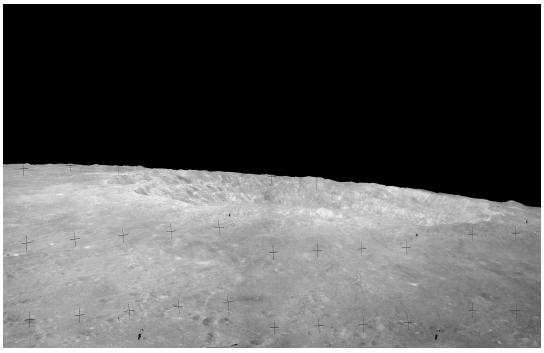
- Scarcity of natural features in environment.
- Only dead reckoning and some active localization implemented. No system for mapping in place.
- Inability to navigate in terrain not remotely mapped before or in structures not mappable by observations from telescopes or orbital satellites (caves, the dark side of the moon, etc.).
- Error prone stereo ego-motion due to aforementioned lack of features and unreliable wheel odometry due to non-cohesive and diverse nature of terrain.
- Daytime only operation since the primary sensors are stereo camera systems.
- Semi autonomous navigation is slow due to limits on communication with Earth (usually only once per day).

Most SLAM methods in Section 1.2 depend on an abundant availability of local points of interest which serve to make environments more uniquely representable, and as a consequence, aid map building and localization. Traditionally, SLAM algorithms have been developed for applications in urban environments which have no dearth of local interest points or landmarks. However, as discussed above, robotic navigation on planetary surfaces does not typically enjoy such settings and can be comparable to navigation in a featureless environment like a desert, as can be seen in Figure 1.2.

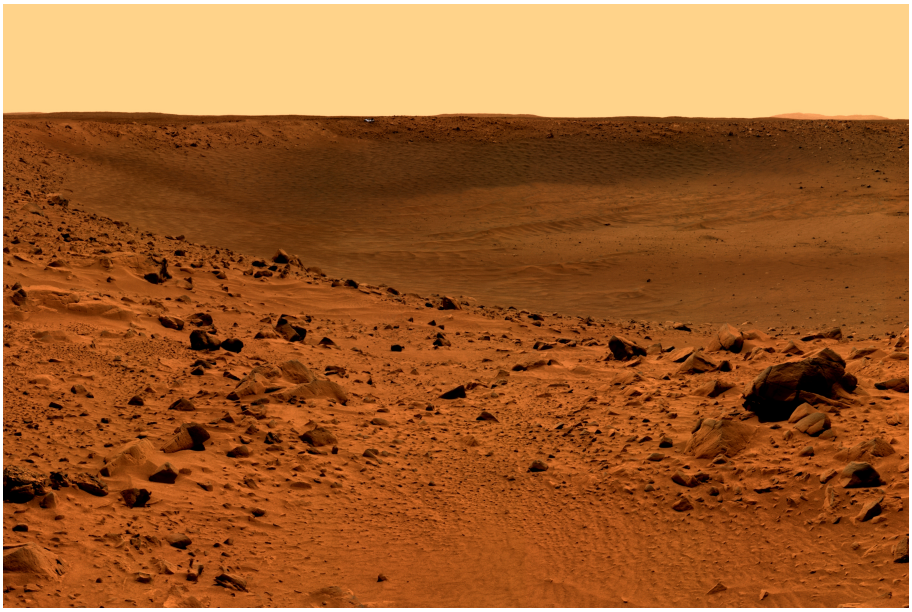
In contrast to NASA's Autonav, our system operates with no knowledge of the terrain and works to build a map of the environment as it navigates in it. We also use a 3D Light Detection and Ranging device (LIDAR) as our primary sensor instead of the stereo camera setup. A laser scanner overcomes the constraint of daytime navigation while providing high frequency three dimensional range data which can be used to build dense environment representations in the form of detailed surface maps.

In addition to addressing the navigation problems due to scarce features, this thesis aims to overcome limitations of a rover navigating in areas (caves, for example) which has not or can not be mapped in advance by methods like orbital reconnaissance.

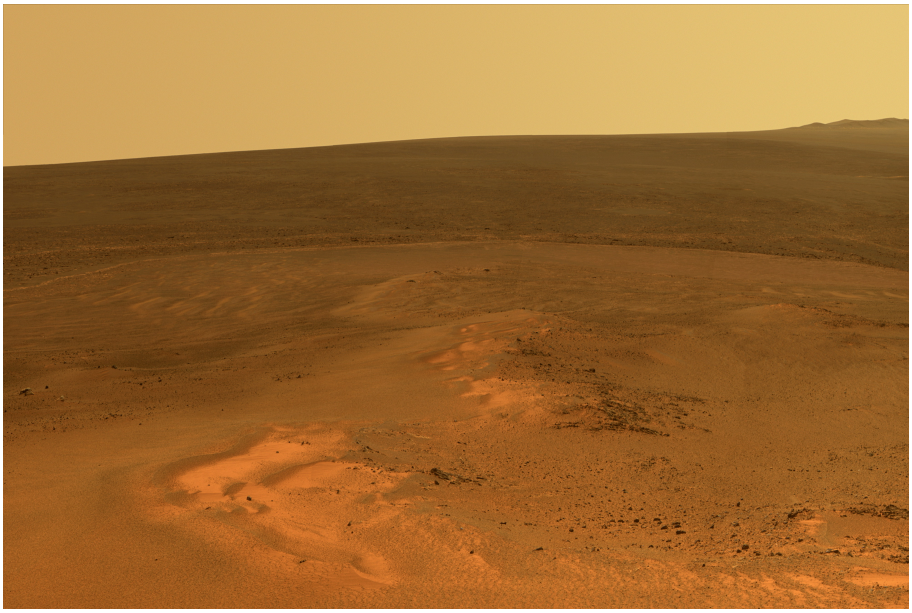
(a) Lunar surface - Apollo 15 mission



(b) Copernicus crater on the moon - Apollo 15



(c) Bonneville crater - Spirit



(d) Greeley Haven - Opportunity

Figure 1.2: Pictures of Lunar and Martian surface by NASA sources. There is a noticeable lack of usable features for autonomous robotic navigation common on both bodies.

1.4 Goal

Within the context of robotic space exploration, the nondescript and barren nature of the surroundings where the rover operates motivates the *primary focus* of this thesis. This work aims to investigate the effectiveness of environment modifiers (or artificial landmarks) in reducing the ambiguity of the scene. We introduce local interest points or features which are used for better localization, easier map creation and to mitigate data association or correspondence errors.

1.5 Thesis Outline

The rest of this thesis is structured as follows

- Chapter 2 provides a comprehensive review of existing literature on SLAM topics pertaining to this thesis. This includes topics on localization and mapping of robotic systems, raw scan matching methods and global map refinement methods. It also gives an overview of the available works on using environment modifiers to facilitate localization and/or mapping.
- Chapter 3 describes the framework of the system this thesis was carried out in. This includes the hardware like the sensors and the robot used; and the in-house software framework used to develop working components and extensions on the robot system.
- Chapter 4 details the landmark detector and the decoupled SLAM system developed as part of this thesis. It also outlines how the detected landmarks can be used to enhance our SLAM system.
- Chapter 5 describes experiments set-up and the results obtained. It shows the results for different test scenarios of the robot and the SLAM system in outdoor environments.
- Chapter 6 summarizes the work, acknowledges limitations and proposes directions for future work that could overcome those limitations.

CHAPTER 2

Literature Review

This chapter provides an overview of the approaches used in our system and the prevalent techniques used in literature. Section 2.1 introduces navigation and mapping for mobile robot systems and traces their origins in literature. It then attempts to give an overview of the SLAM systems in use. Section 2.2 presents a survey of the state-of-the-art in robotic navigation and mapping in 3D environments and Section 2.3 discusses literature related to artificial landmark aided localization.

2.1 Mobile Robot Navigation

Mobility in robots gives them access to many new operational capabilities and opens up new areas of investigation like navigation and locomotion. In structured and static environments like planetary surfaces, robot perception allows for scene map generation, which can be used for localization and motion planning. Perception is achieved using extereoceptive or external sensors which measure the external state of the robots, and proprioceptive or internal sensors, which measure the internal state like odometry (shaft encoders), orientation (inertial measurement units), heading (compass, inclinometer) and absolute global position (GPS/GLONASS receiver). The external sensors are usually laser, sound or vision based. Laser systems, like sonar (sound based) are accurate active sensors which most commonly operate on a time-of-flight principle, where the time taken by a laser or sound pulse to reflect off the environment and be detected by the sensor is used to measure distances. Sonar however lacks appearance data and can be used only in environments with a medium (like water or air), unlike laser systems which can be used in vacuum. Vision based systems are passive, with high resolutions and long range. They estimate depth information, feature location and 3D formations through stereo camera set-ups or through Structure from Motion (Jebara et al., 1999).

The sensor systems discussed above are all plagued by uncertainty and perceptual noise. To accommodate this, algorithms employ probabilistic models of the robot, its environment and the sensors. As a result, almost all state-of-the-art robotic mapping algorithms today are built on probabilistic mathematical foundations (Thrun, 2002). Moreover, they're versions of the Bayes filter model, which is essentially a recursive method to estimate a robot's pose from the sensor

and motion model. A classification of some of the most popular algorithms is shown in Figure 2.1 (adapted from Garcia et al. (2007)). The yellow double ellipses indicate the basis of classification, the bold text shows the classifying label, the blue rectangles represent a class of methods and the green circles show the methods.

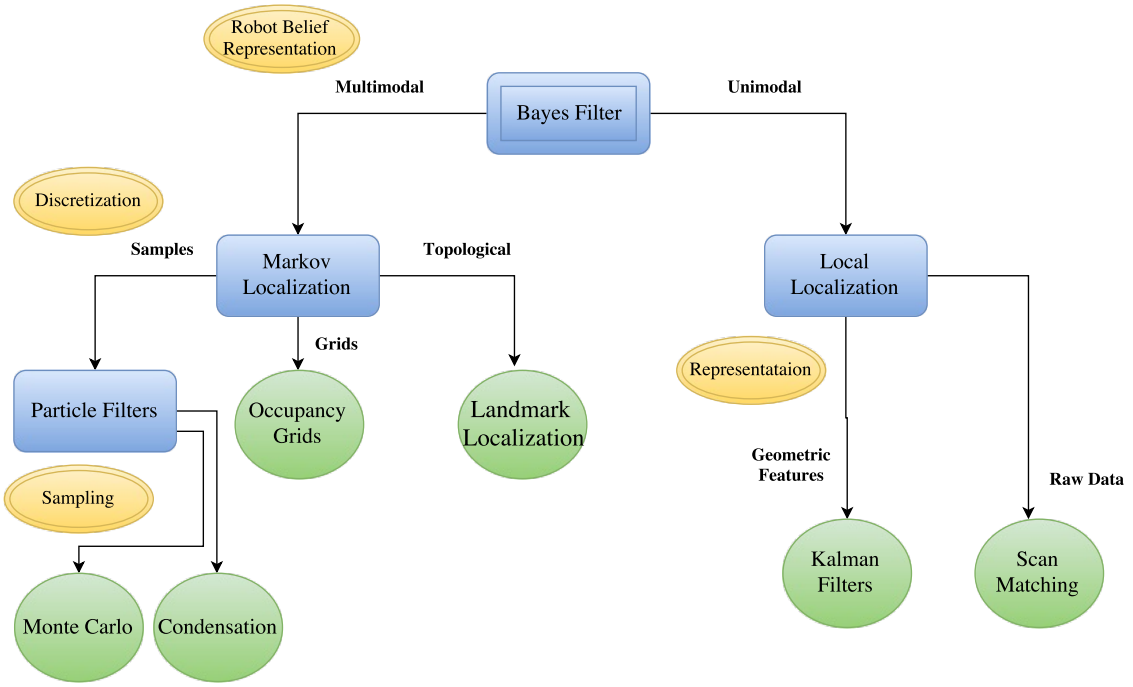


Figure 2.1: A classification of some popular methods used for localization and mapping

Mapping using Kalman Filter (KF) (Gamini Dissanayake et al., 2001) assumes that the uncertainty is multivariate Gaussian, and that both perception and motion model is linear with Gaussian noise. There are two main variations of KF: Extended Kalman Filter (EKF) (Leonard and Newman, 2003) which accommodates environment non linearities by approximating the robot model using linear functions; and Information and Extended Information Filtering (IF and EIF) (Thrun et al., 2003), which is realized by propagating the inverse of the state error covariance matrix. The Unscented Kalman Filter (Wan and van der Merwe, 2002) addresses the linearity assumptions in KF and approximation issues in EKF. The major drawback of Kalman filters is its assumption of independent unimodal Gaussian noise, which restricts their scalability with increasing landmarks and data association in longer paths.

Particle Filters or sequential Monte-Carlo method approximate the belief distribution of a pose by a set of particles not constrained to a grid. They can be applied for global localization or for SLAM (FastSLAM (Montemerlo et al., 2003)). This approach can be tuned to scale linearly with increasing map features, but is sensitive to the "particle depletion problem" (Van Der Merwe et al., 2001) which loses particles in a local loop. Moreover, the performance of particle filters depends on the number of particles.

Occupancy Grid (Thrun, 2001) methods generate probabilistic maps when the robot's poses

are known. The grid represents pose belief as a probability distribution over all possible robot poses. It is therefore multimodal and good for global localization, but assumes independent noise and cannot handle pose uncertainty.

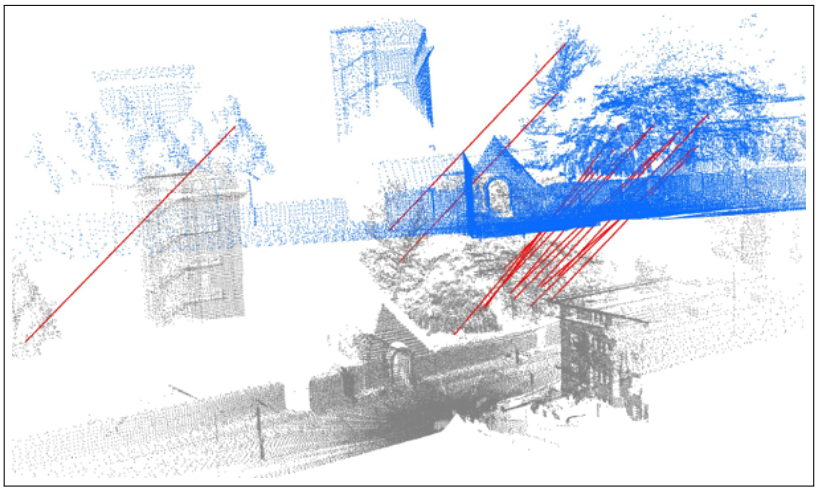
2.2 3D SLAM

Many approaches in literature use passive cameras for 3D SLAM in unstructured environments. Using a stereo camera system, Konolige and Agrawal (2007) used frame-to-frame VO and bundle adjustment for loop closure and demonstrated good alignment results. Relative bundle adjustment was used in a system called Relative SLAM (RSLAM) by Mei et al. (2011) showing localization over an unprecedented 142 km of travel. However, a small field of view, reliance on external lighting and restrictive power requirements for sufficient illumination (Solomon et al., 2014) make cameras unsuitable for autonomous navigation and large scale mapping under a varying illumination profile of a planetary surface. Illuminated sensors with a large field of view, such as laser scanners, are a better alternative.

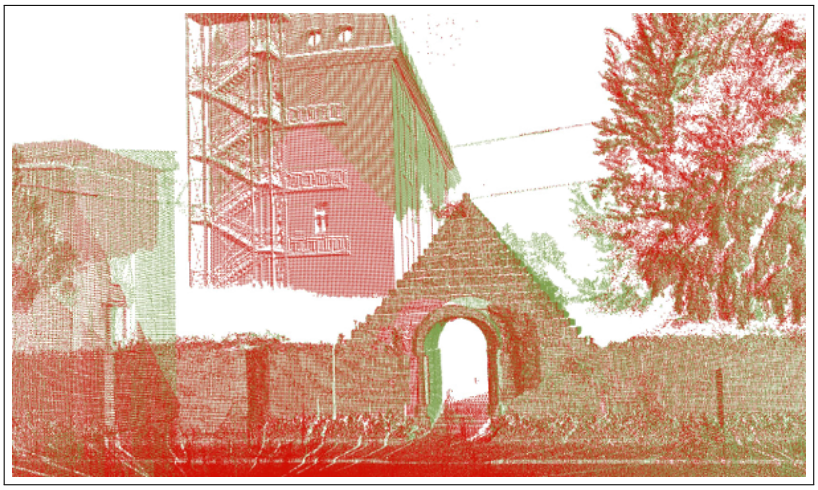
Laser scanners output point cloud data in multiple scans. Aligning data from consecutive scans refers to finding homogeneous relation between them and tracking the position of the scanner through the environment. Current literature for aligning point cloud data offers two approaches: exploiting sparse geometric features and operations on dense point clouds.

In the sparse approach, geometrically distinct interest points are extracted from each scan and tracked. These interest points can be regions of high curvature (Bakambu et al., 2006) or peaks in the terrain (Carle et al., 2010; Tong et al., 2012). This method works with a reduced representation of the scan and is computationally efficient. However, it requires construction of feature descriptors for each point which can be detected across multiple scans. Many geometric descriptors in literature have been introduced, including spin images (Johnson, 1997), point fingerprints (Sun et al., 2003) and local shapes (Taati et al., 2007).

In dense scan matching or registration methods, consecutive scans are aligned by the sensor of a moving robot to estimate its trajectory and create a consistent map. Besl and McKay in 1992 (Besl and McKay, 1992) proposed a method to register and align 3D shapes called the Iterative Closest Point (ICP) and proved its convergence. Lu and Milios (1997) used maximum likelihood to build maps from raw 2D data with unknown correspondences. ICP looks for closest point pairs (using Euclidean, Mahalanobis or other heuristic) in two consecutive scans and estimates the 3D rotational and translational transformation by minimizing a cost function (Figure 2.2). Scan matching isn't suited for global localization since it's essentially a tracking method and requires the ability to model multimodal belief distributions of robot pose. Moreover, this method is highly sensitive to the quality of initial guesses as these are used for initialization. Poor initial guesses make for a wrong start and result in convergence to an incorrect local minimum. Surmann et al. (2003) used ICP to create precise 3D maps using a rotating laser scanner on a mobile robot in a



(a) Geometric registration (or scan matching) matches two scans to find point correspondences



(b) Registration algorithm uses matches to align the scans taken from two different sensor poses

Figure 2.2: Example of 3D point cloud registration of an outdoor scene (Figures from PCL (2011))

structured indoor environment. Other methods are proposed which can simultaneously register multiple point clouds (Craciun et al., 2010), unlike ICP which is valid only for pairs of scans. In the case of 3D SLAM, where the use of full 3D data becomes necessary, algorithms like ICP which specialize in working with highly dense 3D range data outperform all other methods.

Due to sensitivity of the iterative method to noise and poor iteration, many variants of the ICP have been developed to improve the five consecutive steps: selection, matching, rejection, weighing and transformation estimation. Out of these, the first four compose the correspondence finding part, and estimation of the transformation is done by minimizing a given function. The matching step is the biggest bottleneck and improving convergence rate is crucial in making the registration process faster. To improve matching, robust features should be found which can also improve erroneous correspondences between points. This would also enable the registration process to

focus on reliable and relevant regions and produce better transformations. Local descriptors based on structural point cloud analysis, like the geometrical curvature and position uncertainty used in Sharp et al. (2002).

Our scan matching method builds on the work by (Segal et al., 2009) called Generalised-ICP (GICP) which builds a probabilistic framework around 'point-to-plane' ICP. It utilizes Maximum Likelihood Estimation (MLE) as the non-linear optimisation step and uses fast and efficient kd-trees to compute discrete correspondences (Section 4.3.1).

GICP and variants are concerned with pairwise matches, so alignment errors invariably compound. An additional layer is required to ensure global consistency and loop detection. Nuchter et al. (2007) in their 6D SLAM framework used ICP to attain pairwise alignments and used a global relaxation technique as post processing. While it is able to solve pairwise linkage inconsistencies, this method evaluated loop closures with a simple distance criteria (Borrmann and Elseberg, 2008) and long loops were not detected. A multi-frame odometry-compensated global alignment (MOGA) algorithm was proposed by Carle et al. (2010) which used a combination of sparse features and odometry measurements in a batch alignment framework to match laser scan data to orbital maps. Tong et al. (2012) builds upon MOGA but obviates the use of orbital maps and uses a hybrid of sparse features and dense data for alignment. While this method generates detailed maps, it isn't suitable for use in an autonomous exploratory context since it cannot incrementally build estimated maps for navigation.

Our approach uses Graph optimisation (Thrun, 2006) for refining our map, which requires the construction of a graph or a network. The poses of the robot are modelled by nodes in a graph and labelled with their position in the scene. The edges joining two or more nodes encode the constraints between robot poses (Figure 4.9). These poses result from observations or from odometry measurements. In our method, the SLAM problem is decoupled into two tasks: a front-end, which deals with graph construction using ICP, and a back-end, which determines the most likely configuration of the nodes given the constraints in graph edges.

2.3 Artificial Landmarks

Even though our graph optimisation SLAM is specifically tasked with optimising and creating a globally consistent map, the output is only as good as the poses in the nodes and constraints in the edges of the graph. This information is fed in by our variation of the Generalized ICP front-end, which is prone to produce bad transformations due to alignment and data association errors. In the context of planetary navigation, this is most often caused by the ambiguity and lack of geometric features in the environment. Our work aims improve localization and reduce correspondence errors using environment modifiers or artificial landmarks.

Many approaches in literature tackle this problem without using artificial landmarks. Some

rely on natural features (Leonard and Durrant-Whyte, 1991; Burgard and Fox, 1996) and some use methods like single cluster graph partitioning to reject local matches that are not globally consistent (Olson, 2009). Thrun (1998) selects a subset of observed natural landmarks that minimizes the average posterior localization error. However, these methods fail to perform when the environment becomes increasingly ambiguous like in our scenario.

The theme prevalent regarding the use of artificial landmarks to aid navigation, is solving the *landmark placement* problem. The approaches are concerned with finding an optimal set of landmark positions along the desired trajectory of a robot to enhance its localization or navigation performance. Salas and Gordillo (1998) consider it as an art gallery problem (Lee and Lin, 1986) and try to maximize and area in which the robot has a clear line of sight to at least one landmark. Erickson (2011) use the colour information of landmarks to add bounds on visible landmarks of same colours. Sala et al. (2006) propose an extension of this solution where at least n landmarks are visible from every point in the map. A metric for pose uniqueness was introduced to help choose a landmark configuration that would minimize the average ambiguity of the environment (Meyer-Delius et al., 2011). Beinhofer et al. (2013b) proposes a method to place the minimum number of landmarks keeping a bound on the maximum deviation of a robot from its desired trajectory. In an extension (Beinhofer et al., 2013c) he makes the placement method robust to missing landmarks due to occlusion sensor noise. Some methods tackle the issue using uniquely identifiable landmarks (Rafflin and Fournier, 1995; Howard et al., 2001). Even though this facilitates localization and eases association, it requires landmark coding, a detection and identification system and complex deployment strategies. Unlike these methods, since we're using landmarks in an exploratory navigation context, we do not hold the luxury to install environment modifiers in a specific configuration *before* the robot's motion. Additionally, our landmarks are indistinguishable which helps overcome associated logistical challenges.

Like us, there are works in literature that consider autonomous landmark deployment. Many of these address graph like worlds with deterministically observable markers, including Dudek et al. (1997) who localize a robot travelling along the edges of a graph, detecting markers at the vertices; and Bender et al. (2002) who use markers to map a directed graph. Wang et al. (2011) use deterministically observable directed markers to do SLAM in an undirected graph. Batalin and Sukhatme (2003) present an innovative coverage strategy where a robot deploys active markers and uses them to move into the direction suggested by the markers. The robots start with no initial pose estimates. Kleiner et al. (2006) have designed a heuristic that depends on RFID tags and they estimated obstacle density to deploy RFID markers for aiding a SLAM system. Strasdat et al. (2009) and Beinhofer et al. (2013a) use Monte Carlo reinforcement learning for computing a policy for deploying uniquely identifiable landmarks that minimized the distance between the final robot position and the goal. The difference between the two works is that the latter incorporates the spatial structure of the environment into the learning method.

In contrast to the above methods, we deploy indistinguishable landmarks on-line, along the

trajectory of the robot, in no specific configuration.

CHAPTER 3

System Framework

This chapter introduces various software and hardware components of the system used throughout this thesis. Section 3.1 talks about the sensors used for data collection and Section 3.2 deals with the robot system used for experiments. Section 3.3 gives an overview of the Rock robotics software framework and touches on the real time communication aspect of it.

3.1 Sensors

This section will describe the working and performance of the laser scanner systems used for acquiring dense 3D point cloud datasets used throughout development, testing and experiments.

3D laser data collection on a moving platform like the rover can be quite challenging due to the relative motion between the scans and the sensor. Due to cost factors, 2D scanners such as the SICK LMS291 (Figures 3.1a, 3.1b) or Hokuyo URG-04LN, mounted on a rotating platform have been a popular sensing method for robot navigation tasks. However, a large scan time relative to robot motion produces distorted and misaligned point clouds and makes it unfit for use. To counter this, many works (Nüchter et al., 2005; Ryde and Hu, 2007) favour stationary scans by frequently stopping the sensor platform, but making the task inefficient. Circular 2D line scanner systems like the Reigl VQ-250 and Optech lynx manage to provide a detailed 360 degree view, but are prohibitively expensive since two or more of these are required for multiple scan angles and occlusion removal. They are also quite bulky (Figures 3.1c and 3.1d) and not generally favoured for mobile robot applications.

David and Bruce Hall developed a high speed multi channel integrated laser scanner for use at the 2005 Defense Advanced Research Projects Agency (DARPA) challenge, after recognizing the limitation of using stereo vision for fast mapping in the previous year's challenge. They further developed and miniaturised the design to create the Velodyne HDL-64E model which was widely adopted (Huang et al., 2010) in the 2007 version of the challenge.

(a) One scanner configuration for 360 degree scan

(b) 2 scanner configuration (Fraunhofer IAIS)

(c) Reigl VQ-250

(d) Optech Lynx

Figure 3.1: 3D Laser Scanner (LiDAR) alternatives [top: Rotating 2D scanner configurations for 3D data; bottom: Bulky circular 2D scanner systems on SUVs]

In this thesis, we use two scanners from Velodyne's range of high speed multi element 3D lidars: the smaller 16-channel VLP-16 and the 32-channel HDL-32E. The VLP-16 was used during the development process and for hand-held indoor tests, while HDL-32E was used as the sensor for outdoor tests with mobile robot systems. We will discuss VLP-16 in the next section, since HDL-32E shares most of the same principles.

3.1.1 Velodyne VLP-16

The Velodyne VLP-16 or *Puck* is a small, lightweight, real-time 360° LiDAR (or lidar, Figure 3.2) which outputs calibrated reflectivity measurements in addition to range data. Apart from lower resolution, the thing that differentiates this scanner from its costlier counterparts, HDL-32 and HDL-64, is the absence of any visible rotating parts, which makes it highly resilient to challenging environments. Table 3.1 outlines some salient features and compares it with the HDL-32E used on the robot.

Calibrated Reflectivity

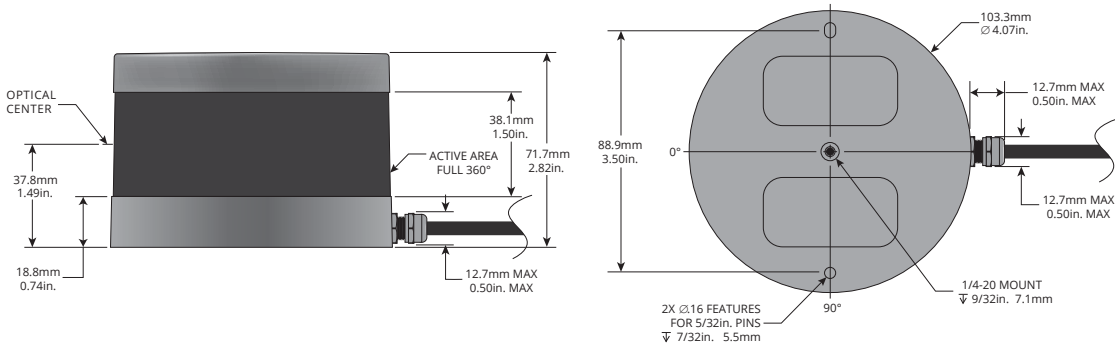
An important feature of the Velodyne lidar is its ability to measure calibrated reflectivity of the surface the laser bounces back from. This feature is the motivating factor for using reflective

	VLP-16	HDL-32
Channels	16	32
Range	100 metres	80-100 metres
Accuracy	+/- 3cm	+/- 2cm
Data Rate	300,000 pts/sec	700,000 pts/sec
Vertical Field of View	30°(+/- 15°)	40°(+/- 20°)
Vertical Resolution	2.0°	1.3°
Horizontal Resolution	0.4°@ 20Hz	0.35°@ 20Hz
Power	8W	12W
Weight	0.83 kg	1 kg
Operating Temperature	-10° to 60°C	-10° to 60°C

Table 3.1: Comparision of Velodyne VLP-16 and HDL-32 LiDARs

(a) Velodyne VLP-16 with a pencil for scale

(b) Light & power efficient (*Pheonix Aerial*, 2014)



(c) Scanner Dimensions (from *VLP-16* (2016))

Figure 3.2: Velodyne VLP-16 LiDAR

artificial landmarks as environment modifiers. The VLP-16 measures the reflectivity of an object with 256-bit resolution. Standard reflectivity standards and retro-reflectors are used for the absolute calibration of reflectivity, which is stored in a the FPGA of the scanner (Figure 3.3). They are calibrated with two types of reflectors:

- **Diffuse reflectors** report values from 0-100. This is also the range for the reflectivity values from most scans in normal modes of operation. Reflectivity from the artificial landmarks feature on the higher end of this scale.
- **Retro-reflectors** report values from 101-255, with 255 being calibrated with an ideal retro-reflector the every value below in that range for imperfect or partially obscured reflectors.

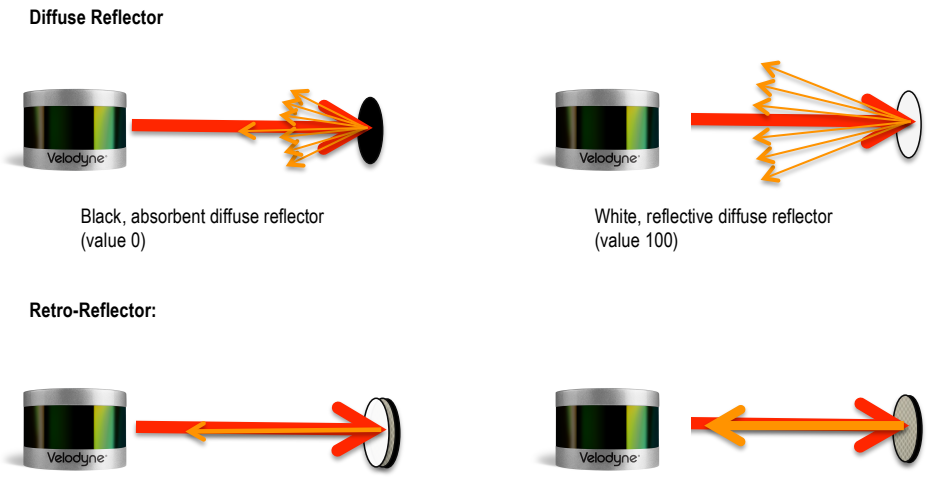


Figure 3.3: 256-bit Factory Calibration of VLP-16 with Corresponding Reflectors (*VLP-16*, 2016)

Sensor Driver

The sensor's driver is developed as an *oroGen* component (Section 3.3) and is responsible for carrying out the following operations in order to acquire the 3D range and reflectivity data:

1. Establish communication over ethernet connection with VLP-16
2. Parse UDP data packets for azimuth/rotation angle (α), measured distance/range (r) and measured reflectivity (I).
3. Calculate the Cartesian coordinates (x, y, z) from the azimuth, range and vertical angle (ω, fixed and depends on laser ID)
4. Store/pass data to other components

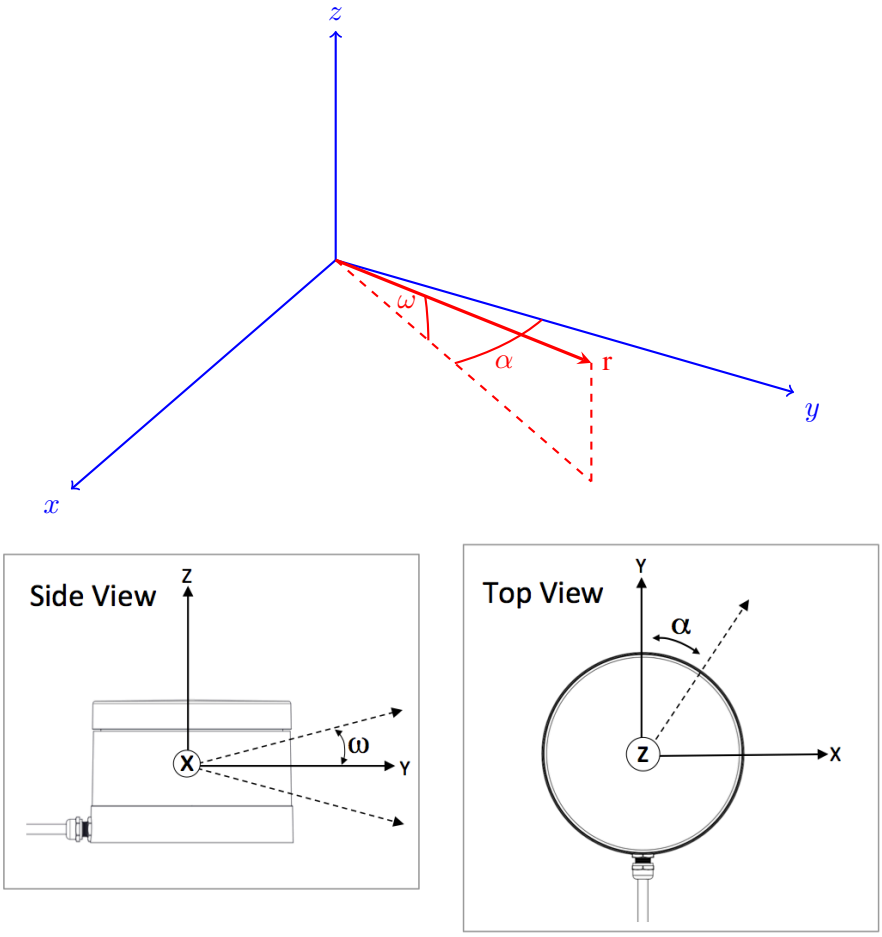


Figure 3.4: Transformation from Spherical to 3D Coordinates

Calculating Cartesian Coordinates

The VLP-16 reports data in spherical coordinate format (r, ω, α) which can be converted into x, y and z coordinates in the sensor's frame of reference like shown in Figure 3.4.

$$x = r * \cos\omega * \sin\alpha \tag{3.1}$$

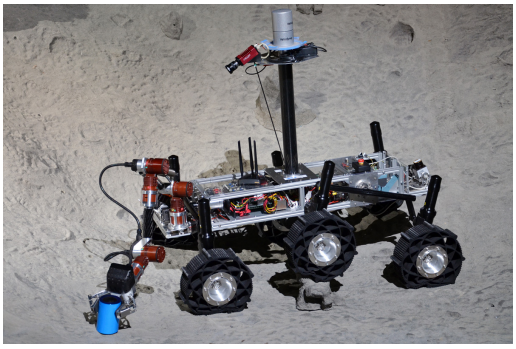
$$y = r * \cos\omega * \cos\alpha \tag{3.2}$$

$$z = r * \sin\omega \tag{3.3}$$

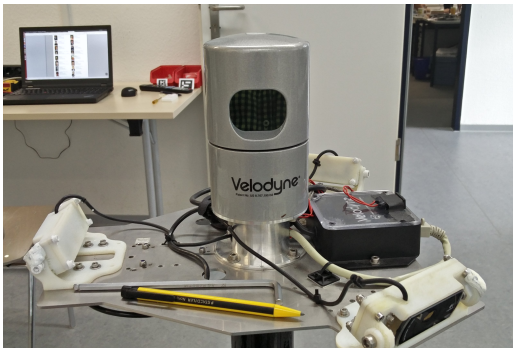
3.2 Rover: Artemis

The Artemis rover used for experiments in Chapter 5 was developed by DFKI as a competitor for DLR's annual SpaceBot Cup (Kaupisch and Noelke, 2014). Artemis (Figure 3.5a) was designed to navigate in challenging terrain and interact with objects.

Artemis uses a 32-channel HDL-32E Velodyne LiDAR (3.5b) which has an output of more



(a) The Artemis rover used in DLR's SpaceBot Cup



(b) Velodyne HDL-32E used in Artemis

Figure 3.5: Rover and sensor used for experiments

than double the number of points as the VLP-16 (Table 3.1). Due to this, we use methods like Voxel Grid downsampling to manage the processing overhead without losing important features of the cloud. The lidar is mounted at a height of 1.5m above ground.

3.3 Robotic Software Framework: Rock

The development work in this thesis is done entirely in C++ and Ruby in the form of extensions to a real time Robotic framework created at the Robotic Innovation Center in DFKI Bremen. This section describes the most essential features of Rock and shows how the thesis benefited from the modularity provided by this solution.

Rock (**Ro**bot **C**onstruction **K**it) is a robot software integration framework developed at DFKI and used there for most in-house robot systems. From the official website (DFKI Robotics Innovation Center, n.d.):

> Rock (*Rock, the Robot Construction Kit*, 2011) is a software framework for the development of robotic systems. The underlying component model is based on the Orocos RTT (Real Time Toolkit). Rock provides all the tools required to set up and run high-performance and reliable robotic systems for wide variety of applications in research and industry. It contains a rich collection of ready to use drivers and modules for use in your own system, and can easily be extended by adding new components.

Frameworks can aid management of complex software and has been shown to be increasingly useful in robotics where a lot of inter-dependent but functionally different software has to be implemented. Given the need for autonomy and complex systems on a robot system deployed for space missions, Rock has been found (Joyeux et al., 2014) to be a strong candidate as a robotic framework owing to its design motivations. Joyeux et al. (2014) also demonstrate it's effectiveness on the same rover used in this thesis, Artemis, which participated in DLR's Spacebot Cup (Kaupisch and Noelke, 2014) challenge.

Figure 3.6 shows the components of Rock that help it achieve the following design motivations:

- **Application agnostic**: Components can be reused across applications and only needs to be adapted at the time of development.
- **Single-purpose components**: Rock has a policy split between libraries and modules, which separates functionality from the communication layer. This makes possible the design of single purpose modules/components which can leverage common functionalities from libraries and still be distinct.
- **Composability**: Rock's native support for flexible module deployments makes it easy to build subsystems using components - and build larger (sub)systems using those subsystems.
- **Robustness**: Rock allows for seamless switching and reconfiguration, at runtime, of different component networks.

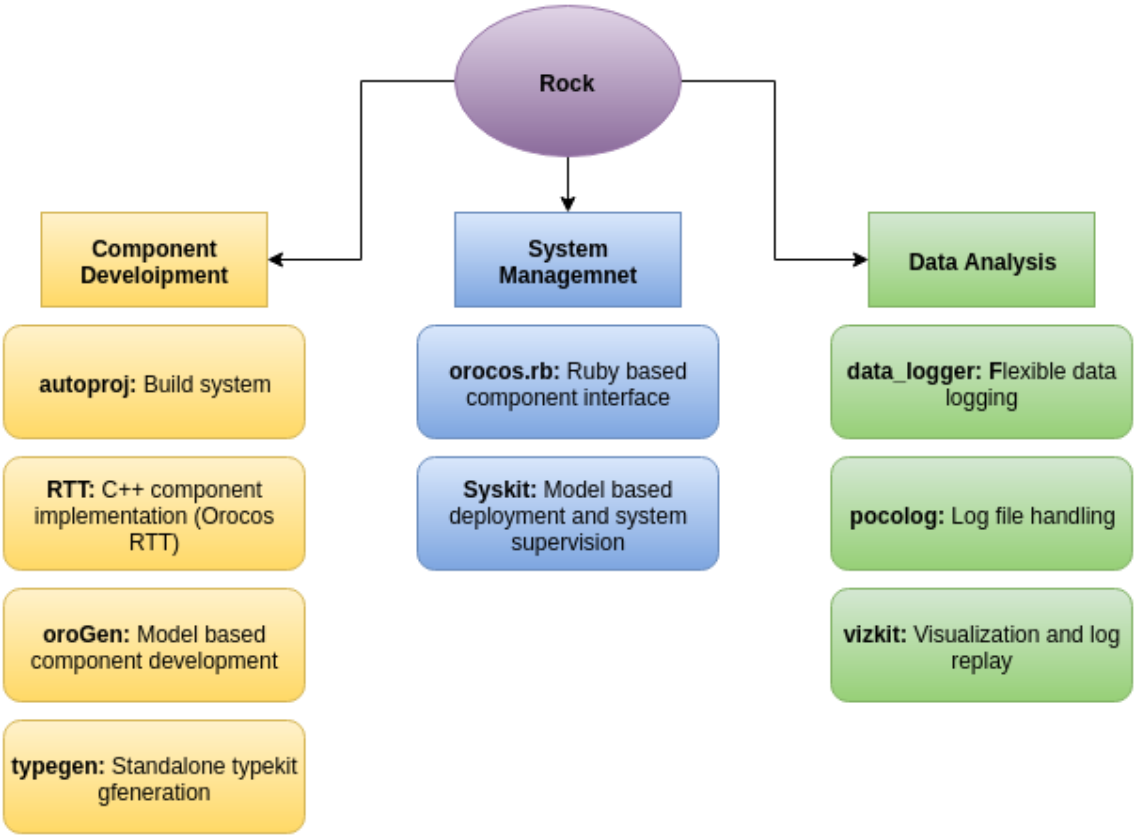


Figure 3.6: Key components of Rock (adapted from Joyeux et al. (2014))

CHAPTER 4

Landmark SLAM

This chapter covers the artificial landmark aided SLAM system developed for this thesis. Section 4.1 describes the design considerations and specifications of the environment modifier used. Section 4.2 deals with real time landmark detection and tracking. Section 4.3 details the range data based SLAM implementation consisting of ICP based front-end and graph optimisation back-end. Section 4.4 explains how to use the detected landmarks in the Graph SLAM implementation and provides an algorithm on how to handle detected landmarks.

4.1 Artificial Landmark Design

In this section we discuss the design considerations for choosing a good artificial landmark for the environment that the rover is used in.

In order to detect our artificial landmarks, we are leveraging the ability of our sensor to detect calibrated reflectivity values of points in our environment. The following factors have to be taken into account in the design or our landmark:

- Surface reflectivity
- Distinguishability: whether or not they should be uniquely identifiable
- Size and Shape

4.1.1 Relative reflectivity

The landmark's surface should be more reflective than most surfaces in our environment of operation. This isn't much of a problem in planetary environments with dull natural structures like sand, rocks and caves. Additionally, reflectivity changes with factors like lighting, environment media, distance and laser angle of incidence; thus the choice of a surface has to have enough reflectivity so as to account and compensate for these variables.

Using a reflective coating or a covering is easier than looking for landmarks with shiny surfaces.

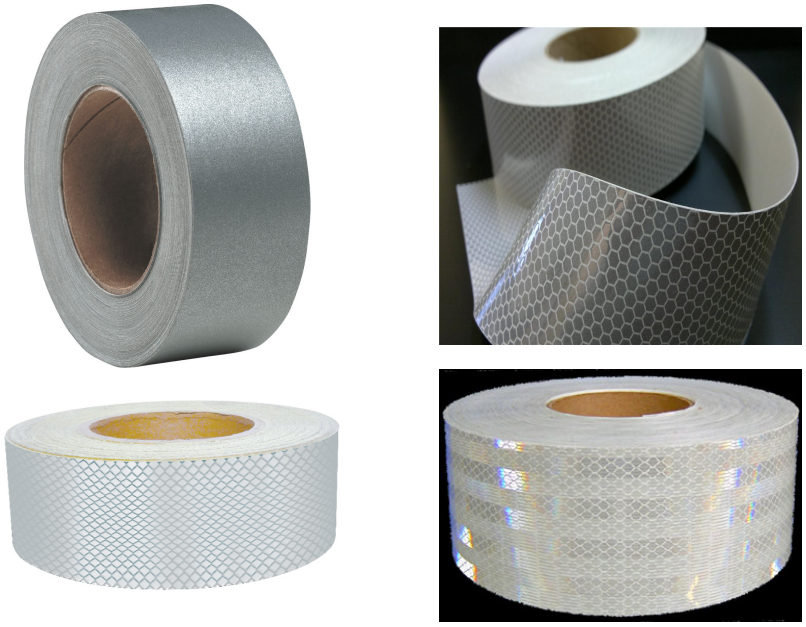


Figure 4.1: Reflective tape types tested for reflectivity return values

This covering can be in the form of reflective paint or tape. After investigating the effectiveness of various reflective tapes (Figure 4.1), we settled for one which returned an average reflectivity of around **75/255** in an environment where most points don't return values higher than **40/255** (Section 3.1.1).

4.1.2 Distinguishability

The landmarks can either be uniquely identifiable or indistinguishable. As discussed in Section 2.3, while unique landmarks are more helpful and make the data association problem simpler, using them can be logistically challenging in terms of storage and deployment. Moreover, colour coding a landmark isn't possible since it necessitates usage of some sort of codes which brings forth problems in detection due to landmark orientation and poses.

Given the above problems with distinguishable landmarks, we use indistinguishable ones which prove to be a lot easier to deploy and detect in the planetary environment.

4.1.3 Size and Shape

As discussed in Section 2.3, our environment modifiers have to be autonomously deployed, so the storage and deployment processes have to be made easier and shielded from possible complications. This can be done by considering the effects of landmark shape and size on logistics, while maximizing visibility and detectability.

Spherical surfaces have the advantage over planar surfaces, where the curvature increases the probability of laser reflectance given the varying sensor orientation and the laser's angle of inci-

Figure 4.2: Artificial landmarks - The first and the third are covered with tape while the middle ball is coated with reflective paint.

dence on landmark surface. Spheres also have the advantage over other regular curved objects like the cylinder when it comes to ease of deployment.

Since the lasers are divergent, there is a bound on the maximum number of points on the surface of our modifiers (landmarks) that can reflect back these lasers. This maximum steadily decreases as the sensor distance from the landmark increases. In our tests with the 16-channel VLP-16 and the third ball in Figure 4.2, we got a maximum of 15 reflected points when the ball was at a distance of 1 metre, all the way up to only 2 points when it was 5 metres away. The ball was not detectable beyond 5 metres.

Spherical artificial landmarks with dimensions in the range of 3 to 5 cm diameter are considered for use in this thesis (Figure 4.2).

4.2 Landmark Detector

This section describes the components used for the landmark detector algorithm. The algorithm is developed in indoor environments using the 16-channel Velodyne VLP-16 before testing on the Artemis robot.

The subsequent subsections describe the components of the landmark detector system. Note that the below methods are not just developed for a reduced cloud consisting of points with high reflectivity values. For generality, they are tested on normal point clouds before adapting them for our case use.

Figure 4.3 shows the pipeline of the landmark detector.

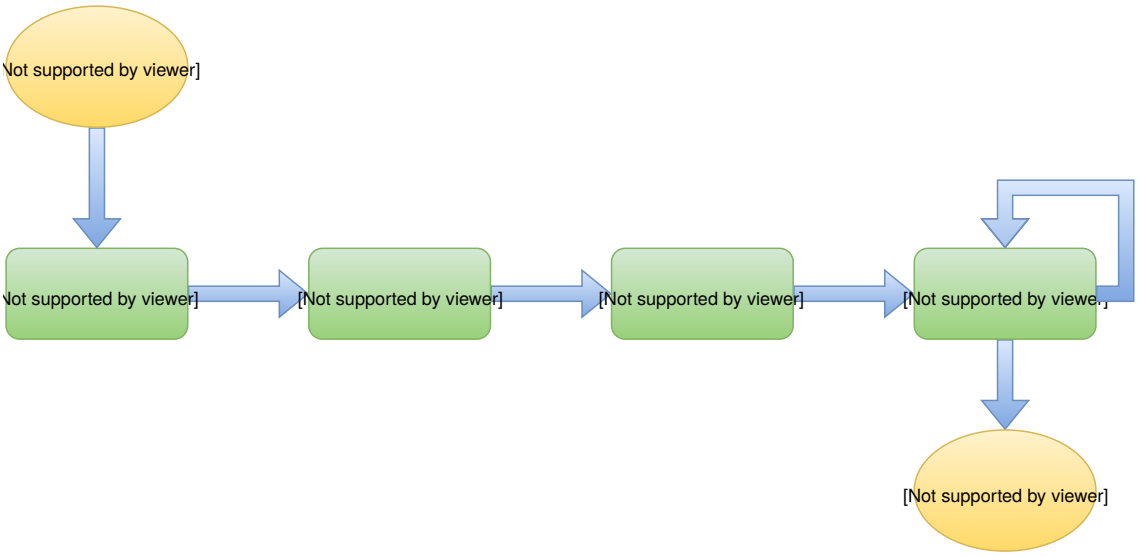


Figure 4.3: Landmark detection pipeline

4.2.1 Thresholding

Thresholding helps reject all points in the cloud which have a lower reflectivity than a predetermined value. The threshold is variable and depends on the distance of a point from the sensor. After repeated tests, an optimal threshold-distance relation (equation 4.1) is developed.

Let there be a point $P = \{x, y, z, I\}$, where x, y and z are its Cartesian coordinates and I is its *normalized* calibrated reflectivity value, that is, $I \in [0.0, 1.0]$. Then the threshold value t for point P is governed by

$$t = t_0 - \frac{D}{40} \tag{4.1}$$

where t_0 is the base threshold value and D is the distance of a point from the sensor. The value of t_0 was determined during tests and is found to be in the range of 0.38 to 0.42. This value holds true for the 32-channel HDL-32E LiDAR as well.

$$D = \sqrt{x^2 + y^2 + z^2}$$

Given the size of the landmarks, the maximum distance they can be detected is around 10metres. So we will not be searching for landmarks in distances ($D > 16metres$) where equation 4.1 would return a negative threshold t.

Simple thresholding of point cloud reflectivity values to detect our reflective landmarks isn't effective. As shown in Figure 4.4, during tests in an indoor environment, we observed a lot of outliers, noise and high reflectivity of multiple planar points like on walls.

4.2.2 Planar Segmentation

The point cloud that has reflectivity values higher than the threshold (white points in Figure 4.4), are seen to mostly lie on walls. We also notice that these undesirable points have reflectivity values similar to the points on our landmark's surface, so stricter thresholding is not the solution. We require a way to reject all points lying on the wall by exploiting the fact that they are on the same plane.

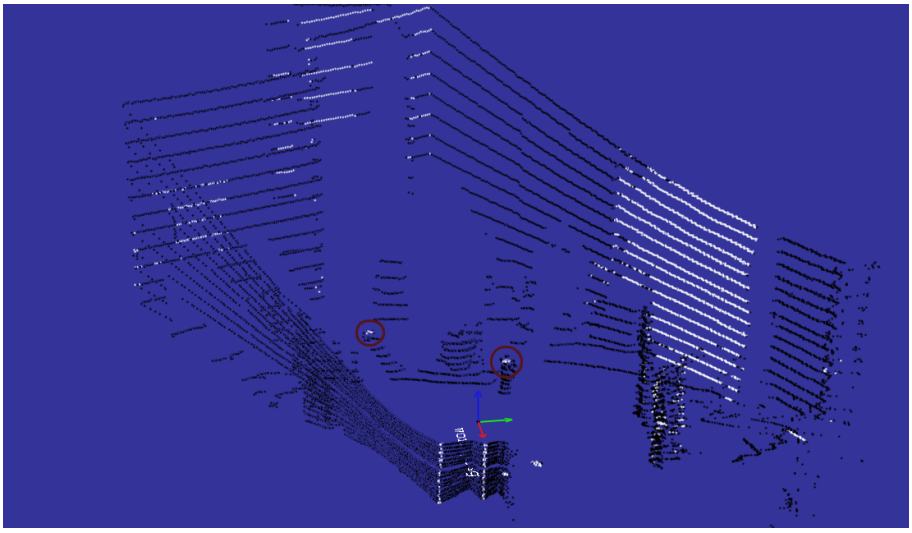
Removing points lying on a planar surface requires a way to segment the point cloud by grouping together points lying on the same 2D surface. We use a segmentation procedure based on the Random Sample Consensus (RANSAC, Fischler and Bolles (1981)) algorithm. The algorithm takes set of observed data values, a parametrized model which can explain or be fitted to the observations, and some confidence parameters as *inputs*. It then performs the following in order:

1. Selects a ransom sample set of points from the input point cloud.
2. Computes a model of a plane given those points
3. Counts the number of points lying on that plane from the global set; these are the *inliers*.
4. Repeats the above process until it finds a model which returns the maximum number of inliers.
5. Tags the indices of the points in this subset of the cloud and calls it the largest plane.
6. Removes the largest plane from the original point cloud.
7. Repeats the above process until the points in the original point cloud goes below a predetermined threshold.

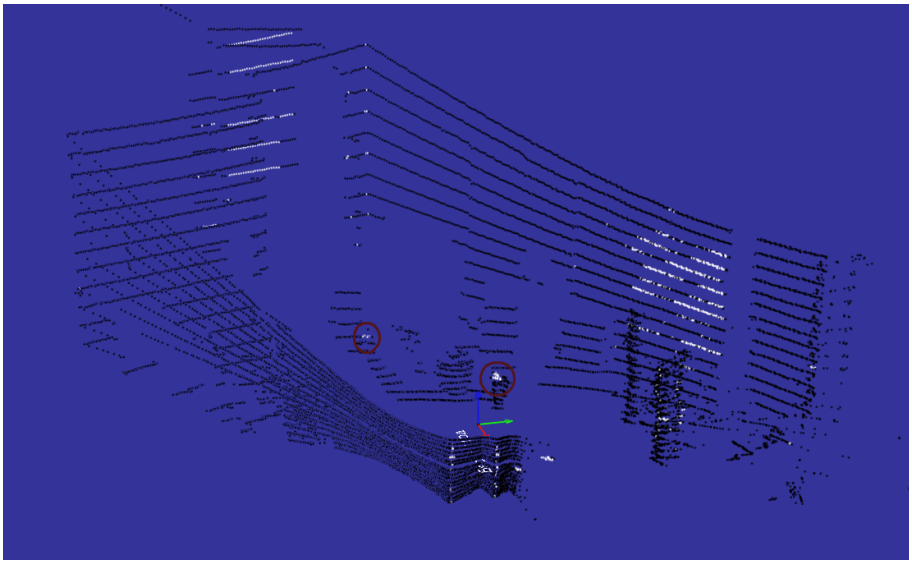
Figure 4.5 shows the effect of planar segmentation on a general point cloud. The cloud to the right shows five different planes extracted from the left. Note that this step is only developed so as to aid the indoor development of the algorithm and it isn't required in outdoor detection of landmarks. This is because a rover wouldn't be expected to encounter smooth plane surfaces in planetary conditions.

4.2.3 Clustering

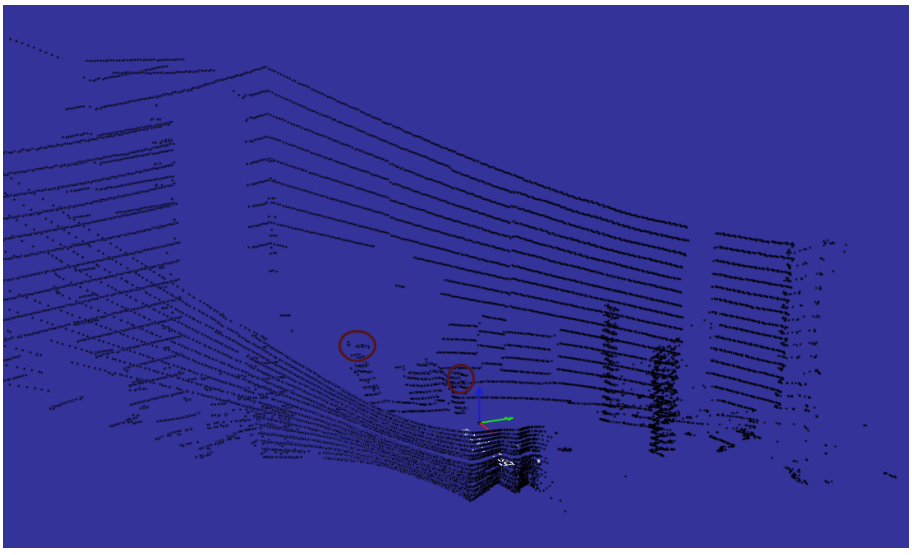
The planar segmentation method above works well for the majority of points on planes. However, as seen in Figure 4.6, it fails to extract all points on the plane and does not deal with noise or points not on any plane. Fig 4.6b shows that we still have points on the walls after removing the 35 largest planes from the original cloud. We need another layer of pre-processing in order to be able



(a) Threshold = 76

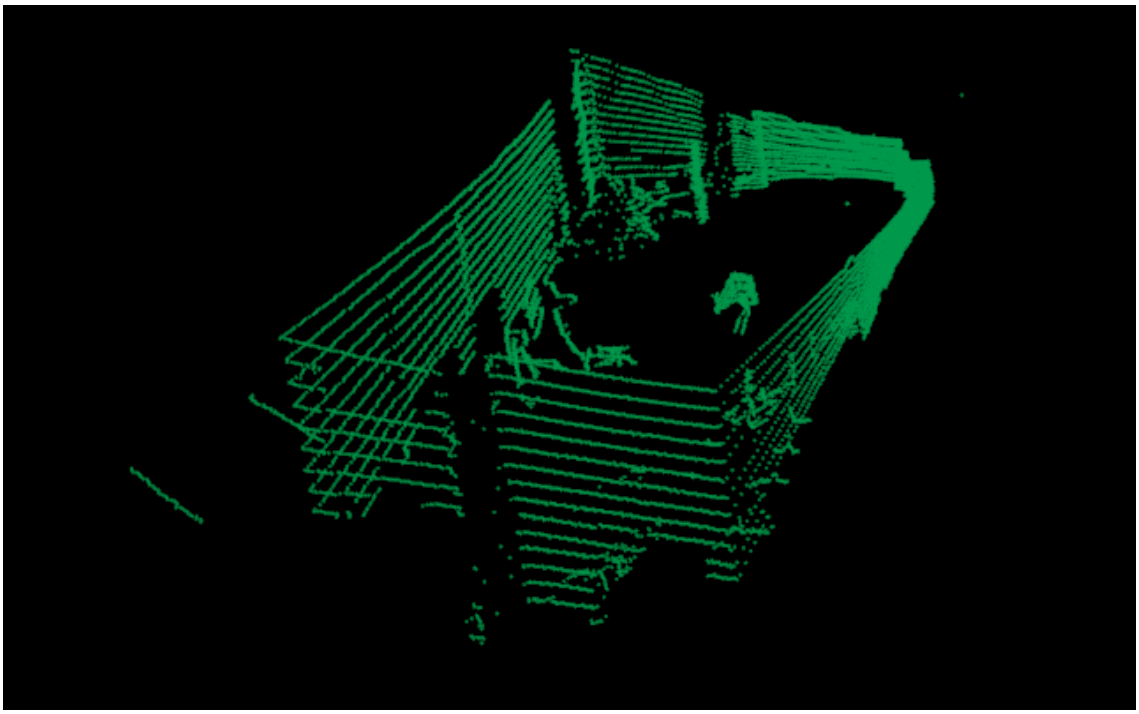


(b) Threshold = 97

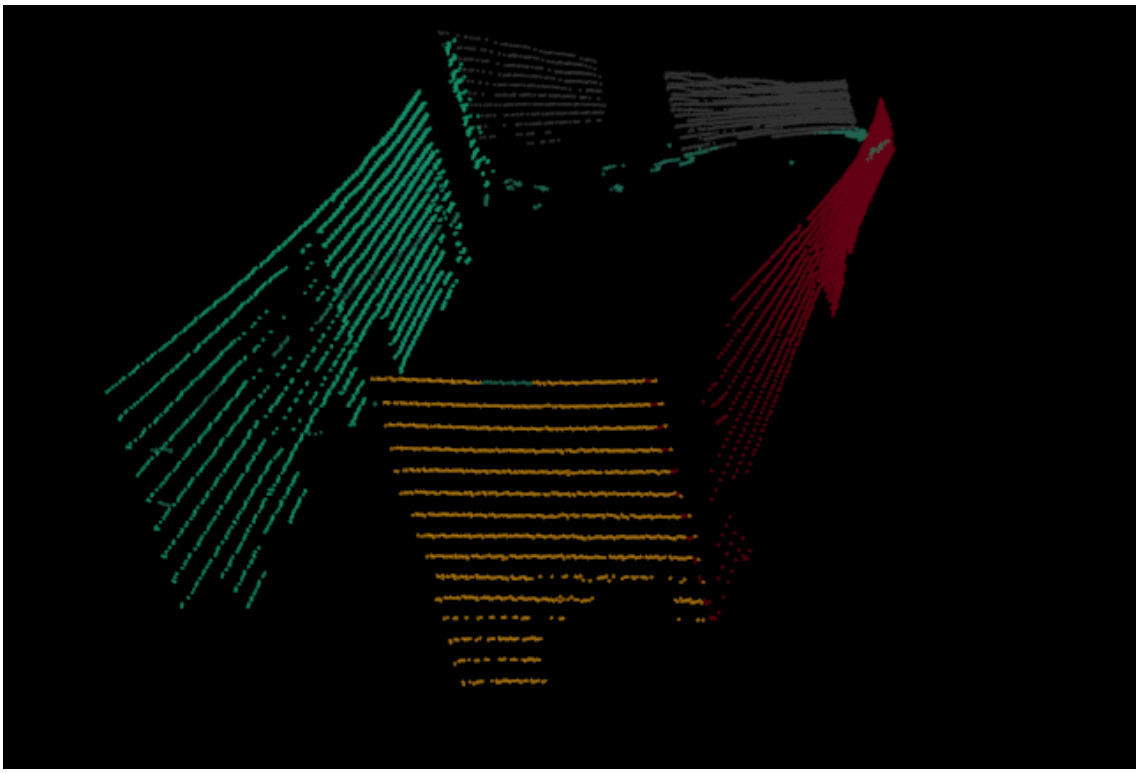


(c) Threshold = 102

Figure 4.4: Effects of pure thresholding of reflectivity values on detection the landmarks (marked in red ellipses). This figure shows that the noise and outliers have roughly the same reflectivity values as our landmarks, since upon increasing it from 97 to 102 we can't see either. The reflectivity values are on a scale of 0 to 255

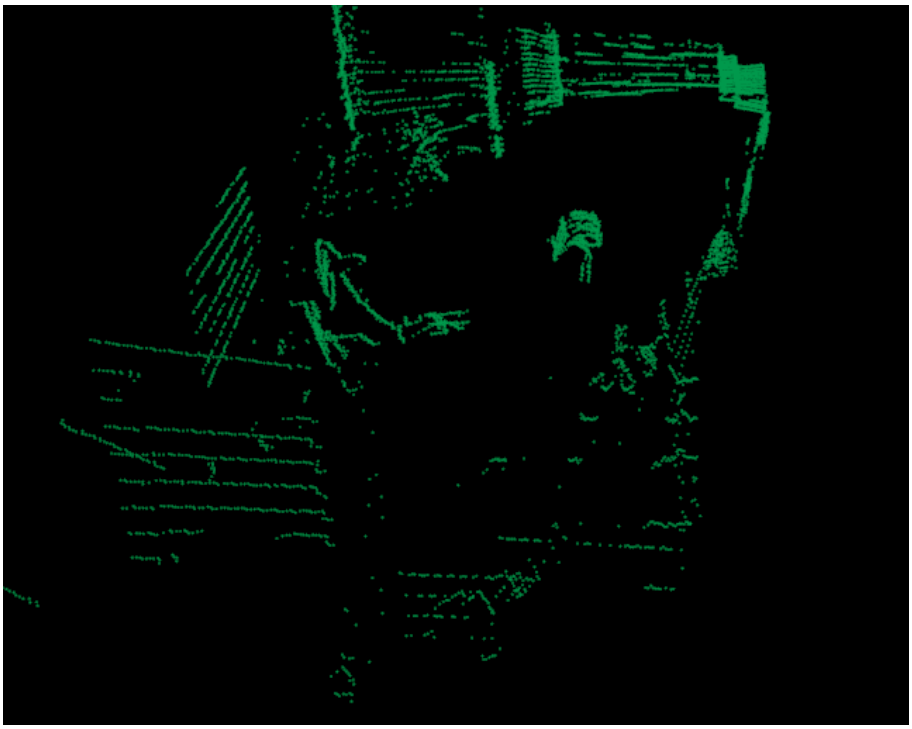


(a) Original point cloud

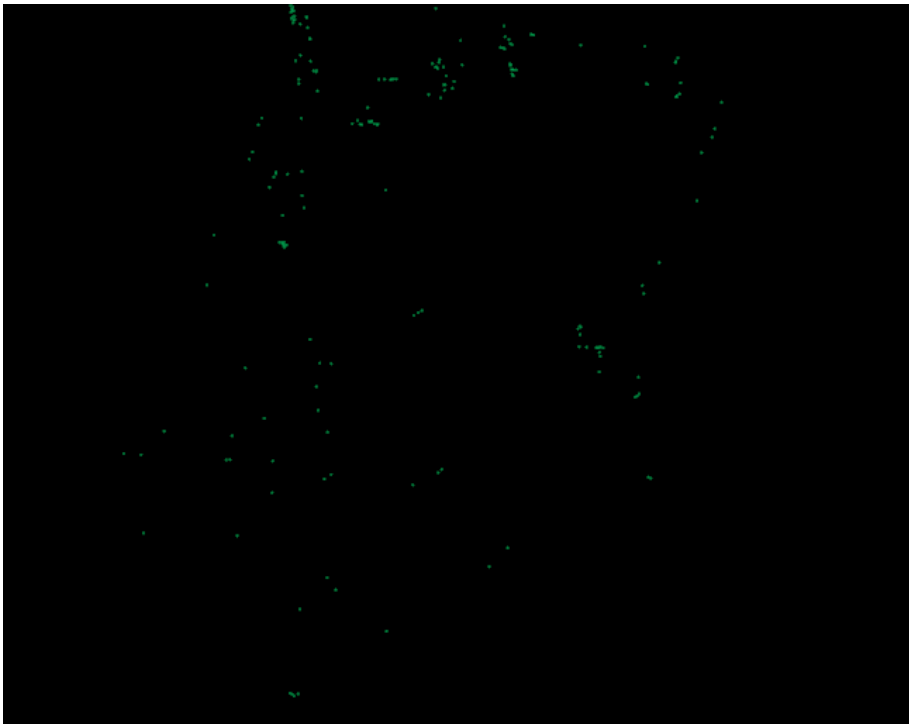


(b) Point cloud with five extracted planes

Figure 4.5: Planar Segmentation using RANSAC



(a) Point cloud after removing 5 largest planes



(b) Point cloud after removing 35 largest planes

Figure 4.6: Remaining points after removing 5 and 35 largest planes from the point cloud in Figure 4.5a

to detect our landmarks, and we take advantage of the fact that the noisy points remaining after planar segmentation are sparse and spread out.

Clustering is the operation where points are grouped together based on some similarity criteria. For us, this similarity criteria is Euclidean distances between points. We use an efficient form of clustering based on kd-tree search to group points which are close together.

The kd-tree (k-dimensional tree) is a data structure used for range and nearest neighbour searches. Since our search space is 3D space, we will be using the 3 dimensional kd-tree.

The clustering algorithm can be tweaked using the following parameters

1. Cluster tolerance: The distance around a point within which another point would be considered its neighbour. The higher this value is, the greater is the chance that distinct landmarks close to each other would be perceived as one. This value has to be greater than the accuracy of the sensor, that is 2cm for the VLP-16 and 3cm for HDL-32E (Table 3.1).

2. Minimum cluster size: This is the minimum number of points required in a vicinity to be classified as a cluster.

3. Maximum cluster size: This is the maximum number of points required in a vicinity to be classified as a cluster.

The values of maximum and minumum cluster sizes should be set keeping in mind the maximum points reflected off from the landmark surfaces are various distances from the sensor, as discussed in Section 4.1.3. Figure 4.7 shows clustering of a point cloud (from Figure 4.5a) after 35 iterations of plane segmentation and removal. The tolerance is 3 cm, and the minumum and maximum cluster sizes are 5 and 10 respectively.

Clustering is accompanied by the calculation of a representative point or a centroid for each cluster. This is simply the mean of the x,y and z coordinates of each point in the cluster.

Let L be the points in a detected cluster.

$$L = \{c_i\}$$

where c_i are the points in the cluster. Let C be the cluster centroid.

$$C = \frac{1}{n}\Big(\sum_{i}^{n} c_i\Big) \tag{4.2}$$

The centroid points are then added to the filtered point cloud and used as an input for polling and landmark SLAM modules.

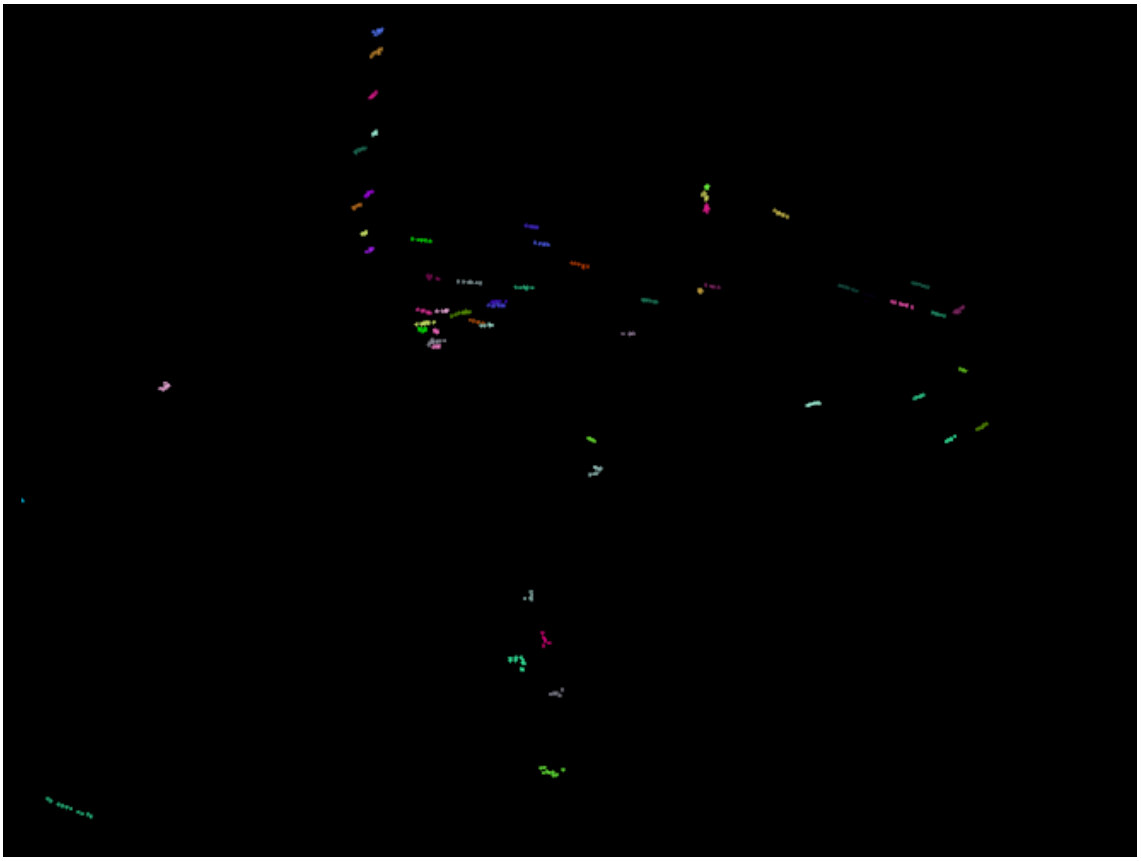


Figure 4.7: Clustering: 77 clusters identified after 35 largest planes were removed from the point cloud in Figure 4.5a

4.2.4 Polling

Thresholding, planar segmentation and clustering provide agreeable results in an indoor environment when the sensor is not in motion as shown in Figure 4.8. When the sensor is in motion however, we notice that landmarks are not detected consistently. We would obtain a centroid in one point cloud and it might disappear in the next one, only to reappear in the subsequent one. This is observed even when the sensor is moving with a low speed (around 5 km/h). Moreover, there might be some misidentified clusters, especially in the outer region of the scan. This is usually because the thresholding layer allows for less lenient filtering of reflectivity values, the further a point is from the sensor (Equation 4.1).

To remedy this, we add another layer which tracks each detected landmark centroid through 3 or more frames (a frame is a 360°scan), and distinguishes between real landmarks and noise. Based on how consistently a landmark is detected, it will also consider a detection even though the landmark might not be visible during a frame. Algorithm 1 shows how this was achieved. Here $past$ is a vector of vector of centroid positions of landmarks detected in the past n frames.

The result of the polling algorithm is the 3D position of a landmark l_k. Let C_k be the centroid of the cluster of landmark l_k which has survived the polling round by consistently being tracked for a set number of frames.

Algorithm 1 Polling

```
procedure TRACKANDFILTER(past, pointCloud)
    past ← structure of vectors of centroids in past n frames
    n ← size of past
    centroid ∈ [x, y, z, match]
    if n > 2 then
        for all i from 1 to n do
            for all centroid1 from past[i].begin to past[i].end do
                for all centroid2 from past[i-1].begin to past[i-1].end do
                    if distance (centroid1, centroid2) > 50cm then
                        centroid1[match] += 1
                    end if
                end for
                if i > 1 AND centroid1[match] == 0 then
                    for j from i to 1 do
                        for centroid3 from past[i-2].begin to past[i-1].end do
                            if distance (centroid1, centroid2) > 50cm +(50cm ∗ (i − j)) then
                                centroid1[match] += 1
                            end if
                            if centroid1[match] > 0 then
                                break
                            end if
                        end for
                    end for
                end if
                for centroid4 from past[n − 1].begin to past[n − 1].end do
                    if centroid4[match] > 2 then
                        pointCloud ← centroid4
                    end if
                end for
                past ← pop out first vector
            end for
        end for
    end if
end procedure
```

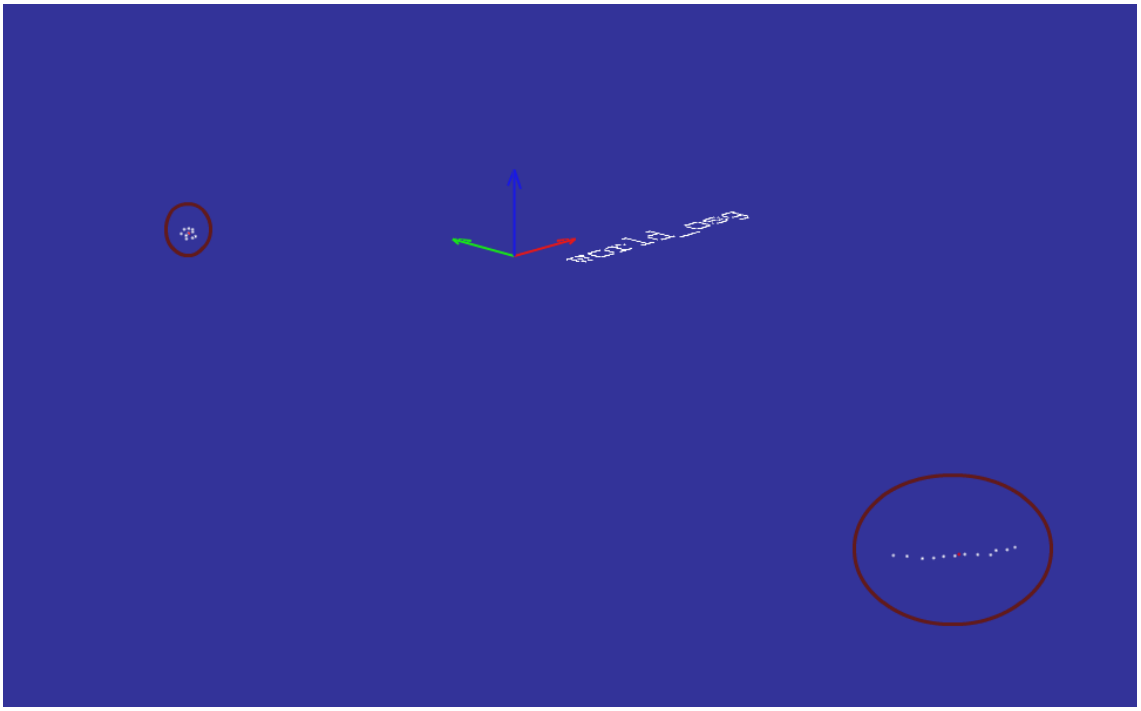


Figure 4.8: The remaining cloud after thresholding, removing planar noise and clustering clearly detects the two landmarks (marked by the red ellipses). The red points represents the centroids of the two clusters.

$$C_k = [x, y, z] \tag{4.3}$$

4.3 SLAM system

This section details the working of the SLAM system. Section 4.4 will look into using landmarks to get better results from this system.

As introduced in Section 2.3, the existing SLAM implementation is decoupled into two main stages: **Front-end** and **Back-end** SLAM.

4.3.1 Front-end SLAM

The front-end is dependent on sensor data and deals with aligning raw data and constructing a pose-constraint representation. This representation is called a graph and it visualizes the spatial constraints between robot poses modelled in the form of a dynamic Bayesian network. The constraints are encoded as edges of the graph and the graph nodes represent the robot poses that result from scene observations as shown in Figure 4.9.

In Figure 4.9, the graph edge consists in a probability distribution over the transformation between two poses (or nodes). This observed transformation (T_{ab}) is calculated by aligning the point cloud

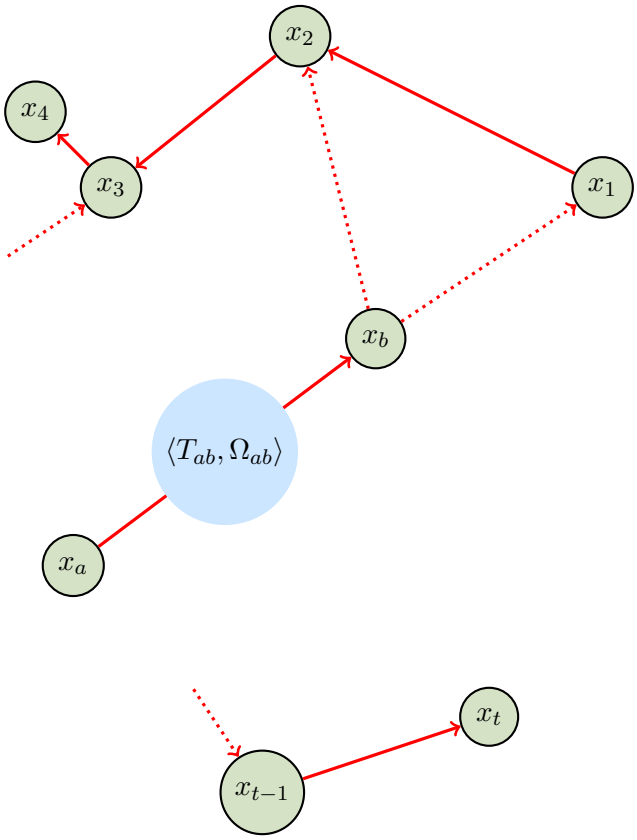


Figure 4.9: A graph representation showing nodes representing robot poses $x_1, x_2..x_{t-1}$ and a constraint between two poses x_a and x_b. Constraints between consecutive nodes are shown by solid red arrows. These constraints (T_{ab}) are a result of sensor observations, and is obtained by the ICP front-end. The dotted arrows arise due to multiple observations of the same part of the environment and are obtained using Breadth First Search in the vicinity of each pose. They are a result of the spatial constraints of the graph. (adapted from (Grisetti et al., 2010))

data acquired at two robot poses using the Generalized Iterative Closest Point algorithm (GICP, as introduced in Section 2.2). A graph is constructed using the results of ICP and associations due to repeated observations of the same part of the environment. These associations (dotted lines in 4.9) are a result of the spatial constraints in the graph.

The transformation estimation by scan alignment and subsequent graph constructions constitutes the bulk of the front-end SLAM of our system.

Generalized ICP

Consider observations made by the LiDAR at two robot poses x_a and x_b in the form of two point clouds.

Let the two point clouds be

$$A = \{a_i\} \qquad B = \{b_i\} \tag{4.4}$$

Let $\mathbf{T}$ be the final transformation which aligns A and B. Using odometry measurements, we also have an initial transformation $\mathbf{T}_0$. So ideally,

$$b_i = \mathbf{T}a_i \tag{4.5}$$

Through the following algorithm we hope to make (T_0) converge to $\mathbf{T}$.

Starting from $\mathbf{T}_0$, we iterate through the point cloud A to calculate $(\mathbf{T} \cdot a_i)$. We then find the closest point in B to $(\mathbf{T}_0 \cdot a_i)$ using 3D Euclidean distance to form a corresponding pair a_i and b_i.

Additionally, we eliminate pairs of points (a_k and b_k), if

$$||b_k - \mathbf{T} \cdot a_k|| \geq d \tag{4.6}$$

where d is the *matching threshold*, which is the distance between two closest points in the two point clouds above which they are not considered corresponding. This accounts for the possibility that some points in one scan will not have a matching point in the second cloud due to reasons like being outside the cloud boundary. This value is a trade off between accuracy and convergence since a large value would lead to incorrect correspondences, while a low value would cause a bad convergence.

The resulting point clouds contain only points which have a match in the other point cloud. The following equation should make $\mathbf{T}$ converge to the correct transformation between the two point clouds.

$$\mathbf{T} = \underset{\mathbf{T}}{argmin}\{\sum_i ||\mathbf{T} \cdot a_i - b_i||^2\} \tag{4.7}$$

We now use probabilistic models to solve the minimization step in Equation 4.7. Let's consider an alternate probabilistic model of the two point clouds A, B and the points inside of them.

$$\hat{A} = \{\hat{a}_i\} \qquad\qquad \hat{B} = \{\hat{b}_i\}$$

where $\hat{a}_1, \hat{a}_2 \cdots \hat{a}_n$ and $\hat{b}_1, \hat{b}_2 \cdots \hat{b}_n$ represent the Gaussian mean of the location of points in point cloud $\hat{A}$ and $\hat{B}$ respectively.

$$\hat{a}_i \sim \mathcal{N}(\hat{a}_i,\, C_i^A) \qquad\qquad \hat{b}_i \sim \mathcal{N}(\hat{b}_i,\, C_i^B)$$

here, $\{C_i^A\}$ and $\{C_i^B\}$ are covariance matrices associated with the measured points. Assuming geometrically consistent correspondences with no occlusion or sampling errors,

$$\hat{b}_i = \mathbf{T}^* \cdot \hat{a}_i \tag{4.8}$$

where $\mathbf{T}^*$ is an assumed correct transformation for the probabilistic model. We define a distribution $d_i^{(\mathbf{T}^*)}$ which is derived from Equations 4.5 and 4.8

$$\begin{aligned} d_i^{(\mathbf{T}^*)} &\sim \mathcal{N}(\hat{b}_i - (\mathbf{T}^*)\hat{a}_i,\, C_i^B + \mathbf{T}^* C_i^A (\mathbf{T}^*)^T) \\ &= \mathcal{N}(0,\, C_i^B + \mathbf{T}^* C_i^A (\mathbf{T}^*)^T) \end{aligned}$$

By applying Maximum Likelihood Estimation (Wikipedia, 2016; Segal et al., 2009), we iteratively compute $\mathbf{T}$:

$$\mathbf{T} = \underset{\mathbf{T}}{argmax} \prod_i \log(p(d_i^{(\mathbf{T})}))$$

where $(p(d_i^{(\mathbf{T})}))$ is the probability of the distribution. This can be simplified to

$$\mathbf{T} = \underset{\mathbf{T}}{argmin} \sum_i d_i^{(\mathbf{T})^T} (C_i^B + \mathbf{T} C_i^A \mathbf{T}^T)^{-1} d_i^{\mathbf{T}} \tag{4.9}$$

The $\mathbf{T}$ obtained here is used to construct the graph which is then forwarded to the SLAM back-end. The advantage of Generalized ICP used here is that by selecting different structures of C_i^A and C_i^B, we can incorporate the information about local structure of both scans to form better correspondences. For example, to implement a plane-to-plane version, from (Segal et al., 2009), set

$$C_i^A = \mathbf{R}_{n_a} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{n_a}^T \tag{4.10}$$

$$C_i^B = \mathbf{R}_{n_b} \cdot \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{n_b}^T \tag{4.11}$$

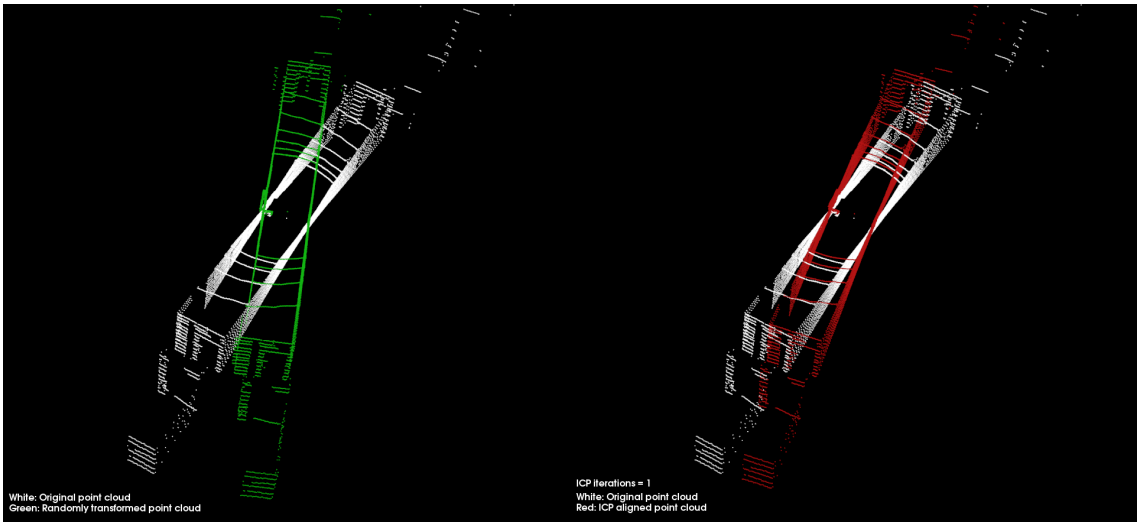
where n_a and n_b are normal vectors at a_i and b_i respectively, the 3×3 matrix and ϵ represent the covariance matrix of a point whose surface normal is e; and $\mathbf{R}_x$ is the rotation matrix from basis vector e to x.

Figures 4.10a and 4.10b show alignment of a point cloud in a structured indoor environment with a randomly rotated version of itself. Figures 4.10d and 4.10c show the same for another indoor environment for double the number of ICP iterations.

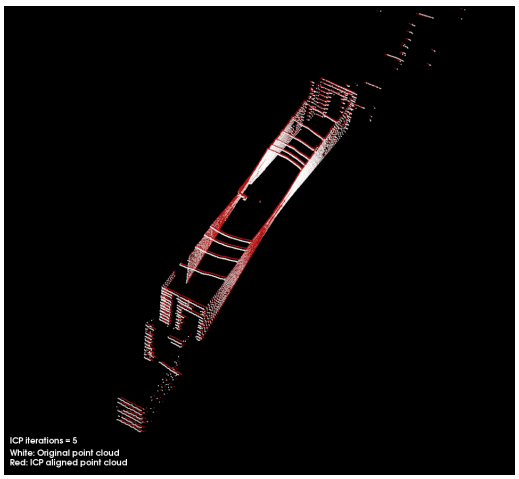
Data Association

Data association is primarily done using the plane-to-plane directive as introduced above for point matching. Essentially, the point cloud is assumed to have more structure than an arbitrary set of 3D points in the sense that points are assumed to be part of a plane.

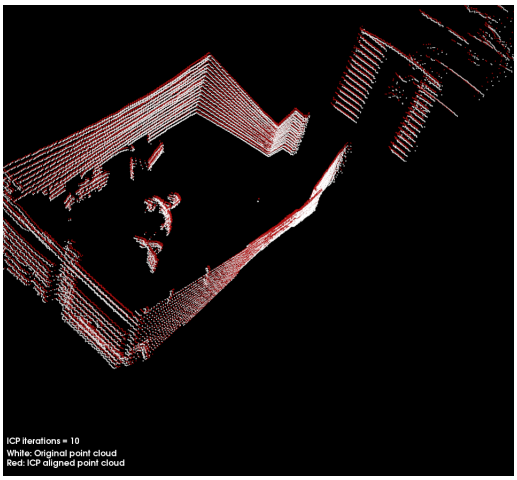
The point cloud is assumed to be locally planar and points are distributed assuming they have a high covariance in the local plane and very low covariance in the direction of the surface normal at



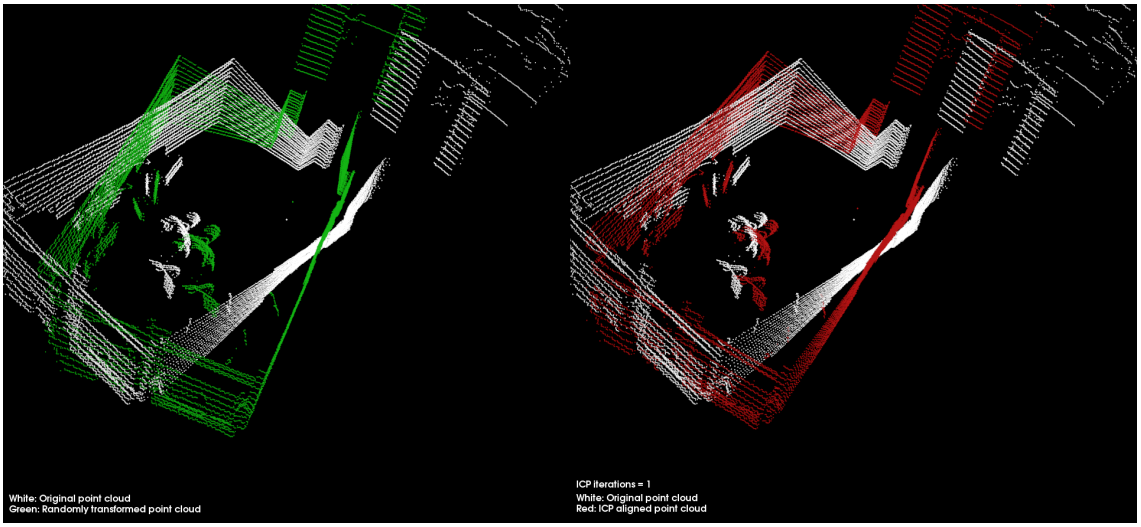
(a) Corridor: Red is after 1 iteration



(b) Corridor: Red is after 5 iterations



(c) Room: Red is after 10 iterations



(d) Room: Red is after 1 iteration

Figure 4.10: Point cloud alignment using ICP in different indoor environments. *White* is the original point cloud, which is randomly rotated to form the *green* point cloud and *red* is the point cloud after n iterations of ICP.

that point. The equations 4.10 and 4.11 describe this covariance.

Additionally, every time a robot pose is registered on the graph, the algorithm looks for nearby poses using breadth first search using a neighbourhood distance criteria. Upon finding a nearby pose, the graph invokes ICP between that and the new pose to determine the transformation and create what is known as an indirect edge (Section 4.4). This helps in establishing constraints between non-consecutive robot poses and proves for better graph optimisation.

Evaluation Criteria

The validity of an ICP estimate can be quantified through a *Euclidean fitness score*. A fitness score (f) is the normalized sum of squared distances between corresponding points in the source and target point clouds and ranges between 0 to 1. A score of 1 means perfect alignment. The transformation **T** obtained from equation 4.9 is used to construct the edges in the graph for consecutive nodes. This forms the observed transformation which is then made globally consistent by the graph optimisation back-end.

Configuration parameters

Our algorithm and experiments use the following parameters to tweak and fit the functionality of ICP to the requirements of a particular task

- Maximum correspondence distance (d): This is the maximum distance between two points in two point clouds to be considered corresponding (Equation 4.6).
- Maximum fitness score: This is the maximum score ICP aims for during iterations (Section 4.3.1).
- Maximum iterations: The number of ICP alignment iterations to perform until the maximum fitness score is achieved.
- Transformation epsilon: Maximum allowable difference between two consecutive lateral transformations for convergence.
- Rotation epsilon: Maximum allowable difference between two consecutive rotations for convergence.
- Correspondence randomness: Number of point neighbours when selecting zones for computing covariances.

  The values for the above parameters depend upon the application requirements and a trade-off is usually made between performance and speed of the alignment.

4.3.2 Back-end SLAM

We now intend to compute a Gaussian approximation of the posterior over the rover's trajectory. Finding the configuration of nodes that maximizes the likelihood of pose observations results in us

obtaining a mean and subsequently the information matrix of the Gaussian. This can be categorised as a constraint maximization problem.

Let

$$\mathbf{T}_{ab}(x_a, x_b) = \mathbf{T}$$

be the **observed** transformation between the rover poses x_a and x_b, and Ω_{ab} its information matrix, as obtained from our ICP algorithm (Equation 4.9). Subsequently, let $\hat{\mathbf{T}}_{ab}$ be the expected mean and information matrix of the transformation between rover poses x_a and x_b.

The error function will be the difference between the expected observation $\hat{\mathbf{T}}_{ab}$ and the real observation $\mathbf{T}_{ab}$ of the transformation between the rover pose x_a and x_b

$$e_{ab}(x) = \mathbf{T}_{ab} - \hat{\mathbf{T}}_{ab}(x_a, x_b)$$

To find the maximum likelihood, we define the log-likelihood ((Grisetti et al., 2010)) l_{ab} of $\mathbf{T}_{ab}$ as,

$$l_{ab} \propto e_{ab}(x_a, x_b)^T \Omega_{ab} e_{ab}(x_a, x_b)$$

The goal here is to find a configuration of poses x^* that minimizes the negative log-likelihood function $\mathbf{F}(x)$ of all observations

$$\begin{aligned} \mathbf{F}(x) &= \sum_{\langle a,b\rangle \in C} e_{ab}^T(x)\, \Omega\, e_{ab}(x) \\ &= \sum_{\langle a,b\rangle \in C} \mathbf{F}_{ab}(x) \end{aligned} \tag{4.12}$$

Essentially, the configuration of nodes x^* is

$$x^* = \underset{x}{argmin}\, \mathbf{F}(x) \tag{4.13}$$

Gauss-Newton

We use the Gauss-Newton algorithm to obtain the solution of Equation 4.13 (like in Grisetti et al. (2010)). We start by approximating the error function (e_{ab}) using first order Taylor expansion and an initial guess of rover poses $\tilde{x}$ with increments of Δx_a and Δx_b.

$$\begin{aligned} e_{ab}(\tilde{x}_a + \Delta x_a,\, \tilde{x}_b + \Delta x_b) &= e_{ab}(\tilde{x} + \Delta x) \\ &\simeq e_{ab}(\tilde{x}) + \mathbf{J}_{ab}\Delta x \\ &\simeq e_{ab} + \mathbf{J}_{ab}\Delta x \end{aligned} \tag{4.14}$$

Here, $\mathbf{J}_{ab}$ is the Jacobian of $e_{ab}(x)$ in $\tilde{x}$. From equations 4.12 and 4.14

$$\begin{aligned} \mathbf{F}_{ab}(\tilde{x} + \Delta x) &= e_{ab}(\tilde{x} + \Delta x)^T\, \Omega_{ab}\, e_{ab}(\tilde{x} + \Delta x) \\ &\simeq (e_{ab} + \mathbf{J}_{ab}\Delta x)^T\, \Omega_{ab}\, (e_{ab} + \mathbf{J}_{ab}\Delta x) \\ &= e_{ab}^T \Omega e_{ab} + 2 e_{ab}^T \Omega_{ab} \mathbf{J}_{ab}\, \Delta x + \Delta x^T \mathbf{J}_{ab}^T \Omega_{ab} \mathbf{J}_{ab}\, \Delta x \\ &= \mathbf{c}_{ab} + 2\mathbf{b}_{ab}\Delta x + \Delta x^T \mathbf{H}_{ab} \Delta x \end{aligned} \tag{4.15}$$

where,

$$c_{ab} = e_{ab}^T \Omega e_{ab} \tag{4.16}$$

$$b_{ab} = e_{ab}^T \Omega_{ab} \mathbf{J}_{ab} \tag{4.17}$$

$$\mathbf{H}_{ab} = \mathbf{J}_{ab}^T \Omega_{ab} \mathbf{J}_{ab} \tag{4.18}$$

To generalise the negative log-likelihood over all poses in x, from Equation 4.12,

$$\begin{aligned} \mathbf{F}(\tilde{x} + \Delta x) &= \sum_{\langle a,b \rangle \in C} \mathbf{F}_{ab}(\tilde{x} + \Delta x) \\ &\simeq \sum_{\langle a,b \rangle \in C} \mathbf{c}_{ab} + 2\mathbf{b}_{ab}\Delta x + \Delta x^T \mathbf{H}_{ab} \Delta x \\ &= \mathbf{c} + 2\,\mathbf{b}\,\Delta x + \Delta x^T\,\mathbf{H}\,\Delta x \end{aligned} \tag{4.19}$$

Since $\mathbf{H} = \sum \mathbf{H}_{ab}$ is obtained by using Jacobians to project measurement errors in the space of path trajectory, it is the *information matrix* of the system. It has a sparse structure and contains non-zero elements between nodes connected by a constraint. The pose increments Δx^* for a configuration of nodes x^*, can be obtained by solving the linear system:

$$\mathbf{H}\,\Delta x^* = -\mathbf{b} \tag{4.20}$$

We now solve the linear Equation 4.20 using sparse Cholesky factorization to obtain the increment Δx^*. This is then added to the initial guess to obtain the optimal configuration of nodes that minimizes the negative log likelihood in Equation 4.12 .

$$x^* = \tilde{x} + \Delta x^* \tag{4.21}$$

We use the graph optimisation software module from Kümmerle et al. (2011) called *g2o*. Our implementation runs about 100 iterations of pose graph optimisation for the current graph configuration every time the back-end is invoked. To reduce processing overhead, we only invoke the back-end after a certain number of ICP iterations. This number is called the optimisation rate ($optimizationRate$).

4.3.3 Coupling

Concluding from the last two sections, our SLAM pipeline is as shown in Figure 5.3. We run the ICP front-end for every scan by the LiDAR and obtain a transformation $\mathbf{T}_{ab}(x_a, x_b)$ between two rover poses x_a adn x_b. Using this, we construct a graph (Figure 4.9) by adding vertexes to it in the form of the two robot poses and an edge between them. After this, we search for nearest pre-existing poses (using breadth first search) in the graph and construct additional edges. This constitutes the front-end.

Then we invoke the graph optimisation back-end for every few number of scans ($optimisationRate$) to obtain a better configuration of our poses (x^*) within the constraints. This continues until the robot has finished acquiring scans and we then use our extensive graph to create the pose map.

The next section introduces a landmark based method to facilitate this process.

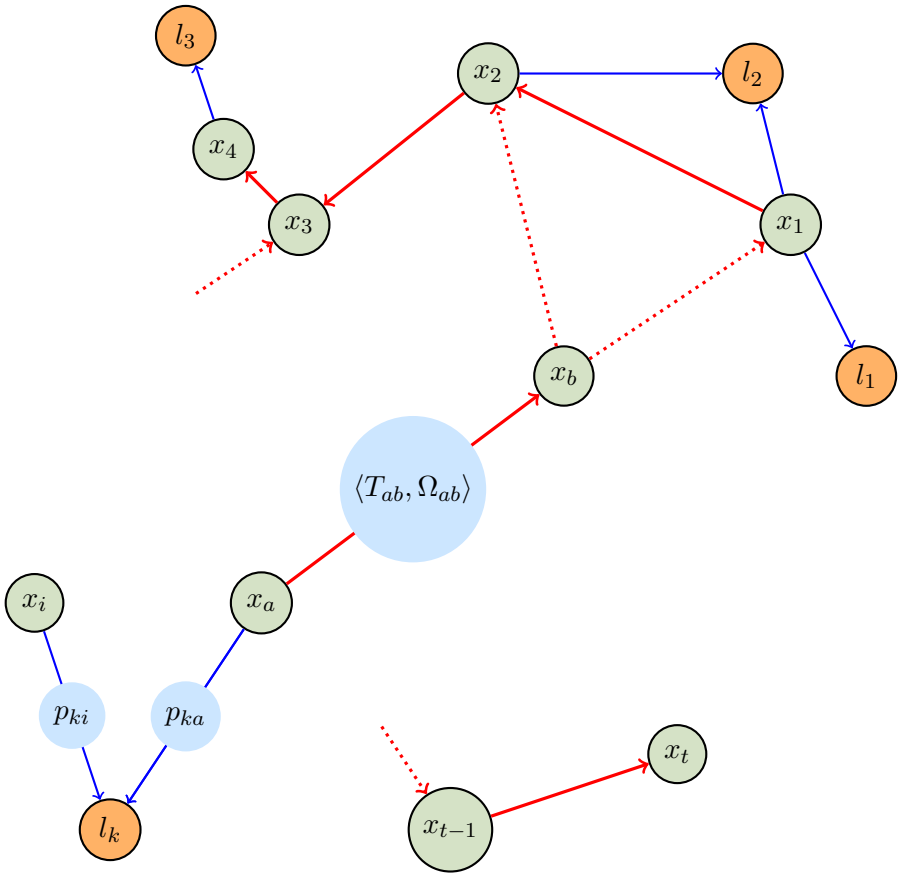


Figure 4.11: Graph representation with artificial landmarks

4.4 Landmark SLAM

We propose altering the graph construction algorithm to include landmark poses as nodes to the graph. We call them secondary nodes.

Figure 4.11 is an extension of Figure 4.9. Using landmarks we append the graph with additional nodes $l_1, l_2, ...l_n$. Here, p_{ka} is the position of landmark l_k in the frame of x_a. This means p_{ka} is the output of the landmark detector when it saw l_k from robot pose x_a. From equation 4.3,

$$p_{ka} = [x, y, z] \tag{4.22}$$

We define and make a distinction between direct and indirect edges in the graph of Figure 4.11. Direct (solid red lines) are edges between consecutive robot poses. They are formed as a direct result of ICP alignments. Every robot pose (except the first and the last) has two direct edges. Indirect (dotted red lines) edges are formed as a result of neighbourhood searches from a pose. They are a means of recognising nearby poses and data association.

Whenever the same landmark is detected by the rover in two different poses x_a and x_i, if the relationship between the two poses meets the criteria defined in the list below, we can calculate an estimated transformation T_{ai}.

The following are the steps to process a landmark once it is detected.

1. Landmark l_k detected from pose x_a.
2. Store landmark centroid in p_{ka}.
3. Landmarks l_k and l_j are now detected from pose x_i.
4. Store landmark centroids in p_{ki} and p_{ji} respectively.
5. If no edge between x_i and x_a exists, break and return.
6. If edge between x_i and x_a exists, check the type of the edge they share. If it's an indirect edge, replace the T_{ai} with one calculated from p_{ka} and p_{ki}.
7. If the edge is direct, check the fitness score of the ICP.
8. If the fitness score is below a threshold value, replace the T_{ai} with one calculated from p_{ka} and p_{ki}
9. Invoke graph optimisation.

Calculation of the transform T_{ai} makes use of the fact that initial estimates of the two robot poses exist (from when they were formed during ICP).

The transformation estimates calculated from this method are given a lower priority to the ones calculated from several ICP iterations. This is why the ICP fitness score is taken into account before using the calculated estimate.

The algorithm 2 summarizes the landmark handling by our graph optimisation back-end.

In the next Chapter, many tests are conducted on real world data with a robot which validates our front-end, back-end and landmark SLAM system.

Algorithm 2 Landmark extension to SLAM system

```
procedure BALLDETECTED(Pose x_i, ballPos p_ki, neighbourPoses X )
    for all x_a ∈ X do
        if p_ka exists then
            if (x_i, x_a) → indirectEdge then
                T_ai ← f(p_ki and p_ka)
                graphOptimize()
            end if
            if (x_i, x_a) → directEdge then
                if (x_i, x_a).fitnessScore()<threshold then
                    T_ai ← f(p_ki and p_ka)
                    graphOptimize()
                else
                    break
                end if
            end if
        end if
    end for
end procedure
```

CHAPTER 5

Experiments and Results

This section briefly describes the test set-up and results obtained from the mapping algorithm. Section 5.1 introduces a map visualization method. Section 5.2 details the experimental arrangement including the problem task, constraints and the test area. Section 5.3 shows the results of various outdoor experiments with the Artemis robot and Section 5.4 summarizes and discusses these results.

5.1 Visualization: Multi-level Surface maps

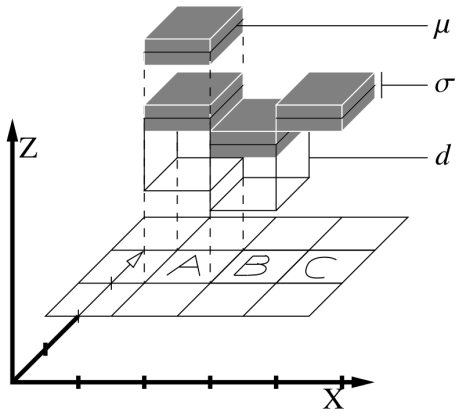
We use multi-level surface maps (MLS, Triebel et al. (2006)) as a map representation in this section. This is a grid based map where each cell in the grid can have multiple surface patches (Figure 5.1a). Patches have their positions expressed in a Gaussian distribution, and each patch can has a mean μ (its position) and a variance σ (its thickness). Furthermore, a patch can also have depth d, depending on whether the object surface it represents is flat or not.

Figure 5.1c shows an MLS map created by our algorithm on moving the VLP-16 sensor in a corridor (Figure 5.1b). The shades of red show uncertainty in a patch, green shows vertical and blue horizontal cells.

5.2 The Test Area

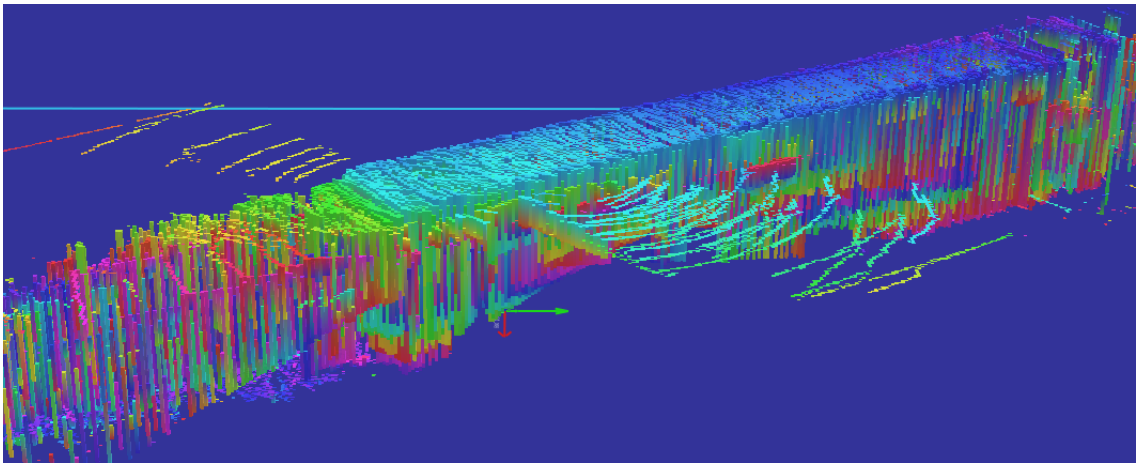
In order to emulate a planetary scene, one of the primary requirements for our test area was for it to be relatively free of features. Finding such places in a city is hard, and we used an empty plot (Figure 5.2) of land for this purpose. Since there were streets and buildings around it, we had to resort to pre-processing methods like filtering out points too far from the sensor to 'blind' the rover from those features. In this way, we were able to approximately simulate the conditions on a planetary surface.

The Artemis rover (4.1.3) is used to carry out test runs on the ground in Figure 5.2a. The rover was retrofitted with wheels meant for traversing hard ground as shown in figure 5.2d. The tests involve moving the rover around various landmark configurations, which were placed above bricks (5.2b) to avoid being occluded by the sparse ground vegetation.



(a) MLS structure (Reproduced: Triebel et al. (2006))

(b) The sensor travels 5 metres in this corridor



(c) MLS map of a corridor

Figure 5.1: Map Visualization

5.3 Experiments

This section shows the results of navigation and mapping by our robot in various outdoor experiments. The tests are conducted in different types of outdoor environments to highlight the problem and the propose a solution.

5.3.1 Navigation without Landmarks

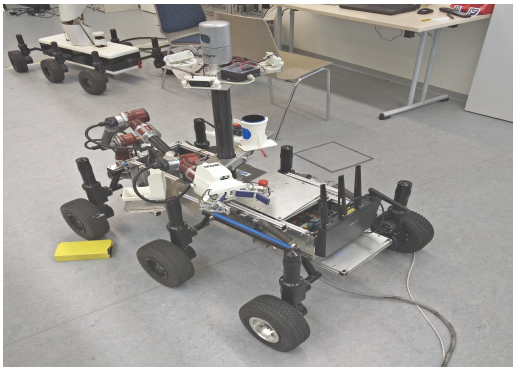
Process Flow

Figure 5.3 shows a functional flowchart for the processes involved in navigation without landmark.

(a) The test area

(b) Landmark placement

(c) Buildings, trees and cars in the vicinity



(d) The Robot: Retrofitted Artemis

Figure 5.2: Test Set-up

Experiment in an Urban Environment

We will discuss the results obtained by performing SLAM (front-end and back-end) on a 140m path in an urban environment, using only the Velodyne HDL-32E mounted on Artemis.

Figure 5.4a shows that this route is inundated with buildings, cars and trees, thus representing a typical urban scene. Using just the Velodyne sensor, our algorithm should perform well without artificial landmarks.

Figure 5.4b shows that this indeed is the case. The SLAM front-end managed to align the point clouds, while the pose graph errors seem to be minimized by the graph optimisation back-end as well. The straight red line shows the rover's trajectory, and the MLS map of the route looks consistent.

Experiments in the Test Area

This subsection will discuss the results of mapping the rover's path across our test area (Figure 5.2a). We test under different trajectories with increasingly limited robot perception (filtering out the point cloud by setting d_{max}).

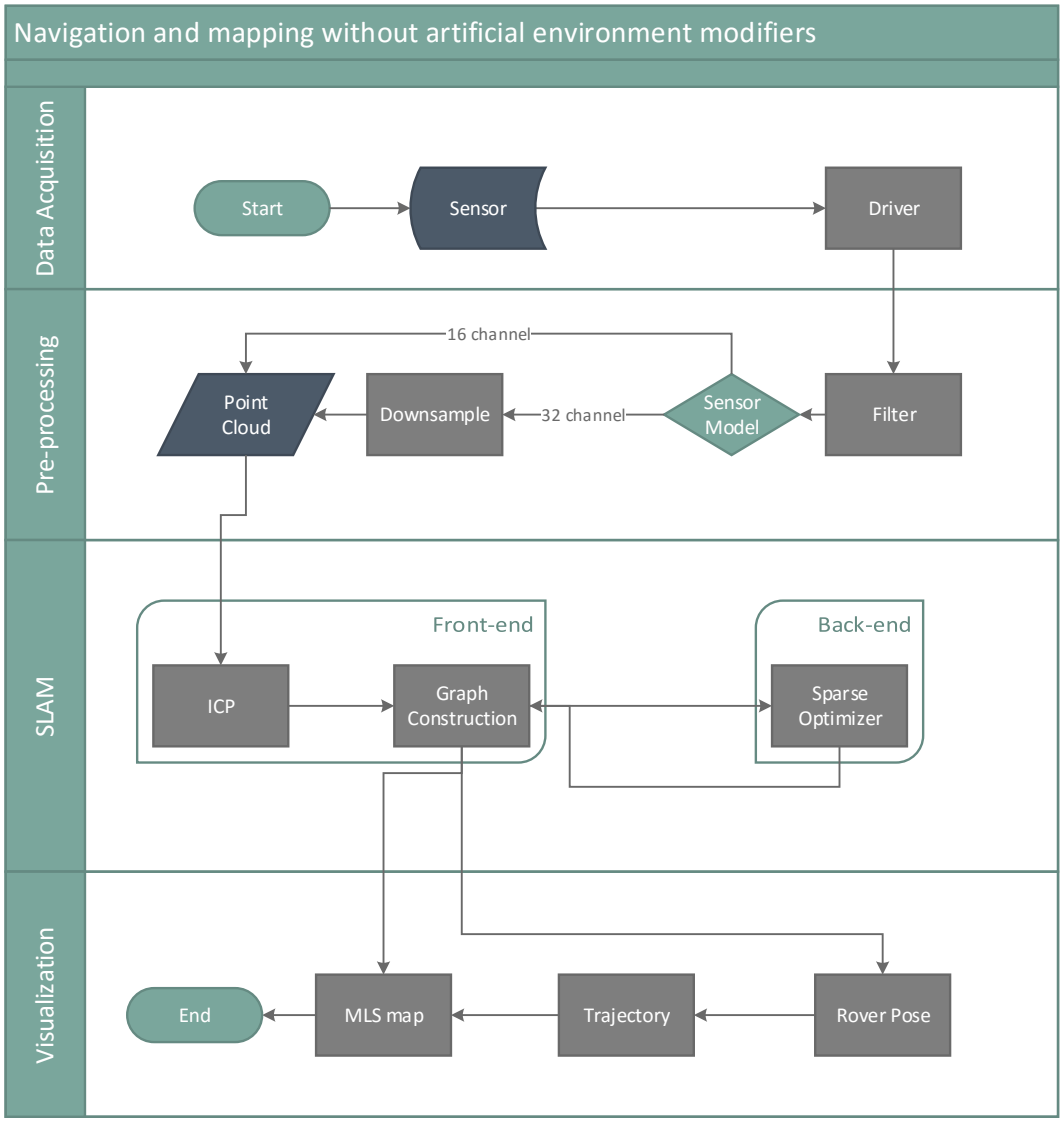
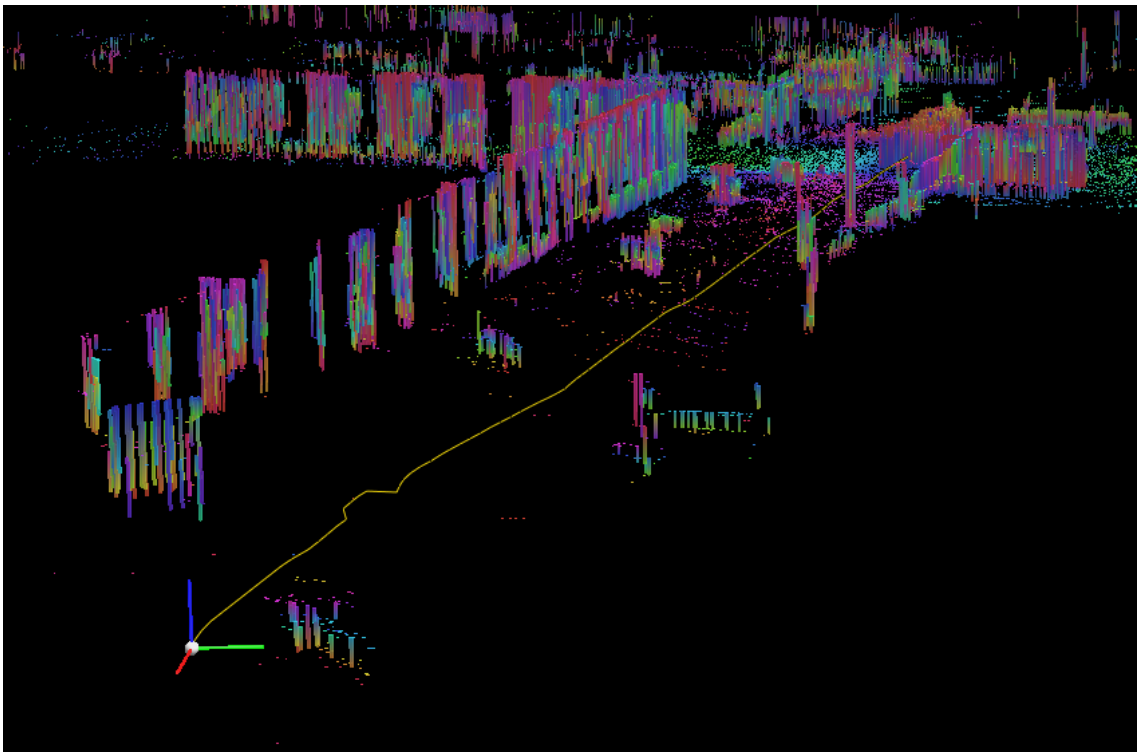


Figure 5.3: Flowchart for navigation and mapping without landmarks

Given our landmark's size, the following tasks are carried out in an area of 15×15 metres of ground. The experiments involve localizing, plotting trajectory and mapping the environment in different robot routes anad for varying artificial landmark configurations. For the first test (Task 2), we control the robot to move in the path shown in Figure 5.5b.

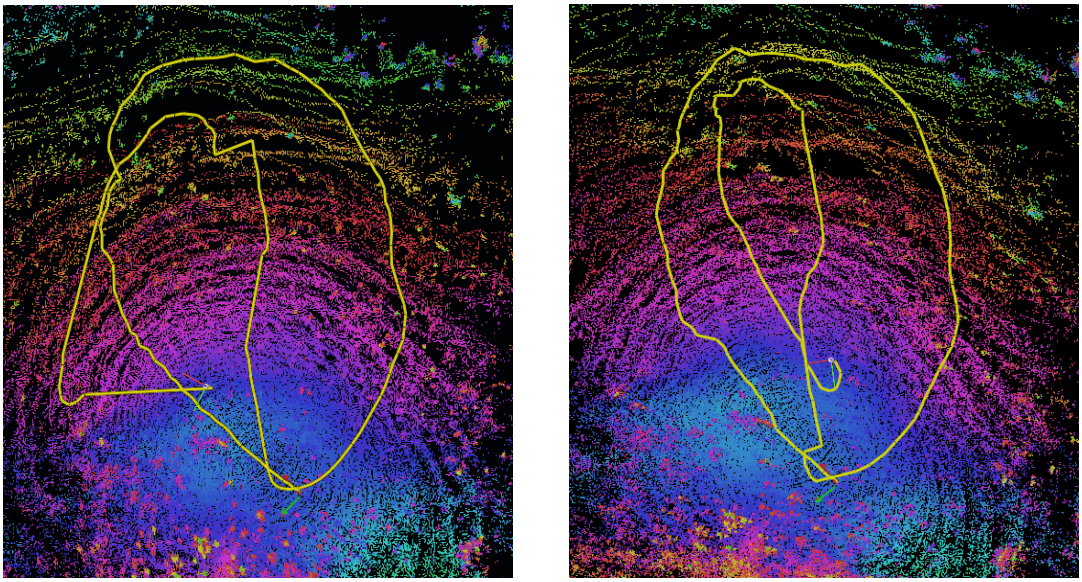
We show results of the task using only the ICP (figure 5.5a), and using both the front-end and the back-end (Figure 5.5b). Here we have not filtered out long range points (lidar range $d_{max} = 75m$) and features like cars, buildings and pedestrians provide enough structure for good ICP alignment. The less error-prone ICP then produces sound estimates for the graph optimisation back-end. Here, back-end runs after every 10 scans, or 10 iterations of the ICP front-end ($optimsationRate = 5$).

(a) The urban test path



(b) Task 1: MLS map along with the robot's trajectory and final pose

Figure 5.4: Test: Urban Environment with full LiDAR range



(a) Only front-end **(b)** Front-end and back-end

Figure 5.5: Front-end and back-end: Robot start pose, trajectory and MLS map at final pose for Task 2 ($d_{max} = 75m$). Both are not using landmarks.

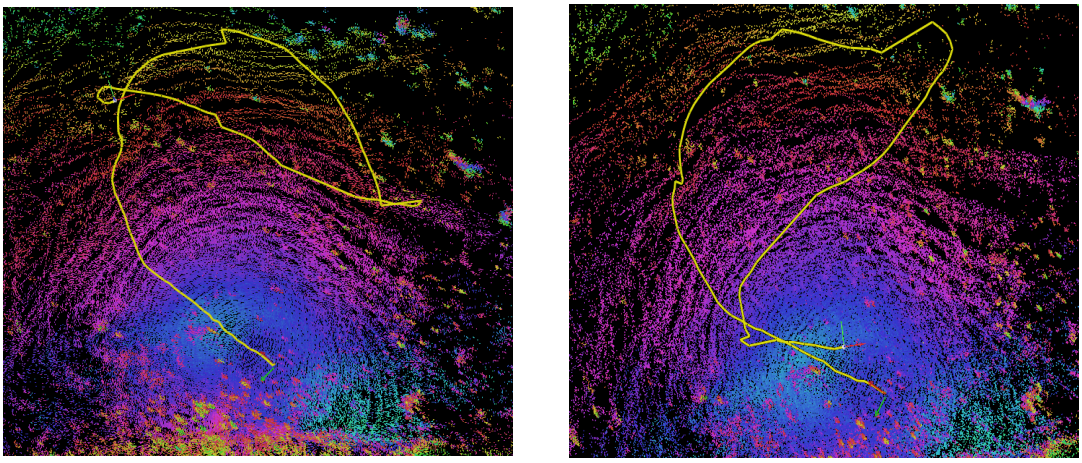
Because of this, the Figure 5.5b will be used as the benchmark against which SLAM with landmarks will be compared in Task 2.

Apart from some sparse vegetation, the test area has structured features like buildings and cars in its environment (as shown in Figure 5.2c). Figure 5.6a shows the same navigation task where points at a distance (d_{max}) greater than 10 metres from the sensor are filtered out. This simulates a planetary surface because of the lack of features in the immediate vicinity of the sensor mounted on our rover.

However, at some poses in the rover's trajectory, features like cars and the street fall within these 10 metres, so Figure 5.6b shows the results of our algorithm with $d_{max} = 5m$ for the same navigation task. These results are obtained by running front-end as well as back-end SLAM. The graph optimisation back-end is run every 10 iterations of ICP and adding nodes to the graph.

5.3.2 Navigation with Landmarks

Five landmarks (figure 5.2b) are placed randomly along the path of the robot in different configurations. The rover then completes certain navigation tasks around them and we present the results in the sections ahead. It's worth noting that for outdoor experiments, the planar segmentation and rejection module of the landmark detector (Section 4.2.2) was not used. This is because of the lack of smooth planes in the outdoor environment which could throw noisy reflectivity values to confuse the detector.



(a) $d_{max} = 10m$ **(b)** $d_{max} = 5m$

Figure 5.6: Limited Range: Task 2 (Not using landmarks)

Process Flow

Figure 5.7 shows a functional flowchart for the processes involved in navigation with landmarks. This is the extension to the flowchart in figure 5.3.

Experiments in the Test Area

Landmarks are placed in various configurations and the results are shown below. The tests are completed for $d_{max} = 10m$ instead of $5m$. This is because in the latter case, the rover isn't able to detect landmarks in most cases since the landmarks are small (and require multiple reflected points for detection), and the sensor is at a height of $1.5m$ on the robot. This obviates the advantage of our landmark SLAM extension (Section 4.4) and returns results similar to Figure 5.6b.

The graph optimisation back-end is invoked in the following cases

- After every 10 ICP iterations.
- If the landmark extension changes a transformation (Algorithm 1.2).

Because of the lack of a GPS module on the test rover, we do not have a ground truth for the robot's trajectory. Even with a GPS module, getting a 10 metre accuracy would have been difficult. Hence we will attempt to compare our landmark SLAM results with those obtained without landmarks but with a large sensor range of $d_{max} = 75m$ (Figure5.5b). We have superimposed landmark positions on that figure, which were arbitrarily determined as shown in Figure 5.6a.

Figure 5.8d shows that most landmarks were detected in a 3 metre radius around the sensor. This is mostly because of the size of our landmarks and the divergent lasers from the scanner. Figure 5.8b shows a marked improvement of the rover localization as compared to the non-landmark result in figure 5.6a for $d_{max} = 10m$.

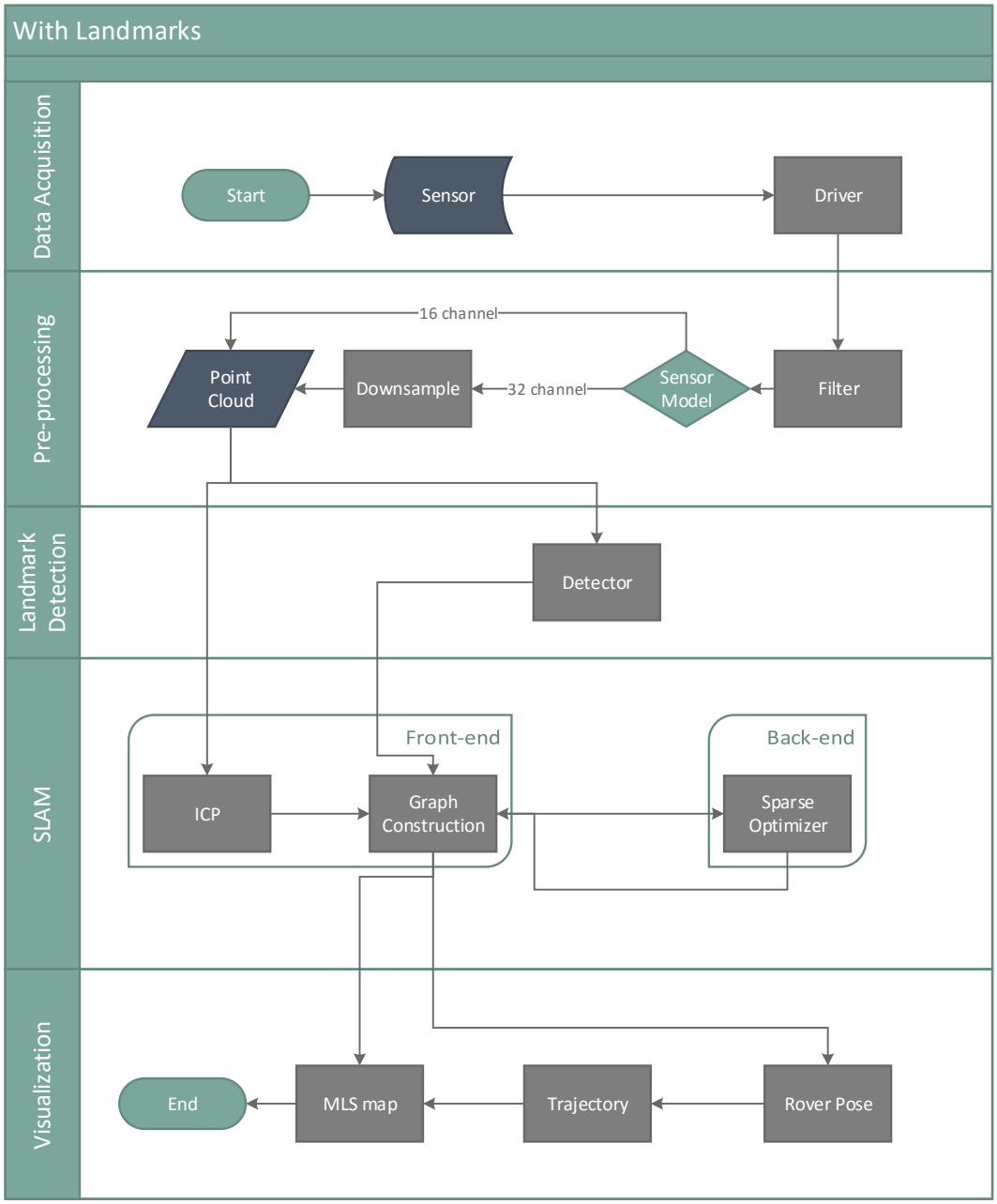
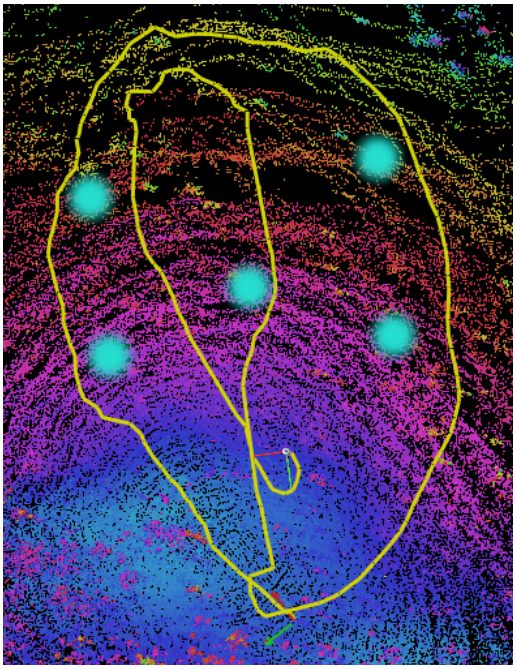
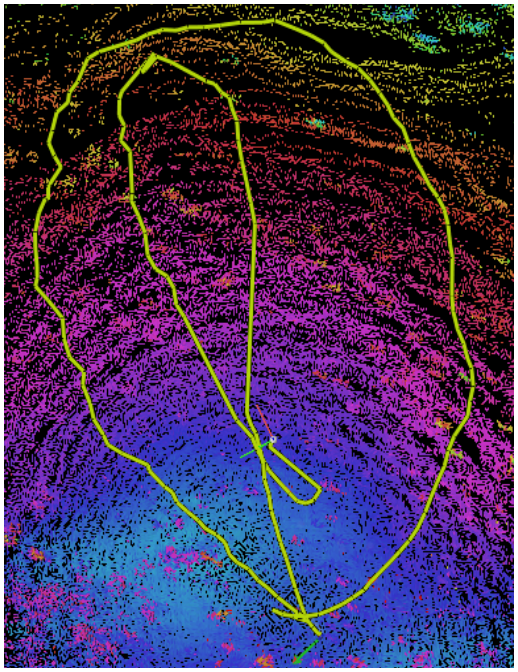


Figure 5.7: Flowchart for navigation and mapping without landmarks

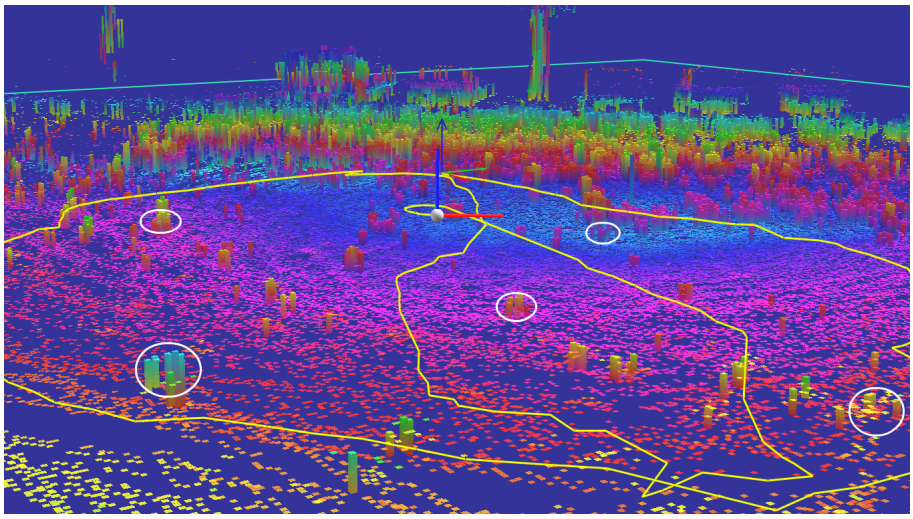
For Task 3, the landmark configuration is superimposed on landmark-less SLAM results with a long sensor range of $d_{max} = 75m$ and shown in Figure 5.9a. This will form our basis of comparison since the abundant structural features in the distance (d > 15m) prove for a good ICP alignment. This particular task doesn't contain sharp turns, or tight loops and the alignment results for $d_{max} = 10m$ are very similar for navigation with and without landmarks as shown in Figures 5.9b and 5.9c. These results seem mostly consistent with our benchmark figure 5.9a, although SLAM with landmarks fits it better. Figure 5.9a shows the landmarks detected (white circles) on an



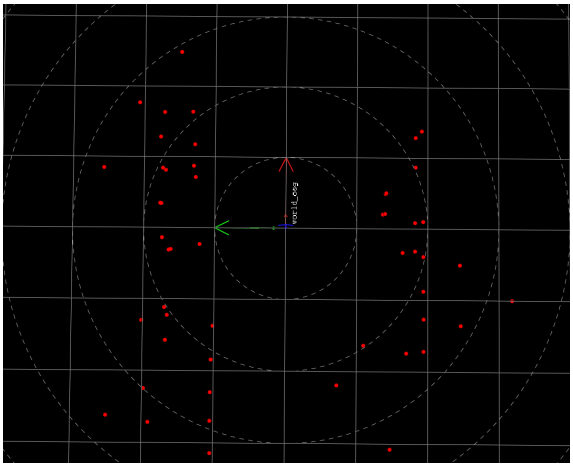
(a) Landmark configuration on Figure 5.5b



(b) Top view: Landmark SLAM ($d_{max} = 10m$)

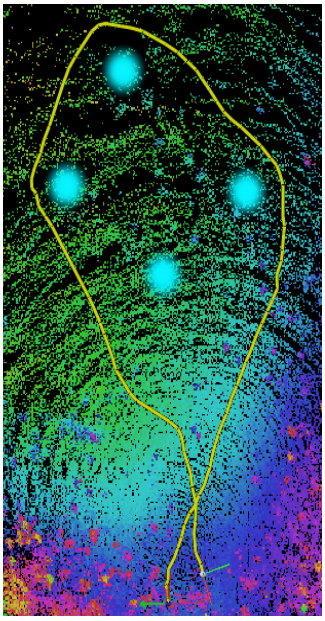


(c) Side view with circled detected landmarks

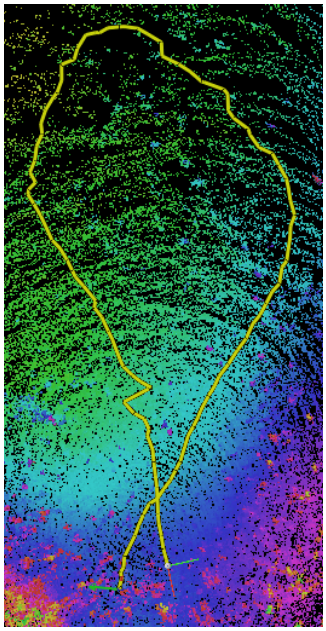


(d) Position of all detected landmarks relative to the sensor at time of detection. Grid size is 1m.

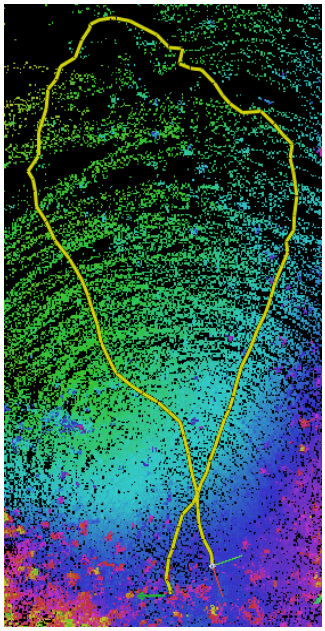
Figure 5.8: Task 2: Navigation with landmarks ($d_{max} = 10m$)



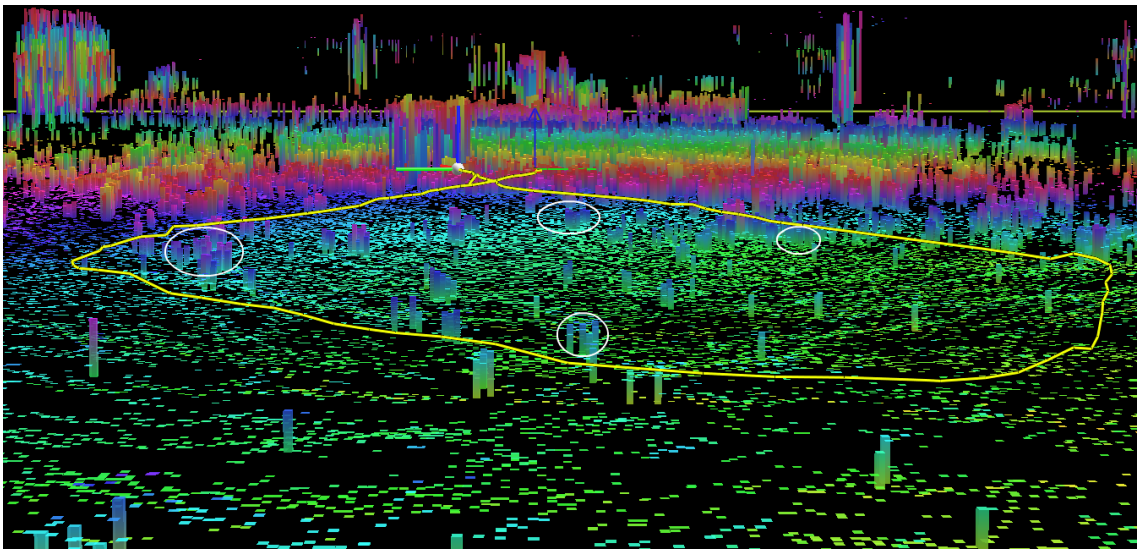
(a) Landmark configuration



(b) Without landmarks ($10m$)



(c) With Landmarks ($10m$)

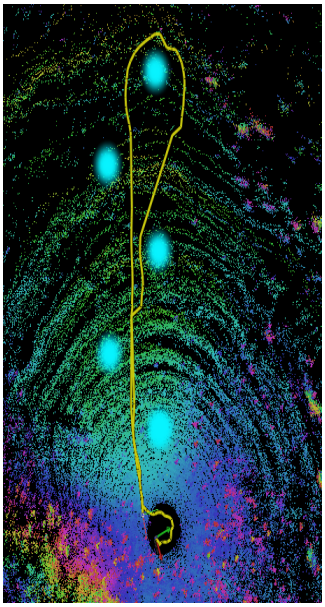


(d) Side view of Figure 5.9b with detected landmarks (circled)

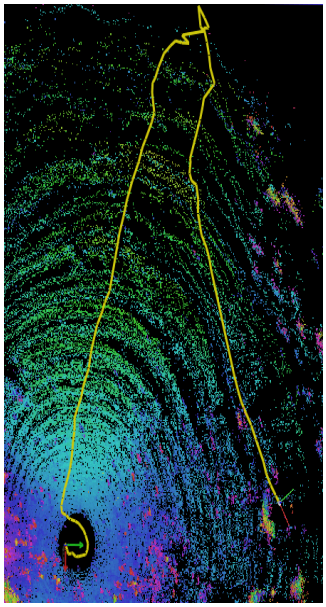
Figure 5.9: Task 3 : Navigation with landmarks

MLS scan during landmark SLAM.

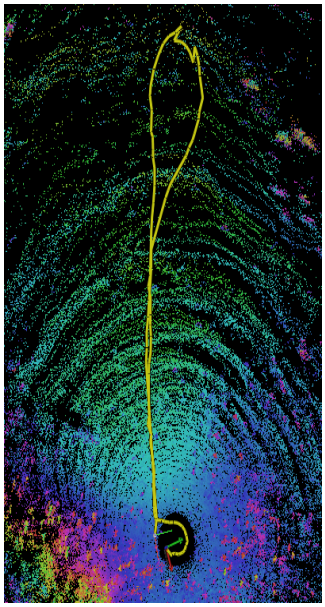
Task 4 consists of travelling straight with landmarks on both sides and returning to the start position after a sharp U-turn. Like before, figure 5.10a provides the benchmark for comparison. This was the result when the algorithm (front-end and back-end) was run with $d_{max} = 10m$ without landmarks. Figure 5.10b shows the results of SLAM without using landmarks with a short sensor range input of $d_{max} = 10m$. It can be seen that the sharp turn resulted in a drift and an error in orientation estimation. This drift got wider as the rover continued and resulted in an erroneous end position. Possible reasons for this might be inability of the ICP algorithm to align point clouds



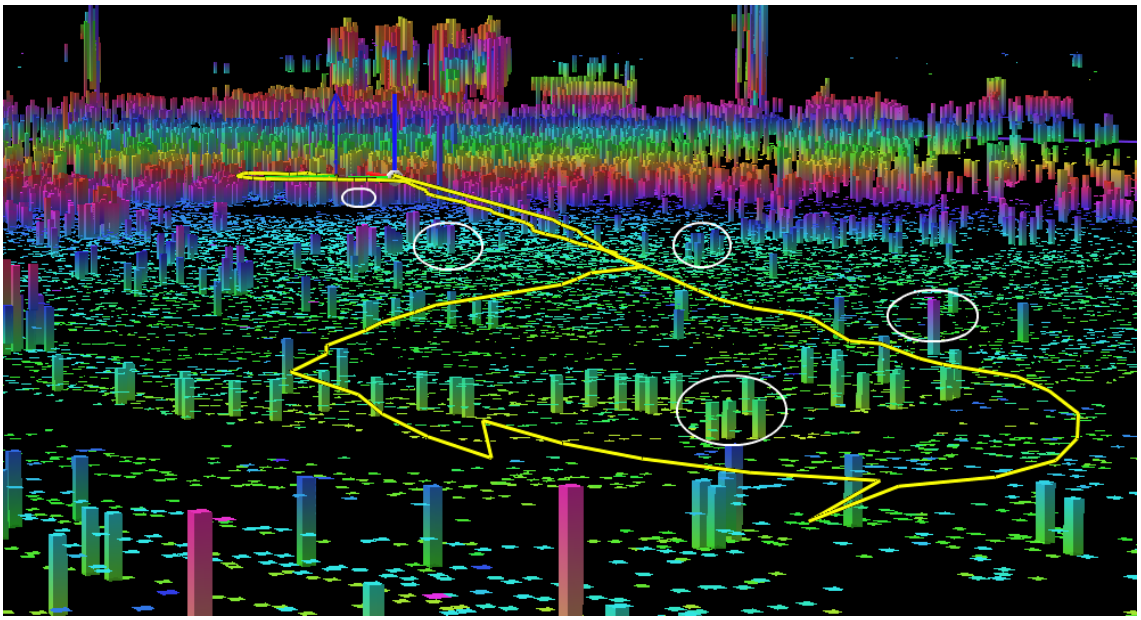
(a) Landmark configuration



(b) Without landmarks ($10m$)



(c) With Landmarks ($d_{max} = 10m$)



(d) Side view with detected circled landmarks in Figure 5.10c

Figure 5.10: Task 4: Navigation with landmarks

generated during the $180°$ turn. Figure 5.10c shows that landmarks were able to correct the bad estimate. However, while the robot end position is similar to Figure 5.10a, the trajectory varies.

5.4 Discussion

The decoupled SLAM system performed well in urban scenes with structures like buildings, cars, trees and the street. The abundance of features in the environment obviated the need to use

our artificial landmark system. Raw scan matching coupled with consistent error minimization by the back-end was enough for a good SLAM system that didn't have to rely on odometry or an Inertial Measurement Unit (IMU).

The test area wasn't secluded enough and since our primary sensor has a range of 80-100 metres (Table 3.1), the features in the vicinity made our SLAM algorithm return sound results. Handicapping our sensor was a way to simulate conditions similar to our use case, which resulted in drift and trajectory errors by our SLAM system (Figures 5.6a, 5.6b, 5.9b and 5.10b). The landmark extension to the back-end (Section 4.4) proved to be useful in these cases. The results obtained using landmarks corrected for trajectory errors (figure 5.9c), bad localization (figure 5.8b) and orientation drift (figure 5.10c). These figures also show that there are minor inconsistencies in the trajectory when compared with results from SLAM without sensor handicap ($d_{max} > 50m$).

CHAPTER 6

Conclusions and Outlook

We developed a lidar-only navigation and mapping solution for operation in challenging environments which makes use of inexpensive deployable landmarks. This method is especially designed for use in robotic space exploration since its not affected by factors like lighting, lack of environment features and reconnaissance prerequisite for navigation - which plague the existing rovers on Mars (Section 1.1). Our SLAM system only requires range data from the laser scanner and produces consistent results without odometry inputs or data from an Inertial Measurement Unit (IMU).

6.1 Key Contributions

1. We implemented a LiDAR-only SLAM system on a rover and tested it on real data.

2. We decoupled the SLAM system into an independent scan matching front-end and a graph optimisation back-end. The front-end was shown to deliver good results by itself in scenarios with a favourable and structured environment. The back-end provided a marked improvement when the front-end struggled. This was mostly during tasks like loops that required good data association.

3. We proposed an artificial landmark based extension to our SLAM system, where the landmarks can be placed randomly and autonomously along the robot's path. The landmark detection system exploits existing sensor capabilities of measuring reflectivities, and requires no additional hardware. This landmark extension proved to be especially helpful in featureless environments where the SLAM system (front-end and back-end) accumulated drift errors.

6.2 Limitations

We acknowledge some of the limitations of our work in this section. These could stem from a variety of reasons including time shortage, methodology, equipment or infrastructure scarcity.

- Qualitative evaluation: Our result evaluation could have been more objective. Given the lack of ground truth data as a consequence of a lack of a global localization system, we relied on comparing our results to the ones obtained without filtering the sensor data. Moreover, we couldn't use a metric (like errors for active vertices) to compare results due to time deficiency. This led us to compare our trajectories by merely looking at it. While we did clearly get a sense of differences between two trajectories, we didn't know how much was that difference exactly.

- Planetary terrain approximation: In addition to sparse vegetation, our test area was surrounded by structural features (Figure 5.2c) which forced us to use measures like filtering out a major portion of the scan in order to simulate a featureless environment. This caused a huge loss of detail which could have been used for better transformation estimates between poses.

- Landmark SLAM results: While the results from landmark SLAM were clearly superior to its counterparts which didn't use landmarks, the landmark trajectories were not entirely accurate when compared with trajectories with long laser range scans. This could have been improved upon using more methods to manipulate the detected landmarks.

6.3 Future Work

The combination of scan matching with graph optimisation is a powerful approach for lidar navigation and mapping. Using reflective landmark modifiers has proven to be a cost-effective, low overhead method to navigate in featureless terrain. This section discusses some of the ways this work could be extended in the future.

The algorithm can be tested in a more representative terrain which lacks structured features in the vicinity. A cave or a desert would be an appropriate choice. It would be interesting to see the results when there is a large volume of point cloud data free of structured features to work with. We estimate that the ICP transformations would be more accurate when a larger amount of points are aligned. Additionally, a more quantitative evaluation of the results would reveal some insights about the algorithm's performance, and its variation with changing configuration parameters.

There is a large amount of literature which tackles the landmark placement problem. Since we have randomly planned our landmarks, it would be fruitful to investigate the effects of planned deployments. A policy to maximize the uniqueness of the environment by placing landmarks in the most optimal configuration along the robot's trajectory could by learned. We think that not only would this be a more efficient use of landmarks, but it would also lead to better data association.

A very interesting extension of this thesis could be adapting it for long range navigation in a featureless terrain under minimal to no lighting conditions. This might provide fascinating insights about large scale graph error minimization and global consistency.

Bibliography

Alismail, H., Baker, L. D. and Browning, B.: 2014, Continuous Trajectory estimation for 3D SLAM from actuated lidar, *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 6096–6101.

Bakambu, J. N., Gemme, S. and Dupuis, E.: 2006, Rover localization through 3D terrain registration in natural environments, *IEEE International Conference on Intelligent Robots and Systems*, pp. 4121–4126.

Batalin, M. and Sukhatme, G.: 2003, Efficient exploration without localization, *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Vol. 2, pp. 2714–2719.

Beinhofer, M., Kretzschmar, H. and Burgard, W.: 2013a, Deploying artificial landmarks to foster data association in simultaneous localization and mapping, *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5235–5240.

Beinhofer, M., Müller, J. and Burgard, W.: 2013b, Effective landmark placement for accurate and reliable mobile robot navigation, *Robotics and Autonomous Systems*, Vol. 61, pp. 1060–1069.

Beinhofer, M., Muller, J., Krause, A. and Burgard, W.: 2013c, Robust landmark selection for mobile robot navigation, *IEEE International Conference on Intelligent Robots and Systems*, pp. 2637–2643.

Bender, M. a., Fernández, A., Ron, D., Sahai, A. and Vadhan, S.: 2002, The Power of a Pebble: Exploring and Mapping Directed Graphs, *Information and Computation* **1**76(1), 1–21.

Besl, P. J. and McKay, N. D.: 1992, A Method for Registration of 3-D Shapes., *IEEE Trans. Pattern Anal. Mach. Intell. ()* **1**4(2), 239–256.

Borrmann, D. and Elseberg, J.: 2008, The efficient extension of globally consistent scan matching to 6 DOF, *Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission The*.

Burgard, W. and Fox, D.: 1996, Estimating the absolute position of a mobile robot using position probability grids, *Proceedings of the national conference on artificial intelligence* pp. 896–901.

Carle, P. J. F., Furgale, P. T. and Barfoot, T. D.: 2010, Long-range rover localization by matching LIDAR scans to orbital elevation maps, *Journal of Field Robotics* **2**7(3), 344–370.

Correa, J. and Soto, A.: 2010, Active visual perception for mobile robot localization, *Journal of Intelligent and Robotic Systems* **5**8(3-4), 339–354.

Craciun, D., Paparoditis, N. and Schmitt, F.: 2010, Multi-view scans alignment for 3D spherical mosaicing in large-scale unstructured environments, *Computer Vision and Image Understanding*, Vol. 114, pp. 1248–1263.

DFKI Robotics Innovation Center: n.d., Rock Robotics Official Website.

Dudek, G., Jenkin, M., Milios, E. and Wilkes, D.: 1997, Map validation and robot self-location in a graph-like world, *Robotics and Autonomous Systems* **2**2(2), 159–178.

Endres, F., Hess, J., Sturm, J., Cremers, D. and Burgard, W.: 2014, 3-D Mapping with an RGB-D camera, *IEEE Transactions on Robotics* **3**0(1), 177–187.

Erickson, L. H.: 2011, An art gallery approach to ensuring that landmarks are distinguishable, *Robotics: Science and . . . .*

Fischler, M. A. and Bolles, R. C.: 1981, Random sample consensus, *Communications of the ACM* **2**4(6), 381–395.

Gamini Dissanayake, M. W. M., Newman, P., Clark, S., Durrant-Whyte, H. F. and Csorba, M.: 2001, A solution to the simultaneous localization and map building (SLAM) problem, *IEEE Transactions on Robotics and Automation* **1**7(3), 229–241.

Garcia, E., Jimenez, M. A., De Santos, P. G. and Armada, M.: 2007, The evolution of robotics research, *IEEE Robotics and Automation Magazine* **1**4(1), 90–103.

Grisetti, G., Kummerle, R., Stachniss, C. and Burgard, W.: 2010, A tutorial on graph-based SLAM, *IEEE Intelligent Transportation Systems Magazine* **2**(4), 31–43.

Howard, A., Mataric, M. J. and Sukhatme, G.: 2001, Relaxation on a Mesh: a Formulism for Generalized Localization, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (Iros), 1055–1060.

Huang, a. S., Antone, M., Olson, E., Fletcher, L., Moore, D., Teller, S. and Leonard, J.: 2010, A High-rate, Heterogeneous Data Set From The DARPA Urban Challenge, *The International Journal of Robotics Research* **2**9, 1595–1601.

Inoue, H., Ono, M., Tamaki, S. and Adachi, S.: 2016, Active localization for planetary rovers, *2016 IEEE Aerospace Conference*, IEEE, pp. 1–7.

Jebara, T., Azarbayejani, A. and Pentland, A.: 1999, 3D structure from 2D motion, *IEEE Signal Processing Magazine* **1**6(3), 66–84.

Johnson, A.: 1997, Spin-images: a representation for 3-D surface matching, *Technology* (CMU-RI-TR-97-47), 138.

Joyeux, S., Schwendner, J. and Roehr, T. M.: 2014, Modular Software for an Autonomous Space Rover, *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014). International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-2014), June 17-19, Montreal,, Québec, Ca*.

Kaupisch, T. and Noelke, D.: 2014, DLR SpaceBot Cup 2013: A Space Robotics Competition, *KI-K{ü}nstliche Intelligenz* **2**8(2), 111–116.

Kleiner, A., Prediger, J. and Nebel, B.: 2006, RFID technology-based exploration and SLAM for search and rescue, *IEEE International Conference on Intelligent Robots and Systems*, pp. 4054–4059.

Konolige, K. and Agrawal, M.: 2007, Frame-frame matching for realtime consistent visual mapping, *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2803–2810.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W.: 2011, G2o: A general framework for graph optimization, *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3607–3613.

Lee, D. T. and Lin, A. K.: 1986, Computational Complexity of Art Gallery Problems, *IEEE Transactions on Information Theory* **3**2(2), 276–282.

Leonard, J. and Newman, P.: 2003, Consistent, convergent, and constant-time SLAM, *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1143–1150.

Leonard, J. J. and Durrant-Whyte, H. F.: 1991, Mobile robot localization by tracking geometric beacons, *IEEE Transactions on Robotics and Automation* **7**(3), 376–382.

Lu, F. and Milios, E.: 1997, Globally Consistent Range Scan Alignment for Environment Mapping, *Autonomous Robots* **4**(4), 333–349.

Maimone, M., Cheng, Y. and Matthies, L.: 2007, Two years of visual odometry on the Mars Exploration Rovers, *Journal of Field Robotics* **2**4(3), 169–186.

Mei, C., Sibley, G., Cummins, M., Newman, P. and Reid, I.: 2011, RSLAM: A system for large-scale mapping in constant-time using stereo, *International Journal of Computer Vision* **9**4(2), 198–214.

Meyer-Delius, D., Beinhofer, M., Kleiner, A. and Burgard, W.: 2011, Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot, *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5173–5178.

Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B.: 2003, FastSLAM 2 . 0 : An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges, *Ijcai*.

Nuchter, A., Lingemann, K. and Hertzberg, J.: 2007, 6D SLAM-3D Mapping Outdoor Environments, *Journal of Field Robotics* **2**4, 699–722.

Nüchter, A., Lingemann, K., Hertzberg, J. and Surmann, H.: 2005, 6D SLAM with approximate data association, *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, Vol. 2005, pp. 242–249.

Olson, C. F., Matthies, L. H., Schoppers, M. and Maimone, M. W.: 2003, Rover navigation using stereo ego-motion, *Robotics and Autonomous Systems* **4**3(4), 215–229.

Olson, E.: 2009, Recognizing places using spectrally clustered local matches, *Robotics and Autonomous Systems* **5**7(12), 1157–1172.

PCL: 2011, Point Cloud Library.
Pheonix Aerial

Pheonix Aerial*: 2014.*

Prentice, S. and Roy, N.: 2010, The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance, Springer Tracts in Advanced Robotics*, Vol. 66, pp. 293–305.*

Rafflin, C. and Fournier, A.: 1995, Learning with a friendly interactive robot for service tasks in hospital environments, Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, Vol. 3, pp. 492–497 vol.3.*
Rock, the Robot Construction Kit

Rock, the Robot Construction Kit: 2011.

Ryde, J. and Hu, H.: 2007, Mobile robot 3D perception and mapping without odometry using multi-resolution occupancy lists, *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007*, pp. 331–336.

Sala, P., Sim, R., Shokoufandeh, A. and Dickinson, S.: 2006, Landmark selection for vision-based navigation, *IEEE Transactions on Robotics* **2**2(2), 334–349.

Salas, J. and Gordillo, J. L.: 1998, Placing artificial visual landmarks in a mobile robot workspace, *Ibero-American Conference on Artificial Intelligence*, Springer, pp. 274–282.

Segal, A., Haehnel, D. and Thrun, S.: 2009, Generalized-ICP, *Robotics: Science and Systems* **5**, 168–176.

Sharp, G. C., Lee, S. W. and Wehe, D. K.: 2002, ICP registration using invariant features, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**4(1), 90–102.

Solomon, L.: 2011, *The privatization of space exploration: business, technology, law and policy*.

Solomon, L. D., Inoue, H., Ono, M., Tamaki, S., Adachi, S., Correa, J., Soto, A., Rupp, T., Levi, P., Ercan, A. O., Yang, D. B., El Gamal, A., Guibas, L. J., Salas, J., Gordillo, J. L., Pedersen, L., Han, C. S., Vitus, M., Kaupisch, T. and Noelke, D.: 2014, DLR SpaceBot Cup 2013: A Space Robotics Competition, *Ibero-American Conference on Artificial Intelligence*, Vol. 1, Springer, Springer, pp. 448–453.

Stentz, A.: 1995, The focussed D* algorithm for real-time replanning, *14th International Joint Conference on Artificial intelligence (IJCAI)* (August), 1652–1659.

Strasdat, H., Stachniss, C. and Burgard, W.: 2009, Which landmark is useful? Learning selection policies for navigation in unknown environments, *2009 IEEE International Conference on Robotics and Automation* pp. 1410–1415.

Sun, Y., Paik, J., Koschan, A., Page, D. L. and Abidi, M. A.: 2003, Point fingerprint: A new 3-D object representation scheme.

Surmann, H., N??chter, A. and Hertzberg, J.: 2003, An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments, *Robotics and Autonomous Systems* **4**5(3-4), 181–198.

Taati, B., Bondy, M., Jasiobedzki, P. and Greenspan, M.: 2007, Variable dimensional local shape descriptors for object recognition in range data, *Proceedings of the IEEE International Conference on Computer Vision*.

Thrun, S.: 1998, Finding landmarks for mobile robot navigation, *Robotics and Automation, 1998. Proceedings. 1998 . . .* , Vol. 2, pp. 958–963.

Thrun, S.: 2001, Learning occupancy grids with forward models, *Intelligent {Robots} and {Systems}, 2001. {Proceedings}. 2001 {IEEE}/{RSJ} {International} {Conference} on* **3**, 1676–1681.

Thrun, S.: 2002, Robotic Mapping: A Survey, *Science* **2**98(February), 1–35.

Thrun, S.: 2006, The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures, *The International Journal of Robotics Research* **2**5(5-6), 403–429.

Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z. and Durrant-Whyte, H.: 2003, Simultaneous Localization and Mapping with Sparse Extended Information Filters, *The International Journal of Robotics Research* **2**3(7), 693–716.

Tong, C. H., Barfoot, T. D. and Dupuis, E.: 2012, Three-dimensional SLAM for mapping planetary work site environments, *Journal of Field Robotics*, Vol. 29, pp. 381–412.

Triebel, R., Pfaff, P. and Burgard, W.: 2006, Multi-level surface maps for outdoor terrain mapping and loop closing, *IEEE International Conference on Intelligent Robots and Systems*, pp. 2276–2282.

Van Der Merwe, R., Doucet, a., De Freitas, N. and Wan, E.: 2001, The Unscented Particle Filter, *Advances in Neural Information Processing Systems* **9**6(6080), 584–590.
VLP-16

VLP-16*: 2016.*

Wan, E. a. and van der Merwe, R.: 2002, The Unscented Kalman Filter*, Vol. 5.*

Wang, H., Jenkin, M. and Dymond, P.: 2011, The relative power of immovable markers in topological mapping, Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1050–1057.*

Weingarten, J. and Siegwart, R.: 2005, EKF-based 3D SLAM for structured environment reconstruction, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 2089–2094.*

Wikipedia: 2016, Maximum Likelihood Estimation — {W}ikipedia{,} The Free Encyclopedia.

Young, A.: 2015, The Twenty-First Century Commercial Space Imperative*, Springer International Publishing.*