RHEINISCHE FRIEDRICH-
WILHELMS-UNIVERSITÄT
BONN

MASTER'S THESIS

Multi-Resolution Surfel SLAM and Rough Terrain Navigation using a Continuously Rotating 2D Laser Scanner

Author:
Mark SCHADLER

Supervisor:
Jörg Stückler

Reviewers:
Prof. Dr. Sven Behnke
Dr. Dirk Schulz

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Computer Science

in the

Autonomous Intelligent Systems
Institute for Computer Science VI

January 2014

Declaration of Authorship

"I hereby declare that I have prepared this thesis independently and without the use of sources and aids other than those stated. All sources have been cited accordingly."

Bonn

Date: ____

Signed: ____

Mark Schadler

“Research is what I’m doing when I don’t know what I’m doing.”

Wernher von Braun

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

Abstract

Autonomous Intelligent Systems
Institute for Computer Science VI

Master of Science in Computer Science

Multi-Resolution Surfel SLAM and Rough Terrain Navigation using a Continuously Rotating 2D Laser Scanner

by Mark SCHADLER

Mapping and real-time localization in 3D environments are prerequisites for the advancement of situation-aware, autonomous mobile robots. When combined with terrain traversability assessment and global planning methods, these capabilities form a solution for autonomous rough-terrain navigation, capable for real-world use.

This thesis proposes a method for SLAM and autonomous robot navigation using a continuously rotating 2D laser scanner.

Multi-resolution surfel representations allow for compact storage and efficient registration of local maps. Through probabilistic graph optimization algorithms, local maps can be globally aligned to an allocentric environment map - suitable for navigation planning.

Real-time pose tracking is performed using a particle filter and individual laser scan lines. Particle filtering robustly handles non-linearity in both observation and motion models and can be easily parallelized for efficient computation.

Navigation planning uses region growing methods within an allocentric map to determine a traversability graph. Using this graph and drivability assessment to determine path costs, the well-known A* algorithm is used for path planning.

This approach is evaluated using both simulation and challenging real-world environments including the DLR Spacebot Cup competition arena and an open dataset from the University of Toronto.

Acknowledgements

First and foremost I would like to thank my thesis advisor Jörg Stückler for his imparted knowledge, dedication, and routine motivational pushes. Additional thanks to Professor Sven Behnke for technical and managerial support including lab facilities and working within his group.

Second I would like to thank my friends both from Bonn and abroad for helping to maintain an enjoyable and fun atmosphere, even during times of great stress.

Finally I would like to thank my family for the emotional and fiscal support of studying such a great distance from home.

Contents

List of Figures

List of Tables

Abbreviations

ATE	**A**bsolute **T**rajectory **E**rror
DGPS	**D**ifferential **G**lobal **P**ositioning **S**ystem
DLR	**D**eutsches Zentrum für **L**uft und **Raumfahrt**
DoF	**D**egrees **o**f **F**reedom
EKF	**E**xtended **K**alman **F**ilter
FoV	**F**ield **o**f **V**iew
ICP	**I**terative **C**losest **P**oint
LDS	**L**inear **D**ynamic **S**ystem
LIDAR	**Li**ght **D**etection **A**nd **R**anging
LM	**L**evenberg-**M**arquardt
MLS	**M**ulti-**L**evel **S**urface
MRS	**M**ulti-**R**esolution **S**urfel
NDT	**N**ormal **D**istributions **T**ransform
SIR	**S**equential **I**mportance **R**esampling
SIS	**S**equential **I**mportance **S**ampling
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
SVD	**S**ingular **V**alue **D**ecomposition

Chapter 1

Introduction

Numerous applications for autonomous robot navigation in rough-terrain environments exist. This includes search and rescue missions, security and monitoring systems, and domestic service robots. These applications are most pressing when the environment is not easily navigable for humans, possibly due to safety or other related concerns.

Providing a navigation solution suitable for complex environments, poses challenges spanning several fields of study within robotics. Thus discussion can focus on several independent but related problems, each having a key impact on the overall system functionality. In this thesis, these issues are separated into three distinct problem groups - each with a proposed solution.

Key to all components within the system is localization - the determination of pose within the environment. Localization within a 3D environment must consider six degrees of freedom (DoF) - three for position and three for orientation. Using laser scanner data, odometry input, and the previous pose estimate, the localization system calculates the most likely current pose estimate.

Mapping is responsible for modeling the environment in a representation suitable for localization and planning activities. With a rotating 2D laser scanner, 3D laser scans can be recorded and aligned allowing for allocentric map creation. Using a

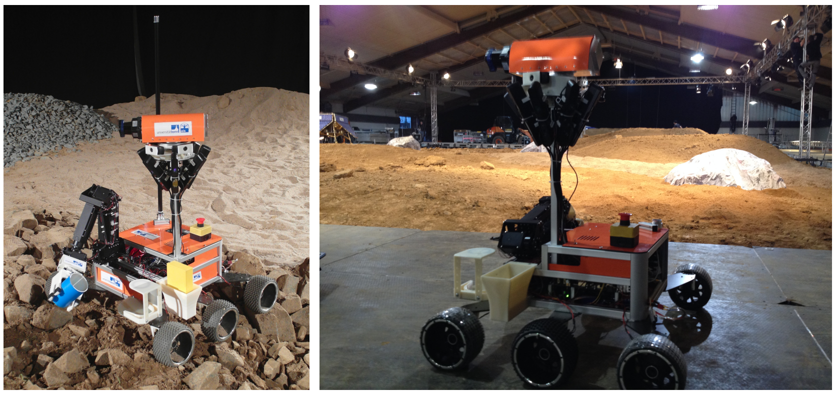


FIGURE 1.1: The Explorer robot platform developed by the University of Bonn pictured with the DLR Spacebot Cup arena.

probabilistic graph optimization framework, map data is globally registered and the resulting pose information can be used to refine the localization system.

Given a pose estimate and an allocentric map, navigation planning can be used to generate paths to target poses. Using drivability assessment, the system should compute trajectories considered safe for the specific mobile platform. Along these trajectories, new data for mapping and localization can be acquired - thus completing the data-cycle of the system.

The presented system has been developed for the Explorer platform at the University of Bonn. The Explorer robot has competed in the Deutsches Zentrum für Luft- und Raumfahrt (DLR) Spacebot Cup competition with reasonable success. Figure 1.1 shows the Explorer robot with the Spacebot Cup competition arena in the background.

In the remainder of this chapter, work related to laser scanners, localization, mapping, and navigation is presented along with proposed improvements. Chapter 2 provides the theoretical background of system components while Chapter 3 details the key thesis contributions and implementation details. Evaluation of the presented method is reported in Chapter 4. Finally in Chapter 5, a conclusion is presented along with potential future improvements.

1.1 Problem Definition

This thesis presents a framework for performing simultaneous localization and mapping (SLAM), drivability assessment, and navigation planning.

SLAM simultaneously tracks the robot pose and builds an environment map given a time series of laser range data. This problem statement is posed as given an initial pose, x_0, and time series of measurements, $z_{1:t}$, determine the most likely robot pose x_t and most likely environment map m_t. Due to system operation discussed in Section 3.1, the mapping component only measures 3D scans during specific time intervals, thus it is possible given some constant k for $m_{t-k}, \cdots, m_t$ to be identical.

Given the current map, m_t, and the robot drivability parameters such as size and maximum drivable incline, drivability assessment determines the drivable space d_t within the map m_t. The navigation component generates a path p_t through the drivable space d_t directed towards a goal g_t that minimizes the path cost based upon drivability metrics.

1.2 Related Work

Within this section, state of the art work within the field of SLAM and navigation is discussed, including identification of approach similarities and differences. With reference to sensing hardware, both works using laser scanners and those using other sensors are presented. When sensing hardware accounts for major differences in operation or theory, these issues are explicitly discussed.

1.2.1 Mapping

Due to the overwhelming computation and storage demands of sensor data generated in even relatively small 3D environments, approaches to mapping must carefully analyze performance characteristics. Initial research to reduce the space and computation requirements focused on discretizing the environment into grids. Roth-Tabak and Jain [1989] use a constant size grid within environments to model free and occupied space. However, this approach is not scalable to large or detailed environments due to the required storage space. Increasing environment size or resolution results in non-linear space enlargements.

Hornung et al. [2013] propose OctoMap, a multi-resolution volumetric mapping system based on octrees and occupancy mapping - see Section 2.3.1 for an overview of the octree data structure. Within an octree, volumetric elements denoted "voxels" store the probability of being occupied. Thus the system naturally maintains the distinction between occupied, free, and unknown for the full environment. Using the octree structure, branching nodes store aggregate data for all children allowing querying of the map at varying resolutions. Occupancy mapping additionally allows for simple fusing of multiple range measurements, however the highest map resolution is constant. The OctoMap library does not implement map-to-map registration and thus needs additional components for localization to correctly fuse sensor data.

Ryde and Hu [2010] present a differing approach to managing the copious data generated by 3D sensors using multi-resolution occupied voxel lists. Contrary to general occupancy mapping, the presented system only stores occupied cells in a large list structure. List storage allows for the implicit dynamic enlargement or reduction of the environment, as well as minimal memory overhead. Using a hash table datastructure, this list can be efficiently queried and updated from new laser data given an accurate pose estimate. Current methods of path planning within this mapping scheme relies upon 2D region growing using neighbor querying within the list structure.

Based on the work from Biber and Strasser [2003], Magnusson et al. [2007], and Stoyanov et al. [2012], Saarinen et al. [2013] propose a mapping system using the normal distributions transform (NDT) and occupancy mapping. The NDT was originally developed for 2D laser scan matching, modeling a laser scan as a set of Gaussian probability distributions. Subdividing the environment into regular cells, allows for a simple calculation of the cells distribution. This distribution is represented by a mean and covariance that express the probability of measuring a sample within the cell [Biber and Strasser, 2003]. Using work from Magnusson et al. [2007] and Stoyanov et al. [2012], who expanded NDT to three dimensions, the alignment framework was combined with occupancy mapping to efficiently process highly dynamic environments. Within this work, NDT map alignment was used to generate an occupancy grid; thus for testing an external localization system was used. This work is highly similar to our surfel based mapping system. We additionally consider the viewing angle of a cell and store up to 6 Gaussian distributions per surfel (corresponding to the six faces of the cell boundary). However, we do not model the occupancy probability and thus are unable to robustly handle dynamic environments.

Research addressing mapping with laser scanners without the necessity to stop during 3D scans has been investigated [Bosse and Zlot, 2009, Elseberg et al., 2012, Stoyanov and Lilienthal, 2009, Maddern et al., 2012, Anderson and Barfoot, 2013]. The work by Maddern et al. [2012] focuses specifically on calibration of 2D and 3D light detecting and ranging (LIDAR) sensors to allow accurate aggregation of laser data while moving. The method uses a pointcloud quality metric based on entropy to determine suitable aggregation parameters. Using this approach could remove the additional requirements of stop-and-go mapping, however the method has the potential to inject additional error into sensor measurements. Due to the use case of our system in primarily space exploration vehicles, the extra time necessary to remain still during scanning is not seen as a pressing issue.

1.2.2 SLAM

Khoshelham [2010] proposes using planar objects extracted from laser scan lines for localization within 3D environments. In this work, plane matching is performed using least-squares optimization, similar to our mapping system. However the localization system requires 3D scans for pose tracking.

Kuemmerle et al. Kuemmerle et al. [2007] apply Monte Carlo localization in Multi-Level Surface (MLS) maps.[1] Contrary to our work where particles are updated using importance weighting from observation likelihoods, Kuemmerle et al. minimize an entropy calculation given the expected and actual measurements.

Klaess et al. [2012] model the environment in surfel maps in a fixed resolution, similar to the 3D-NDT representation from Magnusson et al. [2007]. Localization is performed using a tilting 2D laser by matching line elements extracted from the 2D scan lines in a particle filter framework, assuming motion of the robot in the horizontal plane. Our approach can avoid making the assumption of horizontal plane motion.

Nüchter et al. [2007] model the environment by storing point clouds generated directly from sensing hardware. This direct-data approach has many potential performance and space caveats when naively processed. To handle the computation demands to perform scan registration, a heuristic approach is used to initialize the iterative closest point (ICP) algorithm. The ICP algorithm calculates the localization estimate from the registered map pose, and thus localization is only updated after each scan registration. An additional step after all scans are taken is performed to refine the environment model using a global constraint relaxation technique.

1.2.3 Navigation and Drivability Assessment

Path planing for driving in planar 2D indoor environments is a well-studied topic in robotics. Hornung et al. [2012] consider the variable footprint of a mobile manipulation robot for 2D path planning. Starting from a 3D representation of the environment in an OctoMap, they derive a multi-layered 2D representation for planning. Planning is performed using a lattice graph in 2D space where nodes consist of their 2D position and yaw orientation. Connections between graph

[1] See Triebel et al. [2006] for an introduction on MLS maps.

nodes are implemented as motion primitives, thus implicitly generating smooth trajectories.

Klaess et al. [2012] map the environment in 3D surfel grids using a 2D laser scanner. From this representation, a 2D navigation map is derived using similar drivability features as our work. The height of the robot is additionally considered to plan collision free 2D paths in a 3D environment.

For navigation on non-flat terrain, several approaches generate 2D cost maps from sensor readings to essentially treat path planning in 2D [Lee et al., 2008, Gerkey and Konolige, 2008, Ferguson and Likhachev, 2008]. Work by Lee et al. [2008] uses rapidly exploring random trees for efficient planning within large outdoor environments. Gerkey and Konolige [2008] plan trajectories implicitly by generating robot motion commands to determine costs. Using the cost map, motion commands follow the minimum cost-gradient to execute trajectories.

Recently, Stoyanov Stoyanov et al. [2010] proposed a wavefront-propagation path planner in 3D-NDT maps. Within our work, the wavefront algorithm is used for region growing and the precomputation of node and edge costs. This separates cost calculation from trajectory planning, thus our work does not need to recalculate drivability metrics before each planning step.

1.3 Proposed Improvements

The following section details the proposed improvements to current research concerning mapping, tracking, and navigation of rough terrain environments.

1.3.1 Multi-resolution Surfel Mapping

Presented in Section 2.3, multi-resolution surfel (MRS) maps are used as the underlying structure for modeling the environment using laser range measurements. The multi-resolution feature of this data structure implicitly handles sensor error during local map alignment. In the data association step of registration, maps are processed from fine to coarse resolutions where association candidates are selected from a query volume with size in dependence with node depth (resolution). By controlling the resolution at which laser scanner data is aggregated into a MRS map, one can influence data association and thus registration in accordance with sensor error properties.

The MRS map implementation will be adapted to handle the specific error characteristics of a laser scanner. This update specifically requires the calculation of measurement dispersion and error in dependence with distance. Section 3.2 presents the error calculation used for our specific laser scanner model.

1.3.2 Scanline Localization

Current work concerning 6D localization using laser scanners requires 3D scan measurements. This allows for a direct pose computation using local map alignment. However, this constrains the localization update rate to the rotation speed of the laser - often in the range of $\frac{1}{4}$ Hz - $\frac{1}{2}$ Hz. During motion, aggregating laser scan lines into a 3D scan requires intermediate localization estimates. This needs an additional accurate localization method or a complicated correction model - which has yet to be robustly developed. In order to overcome these issues, this thesis presents a method to perform 6D localization using individual laser scan lines.

Due to the limited data within a single laser scan line, there is insufficient information to create a MRS map and calculate a direct localization estimate through pose optimization. This work suggests using particle filters to allow for a scan line to MRS map observation model. Through this method, the localization update rate is theoretically tied to the sensor update rate and scan line aggregation accuracy issues are avoided. Section 3.3 presents the implementation of this component.

1.3.3 Surfel Drivability Assessment

Drivability assessment is implemented using MRS maps and the terrain assessment characteristics described in Section 3.4.4. This specific contribution focuses on region growing within surfel grids and using the surfel datastructure for cost assessment when considering the entire robot coverage region. Additional issues due to small misalignments of local MRS maps and sensor calibration must be handled using a specialized combination strategy, allowing for planning within the entire mapped environment. Drivability assessment must not only determine costs for drivability, but construct a datastructure suitable for path planning. Section 3.4 details the implementation of drivability assessment.

1.3.4 Path Planning

Path planning is implemented using the well-known A* algorithm within a traversability datastructure created by the drivability assessment component. Specific challenging cases such as planning to unknown regions or obstacles must be carefully handled to maintain a robust navigation system. Section 3.5 documents the integration of A* planning within the presented system.

Chapter 2

Background

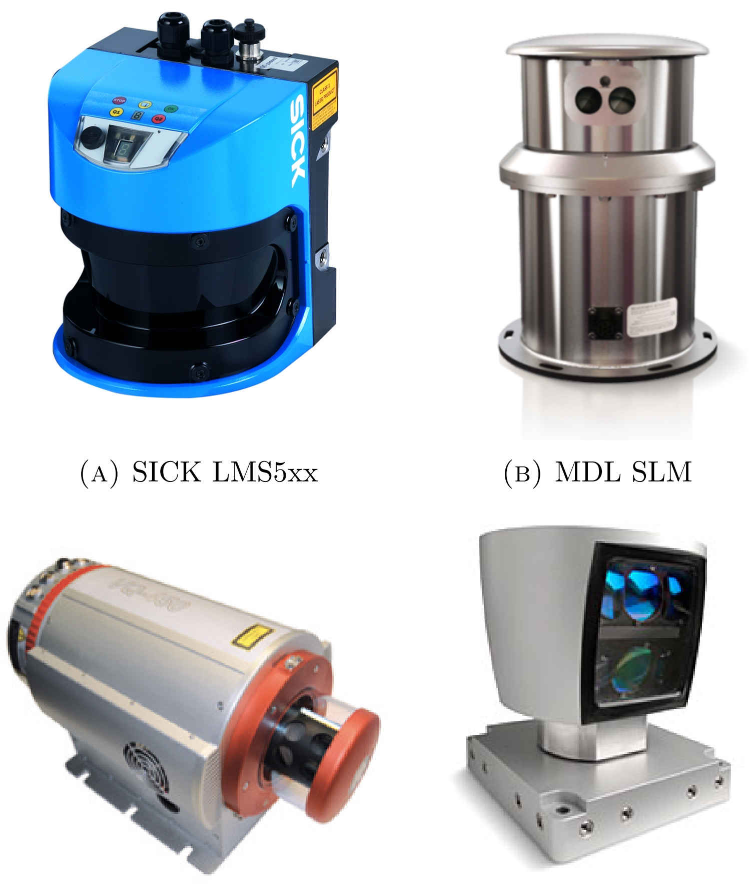
2.1 LIDAR

Light detection and ranging (LIDAR) sensors use directed pulses of light to calculate shape and surface characteristics. LIDAR sensors are used in a number of applications from simple range/motion detection to earth surface mapping [NOAA Costal Services Center, 2012]. This section serves as a short introduction to LIDAR and available sensing hardware. Figure 2.1 shows several laser ranging sensors currently available to the public. For the Explorer platform we have chosen to use a Hokuyo UTM-30LX-EW; an overview of the Hokuyo UTM-30LX-EW sensor is presented including error characteristics and sensor interface information.

2.1.1 Laser Scanner Overview

Laser scanners calculate range measurements by emitting pulses of light and measuring the reflection detection time.[1] In the case of 2D scanners, the emission angle is additionally considered and produces a scan line of individual range readings.

[1] In the form of laser pulses at a wavelength suitable for the application. The wavelength can vary from around 10 μm to 250 nm.



(A) SICK LMS5xx (B) MDL SLM

(C) Riegl VQ-450 (D) Velodyne HDL-64E

FIGURE 2.1: Various 2D laser scanners currently available with brand indicated.

Recent advances in laser scanner technology allow detection of surface characteristics including remission (reflectance) measurements. Due to the consistency of the speed of light, laser range measurements exhibit excellent error characteristics and allow for a relatively high data rate. These factors have increased the attractiveness of using laser scanners sensors in robotic applications.

Three dimensional laser scans can be generated from 2D scanners by aggregating 2D scan lines. However as range information is relative to the scanner itself, tracking of the sensor's position is necessary for data aggregation. Section 3.1 details the specific sensor movement used on the Explorer robot platform to collect 3D laser scans.

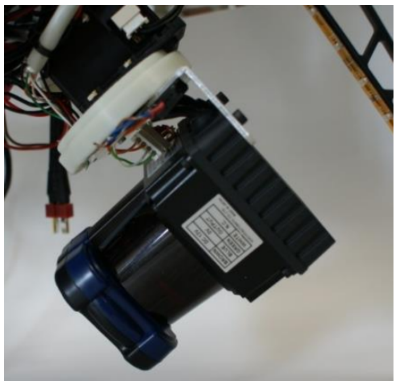


Figure 2.2: The Hokuyo 30-UTMLX-EW 2D laser scanner.

2.1.2 Hokuyo UTM-30LX-EW

The Hokuyo UTM-30LX-EW was selected for the project as the sensor is lightweight and relatively affordable while having an acceptable field of view (FoV) and update rate. The sensor is shown in Figure 2.2.

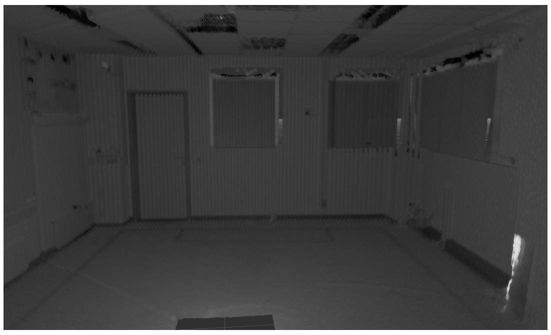
The sensor operates at 40 Hz and generates 1081 range measurements per scan line with a FoV of 270° and angular resolution of 0.25°. The maximum range is 30 m with an accuracy of ±30 mm in the range 0.1 m - 10 m and ±30 mm in the range 10 m - 30 m. Contrary to RADAR, LIDAR sensors are more prone to environmental effects from fog, rain, clouds, and sunlight.[2][NOAA Costal Services Center, 2012] According to the product specification, under these environmental conditions measured ranges will be shorter than true distances.

Figure 2.3 shows an example 3D laser scan from the Hokuyo sensor colored by remission including a corresponding photo of the environment. From visualizing the surface remission we can see additional environmental structure useful for 3D scan matching.

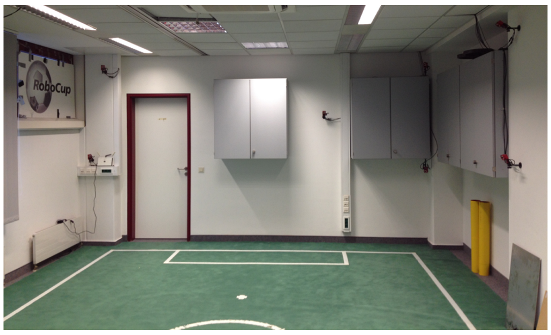
According to the UTM-30LX specification, remission/reflectance readings from the laser scanner vary quadratically with the distance from the sensor.[3] Figure 2.4 shows the variation of intensity readings with distance [Hokuyo Autonomatic CO., 2008].

[2] RADAR uses radio waves to measure range which have very weak absorption characteristics compared to the infrared and visible wavelengths used by LIDAR devices.

[3]A laser scanner from Hokuyo having only minor differences between the UTM-30LX-EW.



(A) Gray-scale indicates surface remission/reflectance.



(B) Photograph of environment.

FIGURE 2.3: Example laser scan with corresponding color image.

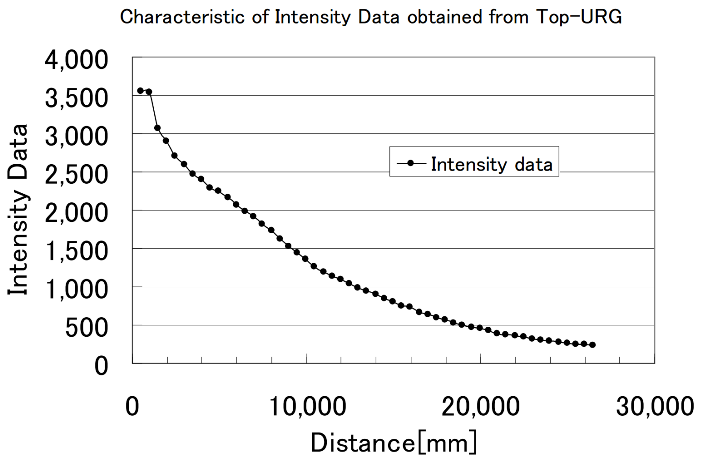


FIGURE 2.4: The variation of laser intensity in accordance with sensor distance for the Hokuyo UTM-30LX [Hokuyo Autonomatic CO., 2008].

Section 2.3.3 details the use of a surface reflectance feature. Problems related to distance dependent reflectance readings are mitigated by considering only the local contrast of surfaces. However, from Figure 2.4 we see there is a non-linear dependence of intensity with distance. By assuming the local neighborhood of surface data must be relatively close in distance, due to a sufficient mapping resolution, we can then assume the distance dependence is approximately linear in the local region.

2.2 Pose Representation

The SLAM tracking problem is presented as the estimation of the most likely robot pose within a map reference frame denoted M. The mapping problem is posed as finding the most likely map from sensor readings and thus must use pose information to combine 3D laser scans. The robot trajectory is assumed to follow rigid body motion; this allows for a compact pose representation suitable for the entire system.

This section introduces rigid body motions for pose representation, which serves as a brief overview for the application of rigid body motion in the context of pose tracking and mapping. A more detail introduction including proofs can be found in the freely available work, "Classical Mechanics" [Shapiro, 2010, p.85-93].

2.2.1 Rigid Body Motion

A rigid body is an object that cannot undergo deformation, thus all distances between any points within the body remain constant regardless of external forces. Applied to three dimensional space using a rigid body $X \in \mathbb{R}^3$, the mapping $g : \mathbb{R}^3 \to \mathbb{R}^3$ is a rigid body motion when the following constraints are satisfied:

1. Distance preservation:

$$|g(p) - g(q)| = |p - q| \ \forall p, q \in \mathbb{R}^3$$

2. Vector cross-product preservation:

$$g(v \times w) = g(v) \times g(w) \ \forall v, w \in \mathbb{R}^3$$

These constraints implicitly result in preservation of angles within the rigid body as well as orientation.[4]

[4] Preservation of orientation preserves the handedness of a coordinate frame. Handedness enforces the direction of "positive" in each axis. In three dimensions a coordinate frame can be left or right handed.

A trajectory can therefore be described by the rigid body motion of an arbitrary coordinate frame attached to a rigid body. This configuration can be expressed as the translation and rotation of a coordinate frame relative to some global frame (here denoted as the map frame M). Often this frame is attached to the center of mass of an object, however as we are not concerned with physical modeling but simple pose representation we can ignore these specific details.

2.2.2 Pose Representation

To express an arbitrary coordinate frame F with respect to a global frame M in $\mathbb{R}^3$, we use a rigid body motion g. The fixed frame M is defined by the basis vectors e_1, e_2, e_3. Thus the translational offset of frame F w.r.t. M can be described by a vector $\vec{t} \in \mathbb{R}^3$. The origin of frame F, o_f can be calculated as

$$o_f = t_1\vec{e_1} + t_2\vec{e_2} + t_3\vec{e_3} \tag{2.1}$$

By defining the basis vectors of frame F as e'_1, e'_2, e'_3, we can calculate their value w.r.t. frame M as

$$e'_i = \sum_{j=1}^{3} R_{ij} e_j : i \in \{1, 2, 3\} \tag{2.2}$$

The mapping g_{MF} defines frame F in the fixed frame M and is fully described by the translation vector $\vec{t}_{MF}$ and rotation matrix $R_{MF} = [u\ v\ w]$ with column vectors $u = g_*(e_1)$, $v = g_*(e_2)$, and $w = g_*(e_2)$. A point P_F in frame F can be expressed in frame M by

$$P_M = R_{MF} P_F + t_{MF} \tag{2.3}$$

The rotation and translation components can be combined using a 4×4 homogeneous transformation matrix T_{MF} with form:

$$T_{MF} = \begin{bmatrix} R_{MF} & \vec{t}_{MF} \\ \vec{0} & 1 \end{bmatrix} \tag{2.4}$$

Using this transformation matrix, we can rewrite Equation 2.3 as

$$P_M = T_{MF} P_F \tag{2.5}$$

Within the context of SLAM, we consider the robot trajectory in the map reference frame M, a time-discrete rigid body mapping $g(t) : \mathbb{R} \to SE(3)$. This mapping gives the configuration of the robot frame in the map frame at time t. The relative difference of frame configuration between time t_1 and t_2 can be calculated as

$$g(t_2, t_1) = g(t_2)^{-1} g(t_1) \tag{2.6}$$

Likewise composition from time t_1 to t_3 can be calculated by composing each intermediate mapping as

$$g(t_3, t_1) = g(t_3, t_2) g(t_2, t_1) \tag{2.7}$$

2.2.3 Alternative Rotation Representation

In the previous section a 3×3 rotation matrix $R \in SO(3)$ is used to represent orientation. In practice this matrix must be calculated from an equivalent parameterization often performed using Euler angles. This section presents this parameterization and highlights the resulting issue of Gimbal lock including a solution.

2.2.3.1 Euler Angles and Gimbal Lock

Euler angles describe a rotation in $\mathbb{R}^3$ by defining a rotation about each basis vector. Euler angles can be compactly represented as a vector $\vec{v} \in \mathbb{R}^3$ with individual components often denoted ϕ, θ, and ψ.

While Euler angles are both succinct and intuitive, several problems occur using this representation. Due to the order of rotations impacting the final result, one must apply rotations with a consistent and fixed order. In three dimensions, twelve orderings exist and can cause subtle implementation and interoperability problems when not carefully addressed and documented.

The worst problem of Euler angles arises when two axes are aligned in parallel. This results in the loss of one degree of freedom known as Gimbal lock. While all rotations can be represented with Euler angles, Gimbal lock causes issues in specific applications such as angle interpolation.[5]

2.2.3.2 Quaternions

Quaternions can be used to avoid Gimbal lock while still representing all possible rotations in a succinct format. In a purely mathematical sense, quaternions are a number system in the form $q = q_0 + q_1 i + q_2 j + q_3 k$ where i, j, k satisfy $i^2 = j^2 = k^2 = ijk = -1$. Quaternions are often represented as $q = (s, \vec{v})$ where the scalar $s = q_0$ and the vector $\vec{v} = [q_1\ q_2\ q_3]$.

Quaternions have several useful properties which allow for simplified calculation. Given a quaternion q, the quaternion conjugate is defined as

$$\bar{q} = (s, -\vec{v}) \tag{2.8}$$

The inverse can be written in terms of the conjugate as

$$q^{-1} = \frac{\bar{q}}{||q||} \tag{2.9}$$

To represent a rotation of θ about an arbitrary axis $e = [e_x\ e_y\ e_z]^T$, a quaternion q is defined as

$$\begin{aligned} q &= (s, \vec{v}) \\ &= \left(\cos\theta, \left[e_x \sin\frac{\theta}{2}\ e_y \sin\frac{\theta}{2}\ e_z \sin\frac{\theta}{2}\right]\right) \end{aligned} \tag{2.10}$$

[5] Consider the difference in angle and numeric representation between 180° and −180°. Within the context of animation where angle interpolation is necessary, using Euler angles to specify rotations create singularities at such points.

Composition of quaternions is defined using quaternion multiplication defined as

$$\begin{aligned} q_1 q_2 &= (s_1, \vec{v}_1) \cdot (s_2, \vec{v}_2) \\ &= (s_1 s_2 - \vec{v}_1 \cdot \vec{v}_2, s_1 \vec{v}_2 + s_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2) \end{aligned} \tag{2.11}$$

Using a quaternion q to rotate a vector $\vec{v}$ is defined as

$$\vec{v'} = q v \bar{q}, \tag{2.12}$$

where quaternion multiplication with a vector $\vec{v}$ is presented in Appendix A.2.

Due to the avoidance of Gimbal lock, compact representation, and relatively straight-forward implementation, quaternions will be used to represent the rotational component of poses within our system.

2.3 Multi-Resolution Surfel Mapping

This section presents the theoretical background of environment modelling with sensory input using multi-resolution surfel maps. In the context of our system, the sensory input consists solely of laser scanner range data further detailed in Section 2.1. By using the error characteristics of the sensor data and spatial modelling techniques, the mapping system can produce a model of the environment. Using optimization methods, this system can additionally provide localization estimates suitable for tracking refinement and navigation.

2.3.1 Octree Spatial-Partitioning

Due to the natural organization of spatial data, specialized data structures can be applied to improve efficiency in map processing. An octree is commonly used to partition data in three dimensional space and allows for efficient querying and updating of spatial data.

An octree organizes data by recursively partitioning space into eight equal volumes, where each volume is represented by a node in a tree data structure. Within each tree node, relevant data concerning points contained within the region is stored. These volumes are defined by the three axis-aligned splitting planes separating the three dimensional parent volume in half. As the splitting planes are not data dependent, query performance can be improved by precomputing node neighbors. Figure 2.5 shows this recursive division scheme by expanding only one child node at each level along with the corresponding tree structure.

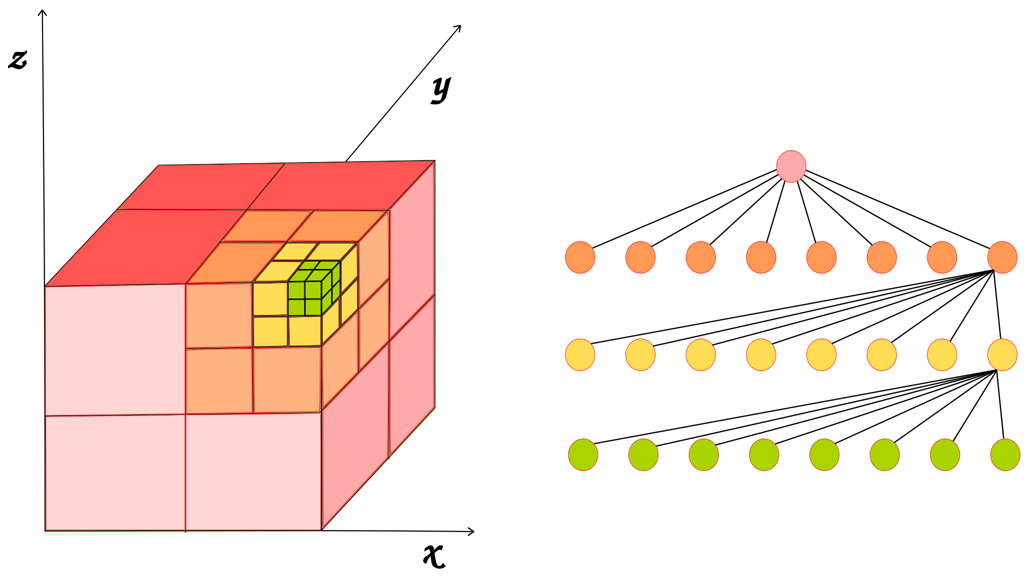


FIGURE 2.5: The spatial subdividing scheme of an octree showing a partitioned volume with corresponding tree nodes.

As parent nodes can maintain aggregate child data, querying an octree at a specific resolution consists simply of recursing to the depth necessary for the specified resolution. Additionally due to the tree structure of an octree, a modified binary search can be implemented allowing for efficient search of nodes. Opposite to search, when inserting data into an octree, a node is only split into its eight children if necessary. Thus unmeasured volume within the environment is not represented; a large benefit when compared to similar structures such as multi-resolution grids.

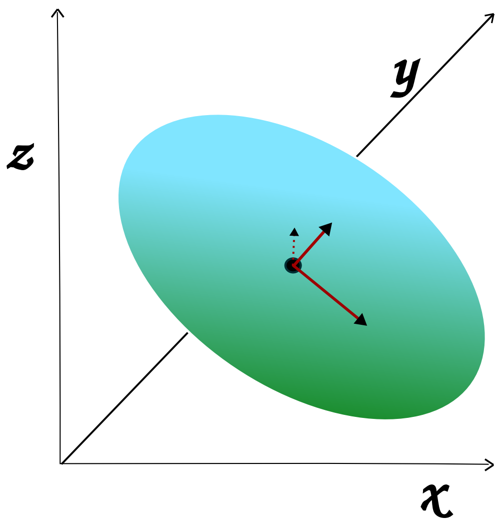


FIGURE 2.6: Three dimensional surfel with covariance matrix principle axes drawn. Solid principle axis - in horizontal XY plane. Dashed principle axis - vertical plane parallel to z axis.

2.3.2 Surface Elements - Surfels

Within the map structure, the smallest modeling unit is the surface element otherwise referred to as a surfel. A surfel is represented by a multi-variate Gaussian distribution modeling the probability of encountering a given measurement within the surfel [Stückler and Behnke, 2013]. Figure 2.6 shows an example surfel in three dimensions modeling only spatial data along with the principle axes.

Sensor data from the laser scanner consists of a three dimensional spatial component and a one dimensional remittance value, thus a four-dimensional Gaussian distribution is used to model the environment. The full specification of a surfel is:

$$\mu = [\mu_x, \mu_y, \mu_z, \mu_{\text{remittance}}], \Sigma$$

This distribution is approximated by the sample mean and covariance of previous sensor measurements within the surfel and thus must be iteratively updated using new sensor information. To maintain space and processing efficiency, a stable iterative method to calculate the data mean and covariance is necessary. Chan et al. [1979] present an accurate single-pass update scheme used within the mapping system. Maintaining point statistics $S(P) := \Sigma_{p \in P} p$ and $S^2(P) := \Sigma_{p \in P} pp^T$ point sets can be combined using the following update rule

$$S(P^{A \cup B}) \leftarrow S(P^A) + S(P^B) \tag{2.13}$$

$$S^2(P^{A \cup B}) \leftarrow S^2(P^A) + S^2(P^B) + \frac{\sigma \sigma^T}{N_A N_B (N_A + N_B)}, \tag{2.14}$$

where $N_{(.)} := \left| P^{(\cdot)} \right|$ and $\sigma := N_B S(P^A) - N_a S(P^B)$. Using these statistics, the Gaussian parameters can be calculated as $\mu(P) = \frac{1}{|P|} S(P), \Sigma(P) = \frac{1}{|P|-1} S^2(P) - \mu \mu^T$

To maintain numerical accuracy, a surfel is only valid when $|P| \geq 10$. Additionally, a surfel is no longer updated when $|P| \geq 10000^2$.

Within both mapping and localization applications, shape of the surfel has far-reaching impacts on data association and observation likelihoods. The three principle spatial axes of the normal distribution can be calculated using singular value decomposition (SVD). The principle axis having the smallest corresponding eigen vector has the least data variation and thus can be used as the surfel normal. This is illustrated in Figure 2.6.

2.3.3 Surfel Features

To aid in map registration, specifically data association, surface and reflectance features are extracted w.r.t. the local surfel neighborhood. Between each surfel s and the 26 neighbors s' (where surfel data exists), three shape features are extracted:

1. Angle between surfel normals $\angle(n, n')$
2. $\angle(n, \Delta\mu)$
3. $\angle(n', \Delta\mu)$

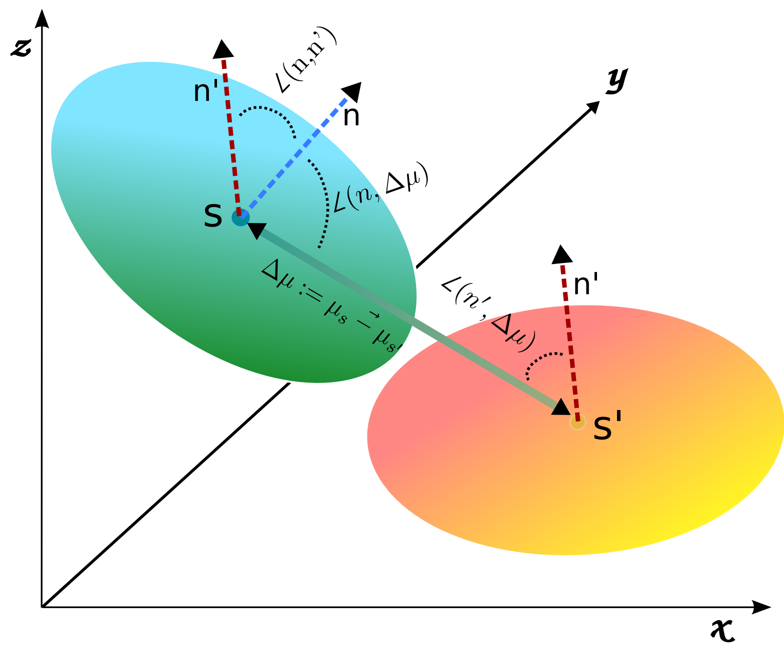


FIGURE 2.7: The three shape features extracted from a surfel's local neighbor.

where $\Delta\mu := \mu - \mu'$. The shape feature calculations are visualized in Figure 2.7. The reflectance feature is simply the difference in surfel remission. These shape features are classified into a three-bin histogram of positive, negative, and equal within a tolerance. Additionally a smoothing process is applied to reduce discretization effects by adding neighbor histograms with a factor $\gamma = 0.1$. The resulting histograms are then normalized by the number of surfel data points.

2.3.4 Map Representation

Multi-resolution surfel (MRS) maps model the environment using an octree to maintain voxel data. The modelling accuracy is naturally dependent upon both the minimum resolution of the octree and the mapping characteristics of a voxel. If each voxel stored one surfel, surfaces within the volume would be modelled irrespective of viewing direction. Even for relatively small volumes, this assumption is insufficient to accurately model even simple surfaces. Thus within a voxel, up to six surfels are maintained - each corresponding to the axis-aligned faces of the volume. Figure 2.8 shows an example of modeling a rectangular prism within a voxel using both one and six surfels. Using only one surfel, one cannot correctly

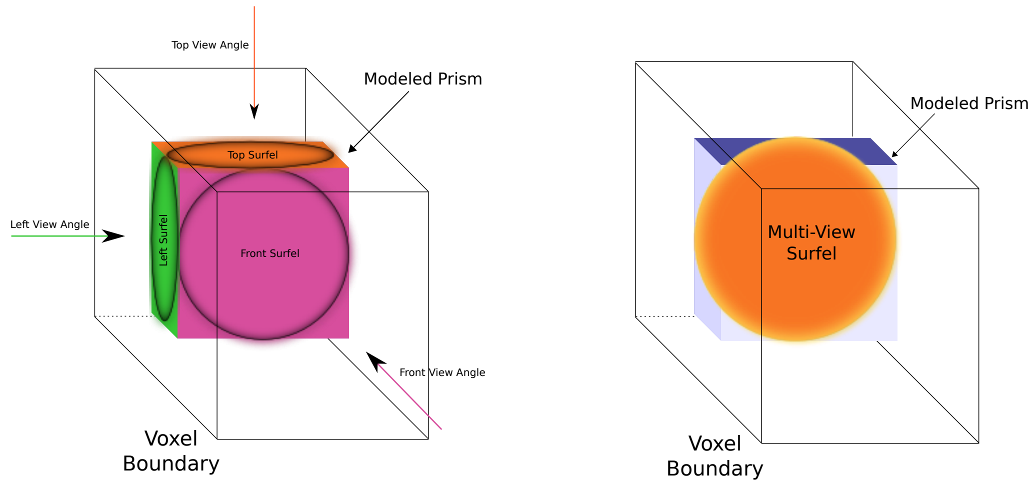


FIGURE 2.8: Left: Modeling of rectangular prism using six surfels per voxel. Right: Modeling of rectangular prism using one surfel per voxel.

assess the surface normal from multiple view directions. This causes problems in 3D scan registration and drivability assessment.

MRS maps model sensor data at a resolution according to the data and error properties of the sensor itself. This is possible by limiting the depth in the octree where a measurement is aggregated. By calculating this depth from sensor properties, the mapping system can be implicitly applied to various sensors by simply updating the error model.

A MRS map can be created from a three dimensional laser scan by inserting all endpoints into the octree and updating the encompassing voxel surfels statistical data. From the sensor orientation, the sensor to voxel view direction can be easily calculated so that the correct voxel surfel is updated. Due to environment structure, neighboring measurement endpoints are often contained within the same voxel. Using this property, neighboring data can often be aggregated into a single surfel update reducing the number of octree search queries.

2.3.5 Map Registration

Given two MRS maps, using registration techniques we can determine pose constraints between their map origins. In the context of 3D laser scans, this registration results in an estimation of the sensor origin with an arbitrary map frame - allowing for localization refinement as well as global map creation. Registration is performed by iterating two main steps: data association and pose refinement.

2.3.5.1 Data Association

Data association matches surfel pairs from a source and target MRS map for use during pose refinement. The association step is performed with fine to coarse resolutions until all surfels have an association candidate, only processing surfels with no child surfel associations. Consequently surfel associations are independent within a given resolution allowing for parallelization of this technique.

As map registration is an iterative process, surfels may or may not have a previous association. These two cases are explicitly handled using different methods. Surfel having no previous association use a volumetric query within the surfel neighbor to find association candidates. The query volume center is determined by the surfel mean transformed by the current registration estimate with the query size calculated as twice the size of the node depths resolution. Using the node depth resolution to determine query size allows for the implicitly adaptation of misalignments from coarse to fine resolutions.

The association candidate having the closest shape-reflectance feature-distance is selected for association only when thresholding this distance $d_f(s, s') \leq \tau$ with $\tau = 0.1$. The feature-distance d_f is calculated as the euclidean distance between the shape and reflectance histograms:

$$d_f(s, s') = \sum_{c \in \{\text{shape}, \text{reflectance}\}} |f_c - f'_c| \qquad (2.15)$$

2.3.5.2 MRS Map Observation Model

With a target MRS map m_t and three-dimensional laser scan data z, we model pose refinement as finding the pose x that maximizes the observation likelihood $p(z|x, m_t)$. Poses $x = (q, t)^T$ encompass a unit quaternion q and translation $t \in \mathbb{R}^3$. Using the scan data z to create a source MRS map m_s, the observation likelihood can be rewritten as:

$$p(m_s|x, m_t) = \prod_{(i,j)\in A} p(s_{s,i}|s_{t,j}) \tag{2.16}$$

where A is the set of surfel associations from Section 2.3.5.1, $s_{s,x}$ is surfel x in the source map m_s, and $s_{t,x}$ is surfel x in the target map m_t. As surfels are modelled as normal distributions, calculating the observation likelihood from Equation 2.16 is defined as:

$$\begin{aligned} p(s_{s,i}|s_{t,j}) &= \mathcal{N}(d_{i,j}(x); 0, \Sigma_{i,j}(x)) \\ d_{i,j} &:= \mu_{t,j} - T(x)\mu_{s,i} \\ \Sigma_{i,j} &:= \Sigma_{s,j} + R(x)\Sigma_{s,i}R(x)^T \end{aligned} \tag{2.17}$$

where $T(x)$ is the homogeneous transform matrix from pose estimate x and $R(x)$ is the corresponding rotation matrix.

2.3.5.3 Pose Optimization

To determine the pose estimate x of the observation map m_o, we optimize the logarithm of the observation likelihood from Equation 2.16 resulting in the loss function

$$L(x) = \sum_{a\in A} \log|\Sigma_a(x)|) + d_a^T(x)\Sigma_a^{-1}(x)d_a(x), \tag{2.18}$$

where A is the complete set of surfel associations, $a = (i, j)$ is the surfel association pair, and $\Sigma_a = \Sigma_{i,j}$, $d_a = d_{i,j}$ from equation 2.16.

The pose optimization process is performed in two stages. Levenberg-Marquardt (LM) optimization quickly provides a fast estimate used to initialize Newton's method for refinement.

2.3.5.4 Pose Estimate Uncertainty

The uncertainty of the registration estimate can be calculated using a closed-form approximation

$$\Sigma(x) = \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1} \frac{\partial^2 L}{\partial s \partial x} \Sigma(s) \frac{\partial^2 L}{\partial s \partial x}^T \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1}, \tag{2.19}$$

where x is the pose estimate, s denotes associated surfels from the target and observation map, and $\Sigma(s)$ is the covariance of all surfel associations. This closed-form solution was derived by Censi [2007] and is further detailed in his work.

2.3.6 Probabilistic Optimization

Pure map-to-map registration could allow for pose tracking and the creation of a global allocentric map. Using a mapping scheme where the newest scan is aligned to the previous scan, forming a change of alignment poses, is very prone to accumulated drift. To mitigate these affects, local loop closure is performed by registering each new observation to all maps within a tolerance distance. Including all registration estimates creates several possible paths from an arbitrary reference frame to a scan, each with a potentially different estimate.

The graph optimization library g2o is used to calculate a consistent pose estimate for each scan given the entire set of alignments.[6] For each 3D laser scan, a keyview is extracted representing the scan coordinate frame w.r.t the first scan coordinate

[6] See Kuemmerle et al. [2011] for an in-depth overview of the g2o framework.

frame. This coordinate frame is calculated using the alignment procedure from Section 2.3.4 and implies a geometric constraint between keyviews. Considering all keyviews as vertices $v \in \mathcal{V}$ and each alignment estimate an edge $e_{i,j} \in \mathcal{E}$, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is maintained. Keyview pose estimates are calculated with the g2o framework given the input graph $\mathcal{G}$.

The resulting keyview estimates have been determined from global optimization and thus allow for the creation of global allocentric maps of the environment.

2.4 Stochastic Filtering

This sections presents the theoretical background of stochastic filtering and mathematically defines the localization problem in the context of robotic motion. Using the general problem definition, a discrete-form solution to the estimation problem is presented in the form of Bayesian filtering. Using the concepts of Bayesian filtering, I describe the Kalman filter and Monte-Carlo filtering methods. This leads to the particle filter, an estimation method that allows for handling of non-linearity in observation and motion and can additionally be used to solve the global localization problem.[7]

Filtering can be described as the determination of a quantity of interest at time t, using a time series of unreliable or noisy observations. Within the context of pose filtering, we are not only interested in an individual quantity of interest but a vector referred to as *state*. Filtering the state of stochastic systems will be referenced as stochastic state estimation and allows for an exact formulation of the tracking problem.[8]

We model a stochastic system using random variables $X \in \mathcal{X}$, $Z \in \mathcal{Z}$ representing the true system state and observation respectively. Applying this nomenclature to

[7] Global localization is the process of determining the absolute pose of an object within an environment given no previous pose knowledge. This problem is considered harder to solve than localization as there is a reduction from global localization to localization however the reverse is not necessarily true.

[8] Stochastic systems cannot be accurately modeled using currently available tools due to both complexity and the non-deterministic nature of the system.

the discrete time-domain, we can pose the tracking problem as the estimation of the system state x_t at time t given previous state estimates $x_{1:t-1} = \{x_1, x_2, ..., x_{t-1}\}$ and actual observations $Z_{1:t} = \{Z_1, Z_2, ..., Z_t\}$.

To model the time-discrete system, we assume there exists a transition function relating a system state x_{t-1} to the successor state x_t having additional system noise.[9] Equation 2.20 shows state propagation where f is the transition function and ϵ_t denotes the constant system noise. [10] We additionally model the expected observation z_t of the state x_{t-1} by Equation 2.21 where h is an observation function and δ_t measurement noise.

$$x_t = f(x_{t-1}) + \epsilon_t \tag{2.20}$$

$$z_t = h(x_t) + \delta_t \tag{2.21}$$

2.4.1 Bayesian Filtering

Bayes' rule, discovered by mathematician Thomas Bayes in the 18^{th} century, allows one to express a posterior probability $p(X|Y)$ in terms of the *prior* probability $p(X)$, the *evidence* $p(Y)$, and the *likelihood* $p(Y|X)$.

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)} \tag{2.22}$$

Equation 2.22 shows the Bayes' rule formulation that can be used to derive a *recursive Bayes filter.* If we assume the stochastic system can be modelled by a first-order Markov process, the recursive Bayes filter provides a solution to the state estimation problem. [11]

[9]System noise may or may not be 0 and can additionally vary with the time parameter.

[10]Function f may or may not be linear. The linearity of f impacts which Bayesian filtering implementation can be used for accurate state estimation.

[11]A first-order Markov process assumes the successive state of a system is only dependent upon the current system state.

Assuming we can model a system by a first-order Markov process in the context of filtering is known as the Markov assumption. Here we assume each measurement z_n is independent of previous measurements $z_1, z_2, \cdots, z_{n-1}$ given the current system state x_n.

Using previously presented notation concerning the system state x and measurement z in a discrete time series, we can insert our quantity of interest into Bayes' rule shown in Equation 2.23.

$$\begin{aligned} p(x_t|z_{1:t}) &= \frac{p(z_{1:t}|x_t)p(x_t)}{p(z_{1:t})} \\ &= \frac{p(z_t, z_{1:t-1}|x_t)p(x_t)}{p(z_t, z_{1:t-1})} \\ &= \frac{p(z_t|z_{1:t-1}, x_t)p(z_{1:t-1}|x_t)p(x_t)}{p(z_t|z_{1:t-1})p(z_{1:t-1})} \\ &= \frac{p(z_t|z_{1:t-1}, x_t)p(x_t|z_{1:t-1})p(z_{1:t-1})p(x_t)}{p(z_t|z_{1:t-1})p(z_{1:t-1})p(x_t)} \\ &= \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \end{aligned} \tag{2.23}$$

To derive an iterative update rule, all components of the resulting formula derived in Equation 2.23 must only depend on the previous state estimate $p(x_{t-1}|z_{t-1})$. By inserting a marginalizing integral we arrive at the recursive update rule for the Bayes filter shown in Equation 2.24. [12]

$$\begin{aligned} p(x_t|z_{1:t}) &= \frac{p(z_t|x_t)\int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1}}{\int p(z_t|x_t)p(x_t|z_{t-1})dx_{t-1}} \\ &= \frac{p(z_t|x_t)\int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1}}{\int p(z_t|x_t)\left[\int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1}\right]dx_t} \\ &\propto \underbrace{p(z_t|x_t)}_{\text{filtering}}\underbrace{\int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1}}_{\text{prediction}} \end{aligned} \tag{2.24}$$

[12] See Appendix A.3 for an introduction to working with probability densities.

2.4.2 Kalman Filtering

A Kalman filter is an implementation of Bayesian filtering where the state propagation function and the measurement function are modelled as linear systems. This allows f from Equation 2.20 to be rewritten as

$$x_t = Ax_{t-1} + \epsilon_t, \tag{2.25}$$

where A is a state propagation matrix. We model the function h from Equation 2.21 as

$$z_t = Hz_{t-1} + \delta_t, \tag{2.26}$$

where H is a measurement matrix. The method uses iterative prediction and correction at each time step to calculate a state estimate x_t.

Using the additional constraint that error is Gaussian distributed allows for a pose estimate x_t that maximizes the posterior probability $p(x_t|z_{1:t})$ - assuming system linearity [Kalman, 1960].

Linear state propagation and measurement functions are often inadequate to model many real-life situations. Due to the Explorer platform using skid-drive steering, the propagation model is certainly non-linear. Additionally, laser scanner data is inherently non-linear due to the complex environment structure. A standard Kalman filter is therefore unusable for our application.

2.4.3 Non-linear Kalman Filtering

Methods have been proposed to deal with non-linearity in both state propagation and measurement.

The National Aeronautics and Space Administration (NASA) was among the first to publish work concerning non-linearity with Kalman Filtering [Smith et al., 1962]. The *Extended Kalman Filter* uses multivariate Taylor series expansion to compute partial derivatives in the form of a Jacobian. The Jacobian is evaluated at each time step and used for prediction updates. This method assumes that the non-linear functions are differentiable.

Similarly, the *Unscented Kalman Filter* deals with non-linear state and measurement functions through approximation, however it is better equipped to handle highly non-linear functions than the EKF. Sampling points, called Sigma points, are selected using a sampling method known as the *unscented transform.* The sigma points are propagated using the non-linear functions and the resulting values are used to calculate a sample mean and covariance [Julier and Uhlmann, 1997].

While these methods can handle non-linearity in both measurement and observation, large non-linearity can cause filter instability. Additionally, due to the observation model presented in Section 3.3.2, a direct pose estimate from a single laser scan line cannot be calculated. Future work presented in Section 5.2 details the possibility of using iterative closest point (ICP) methods to determine a direct pose estimate from scan lines.

2.4.4 Particle Filtering

Particle filtering offers an alternative application of recursive Bayesian filtering capable of handling non-linearity in both propagation and measurement models. Within the context of localization using laser scanners, particle filtering does not require a direct state (pose) measurement, but simply requires calculation of a measurement likelihood $p(z_t|x_t)$.

Contrary to Kalman filtering where the posterior $p(x_t|z_{1:t})$ is approximated using a mean and covariance of a normal distribution, a particle filter uses Monte-Carlo methods described in the next section to approximate the pose x_t maximizing the posterior.

Sequential importance resampling (SIR) allows for approximation of the posterior distribution using a set of weighted particles [Gordon et al., 1993]. SIR is an iterative implementation of *importance sampling* and useful as the true underlying distribution of pose cannot be directly sampled. To describe the particle filter at time t, we consider a set of N particles

$$\mathcal{P} = \{(x_t^{(n)}, w_t^{(n)}) : n \in 1, \cdots, N, \tag{2.27}$$

where $w_t^{(n)}$ is the particle's importance weight, and $x_t^{(n)}$ is the particle's estimate.

The following sections describe the foundations of Monte-Carlo sampling and the SIR filter in the context of particle filtering. Additionally, the bootstrap filter is presented which is the basis of our particle filtering algorithm.

2.4.4.1 Monte-Carlo Sampling

Monte-Carlo sampling attempts to approximate the expectation of a random variable using a set of random samples drawn independently from the underlying probability distribution.

If we consider a random variable $\mathcal{X}$ having probability density function $p_{\mathcal{X}}(x)$, the expected value of a function f of $\mathcal{X}$ is

$$\mathbb{E}[f(\mathcal{X})] = \sum_{x \in \mathcal{X}} f(x) p_{\mathcal{X}}(x), \tag{2.28}$$

assuming $\mathcal{X}$ is discrete.

This expectation can be approximated using Monte-Carlo sampling. Using n samples $\{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\} : x \in \mathcal{X}$ according to the PDF $p_{\mathcal{X}}(x)$, the expected value of f can be estimated as

$$\widetilde{\mathbb{E}}[f_n(\mathcal{X})] = \frac{1}{n} \sum_{i=1}^{n} g(x_i). \tag{2.29}$$

This is also known as the *Monte-Carlo Estimator* of $f(\mathcal{X})$ and has the useful properties that as $n \to \infty$ the error $\left| \widetilde{\mathbb{E}}[f_n(\mathcal{X})] - \mathbb{E}[f(\mathcal{X})] \right| \to 0$.

2.4.4.2 Importance Sampling

As the distribution $p_{\mathcal{X}}(x)$ often cannot be directly sampled, importance sampling provides a way to draw samples from a related distribution and still approximate the expected value of $\mathcal{X}$. This distribution $q_{\mathcal{X}}(x)$ is called the proposal distribution and has the property that $q_{\mathcal{X}}(x) = 0 \implies p_{\mathcal{X}}(x) = 0$. By inserting a neutral term into Equation 2.28 and reordering terms we can calculate the expected value of $f(\mathcal{X})$ as

$$\begin{aligned}\mathbb{E}[f(\mathcal{X})] &= \sum_{x\in\mathcal{X}} f(x)p_{\mathcal{X}}(x)\frac{q_{\mathcal{X}}(x)}{q_{\mathcal{X}}(x)} \\ &= \sum_{x\in\mathcal{X}} f(x)\frac{p_{\mathcal{X}}(x)}{q_{\mathcal{X}}(x)}q_{\mathcal{X}} \\ &= \sum_{x\in\mathcal{X}} f(x)W_{\mathcal{X}}(x)q_{\mathcal{X}}(x),\end{aligned} \tag{2.30}$$

where $W_{\mathcal{X}}(x) = \frac{p_{\mathcal{X}}(x)}{q_{\mathcal{X}}(x)}$ is the importance weight according for the difference between the distributions $p_{\mathcal{X}}(x)$ and $q_{\mathcal{X}}(x)$. Rewriting Equation 2.29 using importance weighting, the Monte-Carlo Estimator using the distribution $q_{\mathcal{X}}(x)$ for sampling, can be calculated as

$$\widetilde{\mathbb{E}}[f_n(\mathcal{X})] = \frac{1}{\sum_{i=1}^{n} W_{\mathcal{X}}^{(i)}(x_i)} \sum_{i=1}^{n} W_{\mathcal{X}}^{(i)}(x_i)g(x_i), \tag{2.31}$$

where

$$W_{\mathcal{X}}^{(i)}(x_i) = \frac{p_{\mathcal{X}}(x_i)}{q_{\mathcal{X}}(x_i)}. \tag{2.32}$$

2.4.4.3 Sequential Importance Sampling

Sequential importance sampling (SIS) filter is the foundation of the SIR filter and offers a framework for combining recursive estimation with importance sampling.

We reformulate the posterior $p(x_{0:t}|z_{1:t})$ using the Markov assumption as

$$\begin{aligned}
p(x_{0:t}|z_{1:t}) &= p(x_t|x_{0:t-1}, z_{1:t})p(x_{0:t-1}|z_{1:t}) \\
&= p(x_t|x_{0:t-1}, z_{1:t})p(x_{0:t-1}|z_{1:t-1}) \\
&= \frac{p(z_{1:t}|x_t, x_{0:t-1})p(x_t|x_{0:t-1})}{p(z_{1:t}|x_{0:t-1})}p(x_{0:t-1}|z_{1:t-1}) \\
&= \frac{p(z_t|\cancel{z_{1:t-1}}, x_t, \cancel{x_{0:t-1}})\cancel{p(z_{1:t-1}|x_t, x_{0:t-1})}p(x_t|x_{\cancel{0:}t-1})}{p(z_t|z_{1:t-1}, x_{0:t-1})\cancel{p(z_{1:t-1}|x_{0:t-1})}}p(x_{0:t-1}|z_{1:t-1}) \\
&= \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{1:t-1}, x_{0:t-1})}p(x_{0:t-1}|z_{1:t-1})
\end{aligned} \tag{2.33}$$

Similarly, we rewrite the proposal as

$$q(x_{0:t}|z_{1:t}) = q(x_t|x_{0:t-1}, z_{1:t})q(x_{0:t-1}|z_{1:t}) \tag{2.34}$$

Using these formulations, the importance weight definition from Equation 2.32, and the Markov assumption, we can compute recursive importance weights as

$$\begin{aligned}
W_t^{(i)} &= \frac{p(x_{0:t}^{(i)}|z_{1:t})}{q(x_{0:t}^{(i)}|z_{1:t})} \\
&\propto \frac{p(z_t|x_t^{(i)})p(x_t|x_{t-1}^{(i)})p(x_{0:t-1}^{(i)}|z_{1:t-1})}{q(x_t^{(i)}|x_{0:t-1}^{(i)}, z_{1:t})q(x_{0:t-1}^{(i)}|z_{1:t})} \\
&\propto \frac{p(z_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{0:t-1}^{(i)}, z_{1:t})}W_{t-1}^{(i)} \\
&\propto \frac{p(z_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)}, z_t)}W_{t-1}^{(i)}
\end{aligned} \tag{2.35}$$

2.4.4.4 Sequential Importance Resampling Filter

The SIR filter estimates the posterior probability using the SIS recursive algorithm. Thus, this filter has the property of convergence to the true posterior as the number of particles $n \to \infty$, assuming the proposal and posterior distributions are sufficiently close.

After many iterations of the SIS filter, many particles have a very small importance weight and thus provide little benefit to the posterior approximation. This is due to the Monte-Carlo sampling of the proposal generates poses that diverge from the true pose. This process is known as *particle depletion*.

To avoid particle depletion, an extra resampling step is performed which re-initializes the set of particles maintained by the filter. The new particles are independently drawn from the current particle set with a selection probability proportional to the importance weights. In this way, particles having a large importance weight will be represented by additional particles in the new set. Particles having small importance weights are likely to be removed by the resampling set. This process focuses particles to areas that better represent the posterior distribution.

2.4.4.5 The Bootstrap Filter

As previously explained, the choice of proposal distribution impacts the accuracy of the algorithm and given a bad proposal can result in divergence of the filter from the true state. The optimal proposal distribution is the posterior distribution itself, however it is often not possible to sample from the distribution.

The Bootstrap filter uses the state propagation density $p(x_t|x_{t-1})$ as the proposal distribution [Gordon et al., 1993]. This choice allows for easy sampling and shows improved results over a random-walk proposal distribution.

By selecting the proposal density $q(x_t^{(i)}|x_{t-1}^{(i)}, z_t) = p(x_t|x_{t-1})$, Equation 2.35 can be simplified to

$$W_t^{(i)} \propto p(z_t|x_t^{(i)})W_{t-1}^{(i)}. \tag{2.36}$$

2.4.4.6 The Particle Filter Algorithm

The previous sections have presented the theory behind recursive sampling theory and stochastic filtering. To present an overview of particle filtering, the general algorithm pseudocode is given in Algorithm 1.

```
Initialization:
  Draw $m$ samples $x_0^{(i)} \sim p(x_0)$
  Set importance weights $W_t^{(i)} = \frac{1}{m}$
for time step $t = 0$ to $n$ do
    for $i = 1$ to $m$ do
        Sampling: $x_t^{(i)} \sim p(x_t|x_{0:t-1}^{(i)})$
        Importance Weighting: $W_t^{(i)} \leftarrow p(z_t|x_t^{(i)})W_{t-1}^{(i)}$
    end
    for $j = 1$ to $m$ do
        Resampling: draw $x_t^{(j)} \propto W_t^{(j)}$
    end
end
```

Algorithm 1: Pseudocode for general particle filtering.

2.5 Drivability Assessment and Path Planning

Within the navigation framework, drivability assessment and path planning must only consider the environment from a global perspective. Low-level planning and robot control are considered external systems, thus the main objective of our planning is to generate trajectories to goal positions. The execution of these trajectories is the topic of other research.

Due to the usage of the navigation system, discussed in Section 3.1, autonomous exploration of environments is not a requirement. While the robot should handle planning to unknown or occupied space, a remote human user can guide exploration to allow for successful trajectory planning. Although the environment is fully three dimensional, the target mobile platform must move locally along planes when considering roll and pitch limitations (common to all rover-type platforms). Therefore, planning methods can be simplified to use a grid-based structure.

To provide adequate trajectories to lower level robot control systems, we must handle planning to unknown space as well as provide safety guarantees that the generated path is traversable. From these requirements, simple but effective graph planning approaches can be successfully used for all planning purposes.

This section gives a general description on graph generation and planning methods used within the navigation system.

2.5.1 Region Growing

In order to apply graph planning methods, we must first generate a traversability graph describing the potential drivable space of the robot. Using a traversability graph as opposed to implicit graph structure within an MRS map saves computation time due to a reduced state space as well as separates map and planning structure.

Wave front region growing is an efficient method of determining this “drive space” and has guaranteed convergence. For our usage, the wave front algorithm acts as a simple space exploration and graph building tool; more advanced uses of wave front exist [Stoyanov et al., 2010].

The wave front algorithm maintains a list of unprocessed cells, initially containing cells deemed “drivable”. Until this list is empty, each cell is removed from the list, processed, and all unvisited cell neighbors are added to the processing queue. During processing cell costs are computed based on the characteristics discussed in section 3.4. If a cell is deemed “not drivable”, no neighboring nodes are added to the processing queue effectively stopping propagation from this location. The pseudo-code for this algorithm is shown in Algorithm 2.

```
input  : Environment map, Initial drive space
output: Drivability Grid
OpenList := { Initial drive space }
ClosedList := {}
while OpenList ≠ {} do
    Current := Front(OpenList);
    ClosedList := ClosedList + { Current };
    CalculateDriveCost(Current);
    if Drivable(Current) then
        for Neighbor in Neighbors(Current) do
            if NOT in ClosedList then
                OpenList := OpenList + { Neighbor };
            end
        end
    end
end
```

Algorithm 2: Pseudocode for region growing using wavefront propagation.

2.5.2 A* Graph Planning

The A* algorithm is an efficient and easy to implement best-first-search method of path planning within weighted graphs using a heuristic. Presented by Hart et al. [1968], A* can be considered a modified version of Dijkstra's algorithm where an admissible heuristic is used to improve the order of node expansion. The improvement of expansion order allows the A* algorithm to drastically outperform the standard Dijkstra's algorithm due to the large branching factor of three dimensional state spaces.

Using standard terminology, A* computes a path along graph edges between a source and target node. Nodes represent vertices within a graph while edges refer to the same graph counterpart. The graph may have both edge and node costs with the constraint that costs must be positive to ensure optimality of the path.

The algorithm works by first initializing a selection queue with the source node. Next, the node having the lowest *expected* total cost to the target node from the queue is selected for search expansion. If the selected node is the target node, searching is finished and the path can be reconstructed from the expansion order. Otherwise the node is considered “explored” and all neighbor nodes that are not yet “explored” are added to the node selection queue. If the neighbor node has already been placed in the selection queue, the node in the queue is updated to have the lower expected cost.

The key to the efficiency of A* comes from determining the *expected* cost of a node to the target node. The expected cost of a node is calculated using the heuristic function which must be admissible. In this context an admissible heuristic means the function may not overestimate the distance to the goal. A heuristic which is always 0 fulfills this constraint, however transforms the A* algorithm into an implementation of Dijkstra’s algorithm that stops once the goal node is reached. Thus one must be careful to find a heuristic that fulfills the admissible constraint but accurately describes the search space. In robot navigation planning, the Euclidean or Manhattan distance is an often used heuristic.

Figure 2.9 shows paths computed using the A* algorithm in a two dimensional grid where black cells indicate impassable walls. All cells that have been expanded during the algorithm execution are colored with varying shades of red, where the lighter the shade the lower the total cost estimate is from the cell. The calculated path is highlighted in blue. The Manhattan distance was used as the heuristic and transitions are allowed to the Moore neighborhood. As one can see, the environment has a large impact on the number and positions of nodes that are expanded.

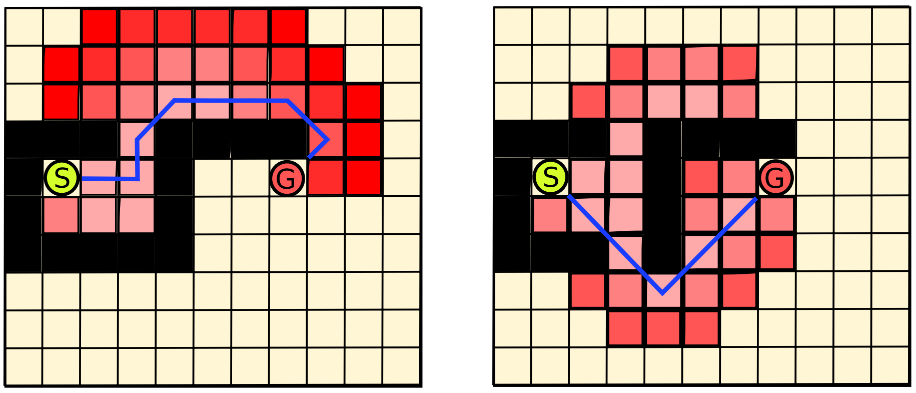


FIGURE 2.9: Example A* path planning in two similar grid environment. “S” is the starting cell and “G” is the goal cell. Cells colored in red have been expanded by the A* algorithm where shading indicates expansion order. Lighter means earlier as deeper means later.

Using a heuristic, the total cost from a node is the sum of the actual cost to reach the node and the heuristic cost h. Using this paradigm, the pseudocode for the A* algorithm is shown in Algorithm 3. Within this algorithm, the “OpenList” maintains all the possible selection queue while the “ClosedList” tracks all “explored” nodes. Maintaining a “ClosedList” ensures no node is processed more than once.

```
input  : Traversability Graph, Starting Node, Target Node
output: Path to Target Node
OpenList := { Starting Node };
ClosedList := {};
while OpenList ≠ {} do
    Current := LowestTotalCostEstimate(OpenList);
    if Current is TargetNode then
        return ReconstructPath(Current);
    end
    ClosedList := ClosedList + { Current };
    for Neighbor in Neigbors(Current) do
        if Neighbor in ClosedList then
            Continue;
        else if Neighbor in OpenList then
            TotalCostEstimate = Cost(Neighbor);
            Update OpenList with minimum TotalCostEstimate;
        else
            OpenList = OpenList + { Neighbor };
        end
    end
end
```

Algorithm 3: Pseudocode for the A* path finding algorithm.

Chapter 3

Method

3.1 System Operation

The presented SLAM and navigation system was developed for the Explorer robot platform at the University of Bonn, specifically for the DLR Spacebot Cup competition. The Spacebot Cup is an autonomy challenge for rover-type robots modelling conditions expected during planetary exploration. The competition involved completion of several tasks including autonomous mapping, navigation, object recognition, and object grasping.

Due to the large amount of divided effort to produce such a robust system, several externally developed components were available for usage and testing of SLAM and navigation. Visual odometry estimates are provided by a fovis implementation using RGB-D cameras [Huang et al., 2011]. The odometry estimate is used by the localization system for an improved motion model. Additional components implementing low-level planning and control have been provided. The competition also permits giving the robot a “rough” pre-planned trajectory, thus mitigating the explicit need for autonomous exploration. Therefore, the navigation system must only consider high-level planning methods to provide robot trajectories.

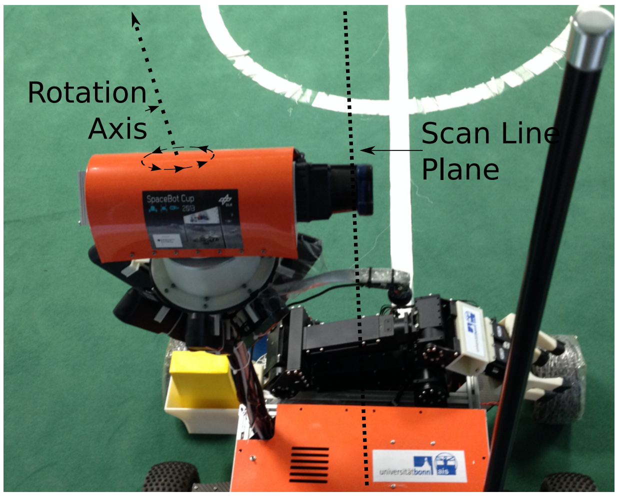


FIGURE 3.1: The Hokuyo laser scanner mounted on the Explorer robot with rotation axis and laser scan line plane indicated.

3.1.1 Map Initialization

To initialize the SLAM and navigation system, an initial 3D laser scan of the environment is recorded and processed by the mapping system. This initial scan defines the map coordinate frame where all localization and mapping scan poses are represented. This initial scan also provides mapping data for localization allowing for pose tracking.

3.1.2 Laser Configuration

The Hokuyo laser scanner is mounted above the robot for improved sensor visibility on a slip-ring attached to a servo. The slip-ring allows for continuous electrical power and data communication during continuous rotation and the servo provides rotational speed control and orientation readings for scan line aggregation.

The laser is rotated about the vertical axis with each laser scan line plane parallel to this line. This allows for good coverage of the environment with minimal measurement dead-zones. With this configuration, only the space directly above and below the robot cannot be measured. Figure 3.1 shows the laser mounted on the Explorer platform indicating the rotation axis and scan line plane.

To accurately map large environments, sensor measurements must be sufficiently dense. To ensure dense data from 3D laser scans, the laser is rotated at a relatively slow speed of $\frac{1}{15}$ Hz during 3D scanning. However, during localization we wish to maximize visible space per time unit and rotate the laser at a higher speed of 1 Hz.

3.1.3 SLAM Operation

Stop-and-go mapping is used for SLAM operation. This process requires an iterative process of 3D laser scans with a stationary pose. Assuming the map has been initialized as described in Section 3.1.1, a 3D scan is recorded during a “stop”. After the mapping system has completed processing the newest mapping data, the robot may “go” to a new region of interest for recording a new 3D scan. During the “go” stage, laser range measurements are processed by the localization system. The next 3D scan registration guess is initialized using the localization system guess after stopping. This process is repeated until an environment has been sufficiently mapped.

3.2 Mapping Error Model

The error properties of the sensor can be captured by varying the map resolution of points added to an MRS map. Through this same method, the reduction in point density due to measurement depth can also be mitigated. This section presents the error model used by the MRS map for the Hokuyo UTM-30LX-EW laser scanner.

To properly capture the sensor data properties, we must consider all possible sources of measurement error and point dispersion. Using the sensor configuration shown in Section 3.1.2 and considering three dimensional mapping, there exist three sources of consideration. The depth error is modelled by the laser scanner sensor properties provided by Hokuyo. Dispersion in the vertical axis comes from the angular resolution of the laser scanner. Dispersion in the horizontal axis comes from rotation and is dependent upon the laser scanner update rate and rotation speed. By calculating the correct depth dependency to mapping resolution for each axis and then using the least precise value, we can conservatively model the sensor error properties.

According to the laser specification provided by Hokuyo, the Hokuyo UTM-30LX-EW has an error of $\pm 30mm$ in the range of $0.1m - 10m$ and $\pm 50mm$ in the range of $10m - 30m$ [Hokuyo Autonomatic CO., 2012]. From this error specification we consider the error to be linear in distance and compute a worse case depth-error constant of $k_d = 0.00027$.

The vertical dispersion of measurement data depends solely on the angular resolution of the laser scanner. According to the laser specification, the Hokuyo UTM-30LX-EW has an angular resolution of 0.25°. By computing the distance between adjacent measurements Δm, both measured at the same distance d, we can compute the vertical error constant as $\frac{d}{\Delta m}$. Using the law of cosines and an angular resolution of 0.25°, we calculate the vertical error constant as $k_v = 0.00436$.

Horizontal dispersion of the measurement data depends on the rotation speed and laser update rate of 40 Hz. During 3D scanning, the laser scanner is rotated at a frequency of $\frac{1}{15}$ Hz. Thus the angular resolution of scanning in the horizontal plane is 0.6°. Using the same method for vertical dispersion we calculate a horizontal error constant of $k_h = 0.01047$.

By selecting the largest error constant, we arrive at a conservative error model for 3D laser scans. Thus the constant $k_h = 0.01047$ is used to determine the voxel resolution of measured points. Given a measurement with distance d, the voxel resolution is proportional to $log_2 (dk_h)$.

3.3 Localization

A particle filter is used for real-time 6D pose localization within an allocentric map. The localization estimate is used to initialize the map-to-map registration for all 3D laser scans excluding the initial scan, extending the global map and thus further improving localization. A particle filter implementation was selected due to the uncomplicated handling of a non-linear observation model. Considering other Bayesian filtering implementations that allow for similar non-linear processing such as an Extended Kalman Filter (EKF), a particle filter allows for a simplified scan-line-to-map observation model and the ability to solve the global localization problem.

3.3.1 State Propagation Model

Robot state transition is modelled using a simple time-discrete linear dynamics system (LDS). The state estimation problem is posed as the estimation of the full 6-DoF configuration of the robot pose $x_t = (q, t)^T$ having orientation q represented as a quaternion and translational component $t \in \mathbb{R}^3$. From an external system, odometry information is given as input to calculate a reasonable state propagation estimate assuming Gaussian noise in both translational and rotational components.

To simplify additive noise calculations, the pose orientation q is converted to Euler angle form denoted $r(x)$. For consistency, the translation part of pose x is indicated by $t(x)$. Odometry input is represented by the relative transform between the current and previous update step $o_t = (\Delta q, \Delta t)^T$ where the orientation delta is again converted to Euler angles for processing. Odometry information is calculated by an external system. The entire LDS update is calculated as

$$\begin{aligned} t(x_t) &= t(x_{t-1}) + t(o_t) + \mathcal{N}(n; 0, \Delta t \Sigma_{t(o_t)}) \\ r(x_t) &= r(x_{t-1}) + r(o_t) + \mathcal{N}(n; 0, \Delta t \Sigma_{r(o_t)}) \end{aligned} \tag{3.1}$$

where Δt is the time since the last motion model update, $\Sigma_{t(o_t)}$ and $\Sigma_{r(o_t)}$ represent the assumed noise covariance for odometry in translational and rotational parts per time unit respectively.

After the update step is performed, the Euler angle vector $r(x_t)$ is converted to a quaternion giving the final state estimate x_t.

3.3.2 Scanline Observation Model

Within the mapping system, full 3D laser scans are used to determine a refined pose estimate using alignment and global graph optimization techniques. Using this method for real-time tracking has many deficiencies mostly related to the maximum update rate of sensor data. Due to the Hokuyo laser scanner data-rate of 40Hz, 3D scans dense enough for map-to-map registration can take seconds and requires the robot to stop; thus we cannot rely on map registration for real-time pose tracking.

To overcome this limitation I propose using individual laser scan-lines to perform observational updates, resulting in a theoretical localization rate of 40 Hz. As each scan-line does not contain dense enough information for accurate map registration, we must use the structure of the particle system to determine an observation update. Transforming the laser scan-line by each particle's pose estimate allows for a simple end-point model calculation of observation likelihood against the global localization map.

Given a laser scanline $Z = \{z_i\}^n \in \mathbb{R}^3$ with n endpoint measurements and a target localization map m_t, we can calculate the observation likelihood of a particle with pose x as,

$$\begin{aligned} p(Z|x, m_t) &= \prod_{i=1}^{n} p(z_i|x, A(z_i, x, m_t)) \\ A(z_i, x, m_t) &= argmin_{s_t} d_{\text{plane}}(s_t, T(x)z_i) \end{aligned} \tag{3.2}$$

where $A(z_i, x, m_t)$ associates the transformed measurement endpoint $z_{\text{transformed}} = T(x)z_i$ to the target map surfel minimizing the plane distance $d_{\text{plane}}(s_t, T(x)z_i)$.

The surfel association is calculated by first performing a volumetric query centered at the transformed measurement endpoint to find surfel candidates. The size of the query volume is dependent upon the measurement's distance to the sensor using the method described in Section 3.2. For each association candidate, the

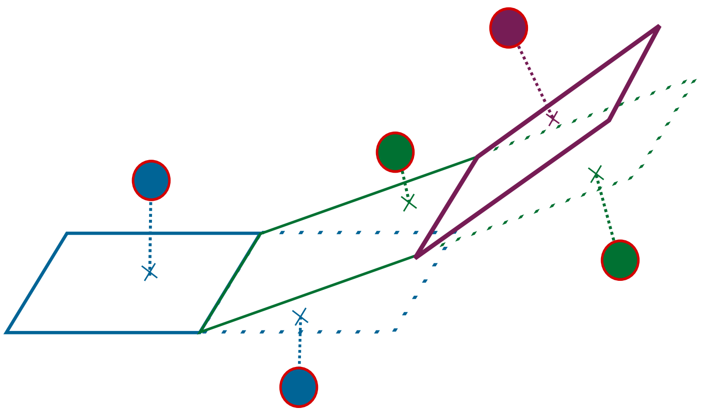


FIGURE 3.2: Illustration of the association method used to associate scanline endpoints to surfels based on the plane distance metric. Three surfel planes are shown along with five measurement endpoints. Each measurement endpoint is color-coded to their associated surfel from minimum plane distance.

distance between the surfel's approximate plane defined by the surfel mean and normal and the measurement endpoint is calculated. The surfel with the smallest plane-to-measurement distance is used to calculate the likelihood following

$$\begin{aligned} p(z_i|A(z_i,x,m_t)) &= p(z_i|x,s_t) \\ &= \mathcal{N}(d_{\text{plane}}(s_t,T(x)z_i;x);0,\Sigma_{sz}(x)) \\ \Sigma_{sz}(x) &= \Sigma_{s_t} + R(x)\Sigma_{z_i}R(x)^T, \end{aligned} \tag{3.3}$$

where Σ_{z_i} is the point measurement covariance in $\mathbb{R}^3$. If no association candidates exist, the likelihood is assigned a default no-association-likelihood corresponding to the sensor models false/random measurement probability. This likelihood is calculated for each particle and normalized by the number of valid points in the scan-line. Figure 3.2 shows the association method for several measurement endpoints.

3.3.3 Importance Weighting

As discussed in Section 2.4.4.2, importance weights $w^{(i)}$ compensate for the mismatch between the target and proposal distribution. As the motion model from Section 3.3.1 represents the proposal distribution, the importance weights for each particle is given by the observation likelihood,

$$w^{(i)} = p(Z|x^{(i)}, m_t). \tag{3.4}$$

Using importance weighting from observation likelihoods and following the standard particle filter algorithm, the particles are resampled according with probability proportional to the their importance weight.

3.3.4 Particle Filter Configuration

Several parameters can be used to tune the performance of the particle filter. This section enumerates each possible parameter including a description of potential side-effects.

1. Particle count

   The particle count has a large impact on localization performance. As the number of particles increases, the accuracy of the filter also increases given an adequate proposal distribution. However hardware performance concerns arise as each particle must have an observation likelihood calculation per laser scan. This value should be tuned to platform hardware and additional constraints on CPU usage. Within our implementation, heavy usage of parallelization allows for fast localization at the cost of high CPU usage.

2. Translation Noise

   Translation noise, denoted $\Sigma_{t(o_t)}$ in Equation 3.1, controls the additive translation noise stochastically added to particles during motion model prediction.

This noise value is calculated as

$$\Sigma_{t(0_t)} = \gamma * \Delta t(o_t) + \tau, \tag{3.5}$$

where γ is the multiplicative noise constants, $\Delta t(o_t)$ is the odometry delta between the current and previous time step, and τ is constant additive noise.

γ should be chosen to compensate systematic over- or undershoot of the odometry system. This parameter assumes that the faster the robot moves (resulting in larger deltas), the more error exists in the supplied odometry. τ is a constant noise added to ensure good sampling of state space.

To avoid confusion, γ and τ are referred to as the “translation sampling variance factor” and “translation sampling variance min” respectively.

3. Rotational Noise
   Rotational noise, denoted $\Sigma_{r(o_t)}$ in Equation 3.1, has the same parameters and effects at the translational noise constant.

   To avoid confusion, γ and τ in reference to rotational noise are referred to as the “rotational sampling variance factor” and “rotational sampling variance min” respectively.

4. Resampling Rate
   Within our system, filter resampling is performed after a configured timeout has been reached. Resampling should be performed to avoid particle depletion and refocus particles to allow for a reduced particle count necessary for satisfactory results. Due to 6D particle filtering with a limited number of particles due to performance reasons, by default the filtering algorithm resamples at each timestep.

3.4 Drivability and Cost Assessment

Using MRS maps for drivability and navigation planning poses several problems due to map discretization. Local MRS maps generated from individual 3D laser scans use the standard multi-resolution scheme based on the sensor error properties. Due to this discretization, downsampling and upsampling of the map surfels would be needed to model regular regions for drivability assessment discussed in Section 2.5. An example of this discretization issue is shown in Figure 3.3.

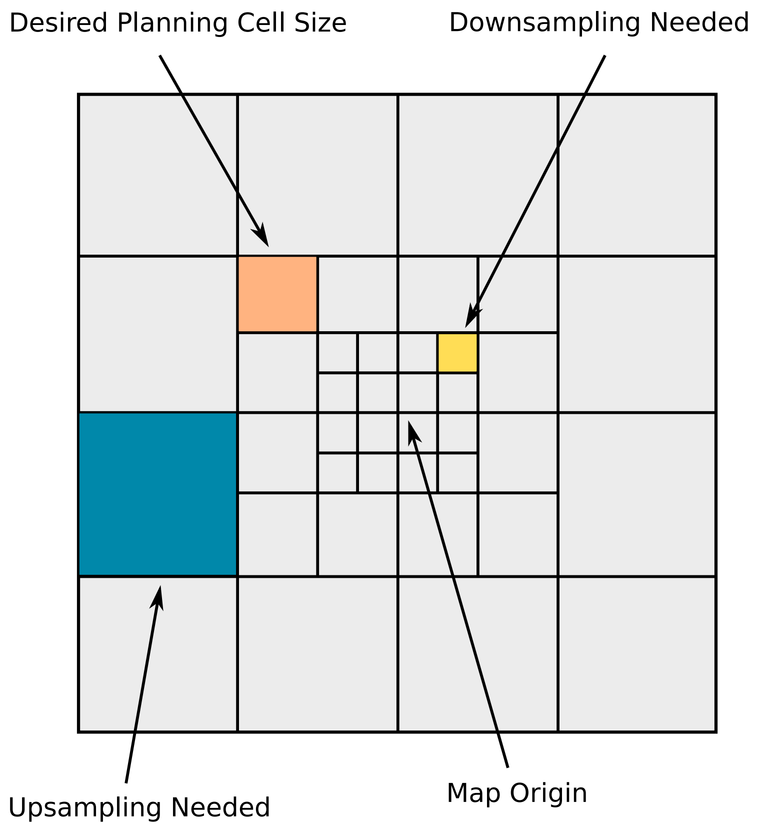


Figure 3.3: Figure showing necessary downsampling and upsampling when using multi-resolution maps for grid planning.

Additional complexity comes from general planning within three dimensions. Three dimensional environment have a large state space as well as complex environment structure such as stairs or tunnels. Due to the our navigation system being used within the Spacebot platform, we are able to assume the drivable space of the environment contains no structures where multi-level surfaces (positioned at the same location within the horizontal plane) exist. With this assumption, planning can be performed within a two dimensional grid structure where surface information is stored for cost and drivability assessment - otherwise known as planning in 2.5 dimensions.

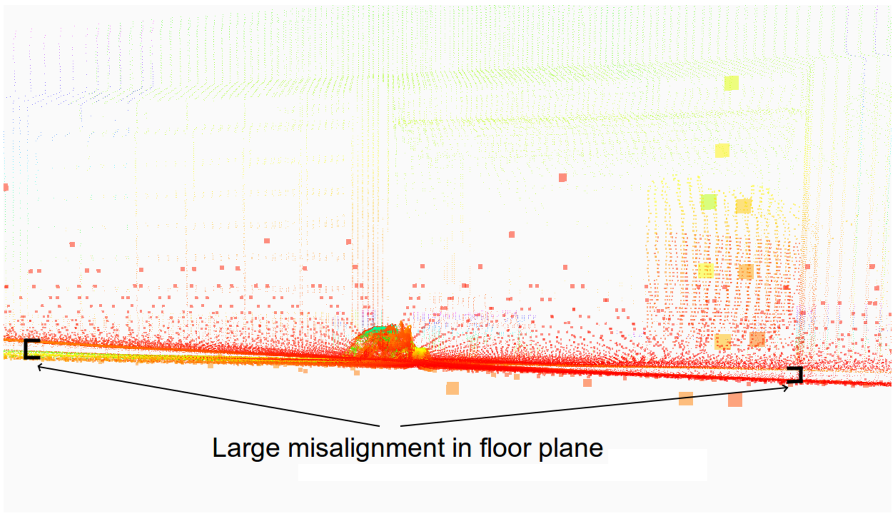


Figure 3.4: A slight misalignment results in large differences between measurement data at far distances. This point cloud has an error of only 3°, yet at a distance of 10 meters (pictured) the misalignment is clearly visible.

3.4.1 Drivability and Cost Map Representation

In 2.5 dimensions a grid is the natural datastructure for region growing and planning, using the implicit graph structure. Surface information must be stored within the grid to allow for terrain assessment. Surfels lend themselves well to this task as they provide all necessary terrain information and have well defined combination rules allowing data aggregation. Thus the data structure used for drivability and planning is a two dimensional grid of surfel structures denoted drive-surfels. Drive surfels contains all traditional surfel information with additional data for drivability flags and cost calculations.

Using the global allocentric map (from localization) for planning is a simple solution to overcome issues related to multi-resolution. This global map could simply be generated at the required resolution for planning resulting in a single data structure for region growing and planning. However to accurately assess drivability, near perfect alignment of all 3D scans would be required. Small inconsistencies in alignment can result in large surfel misalignments at far sensor distances. Figure 3.4 illustrates this issue using two laser scans of a flat surface with a misalignment of 3 degrees pitch. This side-view shows the large error at even relatively close distances (10 meters).

3.4.2 Drivability Map Creation

As previously discussed, the drivability map cannot simply be created by applying a 2D grid structure to the global allocentric surfel map. Thus a method must be created to combine individual 3D scans given their registration estimate resulting in a suitable representation of the environment.

This process starts by generating a surfel map at a predefined planning resolution for each 3D scan. Each 3D scan is transformed by the registration estimate into the map coordinate frame and inserted into a MRS map. By transforming all scans into the map frame, we guarantee an aligned grid structure between all data. This aligned structure allows for simplified aggregation of data.

By iterating over all surfel data in each map, we can generate a global 2D surfel grid by first calculating each surfel's grid index in the horizontal plane and updating the grid cell's surfel using merging criteria. Given n surfels from n 3D scans that correspond to the same cell within the 2D grid, the following criteria is used to determine the sensor data included in the cell's surfel:

1. Any surfel within a thresholded distance from the corresponding scan's sensor origin is used within the drivability map. When data within this threshold distance exists for multiple scans, the closer data is selected.

2. Given no surfels within a thresholded distance from the corresponding scan's sensor origin, the surfel having the highest mean height is selected. Additionally all surfels within a thresholded height difference from the selected surfel is merged.

Criterion 1 ensures that data having minimal sensor error and thus the highest accuracy is always included in the resulting global map. Criterion 2 is used to mitigate small misalignment issues shown in Figure 3.4. Merging data within a height tolerance is used to include data from multiple view perspectives and to maximize the amount of data used.

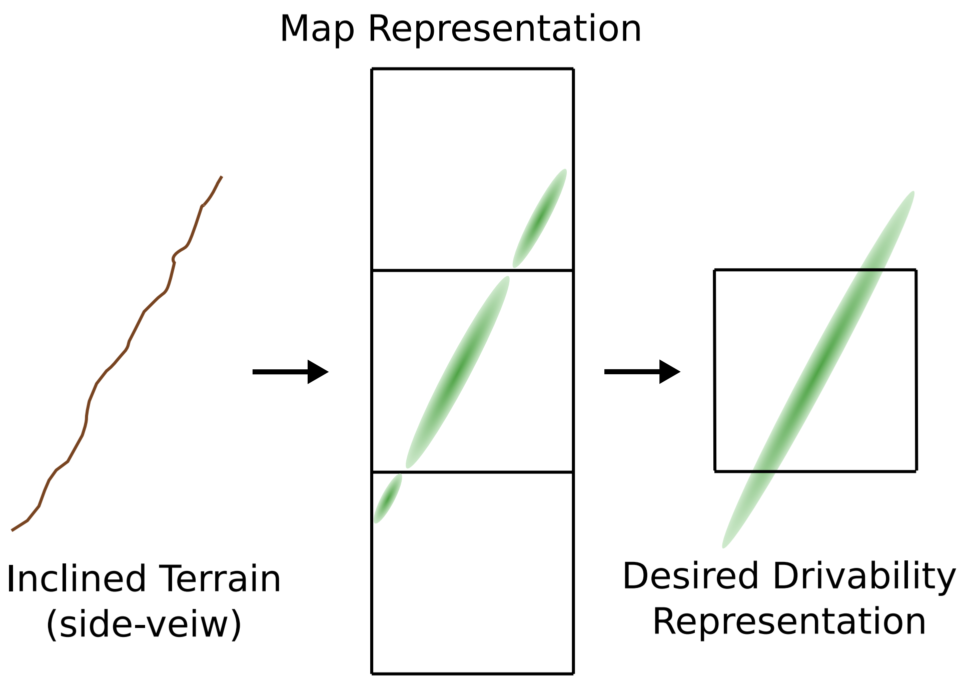

Map Representation
Inclined Terrain (side-veiw)
Desired Drivability Representation

Figure 3.5: Side-view of hillside (left) modelled using surfels. Due to map discretization, the hillside is separated into three neighboring voxels (center) For robust drivability assessment, this data must be combined and represented using an individual surfel (right).

As an additional pre-processing steps, all surfels within a scan are merged to neighboring surfels in the gravity axis. This surfel data would correspond to identical cells within the 2D grid, however this data should always be merged. Merging of data along the height axis is necessary due to potential environment structure conflicting with the map discretization. Figure 3.5 illustrates this issue by displaying three surfels generated from a steep hillside shown from a side-view perspective. Within the diagram, the surfels are in neighboring voxels due to map discretization, however for drivability processing the entire hillside should be represented.

3.4.3 Drivable Region Determination

The wavefront method described in section 2.5.1 is used to generate the drivable region using our surfel grid. To reduce memory consumption, the region growing

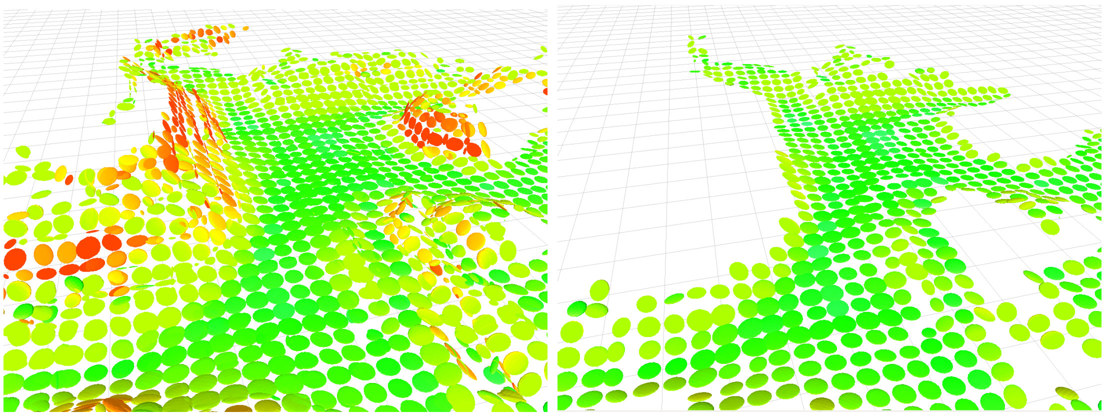


FIGURE 3.6: MRS map with corresponding drivability map indicating similar structure.

itself does not generate an additional datastructure but instead modifies drivability flags within the drive-surfel grid. Figure 3.6 shows a visualization (from above) of a global allocentric surfel map alongside the corresponding drivability grid where only drivable surfels are displayed.

3.4.4 Drivability Assessment

During region growing, we must evaluate the drivable characteristics of the current grid cell, including neighboring cells within the robot region. This allows for precomputation of cost parameters used for planning as well as the reduction of region growing effort as undrivable nodes are not further expanded.

The following characteristics are considered when determining the drivability of an individual surfel where the robot region is centered at the cell center:

1. Robot region data coverage percentage
2. Robot region bumpiness
3. Robot region global incline

3.4.4.1 Region Coverage

The robot coverage space is model by a bounding circle - an overestimate that allows for turning in place and thus negates the affects of orientation on cost. Thus to determine the region coverage centered at a cell, we simply perform a distance query within the pre-processed surfel grid where the distance between the query cell and originating cell centers is less than the robot radius. The coverage percentage is modelled by the number of cells containing valid surfel data divided by the total number of cells within the region.

If region coverage percentage is less than a threshold, the cell is marked "not drivable" and the region growing from the cell is stopped. The recommended region coverage depends on several factors including general environment, safety considerations, and mapping resolution. Generally it is not desirable to require a coverage region of 100% due to occlusions and similar data limitations causing missing surfel data.

3.4.4.2 Bumpiness

We define local bumpiness of a cell $b_l(c)$ as the maximum difference between the cell's surfel mean height and all eight direct neighboring surfel mean heights. This calculation is shown in Equation 3.6 where N is the set of immediate neighbor cells and $\mu(c)$ is the surfel mean within cell c.

$$b_l(c) = \max_{n \in N}(\mu(c).z - \mu(n).z) \tag{3.6}$$

Using local bumpiness, the region bumpiness of a grid cell $b_r(c)$ is defined as the maximum local bumpiness of cells contained within the robot region. Equation 3.7 shows this calculation where $R(c)$ is the set of cells contained within the robot region centered at cell c.

$$b_r(c) = \max\left(0, \max_{c_r \in R(c)} b_l(c_r)\right) \tag{3.7}$$

If the region bumpiness of a cell is greater than a threshold, the cell is marked "not drivable" and region growing from the cell is stopped. The bumpiness threshold should be set depending on the robot ground coverage, robot wheel size, and general planning safeness required.

3.4.4.3 Global Incline

Global incline refers to the incline angle (in radians) of the region with reference to the gravity vector. One must consider global incline due to the constant force of gravity acting upon the robot. To determine the global incline of a cell $i_r(c)$ we aggregate the statistical data of all surfels within the robot region to model a "region surfel" according to the update procedure described in Section 2.3.2. From this surfel, the normal vector is extracted from the covariance matrix and compared against the gravity vector for determining incline. Equation 3.8 shows how to calculate the angle (always positive in radians) between two vectors using the vector inner product and magnitude operator.

$$\theta(\vec{a}, \vec{b}) = \left| acos\left(\frac{\vec{a} \cdot \vec{b}}{||\vec{a}||||\vec{b}||}\right)\right| \tag{3.8}$$

If the global incline angle relative to horizontal is outside a thresholded range ($\pm 20^\circ$ for example), the cell is considered "not drivable" and region growing from the cell is stopped. The incline threshold should be related to the incline driving characteristics of the robot considering both motor capabilities and tipping resistance.

3.4.4.4 Drivability Cost

The cost of a cell $C(c)$ is used as the node code for planning and is dependent upon both region bumpiness and incline. The cost is calculated as

$$C(c) = \alpha b_r(c) + \beta i_r(c), \tag{3.9}$$

where α and β are the weight parameters for bumpiness and incline respectively. As with previous measures of drivability, the total cell cost is compared to a threshold. When the cost is greater than the given threshold, the cell is marked "not drivable" and region growing is halted.

Thresholding the cell costs offers extra safety limitations against a combination of bumpiness and incline resulting in an unsafe path from planning. This threshold should be related to the robot characteristics including resistance to tipping and wheel size.

3.5 Navigation Planning

Navigation planning is performed using the implicit graph structure from the drivability cost grid described in Section 3.4. For path planning, the well-known and efficient A* algorithm presented in Section 2.5.2 is implemented including handling of special cases (described in Section 3.5.5). The following sections describe the integration of an A* planner for producing robot trajectories.

3.5.1 Neighborhood Transition

When considering the grid structure used for drivability and cost assessment, two standard neighborhoods are used. The von Neumann [Weisstein, a] and Moore neighborhood [Weisstein, b] definitions are commonly used to determine legal transitions in a grid-like structure. The von Neumann neighborhood consists of the four direct neighbors in a 2D grid (excluding diagonals); predictably the Moore neighborhood is the eight-neighborhood including diagonals. Due to the kinematics of the Spacebot robot as well as the advantage of generating smoother trajectories, the Moore neighborhood was used as the transition rule. As an additional limitation to improve planning efficiency, a possible transition is only considered valid if the neighbor node is drivable.

3.5.2 Node Costs

The node cost used during planning is the drivability cost described in Section 3.4.4.4. In order to ensure optimality of the A* algorithm and to fulfill the heuristic admissibility requirement, the node cost C_{node} must always be ≥ 0. This constraint can be inspected on a term by term basis referencing the drive cost calculation from Equation 3.9. The first term, region bumpiness, is by definition ≥ 0 from Equation 3.7. The second term, global incline must also be positive due to the absolute value operation in Equation 3.8.

3.5.3 Edge Costs

One issue with using A* planning is the dependence upon a admissible heuristic for efficient path planning. Determining a heuristic function that is both accurate and admissible can be a challenging problem. To drastically reduce heuristic function complexity, the edge cost between two nodes can be modelled as the euclidean distance between the surfel means scaled by a weighting factor γ_d. Within the presented framework, an additional edge cost parameter determined by the orientation change $\Delta\theta$ between the surfels connected by the edge is used - also weighted by factor γ_o. The orientation between two surfels s and s' having region averaged normals n and n' respectively can be calculated as

$$\Delta\theta = \left|\cos^{-1}(n \cdot n')\right|, \tag{3.10}$$

using the vector inner product and the normal vectors n and n' which have unit length.

The weighting factor allows customization of the planning algorithm to generate paths focusing on minimizing different drivability aspects. To summarize, the total edge cost C_{edge} between surfel s and s' can be calculated as

$$C_{\text{edge}} = \gamma_d \, distance\,(\mu(s), \mu(s')) + \gamma_o \Delta\theta \tag{3.11}$$

3.5.4 Heuristic

Without knowing the path to the goal from the current cell, it is not possible to accurately assess the expected sum of node costs to the goal. However due to edge costs using a scaled euclidean distance, the heuristic can also used this scaled euclidean distance to the goal cell. This heuristic is guaranteed to be admissible because node costs are never negative and the sum of edge costs from a cell to the goal must be at least the euclidean distance due to the triangle inequality. As an important note, admissibility can only be guaranteed if the same scaling factor γ_{edge} is used for both edge cost calculation and the heuristic function.

Figure 3.7 shows the use of the euclidean distance heuristic in a two dimensional grid. Due to environment structure, the heuristic does not accurately predict the true cost - however the heuristic is admissible and without a more complex method of environment analysis cannot be easily improved.

3.5.5 Special Navigation Cases

Special care must be taken for planning to target locations of any type listed below.

1. The target location is inside an obstacle
2. The target location is outside the drivable area

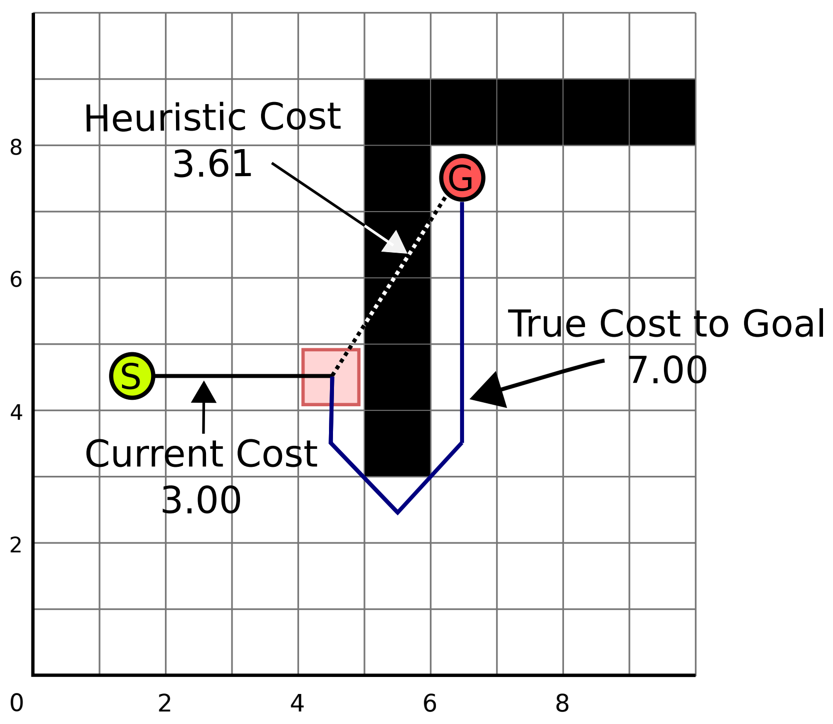


FIGURE 3.7: Example of euclidean distance heuristic function in two dimensional environment.

3. The target location is outside the perceived environment

The results of the general A* path planning algorithm will fail given any of these conditions. In the failure case, the A* algorithm has essentially performed Dijkstra's algorithm for the entire drivability graph. For a robust planning system, we wish to approximate or determine a "close" solution suitable for continuing navigation.

Cases 1 and 2 describe nearly identical situations as the target location has mapping data but is not drivable. In this case, the planning system considers the 10 closest surfels to the target position.[1] The candidate having lowest total path cost and a distance to the target position within "target reached" threshold is selected as the new target surfel. As the system has already computed all minimum cost paths, the path can be easily computed to the new target and returned. If no surfels are within the "target reached" threshold the surfel having lowest total

[1] Using the euclidean distance as a metric.

cost including heuristic is selected as the target surfel and the path is computed as before.

In case 3, there is no guarantee that close surfels allow for a future path to the target location without additional mapping data. In this case, the surfel having the lowest total cost (including heuristic) is selected as the target node and the minimum cost path to this node is computed.

Handling these special cases allows for the navigation system to continue in a "likely" correct direction and allows for additional data collection.

Chapter 4

Evaluation

This chapter details the evaluation results of the presented system. Within simulation, systematic experiments have been performed to demonstrate the accuracy and robustness of the mapping and localization systems, along with additional tests of the drivability assessment and path planning components. Real-world experiments in three distinct environments are presented, including results from the DLR Spacebot Cup. Additionally a public dataset from the University of Toronto, Institute for Aerospace Studies has been used for mapping evaluation.

4.1 Test Environment

All real-world experiments and the University of Toronto dataset has been processed using an Intel Core i7-4770K CPU @ 3.50 GHz CPU with 32GB RAM. [1] This computer is available both onboard and offboard the Spacebot Explorer, however results presented here have been processed offboard from sensor data collected during robot operation.

Simulation experiments have been performed using an Intel Core i7-2670QM CPU with 8GB RAM. [2]

[1] Four main cores with four virtualized cores.
[2] See footnote 1

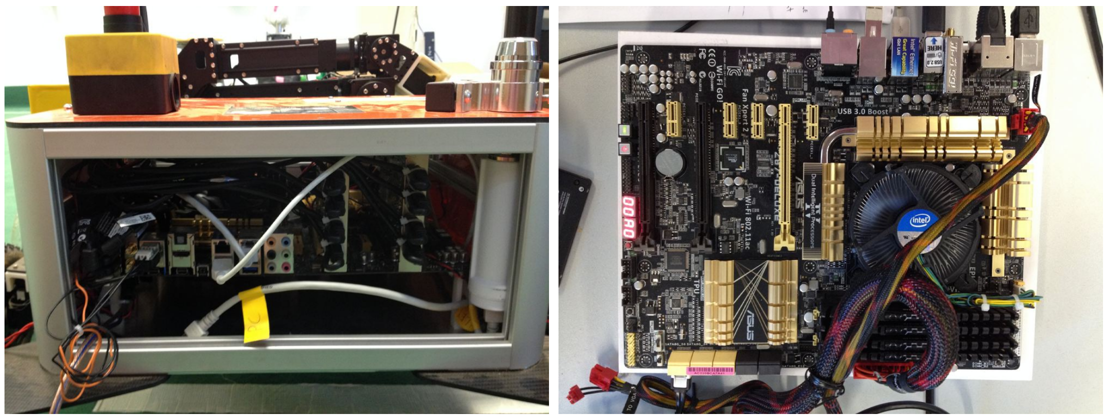


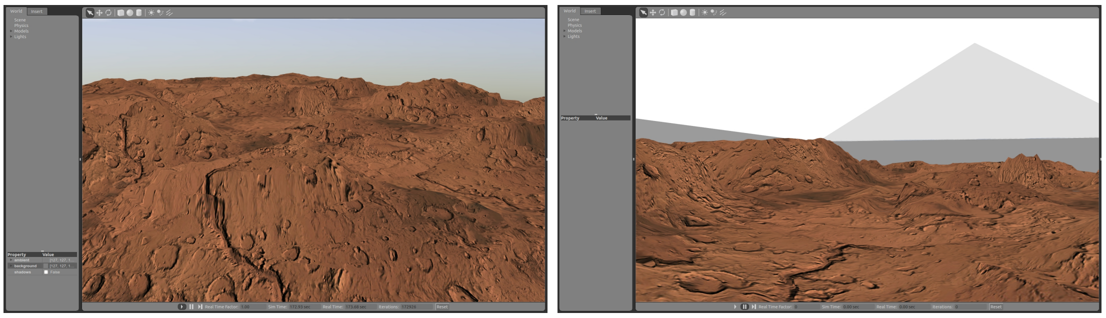
Figure 4.1: The Intel Core i7-4470K CPU @ 3.50 GHz with 32GB RAM used for the Explorer platform.

Figure 4.1 shows the computation unit used for real-world experiments and Explorer operation.

4.2 Simulation Experiments

This section presents the systematic experiments to the mapping and localization systems as well as tests of the drivability assessment and navigation systems. Each component has a section detailing experimental setup and results.

Simulation data was generated using the Gazebo simulation suite using hand modelled terrain. Odometry was also simulated having systematic error in all degrees of freedom. Figure 4.2 shows the simulation environment using for experimentation in the following sections.



(A) Open-air environment. (B) Enclosed environment.

Figure 4.2: The rough-terrain simulation environment shown in the Gazebo simulator.

4.2.1 Mapping Snap-In With Surface Features

Snap-in distance is measured as the maximum distance from the origin that a 3D scan can be transformed and still be successfully registered to itself. A scan is considered successfully aligned if the registration system estimate is equal to the applied transformation. This metric gives information concerning the robustness to the registration guess and thus gives an idea of the required accuracy of the localization system.

As an additional test, experiment data is processed with and without reflectance measurements showing the impact of surface features during registration.

As presented in Section 2.3, gradient descent methods are used to align 3D scans. As with all parameter optimization using gradient descent, it is possible for the registration system to find only a local minimum and thus an incorrect registration estimate. This depends greatly on the initial registration guess. Using surface features should improve registration by limiting the surfel associations and thus modifying the error gradient.

For this set of experiments a 3D scan was used to initialize the mapping system. Next the scan was copied and transformed with an offset of $x \in [-4, 4], y \in [-4, 4]$ at a 0.4 meter interval and rotated with an angle $\theta_{yaw} \in [-\pi, \pi]$ with a 22.5 deg interval and processed using the registration system. If the resulting pose estimate was equal to the initial transform with a threshold parameter $\tau = 0.1$, the registration is considered successful.

Figure 4.3 shows the 3D scan used for testing snap-in distance.[3] Figure 4.4 shows successful registrations where a vector represents both the offset in meters and the rotation from ground truth. Additionally vectors are colored coded to show alignment quality. A registration is considered successful if the sum of the squared distances between transformation matrices is less than 0.1. Red vectors correspond

[3] The scan was taken including a roof and wall similar to those found at the Spacebot Cup. Additionally the data was cut to include just the terrain.

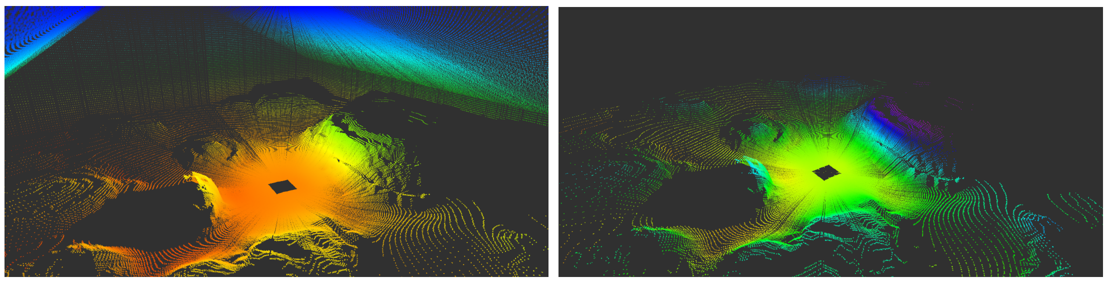
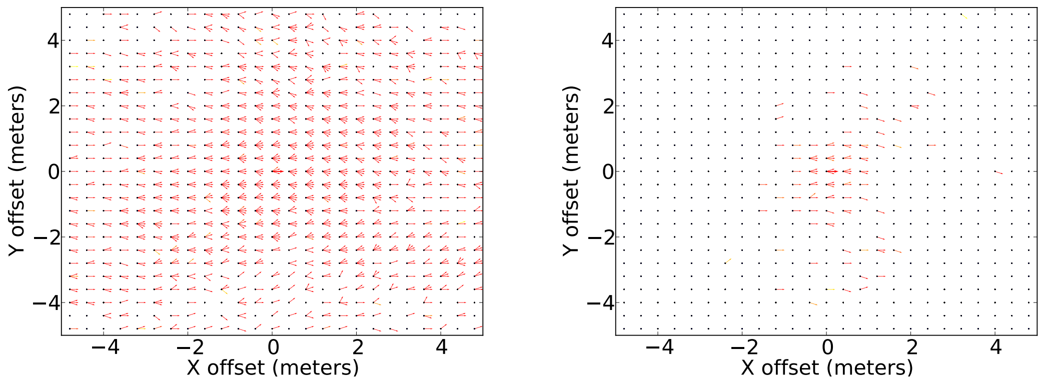

(A) Full scan. (B) Cut scan.

Figure 4.3: Snap-in distance experiment where points are colored by height.



(A) Results using full scan. (B) Results using cut scan.

Figure 4.4: Successful registrations given transformed offset and rotation for the snap-in distance experiment.

to high quality alignment while yellow indicates less accurate. The vector shown in the center in red is the ground truth vector.

As seen in the figures, the presence of large well-defined features such as walls and a ceiling increases the ability to register scans given larger initialization error. Alignment using walls resulted in 1559 successful registrations; within the cut scan only 98 successful registrations were recorded.

To test the effects of surface features on registration, the experiment was also performed with a laser scan of a ceiling. Due to lighting fixtures, the ceiling has a unique reflectance profile and consists of a relatively flat surface where shape does not necessarily provide adequate information for 3D scan alignment. Figure 4.5 shows the ceiling 3D scan with reflectance values shown by gray scale color.

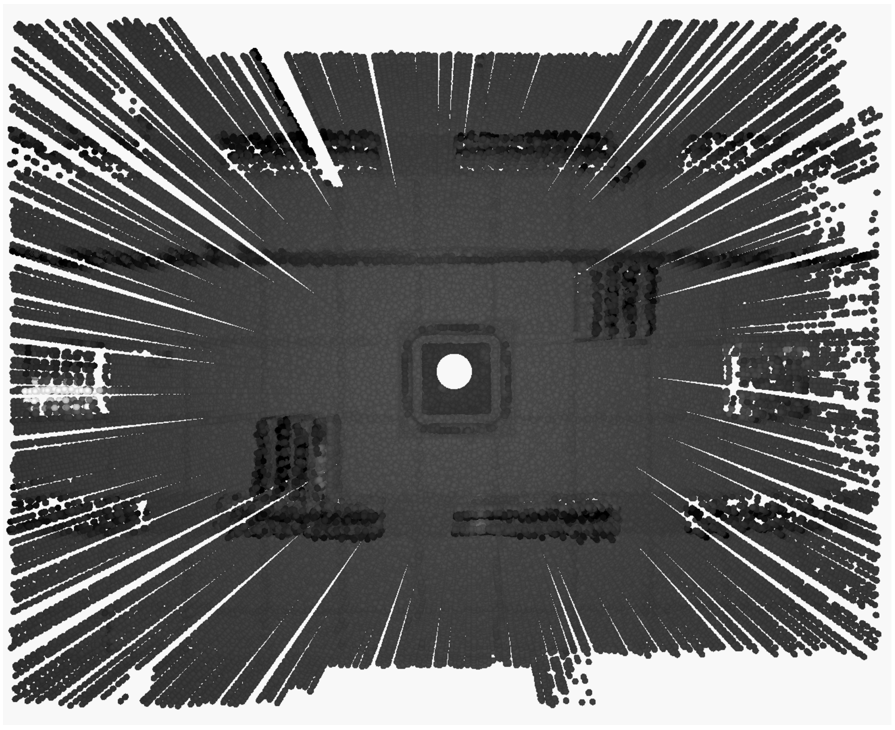
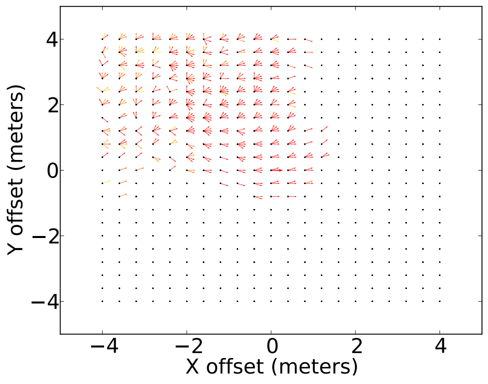
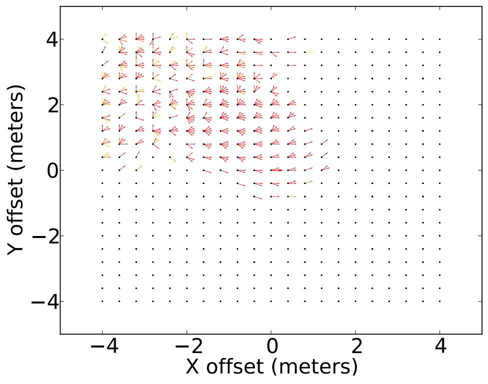


FIGURE 4.5: 3D laser scan of ceiling colored by reflectance intensity.



(A) With reflectance information.



(B) Without reflectance information.

FIGURE 4.6: Successful registrations given transformed offset and rotation for the snap-in experiment testing surface reflectance features.

Using the ceiling laser scan, "snap-in" tests were executed both with and without reflectance information. Figure 4.6 shows successful registrations for both instances in the same format as above. From careful inspection of this figure, one can see using reflectance features improves the robustness of the registration system. Without using the reflectance feature, 415 successful registrations were recorded; including reflectance information resulted in 452 successful registrations.

4.2.2 SLAM

To evaluate the SLAM system within simulation, five mapping and localization runs were performed in enclosed and open-air environments. As localization is a stochastic process, each run must be processed several times with the results averaged. Each experiment was performed using various initial poses and trajectories within the simulation environment shown in Figure 4.2.[4]

In this section I first present the error metric and the SLAM experiment configuration. Results are presented for localization and mapping in the form of ground truth comparisons. Additional qualitative results are presented in the form of aligned point clouds.

4.2.2.1 Absolute Trajectory Error

Sturm et al. [2012] propose benchmarking RGB-D SLAM systems using Absolute Trajectory Error (ATE). ATE is a global error measure that indicates the consistency of a tracked path w.r.t. ground truth information. Our simulation evaluation presents the following ATE error measures: mean, standard deviation, minimum error, maximum error.

To calculate ATE, both the tracked and ground truth poses are sampled including time stamp information. After collecting samples for the entire trajectory, the tracked path is aligned to the ground truth path using a closed form least squares method [Horn, 1987]. Using time stamp information for data association, global error measures can be calculated.

ATE shows localization performance by comparing tracked and ground truth trajectories. Mapping performance is also evaluated by this metric and is presented by comparing the robot poses at 3D scan locations. Mapping results compare the initialization estimate from localization, the graph optimized registration estimate, and ground truth pose information.

[4] Roof and side walls are not shown.

Parameter	Value
Particles	250
Translation Sampling Variance Min	0.00625
Translation Sampling Variance Factor	0.08
Rotational Sampling Variance Min	0.0025
Rotational Sampling Variance Factor	0.15

Table 4.1: Particle filter configuration for SLAM simulation experiments.

4.2.2.2 Particle Filter Configuration

The localization parameters were originally tuned for usage at the Spacebot Cup using the Explorer platform. Due to movement restrictions of the simulated robot (turning in place), the orientation parameters were refined using an initialization experiment. This initialization allowed for reduced noise in orientation and demonstrates the need to tune localization parameters for specific applications.

The tuned filter configuration is shown in Table 4.1. See Section 3.3.4 for a discussion on particle filter parameters and effects.

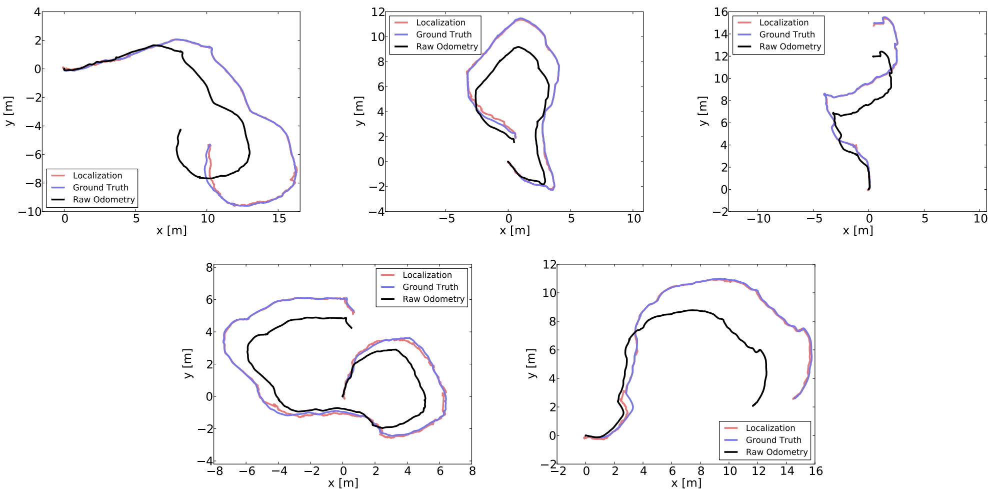
4.2.2.3 Localization Results

The localization estimate was recorded during 10 runs of each SLAM experiment and compared to ground truth using the ATE metric. Figure 4.7 shows an example tracked trajectory, including ground truth and odometry input, for each of the ten total SLAM experiments.

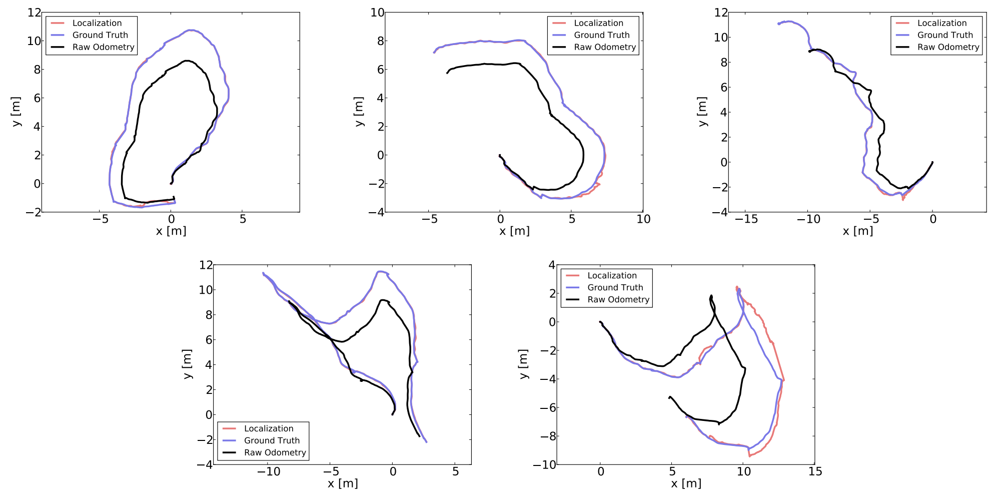
The SLAM experiments had an average duration of 288 seconds with an average path length of 33.8 meters. Table 4.2 presents the average ATE for each of the experiments. Averaging all simulation experiments, a mean error of 0.117 m was recorded - including a minimum and maximum error of 0.005 m and 0.546 m respectively. Analysis shows localization ran with an average update frequency of 32 Hz.

Experiment	Environment	Error metric in meters			
		Mean	Std	Minimum	Maximum
1	Enclosed	0.103	0.091	0.003	0.459
2	Enclosed	0.095	0.084	0.002	0.507
3	Enclosed	0.049	0.038	0.002	0.255
4	Enclosed	0.311	0.240	0.012	1.140
5	Enclosed	0.130	0.127	0.004	0.615
6	Open-air	0.035	0.031	0.002	0.276
7	Open-air	0.054	0.057	0.002	0.334
8	Open-air	0.039	0.046	0.001	0.346
9	Open-air	0.037	0.028	0.001	0.288
10	Open-air	0.412	0.379	0.025	1.528
Average		**0.117**	**0.105**	**0.005**	**0.546**

TABLE 4.2: Tracking ATE results for each simulation experiment in meters.



(A) Open-air environment.



(B) Enclosed environment.

FIGURE 4.7: Selected trajectories for the 5 SLAM experiments comparing odometry, localization, and ground truth trajectories.

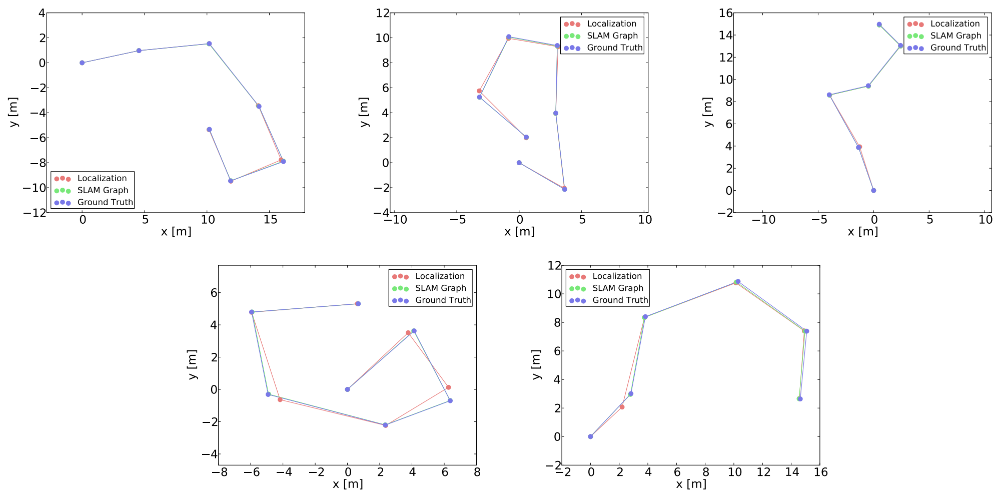
4.2.2.4 Mapping Results

Mapping results including input estimate, registration results, and ground truth pose information were recorded from each simulation experiment. Figure 4.8 shows a selected result from each of the ten experiments.

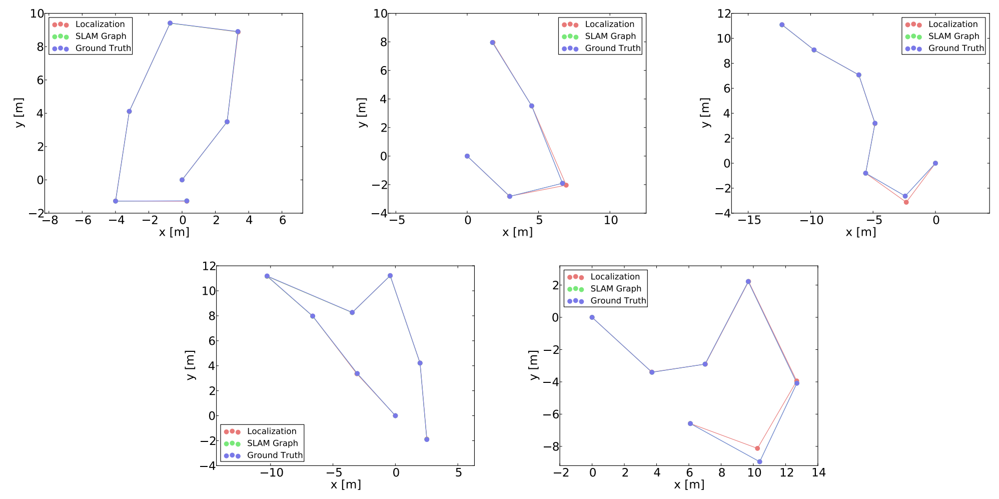
The SLAM experiments had an average duration of 288 seconds with 7 3D scans. Table 4.3 presents the average 3D scan registration error for each experiment. The localization component had an overall average registration guess error of 0.159 m. Through global optimization and map-to-map alignment, the mapping component produced more accurate results with an average error of .029 m.

Experiment	Environment	Scan Registration (m)		Localization (m)	
		Mean	Std	Mean	Std
1	Enclosed	0.014	0.002	0.019	0.011
2	Enclosed	0.012	0.002	0.017	0.010
3	Enclosed	0.013	0.003	0.072	0.149
4	Enclosed	0.012	0.001	0.380	0.763
5	Enclosed	0.011	0.001	0.113	0.210
6	Open-air	0.027	0.018	0.160	0.163
7	Open-air	0.049	0.021	0.071	0.030
8	Open-air	0.028	0.010	0.131	0.173
9	Open-air	0.040	0.040	0.406	0.407
10	Open-air	0.080	0.028	0.222	0.249
Average		**0.029**	**0.013**	**0.159**	**0.216**

Table 4.3: Mapping ATE results for each simulation experiment in meters.



(A) Open-air environment.



(B) Enclosed environment.

FIGURE 4.8: Selected localization guess, mapping estimate, and ground truth 3D scan poses for the 5 SLAM experiments in both open and enclosed environments.

Parameter	Value
Robot size	1.0m
Coverage threshold	50%
Bumpiness threshold	0.20m
Incline threshold	20°
Maximum Cost Threshold	0.48
Bumpiness cost weight	0.5
Incline cost weight	0.5

FIGURE 4.9: Parameters used for the drivability map shown in Figure 4.10.

4.2.3 Drivability Assessment

Drivability assessment was tested under several simulation environments to determine the correctness of terrain analysis. First experiments performed simply show qualitative results of generating drivability maps in simulation. Additional experiments to analyze the incline assessment and bumpiness calculations are presented.

4.2.3.1 Drivable Region Determination

Using the method presented in Section 3.4, a surfel grid including drivability costs was computed for the simulation environment shown in Figure 4.2. Figure 4.10 visualizes the surfel grid where drivable surfels are in gray scale with intensity corresponding to normalized surfel cost. All other visible surfels are undrivable and indicate the reason by color. Table 4.9 lists the drivability parameters and values used for generating the map.

In the Section 4.2.4, path planning is shown for various drivability parameters. These results include drivability maps similar to that shown in Figure 4.10. These results are additional examples of the effects of drivability parameters on the traversability map and surfel costs.

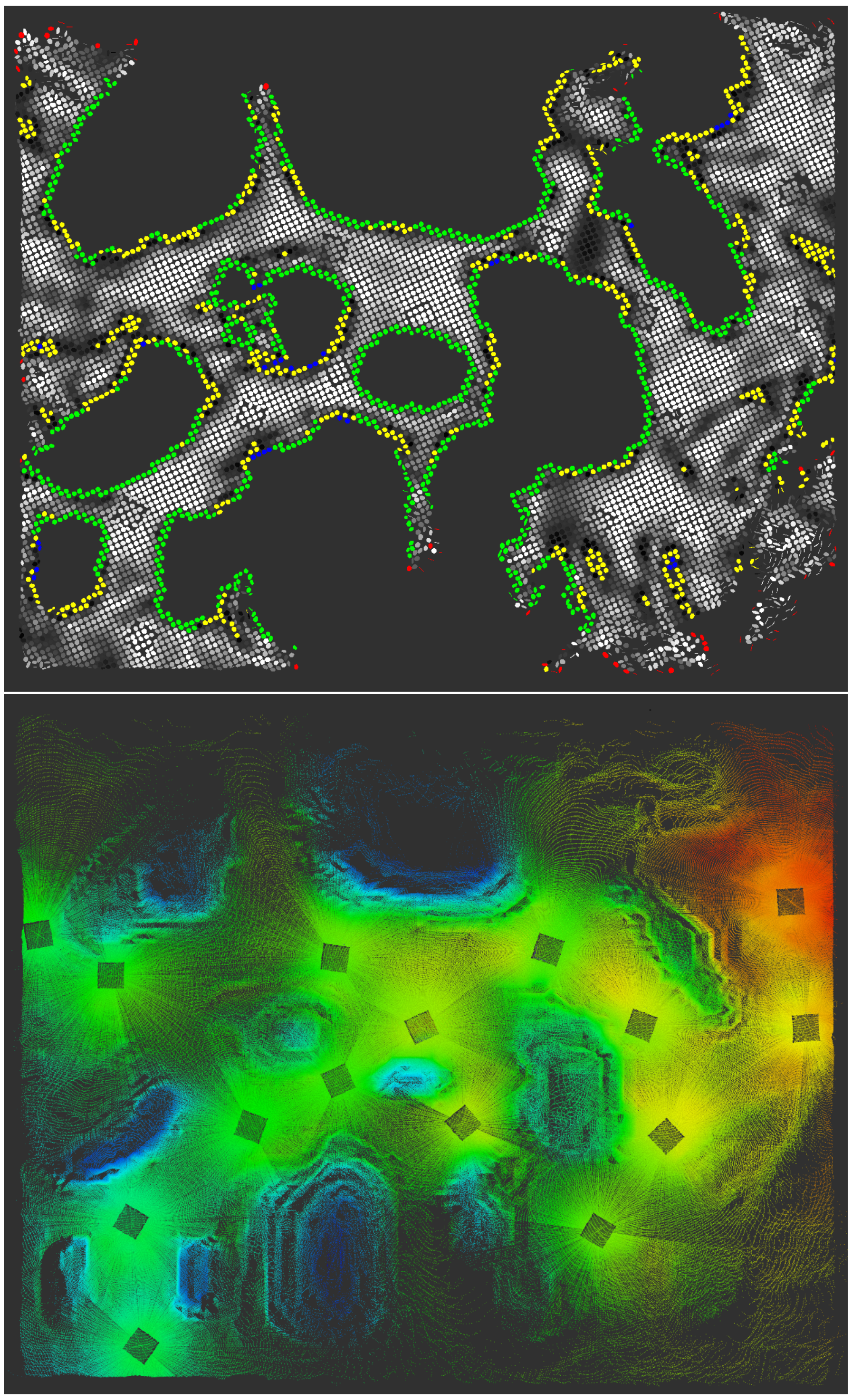


FIGURE 4.10: Aligned point cloud and corresponding drivability assessment results from the simulated environment. Green: maximum bumpiness threshold reached. Blue: maximum incline threshold reached. Red: minimum region coverage threshold reached. Yellow: total cost threshold reached. Gray scale: Drivable surfels where white indicates lower cost.

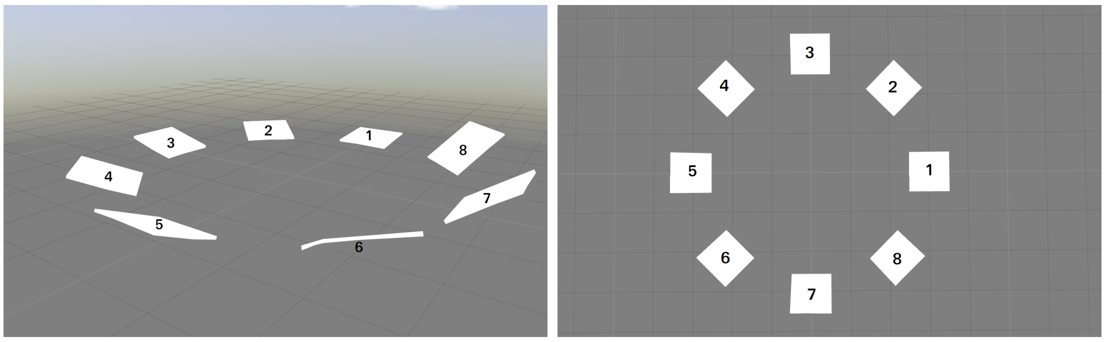


FIGURE 4.11: The custom simulation environment used for testing incline assessment including label for each ramp.

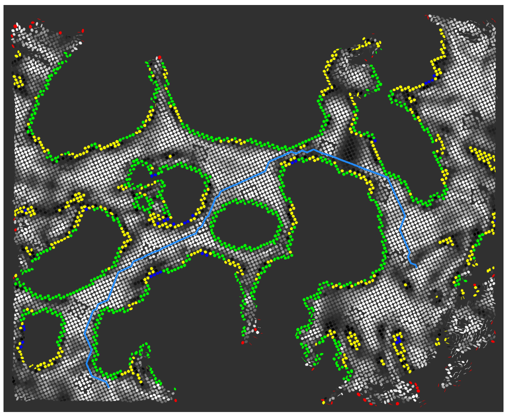
Ramp	True Incline	Measured Incline
1	5°	5.025°
2	10°	10.340°
3	15°	15.440°
4	20°	20.199°
5	25°	25.703°
6	30°	29.990°
7	35°	35.439°
8	40°	40.365°

FIGURE 4.12: Incline assessment results.

4.2.3.2 Incline Assessment

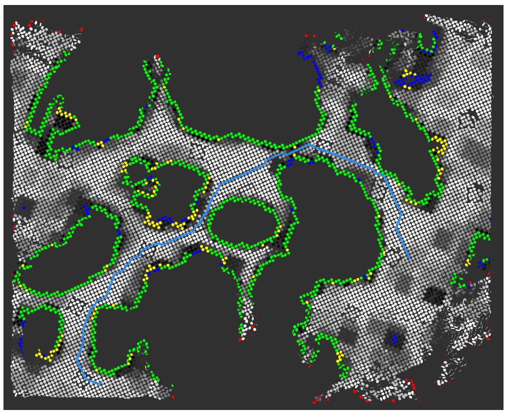
Using a custom simulation environment incline assessment accuracy was tested. Figure 4.11 shows an environment consisting of several ramps with a known incline. Using a complete 3D laser scan of the environment, the inclines of each ramp were queried and compared to the ground truth.

Table 4.12 shows the measured and actual inclines of each ramp in the experiment. The results have been averaged over 10 evaluations. To model the true sensor properties as accurately as possible, simulated laser ranges include Gaussian additive noise. For this experiment, the additive noise had a standard deviation of 0.01 m.



Parameter	Value
Robot size	1.0m
Coverage threshold	50%
Bumpiness threshold	0.20m
Incline threshold	20°
Maximum Cost Threshold	0.48
Bumpiness cost weight	0.5
Incline cost weight	0.5

FIGURE 4.13: Example planned trajectory and drivability map with corresponding drivability cost parameters. This trajectory uses the settings tuned for the Spacebot Cup competition.

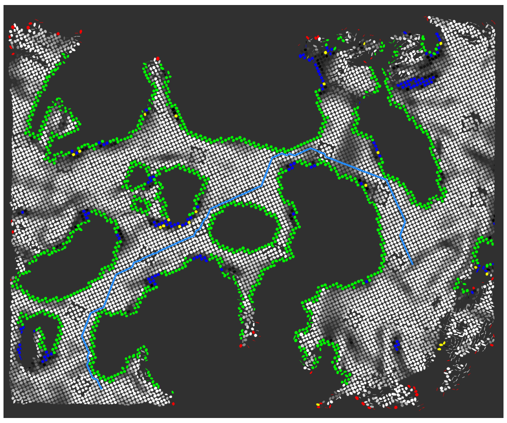


Parameter	Value
Robot size	1.0m
Coverage threshold	50%
Bumpiness threshold	0.20m
Incline threshold	20°
Maximum Cost Threshold	0.32
Bumpiness cost weight	1.0
Incline cost weight	0.0

FIGURE 4.14: Example planned trajectory and drivability map with corresponding drivability cost parameters. This trajectory uses a cost calculation ignoring incline costs.

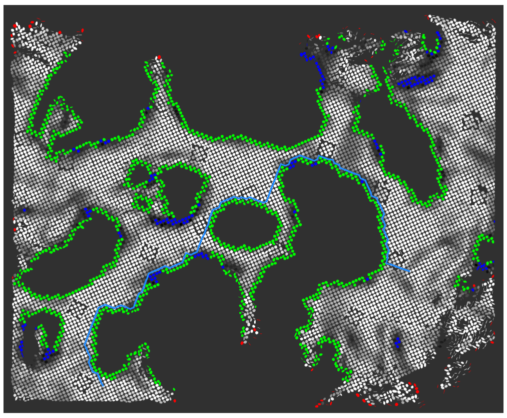
4.2.4 Path Planning

Path planning has been tested within simulation to qualitatively show the A* algorithm as well as the impact of cost weighting factors from Section 3.4.4.4 on the generated trajectory. Figure 4.13, Figure 4.14, and Figure 4.15 show a planned trajectory using various drivability parameter settings using the same start and target locations. Notice the drivability map surfels change between figures due to the different cost parameters. Figure 4.16 shows path planning using only the Euclidean distance heuristic. Here one should notice the shortest available path is generated - often using diagonals to minimize distances.



Parameter	Value
Robot size	1.0m
Coverage threshold	50%
Bumpiness threshold	0.20m
Incline threshold	20°
Maximum Cost Threshold	0.32
Bumpiness cost weight	0.0
Incline cost weight	1.0

FIGURE 4.15: Example planned trajectory and drivability map with corresponding drivability cost parameters. This trajectory uses a cost calculation ignoring bumpiness costs.



Parameter	Value
Robot size	1.0m
Coverage threshold	50%
Bumpiness threshold	0.20m
Incline threshold	20°
Maximum Cost Threshold	0.48
Bumpiness cost weight	1.0
Incline cost weight	1.0

FIGURE 4.16: Example planned trajectory and drivability map with corresponding drivability cost parameters. This trajectory uses no costs for path planning, instead relying only on the Euclidean distance heuristic.

4.3 University of Toronto Outdoor Dataset

We evaluate the mapping component against a publicly available dataset from the University of Toronto, Institute for Aerospace Studies [Tong et al., 2013]. The "a200_met" dataset consists of 25 laser scans recorded with a SICK LMS291-S05 sensor. The open-air environment measures 120 m by 60 m and consists of several terrain features including ridges, craters, and rocks.

For each 3D scan the following measurements are available: odometry estimate, pose estimate, and ground truth pose. The odometry estimated was produced using a Point Grey Bumblebee XB3 stereo camera. The pose estimate was calculated using spar-feature-based-batch alignment presented by Tong et al. [2011].

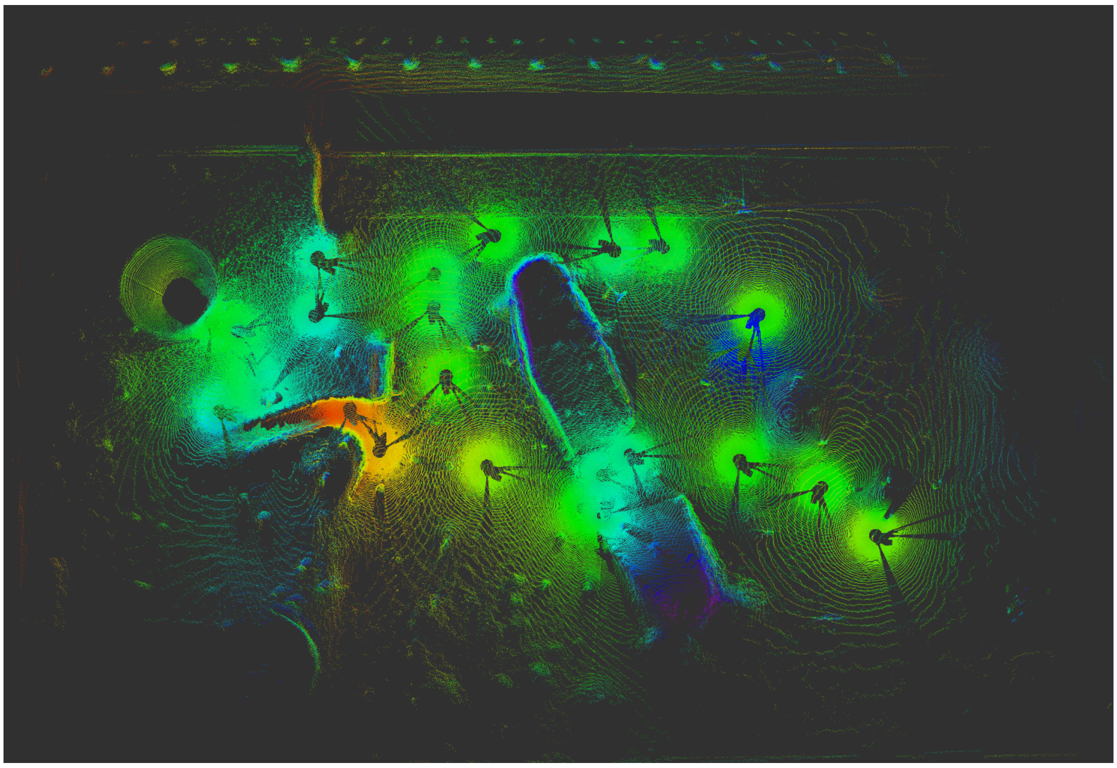


FIGURE 4.17: Aligned point cloud from the University of Toronto dataset using different registration estimates.

Ground truth data was measured using an on-board differential global position system (DGPS). As laser data between 3D scan positions is not provided, this dataset is only used to test the mapping system.

Figure 4.17 shows the aligned point cloud from all scans using the odometry estimate as an input guess. Within this point cloud, environmental features such as trees, ridge faces, and several others are clearly visible. One may also notice small misalignments, specifically when looking for straight lines below the row of trees. However when compared to the odometry guess and ground truth, the mapping results show a clear improvement.

Comparing against the provided ground truth data is however problematic. Figure 4.18 shows the combined point cloud when setting pose estimates to ground truth. From visual inspection it appears the ground truth data has been somehow corrupted or was incorrectly measured. Thus only qualitative visual inspection can be used for a comparison.

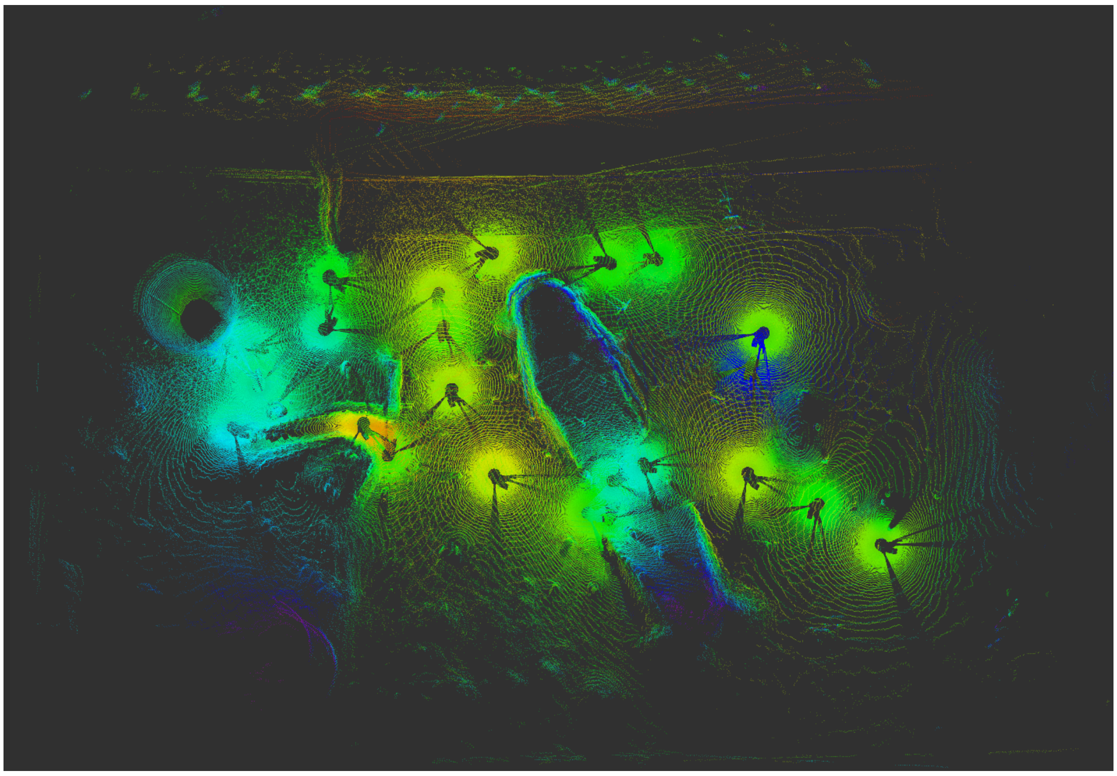


FIGURE 4.18: 3D scans aligned by the supplied ground truth data without registration.

3D Scans	0-1	1-2	2-3	3-4	4-5	5-6	Average Error
Ground Truth	4.20	4.94	6.53	4.45	4.54	8.54	
Localization	4.30	5.26	6.39	4.44	4.85	9.40	0.29
3D Scan Registration	4.18	4.90	6.37	4.40	4.49	8.43	0.07

TABLE 4.4: Parking garage distances between 3D scans in meters.

4.4 Parking Garage

The parking garage experiment was conducted in an enclosed parking space approximately 25×60 meters in size. Figure 4.19 shows the environment highlighting structures including vehicles, girders, support beams, and windows.

Laser scanner data was recorded using the Explorer platform which includes visual odometry. Seven full 3D scans were taken at an average distance of 5.53 m along an approximately 35 meter path. Ground-truth was hand-measured and consists of the relative distances between keyframe poses.

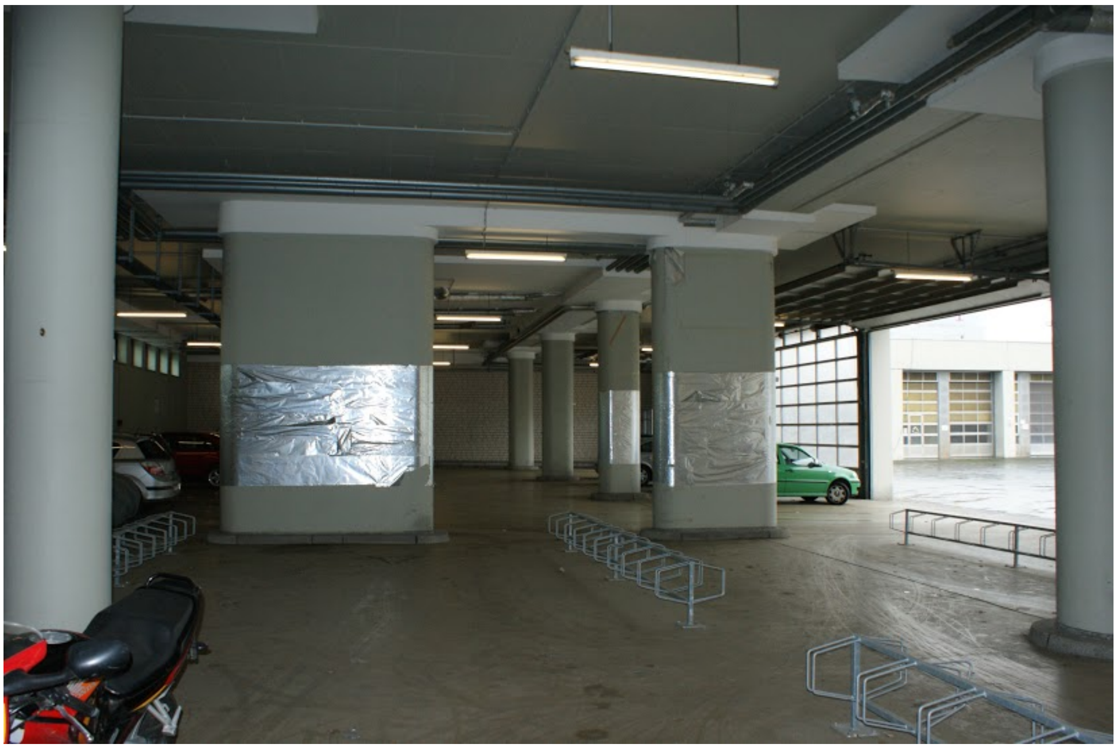


FIGURE 4.19: The parking garage experiment environment.

Using 250 particles for localization with reduced motion model noise in the z-axis (parallel to gravity), 3D scan registration with graph optimization resulted in a minimum and maximum error of 2 cm and 13 cm respectively. Table 4.4 shows a comparison between measured ground truth, localization estimate, and 3D scan registration estimate over the course of the experiment. This comparison consists of the estimated relative distance from the previous key-view in meters.

Localization had a measured update rate between 30 Hz and 35 Hz. Figure 4.20 shows the tracked trajectory of the robot during the experiment compared to pure visual odometry along with 3D scan registration poses. The corresponding point cloud generated using the final pose estimates is shown in Figure 4.21.

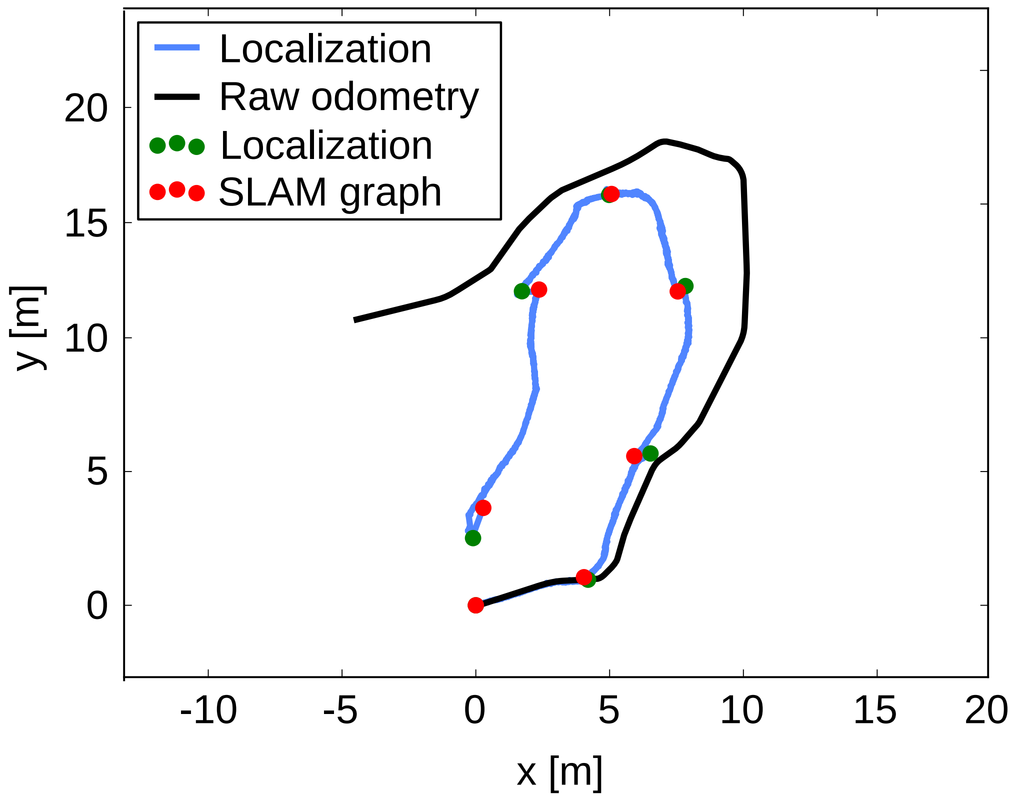


FIGURE 4.20: Parking garage experiment's tracked vehicle movement viewed from above - blue path. Black path - raw odometry input. Green circles - the tracking estimate when starting a 3D registration. Red circles - the registration estimate.

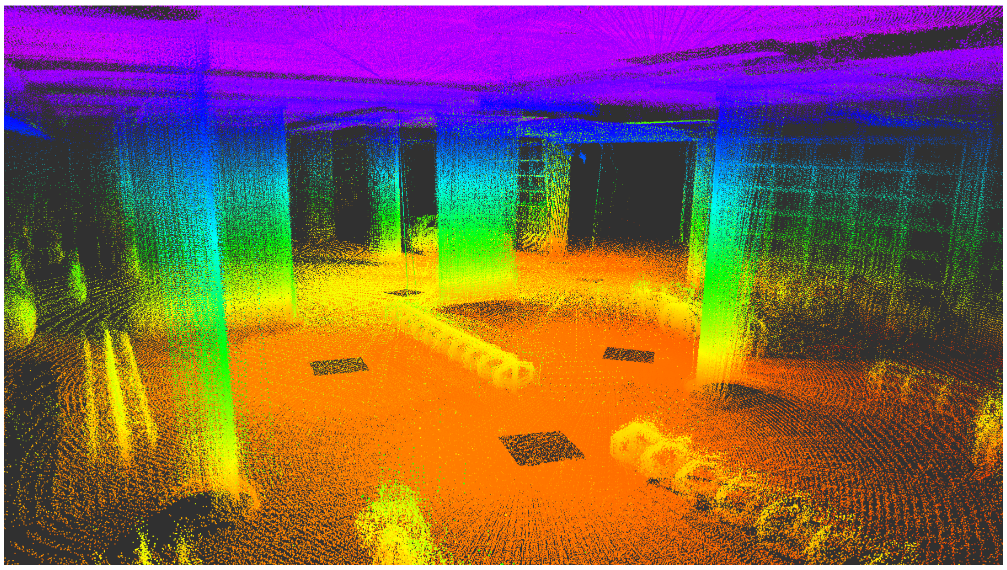


FIGURE 4.21: Parking garage experiment registered point clouds colored by height for visualization.

3D Scans	0-1	1-2	2-3	3-4	4-5	5-6	Average Error
Ground Truth	6.60	8.25	6.59	6.69	5.69	4.37	
Localization	6.59	8.26	6.47	7.27	5.97	4.57	0.20
3D Scan Registration	6.58	8.18	6.55	6.66	5.67	4.35	0.03

Table 4.5: Courtyard distances between 3D scans in meters.

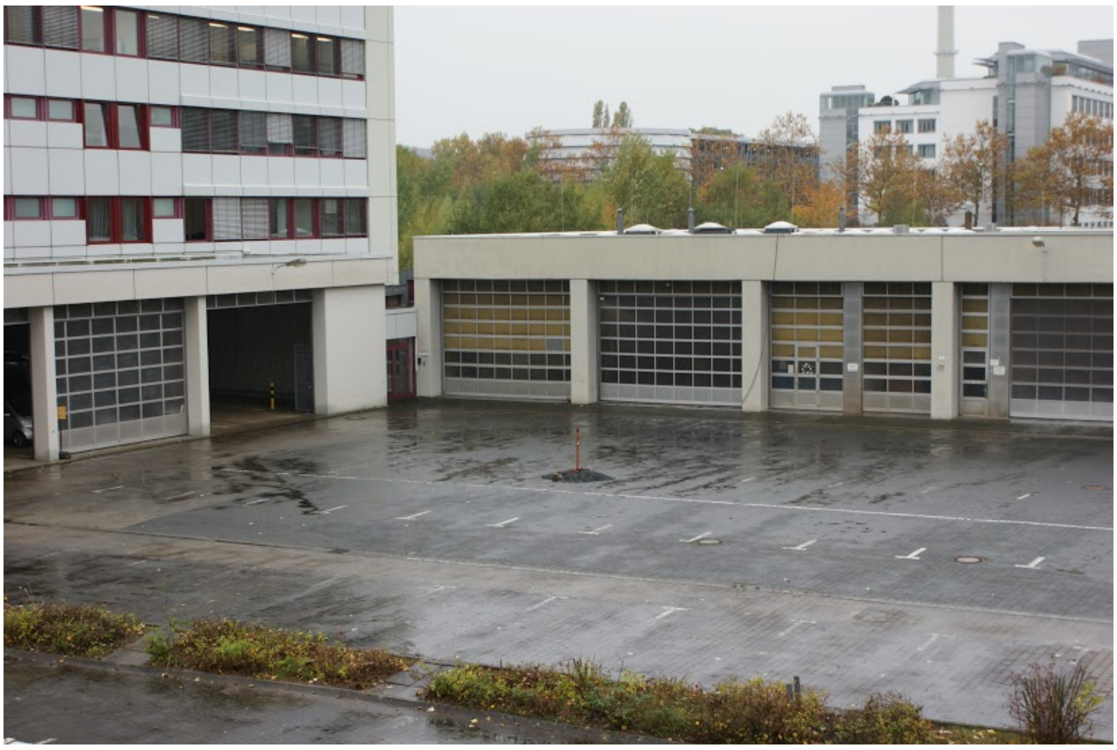


Figure 4.22: The LBH courtyard experiment environment.

4.5 Courtyard

An experiment testing the tracking and mapping abilities in an open-air environment was conducted at the University of Bonn Landesbehördenhaus (LBH) courtyard. Due to the lack of a roof and relatively feature-bare environment, a few boxes and garbage contains were arranged to create a denser test environment. Figure 4.22 shows an aerial view of the experiment environment.

Using the Explorer platform, seven 3D scans where taken at an average distance of 6.37 meters. Ground truth is again presented in the form of relative distance between 3D scan key-views. Using this metric, scan registration received minimum and maximum relative distance errors of 0.02 and 0.05 meters respectively. From the tracking system minimum and maximum errors of 0.01 and 0.58 meters were recorded.

Table 4.5 shows a comparison between measured ground truth, localization estimate, and 3D scan registration estimate over the course of the experiment. This comparison consists of the estimated relative distance between key-views.

Similar to the garage experiment, the localization update rate was measured between 30 Hz and 35 Hz. Figure 4.23 shows the tracked trajectory of the robot during the experiment compared to pure visual odometry along with 3D scan registration poses. The corresponding point cloud generated using the final pose estimates is shown in Figure 4.24.

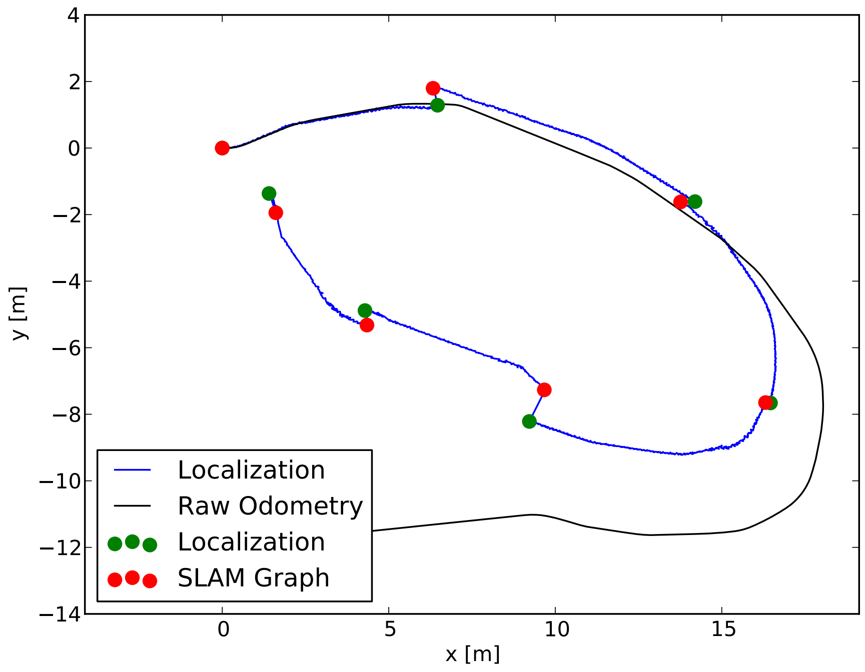


FIGURE 4.23: Courtyard experiment's tracked vehicle movement viewed from above - blue path. Black path - raw odometry input. Green circles - the tracking estimate when starting a 3D registration. Red circles - the registration estimate.

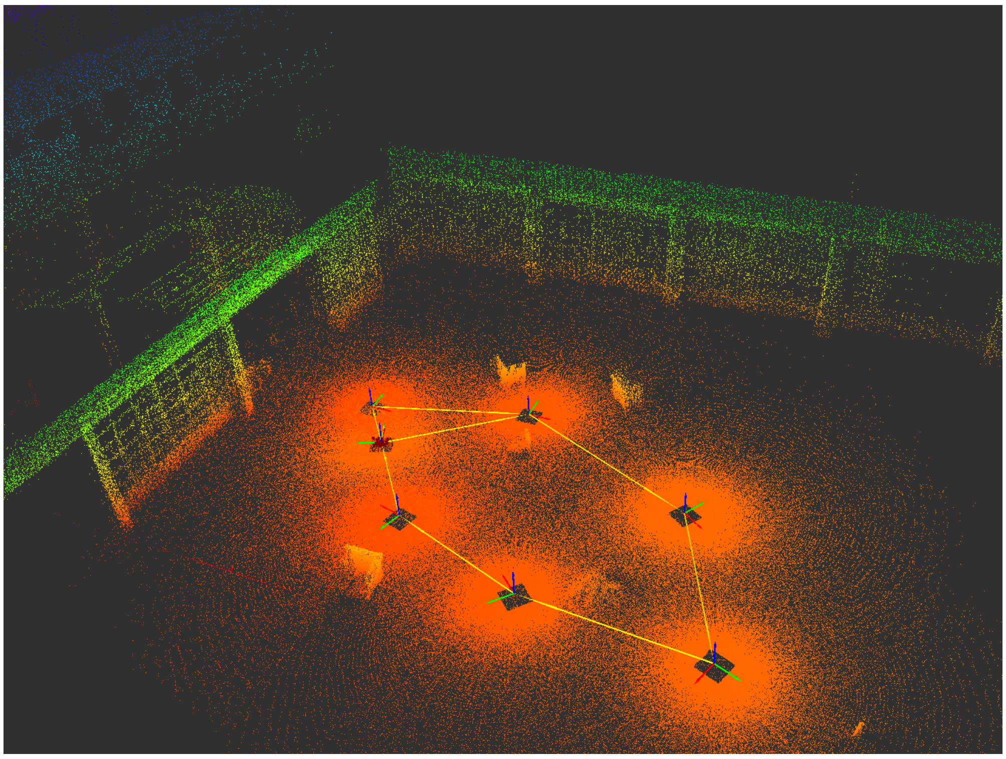


FIGURE 4.24: Courtyard experiment registered point clouds coloured by height for visualization.

4.6 Spacebot Cup Arena

The Spacebot Cup competition arena, shown in Figure 4.25, modelled a terra-planet environment potentially encountered during space exploration. The arena consisted of rough terrain with various obstacles such as rocks and even a "lander base station". The terrain consisted of various types of materials including sand, stones, and compacted dirt adding to the complexity of wheel odometry and tracking systems.

The arena environment was indoors and thus contained features not reliably present in space exploration applications - including a ceiling and walls. To show robustness and applicability of the entire system, from each recorded dataset two versions have been created. In the following sections the "un-cut arena" refers to the original laser data which includes all building features. Similarly the "cut

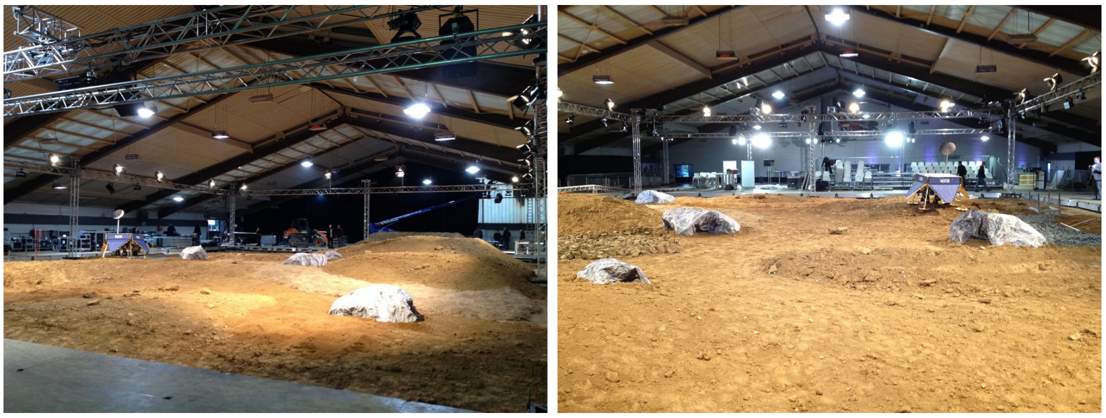


FIGURE 4.25: The Spacebot Cup arena.

arena" will reference laser data which has been filtered to remove all measurements outside the arena. [5]

Due to the nature of the Spacebot Cup competition, additional robots and researchers were present during data recording. Thus the laser data contains erroneous measurements mainly resulting from people moving within the arena during scanning. These points have not been filtered from the datasets and is specifically discussed where evaluation is impacted.

Using the Spacebot arena as a testing environment, two separate runs were recorded which included visual odometry and laser scanner data. Due to no additional means of measurement, no ground truth data is available. However from visual inspection of the aligned points clouds and comparison to the tracked trajectory and supplied odometry, these results show the qualitative accuracy of the system.

Run 1 consists of 8 3D scans taken over a duration of 854 seconds. The robot trajectory has a length of ˜ 80 meters. Run 2 has a duration of 971 seconds and consists of 14 3D scans. This run has a total traveled distance of ˜ 75 meters.

During each run, problems with the laser scanner driver resulted in invalid measurement readings. This problem is detectable within the data and thus during these outages no localization or scan aggregation is performed. Run 1 has a 45 second period where laser data is invalid and occurs during motion between two 3D

[5] Walls outside the arena bounds and the ceiling were removed from the laser data. This was achieved using scan registration results from the initial experiments where all laser data was processed.

Parameter	Value
Particles	250
Translation Sampling Variance Min	0.00625
Translation Sampling Variance Factor	0.08
Rotational Sampling Variance Min	0.0025
Rotational Sampling Variance Factor	0.15

Table 4.6: Particle filter configuration for Spacebot Cup arena experiments.

scan poses. Effects from this issue is specifically addressed in the results discussion below. Run 2 has a 25 second period of invalid laser data and occurs during the start of a scan and can be safely ignored without impacting the system results.

4.6.1 Particle Filtering Configuration

Table 4.6 presents the particle filter configuration used during Spacebot Cup experimentation. The values were experimentally derived during development of the Spacebot platform and tuned from initial data captured at the Spacebot Cup. These settings were used for all Spacebot arena dataset experiments. See Section 3.3.4 for a discussion on the parameters and effects.

4.6.2 SLAM

To test SLAM within an unknown environment, both recorded runs were processed on-line using qualitative analysis to judge accuracy. As previously mentioned, for each run an “un-cut” and “cut” dataset has been recorded.

Figure 4.26 and Figure 4.27 show the aligned 3D scans and tracked trajectory for both run 1 and 2 respectively. Due to a highly dynamic environment from people and robots combined with their proximity to the robot, the run 1 “cut” dataset is unable to be processed.

Within the aligned point cloud, one can see fine-grain details such as the rocky slope of an incline or the surrounding arena structure. Additional features such

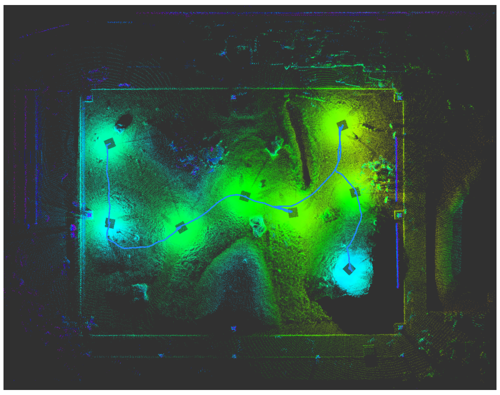


FIGURE 4.26: Spacebot arena run 1 aligned point cloud and tracked path. The ceiling has been removed from the uncut scan for viewing purposes.

as large boulders or the elongated ditch are also clearly visible. The mapping results are considered accurate through visual inspection and a comparison with arena photographs. In addition, there is no clearly visible difference between the aligned scans of the "un-cut" and "cut" datasets. This indicates that even without enclosing features such as ceilings and walls the mapping system can robustly handle open-air environments.

Inspecting the tracked path highlighted in blue reveals details about the localization component while performing SLAM. At each 3D scan location, a rectangular area is visible within the aligned point clouds. The center of the rectangular area is the robot pose according to the mapping system. If the localization gave perfect estimates, the localization path would pass directly through this center. After analyzing each of the 3D scan poses, it is clear the localization system adequately performs for mapping. Localization had a consistent update rate of about 25 Hz.

There is a noticeable difference between the "un-cut" and "cut" datasets for tracking. The tracked path is both smoother and more accurate when using the "un-cut" dataset, most likely due to sharp ceiling and wall features useful for localization.

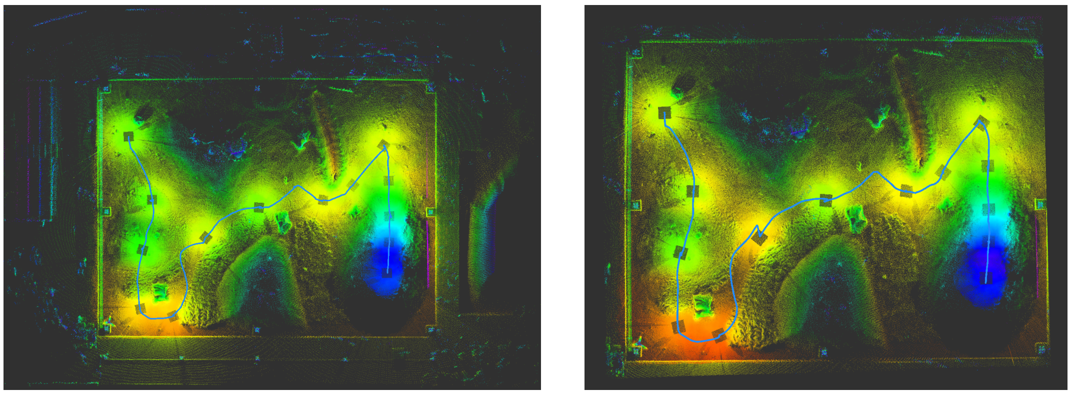
This qualitative analysis of localization is further founded by the results shown in Table 4.7 and Table 4.8. These tables present the error in meters between the localization estimate and registration estimate for each 3D scan from run 1 and 2 respectively.

3D Scan	Localization Guess Error (meters) Uncut Dataset
0-1	0.200
1-2	0.108
2-3	0.052
3-4	0.086
4-5	0.056
5-6	0.151
Average	0.109

Table 4.7: Localization guess error compared to mapping during SLAM for Spacebot Cup Run 1

	Localization Guess Error (meters)	
3D Scan	Uncut Dataset	Cut Dataset
0-1	0.210	0.032
1-2	0.051	0.063
2-3	0.072	0.093
3-4	0.126	0.143
4-5	0.050	1.087
5-6	0.165	0.191
6-7	0.070	0.273
7-8	0.051	0.280
8-9	0.023	0.608
9-10	0.044	0.291
10-11	0.077	0.367
11-12	0.022	0.417
12-13	0.040	0.334
Average	0.077	0.321

Table 4.8: Localization guess error compared to mapping during SLAM for Spacebot Cup Run 2


(A) Full uncut scan.

(B) Cut scan.

Figure 4.27: Spacebot arena run 2 aligned point cloud and tracked path. The ceiling has been removed from the uncut scan for viewing purposes.

4.6.3 Drivability Assessment and Path Planning

The Spacebot arena was also processed for drivability characteristics and used for trajectory planning evaluation. Due to the method of combining 3D scans for drivability analysis presented in Section 3.4 the results have been impacted by erroneous measurements of people. Map locations where drivability evaluation does not have a clear correspondence to mapping results are due to these unavoidable measurements.

Figure 4.28 shows the drivability map generated from the first Spacebot arena dataset. The aligned point cloud is also shown for a side-by-side comparison. Figure 4.29 shows the drivability map for the second dataset. Within this figure both the "un-cut" and "cut" datasets have been processed.

Using drivability assessment previously demonstrated, several trajectories have been planned from several source and goal locations - including planning to special cases of unmapped environment and obstacles. Figure 4.30 show two example paths planned between "Start" and "Target" positions.

Figure 4.31 uses the drivability assessment data from the run 2 "cut" dataset to show handling of special cases - specifically planning to unknown space and within obstacles. As shown in the figure, the planning component successfully creates a

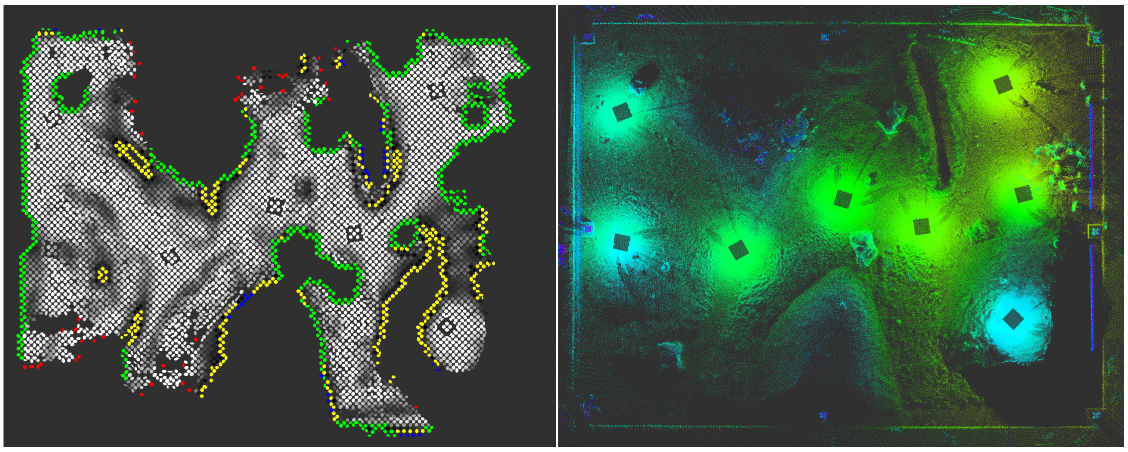

(A) Drivability map from uncut dataset. (B) Corresponding aligned point cloud.

FIGURE 4.28: The drivability assessment results from the first Spacebot arena dataset. The drivability map is color-coded to allow for easy readability. Green: maximum bumpiness threshold reached. Blue: maximum incline threshold reached. Red: minimum region coverage threshold reached. Yellow: total cost threshold reached. Gray scale: Valid surfels where white indicates lower cost.

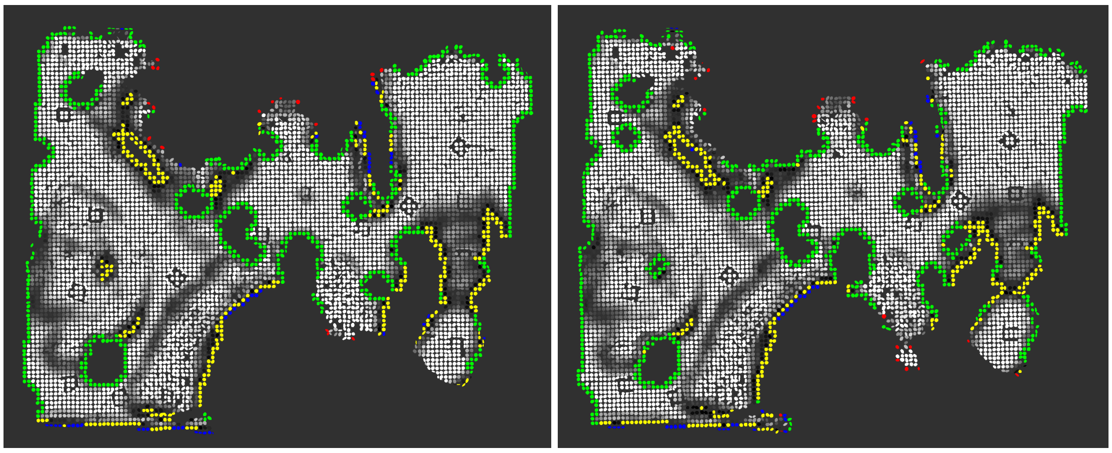

(A) Uncut dataset. (B) Cut dataset.

FIGURE 4.29: The drivability assessment results from the second Spacebot arena dataset. The drivability map is color-coded to allow for easy readability. Green: maximum bumpiness threshold reached. Blue: maximum incline threshold reached. Red: minimum region coverage threshold reached. Yellow: total cost threshold reached. Gray scale: Valid surfels where white indicates lower cost.

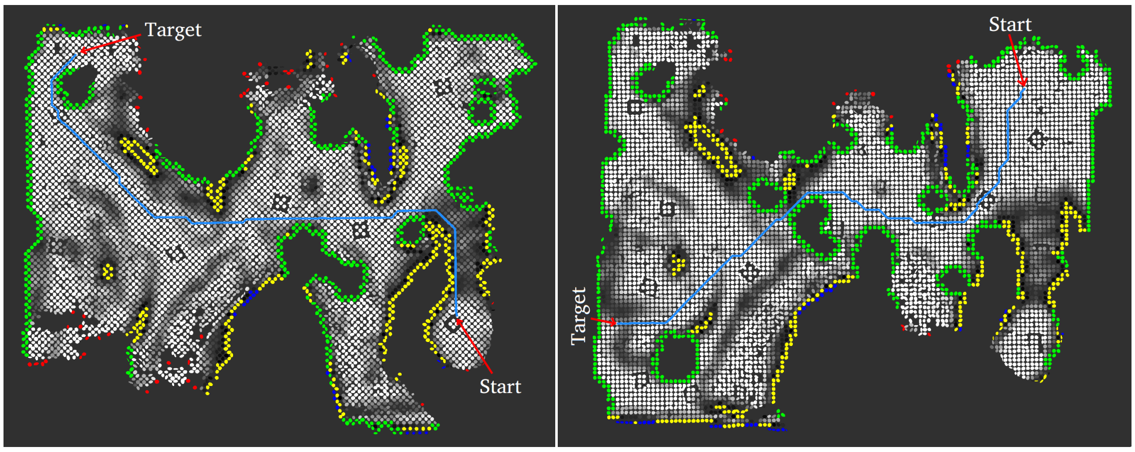
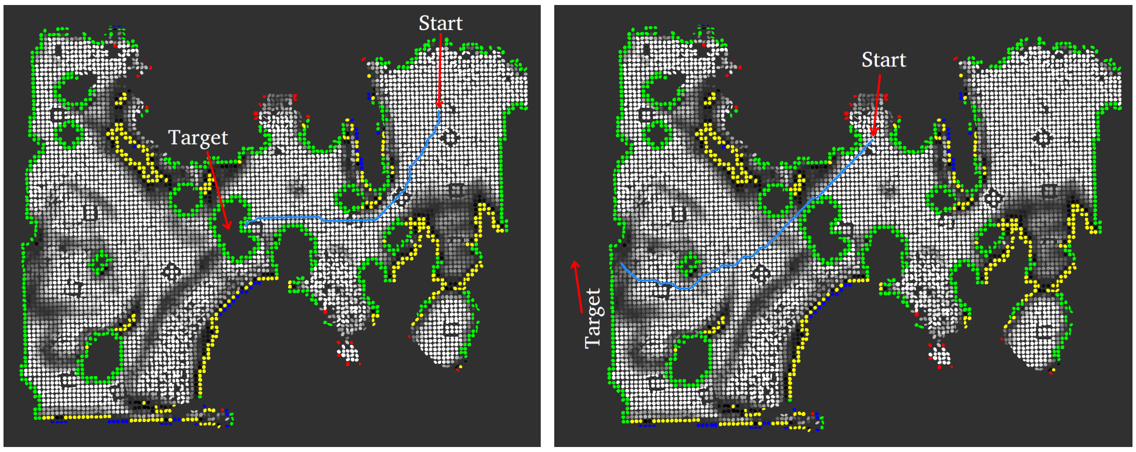


FIGURE 4.30: Two example trajectories from the planning component.



(A) Planning to obstacle. (B) Planning to unknown space.

FIGURE 4.31: Two example trajectories from the planning component where the target location cannot be reached.

trajectory leading to the general desired target area that is reachable by the robot given the current mapping information.

4.7 Summary

This chapter presented the experimental results of the SLAM and navigation systems in both simulation and real-world environments.

Simulation experiments presented in Section 4.2 show the accuracy of the SLAM system. After averaging results over five simulation experiments, the localization system had an average ATE of 0.117 m. Using the same experiment data, the mapping system had an average ATE of 0.029 m.

An external dataset has been processed by the mapping system in Section 4.3. While no comparison to ground truth data is made, the resulting aligned point clouds contain visible evidence of mapping accuracy.

Additional real-world experiments of the mapping and localization system are presented in Section 4.4 and Section 4.5. From these experiments, the localization-based registration guess had an average error of 0.29 m and 0.20 m when comparing relative distances between scans for the garage and courtyard experiment respectively. Using the same data, the mapping system had an average error of 0.07 m and 0.03 m.

Results from the DLR Spacebot Cup are presented in Section 4.6. Qualitative results from SLAM and drivability assessment are presented including example planned trajectories and aligned point clouds.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis has presented an approach for SLAM, drivability assessment, and navigation planning within the context of the Explorer robot platform. A continuously rotating 2D laser scanner is used as the sensing system for both mapping and localization.

Using a particle filter and an observation calculation from single 2D laser scan lines, the localization system can suitably track the robot pose for mapping and planning activities. In simulation experiments, localization estimates had an average absolute trajectory error of 12 cm. Real world experiments recorded at the University of Bonn, show a localization error of 20 cm and 29 cm when compared to mapping estimates. Throughout testing, this system ran in real-time with an update frequency between 25-35 Hz.

Multi-resolution surfel maps allow for compact modelling of complex environments and an accurate mapping system. The MRS map implementation has been adapted to model surfaces based on the error properties of the Hokuyo UTM-30LX-EW laser scanner. In simulation, 3D scan registration had an average error of 3 cm when compared to ground truth information. In real world environments, the mapping system produced errors of 7 cm and 3 cm when compared to hand-measured distances.

Drivability assessment is performed within MRS maps by analyzing several drivability factors. These factors include terrain bumpiness, incline, and robot coverage. Through the precomputation of costs, efficient path planning can be performed using the A* algorithm.

Experiments have demonstrated the accuracy of the SLAM and navigation systems. Simulation experiments have allowed for precise ground truth comparisons of mapping and localization results. Complex real-world experiments have tested the ability of the tracking system to initialize 3D scan registration as well as the robustness of map alignment. A freely available dataset from the University of Toronto is also used to show qualitative mapping results. Data from the DLR Spacebot Cup show the applicability of the system to land-rover type platforms.

5.2 Future Work

A number of promising options are available for future work relating to the presented SLAM and navigation system.

The first route would be to calculate an improved proposal distribution for use in the localization system. The current system uses bootstrap filtering and thus the state propagation model as the proposal distribution. By using ICP methods, each laser scan line could be registered to the MRS map. This would allow for an improved proposal distribution.

A second direction of future work would be to implement occupancy mapping within MRS maps. This would allow for the distinction between free and unknown space. This additional information could be used to improve the observation models used by the mapping and localization systems.

Appendix A

Fundamentals

A.1 Non-linear Least Squares Optimization

Parameter optimization wishes to minimize the weighted sum of squared error between a set of truth values $\boldsymbol{y} = y_1, y_2, \cdots, y_n$ and an assumed non-linear function $f(x)$ parameterized by $\boldsymbol{x} = x_1, x_2, \cdots, x_m$. The error for instance i is defined as

$$e_i(\boldsymbol{x}) = y_i - f_i(\boldsymbol{x}). \tag{A.1}$$

We define the total error function $S(x)$ as

$$S(x) = \sum_i^n e_i^2. \tag{A.2}$$

We find the parameters $\boldsymbol{x}$ that model the truth values $\boldsymbol{y}$ by minimizing this function. Function minima occur when the function gradient $\frac{\partial S}{\partial \boldsymbol{x}} = 0$.

This section presents common methods used to iteratively compute the parameters $\boldsymbol{x}$ where the gradient is zero.

A.1.1 Gradient Descent

Graident descent iteratively adjusts the parameters $\boldsymbol{x}$ in the direction of the error function gradient. Iterative values of the parameters at time $t + 1$ are computed as

$$x_{t+1}^{(i)} = x_t^{(i)} - \alpha \frac{\partial S}{\partial x^{(i)}}, \tag{A.3}$$

where α is the adjustment rate parameter.

This method converges to a local minimum however when very close the convergence rate is relatively slow.

A.1.2 Newton's Method

Newton's Method is used to find stationary points of a function. A stationary point exists where the gradient is zero and thus this algorithm can be applied to our optimization problem. Newton's Method iteratively approximates the objective function by modelling a quadratic function about the current parameter value. The iterative update is given by

$$x_{t+1}^{(i)} = x_t^{(i)} - \alpha H_S^{-1} J, \tag{A.4}$$

where α is the adjustment rate parameter, H_S is the Hessian matrix of second derivatives of S, and J is $\frac{\partial S}{\partial \boldsymbol{x}}$ - the jacobian of S.

Newton's Method converages faster than gradient descent when close to stationary points.

A.1.3 Gauss Newton Method

The Gauss Newton method is a variant of Newton's Method were the objective function is assumed to be quadratic near stationary points. This method approximates the Hessian as $H_S = J^T W J$ where J is the Jacobian matrix and W is a diagonal weighting matrix for the residuals e_i.

The Gauss Newton iterative update is computed as

$$x_{t+1}^{(i)} = [J^T W J]^{-1} J^T W e_i(x). \tag{A.5}$$

This method converges faster than Gradient Descent when close to the minimum.

A.1.4 Levenberg-Marquardt Method

The Levenberg-Marquardt method implemented damped Gauss Newton iterations by inserting an additional parameter. The update is now defined as

$$x_{t+1}^{(i)} = [J^T W J + \alpha I]^{-1} J^T W e_i(x), \tag{A.6}$$

where α is the adjustment rate parameter. When α is large, this update reduces to Gradient Descent. When α is small the resulting update performs similar to the Gauss Newton Method.

Through varying the parameter α, this method can use the benefits of both Gradient Descent and Gauss Newton. Fast convergence with Gradient Descent when far from the minimum; fast convergence with the Gauss Newton Method when close to the minimum.

A.2 Quaternion Rotations

The rotation of vector $\vec{e}$ by quaternion $q = (s, \vec{v})$ in $\mathbb{R}^3$ is defined as

$$\vec{v'} = qv\bar{q}, \tag{A.7}$$

where $\bar{q}$ is the conjugate of quaternion q.

Multiplication between a quaternion and vector is performed by converting the quaternion to its rotation matrix form. For a quaternion $q = (w, x, y, z)$ where w is the scalar component referenced as s above, we define the 3x3 rotation matrix as

$$M_q = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + xw & 1 - 2x^2 - 2y^2 \end{bmatrix}. \tag{A.8}$$

Using this definition, multiplication with a quaternion and thus a rotation reduces to simple matrix multiplication.

A.3 Probability Density Functions

This sections presents a short introduction to the law of probability used for working with probability density functions on random variables. A probability density function (pdf) is a function that models the probability for a random variable to take a specific value. A pdf modelled by $p(x)$ has the following properties:

1. Total Probability - The integral over the entire variable state space is equal to 1. This essentially means the true value of the variable is within the state space. In a discrete state space, we can define this property as the following constraint
$$\sum_{x \in \boldsymbol{X}} p(x) = 1 \tag{A.9}$$

2. Non-negative - The probability density has a non-negative value for the entire state space.

A.3.1 Marginalization

Marginalization uses knowledge of an external random variable y with a dependence to the random variable x to calculate information about x. We can reformulate the pdf of x, $p(x)$ in terms of y as

$$p(x) = \sum_{y \in \boldsymbol{X}} p(x, y). \tag{A.10}$$

Using the Bayes' formula, we can rewrite this definition to use conditional probabilities. This form is shown in Equation A.11.

$$p(x) = \sum_{y \in \boldsymbol{X}} p(x|y)p(y) \tag{A.11}$$

A.3.2 Bayes' Formula

Bayes formula allows one to express a posterior probability $p(x|y)$ using a prior probability $p(x)$ and a likelihood probability $p(y|x)$. This formula is used in Bayesian filtering to simplify random variable calculations and is heavily applied in the derivation of particle filtering. Bayes' rule is defined as

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \tag{A.12}$$

where $p(y)$ is a normalization term.

Bibliography

LTD Hokuyo Autonomatic CO. Scanner laser range finder utm-30lx specification. `http://wiki.ros.org/hokuyo_node?action=AttachFile&do=get&target=UTM-30LX_Specification.pdf`, 2008.

Y. Roth-Tabak and R. Jain. Building an environment model using depth information. *Computer*, 22(6):85–90, 1989. ISSN 0018-9162. doi: 10.1109/2.30724.

Armin Hornung, KaiM. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. ISSN 0929-5593. doi: 10.1007/s10514-012-9321-0. URL `http://dx.doi.org/10.1007/s10514-012-9321-0`.

Julian Ryde and Huosheng Hu. 3d mapping with multi-resolution occupied voxel lists. *Autonomous Robots*, 28(2):169–185, 2010. ISSN 0929-5593. doi: 10.1007/s10514-009-9158-3. URL `http://dx.doi.org/10.1007/s10514-009-9158-3`.

P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2743–2748 vol.3, 2003. doi: 10.1109/IROS.2003.1249285.

M. Magnusson, T. Duckett, and A.J. Lilienthal. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007.

Todor Stoyanov, Martin Magnusson, and Achim J. Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31:1377–1393, 2012. doi: 10.1177/0278364912460895.

Jari P. Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J. Lilienthal. 3d normal distributions transform occupancy maps: an efficient representation for mapping in dynamic environments. *The International Journal of Robotics Research*, 2013. doi: 10.1177/0278364913499415. URL `http://ijr.sagepub.com/content/early/2013/09/16/0278364913499415.abstract`.

Michael Bosse and Robert Zlot. Continuous 3D scan-matching with a spinning 2D laser. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4312–4319, 2009.

J. Elseberg, D. Borrmann, and A. Nuechter. 6DOF semi-rigid SLAM for mobile scanning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1865–1870, 2012.

T. Stoyanov and A.J. Lilienthal. Maximum likelihood point cloud acquisition from a mobile platform. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, pages 1–6, 2009.

Will Maddern, Alastair Harrison, and Paul Newman. Lost in translation (and rotation): Fast extrinsic calibration for 2D and 3D LIDARs. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2012.

Sean Anderson and Timothy D. Barfoot. Towards relative continuous-time SLAM. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1033–1040, 2013.

Kourosh Khoshelham. Automated localization of a laser scanner in indoor environments using planar objects. In *Int. Conf. on Indoor Positioning and Navigation*, 2010.

R. Kuemmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte Carlo localization in outdoor terrains using multi-level surface maps. In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, 2007.

R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

J. Klaess, J. Stueckler, and S. Behnke. Efficient mobile robot navigation using 3D surfel grid maps. In *Proc. German Conf. on Robotics (ROBOTIK)*, 2012.

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam&mdash;3d mapping outdoor environments: Research articles. *J. Field Robot.*, 24(8-9):699–722, August 2007. ISSN 1556-4959. doi: 10.1002/rob.v24:8/9. URL `http://dx.doi.org/10.1002/rob.v24:8/9`.

Armin Hornung, Mike Phillips, Edward Gil Jones, Maren Bennewitz, Maxim Likhachev, and Sachin Chitta. Navigation in three-dimensional cluttered environments for mobile manipulation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 05/2012 2012.

Jinhan Lee, C. Pippin, and T. Balch. Cost based planning with rrt in outdoor environments. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 684–689, 2008.

Brian P. Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *Proc. of the Workshop on Path Planning on Costmaps, IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

Dave Ferguson and Maxim Likhachev. Efficiently using cost maps for planning complex maneuvers. In *Proc. of the Workshop on Planning with Cost Maps, IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J. Lilienthal. Path Planning in 3D Environments using the Normal Distributions Transform. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

NOAA Costal Services Center. Lidar 101: An introduction to lidar technology, data, and applications, 2012.

Joel A. Shapiro. Classical mechanics, 2010.

J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 2013.

T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computer sample variances. 1979.

A. Censi. An accurate closed-form estimate of ICP's covariance. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3167–3172, 2007.

R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3607–3613, 2011.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, pages 35 – 45, 1960.

G.L. Smith, S.F. Schmidt, and L.A. McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. 1962.

Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *International Sympossium Aerospace/Defense Sensing, Simulation, and Controls 3*, 1997.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proc. of the Int. Symp. on Robotics Research (ISRR)*, 2011.

LTD Hokuyo Autonomatic CO. Scanner laser range finder utm-30lx-ew specification. `http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/utm-30lx-ew/data/UTM-30LX-EW_spec_en.pdf`, 2012.

Eric W. Weisstein. von neumann neighborhood., a. URL `http://mathworld.wolfram.com/vonNeumannNeighborhood.html`.

Eric W. Weisstein. Moore neighborhood., b. URL `http://mathworld.wolfram.com/MooreNeighborhood.html`.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2012.

B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, 1987.

C. Tong, D. Gingras, K. Larose, T.D. Barfoot, and E. Dupuis. The canadian planetary emulation terrain 3d mapping dataset. *Int. Journal of Robotics Research (IJRR)*, 2013.

C. Tong, T.D. Barfoot, and E. Dupuis. 3d slam for mapping planetary worksite environments. *Journal of Field Robotics, Special Issue on "Space Robotics"*, 2011.