# ABSTRACT


Title of dissertation: ACCURATE SLAM WITH APPLICATION FOR AERIAL PATH PLANNING

Chen Friedman , Doctor of Philosophy, 2013

Dissertation directed by: Professor Inderjit Chopra
Alfred Gessow Rotorcraft Center
Department of Aerospace Engineering
University of Maryland

and: Professor Omri Rand
Faculty of Aerospace Engineering
Technion–Israel Institute of Technology



This thesis focuses on operation of Micro Aerial Vehicles (MAVs), in previously unexplored, GPS-denied environments. For this purpose, a refined Simultaneous Localization And Mapping (SLAM) algorithm using a laser range scanner is developed, capable of producing a map of the traversed environment, and estimating the position of the MAV within the evolving map. The algorithm's accuracy is quantitatively assessed using several dedicated metrics, showing significant advantages over current methods. Repeatability and robustness are shown using a set of 12 repeated experiments in a benchmark scenario.

The SLAM algorithm is primarily based on an innovative scan matching approach, dubbed Perimeter Based Polar Scan Matching (PB-PSM), which introduces a maximum overlap term to the cost function. This term, along with a tailored cost minimization technique, are found to yield highly accurate solutions for scan matching

pairs of range scans. The algorithm is extensively tested on both ground and aerial platforms, in indoor as well as outdoor scenarios, using both in-house and previously published datasets, utilizing several different laser scanners.

The SLAM algorithm is then coupled with a global A* path planner, and applied on a single rotor helicopter, performing targeted flight missions using a pilot-in-the-loop implementation. Targeted flight is defined as navigating to a goal position, defined by relative distance from a known initial position. It differs from the more common task of mapping, as it may not rely on loop closure opportunities to smooth out errors and optimize the generated map. Therefore, the importance of position estimates accuracy increases dramatically.

The complete algorithm is then used for targeted flight experiments with a pilot in the loop. The algorithm presents the pilot with nothing but heading information. In order to further prevent the pilot from interfering with the obstacle avoidance task, the evolving map and position are not shown to the human pilot. Furthermore, the scenario is introduced with artificial (invisible) obstacles, apparent only to the path planner. The pilot therefore has to adhere to the path planner directions in order to reach the goal while avoiding all obstacles. The resulting paths show the helicopter successfully avoid both real and artificial obstacles, while following the planned path to the goal.

# ACCURATE SLAM WITH APPLICATION FOR AERIAL PATH PLANNING

by

Chen Friedman

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Inderjit Chopra, Chair/Advisor
Professor Omri Rand, Co-Advisor
Professor Norman Wereley
Professor Robert Sanner
Professor Derek Paley
Professor Rama Chellappa

# Acknowledgments

I would love to take this opportunity to thank the people who helped make this thesis possible. This experience has definitely been one for the books, and it would not be the same without the endless support I received from my loving wife Mor. As one of UMD's aerospace graduate herself, Mor was always there to hear both my success stories and my rants. Mor was always able to lend an ear, even when she knew (almost) exactly what I was about to say.

My loving parents were a true positive influence. Although both are not so technical people, they had quite an interest in me sharing this experience with them.

As for people directly responsible for the success of this thesis, my sincere thanks goes to Shane Boyer, the MAV pilot who had the audacity to attempt flying indoors, through corridors not much larger than the helicopter's rotor itself, while tethering the helicopter to a 10 *meters* data cord. I see a direct connection between the success of any helicopter-related results in this thesis and Shane's flying capabilities. His endless availability and willingness to attempt all of my rather crazy experimental ideas, as well as his dedication and interest in my project have played a key role in the aerial portion of this thesis.

I'd like to thank Graham Bowen-Davies for being the friend that he is. As part of the same research group, Graham was often my address to run ideas by, invent experimental techniques, and solutions that satisfied various requirements of this project. Graham had the patience to endure my spirit while contributing original ideas, and checking extrinsic issues, which were out of the group's scope, in most

cases.

My thanks is also extended to all the students at the rotorcraft center, specifically those who shared the weekly meeting experience, and contributed from their time to ask questions, offer ideas, and criticize my work. Same goes for those students who simply were at UMD at the same time, to share the Ph.D. degree experience with me. Some of these students include (alphabetical order): Brandon Bush, Jared Grauer, Jörgen Rauleder, Kumar Ravichandran, Robert Vocke III, and Kan Yang.

It may seem odd, but many parts of this thesis work were carried out outside the office, in one of many air-conditioned branches of "Starbucks Coffee". This includes all aspects of coding, debugging, and thesis writing (much to the astonishment of my peers). The great work environment is hereby much appreciated, as is the great coffee and occasional snacks. This "thank you" is also extended to the understanding and kind employees, who did not kick me out after many 5 hour straight shifts, with no less than two laptops on one table.

Finally, I'd like to thank both my advisors for their contribution towards this thesis. Thanks is due to Dr. Omri Rand, for his advice throughout this process, his thoughts and ideas, and his willingness to travel many times to the USA. The funding for this projects came with much help from Dr. Inderjit Chopra, who secured funding for both the project and my degree. I extend my thanks to him for that as well.

# Contents

# List of Tables

# List of Figures

# List of symbols

| | |
|---|---|
| $C_{k,l}$ | The current occupancy of the $[k, l]$ cell |
| $F_i$ | The $i^{th}$ point's cost contribution |
| $I$ | The index of the final representative cell for a virtual ray casting operation |
| $N_{cells}$ | The number of occupied cells found by the current cast ray |
| $N_{empty}$ | Threshold for defining the beginning of a new object in a ray casting operation |
| $N_{thickness}$ | Threshold for the number of occupied cells along a single ray that can be considered as a single obstacle |
| $P$ | Perimeter length created by all the points that were successfully matched in the Current scan |
| $P_0$ | Perimeter length created by all valid Reference Scan points |
| $R_{min}, R_{max}$ | Minimum/maximum considered range, respectively |
| $T_E$ | Threshold for eliminating a point's cost contribution ("matching anomaly" |
| $T_F$ | Threshold for disqualifying a scan matching solution |
| $T_M$ | threshold for successfully matched points |
| $T_O$ | Occupancy threshold for the isolated point filter |
| $V_1, V_2$ | Linear translation velocity of the right and left driven wheels of a ground platform, respectively |
| $W$ | The occupancy value of a cell in the occupancy grid |
| $d$ | The occupancy grid's resolution |
| $f$ | Total cost function value |
| $n_c$ | Total number of points that contribute to a given cost function (used for normalization) |
| $n_s$ | Parameter for the size of the inspected area for the isolated point filter |
| $r$ | Range measurement |
| $x, y, z$ | Position coordinates in x, y, and z directions, respectively |
| $\Delta x, \Delta y$ | Translation distance in x and y directions, respectively |
| $\alpha$ | Angle definition used in the outlier filter |
| $\psi$ | Platform's azimuth angle |
| $\Delta\psi$ | Change in azimuth |

| | |
|---|---|
| $\sigma$ | Standard deviation for a laser range measurement |
| $\theta$ | A scan point angle |
| $\dot{()}$ | Derivative with respect to time |
| $()'$ | Roto-translated coordinate system |
| $()''$ | Interpolated values in the roto-translated coordinate system |
| $()_C, ()_R$ | Quantity with respect to the Current and Reference scan, respectively, during scan matching |
| $()_L$ | Quantity with respect to a laser scan |
| $()_V$ | Quantity with respect to a virtual scan |
| $()_g$ | Global reference frame |

# Chapter 1

# Introduction

The motivation for this work is to provide the capability for path planning missions using rotary-wing micro aerial vehicles (MAVs), in previously unexplored, GPS-denied environments. In particular, targeted flight is considered as a path planning case study. Targeted flight is defined as navigating to a goal position, defined by relative distance from a known initial position. Operation in GPS denied environments is imperative for a variety of unmanned vehicle mission scenarios including surveillance, search and rescue, and biological chemical agent detection.

A successful targeted flight mission would accurately navigate a robotic platform from an initial point to a final point, while avoiding obstacles that are discovered along the traversed path. Targeted flight differs from the more common task of mapping, as it may not rely on returning to previously explored areas, in order to obtain opportunities for loop closure, which may be used to smooth out errors and optimize the generated map. Therefore, position estimates accuracy becomes a necessity.

Therefore, targeted flight requires a highly accurate mapping and localization algorithm, which may not depend on loop closure opportunities. This is typically obtained using a combination of accurate sensors and algorithms. The mapping and position estimates are required to be accurate throughout the traversed path. Targeted flight may not depend upon optimizing the map for error reduction. In addition, an algorithm for obstacle avoidance and path planning is required to navigate the platform towards the final target.

In this thesis, a refined Simultaneous Localization And Mapping (SLAM) [1] algorithm is developed, and its accuracy is quantitatively assessed using a dedicated metric, showing comparative advantages over current methods. Repeatability and robustness are checked using a set of 12 repeated experiments in a benchmark scenario.

The SLAM algorithm is then coupled with a global A* path planner, and applied on a single rotor helicopter, performing targeted flight missions using a pilot-in-the-loop implementation. The pilot is provided with only heading information and artificial (invisible) obstacles are introduced, apparent only to the path planner, to prevent the pilot from interfering with the obstacle avoidance task. The resulting path shows the helicopter successfully avoid both real and artificial obstacles, while following the planned path to the goal.

In the current chapter, a general description of autonomous operation in GPS-

denied environments is given, followed by a literature review of both SLAM techniques and scan matching, highlighting the potential advantages and possible shortcomings of various currently available algorithms. The constituting elements of SLAM are reviewed with respect to sensors, platforms, the loop closure problem, and several algorithm identifiers.

One of the main contributions of this thesis is a refined highly accurate scan matching technique, for performing SLAM. Therefore, the available literature will be reviewed focusing primarily on the accuracy reported by other authors using different SLAM algorithms. The rationale for the required accuracy is also discussed, particularly with respect to targeted flight mission types.

## 1.1 Autonomous Operation in GPS Denied Environments

For an unmanned Micro Aerial Vehicle (MAV) operation, assuming position information is given (GPS, beacons, etc.), on board sensory data (*e.g.* laser scanner, vision sensor, etc.) can be combined to build a map (since the position information input is independent of sensor's readings). On the other hand, if the map is known a priori, the MAV can position itself inside the known map using sensory data and appropriate tools (such as scan matching). These techniques are quite robust as the map is known and serves as a deterministic anchor in the estimation process.

However, when both the map and position are unknown, the MAV is required to simultaneously estimate both at the same time. A map can be built by sequentially adding sensory data (*e.g.* laser scans, visual pictures, extracted features, etc.) to a database, as the MAV moves in space. The sensory data is added based on the estimated location of the platform and the map is interchangeably used to estimate position. Thus, estimating the map and the position within the map is a joint estimation process, which is widely referred to as Simultaneous Localization And Mapping [1, 2].

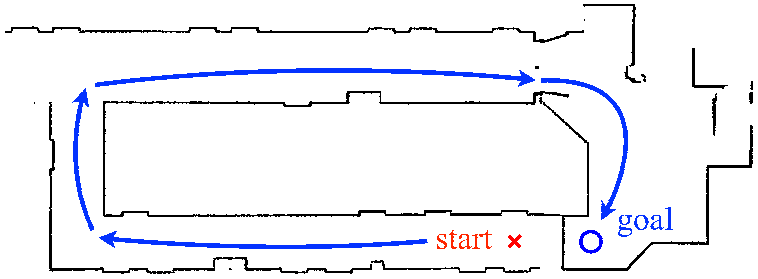
SLAM is therefore a bootstrapping process, coupling position and map estimations. As such, the accuracy of the map is imperative for accurate position estimates, and simultaneously, estimated position information must be precise enough to allow accurate map evolution. Inaccurate position would register measurements of the environment in the wrong locations in the map, and thus would degrade the map's accuracy. Therefore, all subsequent position estimates would be prone to errors if carried out using an inaccurate map.

### 1.1.1 Mapping Vs Targeted Flight

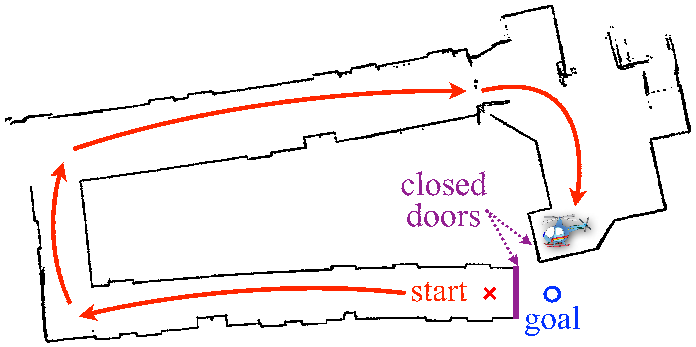
The task of mapping an environment relies on returning to previously visited locations. The drift that accumulates while a platform maps a given area may be only be reduced if an added constraint is introduced. This may be obtained when a relation is discovered between a currently explored area and a previously explored

area. Such an occasion occurs when the platform revisits previously explored areas, in which case, new incoming data has overlap with the already evolving map.

Loop closure algorithms are capable of discovering such possible overlaps. These algorithms may be based on feature recognition, or basic scan matching (see Sub-Section 1.2.5 for a detailed review). The main result is a constrained relation between pose estimate pairs. An optimization algorithm may then be employed to solve a complete problem of relative constrained relations between complete sets of pose estimates to obtain map consistency, thus reducing the accumulated drift.




(a) Low drift algorithm map




(b) High drift algorithm map

Figure 1.1: Example for a scenario where loop closure opportunities may not exist, and large drift may prevent a successful mission. Initial and goal points marked with an 'x' and a circle, respectively. Mapping results with low and high drift algorithms (top and bottom, respectively). Robot path is marked by arrows.

In contrast, targeted flight may not rely on such algorithms, as loop closure opportunities may not exist. An illustration of such a scenario is presented in Figure 1.1. The robot is directed to reach a goal on the other side of a closed door (marked with a circle), defined as relative distance to the initial position (marked with an 'x'). The map on the left has very low drift, and so the robot can successfully plan a path to the goal. However, the map on the right, which has significant drift associated with it, would prevent the robot from reaching its goal, and completing the mission successfully. In such a scenario, loop closure may not be used to optimize the map, as the vehicle does not return to a previously visited area.

### 1.1.2 Importance of SLAM Accuracy

The accuracy of SLAM algorithms is poorly addressed in the majority of published literature, as will be shown laser in this chapter. The bootstrapping nature of SLAM, discussed above, results in a strong dependency of most algorithms on the ability to perform loop closure detection, and map optimization to reduce the accumulated drift (see Sub-Section 1.2.5 for additional details). The accuracy obtained before and after employing the map optimization algorithms differ significantly (typically accumulated drift becomes large enough to be easily recognized by the naked eye).

Algorithm accuracy is quite challenging to measure. The main reason is the size of the experimental scenarios, with dimensions that are typically on the order of 100 $m$. Motion capture systems may only be used on smaller scenarios, to provide ground truth data to which position estimates may be compared [3]. Typical larger scale experimental scenarios may be evaluated by comparing selected dimensions [4] or using GPS data in case of outdoor experiments [5].

In the published literature, accuracy is typically reported in a qualitative form. This will include images of the resulting map, which are compared with floor plans or aerial photographs (for either indoor or outdoor experiments, respectively). Due to the scenario size, the comparison is typically performed on a global scale, showing the entire scenario without focusing on local, smaller scale accuracy (examples may include the works by Bachrach et al. [6], and Grisetti [7]). In some cases, the true map is not presented, and neither quantitative nor qualitative comparison is provided [3]. In such cases, the consistency of the map, especially around loop closure opportunity areas is typically of a higher concern.

Map consistency is described as a correct loop closure of a traversed path, as examined by the naked eye. After loop closure and map optimization algorithm have been carried out, the resulting SLAM-generated map is analyzed with specific regard to the loop closure opportunities that were attempted. Algorithm comparison may in some cases be determined based on successful loop closure abilities [8, 9].

In the rare cases when accuracy is quantified, the reported results are for a simulated environment, as the information is available for both estimated and simulated true quantities, which makes comparison fairly easy. However, the accuracy in experiments conducted in real environments is significantly lower, as compared to that achieved using simulated data. This was clearly shown in the work by Segal, Hähnel, and Thrun [10], who employed the same algorithm on both simulated and true laser scanner data. They showed approximately an order of magnitude difference between the average error obtained using simulated data, as compared with true laser scanner data, collected both indoor and outdoor.

Improving the accuracy of the SLAM estimates (prior to the use of map optimization algorithms) may have a positive effect on both mapping and targeted flight missions. As explained in Sub-Section 1.1.1, high accuracy may be a key in the success of targeted flight mission in areas with absolutely no opportunities for loop closure. High drift algorithms may result in a mission failure. In mapping missions, the use of loop closure and map optimization algorithms may directly benefit from higher accuracy SLAM estimates in two ways: the higher accuracy will provide a better

initial guess for the loop closure algorithm when testing loop closure candidates, and the optimization algorithm will converge faster since the initial map to be optimized will be closer to the well optimized solution (which is a consistent map).

## 1.2 Background and Technical Challenges

This section reviews previously published algorithm for SLAM, scan matching, with specific regard to relative advantages of the different approaches, as well as the range of applicability with regards to autonomous platforms configurations. The sensors used for SLAM in the current state-of-the-art are also reviewed. The review will also outline the relatively little attention given to SLAM accuracy, an area that is thoroughly discussed in this thesis.

As mentioned above, the motivation of this research was to perform targeted flight, which requires accurate SLAM algorithms without relying on loop closure and map optimization capability. This section will highlight the currently available algorithms, their accuracy (reported or estimated by the author), which will support this research motivation.

### 1.2.1 SLAM Techniques

The area of SLAM has received a tremendous amount of attention over the last decade. The main reason was the desire to operate autonomous vehicles in a GPS-denied environments, thus allowing for indoor, underground, and in some cases a more robust autonomous vehicle operation (as GPS data may not be available at all times). Generally, a SLAM algorithm provides the ability to estimate the vehicle's state vector given its previous state in conjunction with available measurements and commands. This is combined with some model for the environmental observations (sensory data). The result is both a map of the scenario (in some form), and a full or partial vehicle state estimate.

There are several identifying properties that may be used to categorize SLAM algorithms:

- Extended Kalman Filter (EKF) based
- Particle Filter (PF) based (sequential Monte-Carlo sampling)
- Dynamic model dependent
- Dynamic model independent
- Scan matching based
- Two dimensional
- Three dimensional

### Extended Kalman Filter based SLAM

Extended Kalman filter SLAM algorithms are probabilistic based approaches, relying on some form of Bayesian filters. The main idea behind this algorithm category is obtaining an initial estimate (prior), which is based on a plant model for the platform dynamics, and the given commands, which is used to calculate a refined estimate (posterior) based on the new sensory input. Generally, the a probabilistic approach builds a coupled Bayesian filter equation, for both the vehicle's states and the map observations [1] (such as map features location).

An EKF based SLAM algorithm is based on the formulation of a coupled estimation problem for both the vehicle states, and a set of landmark locations in the evolving map. The two sets of of estimated variables use the Kalman filter data fusion approach to produce improved estimates based on incoming sensory data and models for both the platform as well as the environmental sensors. This approach was quite common in the early years of SLAM research due to its relatively simple formulation, and some available simplified theoretical results, to which the algorithms' results may be compared. Several attempts at EKF SLAM are well documented in the review by Durrant-Whyte and Bailey [1]).

The differences between the research efforts are typically subtle, and tend to focus on methods for reducing computational complexity, and the use of different loop closure strategies [1, 11, 12]. Although EKF algorithms were initially quite popular, several deficiencies are associated with EKF SLAM:

i. Non-linearity of both the sensor model, the platform dynamic model, and other platform sensors such as Inertial Measurement Unit (IMU). These are all linearized in the EKF approach. Although optimal estimates are guaranteed when using the Kalman filter on a linear system, there is no such guarantee when the system and sensor are non-linear.

ii. The computational complexity of the algorithm increases rapidly as $O(n_P^2)$, with $n_P$ being the total number of position estimates and observed landmarks. The EKF is in fact estimating all the platforms positions in combination with all the landmarks at each step. The need for a matrix inversion in the EKF algorithm formulation is the source of the computation complexity. In most scenarios, the number of pose estimates is quite large, and it is desired to have estimates for as many vehicle states as possible. Additionally, the number of landmarks typically increases with the traveled distance.

    Many techniques have been developed to reduce the complexity, but typically at the expense of increased estimation uncertainty [1]. Some techniques were deemed inconsistent, as they achieved a theoretical covariances (a measure of uncertainty) that were lower than the optimal solution's covariance [1, 12]. One typical method for reducing the complexity was sub-mapping, where the estimation was employed on subsets of poses and landmarks, such that the computational workload was feasible. The sub-maps and position subsets were later merged [1] to form a globally consistent map.

iii. The sensor model that is required for the formulation is typically quite complex. In the case of a laser scanner, which is a very common SLAM sensor, modeling the erroneous measurements that may be output from the laser scanner proved to be quite challenging [2]. The probability density function (PDF) for each laser beam is required to contain probabilities for zero-range measurements, maximum range measurements, outlier measurements (see Sub-Section 2.3.1 for a detailed explanation), and typical laser noise. Only the laser noise is easy to model (using a Gaussian PDF).

    In fact, the main problem with PDF-based modeling is that outliers mainly occur on surface discontinuities, and so they may not be modeled based on probability [13]. They are rather an outcome of certain conditions in the environment. Moreover, outliers may take on values only between the two ranges of the surface discontinuity, and this may is typically not considered by the probability distribution [2].

iv. Extracting a small number of key landmarks are essential for the formulation of the estimation problem. An EKF based algorithm may not estimate an entire occupancy grid map, made of a relatively high number of occupied cells. Such a form of environmental map requires the extraction of a considerably smaller number of features. This entails an additional feature extraction algorithm, which typically outputs the larger, more dominant features in the map. Therefore, scenarios that do not contain many large features may not be good candidates for this approach.

**Particle Filter based SLAM**

Like EKF-based algorithms, particle filter SLAM algorithms are also based on Bayesian filters, with a prior belief and a posterior refined estimate for both map features and vehicle states. Solutions to the SLAM problem that use particle filters are based on sequential Monte-Carlo sampling of multi-hypotheses for the map and position estimates.

The advantage of particle filter based algorithms is that each estimate is a result of a set of hypotheses (particles) [1, 8], evaluated using the available platform and sensor mathematical models. These techniques may overcome some of the shortcomings of EKF algorithms [14], and are often compared to EKF based algorithm mainly in terms of robustness (since both are designed to be capable of estimating non-linear problems).

The computational complexity of particle filters depends mainly on the number of particles used (number of hypothesis attempted in a sequential Monte Carlo fashion). Each particle represents a hypothesis for a map and a position estimate, which together with the mathematical models yields a probability estimate for it's correctness (generally a covariance value).

Typically, the number of particles used is adjusted to reflect the available computational resources. However, using too few particles may result in low-quality estimates. This is manifested as an over-confident estimates (artificially low covariance values),

resulting in an inconsistent algorithm (as shown by Bailey, Nieto, and Nebot [14] while analyzing the FastSLAM particle filter based algorithm [8]). Moreover, in some cases, the number of particles is too small to allow sufficient successful loop closure hypotheses [8].

For aerial platforms, since computational resources are further limited (due to limited payload capability), particle filter algorithms may not allow for real time SLAM capability. For this reason, Achtelik et al. [15] and Bachrach et al. [6] demonstrated real-time SLAM using a particle filter based algorithm, with the aid of additional off-board computational capabilities. The relatively high number of particles that were required to tackle the myriad of features in the attempted scenario was the main reason for the on-board computational resources shortcoming.

Generally, particle filter based SLAM algorithms share the same drawbacks as EKF-based algorithms mentioned above. These include implementation difficulties on platforms with complex dynamic models, modeling complexity of sensors, and the need for feature extraction (if a landmark based map is used).

### Dynamic Model Dependency

SLAM methods that make use of the platform's dynamic model require that the platform be mathematically modeled in the form of:

$$\dot{\underline{x}} = f(\underline{x}, t) \tag{1.1}$$

where $\underline{x}$ is the state vector and $t$ is the time. The mathematical model is used to obtain a prior estimate of the platform states based on the input commands.

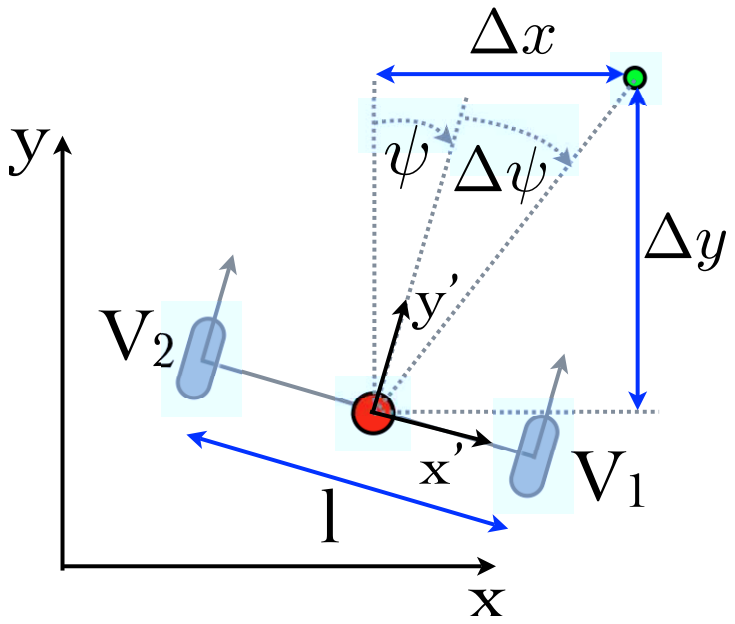



Figure 1.2: Differential drive robot schematics.

Previous works with Bayesian filter have been demonstrated mainly on ground

vehicles with differential drive systems [1, 8, 11, 12, 14, 16], which appear simple to model. The wheels provided odometry information, while other sensors (typically a laser scanner) were used as the environmental sensor, for SLAM purposes.

Figure 1.2 presents a schematics of a differential drive ground robot. A differential drive platform typically has two motor-driven wheels on the sides and a third wheel for stability. Longitudinal motions are controlled by equal rotation of the wheels (using equal RPM commands to both wheels, generating equal values $V_1$ and $V_2$ in Figure 1.2), while changing azimuth is controlled by differentially rotating the two motor-driven wheels (using different RPM commands to the two wheels, generating different values for $V_1$ and $V_2$). The model equations for this differential drive system may be written as follows:

$$\begin{aligned} &\dot{x} = V sin(\psi) \ ; \ \dot{y} = V cos(\psi) \\ &\dot{\psi} = (V_2 - V_1)\tfrac{1}{l} \ ; \ V = \tfrac{1}{2}(V_2 + V_1) \end{aligned} \tag{1.2}$$

where $x$ and $y$ are the global position coordinates, $V_1$ and $V_2$ are the translation velocities of the right and left driven wheels respectively, $\psi$ is the azimuth angle, relative to the inertial frame (zero azimuth is pointing "North"), and $V$ is an average translational velocity. The rotated coordinate system is marked by $(x', y')$ in Figure 1.2. The length $l$ is simply the wheelbase width.

Typically, odometry is obtained from such configurations by means of wheel encoders, combined with the measured wheel radius. Integrating the wheel encoder data yields the platform traveled path. However, there are many error contributions to this estimate, including the fact that the wheels are never perfectly round, slippage that occurs on all platforms (to some degree), surface inclinations, and encoder measurement errors. The goal with odometry is to get a prior estimate for $(\Delta x, \Delta y, \Delta\psi)$, which are the relative motion in $x$, $y$, and $\psi$ respectively.

The success of Bayesian filter based systems increases with the accuracy of the plant model. Therefore, systems that are simpler to model mathematically allow for higher accuracy prior estimates, which in turn contribute greatly to the accuracy of the posterior estimation. Differential drive robots provide the most basic form of a ground platform. Therefore, these platforms became the most popular platform of choice for SLAM experiments. A slightly more complex model may be that of a four wheel platform, *e.g.*, a car, featuring more wheels, as was used in the work by Dissanayake et al. [11]. The accuracy of the prior motion estimates deteriorates as modeling the additional wheels is likely to introduce additional errors. The accuracy of the posterior estimates, generated by the probabilistic filter will be reduced as a direct result of relatively less reliable plant model.

For the same reason, performing SLAM on platforms whose dynamic model is complex may result in a relatively lower accuracy, and is normally not attempted. Such platforms include a walking person, and rotary wing platform configurations. When aerial vehicles are used, the configuration of choice is typically a quad-rotor, which is modeled using four thrust values, controlled by four direct RPM commands.

The well defined rigid body dynamic equations complete the platform's dynamic model. However, modeling a single rotor helicopter, coax, tandem, ducted fan and other configurations is considerably more complex, and typically requires highly non-linear equations, in conjunction with a set of simplifying assumptions. Such models are considered to be of lower fidelity [17].

Although these configurations have several operational advantages over quad-rotor configurations (such as compactness, performance, etc.), SLAM has traditionally been demonstrated on quad-rotor configurations (with the exception of the work by Thrun, Diel, and Hähnel [18] with a single rotor helicopter, and the work by Steder et al. [19], performed on a coaxial helicopter).

Moreover, regarding SLAM using a walking person platform, with the exception of the works by Clemente et al. [20] and Saarinen et al. [21], the author found no other work that demonstrate SLAM on a walking person (or a humanoid robot). Modeling of a walking platform is considered to be much more complex as compared to modeling wheeled ground vehicles.

### Dynamic Model Independency

Several works have presented SLAM algorithms that do not require a dynamic model for the platform carrying the sensors. These algorithms may therefore be applied to essentially any platform with fewer limitations on the platform's dynamics, thereby also removing the dependency on modeling accuracy.

Such algorithms may still have the prior and posterior estimates structure, since the simple differential relations between position, velocity, and acceleration may still be used. Since these relations do not depend on the platform configuration, it is easy to implement them in order to utilize IMU data, and provide improved initial guesses to scan matching techniques [21, 22].

Saarinen et al. [21] presented such an algorithm that was based on scan matching incoming laser scans against the evolving map. They presented results in a typical corridor like environment, while the laser is carried by a walking person traversing the corridors. The path was relatively straight, going to the end of the corridor and coming back to the initial point. However accuracy was relatively poor, and manual map correction had to be introduced.

Steder et al. [19], suggested a visual SLAM algorithm, based on camera imagery. They presented results using a blimp, a helicopter, and a walking person. The walking person platform was used to simulate a data acquisition from a free-flying platform (as a precursor for the helicopter experiments). This practice was also used in this work.

### Scan Matching Based SLAM

Scan matching based SLAM has also received considerable interest in the literature. The main difference is usually replacing or complementing the odometry obtained from the platform's wheel encoders with some form of scan matching of

incoming sensory data. Scan matching methods range from laser scan matching [3] to matching imagery features [23, 24].

When using laser scanners, there is the advantage of using scan matching as opposed to wheel encoders due to their relatively higher accuracy. Matching sequential laser scans (also known as laser odometry) is typically more accurate than odometry integrated from wheel encoders. When using vision sensors for scan matching, the advantage stems from fusing them with IMU data, as in the work by Shen et al. [24]. SLAM using scan matching may be performed in both 2D [21, 25] and 3D [22] environments.

Scan matching gives an estimate for the position from where the latest scan was taken. Based on that position estimate, the latest scan information can be updated into the map, and the process repeats with the next laser scan. This can be performed using a newly acquired laser scan and a scan of the evolving map, made from an approximate position (*e.g.* by means of ray casting), or simply between sequential laser scans. Typically, the latter is likely to result in relatively larger accumulation of errors as demonstrated by Bailey and Nebot [26].

Some previously published SLAM works rely only on scan matching techniques to generate both map and position estimates [3, 4, 21, 27, 28]. SLAM algorithms based on scan matching typically do not require a dynamic model for the platform's motion, and in principle may be applied to any platform regardless of its dynamic behavior. The majority of the work in this area makes use of 2D laser scanners [4, 21, 27, 28]. However, some research efforts use 3D laser scan [18, 22], and some were also extended for 6D SLAM (*e.g.* Nuchter et al. [22]).

### Two-Dimensional SLAM

SLAM methods were initially demonstrated in two dimensions, where planar translation and rotation were the states estimated by the SLAM algorithm [1, 8, 11, 12, 14, 16]. Apart from cameras, most sensors that are used for SLAM operate in two dimensions, providing a scan of the environment (*e.g.* sonar beam arrays and laser scanners). Moreover, the computational complexity of SLAM algorithms increases significantly with the number of degrees of freedom.

Three-dimensional SLAM doubles the estimated platform states, as it not only requires estimating the third dimension ($z$), but also requires estimating pitch and roll angles, which together form a six degree of freedom estimation problem. The majority of the available SLAM research today is carried out in two dimensions, due to the availability 2D laser scanners and the lower computational requirements.

A known problem with 2D SLAM is that of "floor scans". These are instances where the environmental sensor (typically a laser scanner), is perturbed from the desired plane in the scenario, and the resulting scan for that instant contains objects that do not exist in the desired plane, such as the floor or the ceiling. One may consider a scenario where the floor is uneven, a wheeled robot trips over an object, or an aerial robot pitches at a significant angle [29]. Any of these instances may cause the two dimensionality assumption to become invalid, as the sensor will pick up objects that do not lie in the plane of interest.

A very good example for floor scans is discussed in the work by Ho and Newman [30], who present a typical corridor map which includes several such floor scans. In the work by Ho and Newman, the laser scanner is located relatively close to the ground. This is a contributing factor that may increase the occurrence of floor scans (for both laser scanners and sonar beam sensors). The sensor may pick up reflections from the floor before the beams have reached the farther object. This issue becomes even more pronounced when the beams are directed at objects at relatively larger distances.

### Three-Dimensional SLAM

Application of SLAM in three dimensions may be achieved by both 2D and 3D sensors. For example, some research efforts made use of 3D laser scans, that were obtained by mounting a 2D scanner on a tiltable mount (as in the work of Nuchter et al. [22]). The result of each scan is called a "point cloud", and is in fact a set of range values, with each range having a bearing in both pitch, and yaw. Performing scan matching with point cloud is naturally more complex as compared with the more common 2D scan matching techniques [22], mainly because of the increased number of data points. Point cloud data may also be obtained using structured light techniques as in the work by Bachrach using the Microsoft Kinect© sensor [31].

Another approach for 3D SLAM was that of Thrun et al. [18], who performed 3D surface modeling using a 2D laser scanner. However, in their approach, the 2D laser scanner was fixed (rather than mounted on a tiltable surface), providing 2D scans in approximately the same orientation. The scans were then aligned using a probabilistic scan matching approach, which included the use of IMU data. The objects in the scenario were presented using the aligned scans, and surfaces were created using polygons created by sets of nearby points. These results were shown to be useful in representing real scenarios, which could be easily interpreted by a human operator.

### Motivation for Proposed SLAM Algorithm

It is quite clear that for targeted flight mission we require a SLAM algorithm that may not rely on loop closure. The algorithm developed in this work is based on scan matching, independent of the platform's dynamic model, and uses a laser scanner for two-dimensional environments. As will be shown, the current algorithm achieves a significantly lower drift as compared with previously published algorithms, without the use of loop closure algorithms.

The algorithm's independency on a dynamic model contributed greatly to the ease of testing it on multiple platforms. Ground platforms were consistently used as a preliminary step towards testing on the aerial platform. The algorithm development and the experimental process benefitted greatly from reduced complexity, as many aspects were captured and investigated in the ground platforms phase. Ground platforms are considerably less complex as compared with aerial platforms, and therefore provided an easier development process.

### 1.2.2 Environmental Sensors for SLAM

On board sensors are typically used to survey the traversed environment. These sensors provide the algorithm with information about the nearby objects (depending on the available field of view and the nature of the sensor). Several different sensors may be used within SLAM algorithms, including sonar beam arrays, laser scanner, image sensors (cameras), optic flow sensors, and structured light based sensors. Details on each sensor are given below, followed by a comparison table of sensor properties (presented in Table 1.2).

#### Sonar Beam Array

Sonar sensors operate based on time of flight of the sonar signal. These sensors are capable of measuring distance typically with an accuracy of approximately 10% of the measured distance. Sonar sensors were initially used on robots for surveying the surrounding environment, in an attempt to learn the obstacles around the moving robots. Elfes [32] was the first to apply sonar range data for learning occupancy grid maps using a motorized ground robot. He used a circular array of sonar sensors to provide several range measurements that cover a large field of view around the moving platform.

The main disadvantage associated with sonar sensors is their relatively wide beam, causing a difficulty in measuring sharp changes in the environment such as corners. The wide beam essentially gives an average range measurement of all objects that are included in its own field of view. Moreover, the response frequency of sonar beams is slow as it relies on the speed of sound, as opposed to laser range scanners, which use the same time-of-flight method, but are much faster since they depend on the speed of light. For example, a 30 $m$ range will take approximately 0.2 $s$ to measure, while at the same time? a laser scanner produces a complete scan of its entire field of view.

#### Laser Scanner

A laser scanner is a sensor that typically uses either time-of-flight or phase difference measurement of a laser beam to obtain a range measurement. In most sensors, a laser beam is generated, and a rotating mirror at the top of the sensor turns the beam in its plane of rotation, while the sensor records several range measurements around the sensor's field of view. The sensor output is a set of range measurements at specified azimuth angles.

Laser scanners are by far the most popular choice for SLAM algorithms [3–6, 8, 9, 11, 15, 21, 27–29, 33–43]. The main reason is the relatively high accuracy of the range measurements, typically estimated at $0.5\% - 1\%$ of the measured distance (for each beam). As mentioned above, sensor accuracy contributes greatly to the overall SLAM accuracy and thus laser scanners have become very popular for SLAM capable robots. An additional reason is that laser scanners maintain their effectiveness in most cases, regardless of lighting, smoke, and other degraded visual conditions.

The main disadvantages of using laser scanners is their high price and weight as compared with vision sensors. Moreover, the laser scanner output data contains no

information other than the range measurement itself, and in some cases a measure of surface reflectivity. Other sensors provide a richer output that may be used in a myriad of ways to improve the resulting map usability (*e.g.* visual data, color, etc.). Typically, a map created using images may provide a better understanding of the scenario as compared with a map created from point measurements only. Moreover, laser scanners do not operate as well in the presence of daylight. Generally, daytime outdoor operations will result in a significantly reduced detectable range of the scanner (a reduction of approximately 50%).

### Structured Light

Structured light is a method that uses visual algorithms to analyze a light structure, projected on the surface of interest. A light source is used to project a certain pattern onto the surrounding environment, while the refractions from nearby objects are recorded by a camera. The different refractions contain information about the different range to the surrounding obstacles and current algorithms are capable of producing a depth map from the structured light analysis.

Such a a sensor package is the Kinect© sensor by Microsoft. The light source is infra red, and hence invisible to the human eye. Together with the camera, this sensor package provides the both a picture and an associated depth map (also known as "RGB-D" map). This sensor package (although quite heavy and bulky) was used by Backrach et al. [31] to produce three dimensional environment maps and vehicle state estimates, without GPS, using a quad-rotor platform. They demonstrated station-keeping capabilities, as well as mapping a relatively small scenario. The visual maps are useful as the images are fused into the map, providing a useful representation for the traversed environment, that is easily interpreted by a human operator.

### Vision Sensors

Vision sensors have the advantage of being passive, as they do not emit energy, and thus may contribute to stealth operation. In addition, their power consumption is considerably lower than that of laser scanners, and their weight is also significantly lower (approximately two orders of magnitude lower in both cases). These features make them attractive for autonomous operations using MAVs. Vision sensors provide sequence of images (video signal), which are typically a two dimensional image of a three dimensional environment (with a certain field of view).

The main disadvantage of vision sensors is the relatively larger computational resources that are required for image processing, and the lower accuracy of the SLAM estimates [44]. Image processing algorithm such as feature extraction, image matching, and image pre-processing are typically pixel based operations, *i.e.*, depend on the sensor resolution. Typical vision sensors contain approximately one million pixels (or more), which is three orders of magnitude more than the number of laser points in a single laser scan.

in recent years, there has been a growing interest in vision based SLAM, particularly for aerial applications, due to the low-payload and power associated with the

vision sensors. Vision based SLAM was investigated by Steder et al. [19], presenting an algorithm that may be suitable for both monocular and stereo vision systems. Artieda et al. [44] discuss feature extraction and image matching of images acquired from an MAV. They presented a comparison between the SLAM results and a GPS true position data, Achieving an accuracy of approximately 10% of the traveled distance on a relatively small scenario. Weiss, Scaramuzza , and Siegwart [45] show how monocular SLAM estimates (using a single camera) may be used for vehicle stabilization and navigation in GPS-denied environments. Lastly, Shen et al. [24] present an algorithm to fuse vision based information as well as IMU based information, on a quad-rotor for performing SLAM outdoors.

### Optic Flow Sensors

Optic flow is defined a measure of changes that occur in a visual image, caused by the relative motion between the sensor and the scenario. Naturally, the closer the objects in the image are to the sensor - the higher the optical flow will be, as even small movements will create a large change in the viewed scene. This reveals the dependency of optical flow values on the distance from the sensor to the objects themselves, and the translational speed of the vehicle. Therefore, optic flow sensors can only measure distance relative to the platforms speed.

Since SLAM typically requires range estimates to nearby objects, a velocity estimate provided by an additional sensor is required (this estimate may be obtained from an IMU, or from other sensors, as in the work by Kendoul, Fantoni, and Nonami [46]). This allows the estimation of absolute distances.

Optic flow measurements may be extracted from vision sensors, by means of image processing, in which case, the weight and power consumption are quite similar to vision sensors. However the computational resources required for optic flow extraction from images may be considerably lower as compared to image matching.

Generally, the accuracy of distance measurements from optic flow sensor is lower as compared with a laser scanner measurement, particularly due to the dependency on a scaling factor that depends on the platform speed. The estimated accuracy in the work by Kendoul, Fantoni, and Nonami [46] is approximately 10%, which is similar to the accuracy obtained by vision-based SLAM algorithms (see Sub-Section 1.2.2).

### Sensor Comparison

Table 1.1 presents a comparative overview of the available environmental sensors for SLAM, compared relatively to each other with respect to weight, range, power, response time, computational resources required, and accuracy that may be achieved for SLAM. One fundamental difference between the sensors is being active or passive, *i.e.*, does the sensor emits energy (active) or merely absorbs it (passive). The three sensors that are active and emit energy are the sonar, laser scanner, and the structured light, while the camera and optic flow sensor are passive. The sonar and laser sensors provide a range measurement based on emitting energy and sensing the returned energy. The structured light provides a range estimate based on an algorithm that

Table 1.1: SLAM Sensors comparison

| Sensor | Weight | Range | Power | Response Time | Comp. Resources | SLAM Accuracy | Price |
|---|---|---|---|---|---|---|---|
| Sonar | Heavy | Med | Med | Slow | Low | Med | Low |
| Laser | Heavy | High | Med | Very fast | Low | Very high | High |
| Camera | Light | Med | Low | Med | Med | Med | Very Low |
| Optic Flow | Light | Med | Low | High | Med | Low | Low |
| Structured Light | Heavy | Med | Med | Med | High | High | Low |

analyzes the changes in a light pattern that is projected on the surroundings.

Energy emitting sensors are typically associated with a higher power required, relatively to cameras and optic flow sensors, which are passive sensors (*i.e.* no energy is emitted). However, the energy requirements are within the capabilities of current MAVs (typically, a small scale laser scanner requires approximately 10 $W$). In terms of weight, cameras and optic flow sensor are much lighter as compared to the other sensors with typical weights of under 10 $grams$. Weights of laser scanner vary significantly, based mainly on range capability. Those mounted on ground vehicles typically weigh approximately 1 $Kg$, and those mounted on aerial platforms typically weigh approximately 250 $grams$. Each sonar sensor weigh approximately a few grams, and so a sonar sensor array used for SLAM would be in the same weight range of laser scanners.

With respect to range, laser sensors typically have the highest range, which can reach 80 $m$, while the other sensors typically offer usable information for a much shorter range of approximately less than 10 $m$. Laser scanners also have the fastest response time, as they rely on speed of light for measuring a single beam, and therefore are limited by the mirror rotation speed. Laser scanner frequency is approximately 40 $Hz$ and some algorithms are capable of processing the data at this rate as well [29]. On the other hand, while camera frame rate may be in the same range, typical vision algorithms may not be able to analyze the data for each frame. Therefore, camera sensors typically provide a somewhat lower response time [24]. The computational resources associated with vision sensors are typically higher, mainly due to the increased workload for analyzing vision images.

The accuracy achieved by vision sensors is relatively lower as all algorithms require extraction of features from the images in some form, while the laser scanners provide much accurate range measurement that produce higher quality SLAM estimates. Sonar sensors are not as accurate as laser and structures light, and are therefore associated with relatively less accurate SLAM performance. The impor-

tance of accuracy was the main driving factor in choosing the laser scanner as the environmental sensor, since the laser scanner provides the best accuracy. However, with respect to price, the laser scanner is by far the most expensive sensor (with a price tag in the range of thousands of dollars), while the other sensors have a price that is lower by at least an order of magnitude.

### 1.2.3 Platforms Used for SLAM

SLAM has been demonstrated on a myriad of platforms, from wheeled ground robots [4, 5, 8, 9, 11, 27–29, 32–40, 43, 47–51], through a walking person (for personal localization) [20, 21], to aerial vehicles such as helicopters [3, 6, 15, 19, 24, 42, 44, 45], and the relatively slower moving blimp [19].

The ground platforms are by far the most popular as they provide considerably higher payload (and subsequently more computational resources), and thus laser scanners, which are the most accurate environmental sensor, may be easily incorporated. In addition, the added difficulty and risk in aerial experiments also contribute to the scarcity of aerial SLAM work. As mentioned above, ground vehicles are also easier to model as a set of dynamic equations, and so dynamic model-dependent algorithms may be easier to implement on ground robots. The availability of wheel odometry which is considered to be the cheapest in terms of computational requirements, is also a contributing factor to the popularity of ground vehicles.

Another advantage is the predominantly two-dimensional motion that wheeled ground vehicles have (assuming a level environment). Most wheeled robots have relatively small pitch and roll angles while traversing a typical office like environment. Naturally, these vehicles may trip over objects such as wires on the ground, which may interfere with the usefulness of the data set [29], but the effect will be local and relatively small as compared with the entire traversed path.

A walking person may be used as the platform that carries the environmental sensor, and the complete system may then be used for self localization of ground forces [21]. As mentioned above, a walking person may be used as a pre-validation platform for a free flying MAV, as was done in the work by Steder et al. [19].

A considerably smaller number of applications have used aerial platforms with the majority of the work carried out using quad-rotor micro aerial vehicles [3, 6, 15, 24, 42, 45], which as discussed above, are relatively less complex to model. Research using other configuration was found to be quite scarce. A single rotor configuration was utilized by Thrun, Diel, and Hähnel [41] and Artieda et al. [44], and a coaxial configuration was utilized by Steder et al. [19]. All the methods that were employed on these platforms were independent of the platform's dynamic model.

Using a single main rotor, tail rotor configuration is considered to be less expensive, and may be more compact, for indoor operations in constrained environment. These vehicles are readily available commercially, with a wide range of available payloads. Since the algorithm proposed in this work is independent of the platform's dynamic model, a single rotor configuration was the platform of choice for this work.

### 1.2.4 Mapping Forms

There are several methods for storing environmental maps for SLAM algorithms. The occupancy grid approach [32] is by far the most common way to represent an environment. The environment is represented by a square grid, and each square represents a piece of the environment. Cells receive a value above zero if the SLAM algorithm determines them to be "occupied". The cells with a value of zero are considered as the "free space", through which the robot can move (additional details may be found in Section 2.1).

The other fairly common mapping form is a map based on landmarks' location [1, 37], which is typically used in most EKF based SLAM algorithms. This map is essentially a collection of major landmark locations in a global reference frame. This approach keeps the number of tracked objects to a minimum as each landmark is represented using a single location (which helps maintain a low number of estimated states for the EKF formulation).

Nguyen, Harati, and Siegwart [4] used a different form of mapping, which works on scenarios with primarily flat surfaces. Each surface that was extracted from sensory inputs was recorded in the map as a planar surface with its own dimensions. This mapping form naturally suits only for certain environments, where the above assumption holds.

#### Motivation for Choosing Occupancy Grid

The current algorithm uses an occupancy grid for representing the estimated map. Occupancy grid allows the bets representation for the environment as it does not assume anything about the typical shapes of the objects in the scenario. It can therefore be used in any scenario. The one parameter that controls the fidelity of the environmental representation is the occupancy grid resolution. An occupancy grid does not require a dedicated algorithm for feature extraction, which greatly simplifies implementation and dependency on tunable parameters.

Moreover, the update stage of new information into the occupancy grid, as well as other computational algorithms employed on the occupancy grid, all maintain a constant computational complexity (unlike the feature based maps whose complexity grows with the number of features in the scenario). The occupancy grid provides a very clear map, as the results of this research will show, allowing for maintaing highly accurate maps of the traversed scenarios. The occupancy grid is later used for the path planning task with relative ease, as the free cells are used to calculate obstacle free paths to the goal.

### 1.2.5 Loop Closure Algorithms

SLAM algorithms in general accumulate errors in both position and mapping estimates during the estimation process. Therefore, a platform that returns to a previously mapped area from a different direction may face map discontinuity issues. Local maps created for the same area viewed from two different places may not align seamlessly.

This, in fact, may be leveraged for reducing the accumulated error by establishing a relation of translation and rotation between two observations of the same scene, taken from different platform positions. The established relations allow for a map optimization process that attempts to minimize the accumulated map error, using the constraints between the pose pairs.

One common solution is to employ a loop closure algorithm that continuously operates in the background of the SLAM process, and searches for opportunities to match map frame pairs of the same scenario. The complete set of pair-wise relations is then brought together as a pose-graph, with the inter-frame relations acting as constraints. The pose graph with the constraints may then be optimized using different optimization techniques. A map correction process is then employed to repair and piece together the misaligned local maps into one continuous global map.

Representative examples for loop closure detection and map optimization algorithms are included in the works by Bosse et al. [52], Ho and Newman [30], Stachniss et al. [9], Konolige [36], and Olson, Leonard, and Teller [53]. The above examples all make use of Bayesian filter based SLAM methods, and the loop closure algorithm is probability based as well, and requires relatively complex implementations in some cases [52].

Loop closure algorithms may fail when repeated structures appear throughout the mapped area, as those may produce false loop closure detections. It is important to note that once the map has been corrected using a false loop closure - it cannot be recovered. The now optimized pose estimates are used to re-align the input sensory data, and the map is then updated with the assumption that the new optimized set is now optimal.

Moreover, in mapping missions, a scenario may need to be traversed more than once in order to guarantee a high probability loop closure candidate. Naturally, this is not an optimal solution as resources may be limited (*e.g.* battery life, mission time, operational stealth, etc.). An example for a scenario where a loop had to be traversed twice in order to achieve a successful optimized map using a loop closure algorithm (see the work of Stachniss et al. [9]).

### 1.2.6 Static and Dynamic Environments

A static environment is a fairly common assumption made in robotics. It greatly simplifies the associated algorithms, since it is expected that subsequent environmental sensor readings of the same objects will generate similar results (excluding sensors' noise). However in dynamic environments, parts of the scenario may change position and orientation between subsequent environmental readings.

A dynamic environment requires additional pre-processing for the environmental input data, in attempt to determine which parts of the data represent a static environment, and therefore may be updated into the evolving map, and which parts represent moving objects and should therefor be excluded from the map update step. Dynamic parts of the scenario should also be excluded from participating in any scan matching step.

The algorithm for identifying moving objects from an environmental reading depends on the nature of the environmental sensor, as well as on the expected moving object, as those algorithms are tuned to identify objects of certain kind (*e.g.* people, doors, outdoor vegetation, etc.).

For mapping missions, Walcott-Bryant et al. [51] have suggested that for mapping an environment with moving objects, a multi-pass method may be used, such that the environment is essentially mapped more that once, with the robot traversing a similar path every time. Information about changes in the mapping of the same scenario at each traversed path is stored and used to update the map further, while tagging objects as "static" or "dynamic".

However, the above solution may not be used for targeted flight missions, as the scenario is only traversed once. In cases where multi-readings of the same scenario are not available, objects in the map need to be deleted. Deletion may be based on the number of environmental scans that have included these objects. This way, a dynamic object may be present in several scans, but each time at a different place, while a static object will maintain its position in all scans. The dynamic object will subsequently be disregarded from the environmental scan, and be deleted from the evolving map. Such an approach is implemented in the current work (Sub-Section 3.3.3).

### 1.2.7 Scan Matching Techniques

The process of scan-matching between two environment scans results in the appropriate roto-translation values required to match one scan on top of the other. Many types of scan matching algorithms exist [26, 33, 54–58]. However, since each algorithm has strengths and weaknesses and this work relies solely on scan matching for both position and map generation, the most promising was found to be the use of brute force for minimizing the scan matching cost function.

This thesis limits the discussion to local scan matching which is performed between two subsequent scans of the environment (unlike global scan matching which matches between a laser scan and a complete map [59]). The scan of the environment may be carried out using laser scanners, sonar range sensors, or cameras. In this work, we focus on representing the environment using a 2D laser scan. The result of a single scan is a set of range measurements given over a set of azimuthal angles in the scanner plane. For this type of data, three categories of scan matching techniques exist:

1. Feature-to-Feature [55]: this technique extract features (such as lines, corners, etc.) from both the current and the reference scans. The features are then matched using some algorithm to get the translation and rotation between the two scans. Correspondence between the features needs to be established correctly in order to assure accuracy, speed, and convergence. These techniques are relatively fast as they reduce the amount of data from the number of lasers points, to a much smaller number of extracted features.

2. Point-to-Feature [54, 58, 60]: in this technique, features are extracted only from the reference scan, while the points from the current scan are associated with those features to establish the correct solution. Here too, establishing the

right correspondence between points and features is key to the success of this technique.

3. Point-to-Point [29, 55]: this technique is considered to be more robust as the scanned environment does not have to be comprised of geometric features. The technique makes use of the point sets themselves in the computation of the scan matching solution. However, the number of points dominates the complexity of the algorithm. The correspondence between the matched points can be based on inter-point distance such as in the Iterative Closest Point (ICP) algorithm [61], range from the origin as in Iterative Matching Range to Point (IMRP) [55], or other variants.

Since these techniques have strengths and weaknesses - a number of works attempted a synthesis of two techniques that would complement each other and yield an overall better, more optimal solution. Examples include the Iterative Dual Correspondence (IDC) algorithm which combines ICP and IMRP [55], and combining IDC with a line-based algorithm [60] to form a hybrid algorithm that works well in either a polygonal or non-polygonal environments [54]. These attempts also aim to take advantage of the computational complexity advantages of each algorithm. There are several common ingredients between these methods:

i. Start with an initial guess.

ii. Project the Current Scan onto the Reference Scan's coordinate system.

iii. Eliminate points that are either measurement outliers or occluded.

iv. Define correspondence between the points or features in the two scans.

v. Calculate a cost function to evaluate the match.

vi. Employ a minimization algorithm to minimize the cost.

### Motivation for Developing the Current Scan Matching Algorithm

The current scan matching algorithm is a Point-to-Point algorithm. The main advantage of this approach, as mentioned above, is its robustness as it is not limited to environments with certain features and assumptions. Moreover, there is no need for a feature extraction algorithm. This approach has no assumptions about the shape of objects in the environment.

Another relatively simple but important feature that was developed is a new outlier filter, which excludes wrongful laser range measurements that occur when the laser scans surfaces with large discontinuities (see Sub-Section 2.3.1 for more information and schematics). A scan matching algorithm that uses these points may yield poor solutions as these points do not represent real objects and thus should be discarded [62]. Previous approaches for eliminating outliers use a simple thresholding approach (also known as Median Filtering [13, 29]), which may fail in some cases as

the set threshold may not fit all possible cases. However, the new filter is based on a more robust approach, which is guaranteed to eliminate all outliers in a laser scan.

The current approach outputs improved scan matching solution as compared to previously published algorithm (see Chapter 5), and is a key for achieving highly accurate SLAM estimates. Since the SLAM methodology in this work is primarily based on the scan matching algorithm, it was important to develop a new and highly accurate algorithm that was not previously available.

### 1.2.8 Path Planning

A path planning algorithm outputs a set of segments (or waypoints) that comprise a path from a start position to a goal position. Path planning algorithm may have the following properties:

i. Heuristic methods - based on some guiding function.

ii. Physical analogy based - uses a physical representation to generate a path.

Heuristic guided planners include graph search methods (*e.g.* Dijkstra's Algorithm, A*, D*, D*Lite, etc.), which are employed on a grid of nodes that span the operational environment. Of these methods, the A* method has the advantage of being fairly easy to code. However, it is considered to be slower as compared with D*, and D*Lite [63]. The latter two algorithms have improved computational speed, at the expense of a more complex programing effort of the search algorithm. The path planners in these cases may not be decoupled from the localization, mapping and other blocks of the mission planning algorithm. Note that the path planner, as defined in this work, is also responsible for the obstacle avoidance task, and therefore the generated path must not pass through any known obstacles.

Probability based search may be utilized by randomly sampling the environment to get a searchable graph. Two popular examples for probabilistic based algorithms include the Probabilistic Road Map (PRM) [64], and the Rapidly exploring Random Tree (RRT) [65]. The PRM has the advantage of being suitable for large scenarios with relatively lower computational resources. However, path solutions using PRM may result in jagged edges, which require smoothing. In addition, graph nodes may be found in close proximity to obstacles, which may reduce mission safety. The RRT algorithm is built on clusters of search trees, which continue exploring the environment through the tree branches, as long as an obstacle is not encountered. The main disadvantage of this approach is the lack of a goal-directed search approach, which may result in exploring unnecessary areas of the scenario.

Other methods mimic a physical system, for example the potential field approach. In this approach, each detected obstacle receives a potential value, while the goal receives a value with a negative sign. The potentials are used to derive a force field, while the platform is considered as a particle with a certain virtual mass that moves through the force field. The resultant virtual forces yield acceleration values, which may be integrated to obtain the trajectory. While the potential field approach requires relatively low computational resources, and paths may be generated fairly fast, the

problem of reaching a local potential minima is quite common. In such cases the resultant force on the virtual particle becomes zero, and the particle therefore does not move, although the goal has not been reached [66].

Another example is genetic algorithms, based on biology. These algorithm develop a path based on evolution of an initial pool of guessed paths, which evolve through generations of mutations [67]. These methods typically require more computational resources and several initial guesses to begin the process. This might prohibit their implementation on aerial platforms.

The path planning algorithm used in this work is a relatively simple A* algorithm [68]. The algorithm is based on graph search theory, and is designed to find the lowest cost path from an initial node in the graph to a final node. The algorithm is guaranteed to find a path if one exists. For the implementation of the algorithm, the occupancy grid is used, and a coarse version is created. Every occupied cell-center is considered as a node in the graph with possible connections to nearby cells. The A* algorithm operates at each step of the SLAM algorithm to find an obstacle free path from the current location to the goal position.

The path planning algorithm is used in this work for demonstration purposes of targeted flight capability on MAVs using the SLAM estimates. The A* algorithm may be substituted with other algorithms as there is no tight coupling between the SLAM algorithm and the planned path. The SLAM algorithm produces accurate estimates regardless of the traversed path (unlike the work by Roy, Gordon, and Thrun [69], where a controller was designed to plan a path that would support better SLAM performance).

The A* algorithm is a global path planning algorithm, which plans a complete path assuming the environment is known. In the current work, the algorithm is invoked at each step, and uses the currently available map. The algorithm therefore changes its planned path based on new information that is updated into the evolving map at each step. This, in fact, introduce dynamic updates in the path planning task, and local path segments are updated to assure an obstacle free path.

### 1.2.9 Accuracy Survey

In this section, previously published papers are arranged in a chronological list, highlighting important research characteristics. The papers contain results for performing SLAM on different platforms and algorithms. The main purpose of the list presented in Table 1.2 is to highlight the scarcity of attention given to the resulting accuracy of the estimated maps and platform states.

The previously published papers are reviewed with respect to reported accuracy, loop closure algorithm usage, platforms used, and number of scenarios attempted. Because accuracy was not quantified in many papers, the table lists estimates made by the author, based on availability of data (not all papers provided sufficient data for evaluation).

Some important highlights are brought below. It is important to note that this survey only included reports of experimental map accuracy. Simulation results and

covariance estimates were not included as those were typically found to significantly differ from actual accuracy measurements [10].

The definitions pertaining to symbols used in the Table 1.2 are as follows:

i. "Small Scale" – relates to mapping areas smaller than 10 $m$ length of the largest dimension, which typically allow the use of a motion capture system.

ii. Accuracy – measured by percentage of the traveled distance, and so it represents the accumulated drift.

iii. "$\sim$" – means the accuracy was not explicitly stated in the original work, but instead was approximated by the current author using the published figures.

Table 1.2: Previous work reports on accuracy

| Paper Title | Year | Accuracy | | Loop Closure | Platform | | | Scenarios |
|---|---|---|---|---|---|---|---|---|
| | | Small Scale | Large Scale | | Wheeled | Human | Aerial | |
| Gutmann, J. S. and Konolige, K., "Incremental Mapping of Large Cyclic Environments" [35] | 1999 | – | – | ✓ | ✓ | | | 4 |
| Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M. "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem" [11] | 2001 | – | $\sim$ 0.1% | ✓ | ✓ | | | 1 |
| Thrun, S. "A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots" [47] | 2001 | – | – | ✓ | ✓ | | | 4 |
| Hähnel, D. H., Burgard, W., Fox, D., and Thrun, S., "An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from raw Laser Range Measurements" [8] | 2003 | – | – | ✓ | ✓ | | | 2 |
| Brenneke, C., and Wagner, B., "A Scan Based Navigation System For Autonomous Operation of Mobile Robots in Man-Made Environments" [27] | 2003 | – | $\sim$ 1.5% | ✓ | ✓ | | | 1 |
| Konolige, K., "Large Scale Map Making" [36] | 2004 | – | – | ✓ | ✓ | | | 1 |

| Paper Title | Year | Accuracy | | Loop Closure | Platform | | | Scenarios |
|---|---|---|---|---|---|---|---|---|
| | | Small Scale | Large Scale | | Wheeled | Human | Aerial | |
| Stachniss, C., Grisetti, G., Hähnel, D., and Burgard, W., "Improved Rao-Blackwellized mapping by adaptive sampling and active loop-closure" [9] | 2004 | – | – | ✓ | ✓ | | | 2 |
| Borges, G. A., and Aldon, M. J., "Line Extraction in 2D Range Images for Mobile Robotics" [34] | 2004 | – | – | ✓ | ✓ | | | 2 |
| Saarinen, J., Mazl, R., Kulich, M., Suomela, J., Preucil, L., and Halme, A., "Methods for Personal Localisation and Mapping" [21] | 2004 | – | $\sim$ 2% | – | ✓ | | | 2 |
| Nieto, J., Bailey, T., and Nebot, E., "Scan-SLAM: Combining EKF-SLAM and Scan Correlation" [37] | 2005 | – | – | ✓ | ✓ | | | 1 |
| Sim, R. and Roy, N., "Global A-Optimal Robot Exploration in SLAM" [38] | 2005 | – | – | ✓ | ✓ | | | 0 |
| Brunskill, E. and Roy, N., "SLAM using Incremental Probabilistic PCA and Dimensionality Reduction" [48] | 2005 | – | – | ✓ | ✓ | | | 1 |
| Nieto, J., Bailey, T. and Nebot, E., "Recursive scan-matching SLAM" [39] | 2006 | – | $\sim$ 1% | ✓ | ✓ | | | 2 |
| Olson, E., "Fast Iterative Alignment of Pose Graph with Poor Initial Estimates" [53] | 2006 | – | – | ✓ | ✓ | | | 1 |
| Newman, P., Cole, D., and Ho, K., "Outdoor SLAM using Visual Appearance and Laser Ranging" [40] | 2006 | – | – | ✓ | ✓ | | | 1 |
| Thrun, S., Diel, M., and Hähnel, D., "Scan Alignment and 3-D Surface Modeling with a Helicopter Platform" [41] | 2006 | – | – | ✓ | | | ✓ | 3 |
| Nguyen, V., Harati, A., and Siegwart, R., "A Lightweight SLAM Algorithm using Orthogonal Planes for Indoor Mobile Robotics" [4] | 2007 | – | $\sim$ 1.5% | | ✓ | | | 1 |
| Diosi, A. and Kleeman, L., "Fast Laser Scan Matching Using Polar Coordinates" [29] | 2007 | – | – | ✓ | ✓ | | | 1 |

| Paper Title | Year | Accuracy | | Loop Closure | Platform | | | Scenarios |
|---|---|---|---|---|---|---|---|---|
| | | Small Scale | Large Scale | | Wheeled | Human | Aerial | |
| Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., and Tardós, J. D., "Mapping Large Loops with a Single Hand-Held Camera" [20] | 2007 | – | – | ✓ | | ✓ | | 2 |
| He, R., Prentice, S., and Roy, N., "Planning in Information Space for a Quadrotor Helicopter in a GPS-Denied Environment" [42] | 2008 | $\sim$ 10% | – | ✓ | | | ✓ | 1 |
| Kollar, T. and Roy, N., "Trajectory Optimization using Reinforcement Learning for Map Exploration" [70] | 2008 | – | $\sim$ 3.5% | ✓ | ✓ | | | 1 |
| Kollar, T. and Roy, N., "Efficient optimization of information-theoretic exploration in SLAM" [43] | 2008 | – | – | ✓ | ✓ | | | 2 |
| Steder, B., Grisetti, G., Stachniss, C., and Burgard, W., "Visual SLAM for Flying Vehicles" [19] | 2008 | – | $\leq$ 8% | ✓ | | ✓ | ✓ | 4 |
| Nùñez, P., Vàzquez-Martìn, R., Bandera, A., and Sandoval, F., "Fast laser scan matching approach based on adaptive curvature estimation for mobile robots" [33] | 2009 | – | $\sim$ 1% | | ✓ | | | 2 |
| Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N., "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-Denied Environments" [15] | 2009 | 5% | – | ✓ | | | ✓ | 3 |
| Olson, E., "Real Time Correlative Scan Matching" [71] | 2009 | – | – | ✓ | ✓ | | | 1 |
| Ji, X., Zhang, H., Hai, D., and Zheng, Z., "An Incremental SLAM Algorithm with Inter-calibration between State Estimation and Data Association" [49] | 2009 | – | – | ? | ✓ | | | 1 |
| Garcia-Favrot, O., Parent, M., "Laser Scanner Based SLAM in Real Road and Traffic Environment" [5] | 2009 | – | $\sim$ 0.2% | ? | ✓ | | | 1 |

| Paper Title | Year | Accuracy | | Loop Closure | Platform | | | Scenarios |
|---|---|---|---|---|---|---|---|---|
| | | Small Scale | Large Scale | | Wheeled | Human | Aerial | |
| Artieda, J., Sebastian, J. M., Campoy, P., Correa, J. F., Mondragòn, I. F., Martìnez, C., and Olivares, M., "Visual 3-D SLAM from UAVs" [44] | 2009 | – | $\sim 10\%$ | ✓ | | | ✓ | 1 |
| Park, S. and Park, S., "Spectral Scan Matching and Its Application to Global Localization for Mobile Robots" [28] | 2010 | – | $\leq 5\%$ | ✓ | ✓ | | | 1 |
| Bachrach, A., Prentice, S., He, R., and Roy, N., "RANGE-Robust Autonomous Navigation in GPS Denied Environments" [6] | 2011 | – | $\leq 1.9\%$ | ✓ | | | ✓ | 3 |
| Shen, S., Michael, N., and Kumar, V., "Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV" [3] | 2011 | $\sim 1.5\%$ | – | ✓ | | | ✓ | 2 |
| Weiss, S., Scaramuzza, D., and Siegwart, R., "Monocular-SLAMBased Navigation for Autonomous Micro Helicopters in GPS-Denied Environments" [45] | 2011 | – | – | ✓ | | | ✓ | 1 |
| Valencia, R., Vals Mirò, J., Dissanayake, G., and Andrade-Cetto, J., "Active Pose SLAM" [50] | 2012 | – | – | ✓ | ✓ | | | 1 |
| Walcott-Bryant, A., Kaess, M., Johannsson, H., and Leonard, J., "Dynamic Pose Graph SLAM: Long-term Mapping in Low Dynamic Environments" [51] | 2012 | $\sim 0.5\%$ | – | ✓ | ✓ | | | 2 |
| Shen, S., Mulgaonkar, Y., Michael, N., and Kumar, V., "Vision-Based State Estimation for Autonomous Rotorcraft MAVs in Complex Environments" [24] | 2013 | $\sim 10\%$ | – | – | | | ✓ | 1 |

The main conclusions that can be drawn from Table 1.2 are as follows:

1. The majority of the works have used wheeled (ground) platforms. The typical ground robot is a differential drive system, which can be easily modeled using the rotation of the two wheels, measured by wheel encoders. Research using aerial platforms was only performed by a few research groups: MIT [6, 15, 42], University of Freiburg [19], Universidad Politécnica de Madrid [44], University

of Pennsylvania [3, 24], and ETH Zurich [45] (out of 36 reviewed papers). The main reason is the modeling complexity associated with an aerial platform (in addition to the relative experimental difficulty with flying vehicles).

2. The majority of the works do not report quantitative accuracy, neither for a small scale nor for a large scale scenario. In most of the cases, this author had to estimate the accuracy based on available figures (accuracy estimates are marked by $\sim$).

3. The majority of the researchers rely on a loop closure algorithm to perform mapping missions. The loop closure algorithm helps optimize the resulting map as explained in Sub-Section 1.2.5. The mapping tasks performed in these researches, typically provide multiple opportunities to re-visit previously explored areas (and thereby exploit a loop closure algorithm). These algorithms may fail in targeted flight missions, with no or very few loop closure opportunities.

The above list and its conclusions clearly show the little attention given to experimental SLAM accuracy. It also reveals a dependence on loop closure algorithms for obtaining consistent maps, and a general shortage of work on SLAM for aerial platforms. The need for SLAM accuracy without relying on loop closure algorithms is a requirement for targeted flight path planning.

## 1.3 Thesis Outline

Following this introduction chapter, the thesis continues with Chapter 2, which describes several existing robotics tools that are later utilized to form the PB-PSM algorithm, and the coupled SLAM-Path Planning algorithm for targeted flight. Additional algorithms that were used for benchmarking purposes are also presented. Chapter 3 presents the new PB-PSM algorithm developed in this thesis, with emphasis on the innovations and refinements that contribute to the accuracy. Sufficient details for a complete reproduction of both the algorithm and the results, are also included. Analysis of some of the limitations and assumption is described using analytical case studies. A description of several new accuracy metrics used for benchmarking is also included.

The experimental setup is then described in Chapter 4, including the laser sensor, the platforms used, and an overview of all the scenarios that were attempted in the experiments. Chapter 5 presents the results, starting with the accuracy of this algorithm using the custom accuracy metrics, including a detailed evaluation of the individual contribution of each algorithmic component to the overall accuracy. A comparison against previously published algorithms is also presented. The limitations of the algorithm are then discussed, with specific examples for possible failure modes, results obtained using the various platforms, and scenarios (both indoor and outdoor), also using previously published datasets, further validating the algorithm. Several examples of the application of the SLAM algorithm for path planning and obstacle avoidance are provided. Chapter 6 presents summary conclusions for the entire thesis, and also outlines the future work.

## 1.4 Summary of Contributions

This work focused on developing and testing a refined SLAM algorithm. The main contributions of thesis are listed below.

### 1.4.1 Major Contributions

1. Developed a novel scan matching algorithm, referred to as Perimeter Based Polar Scan Matching (PB-PSM). Three key elements distinguish PB-PSM from previously published scan matching algorithms:

   - Use of exhaustive search (using adaptive search grids) to minimize the scan matching cost function (significantly reduces the problem with local minima).
   - Use of a perimeter matching term that maximizes overlap between the two matched scans.
   - Employing relatively tight convergence criterions on both the azimuth and planar cost minimization processes.

2. Coupled the PB-PSM scan matching algorithm, with virtual scans, and map update algorithm, to form a refined SLAM algorithm. The algorithm was extensively tested on 5 indoor scenarios, 8 outdoor scenarios, 3 different platforms, 3 different laser scanners, and both in-house and previously published datasets.

   The new algorithm is shown to produce highly accurate maps and position estimates, and produces significantly lower drift as compared with previously published algorithms, quantified at approximately 0.1% of the distance traveled. This is the lowest SLAM-related drift reported, to the best knowledge of the author.

3. Coupled a SLAM algorithm with an A* path planner to achieve a navigation system in GPS denied, previously unexplored environments. This technique was experimentally tested on multiple platforms, including a helicopter.

4. Identified the importance of SLAM accuracy **without** the aid of loop closure algorithms (and subsequently without map optimization). This is important for targeted flight and scenarios that may not contain loop closure opportunities. In these cases, the success of the mission depends on the accuracy of the map and position estimates (and associated low drift), provided by the SLAM algorithm.

5. Developed a novel metric for quantifying the accuracy of an occupancy grid map that is the result of a SLAM algorithm. The discussion about accuracy quantification is generally missing from most previous work on SLAM. The new metric gives an average distance of a occupancy grid cell from a measured true map. While other proposed metrics may only be employed on simulated environments, or be affected by human intervention [72], the proposed metric

is fairly easy to calculate, is employed on true, measured maps, and involves no human intervention.

### 1.4.2 Additional Contributions

1. Developing a new and intuitive mixed pixel filter. Mixed pixels are laser measurements that form on the edge of surface discontinuities, where the range reading receives an intermediate value between the two surfaces' ranges. The new filter avoids the typical thresholding approach which is not robust to all mixed pixels. Instead, it uses a shallow angle definition which is guaranteed to remove all mixed pixels, and the tunable parameter is a much more intuitive value of a shallow angle.

2. The PB-PSM algorithm's accuracy is benchmarked against previously published algorithms using individual scenes, as well as series of scan matching operations. The latter is considered to be more challenging since the scan matching error is cumulative by nature. Therefore an accumulated error over a series of scan matching operations is considered to be a significantly more challenging test. PB-PSM is found to be significantly more accurate as compared to previously published algorithms.

3. Quantifying the contribution of using virtual scans to SLAM accuracy, as compared with laser odometry (sequential laser scan matching). The use of virtual scans is shown to have a significant effect, which on average improves the map quality by approximately 50%, using the newly developed metric.

4. The use of artificial obstacles for testing cluttered environment. Artificial obstacles which are considered by the path planning are introduced, but do not physically exist. Hence this technique allows for safer testing in highly cluttered environments, as the probability for obstacle collision is minimized.

5. An analysis of the effect of laser scanner velocity and scan frequency on the scan error. Analytical result developed for the error size as a function of the affecting parameters.

# Chapter 2

# Existing Algorithms

This chapter describes the existing tools that were used in this work. The concept of an occupancy grid is presented, followed by the virtual scan, which operates on the occupancy grid map to provide a scan of the evolving map. The scan matching tool is presented in general, followed by two previously published scan matching methods which were used for benchmarking against the novel PB-PSM algorithm developed later in this work. Lastly, the A* path planning algorithm implementation, that was coupled with the SLAM algorithm is presented.

## 2.1 Occupancy Grid

An Occupancy Grid (OG) is a way of representing the surrounding environment using a grid of cells: rectangles (typically squares) in two dimensions (pixels), and prisms in three dimensions (voxels) [32, 73, 74]. Each cell stores information about the possibility of that area in the map being occupied by a physical object. This data comes from the laser scan data (or other environmental sensors). The OG is also the virtual map that is built by the autonomous agent. It represents the environment that was explored thus far. The OG evolves with every update of sensory information, taken by the robotic platform.

At every step of the complete SLAM algorithm, after estimating the laser scan's position and orientation, the laser range measurements may be used to update their associated cells occupancy, as shown in Figure 2.1. Each laser point that is recorded in a particular cell increases the probability of the area represented by that cell being occupied.

An example of an occupancy grid representation of a corner scenario is presented in Figure 2.2. This occupancy grid was received after several successful steps of mapping the featured corner. In this case, the resolution of the occupancy grid is 10 $mm$ by 10 $mm$, and so each cell in the OG matrix represents an area of 1 $cm^2$.

The occupancy grid may be used to derive a probability value for cell occupancy. In the current work, this may be achieved by normalizing the occupancy grid by its maximum occupancy. This would results in occupancy values between 0 and 1. These values may be considered by position estimation algorithms in a number of ways:

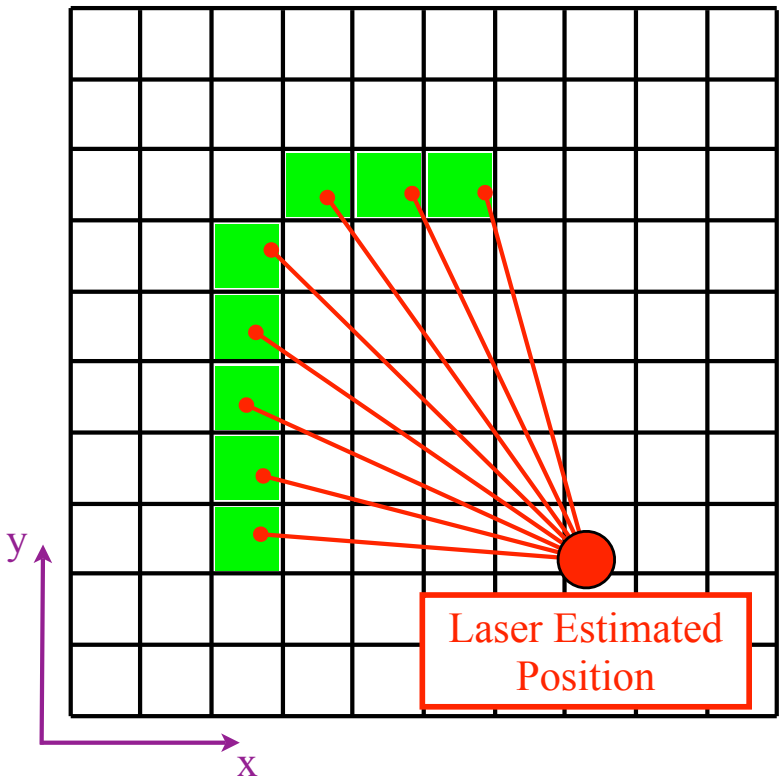



Figure 2.1: Occupancy grid schematics.

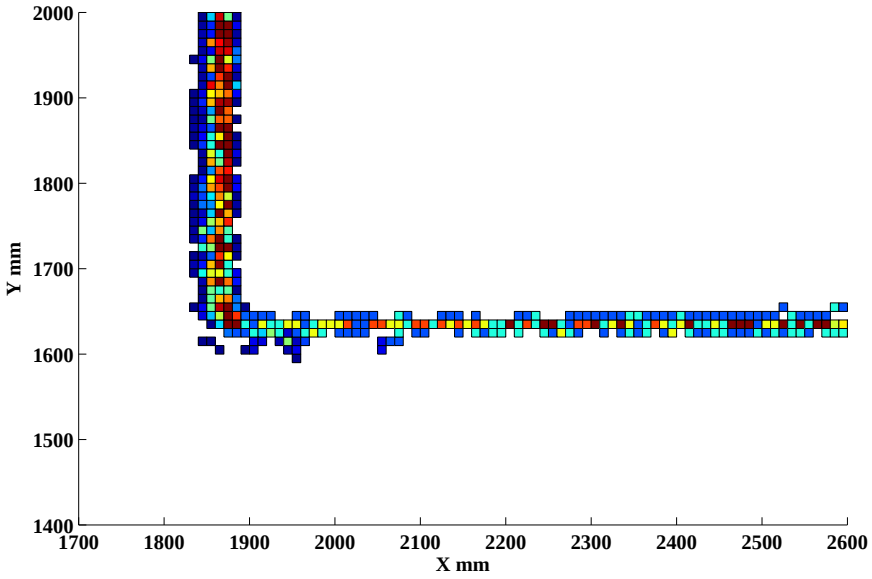



Figure 2.2: Example of an occupancy grid.

1. as weights for scan matching (attempting to better match cells with higher occupancy).

2. as a measurement model for the virtual scan (see Section 2.2).

3. as range measurements probability in probabilistic based approaches (such as the EKF and particle filters, mentioned above).

### 2.1.1 Occupancy Grid Map Update

Updating the OG map with the newly acquired laser scan data requires the current position and azimuth estimates to be accurate. Therefore, it is performed only if the scan matching process results in a well-minimized function, otherwise the current laser scan is not inserted into the occupancy grid. In the rare case of scan matching failure, one can either hold the platform's position update, or establish an estimate based on extrapolation using previous steps (although it would have to be marked and treated as a "non-valid" position estimate). In such cases, the algorithm relies on the success of the next scan matching process to correct the un-supported estimate.

The corresponding cell index for the $k^{th}$ updated laser point is found by:

$$i_k = \lfloor x_k/d \rfloor + 1 \; ; \; j_k = \lfloor y_k/d \rfloor + 1 \tag{2.1}$$

where $i_k, j_k$ are the indices of the cell to be updated, $x_k, y_k$ are the cartesian coordinates of the $k^{th}$ laser point, and $d$ is the occupancy grid resolution (the cell's edge length, in $mm$). The definition of Eq. (2.1) refers to indices that start at 1.

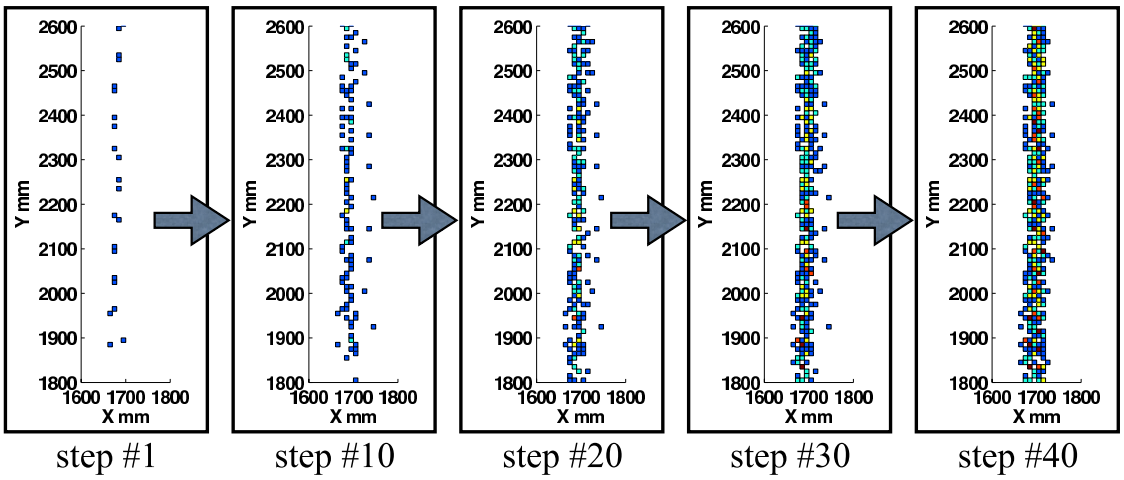



Figure 2.3: Example of a map evolution for a vertical wall (no sensor model considered). The last figure (Step #40, on the right), shows the result after applying the isolated point filter.

The evolution of the OG map for a representative obstacle is shown in Figure 2.3. In this case, each occupied cell represents a laser point that was recorded in that cell. Figure 2.3 shows how the object evolves over 40 steps, and also shows the result

after employing the isolated point filter, which removes several isolated points (see Sub-Section 3.3.3 for a detailed explanation).

### 2.1.2 Occupancy Grid Scalability

An occupancy grid is a relatively demanding map form, in terms of required memory. Other mapping form such as obstacle-based maps [1, 37] only keep position information for the discovered objects. Their required memory only grows when introducing newly discovered objects. However, the occupancy grid which represents each small part of the environment with a cell in the memory, requires memory allocation for each cell, whether occupied or free. This way, the entire operational space is represented in the memory, regardless of the number of occupied cells.

An occupancy grid may be represented using a sparse matrix approach, which only stores a list of the occupied cells and their cell coordinates. However, using this approach, every query for cell occupancy requires a full search for that cell in the list. In the case of a virtual scan, such cell queries appear for every cell crossed by every beam, and so a typical *single* virtual ray requires a very high number of queries, on the order of 1,000. Therefore this approach is not suitable for the algorithm proposed in this work.

In this work, the largest area that was attempted was 60 $m$ by 40 $m$. Using an occupancy grid resolution of $d = 10$ $mm$, this area was represented by 24,000,000 cells. Although the stored numbers may be integers, even assuming a double precision matrix, this array requires approximately 48 $MB$, which is well within currently available on-board memory.

In terms of required memory, the occupancy grid matrix is by far the largest variable in the program. It is approximately three orders of magnitude larger than the next sized variable. Therefore, its size forms the upper bound on the maximum area size possible. As of today, a typical on-board memory size may be considered to be of the order of 1 $GB$. The number of cells that may stored is approximately 500,000,000. Since each cell represents an area of 1 $cm^2$, the total number of cells may represent up to 50,000 $m^2$ (approximately 12 Acres). Therefore, the current approach may be employed on the majority of indoor and outdoor scenarios, considering typical MAV operational environments.

## 2.2 Virtual Scan

The Virtual Scan is a scan of the virtual world, achieved by performing a "simulated laser scan" produced by a series of ray casting operations, searching for occupied cells in the occupancy grid. The virtual scan is executed from the previous estimated vehicle position and azimuth (or an estimated current position and azimuth), and produces a set of range values to the nearest occupied cells. It is essentially a snapshot of the virtual environment, which serves as the reference scan from the map that was built thus far. The virtual scan is later matched against a new laser scan

obtained from the current, true laser scanner position to get a corrected estimate of the current position.

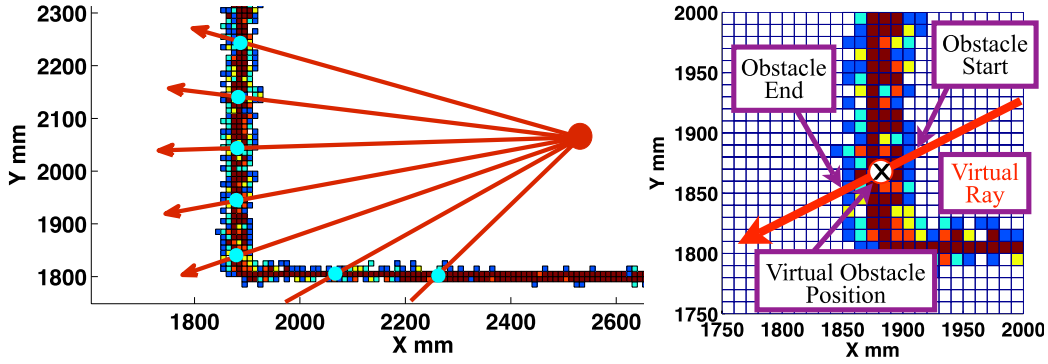



Figure 2.4: Virtual scan Illustration of a corner.

Figure 2.4 depicts an example of this process where on its left is an occupancy grid representation of a corner (virtual map) with the virtual scan origin marked with a red circle, and virtual rays are marked with red arrows. The occupancy grid is color coded with the cells' hit count values (higher values – warmer colors). On the right is a close up view of the same corner presenting just a single virtual ray for clarity. In the general case, the ray encounters a "thick" wall made of several clustered occupied cells created from several previous scans. A ray-casting operation logs the start and end of the cell cluster along the ray and then defines the wall location as weighted average location using all the logged cells along that ray and their respectable weight. These three markings are marked as "Obstacle Start", "Obstacle End", and "Virtual Wall Position" in Figure 2.4. The virtual scan result is, in fact, an average of all the previous laser readings taken up to that point in time. This means that when scan matching is later carried out, the newly acquired laser scan is matched against an average of all previous sensory data.

### 2.2.1 Obstacle Rendering Algorithm

Because of the laser sensor's inherent noise, and the resolution of the OG, any obstacle will eventually be represented by a collection of occupancy grid cells. Once the ray casting operation registers the beginning of an obstacle ("obstacle start"), it records all the cells' values until it reached the end of the wall ("obstacle end"). The cell that best represents the obstacle location can be found using Eq. (2.2):

$$I = \left\lfloor \frac{\sum_{j=1}^{N_{cells}} jW_j}{\sum_{j=1}^{N_{cells}} W_j} \right\rfloor \tag{2.2}$$

where $I$ is the index of the final representative cell, $N_{cells}$ is the number of occupied cells found by the current ray, $W_j$ is the occupancy of each cell. The result is the

index of the cell that best represents the obstacle encountered by the current ray. The actual virtual radius is determined using the mid point between the entry and exit points of the current casted ray to that representative cell. The values that $W_j$ can take depend on the way information is stored in the occupancy grid. If the occupancy grid stores laser hits only, then $W_j$ represents the number of laser points recorded in that cell. If the occupancy grid is used for storing probabilities for cell occupancy [32], then $W_j$ may receive value between 0 and 1.

The process is controlled by two, user-input parameters - the maximum number of captured occupied cells, and the definition for an "obstacle end". The first limitation is intended for scanning thick objects, where the ray will pick up numerous cells. The result would be a virtual scan point that does not accurately represent the scanned obstacle due to the occupancy grid's resolution. Hence, the number of occupied cells along a single ray, denoted as $N_{thickness}$ is limited (in this work, a value of $N_{thickness} = 30$ was used throughout).

The second parameter defines when an inspected occupied cell is considered to be the end of an obstacle. Due to the sparse nature of the OG, picking up a non-occupied cell, does not necessarily constitute that the obstacle has ended. Isolated empty cells can form within a populated cell area, mainly due to laser noise, but also due to occupancy grid resolution. Hence, the ray casting continues until at least $N_{empty}$ unoccupied cells are found (typically $N_{empty} = 10$).

The resolution of the occupancy grid may, of course, be adjusted. High resolution allows representation of finer detail in the environment, and decreases the error in the virtual scan since the representation of the environment is more accurate. However, it comes at the expense of memory requirements, and more importantly - computational requirements of the virtual scan. A single ray casting operation has complexity of O(R/d) where $R$ is the range of the casted ray, and $d$ is the resolution of the OG, defined as the cell's edge length, in $mm$. So higher resolution involves a linear increase in computational time (only for the virtual scan step).

The virtual scan described here is accelerated using a coarse occupancy grid which contains exactly the same information as the fine occupancy grid, but with a coarser resolution (typically with $d = 1000\ mm$). The same ray casting operation is employed on the coarse occupancy grid, until an occupied cell is found, at which point the search process transitions to the fine occupancy grid for the remainder of the ray casting operation. The coarse occupancy grid also undergoes the same updates as the fine occupancy grid.

### 2.2.2 Spacial Accuracy

Since the occupancy grid has a finite resolution, objects may be misrepresented when using finite resolution occupied cells. This problem may be more pronounced as resolution is lowered. In the extreme case, an object's true position can be close to a cell boundary, and therefore, most of the cell area does not actually represent that obstacle. As detailed above, the virtual scan will represent that cell using the mid-point of the virtual ray's entry and exit points to and from that cell, respectively.

Figure 2.5 shows two possible misrepresentations of an object as a result of the

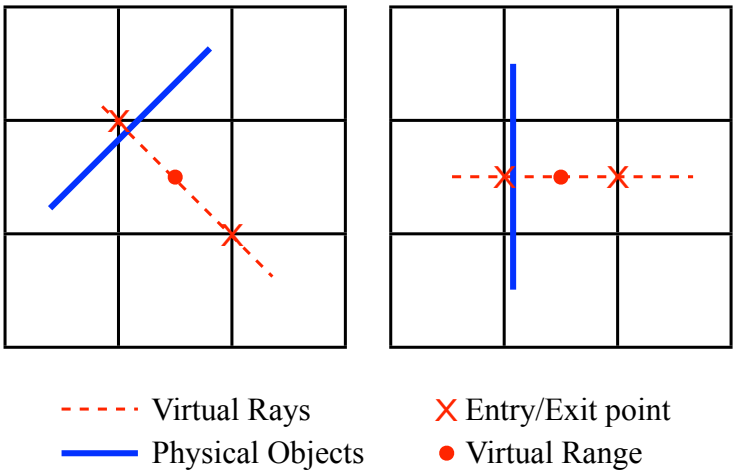



Figure 2.5: Misrepresentation of true objects when using an occupancy grid.

finite OG resolution. The scenario on the left causes a range error of approximately $\frac{\sqrt{2}}{2}d$, or half the diagonal length of a cell, while the scenario on the right results in a smaller range error of approximately $0.5d$. However, for an object that spans several cells in length, the scenario on the left would misrepresent the range to the object every other cell, while the range error in the scenario on the right would be the same for all cells. The effect of OG resolution on the performance of the algorithm is investigated, and presented in Sub-Section 5.3.4.

It is noteworthy that the method developed herein represents the occupied cells in a better way, as compared with Bresenham's algorithm [75], which has two main deficiencies:

i. It does not start or end at the defined point coordinates, but rather the cell center where those coordinates lie. As discussed above this may introduce a significant shifting error.

ii. It only considers one crossed cell per column of the OG, and thus may occasionally miss cells that are actually crossed by the beam (and may be occupied).

The second deficiency may be clearly seen when examining a schematic example of Bresenham's algorithm performance, presented later in Figure 2.14 (see Sub-Section 2.6.4) . Both cell $(2, 2)$, and $(4, 3)$ are not checked by the algorithm. For the virtual scan to be more accurate, if either of these cells is occupied - they should be the beginning of the object representation ("obstacle start").

However, Bresenham's algorithm is faster than the proposed ray casting method and so a certain speedup may be achieved at the expense of a slightly lower overall accuracy. Bresenham's algorithm is described in Algorithm 4, in Sub-Section 2.6.4 (used in the path planning block for way point definition).

## 2.3 Scan Matching

The process of scan-matching between two environment scans results in the appropriate roto-translation values (in this case - in two dimensions) required to match one scan on top of the other. Many types of scan matching algorithms exist [26, 29, 33, 54–58]. The differences between the algorithms typically lies in the data association techniques (see Sub-Section 1.2.7), or in the cost minimization technique (*i.e.*, using a gradient based method, or a form of brute force search).

As described in Sub-Section 1.2.7, the scan matching process starts with an initial guess, projects the Current scan onto the Reference scan coordinate system, eliminates undesired points, performs data association between the two scans, and minimizes a cost function that is built using the associated data. This section describes the common point filters used in this work, and the data association method used in both the previously published PSM, and the novel PB-PSM developed in this work.

### 2.3.1 Point Filtering

The acquired laser points are passed through a series of filters designed to leave only valid laser points for the scan matching process. The filters are described below in detail. Note: all of the filters' parameters depends on the laser scanner capabilities (range, angular resolution, accuracy, and sensitivity), and are updated based on the employed laser sensor.

#### Minimum/Maximum Filter

If the scanner does not pick up any reading, it outputs either zero or it's own maximum detection range. Therefore, the minimum/maximum range filter was designed to exclude validity of points with range values below a minimum threshold of 400 $mm$ and above a maximum threshold of 29000 $mm$ (these values are attributed to the UTM-30LX Hokuyo sensor [76]). In addition - those points are not considered in subsequent filters. These points will not be updated into the map in the map update stage (Sub-Section 2.1.1).

#### Occlusion Filter

After the laser scan is transformed (using Eq. (3.1) below), some scanned objects may become occluded by others. Naturally, occluded points should be excluded from the scan matching process as they cannot contribute to the matching process [29, 58, 62]. A simple example is a scan that's performed right when the laser has passed around a corner, as in Figure 2.6, where the blue circles are the originally acquired laser points (FOV marked in dashed black lines). The points that are picked up by the laser after passing the corner cannot be viewed when the scan is transformed to the new origin (marked as 'x').

Transforming the scan to a any point that's located before the corner would result in an occluded wall, since the same wall could not have been observed from that point.

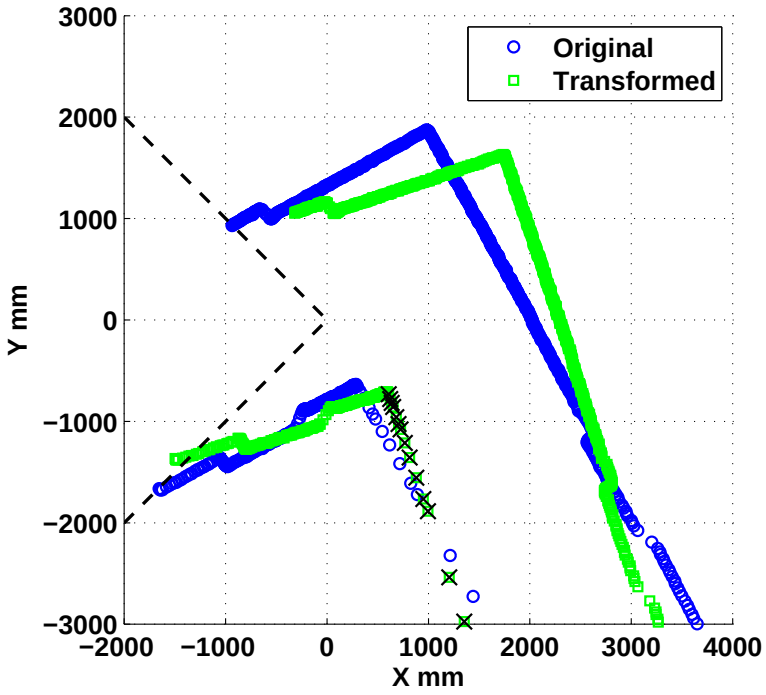



Figure 2.6: Example scan matching with occluded points.

The geometric result of this transformation is a change in the order of the laser scan angles, as depicted in Figure 2.7.

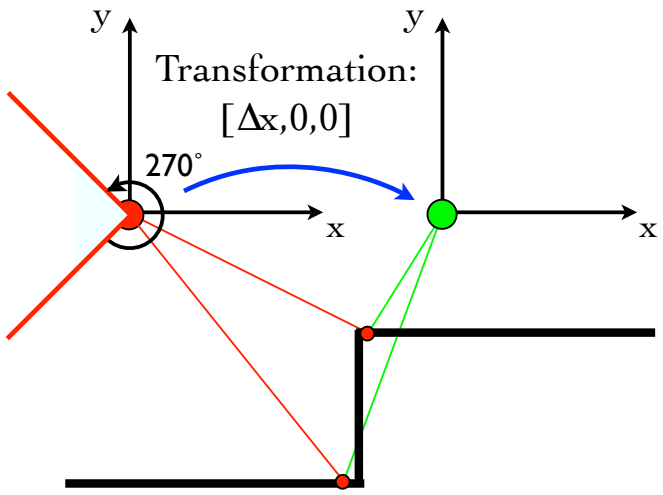



Figure 2.7: Schematics of occlusion detection by angle order switching.

The filter is therefore built to identify such switch backs in the previously ordered laser points' angles, and remove only the ones that are occluded. Note that in some cases due to the inherent laser sensor noise, some points are wrongfully filtered out.

However, the number of wrongfully eliminated points is minuscule (on the order of 10 points), and so it does not degrade the overall performance in any way. The Occlusion filter is employed only on points that are still "active", and as such, it must follow the min/max filter immediately so it would not be affected by the operation of other filters.

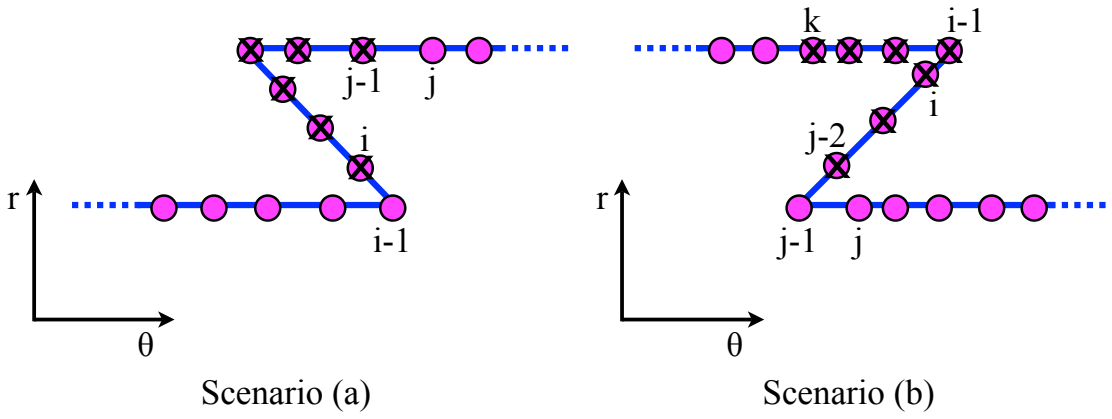



Figure 2.8: The two possible occlusion scenarios.

The filter is in the form of a simple sweep algorithm, starting from the smallest angle (from left to right). Figure 2.8, shows two scenarios of a series of laser measurements (circles, connected by a line for clarity), with their respective angles $\theta$ on the '$x$' axis, and range on the '$y$' axis (after a roto-translation). Each new laser measurement is tested for an angle switch back event, and when a switchback is discovered the algorithm acts according to one of the scenarios depicted in Figure 2.8, eliminating only the points that are in fact occluded (marked with an 'x'). The complete algorithm for the occlusion filter is described in Algorithm 1.

The filter was designed to have $O(n_C)$ complexity, with $n_C$ being the number of points in the laser scan. Since the laser measurements are supplied from the scanner sorted by their respective angle - there's no need to employ a sorting algorithm (which avoids a complexity of $O(n_C log(n_C))$). Every point is examined for angle switchback, and at most examined again for being part of an occluded range. Therefore, the complexity is bounded by $O(2n_C)$ which represents a linear complexity of $O(n_C)$.

An example of the elimination process is given in Figure 2.9, where the radii values are plotted against their respective angle, after transformation. The laser points that cannot be observed are marked with an 'x'.

### Outlier Filter

A rather common and inherent laser scan error occurs when the scenario contains surface discontinuities. This is quite typical in any environment that contains edges.

**Algorithm 1** Occlusion Filter algorithm

$i \leftarrow 1$ ▷ Primary index
$\theta_{curr} \leftarrow \theta_1$ ▷ Initialization with first point's angle
**while** $i < n_C$ **do**
  **if** $\theta_{i+1} < \theta_{curr}$ **then**
    **if** $r_{L_{i+1}} > r_{L_i}$ **then** ▷ Scenario (a) - increasing range values, subsequent points are occluded
      **while** $\theta_{i+1} < \theta_{curr}$ & $i < n_C$ **do**
        Eliminate occluded point $i+1$
        $i \leftarrow i+1$
      **end while**
    **else** ▷ Scenario (b) - decreasing range values, previous points are occluded, up to the next switching point's angle
      **while** $\theta_{i+1} < \theta_{curr}$ & $i < n_C$ **do** ▷ Scan forward until the difference in angles becomes positive again
        Eliminate occluded point $i$
        $i \leftarrow i+1$
      **end while**
      $\theta_{curr} \leftarrow \theta_i$ ▷ Record the angle of the first point that becomes visible
      $j \leftarrow i-1$
      **while** $\theta_j > \theta_{curr}$ & $j > 1$ **do** ▷ Scan Backwards and eliminate all points with angles greater than that of the first visible point
        Eliminate occluded point $j$
        $j \leftarrow j-1$
      **end while**
    **end if**
  **end if**
**end while**

Wherever a range discontinuity occurs, the laser beam may capture two surfaces with a depth difference, producing an outlier measurement [13, 29, 54, 58] that can take on any range value between the two different surfaces' ranges. An example of a laser scan that produced several outliers is presented in Figure 2.10(a). Outlier points are also known as mixed pixels [29, 58].

Mixed pixels typically lie in the free space, and thus do not represent any real object. Since the nature of outliers changes from one scan to another (as they are considered to be an anomaly of the laser scanner), one may not use the outliers points when performing scan matching, as those would undoubtedly introduce errors into the process. Therefore, outlier points should be ignored altogether.

Conventional approaches to eliminating such mixed pixels include Median Filters [13, 29], and RANSAC-based methods [15, 34, 62]. Median Filters involve evaluating the distance between the mixed pixel and the preceding point. The suspected mixed pixels would be accepted if the range is smaller than a set threshold, typically set after some experiments with the laser scanner. An improvement for the above

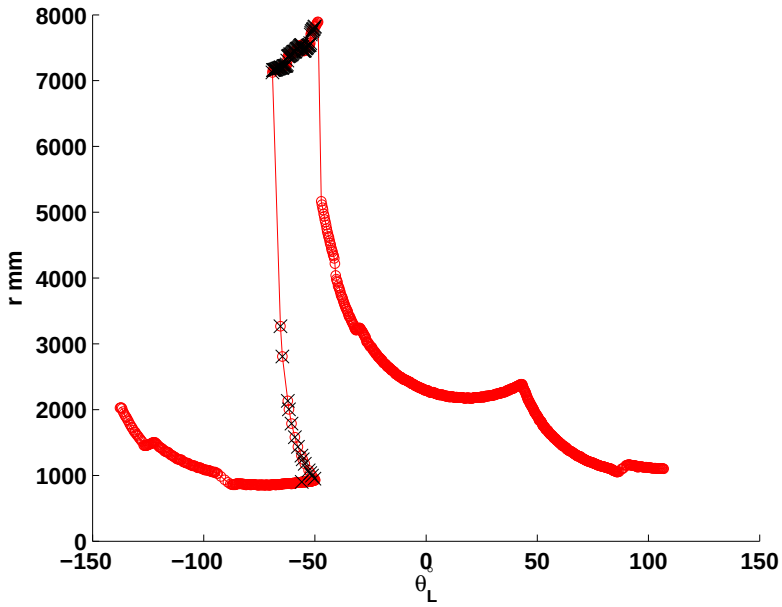
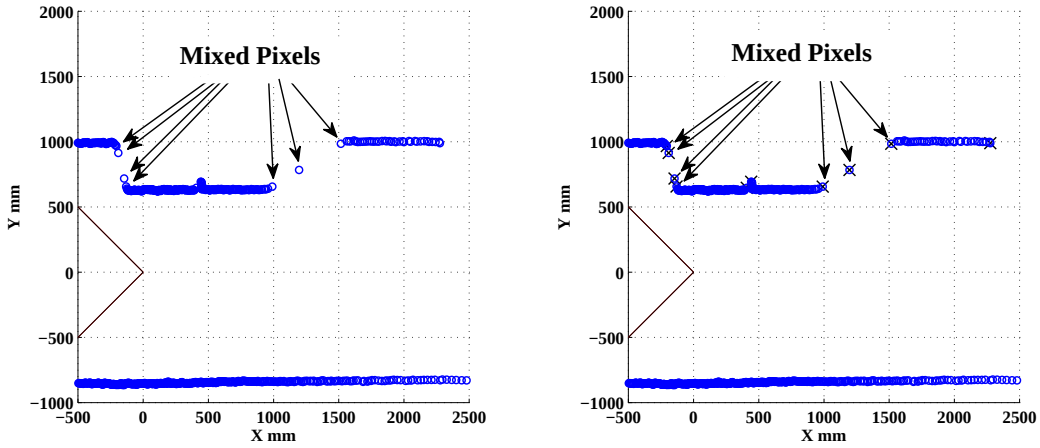



Figure 2.9: Example of a case that introduces object occlusion.




(a) Before outlier filter

(b) After outlier filter

Figure 2.10: A laser scan (circles) before and after applying the outlier filter. All the points that may be mixed pixels are eliminated (crossed out). The two lines show the field of view boundaries (scanning counter clockwise).

approach could be to consider the range from the mixed pixel to both the preceding point and to the succeeding point. However, for any set threshold, some mixed pixels may avoid being tagged as such.

The RANSAC-based approaches attempt to fit a a straight line to the nearby area of the investigated point. If a line cannot be fitted to the investigated point along with its nearby neighbors - the point is tagged as an outlier. This method

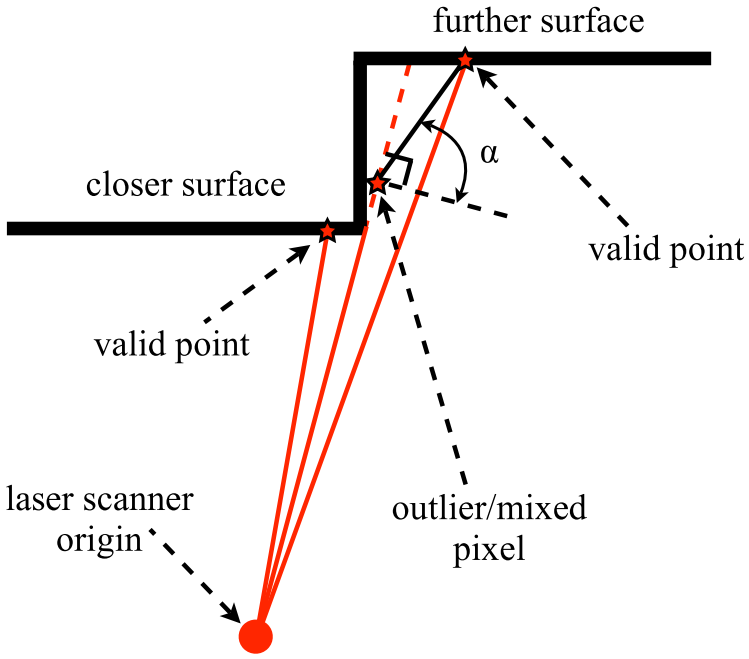



Figure 2.11: Example of identifying an outlier(mixed pixel) laser measurement.

typically has two parameters: the definition of a neighboring area (how many points to consider on both sides of the investigated point), and a threshold for a good line fit. Observing Figure 2.10, it can easily be claimed that using both methods, every set threshold will only eliminate some of the mixed pixels, but not all of them.

This work proposes a new concept for the identification of outlier points. The proposed outlier filter relies on examining the angle between successive points. As mixed pixels lie on range discontinuities, the angle between a mixed pixel and it's preceding point would typically be relatively shallow. Thus, outlier candidates may be identified using the relative angle between the beam and the line connecting the candidate point to the preceding point.

Figure 2.11 presents a schematic description of how outlier points are generated. Laser measurements are marked as red stars, laser scanner rays are marked as red lines, and the laser origin is marked as a red circle. The filter checks for the angle between a line connecting every two neighboring laser points and a line perpendicular to the laser beam angle, denoted as $\alpha$ in Figure 2.11. If $\alpha$ is close to 90° – the point is discarded. In some cases, points can be wrongfully eliminated (due to laser noise creating the same conditions between two neighboring laser points). However, the vast amount of laser points in each scan provides sufficient information for the algorithm to perform well. A value of 85° is used throughout this work. The complete algorithm for the outlier filter is described in Algorithm 2.

Figure 2.10(b) presents the result after applying the outlier filter described above. The eliminated points are crossed out with a blue 'x' mark. As discussed, an adverse effect of using this filter is that several points may be wrongfully considered as outliers, mainly due to sparse laser noise. Some examples may be seen on the right side of the

**Algorithm 2** Outlier Filter Algorithm

$i \leftarrow 1$ ▷ Primary index
$\theta_{curr} \leftarrow \theta_1$ ▷ Initialization with first point's angle
**for** $i = 1 \rightarrow n_L - 1$ **do** ▷ Inspecting all the laser scan points
  $r_1 \leftarrow r_i$
  $r_2 \leftarrow r_{i+1}$
  $h \leftarrow r_1 sin(\Delta_\theta)$
  $e = |r_1 cos(\Delta_\theta) - r2|$
  $\alpha \leftarrow atan(e/h)$
  **if** $\alpha > \alpha_S$ **then** discard both $i^{th}$ and $i + 1^{th}$ points
  **end if**
**end for**

upper wall in Figure 2.10(b).

A positive byproduct of using the angle-based outlier filter is that it also eliminates laser points where the angle of the beam with the measured surface is relatively shallow. Measuring walls at a shallow angle is likely to produce larger errors [77] and thus such points are less desired when performing scan matching.

#### Field Of View Filter

The virtual scan does not necessarily cover 360°. Hence, in cases where the FOV of the laser sensor differs from that of the virtual scan, with the addition of rotation and translation, it is necessary to identify the right overlap between the two scans so that the scan matching process will only be carried out on potential matching points. The FOV filter is designed to exclude points that lie outside the overlapping area, on both sides of both scans. However, note that Current scan points that are eliminated by this filter are still being updated into the OG (Sub-Section 2.1.1), after the scan matching process is completed. Discard any points in both the Current and the Reference scans which fulfills one of the conditions given in Eq. (2.3):

$$\theta_R > \theta_{C_{max}} \; ; \; \theta_R < \theta_{C_{min}} \; ; \; \theta_C > \theta_{R_{max}} \; ; \; \theta_C < \theta_{R_{min}} \tag{2.3}$$

These points become outside the overlapped field of view after roto-translation. This practice is similar to the one used in other works [78]. Note that while $\theta_{R_{min}}$ and $\theta_{R_{max}}$ are constant, $\theta_{C_{min}}$ and $\theta_{C_{max}}$ change for every different rotation value.

### 2.3.2 Linear Complexity Data Association

The data association between the Reference and the Current scans is the process of determining which point in the Current scan corresponds to which point (or points) in the Reference scan (or vice versa). The two sets of range measurements are considered abstract, as they only give range and bearing, without any additional information about which object is actually described by the scan points.

The formulation of the cost function is comprised of contributions made by corresponding points from both scans. Ideally, if correspondence was known, the cost function would accurately described the quality of the matching between the two scans, and every scan matching operation could have been solved in as little as a single iteration [78]. However, since the correspondence between the two point sets is generally unknown, it must be established by correlating properties between the points in both sets. If the corresponding is not accurate, the cost function may not be well-minimized, which would result in additional iterations.

As reviewed in Sub-Section 1.2.7, correspondence may be determined using closest point (as in ICP), using the "matching range" rule (the point is associated to the point with the most similar range), "matching bearing" rule (as in PSM), or based on point to feature distance (if features were extracted from one of the scans, see Sub-Section 1.2.7). In this work, the data correspondence described by Diosi and Kleeman [29] was utilized, with some minor refinements. Finding point correspondence between the two scans is described as pseudo-code in Algorithm 3:

**Algorithm 3** Point correspondence search

$k \leftarrow b$ ▷ b: first Current Scan point within the FOV of the Reference Scan
**for** $i = a \rightarrow n_R - 1$ **do** ▷ a: first Reference Scan point within the FOV of the Current Scan
  **if** Reference Scan point $i$ is active **then**
    **while** $k < n_C$ **do** ▷ k: running index on the Current Scan
      **if** (k) and (k+1) are both valid laser points **then**
        **if** $\theta_{C_k} \geq \theta_{R_i} \geq \theta_{C_{k-1}}$ **then**
          Establish correspondence between point $i$ in the Reference Scan and points [$k-1$,$k$], in the Current Scan
        **else**
          $k \leftarrow k + 1$
        **end if**
      **else**
        $k \leftarrow k + 1$
      **end if**
    **end while**
  **end if**
**end for**

The correspondence search is claimed to have a linear complexity of $O(n_R)$ (assuming $n_R \geq n_C$), as each point in the Reference Scan is matched only once to a single pair of points from the Current scan, and the $k$-index does not consider any point in the Current Scan more than once.

The linear complexity of the point correspondence is a key feature that allows the use of an adaptive direct search for the function minimization. (which is a form of exhaustive search). The number of cost function evaluations required by adaptive direct search is significantly larger than the one required by gradient search methods. Therefore, using adaptive direct search with a higher complexity cost function may be

quite challenging. A representative comparison of the required computational time for the same case, using this cost function and the ICP cost function with a complexity of $O(n^2)$, revealed a significant advantage, in favor of the linear correspondence search approach.

#### Range Interpolation

The cost contribution for each valid point in the reference scan is calculated using the range difference between the reference scan point's range, and a linearly interpolated range value for the two neighboring points in the currant scan, found using the algorithm in Sub-Section 2.3.2. The interpolated range value is given by:

$$r''_C = r'_{C_{left}} + \frac{r'_{C_{right}} - r'_{C_{left}}}{\theta'_{C_{right}} - \theta'_{C_{right}}} \left(\theta_R - \theta_{C_{left}}\right) \tag{2.4}$$

where the $()'$ system represents the current scan after roto-translation, $r'_{C_{right}}$, and $r'_{C_{left}}$ are the ranges to the neighboring points on the right and left, respectively, and $\theta_{C_{right}}$ and $\theta_{C_{left}}$ are the beam angles for the neighboring points on the right and left, respectively.

## 2.4 PSM Scan Matching Algorithm

In Polar Scan Matching (PSM), developed by Diosi [78], the translation is estimated using linear regression theory, while the rotation is estimated using exhaustive search (although with a different cost function, as compared with PB-PSM). Estimating rotation is performed assuming the pose is known, while estimating translation is carried out assuming the orientation is known. Since neither is generally true, the complete algorithm is carried out in an iterative fashion.

### 2.4.1 Translation Estimation

Estimating the translation between two scans in PSM is carried out while assuming no rotation difference exists between the two scans. Therefore, it is also assumed that the data association between the two scans is correct (this assumption is inherently wrong when the rotation has not been estimated correctly, as is the case of intermediate iterations, before convergence has been achieved). The error to be minimized is defined based on the squared difference between the ranges of the Reference scan and interpolated ranges in the Current scan, obtained in a similar way as in PB-PSM, using Eq. (2.4). The goal in translation estimation is to find a pair of $(x_c, y_c)$ that would minimize the weighted least square error between the Current and the Reference scans, given by Eq. (2.5):

$$\varepsilon_{LS} = \sum w_i (r_{R_i} - r''_{C_i})^2 \tag{2.5}$$

where $r_{R_i}$ is the $i^{th}$ range from the Reference scan, $r''_{C_i}$ is the interpolated range from the associated points in the Current scan, and $w_i$ is a weight assigned to each matched pair, used to reduce the effect of bad matches. Minimizing the weighted sum may be achieved by exploiting linear regression. The range differences may be modeled as follows:

$$(r''_C - r_R) = H \begin{bmatrix} \Delta x_c \\ \Delta y_c \end{bmatrix} + v \tag{2.6}$$

where $\Delta x_c$ and $\Delta y_c$ are the translation values in the $x$ and $y$ directions, respectively, $v$ is a random noise vector, and the Jacobian matrix $H$ is given by Eq. (2.7):

$$H = \begin{bmatrix} \frac{\partial r''_{C_1}}{\partial x_c} & \frac{\partial r''_{C_1}}{\partial y_c} \\ \frac{\partial r''_{C_2}}{\partial x_c} & \frac{\partial r''_{C_2}}{\partial y_c} \\ \vdots & \vdots \end{bmatrix} \tag{2.7}$$

The position change vector $[\Delta x_c \Delta y_c]^T$ may then be found using Eq. (2.8):

$$\begin{bmatrix} \Delta x_c \\ \Delta y_c \end{bmatrix} = \left(H^T W H\right)^{-1} H^T W (r_R - r''_C) \tag{2.8}$$

where $W$ is a diagonal matrix that contains the weights of each matched pair or ranges. The above described algorithm results in a linear complexity of $O(n)$, similar to the complexity of the PB-PSM algorithm (see Sub-Section 3.3.4). This is mainly due to the linear complexity of the data association search in polar coordinates (see Sub-Section 2.3.2).

### PSM Weight Determination

The weights are distributed according to the Eq. (2.9), following Diosi [78]

$$w_i = 1 - \frac{\Delta r_i^m}{\Delta r_i^m + c^m} \tag{2.9}$$

where $\Delta r$ is the range difference between the matched range pair, and $c$ is a constant. Eq. (2.9) will have a weight of 1 for a zero range difference (perfect match), and the weight will be reduced as the difference grows. The parameter $c$ determines where the function changes from 0 to 1, and $m$ determines how quickly that change happens.

In the PSM algorithm, a large range difference above a set threshold is not considered, and is therefore given a weight of zero. This is identical to the $T_E$ threshold employed in PB-PSM (see Sub-Section 3.1.1). In fact, the same threshold value of $T_E = 1000\ mm$ was used in this work as well as in the work by Diosi [78].

### 2.4.2 Rotation Estimation

Assuming the pose is known, a change in rotation performed in polar coordinates is represented by a shift of the range values to the left or right. Therefore, if the Current and Reference scans contain information from the same objects, the correct orientation may be found by rotating the Current scan until the range values agree with those of the Reference scan. In PSM, an exhaustive search with a window of $\pm 20°$, with incremental steps of $1°$ was implemented. For each rotation candidate, the average absolute range residual is calculated as follows:

$$\begin{aligned} &\sum_{i=1}^{n_C} |r''_{C_i} - r_{R_j}| \\ &j = i + \tfrac{\Delta\psi}{\Delta\theta} \end{aligned} \tag{2.10}$$

Note that Diosi and Kleeman picked the search window size and incremental steps $\Delta\theta$ such that $\frac{\Delta\psi}{\Delta\theta}$ inherently yields an integer. However, in the general case, the index $j$ may be defined as follows:

$$j = i + \left\lfloor \frac{\Delta\psi}{\Delta\theta} \right\rfloor \tag{2.11}$$

Interpolation may also be employed to increase the accuracy of $r_{R_j}$, between two neighboring values. However, the added computational time may not be desirable.

Additionally, the three points with the lowest residuals are fitted with a parabola for a further refinement of the solution. The parabola's minimum point is used as the refined rotation solution. Using the $1°$ search resolution, the three points with the lowest values may be given by $(0, e_0)$, $(-1, e_{-1})$, and $(1, e_{+1})$. Let the minimum error point be $(m, e_m)$, so that the error-parabola may be described by Eq. (2.12):

$$e = at^2 + bt + c \tag{2.12}$$

Finding $a$, $b$, and $c$ may be done by using the three known points on the parabola. The abscissa of the minimum error in the parabola is defined by Eq. (2.13):

$$\frac{-b}{2a} = \frac{e_{+1} - e_{-1}}{2(2e_0 - e_{-1} - e_{+1})} \tag{2.13}$$

This step allow for a somewhat improved accuracy of the rotation estimation, for a given search window and resolution. The search grid in this method may be either fixed or refined every iteration.

## 2.5 Scan Matching Using ICP

The ICP implementation presented here was used by Diosi [78] for performance benchmarking against his PSM method. The description follows that given in the work by Diosi [78]. It begins with a median filter, applied to the range readings of both the Reference and Current scan. In each iteration, the projection of the

Current scan is performed in a similar manner to PB-PSM, followed by the checking for occluded points. This is followed by checking if two neighboring (in a bearing sense) Reference or Current scan points occlude the Current scan point being checked. Occluded current scan points are then removed, if they are at least one meter further back than their interpolated reference counterparts (similar to the elimination filter in PB-PSM). Current scan points not in the field of view of the Reference scan are also removed.

After scan projection, the implementation of the closest point association rule follows. For each remaining Current scan point the closest reference scan point is sought in a $\pm 20^{\circ}$ window. No interpolation between neighboring Reference scan points is implemented, which increases speed, but reduces accuracy. Associated points with larger than a set threshold distance are ignored (similar to the thresholds used for both PSM and PB-PSM). Then the worst 20% percent of associations are found and excluded. From the remaining associated point pairs, pose corrections are calculated using equations from the work by Lu and Millios [55] and the current pose is updated. Note that the removal of 20% of the worst contribution requires sorting and therefore this method has some additional complexity, not found in either PSM or PB-PSM.

## 2.6 Navigation Algorithm

In this section, the algorithm responsible for planning the platform's path is described. This algorithm uses the position and map estimates that are the result of the SLAM algorithm. Additionally, the same two reference frames, the global and the body frame, are used both in the Path Planing module, and the SLAM module.

Note that in this work, we refer to "path planning" as a task of computing a path in the form of a sequence of vehicle's poses in a global frame. Therefore, in addition to the essential features that were mentioned in Sub-Section 1.2.8, a path planner should satisfy the following requirement as well:

- Feasibility: a path planner should produce a plan that arrives at a "goal state" (or close enough to it) while keeping a safe distance from obstacles and remaining within the limitations of the helicopter configuration (geometry, turn radius, continuity of velocities and accelerations, etc.).

- Re-planning: in the case of an a priory unknown map, a planner should be capable of recalculating an already planned path according to the newly obtained information about the environment. Obviously, to follow a path that was planned according to partially known information about the surrounding, without re-planning capabilities during the execution, is problematic because of the possibility of collision with obstacles. Moreover, this is true even for a known map, when the path can be computed before the execution starts, since during the flight, the vehicle may discover that what was considered as "known map" is inaccurate, or does not include all of the terrain features. In addition, execution errors and external disturbances may cause a drift of the vehicle lo-

cation compared with the planned path. Therefore, a re-planning capability is absolutely essential.

- Optimality: a path planner should find an optimal path with respect to some specified criteria. Common criteria are path length and clearance (*i.e.*, the minimal distance from the obstacles). Practically, global optimal solution is not achievable since the exact and complete information about the entire surrounding is not always available. Moreover, local optimal solution (between two steps) is not always important as well, since its influence is minor. Therefore, in most of the actual tasks, it is sufficient to find only a feasible path.

### 2.6.1 Definition of the Path Planning Problem

The definition of the Path Planning problem can be expressed in the following form: let the vehicle move in the environment represented in the memory by an occupancy grid (OG), and $p_t$ is a pose of the vehicle at time $t$. As mentioned, the vehicle pose $\mathbf{p}$ is a combination of the vehicle location vector $\underline{\mathrm{x}}$ and the vehicle orientation $\psi$. In a 2D case, the pose is a three dimensional vector: $\mathbf{p} = (x, y, \psi)$, which consists of the vehicle coordinates $x$ and $y$ and the azimuth angle $\psi$ in the global coordinate system.

The set of obstacles is denoted as $O$ and includes all occupied cells of the occupancy grid. Thus, the free space of the environment is the set $W_{free} = \mathcal{OG} \setminus \mathcal{O}$. Suppose $\mathbf{p}_{start}$ is the vehicle start pose, $\mathbf{p}_{goal}$ is the goal position, then the solution of the path planning problem is a sequence of poses $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_T\}$ such that:

1. this path is free of collision with the obstacles: $\bar{\mathcal{P}} \cap \mathcal{O} = \emptyset$, where $\bar{\mathcal{P}}$ is the set of cells in the OG that contains the elements of $\mathcal{P}$

2. the edges that connect any two subsequent points of the path do not collide with obstacles: let $l$ be the line that connects $\mathbf{p}_{t-1}$ and $\mathbf{p}_t$ and let $L$ be the set of cells of the OG that $l$ goes through. Then, $L \subseteq \mathcal{W}_{free}$. This should hold for any $t = \{2, \ldots, T\}$

3. it makes the vehicle move from the start location to the target: $\mathbf{p}_1 = \mathbf{p}_{start}, \mathbf{p}_T = \mathbf{p}_{target}$.

### 2.6.2 Path Planning over a Graph

Many graph-search methods assume that the environment is represented as a 2D occupancy grid with a uniform resolution, in which cells are associated with a traversal cost. In most cases, the traversal costs reflect the physical difficulty of the vehicle to navigate through corresponding sub-areas in the environment. By approximating this grid with a graph, in which nodes indicate the center of each grid cell and edges connect nodes within adjacent grid cells, the path planning problem can be considered as a search problem within a graph from a start node to a target node.

The A* algorithm is a common method for searching a path on a graph. It starts at the root (the start node) and expands the neighboring cells of the current node ("children") with respect to a chosen function $f(n)$, where $n$ is a node, and so on until finding the target node. The function $f(n)$ represents the estimated cost of the shortest path that traverses from the start node to the goal node through that particular node $n$. This cost function consists of two parts: a function $g(n)$, that is the total length of the path traversed so far from the start up to the current node $n$, and a heuristic function $h(n)$. The heuristic hypothesizes an expected "cost to travel" through the path from the current node to the goal node within the graph. It is known that if the heuristic is admissible, *i.e*,. it never overestimates the actual cost of the traverse to the target, the A* algorithm is optimal and always returns the shortest solution [79].

The Euclidian distance $\left(\sqrt{\Delta x^2 + \Delta y^2}\right)$ and Manhattan distance ($|\Delta x| + |\Delta y|$), where $\Delta x = x(n_2) - x(n_1)$, $\Delta y = y(n_2) - y(n_1)$, are the most widely used admissible heuristic functions for path planning tasks. The traversal cost $g$ typically represents the actual distance through the graph where diagonal distances are scored proportionally: if $c > 0$ is a cost of traversing a cell in the horizontal or vertical direction, then the cost of traversing the cell diagonally is equal to $c\sqrt{2}$. Obstacles can be also incorporated in this scheme by introducing the infinite traversal cost $c \to \infty$ for occupied cells.

The formal description of the A* algorithm within the current algorithm is given in Sub-Section 2.6.4. It is a simple algorithm, and when using a proper and efficient heuristic, it is guaranteed that an optimal solution will be returned. These advantages make the A* algorithm widely used in path planning tasks for different types of autonomous vehicles.

### Example for A* Search Evolution

Figure 2.12 (a-g) presents an example of the A* process evolution on a simple two-route pose-graph, from Start to Goal. The start and goal are marked by full circles (red and blue, respectively, with the general direction of motion from left to right). The search-graph is made of 5 nodes, marked by letters from 'a' to 'e', and a set of arrows signifying possible paths, connecting the nodes, where each path is associated with a cost to traverse it (in purple text).

The algorithm's input is shown in Figure 2.12 (a), presenting the search graph with the costs associated with traversing each segment. The algorithm then begins by calculating the cost associated with reaching directly to the goal from each node (Figure 2.12 (b)). The search begins with calculating the associated cost with the first two options: $f(a)$ and $f(d)$, shown in Figure 2.12 (c). Since $f(a) < f(d)$, the algorithm continues to calculate $f(b)$ (shown in Figure 2.12 (d)). At this point, the bottom branch still has the lower cost, and so the algorithm continues to calculate $f(c)$ (Figure 2.12 (e)). Since $f(c) > f(d)$, the top branch now becomes the active branch, and $f(e)$ is calculated (Figure 2.12 (f)). At this point, the lower branch is eliminated due to its higher cost, and the final path solution is highlighted in Figure 2.12 (g).

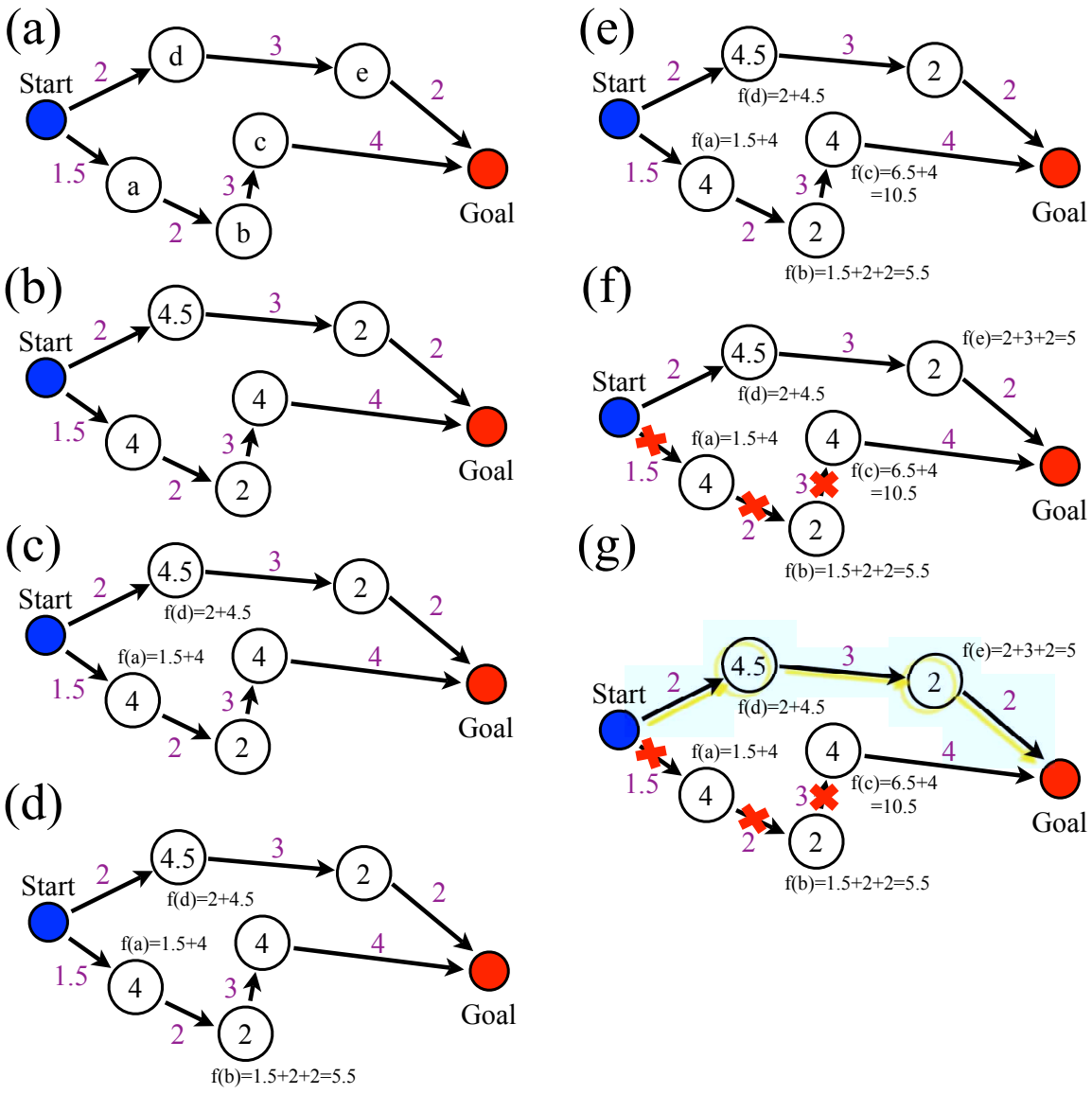



Figure 2.12: An example of the A* process development.

### 2.6.3 Goal Definition

The goal in this work is defined as a distance to be traveled relative to the initial position. The absence of a priori known map prohibits any other goal definition. Moreover, it is assumed that the platform has no artificial intelligence for higher level motion commands. Therefore, motions defined by relative distance may be used for the goal definition, This definition may later be changed based on the incrementally built map coupled with a decision making algorithm (*i.e.* dynamic goal definition).

### 2.6.4 A* Formulation Using an Occupancy Grid

The Path Planner module receives information about the obstacles stored in the OG, the current vehicle position and goal location. Due to the relatively high resolution of the occupancy grid used in this work, the global planner, proposed in this work, makes use of an additional OG with a lower resolution, reducing thereby the workspace and computational costs.

The A* algorithm is applied based on its spacial OG, namely $\mathcal{OG}^*$, whose cells are associated with a value between 0 and 1. This cell value represents the occupancy state of the correspondent area in the environment (where 0 would be free and any value above zero means "occupied"). Although the occupancy is stored in the cells, only binary information is used by the algorithm, because the A* algorithm in this work is only aware of the free workspace $\mathcal{W}_{free}$ and the obstacle space $\mathcal{O}$. The hit values, stored in the cells of the OG, are not being used for the global path planner in this work.

The resolution of $\mathcal{OG}^*$ is selected based on computation resources available, as well as practical reasons such as vehicle proximity. Generally, if a relatively fine resolution is chosen to obtain smoother paths, a safety radius may be implemented around the edges of all obstacles, to secure the platform from collision. The platform is considered as a point by the A* algorithms.

#### Search Grid Generation

The generation of the A* occupancy grid is identical to the generation of the fine resolution SLAM occupancy grid, described in Section 2.1. The update stage that occurs after a successful scan matching is performed on both the SLAM and the A* occupancy grids. Moreover, when cells are cleared using the isolated point filter, for example (see Sub-Section 3.3.3), the corresponding cells are also cleared in the A* occupancy grid as well.

#### Distance Cost Calculation

As mentioned above, each cell center in the A* occupancy grid is considered to be a node in the search graph. In this work, we consider eight directions of movement, including moving along cell diagonals, as presented in Figure 2.13. The traversal cost in any of the horizontal or vertical directions is $D_{straight} = d_{A*}$ and the traversal cost along the diagonal is $D_{diag} = \sqrt{2}d_{A*}$, where $d_{A*}$ is the resolution of the A* occupancy grid. The diagonal heuristic function is calculated as follows [80, 81]:

$$
\begin{aligned}
&h\left(n\right) = D_{diag}h_{diag} + D_{straight}\left(h_{straight} - 2h_{diag}\right) \\
&h_{diag} = \min\left(\Delta i, \Delta j\right) \;\; ; \;\; h_{straight} = \left|\Delta i + \Delta j\right| \\
&\Delta_i = \left|i_{curr} - i_{target}\right| \;\; ; \;\; \Delta_j = \left|j_{curr} - j_{target}\right|
\end{aligned}
\tag{2.14}
$$

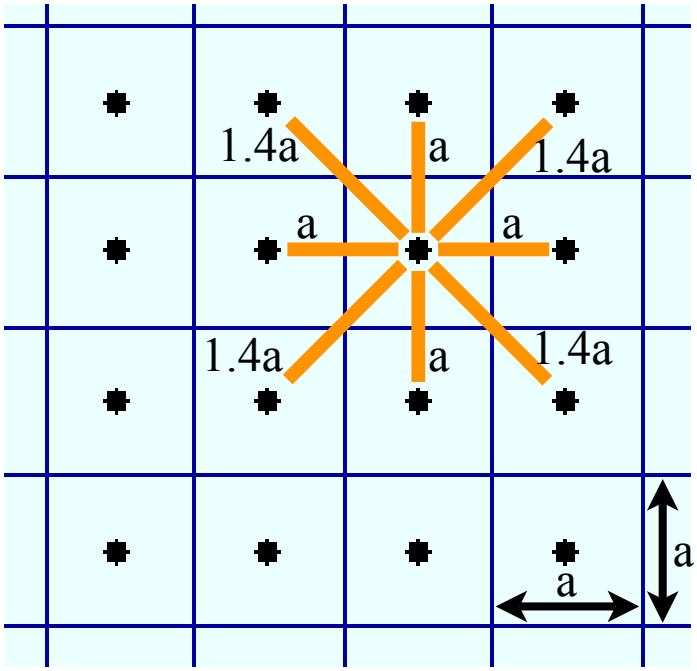



Figure 2.13: A* search grid directions.

where $i_{curr}, j_{curr}$ are the current cell indices, $i_{target}, j_{target}$ are the target cell's indices, $h_{diag}$ is the number of diagonal segments, $h_{straight}$ is the number of horizontal or vertical segments (*i.e.* the Manhattan distance). Hence, two heuristics $h_{diag}$ and $h_{straight}$ are combined, considering the cost of both diagonal steps $D_{diag}$, and straight steps $D_{straight}$. In order to maintain the admissibility of the combined heuristics, the diagonal cost should be less than the cost of two straight steps: $D_{diag} \leq 2D_{straight}$. Such combined heuristics chooses the distance based on whichever of these two (the diagonal or the straight heuristic) gives a lower value.

### Waypoint Search

Once the global path is found, a **waypoint** position in terms of the OG can be calculated. We consider the waypoint as a farthest visible cell of the OG in line of sight along the path computed by the A* algorithm. In order to draw an imaginary line from one cell to another, the Bresenham line algorithm [75] is used. Initially it was developed for drawing lines on a computer screen or by plotters on paper. Because the computer screens have the matrix structure which is very similar to a grid structure (cells may be considered the same as pixels), so this algorithm can be implemented for finding waypoints straightforward. The idea of this method lies in the rasterization of a continuous line with respect to the grid resolution and looks as follows. Let $i_0, j_0$ be the indexes of the start cell, and $i_f, j_f$ be the indexes of the final cell, and we need to find through which cells the line between the start and final cells will be drawn. If only the first octant of the grid is considered, namely when $\delta_i = i_f - i_0 \geq 0$ and $\delta_j = j_f - j_0 \geq 0$, and only increment along the $i$-axis is allowed,

there are only two possibilities:

- the line may be plotted through cell $(i+1, j)$,
- the line may be plotted through cell $(i+1, j+1)$.

One of these possibilities is chosen based on the differences between the midpoint $(i+1, j+0.5)$ and the line from the start to the final cell Figure 2.14. The line equation in this case can be written as

$$f(i,j) = 0 = \delta_j i - \delta_i j + \delta_i j_0. \tag{2.15}$$

It should be noted that this equation involves only integer values of $i$ and $j$.

Thus, considering the start cell, if the value of the midpoint of the next cell $f(i_0+1, j_0+0.5)$ is positive then the midpoint lies below the line, and the cell $(i_0+1, j_0)$ should be chosen. This is equivalent to the difference $f(i_0+1, j_0+0.5) - f(i_0, j_0)$. For the first choice, the difference should be greater than zero, otherwise the cell $(i_0+1, j_0+1)$ will be chosen.

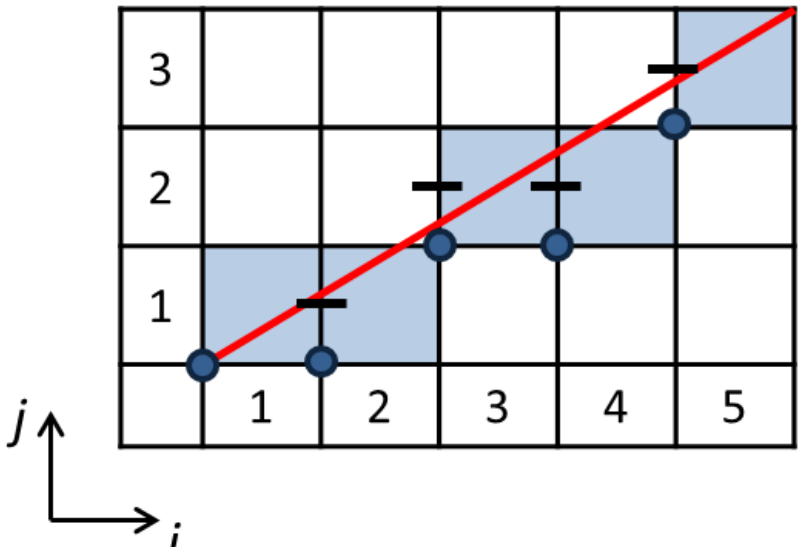



Figure 2.14: Bresenham's algorithm for drawing a line.

The described routine can be easily extended for another octants by changing the increment of the $i$-direction from $+1$ to $-1$, swapping the directions and so on. The main advantage of this method is its speed because it uses integers only.

Bresenham's algorithm outputs the sequence of the cells through which the line of sight from the current cell to another cell of interest traverses. Thus, by checking the traversed cells' occupancy, one may conclude whether that particular cell of interest is seen from the current position. Starting from the goal cell, the farthest visible cell can be found out. The algorithm for waypoint recognition in terms of grid cell is described in Algorithm 4.

**Algorithm 4** An algorithm for Waypoint Recognition.

**Input:** cell sequence $\mathcal{P}_{A*} = \{(i_0, j_0), ... (i_{goal}, j_{goal})\}$ representing the path obtained by A* algorithm

checking each cell starting from the goal:
**for** $(i_{curr}, j_{curr}) = (i_{goal}, j_{goal}) : (i_0, j_0)$ **do**
Invoke Bresenham's algorithm: $\mathcal{S} \leftarrow BresenhamAlgorithm\left((i_0, j_0), (i_{curr}, j_{curr})\right)$
▷ get cells along line of sight from $(i_0, j_0)$ to $(i_{curr}, j_{curr})$
**if** $\mathcal{S} \in \mathcal{W}_{free}$ **then** ▷ all cells $\mathcal{S}$ are free
$(i_w, j_w) \leftarrow (i_{curr}, j_{curr})$ ▷ current cell is the waypoint cell
**return**
**end if**
**end forOutput:** the waypoint cell indexes $(i_w, j_w)$

### 2.6.5 A* Algorithmic Pseudo-Code

The A* algorithm is described in Algorithm 5. The algorithm takes the following as input:

1. The A* occupancy grid $\mathcal{OG}^*$
2. Current vehicle pose $\mathbf{p}_t$
3. Goal position $\mathbf{x}_{target}$.

And outputs a set of segments that connect the nodes that are traveled through, towards the final goal. Note that the A* algorithm transforms the grid representation to the graph representation; the grid's cell centers become the graph nodes. Throughout the algorithm description, nodes and cells are used interchangeably.

In order to operate on an evolving map, the A* algorithm finds the shortest path from the current vehicle position (not from the start position), to the goal. Each time the path has to be recalculated the new graph structure is created and passed to the algorithm, with the current vehicle position representing the start node for the new graph. Thus, such configuration of the A* algorithm fulfills the re-planning feature mentioned above, and incrementally speeds up while the vehicle is approaching to the goal (as the computed path becomes shorter).

## 2.7 Summary - Existing Algorithms

The algorithms presented in this chapter presented the basic concepts and algorithms that are used in this work for the developed PB-PSM algorithm, as well as for benchmarking purposes. These included the occupancy grid, virtual scan, scan matching (including all point filters, data association, and cost contribution), the A* path planning algorithm, and previously published scan matching algorithms. These algorithms are used in Chapter 3 as part of the PB-PSM scan matching algorithm, the developed SLAM, and its coupling with the A* path planning algorithm.

**Algorithm 5** A*-based search algorithm applied for global path planning.

**Input:** create the A* occupancy grid $\mathcal{OG}^*$
**Input:** calculate start node $n_0$ (using current vehicle position, and Eq. (2.1))
**Input:** calculate goal node $n_{goal}$

Create $O =$ **open** and $C =$ **closed** lists.
Initialize $O$ with the start node $n_0$ as a root of a graph.
**while** $O$ is not empty **do**
  Find $(n_{curr} \in O | f(n_{current}) \leq f(n)), \forall n \in O$ ▷ find lowest cost path.
  $O = O \setminus n_{curr}$ ▷ Remove $n_{curr}$ from the open list $O$.
  **if** $n_{curr} \notin C$ **then**
    **if** $n_{curr} = n_{goal}$ **then**
      Extract the solution $\mathcal{P}_{A*}$ and return it
    **end if**
    $C = C \cup n_{curr}$ ▷ Add $n_{curr}$ to the closed list $C$
    **for** each neighbor $n_{neig}$ of the node $n_{curr}$ in $\mathcal{OG}^*$ that is not in the closed list $n_{neig} \notin C$ **do**
      Calculate $g' = g(n_{curr}) + D(n_{curr}, n_{neig})$ ▷ neighbor's tentative $g$-score, where $D(n_1, n_2)$ is the distance between nodes $n_1$ and $n_2$
      **if** $n_{neig} \notin O$ **or** $g' \leq g(n_{neig})$ **then**
        Add the successor node to the open list
        Calculate the $g$-score: $g(n_{neig}) \leftarrow g'$
        Calculate the $f$-score: $f(n_{neig}) \leftarrow g(n_{neig}) + h(n_{neig})$
        Update the back-pointer to the parent $n_{curr}$ in order to further extract the solution
      **end if**
    **end for**
  **end if**
**end while**
**return** unsolvable **Output:** the A* path $\mathcal{P}_{A*}$

# Chapter 3

# Novel Algorithms

This chapter presents the novel algorithms developed within this work. The chapter begins with presenting the PB-PSM scan matching algorithm in great detail, using all point filters that were introduced in Chapter 2. The cost function construction is described, including cost rewarding options, and minimization process. The multiple minima problem is then presented and analytically analyzed, showing a validation of the adaptive direct search solution method. The extraction of statistical properties from the scan matching solution are then presented, and their possible applications are discussed.

The SLAM algorithm developed in this work follows, with a full description of the coupled SLAM and path planning algorithm for targeted flight operation. The chapter concludes with the associated assumptions and limitations, complemented by analytical analyses, and several novel accuracy metrics are introduced, which are used later for measuring the accuracy of the SLAM map and position output.

## 3.1 Scan Matching Using PB-PSM

This section describes the construction of the PB-PSM scan matching algorithm. Like the PSM algorithm, PB-PSM uses the polar coordinate nature of the range scans. It shares the same data association method as PSM [29], using the "matching bearing" rule, as well as the derivation of the cost function contribution of each associated point pair to the total cost function. However, the cost construction is complemented with a rewarding term which encourages matches with greater overlap between them. Additional rewarding terms are discussed.

The cost minimization process is then described. Since this work relies primarily on scan matching for both position and map generation, the most promising cost minimization method was found to be the use of some measure of brute force in minimizing the cost function, in order to overcome the common multiple local minima problem. An adaptive direct search method is described, and its ability to accurately find the global minima is shown and analyzed analytically. This section makes use of the

### 3.1.1 Cost Function Construction

For any given triplet of $\Delta x, \Delta y, \Delta\psi$, the cost function to be minimized comprises of the absolute differences in radii between the Reference Scan and the roto-translated Current Scan (interpolated values, see details below). Note that only valid points are considered for contributions to the cost. The cost is then rewarded based on the amount of perimeter overlap achieved between the two scans for the attempted triplet. The cost function calculation for any given pair of scans, is constructed by the following steps (after employing the Minimum/maximum range filter, and the Outlier filter at the beginning):

i. Acquire the current laser scan.

ii. Outlier filter: discard laser points on along edges of surface discontinuities (see Sub-Section 2.3.1 for more details).

iii. Calculate $P_0$ - the length of the perimeter created by all valid Reference scan points, using Algorithm 7.

iv. Acquire a virtual scan of the environment (by ray casting over the occupancy grid), from the previous position estimate, or any improved estimate for the current position (see Section 2.2).

v. Minimum/maximum range filter: discard points where $r_C < R_{min}$ or $r_C > R_{max}$ (Sub-Section 2.3.1).

vi. Roto-translate the laser scan with proposed motion. The roto-translation equations are given in Eq. (3.1):

$$\begin{aligned} x' &= r \cdot \cos(\theta_C + \Delta\psi) + \Delta x \\ y' &= r \cdot \sin(\theta_C + \Delta\psi) + \Delta y \\ r' &= \sqrt{x'^2 + y'^2} \\ \theta' &= tan^{-1}(y', x') \end{aligned} \tag{3.1}$$

The results $x'$ and $y'$ are the laser point's roto-translated cartesian coordinates, while $r'$ and $\theta'$, are the roto-translated laser points in polar coordinates (range and bearing, respectively).

vii. Field of view filter (Sub-Section 2.3.1).

viii. Occlusion filter: discard laser points that become occluded by other surfaces after the roto-translation. This is found by searching for angle reversal in the roto-translated laser points (Sub-Section 2.3.1).

ix. Data association: For each valid virtual scan angle, find $r'_{C_{right}}, r'_{C_{left}}$, which are the corresponding laser points on the right and left respectively (Sub-Section 2.3.2).

x. Calculate the interpolated range $r''_C$ (Sub-Section 2.3.2).

xi. Calculate the current point's contribution to the cost function using Eq. (3.2):

$$F_i = |r''_{C_i} - r_{R_i}| \tag{3.2}$$

xii. Discard contributions where $F_i > T_E$, where $T_E$ is the elimination threshold (see details below).

xiii. Calculated $P$ - the length of the perimeter created by all the Reference scan points for which $F_i < T_M$, where $T_M$ is a threshold for successfully matched points (see Sub-Section 3.1.2).

xiv. Calculate the final cost function value given by Eq. (3.3):

$$f = \left(\frac{1}{n_c}\sum_{i=1}^{n_c} F_i\right)\left(1 - \frac{P}{P_0}\right) \tag{3.3}$$

The cost $f$ is normalized by $n_c$ – the number of points that contribute to the total cost. This normalizes the total cost values between different matching attempts. The cost is rewarded using the term $\left(1 - \frac{P}{P_0}\right)$. This reduces the cost for matching attempts with larger matched perimeters. It essentially results in favoring roto-translation matches with a higher overlap between the two scans. When this term is not used, small surfaces may sometimes be inadvertently discarded in favor of matching the larger object in a scene (as will be shown later). Other rewarding function may be used, however the linear rewarding function used here was found to be adequate.

Unlike the PSM algorithm, PB-PSM does not make use of least squares for the translation solution. In fact, a valid point only contributes the absolute distance between itself and its matched counterpart. This has a relative advantage as some wrongful matches, which may be the result of errors in one of the scans, will not contribute the square of their respective distance from their matched counterpart. Thus, the effect of a singular contribution is less likely to significantly sway the solution.

### Eliminating Matching Anomalies

After the matching loop is performed, some contributions of matching anomalies are eliminated (matching anomalies are points with matched distance over three orders of magnitude larger than the typical occupancy grid cell size). This is based on the possibility of wrong pairings of points from different surfaces. Due to the occupancy grid's sparse nature, in some cases, a virtual ray can "see" through walls (go

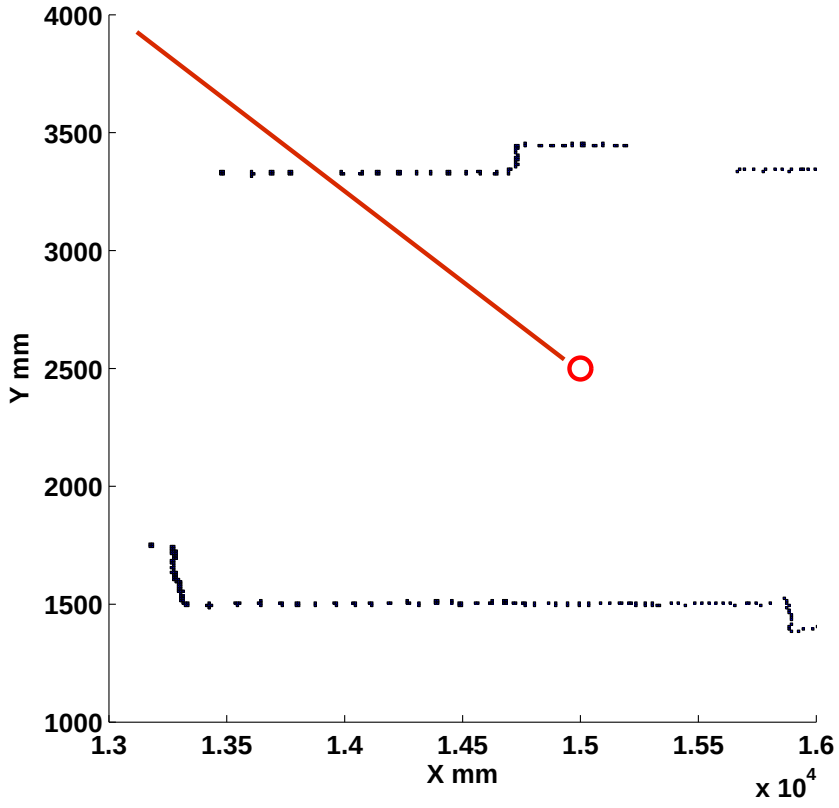



Figure 3.1: Example of a sparse occupancy grid where a virtual beams can penetrate obstacles.

through the actual location of the obstacle as it is not completely mapped yet). An example is given in Figure 3.1.

### Threshold Settings

1. The minimal radius was set on $R_{min} = 400\ mm$ considering the platform size.

2. The maximal radius was set on $R_{max} = 29000\ mm$ to avoid readings at the very end of the laser sensor's maximum range of 30 $meters$.

3. The "matching anomaly" threshold $T_E$, is a user defined threshold, and is typically set to be two orders of magnitude larger than the typical cell size. This additional point filtering is designed to eliminate the possibility of wrong pairings of points from different surfaces, that may occur due to the occupancy grid being sparse in some cases, and therefore some virtual rays "see" through walls. In this work, this number is set to $T_E = 1000\ mm$.

4. Typically, the "matched point" threshold $T_M$ is set to be a number of the same order of magnitude of the occupancy grid, and with a certain respect to the

laser range accuracy. In the current work, a value of $T_M = 50\ mm$ was used, as the grid cell size was 10 $mm$, and the laser had an accuracy of approximately 0.5% (for ranges over 1 $m$), *i.e*, for a typical corridor length of 10 $m$, the laser accuracy would still be able to provide a good match, under the above set threshold.

### 3.1.2 Cost Function Rewarding

This work introduces a new and innovative way to overcome a common problem in scan matching. Many scenarios contain objects with significant differences in size. The representation of these objects in a laser scan depends strongly on the proximity of the laser scanner; The closer the laser scanner is - the more laser points will represent the object. Specifically, in office-like scenarios, surrounding room walls and corridors constitute the majority of the laser scan, while smaller objects such as door posts, trashcans, pillars, etc., are typically represented by a significantly smaller number of laser measurements. This section discusses several optional methods for rewarding the cost function, which is important in order to guide the scan matching process convergence towards the correct scan matching solution.

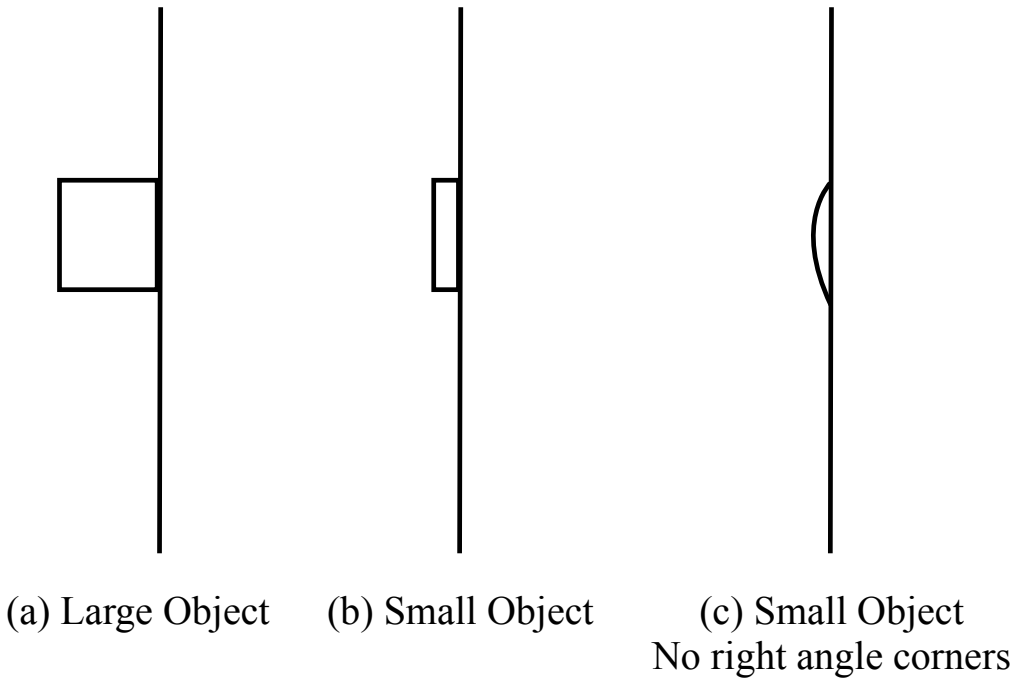



Figure 3.2: Possible scenarios with objects of different scale.

Three scenario examples are given in Figure 3.2, scenario (a) presents a wall with a fairly large extruded box-like object, scenario (b) presents a wall with a relatively

small-sized object, and scenario (c) presents the same as scenario (b), but the object has not sharp corners. These three representative scenarios are used hereafter to describe the challenges involved with scan matching in environments that contain objects of different scales.

Several attempts were made in the literature [78] as well as in this work to find a suitable bias for the cost function, in order to improve it's performance in scenes that contain objects of different size and shape. A discussion of several possible solution is brought herein.

### Cost Rewarding Based on Number of Points

One simple approach is rewarding the cost function based on how many points contributed. Ideally, even the few number of points that represent the outstanding objects will have an effect, and the solution convergence process will be drawn toward matching those objects as well. The rewarding term in this case is given by Eq. (3.4):

$$k = 1 - \frac{n_{con}}{n_R} \tag{3.4}$$

where $k$ is the cost rewarding coefficient ($0 \leq k \leq 1$), $n_{con}$ is the number of points that had contributions to the cost function value, and $n_R$ is the total number of points in the reference scan. Points that may not contribute to the final cost function value include occluded points, outliers, and points whose contribution was eliminated by the elimination threshold ($T_E$, see Sub-Section 3.1.1).

However, it was found that reducing the final cost based on the number of matched points did not always work in favor of finding the globally optimal scan matching solution. Often walls that are relatively close to the origin are represented by many points, and therefore receive a significantly higher "weight", as compared to walls that lie further away, although in practice those are equally as important.

Moreover, the farther points are typically associated with a higher laser noise magnitude, as the laser noise is a percentage of the measured range. Therefore, an apparent better match of closer objects may cloud the importance and contribution of farther points. This phenomenon was commonly observed in typical office scenarios, where the door posts which are of the order of 10 $cm$ were in many cases "ignored" by the scan matching minimization process, which yielded a sub-optimal solution.

### Cost Rewarding Based on Range Histograms

Cost rewarding was also attempted based on histogram matching, where after the roto-translation (Eq. (3.1)), the range values in both scans would be sorted into a histogram with $N_b$ bins. The bin values from both scans form two vectors $h_1$ and $h_2$, which may be used as follows:

$$k = 1 - \frac{h_1 \cdot h_2}{max\left(h_1^2, h_2^2\right)} \tag{3.5}$$

where $k$ is the cost rewarding coefficient ($0 \leq k \leq 1$). This method proved to be efficient only in a limited number of scenarios, while often leading to wrong solutions in others. An additional attempt was made in Cartesian coordinates, where two histograms were generated, for both the $x$ and the $y$ dimensions. This approach was found to be more useful for planar translation estimation, but failed to produce good results on many different scenarios which included a significant rotational component.

### Cost Rewarding Based on Line Matching

Another method to give considerations to smaller objects is segmenting the scan into lines. Both scans may be fitted with a set of lines, which may then be used for carrying out a fast line matching operation. The solution from the line matching operation may be used to give an improved initial guess for the scan matching process, or as a rewarding term for the final scan matching cost function. Fitting straight lines to a set of points may be done using least squares or end-point fitting. The latter method was selected as the computational cost is significantly reduced, with similar results. A pseudo code for line extraction based on end-fitting is given in Algorithm 6.

**Algorithm 6** Line extraction using end-point fitting

```
flag ← true
while flag = true and n_Li <= n_Li_max do      ▷ while the algorithm is not finished and the max number of lines is not reached
    flag ← false                                ▷ assuming this is last iteration
    for i = 1 → n_Li do                         ▷ checking all the lines created thus far
        d_max ← max(d)   ▷ registering largest distance between the points and the current line
        if d_max > ε_L then
            Split current line, make farthest point a new node-point.
            flag ← true                         ▷ one more iteration required
        end if
    end for
end while
```

The method requires the definition of $n_{Li_{max}}$, the maximum number of lines that can be output from the algorithm, as well as $\epsilon_L$, the threshold for the fitting of a point to its representative line. These parameters are user controlled, and should be set based on some experience with several scenarios. Typically, in this work, values of $n_{Li_{max}} = 50$, and $\epsilon_L = 5\ cm$ were used, which guaranteed capturing all the lines with significant resolution, in most of the indoor scenarios attempted.

The output lines may be used for fast scan matching in a number of different ways. The methods and associated advantages and disadvantages are brought herein:

1. using the line angles to estimate the rotation between the two scans. This may be done by generating histograms of the line angles in both scans, and searching for the rotation angle of the Current scan that will generate the best

match with the histogram of the Reference scan. However, discrepancies in line lengths, errors from the line extraction process, and a relatively low number of lines are not desired, as the histograms that are based on them are quite inaccurate.

2. using the line lengths in a similar manner to the process described for the angles. This approach has the same pitfalls as the previous one described for the line angle histogram.

3. re-distributing $n_p$ points along each line, and performing a point-based scan matching. This method makes sure that each line receives the same representation in terms of number of points. Therefore, it is expected that this method will overcome the common pitfall of rewarding based on number of points (object size discrepancy discussed above). However, after several attempts, it was found that the issues of line extraction mentioned above, also affected the outcome in this method, and often this method had a negative influence on the solution and its convergence.

The above three options were briefly attempted as part of this work. Although some of these methods proved to be useful on some occasions, there have been many scenes where the number of lines was minimal (mainly long corridors), or where the thresholds set for the line extraction algorithm yielded undesired results (such as not representing door posts, while including their length into nearby walls). Based on the lessons learned from these experiences, these methods were not used in the results presented in this work.

### Cost Rewarding Based on Corner Matching

Complementing the line extraction method, features can also be extracted from a scan in the form of corners [78]. The parameters associated with corner extraction are similar to those of line extraction. In fact, corner extraction may be a by-product of the line extraction algorithm described above, where every line end-point may be considered as a corner. Another approach is to examine straight line fitting through sequential points, and when large change in the line orientation occurs - the last point added tot eh inspected set is defined as a new corner and inserted into the corner list.

The corner points may be used as the basis for performing a point-based scan matching. The main advantage is that the number of corners is expected to be significantly lower than the total number of points in the scan. Note that the threshold for defining a corner may not allow some blunt (obtuse) angles. The reason is that because of typical laser noise, many points may falsely qualify as corners, and thus this approach may fail.

The main disadvantage of using this approach may be in scenarios that do not contain many corners (*e.g.* long corridors with not too many features), such as the scenario presented in Figure 3.2 (c). In such scenarios, the usage of this approach may result in a relatively low number of corners, and in some cases no corners at all. Moreover, some scenarios may only contain round objects, which may fall outside of

the corner threshold. An example includes the forest environment used in this work (see Sub-Section 4.3.5) which contains only round tree-trunks. The success of any feature extraction approach depends on how accurate do the features represent the environment. Therefore if features like lines and corners are used in an environment without straights objects and corners, the accuracy is expected to be lower. This approach was not attempted in this work.

### Cost Rewarding Based on Perimeter Matching

In order to provide a suitable bias in the cost minimization process, the final cost is reduced based on the ratio of matched perimeter to total perimeter captured by the laser scanner. The cost after applying the reduction term is given by:

$$f = f\left(1 - \frac{P}{P_0}\right) \tag{3.6}$$

where $P_0$ is the total perimeter of the reference scan, and $P$ is the perimeter of the portion of the reference scan that was successfully matched to points in the current scan. Since both perimeter values are calculated based on the reference scan, there is no possibility for $P > P_0$, and the overall perimeter matching rewarding term in Eq. (3.6) is at the very least semi positive definite.

Moreover, the perimeter matching term may not be identically zero since it requires that every single point in the Reference scan has two valid associated points in the Current scan. This situation cannot exist, even in the case of an ideal and identical scan of the same scenario. Since the number of points in both scans is equal ($n_R = n_L$), one of the outermost points in the Reference scan won't have two neighbors, even when scanning identical scenarios, with no shift between the scans.

In most scenarios, not all points are matched and therefore the term typically yields a positive number smaller than one. Moreover, line segments that pass through an empty space in the scan should not be included. This can be examined using the scan resolution and range: if the distance between two connected points is larger than $R_{max}\Delta\theta$, then the contribution to the perimeter is discarded, where, $\Delta\theta$ is the resolution of the reference scan. Every point in the Reference scan that is matched may only be considered once, with its associated segment length to its succeeding point. The calculation of $P_0$ is given in Algorithm 7, below.

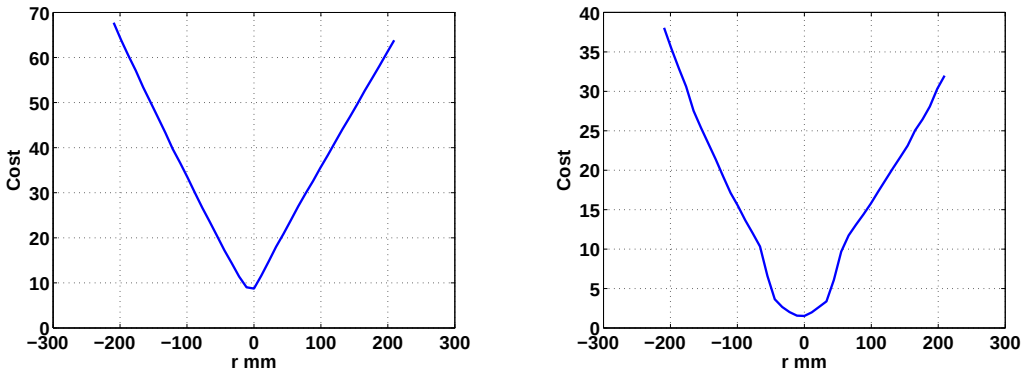
Note that in case that the reference scan is a virtual scan, checking for a valid point may also be done using the occupancy of the representative cell for that virtual ray, not just using the range value itself.

For a point to be considered as "matched" the contributed cost was required to be of the same order of the grid resolution. This is another measure designed to eliminate matching anomalies from interfering with finding the accurate scan matching solution. The result is that the longer the perimeter created by the matched points is – the lower the total cost becomes. This modification was found to be directly responsible to improving scan matching solutions, and in some cases even correct past mapping mistakes, that were wrongfully updated into the occupancy grid.

**Algorithm 7** Calculating $P_0$

```
P_0 ← 0                                              ▷ Initializing perimeter value
i ← 1                                                ▷ Initializing point index
while i ≤ n_R − 1 do
    if r_i < R_max then                              ▷ If current point is valid
        j ← i + 1
        flag ← FALSE
        while flag = FALSE do                        ▷ searching for the next valid point
            if r_j ≤ R_max then
                j ← j + 1
            else
                flag ← true
            end if
            if j > n_R then break
            end if                                   ▷ in case the last point was reached
        end while
        if j > n_R then break
        end if                                       ▷ exiting the outer loop as well
        Δ_x ← r_{V_i} cos(θ_i) − r_{V_j} cos(θ_j)
        Δ_y ← r_{V_i} sin(θ_i) − r_{V_j} sin(θ_j)
        ΔP_0 = sqrt(Δ_x^2 + Δ_y^2)
        if ΔP_0 < R_max Δθ then                      ▷ Empty space segments check
            P_0 ← P_0 + ΔP_0
        end if
        i ← j − 1
    end if
    i ← i + 1
end while
```

The algorithm for calculating the matched perimeter is identical to the one described above for calculating $P_0$, except that each point validity is also measured by that point's contribution to the final cost function, with the added condition being $F_j \leq T_M$. Typically, for a grid resolution of 10 $mm$, a value of $T_M = 50$ $mm$ was used, and therefore a point is considered to be matched if its matching contribution is smaller than 50 $mm$.




(a) Without perimeter matching term (b) With perimeter matching term

Figure 3.3: Effect of perimeter matching term on the cost function shape.

The effect of the perimeter matching term is also visualized in Figure 3.3, where the cost function for a a representative scenario and iteration is shown with and without the use of the perimeter matching term. The use of the perimeter matching term cause a dip in the cost function shape which contributes for a more robust minimization.

### 3.1.3 Cost Function Minimization

To find the best scan matching solution, an adaptive direct search method was constructed, where the best rotation angle between the scans was found first, followed by the best pure translation. This process is repeated in an iterative manner while continuously reducing the range of the search grid in a geometric rate after each iteration (thus refining the search grid, in both the plane and azimuth, while the number of grid points is kept constant). It was found that estimating rotation before translation is typically beneficial for convergence, as rotation has a more pronounced effect on the cost function. In the plane, the search grid shape is in the form of a circle, and the points are evenly distributed along the radial direction and about the azimuthal direction. Additional search grid shapes were attempted (ring-section and square shaped); however, the circular grids were found to be the most robust.

Using the algorithm described above, the best possible scan matching is, in most cases, guaranteed as long as the search range is large enough, and the search grid is fine enough to adequately cover the solution area. In the majority of the results presented, 50 points were used for the azimuthal grid, and the planar grid size was 7 by 7, so each iteration required 100 function evaluations. It was found that for most

cases, the above search grid resolution appears sufficient. A mesh refinement scheme may be employed for cases where the resulting cost function is too high. However, this was not required in the current work.

Convergence is defined based on the change between two subsequent iterations as in Eq. (3.7):

$$[\Delta x^n, \Delta y^n, \Delta \psi^n] = \left[x^n - x^{n-1}, y^n - y^{n-1}, \psi^n - \psi^{n-1}\right] \tag{3.7}$$

where $n$ is the iteration number. Convergence is achieved when the above values are less than $\varepsilon_t = 1\ mm$ for translation (both $x$ and $y$), and $\varepsilon_\psi = 0.01°$ in azimuth. On average, it was found that a total of approximately 8 iterations were required for each scan matching process. Cases of scan matching failure are identified by $f > T_F$ where $f$ is the final cost, and $T_F$ is an elimination threshold, which is typically set to 10.

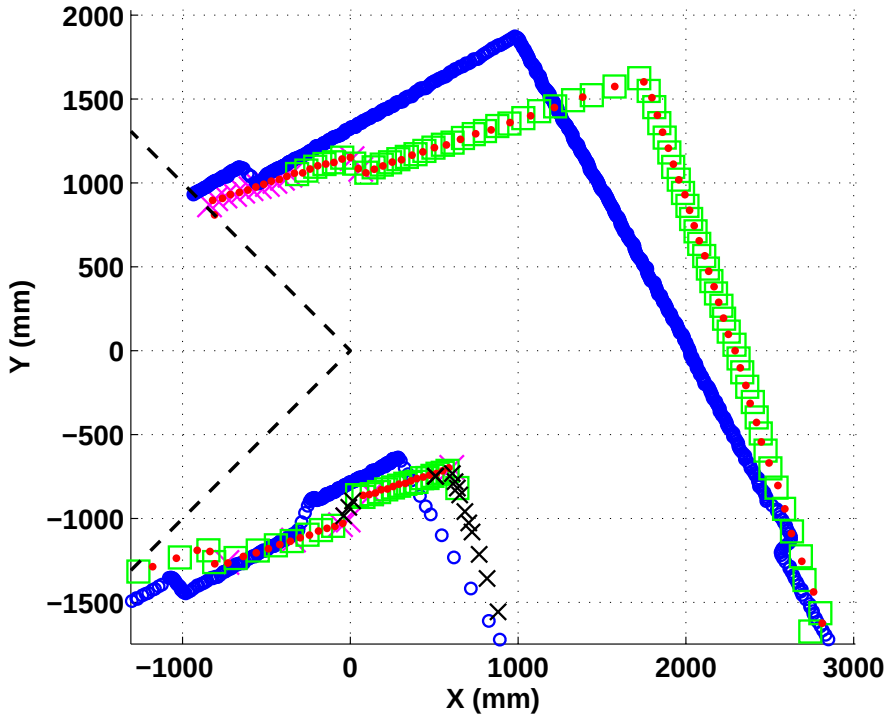



Figure 3.4: Scan matching example on a simple corner-like geometry.

An example of a typical scan matching result is given in Figure 3.4, where circles represent the Current Scan's laser readings taken from the origin at $[0, 0]$ (all points are shown in order to accurately represent the geometry captured by the laser scanner), while the laser FOV is marked with dashed lines. The Reference Scan is represented by dots, and the matched roto-translated Current Scan points are represented by squares (here, not all points are shown, for clarity). Points that were eliminated by the various filters described above are crossed with 'x' (some points are outside the field of view, some are possible outliers, and some are occluded as the laser

picked up points from around the corner while the Reference Scan was taken from the previous location). The current algorithm is implemented in two dimensions and assumes that both planar and rotational platform motions are slow compared to laser scanner speed. The algorithm also assumes a relatively small laser sensor pitch and roll attitudes (ensuring a 2D environment).

**Notes About Cost Function Minimization**

- Finite Resolution search grid: it is important to note that at every iteration, and particularly in the last iteration, the minimum cost is always found on one of the search grid nodes. Since the search grid has a finite resolution, the minimum point may not be the true minimum point. However, since in this work, the typical employed convergence requirements are relatively tight, the difference between the found and the actual minimum points may not be larger than the convergence requirement.

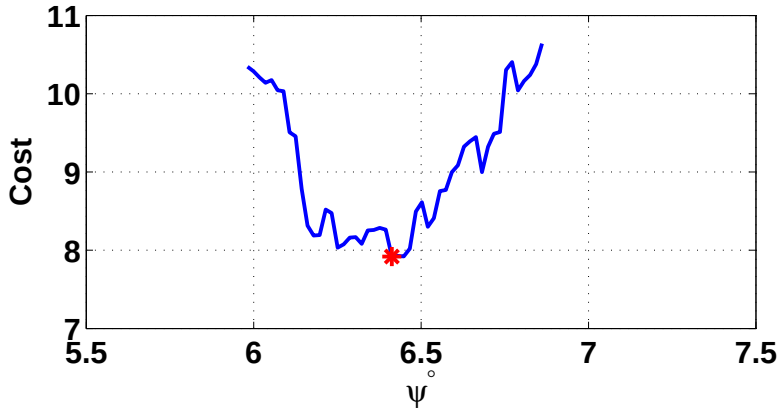
- The search process used here separates the rotation variable $\psi$ from the two translation variables $x$ and $y$. This separation was utilized to reduce the computational cost of the minimization by reducing the total number of cost function calls. A three dimensional exhaustive search grid was not attempted as it was found to be computationally infeasible (similar search grids would have resulted in 5000 function calls for each iteration). This separation concept was also utilized by Diosi and Kleeman [29]. The minimization process separates the rotation variable $\psi$ from the translation variables $x$ and $y$, since they create fundamentally different errors in the cost function.

### 3.1.4 Multiple Minima

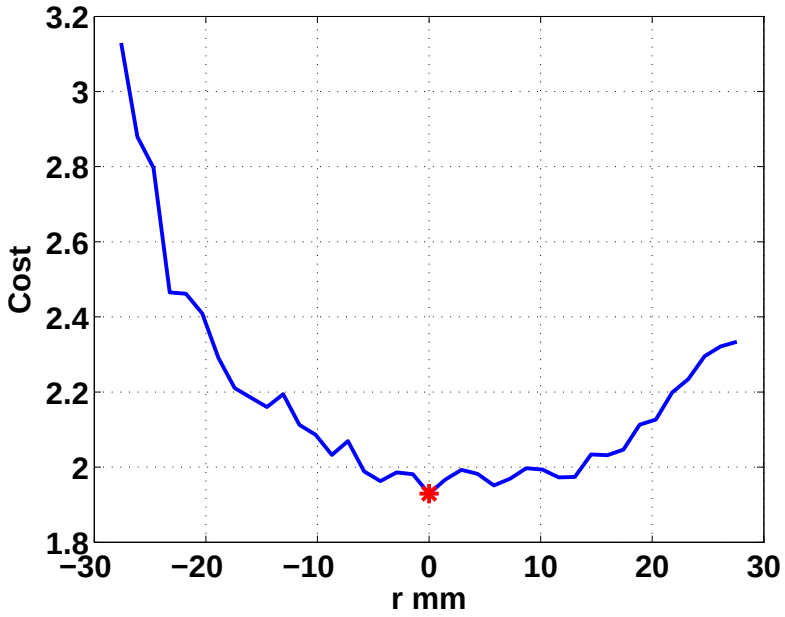
The minimization approach used in PB-PSM is a form of an exhaustive search. Although it is a more computational demanding as compared with gradient search methods, the main reason for choosing this method is the existence of multiple local minima for the cost function. While gradient search based methods are susceptible to local minima, and may therefore converge to the wrong solution, an exhaustive search method is more immune to this problem, provided that the search grid is fine enough.

Two examples showing the existence of multiple local minima are presented in Figure 3.5. Figure 3.5 (a) shows the phenomenon for the azimuthal search phase, where several local minima are clearly visible, with significant differences in the azimuthal angles. It is important to note that capturing the azimuthal rotation accurately is an important key in achieving overall accuracy. It will be shown later that differences greater than $0.01^\circ$ were found to be significant for achieving overall accuracy (see the results chapter, Sub-Section 5.2.1).

Figure 3.5 (b) presents the second example, taken from a planar search for the translation solution. The search grid shape is circular, and so a cross section is drawn with the $x$ axis showing radial values (positive to one direction for the cross




(a) Azimuthal search multiple local minima (asterisk marks global minima).




(b) Translation search, cross section of the cost function planar surface.

Figure 3.5: Multiple local minima. Top: varying $\Delta\psi$ only, showing multiple local minima, with a significant angle difference. Bottom: showing a similar phenomenon for the planar translation search, varying $\Delta x$ and $\Delta y$, with $\Delta\psi$ constant (negative values correspond to the opposite side of the search grid).

section and negative in the other direction), and the $y$ axis showing the values of cost function, achieved while $\Delta\psi$ is kept constant. The existence of many local minima is clearly seen in this Figure 3.5 (b), with significant differences of up to 20 $mm$ in their location.

These patterns were observed in nearly every scan matching scene that was attempted. They are especially pronounced when the minimization process has completed several iterations. The sometimes small differences between the global minima location and a given local minima location may accumulate over the distance trav-

eled by the platform and form a distorted map. Therefore, the conclusion was to use adaptive direct search throughout the minimization process, in order to maintain the higher accuracy solution output.

### Minimization Process Analytical Validation

In order to test the minimization procedure performance, an analytical function is utilized, which has the same general characteristics as exhibited by the PB-PSM cost function. As seen in Figure 3.5, for a given $x$ and $y$ values, the shape of the cost as a function of $\psi$ is approximately a parabola, and for a given $\psi$ value, the shape of the cost plane formed by varying $x$ and $y$, resembles a paraboloid. For this reason, the function given in Eq. (3.8), which features both above mentioned properties, has been selected for the suggested evaluation:

$$C = \left(a_1\psi^2 + b_1\psi + c_1\right)\left(a_2x^2 + b_2y^2 + c_2\right) \tag{3.8}$$

where $a_1, b_1, c_1, a_2, b_2, c_2$ are constant coefficients, and $C$ is the resulting cost. To conform with the behavior exhibited by the scan matching cost function, $C$ must remain positive. For convenience purposes, $c_2$ is assumed positive, which then requires the quadratic function in $\psi$ to be positive for all $\psi$, which means it must have a negative determinant. These conditions satisfy a positive cost for all $x$, $y$, and $\psi$.

Under the above assumptions, this function has a single unique global minima, which occurs when both the quadratic function in $\psi$ and the second term achieve their minimum value. The analytical solution for the global minima of the first term is given by $-\frac{b_1^2}{4a_1} + c_1$, where $\psi = -\frac{b_1}{2a_1}$. For the second term, the minimum is simply achieved when both $x$ and $y$ are zero, and it equals $c_2$. The cost global minimum point is therefore given by:

$$C_{min} = c_2\left(-\frac{b_1^2}{4a_1} + c_1\right) \tag{3.9}$$

For this evaluation, all constant coefficients were set to: $a_1 = 1$, $b_1 = 2$, $c_1 = 3$, $a_2 = 4$, $b_2 = 5$, $c_2 = 6$, and therefore the solution for this function's global minima is given by $[x, y, \psi] = [0, 0, -1]$, wit the minimum point having a value of $C = 12$. The initial guess for this case was set as $[x_0, y_0, \psi_0] = [-0.5, 0.5, 0]$, the search grid in $\psi$ had 50 points, and the translation search grid was a 10 by 10 grid (as in the experimental cases). It was found that the solution accuracy increases with the tightening of the convergence requirements, and for a uniform convergence requirement of $\varepsilon = 0.001$ for all variables, the result from the adaptive direct search algorithm was $[x, y, \psi] = [-0.00167, -0.000254, -0.99922]$, with the minimum point achieved at $C = 12.00009$. Therefore it is concluded that for this representative function, the adaptive direct search achieves the correct minimum point.

An additional function was attempted, which is more challenging as it introduces multiple minima. The translation term remained unchanged, while the rotation term

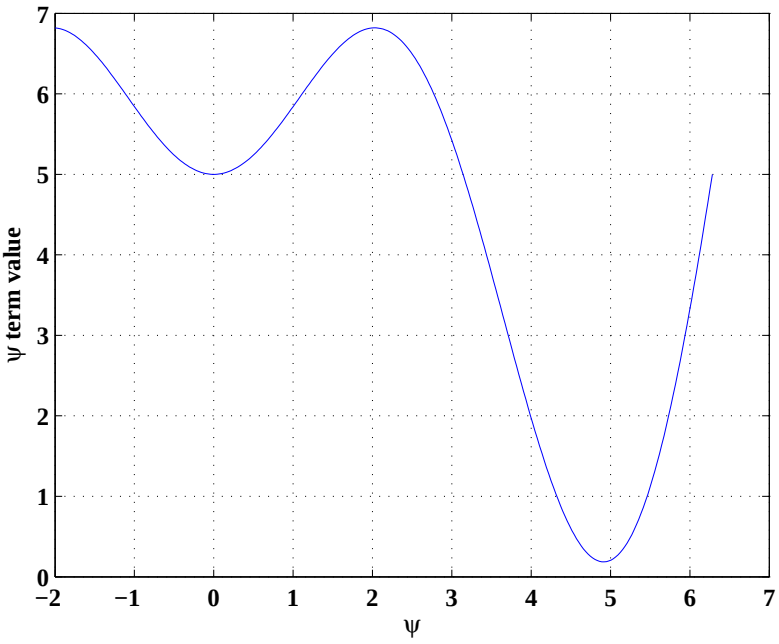



Figure 3.6: Multiple local minima analytical test function.

was changed to a mix of trigonometric and polynomial function.

$$C = (a_1 \psi sin(\psi) + b_1)\left(a_2 x^2 + b_2 y^2 + c_2\right) \tag{3.10}$$

where $a_1$, $b_1$, $a_2$, $b_2$, $c_2$ are constant coefficients, and $C$ is the resulting cost. The azimuthal term is plotted in Figure 3.6, with values of $a_1 = 1$, and $b_1 = 5$ (which is used to maintain a positive function value, as is required from the cost function), and one may see the two minima points in the range plotted. The same range was used for the minimization process using the adaptive direct search. In this example, $a_2 = b_2 = c_2 = 1$, for simplicity.

As in the previous example, the analytical solution for the cost minimum requires both term to achieve their respective independent minimum values. The translation term minimum point remains unchanged. The rotational term minimum point may not be found analytically since it requires the solution for the zero derivative of the form:

$$f'(\psi) = sin(\psi) + xcos(\psi) \tag{3.11}$$

Newton's method was therefore employed to found the zero derivative point as follows:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \tag{3.12}$$

where $f'(x_n)$ is the derivative of the azimuthal term, and $f''(x_n)$ is the second derivative, both evaluated at the point $x = x_n$. Here, $n$ is the iteration number of Newton's method. The solution for the zero derivative was found at $\psi = 4.9318$ (using an

initial guess of $x_0 = 5.5$), which gives an azimuthal function value of $C = 0.1855301$.

The initial guess was $[x_0, y_0, \psi_0] = [-0.5, 0.5, 0]$, which placed the azimuth search right at the local minimum at $\psi = 0$. However, the adaptive direct search successfully found the global minimum point value of $C = 0.185532$, located at $[x, y, \psi] = [-0.00088, -0.000511, 4.9122]$. Therefore it is concluded that the adaptive direct search method that was developed in this work is capable of handling multiple local minima, when attempting to find the global minima point.

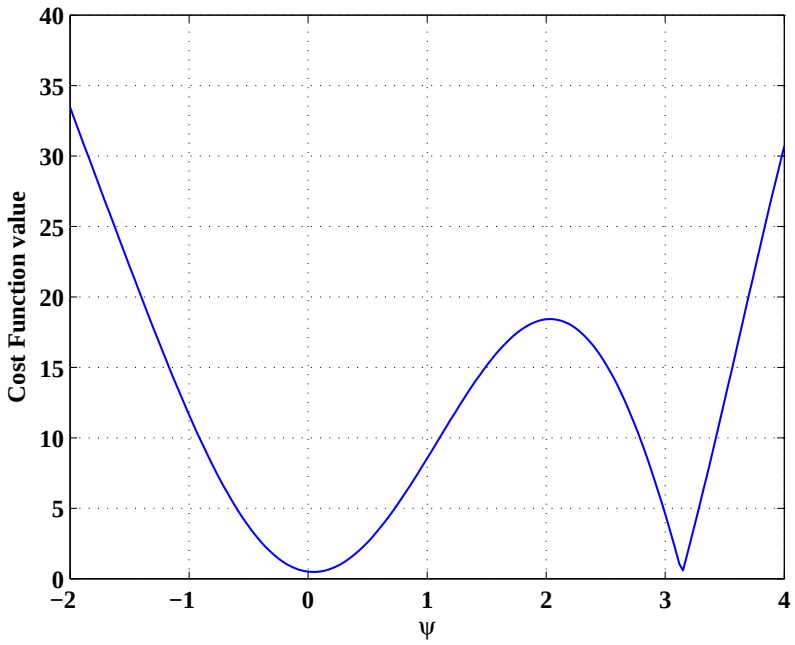



Figure 3.7: Multiple local minima, with contradicting requirements.

An additional test was carried out where the chosen cost function was of the following form:

$$C = |a_1 \psi sin(\psi)|(a_2 x^2 + c_2) + (-\psi e^{-\psi} + 0.5)(b_2 y^2 + c_2) \tag{3.13}$$

where the absolute value on the first term, and the constant 0.5 in the second term are required to maintain a positive cost value for all $\psi$ values. In this case, the two terms in $\psi$ have a different minima points, and therefore the global minima would be a compromise between the minimization of both terms. The coefficient $a_1$ was set to $a_1 = 10$ (in order to get a minimum point within a reasonable range of values), while all the other coefficients were set to $a_2 = b_2 = c_2 = 1$.

The variation of the cost with $\psi$ for Eq. (3.13), with $x = 0$, and $y = 0$, is shown in Figure 3.7. Although the minimum point remains at $[x, y] = [0, 0]$, an analytical solution for $\psi$ may not be found for this cost function, due to its highly non-linear nature. The global minimum point was therefore found using an exhaustive search in $\psi$, with a relatively high search grid resolution. The exhaustive search based solution was found to be $\psi = 0.04562$, with $C = 0.47721905$. The adaptive direct

search in this case produced a minimum point of $C = 0.47721917$ at $[x, y, \psi] = [-0.00046, -0.000459, 0.04557]$. This means that the adaptive direct search was able to minimize this function as well with good accuracy.

## 3.2 Statistical Properties Extraction

Statistical properties may be extracted from the PB-PSM algorithm output. These include the mean, standard deviation, and covariance of the final scan matching solution. These quantities may give a measure for the scan matching quality for a given scan matching scene. Moreover, these quantities may be used for data fusion with other algorithmic outputs or sensors that may be available on board the platform, such as GPS signals (when available), IMU data, compass heading measurements, and supplementary dead reckoning algorithms.

Having another measure for the quality of a given scan matching result is also important as a supplementary information for the final cost function value, which is used in this work to determine a successful scan matching result. Moreover, the calculation of this measure is not too computationally demanding, as will be shown below.

### 3.2.1 Calculation of Mean and Covariance

For a given scan matching scene, the calculation of the mean and covariance makes use of the individual contributions to the cost function, made by every point in the Reference scan. A single point-pair contribution is given by $F_i^s$ (the '$s$' stands for "signed"), shown in Eq. (3.14). This form is similar to Eq. (3.2), except here, the absolute value is removed, in order to be able to use the sign difference between the different contributions:

$$F_i^s = r''_{C_i} - r_{R_i} \tag{3.14}$$

where $r_{R_i}$ is the range of the $i^{th}$ Reference scan point, and $r_{C_i}$ is the interpolated range value from the two neighboring points in the Current scan. The signed contributions may then be used for calculating the mean, covariance, and standard deviation of the scan matching solution following Eq. (3.15):

$$\begin{gathered} S_0 = n_c \ ; \ S_1 = \sum_{i=1}^{n_c} F_i^s \ ; \ S_2 = \sum_{i=1}^{n_c} (F_i^s)^2 \\ \mu = \tfrac{S_1}{S_0} \ ; \ \sigma = \sqrt{\tfrac{S_0 S_2 - S_1^2}{S_0(S_0-1)}} \end{gathered} \tag{3.15}$$

and the covariance is simply $\sigma^2$.

#### Usage as Added Criterion

After an extended investigation, it was found that in borderline cases of scan matching, where the final cost $f$ was found to be greater than the final cost acceptance

threshold $T_F$, using the mean as a secondary acceptance test was found to produce favorable results (*i.e.*, accepting good matches that were otherwise rejected). Manual examination of those borderline cases showed that their match was, in fact, fairly accurate, except for a low number of points, which caused the final cost to break the threshold by a small margin. In these cases, the secondary test is checking for $\mu < d$, where $d$ is the occupancy grid resolution, which was chosen as a threshold for this test. Therefore, if the average of all contributions is lower than a single cell size - the scan matching is declared successful, even if the final cost $f$ may be slightly higher than it's set threshold $T_F$.

Another method to make use of the statistical properties of the matched scans is by separating the results in the $x$ and the $y$ direction, and comparing the properties. Ideally, successful scan matched scenes should have approximately the same mean in both directions. The mean properties can be extracted using Eq. (3.16):

$$\begin{aligned} S_{1_x} &= \sum_{i=1}^{n_c} F_i^s cos(\theta_{R_i}) \;;\; S_{1_y} = \sum_{i=1}^{n_c} F_i^s sin(\theta_{R_i}) \\ \mu_x &= \frac{S_{1_x}}{S_0} \;;\; \mu_y = \frac{S_{1_y}}{S_0} \end{aligned} \tag{3.16}$$

This additional statistical information may then be utilized to draw additional conclusions about the quality and the confidence level of the examined scan matching solution. This was not attempted in the current work.

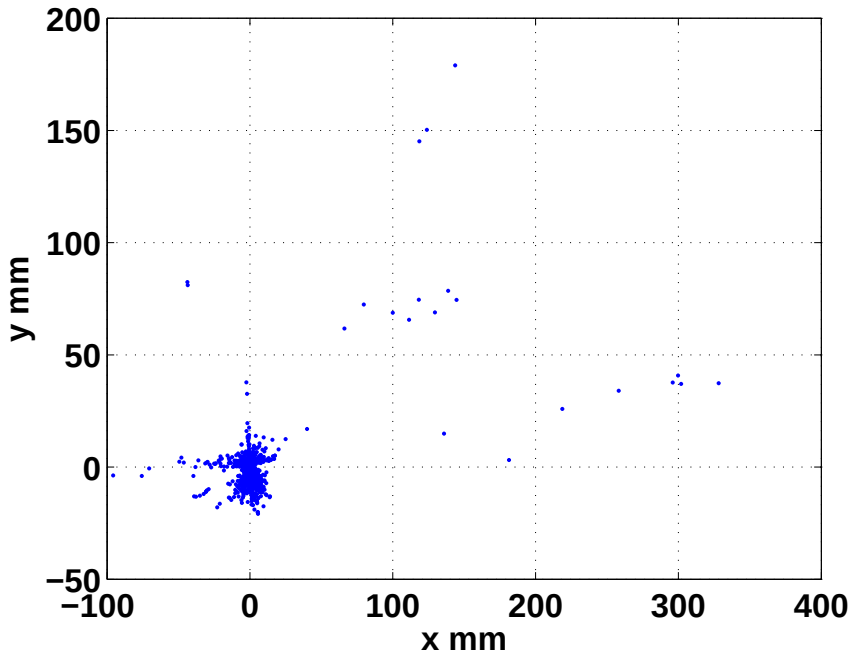



Figure 3.8: Mapping cost contributions around the origin.

An example for the above analysis is shown in Figure 3.8, for a successful scan matching scene. Figure 3.8 shows the contributions to the cost function, transformed

to the origin using their corresponding angles ($\theta_R$, the reference scan angles). It can be seen that most contributions are clustered around the origin, with only a few outliers having a large cost contribution. Figure 3.8 contains approximately 1000 points. The statistical properties for this case were calculated as: $\mu = 3.41$, $\mu_x = 2.54$, $\mu_y = -0.24$, $\sigma = 30.84$, $\sigma_x = 27.9$, $\sigma_y = 13.32$.

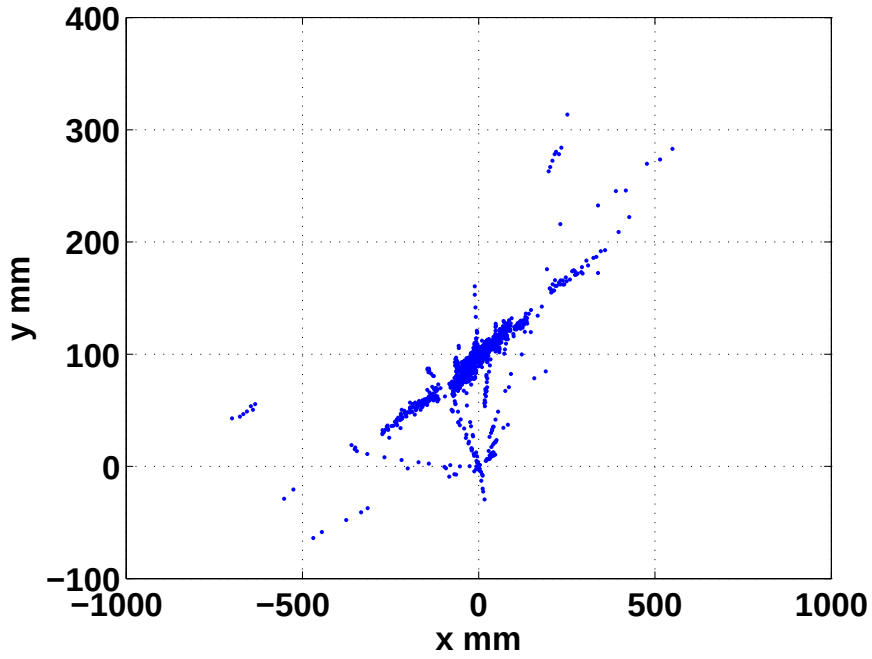



Figure 3.9: Mapping cost contributions around the origin for added $\Delta y$.

A complementary example for the same scene, but with the analysis performed using a distorted solution is given in Figure 3.9. The correct scan matching solution in this case was changed by $\Delta y = 100\ mm$. The changes in the distributions of the cost contributions is quite visible, and the statistical properties calculated for this case are as follows: $\mu = -14.6$, $\mu_x = -12$, $\mu_y = 87.8$, $\sigma = 164.57$, $\sigma_x = 131.37$, $\sigma_y = 46.67$. Those properties are, as expected, significantly larger than the ones obtained with the right scan matching solution.

Figure 3.10 presents the effect of an error in the azimuth on the statistical properties and the cost contributions distribution for that case. In this example, the right scan matching solution was used with an added $\Delta\psi = 2°$. It is quite evident in Figure 3.10 that the added rotation creates significant distortion to the distribution of the cost contributions.

The statistical properties for this case were: $\mu = 10.67$, $\mu_x = -1.28$, $\mu_y = 13.61$, $\sigma = 133.67$, $\sigma_x = 120.64$, $\sigma_y = 56.94$. As expected, the effect of the rotational error on both $\mu_x$ and $\mu_y$ is quite minor. However it is manifested in a strong way in the overall standard deviation $\sigma$ (as well as in the standard deviation in both direction). As before, it is quite clear that these properties are significantly larger and may be used for additional screening of scan matching solutions.

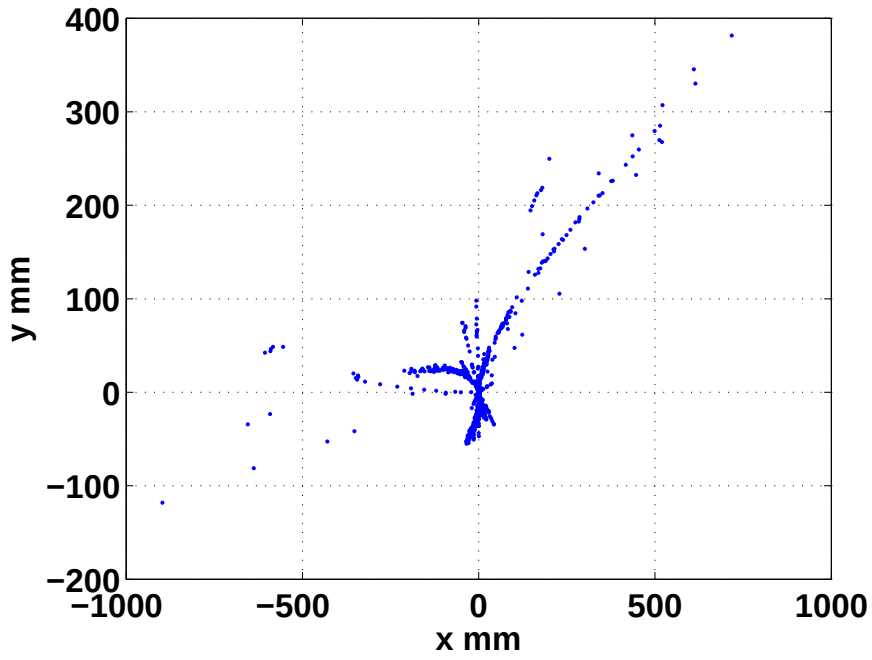



Figure 3.10: Mapping cost contributions around the origin for added $\Delta\psi$.

### Additional Usage of Statistical Properties

The statistical properties described above may also be used as a measure of confidence for the scan matching process. In a similar manner to sensors, where measurement noise and error are characterized by statistical properties such as mean and covariance. These properties are used by Kalman Filter and Particle Filter algorithms to fuse information from different sources. The same statistical information, extracted from the scan matching operation may be used in such probabilistic frameworks, to fuse scan matching output with other sensory inputs, as well as with a vehicle model prior estimates.

## 3.3 SLAM Algorithm

Operation of a vehicle in a GPS-Denied environment, with no a priori known map requires that the vehicle would have the capability of estimating it's position and orientation within a self-built map. As mentioned above, this capability is widely known as Simultaneous Localization And Mapping (SLAM). This section discusses a refined algorithm for performing SLAM in two dimensions. The SLAM algorithm receives only laser scans as sensory input, and outputs position estimates and a map in the form of an occupancy grid [32, 73]. The method employed is based on scan matching incoming laser scans against the map.

The SLAM method presented here is completely decoupled from the platform's dynamic model and thus it does not require any dynamic modeling for the platform

carrying the sensor. it relies solely on a laser scanner and a scan matching algorithm to obtain both map, position, and azimuth information. Decoupling the SLAM algorithm from the dynamic model allows its implementation on a myriad of both ground and aerial platforms including some configurations that are considered to be challenging to model (*e.g.* conventional main-rotor tail-rotor platforms, coaxial helicopters, and human-like platforms).

### 3.3.1 General Description

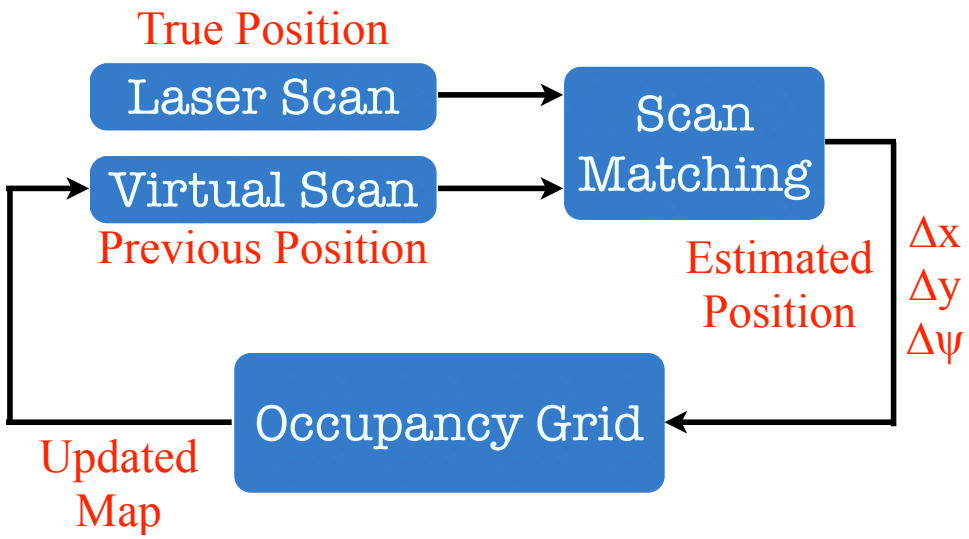



Figure 3.11: Block diagram for the SLAM process.

The SLAM block diagram is presented in Figure 3.11. The algorithm is initially given both $[x, y]$ coordinates and azimuth in the virtual map (could be arbitrary if global localization is not required). The platform then moves and generates both position information, and an occupancy grid representation of the environment incrementally. In each step, a laser scan is performed from the laser's true position, followed by a virtual scan of the occupancy grid from an estimated position. The PB-PSM scan matching algorithm is then employed on the two scans, to obtain the estimate for the current platform position and orientation. The laser measurements are then updated into the OG based on the new position estimate, and the process repeats.

Note the absence of both path planning, and decision making block as they are not considered to be part of the SLAM process by itself. In the case of targeted flight, the global path planner receives the current position at each step, plans an obstacle free path to the goal, and also check if the goal is reached (examined using a proximity threshold). The SLAM algorithm execution is terminated once the goal is reached. A detailed presentation of the coupled SLAM-Path Planning algorithm implementation is brought in Sub-Section 3.4.

The SLAM algorithm is comprised of the following steps:

i. Perform one laser scan.

ii. Perform a virtual scan from an estimated position.

iii. Carry out Scan matching between the laser scan and the virtual scan. The scan matching solution is in the form $[\Delta x, \Delta y, \Delta\psi]$, which are in the virtual scan's reference frame.

iv. Using the virtual scan's azimuth, convert the scan matching solution into the global reference frame using the following relation:

$$\begin{aligned} \Delta x_g &= \Delta x cos(\psi) - \Delta y sin(\psi) \\ \Delta y_g &= \Delta x sin(\psi) + \Delta y cos(\psi) \end{aligned} \tag{3.17}$$

This is the new position estimate in the global map (note that $\Delta\psi_g = \Delta\psi$).

v. Using the new position estimate, update the latest laser scan into the occupancy grid (see Sub-Section 2.1.1, Eq. (2.1)).

### 3.3.2 Initial Guess for the Virtual Scan

As mentioned above, the virtual scan may be taken from either the previous estimate for the position and azimuth, or from an initial guess for the current position and azimuth. The main advantage in having a good initial guess for the current platform position is providing an initial guess from where to take the virtual scan from. Ideally, one may consider taking the virtual scan from the correct current position which should result in a successful scan matching, with minimal computational effort, and maximal perimeter overlap between the two scans. However, any initial guess that is more accurate that simply using the previous pose estimate may reduce the required size of the adaptive direct search grid, which would, in turn, reduce the overall SLAM computational requirements. This section discusses the advantages and disadvantages of several options for the initial guess.

#### No Initialization

The most basic option is to take the virtual scan from the previous pose estimate. This option works quite well, and was used in many of the results presented in this work. Specifically, it works well with platforms that have a very unpredicted motion pattern (*e.g.*, aerial platforms, such as the single rotor helicopter, used in this work). Although this approach requires absolutely no computational resources, this approach has several deficiencies:

1. The search grid for the function minimization using adaptive direct search is required to be larger than the largest step magnitude that the platform may experience in a single time step. This is done in order to guarantee that the right pose solution is within the search grid. It is important to note that this restriction is true for both the planar translation search grid, and for the azimuthal rotation search grid.

2. Given the search grid size limitation, since the grid typically covers a relatively large search area, the number of grid points (search grid resolution) that is required to provide good coverage, and capture the cost function behavior in a good way is increased. If the search grid size is increased without increasing its resolution, there may be a possibility of converging the grid towards a local minima, since the global minima area would be missed by most search grid points.

### Extrapolation From Previous Motion

One way to obtain an initial guess is to extrapolate the current position from the previous motion of the vehicle. This extrapolation may be done using two (or more) of the previous pose estimates. The extrapolated current position estimate is given by Eq. (3.18):

$$\begin{aligned} &\Delta x_g = x_g|_{(s-1)} - x_g|_{(s-2)} \ ; \ \Delta y_g = y_g|_{(s-1)} - y_g|_{(s-2)} \\ &\Delta\psi = \psi|_{(s-1)} - \psi|_{(s-2)} \\ &x_g = x_g|_{(s-1)} + \Delta x_g \ ; \ y_g = y_g|_{(s-1)} + \Delta y_g \\ &\psi_g = \psi_g|_{(s-1)} + \Delta\psi_g \end{aligned} \tag{3.18}$$

where $s$ is the current step number. This extrapolation may also be performed using more than the two previous steps, however, if the SLAM estimation frequency is relatively low (approximately 2 $Hz$ in this work), the estimation may not benefit from considering more steps. The computational resources for this approach are very minimal. However, although this approach was found to be very helpful in straight motion through corridors, it was considerably less effective in relatively fast maneuvering around corners, and in fact caused some failures in some cases.

### Using Additional Sensors

Although the work presented here made use of a single sensor (laser scanner), if additional sensor are available on board the platform, they may be used for providing the algorithm with an estimated current pose. The requirement for additional sensors adds some complexity to the platform, and certainly has an effect on the cost. The computational resources associated with using the measurements of all sensors is not considered to be significant, although it is not negligible. Added sensors also require additional communication capabilities (channels, frequencies, antennae, etc.).

**Wheel Encoders:** wheel encoders are quite a common sensor to place on a wheeled ground platform. The encoder counts the number of rotations of the wheel to which it is attached. Typically, wheeled ground vehicles are relatively easy to model, as shown in Sub-Section 1.2.1, for a differential drive robot. The platform model,

along with the wheel encoder measurements may be used to produce an initial guess for the platform's position and azimuth. This process is known as "Dead Reckoning".

**Inertial Measurement Unit:** if an Inertial Measurement Units (IMU) is available, the measurements may be integrated over time, for each step. The IMU-based dead-reckoning is known to drift over time. However, in this case, it is proposed to integrate over very short time periods, from the end of the previous step (when the position is updated) to the beginning of the current step (when the current laser scan is taken), which is typically of the order of a single second.

**Global Positioning System:** in the case of outdoor operations, GPS signal may not be available at all times. Moreover, the GPS signal rate is typically available on the order of 0.5 $Hz$, while higher-rate vehicle pose estimates may be required from the SLAM algorithm. Moreover, the GPS measurement, although external to the vehicle in a global frame, is not perfectly accurate, and typically has an error of the order of $O(1\ m)$.

The available GPS signals may be very useful for providing initial estimates for the SLAM algorithm, in the form of vehicle pose estimates for the current step. This may also be achieved using intermittent GPS measurements that may be available based on reception in the outdoors.

**Compass** a compass may provide an estimate for the vehicle's azimuth, in a global reference frame. These measurements may easily be converted to the platform's reference frame, using the initial position, in relation to the first compass measurement. Although a compass does not provide any information about the vehicle pose, it may significantly reduce computational complexity by providing a good initial guess for the azimuthal direction. If a compass sensor is available, the adaptive direct search may be adjusted, with significantly less resources required for the a azimuth estimation (since the search window size will be reduced).

### Using a Vehicle Model

A model for the platform's dynamics is considered to be a set of mathematical equations, which enables the prediction of some or all of the platform's states, as a function of the inputs and additional parameters. The equations are typically given in the form of a time derivative of a set of platform states, as in Eq. (1.1), where the right hand side may be either linear or non-linear.

Various levels of modeling may be used with different set of simplifying assumptions. For example, modeling a wheeled platform typically assumes no slippage between the wheels and the ground, while modeling quad-rotors typically assume that the thrust of each rotor is controlled by direct RPM inputs (ignoring inflow effects, climb and descent velocities, edgewise velocities, and inter-rotor effects).

Typically, the commands input to the vehicle are also known (although there is always some added input noise). Using the vehicle model and the known commands, a prior belief for the vehicle states may be derived. An example is given here for a

general dynamic model. The model is given by Eq. (3.19):

$$\begin{aligned} \dot{\underline{x}} &= A(t)\underline{x} + B(t)\underline{u} \\ \underline{y} &= C(t)\underline{x} + D(t)\underline{u} \end{aligned} \tag{3.19}$$

where $\underline{x}$ is the state vector, $\underline{u}$ is the input command vector, and $A(t)$, $B(t)$, $C(t)$, and $D(t)$ are the dynamic model linearized matrices. Now, even without any measurement $\underline{y}$, time integration may be used to calculate the current state vector $\underline{x}$ using the known inputs $\underline{u}$. This way, an initial guess for the SLAM algorithm may be found even without any additional on-board sensors. This pose estimate is also known as a "prior", as it does not reply on any measurement update.

If the system is given in discrete time form as in Eq. (3.20):

$$\begin{aligned} \underline{x}_{k+1} &= F_k\underline{x}_k + G_k\underline{u}_k \\ \underline{y}_k &= H_k\underline{x}_k \end{aligned} \tag{3.20}$$

where $k$ is the time step number, $F_k$ is the discrete time state transition matrix at time $k$, $G_k$ is the input matrix at time $k$, and $H_k$ is the measurement matrix at time $k$. Hence given these matrices, the vehicle pose and azimuth at any time $k$ may be calculated. The solution for $x_k$ is given by Eq. (3.21):

$$\begin{aligned} x_k &= \Phi(k,0)x_0 + \sum_{j=0}^{k-1} \Phi(k,j+1)G_j\underline{u}_j \\ \Phi(k,j) &= \begin{cases} F_{k-1}F_{k-2}\dots F_j & k \geq j-1 \\ I & k = j \\ 0 & k < j \end{cases} \end{aligned} \tag{3.21}$$

where $\Phi(k,j)$, is the discrete time state transition matrix from time $j$ to time $k$, and $I$ is the identity matrix. Note that $k$ may represent a smaller time step as compared to the SLAM time step. This would be desired in order to receive a higher accuracy initial guess.

The advantage in using a platform model for the above purpose is, once again, providing an initial guess for the platform current position, from where the virtual scan is taken. This may reduce the required size of the adaptive direct search grid, which would, in turn, reduce the overall SLAM computational requirements. The computational cost of calculating an initial guess using the above described approach is considered to be relatively low. However, not all platforms may be easily modeled, and the commands given to the platform may not always be known (*e.g.*, a walking person platform or a single rotor helicopter).

#### Using a Vehicle and a Sensor Model

Assuming both a vehicle model and some sensor model are known, the information that be obtained may be fused together using a Kalman filter algorithm which may produce an improved estimated state vector of the platform, as compared with the above two examples of a sensor measurement alone or a vehicle model-based prior estimate alone.

Quite a common example may be the case where an IMU measurement is available (a very common sensor on aerial platforms). The Kalman filter may then fuse the IMU measurements with the vehicle model using the measurement equation parts of Eq. (3.19), and Eq. (3.20). The fused information may yield a significantly improved initial guess as compared with the previous two options (depending on the modeling quality, and the associated sensor accuracy).

Using a Kalman Filter requires a mathematical model for the sensor behavior as well. For IMU's, compasses, wheel encoders, and GPS systems, the noise that is associated with the measurements may typically be described as Gaussian noise, which is typically easy to represent in the sensor's probability density function.

The main advantage in combining a platform model with sensor models is the significant improvement of the vehicle state prediction, that is associated with a bayesian filtering approach. As in the previous approaches, this may significantly reduce the computational resources for the SLAM algorithm by reducing the required size of the adaptive direct search grid.

### 3.3.3 Isolated Point Filter

In order to handle relatively small scan matching solutions, and possible spurious errors, the map is scanned every few steps for isolated occupied cells. Assuming several map updates were made, these occupied cells, being isolated from their immediate occupied surroundings, may be treated as an error in the map, and removed (*i.e.* occupancy value is reset to zero). The map sweep examines each occupied cell for the following condition:

$$\sum_{i-n_s}^{i+n_s} \sum_{j-n_s}^{j+n_s} C_{k,l} < T_O \tag{3.22}$$

where $i$ and $j$ are the indices of the examined occupied cell, $k$ and $l$ are running indices that scan the immediate surrounding cells, $n_s$ is a parameter for the size of the inspected area, $C_{k,l}$ is the current occupancy of the $[k,l]$ cell, and $T_O$ is the occupancy threshold.

Typically, the value used was $n_s = 1$, which means only neighboring cells were summed. The occupancy threshold parameter was set for $T_O = 1$, which implies that is all the cells' occupancy together reaches more than a unit value - the cell's occupancy is not reset, and it is kept in the map. The two set values is intended to minimize the examined surroundings. An example of an occupancy grid map after the isolated point filter is applied is shown in Figure 2.3. The isolated point filter is employed every $n_{IPF} = 40$ steps, or in cases of scan matching failure.

### 3.3.4 Computational Complexity

Unlike several other methods, where the complexity grows with the number of observed obstacles [1, 8], in the current approach, the complexity is kept constant throughout the mission. The complexity in the proposed algorithm is comprised of the number of ray casting operations, and the number of cost function evaluations required to perform a successful scan matching step. Both are not independent of the number of obstacles observed thus far. Therefore, the proposed algorithm is not limited by the scenario size, or the number of obstacles. It is capable of mapping highly clustered as well as sparse environments of various sizes. The only limitations come from the size of the available memory, and the computational resources which determine the frequency of the state estimates.

## 3.4 Coupled Path Planning-SLAM Algorithm

The combined algorithm which includes the SLAM and the A* path planner is described using a flowchart, presented in Figure 3.12. The Start and Goal are the main user-defined inputs (besides the algorithm's parameters, described in this chapter). Initialization includes memory allocation, arrays initialization, turning on the laser scanner, graphics settings, etc.

The algorithm then begins by taking the first laser scan. Since there is no information yet in the map, no virtual scan is carried out, and the first laser scan is updated into the map using the user-defined start position. Since the goal has not been reached at the first step, the "Goal Reached?" condition returns "no", and the A* algorithm plans a path to the goal, considering the information that is currently available in the occupancy grid map. This is followed by some platform motion to a new position. The motion may be user-controlled, or autonomously executed by the vehicle.

After the first step, all the subsequent steps start with a laser scan, and since the occupancy grid now contains information, a virtual scan may be taken (from either the previous position or an initial rough estimate of the current position). The two scans serve as the input to the PB-PSM scan matching algorithm. A successful scan matching is identified by a final cost that is lower than the set threshold. If the scan matching was indeed successful, the new laser scan is updated into the map using the current position estimate from the scan matching. However, if the scan matching failed, the laser scan is not updated into the map, as the estimated current position is inaccurate. Instead, the algorithm invokes the isolated point filter to try and clean the map from erroneous features (floor scans, previous wrong matches, etc.).

The isolated point filter is employed on the occupancy grid every $n_{IPF}$ steps, by simply checking the current step number (denoted by $s$) for being an integer multiple of $n_{IPF}$. The algorithm then checks if the new position is within the proximity threshold to the goal position. If the goal has been reached, the algorithm terminates successfully, and if the goal has not been reached, a new path is generated by the A* planner, and the platform continues to move.

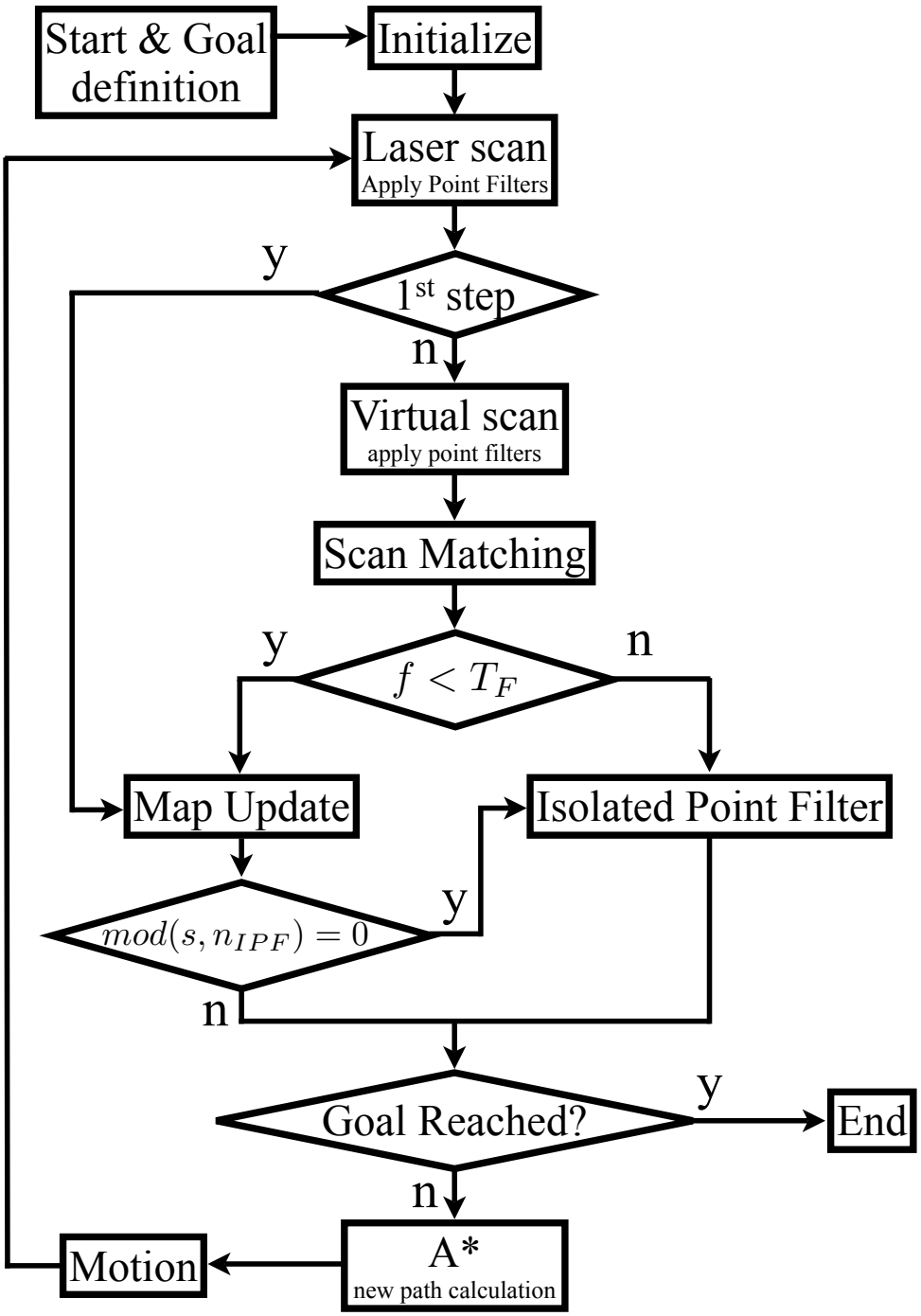



Figure 3.12: The complete algorithm, coupling the SLAM and path planning algorithms.

**General Comments**

1. Scan matching success may also be defined based on statistics, as described in Section 3.2. If this option is used, it follows the check for $f < T_F$.

2. The isolated point filter is typically invoked every few dozens of steps. In this work, a value of $n_{IPF} = 40$ was used throughout.

3. Although the isolated point filter is invoked in case of a failed scan matching operation, this does not apply in the first few steps, since the map contains very little information. Therefore, many points may be considered as isolated, and be wrongfully removed. In the current work , the isolated point filter is only employed after at least 5 steps have been carried out.

4. Checking wether the goal has been reached is done using a threshold of 0.5 $m$.

## 3.5 Assumptions and Limitations

Several assumptions are considered by the proposed algorithm. The description and importance of the various assumptions is brought below. Some of the assumptions are more critical than others, and therefore details about how each assumption may be violated are also discussed. Examples for failure modes are given in Sub-Section 5.3.5.

### 3.5.1 General Assumptions

#### Two Dimensional Motion

The algorithm proposed in this work assumes a two dimensional planar motion, where the platform may move in two directions in the plane, or rotate in the azimuthal direction. The resulting estimates are of the both the $[x, y]$ motion, as well as the azimuthal angle $\psi$. Moreover, the sensor used in this work is a two dimensional laser range scanner which physically scans the environment using a planar sheet of laser beams. Therefore, a three dimensional motion will not be captured accurately. Nevertheless, as with most experimental platforms, the motion is rarely purely two dimensional, and in particular, a flying helicopter is not likely to remain level at all times. A discussion about this assumption and the associated limitations is provided below.

#### Small Pitch And Roll Angles

During the experiments, pitch and/or roll motions may change the measured objects from the surroundings. For a 10 $m$ range, a pitch angle of 10° would change the measurement by approximately 15 $cm$. Changes of this magnitude are still comparable with the laser noise itself, and thus do not pose a significant difficulty. However, when the pitch and/or roll angles are large enough to cause the laser to scan the floor, the shape of the laser scan changes significantly and no longer represents the mapped environment. A scan that contains part of the floor is very likely to produce a failed scan matching results. As such, the scan will not be updated into the map, and the assumption in this case is that the next scan will be a valid one, so the SLAM process may be continued. If the scan matching process is successful, and the

scan data is inserted into the map, the cost-contribution elimination threshold (see Sub-Section 3.1.1) may help remove the wrongly inserted data.

#### Elevation Changes

Another possible violation of the two dimensionality assumption is a change of altitude. Typically, an environment may contain obstacle of various heights, and so the laser would produce different scans of the environment, based on its current altitude. The assumption is that the relative portion of the scan that is affected by an altitude change is small. An example may be a trashcan in a hallway, where the relative portion of the trashcan as compared with the hallway walls is small. Scans that include the trashcan and scans that are conducted above it may still have enough feature overlap in them, to allow a successful scan matching result. It is assumed that such objects will either be cleared of the map by the isolated point filter (see Sub-Section 2.1.1), or in the case that they are scanned more than once - they remain in the map, but do not pose a difficulty for the follow-on scan matching processes (due to their relative small size.

#### Information Rich Laser Scans

Generally, the laser scans need to contain enough information to allow successful scan matching. This definition translates into having at least one major obstacle in the scan, for each of the two dimensions. Typically, corners which comprise of obstacles in two opposing directions fulfill this requirement (not necessarily right angle corners). A typical scan matching failure mode may appear when either scan does not possess this feature. A typical example may be a scan of a long corridor, that contains only two parallel walls, with no feature to help match the longitudinal direction.

### 3.5.2 Platform Speed Limitations Analysis

During all the experiments, the laser sensor collected data while the platform is in motion. Since the set of range measurements is collected in a sequential fashion, it is likely that scanning while the platform is in motion will differ from a purely static scan. If the laser's scan speed is significantly faster compared to the platform's turn rate – the scan measurements are less affected by the platform's motion, and thus the laser signature of the environment is closer to static scan conditions. For the results presented in this thesis, the platform's highest rotational velocity was 20 $deg/sec$, while the laser rotates at 14400 $deg/sec$. Therefore, the assumption is that laser scans need no correction for platform motion. This assumption is largely used by the robotics community. The current algorithm assumes that both planar and rotational platform motions are slow compared to laser scan speed. This section presents a detailed analysis of the limitation as a results from the laser scanner frequency.

Generally, a scan of an object would naturally result in a different set of range values if the laser is static or moving at a certain velocity. Moreover, the distortion

depends on the following parameters:

1. Laser scanner velocity
2. Laser scanner frequency (scans per second)
3. Laser scanner motion direction in the plane
4. Laser scanner motion direction in the azimuth

The above may be modeled under several assumptions, to achieve some bounds for the highest platform velocity that may be used in a certain environment, using a given laser scanner. The assumptions include:

1. Excluding laser noise.
2. A single laser measurement is carried out instantly.
3. Smooth platform motion at a constant velocity.

An equation for a set of range measurements of a straight wall is analytically developed below, including the effects of platform velocity, and motion direction.

### Model Problem

The scenario considered is described in Figure 3.13. A laser scanner is moving towards a vertical wall at a velocity $V$, and direction $\alpha$. For simplicity, the following assumptions are made:

1. The laser azimuth is fixed, and it is constantly pointed "up" (North), while moving towards the wall.
2. The laser's field of view begins with a horizontal beam pointed to the right (East), represented by $\theta_L = 0°$.
3. The laser has a 360° field of view, and the measurements are equally distributed around it.
4. The laser introduces no noise into the range measurement, and so any measurement returns the true range to the wall from the current laser's position.

The above assumptions allow the analysis to focus on the error as a result from the platform's motion only.

The laser scanner location is given by Eq. (3.23) as:

$$x_1(t) = Vcos(\alpha)t \ ; \ y_1(t) = Vsin(\alpha)t \tag{3.23}$$

where $t$ is the time, and $x_1$ and $y_1$ are the laser scanner coordinates in the $x$ and $y$ directions, respectively. Every $i^{th}$ laser measurement has a range $r_{L_i}$ and a bearing $\theta_{L_i}$. We assume that every laser measurement is taken instantly, and therefore this analysis

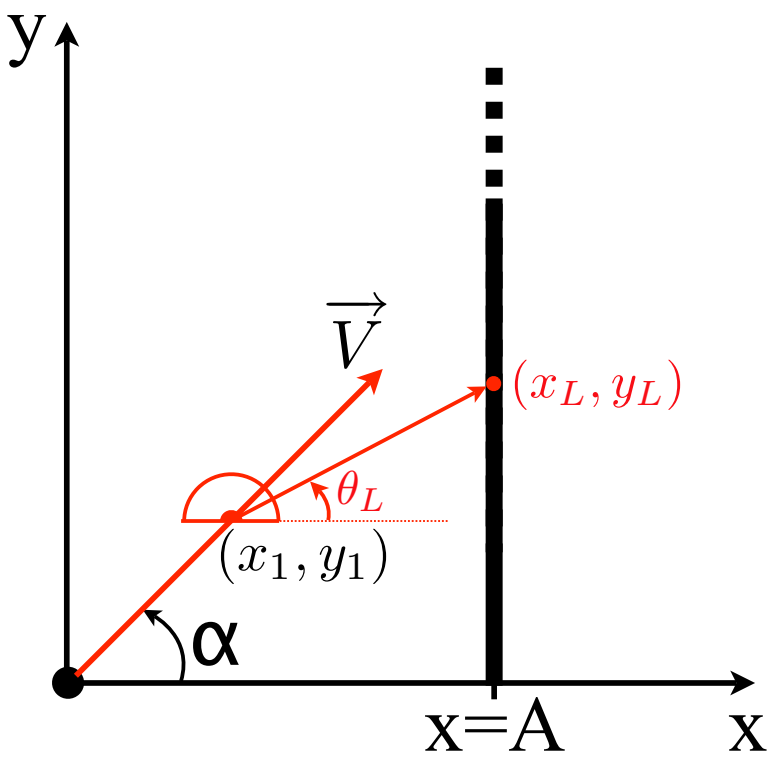



Figure 3.13: Laser scanner moving towards a wall.

only considers the time it takes the laser's mirror to rotate between measurements, which is given by Eq. (3.24):

$$\Delta t = \frac{1}{f\left(\frac{FOV}{\Delta\theta}\right)} \tag{3.24}$$

where $f$ is the laser's scan frequency, FOV is the laser's field of view, $\Delta\theta$ is the laser scanner's angular resolution. Therefore $\left(\frac{FOV}{\Delta\theta}+1\right)$ is the number of laser points in a complete scan. The bearing $\theta_L$ changes with time as the scan progresses according to Eq. (3.25):

$$\theta_L(t) = \Delta\theta\frac{t}{\Delta t} \tag{3.25}$$

Since the wall in this example is vertical, the true wall's $x$-coordinate is always $x = A$ (see Figure 3.13). The laser measurement $y$ coordinate may be found by Eq. (3.26):

$$y_L = y_1 + tan(\theta_L(t))(A - x_1) \tag{3.26}$$

The range measurement to the $i^{th}$ point may be found using Eq. (3.27)

$$\begin{aligned} r|_{\theta=\theta_L} &= \sqrt{(\Delta x)^2 + (\Delta y)^2} \\ \Delta x &= x_L - x_1 = A - x_1 \\ \Delta y &= y_L - y_1 = y_{(x=A)} - y_1 = tan(\theta_L)\Delta x \\ r|_{\theta=\theta_L} &= \sqrt{tan^2(\theta_L)(A-x_1)^2 + (A-x_1)^2} \\ r|_{\theta=\theta_L} &= (A-x_1)\sqrt{tan^2(\theta_L)+1} \end{aligned} \tag{3.27}$$

The time varying range measurement for a given $\theta_L$ is therefore a function of $Vcos(\alpha)$, the component of the velocity in the direction normal to the wall, and of $f$, the laser scanner frequency. Therefore, $r(t)$ may be written as follows:

$$r(t) = (A - Vcos(\alpha))\sqrt{tan^2(\theta_L)+1} \tag{3.28}$$

Note that all the range measurements are treated as if they were all taken from the position at $t = 0$ (*i.e.*, all ranges are assumed to be take instantaneously). The coordinates of all laser measurement, as a function of time may then be found using Eq. (3.29):

$$\begin{aligned} x_{L_i}(t) &= x_1|_{t=0} + r(t)cos(\theta_L(t)) \\ y_{L_i}(t) &= y_1|_{t=0} + r(t)sin(\theta_L(t)) \end{aligned} \tag{3.29}$$

One way to defined the error between the true wall and the scan result may be as follows:

$$\varepsilon_L = \frac{r_{true} - r_L}{r_{true}} \tag{3.30}$$

this error definition requires some reference, as it grows with the progression of the scan (*i.e.*, ranges that are measured later in the scan will have a large error, simply because the laser moved a greater distance). We therefore define an error for the above model problem based on the error for the range measured at $\theta_L = 45°$. For that particular laser angle, the error becomes:

$$\varepsilon_{L_{45°}} = \frac{A\sqrt{2} - (A - Vcos(\alpha)t_{45})\sqrt{2}}{A\sqrt{2}} = \frac{Vcos(\alpha)t_{45}}{A} \tag{3.31}$$

where $t_{45}$ is the time it takes the laser to take the measurement at $\theta_L = 45°$ (assuming the first measurement was at $t_0 = 0$). Hence, there is a dependency on the scan frequency since $t_{45}$ decreases with increasing scan frequency according to:

$$t_{45} = \frac{1}{f}\frac{45°}{360°} = \frac{1}{8f} \tag{3.32}$$

Now the error $\varepsilon_{L_{45^\circ}}$ may be expressed as a function of the velocity component normal to the wall and the laser scan frequency as follows:

$$\varepsilon_{L_{45^\circ}} = \frac{V_\perp}{8fA} = \frac{V_\perp}{f} C_{45}$$

$$V_\perp = V cos(\alpha) \tag{3.33}$$

$$C_{45} = \frac{1}{8A}$$

where $V_\perp$ is the velocity component normal to the wall (which causes the distortion), and $C_{45}$ is a constant that is directly associated with the examined laser point at $\theta_L = 45^\circ$. It can be seen that the error relative to the measured length increases linearly with the velocity component normal to the wall, and is inversely dependent on the laser scan frequency. As expected, a faster scanning laser range finder is highly desired in high speed SLAM applications.

**Wall Representation**

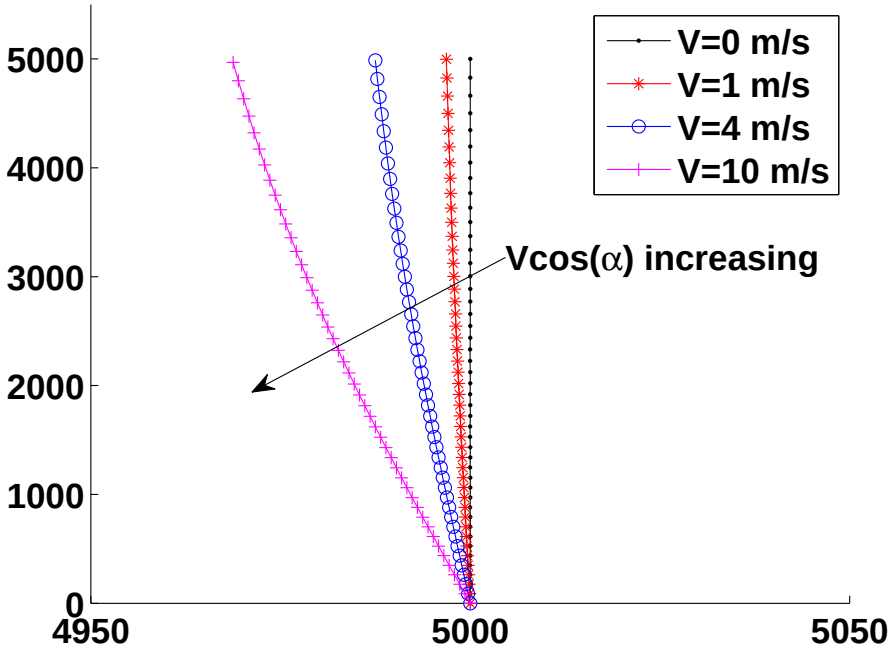



Figure 3.14: Distorted wall as a function of platform speed.

Since all laser measurements are considered as if they were all taken from the same position (given by $(x_1|_{t=0}, y_1|_{t=0})$, the actual shape of the registered wall is distorted. An example is given in Figure 3.14, based on the schematics presented in Figure 3.13. The true vertical wall is the result of a completely static case (black line with points $V = 0\ m/s$), and it is compared with three platform velocities of 1 $m/s$, 4 $m/s$, and

10 $m/s$. The motion direction was kept constant at $\alpha = 45°$. In this example, only the first 45° of the scan is shown, for clarity. As the velocity is increased, the wall appears to be more bent towards the laser origin of motion. This results is expected since the range values decrease as the laser approaches the wall.

**Effects of Platform Speed and Scanning Frequency**

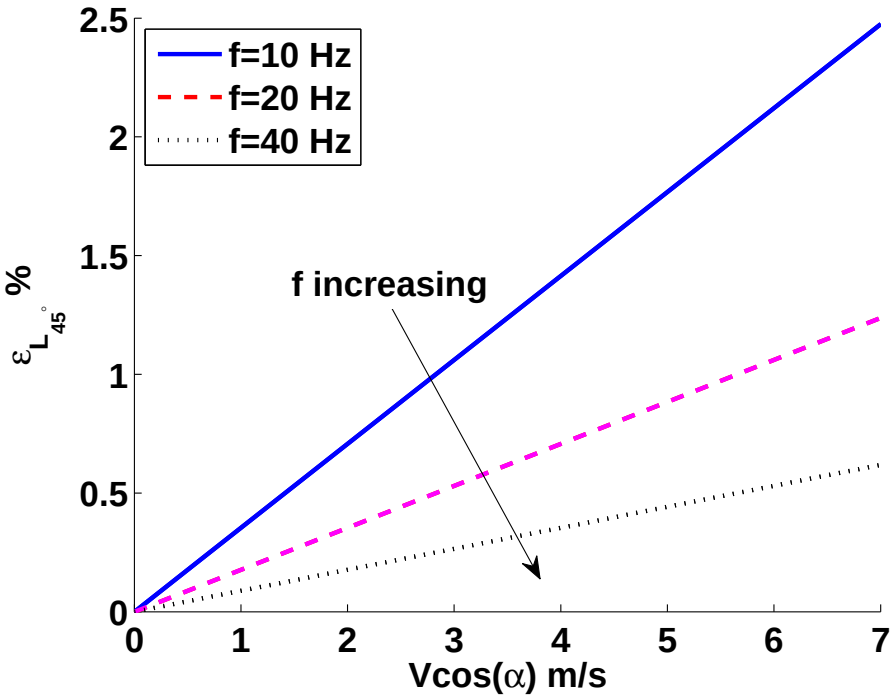



Figure 3.15: Relative scan error as a function of platform speed.

Figure 3.15 presents the relative scan error $\varepsilon_{L_{45^\circ}}$ in percents, as a function of $V_\perp$, the velocity component perpendicular to the wall. The highest velocity in this case was again $V = 10\ m/s$, with a constant motion azimuth of $\alpha = 45°$. As it was shown above, the error increases linearly with $V_\perp$, while higher frequencies result in a significantly reduced relative error.

It is important to note that a laser scanner's error in measuring range is typically of the order of 0.5% of the measured distance. Therefore at a scan rate of $f = 40\ Hz$, the error that is introduced due to platform motion is smaller than the inherent measurement error for almost the entire velocity range. Moreover, SLAM capable platforms, in particular aerial platforms, typically move at relatively slow velocities of less than 1 $m/s$. Therefore a scanner with a scan rate of $f = 40\ Hz$ may be considered to be quite accurate for this range of velocities.

To conclude, error as a result from platform motion is important, and the representation of the measured objects in the occupancy grid forms the total error (excluding laser noise, as mentioned above). Reducing the final representation error of objects

which are scanned while the platform is in motion is important for the final map accuracy. To the best knowledge of the author, there is no correction for the suggested error due to platform motion in the available literature.

### 3.5.3 Dynamic Environments

The algorithm is primarily intended for a static (motionless) environment. This means that all the objects in the field of view of the laser are all assumed static, and may all be used to form the map, since they are not expected to move. Although some scenarios included some moving objects (*e.g.* walking people caught by the laser scanner), the inclusion of those objects in the map may prohibit a successful scan matching solution.

#### Dynamic Environment Description

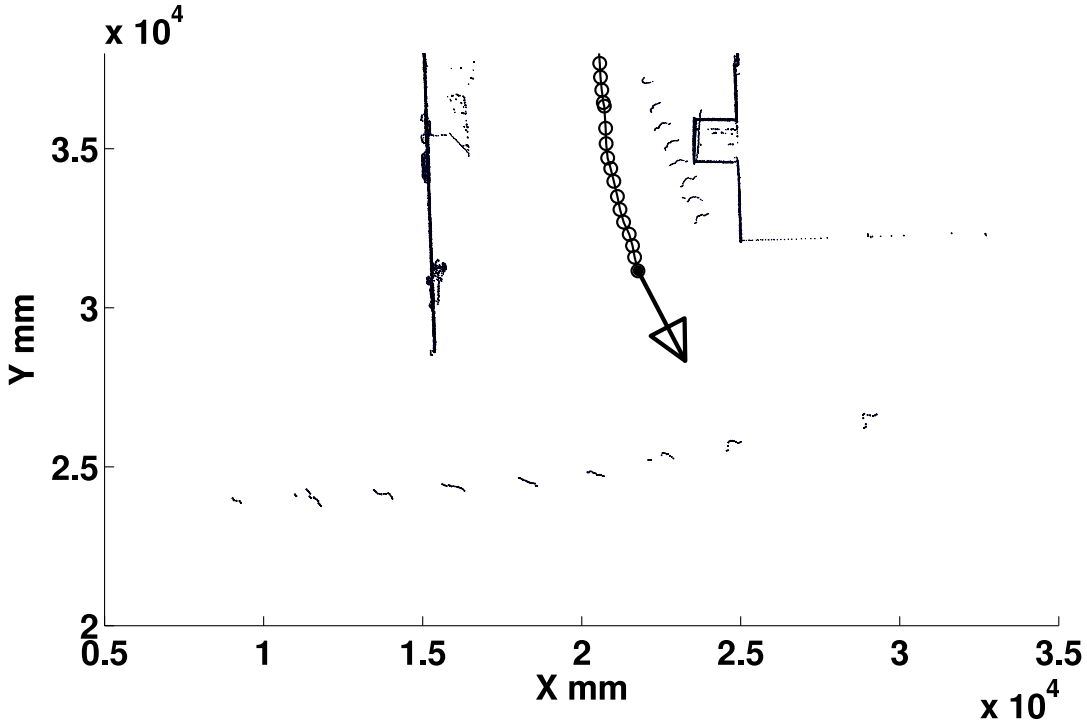



Figure 3.16: Example of moving objects caught by the laser scanner, and added into the map, leaving a trail of occupied cells. Traveled path is marked with circles, heading is marked with an arrow. A person was walking on the left hand side, and a car was driving in front of the laser.

An example for a dynamic environment, and it's representation in the occupancy grid map, is presented in Figure 3.16, where both a moving car and a walking person were captured during an experiment. At each step, the laser scans were recorded into the OG map, leaving a trail of occupied cells along the entire car and person's trajectories.

A possible solution to this problem may be the inclusion of an algorithm that identifies the portions of the laser scan that contain moving objects, and excludes them from both the scan matching process and the map update step. This may also be achieved with additional sensors such as a camera (at the expense of additional computational resources). An analysis of the effects that moving objects may have on the scan matching process is presented below.

**Effects on Scan Matching**

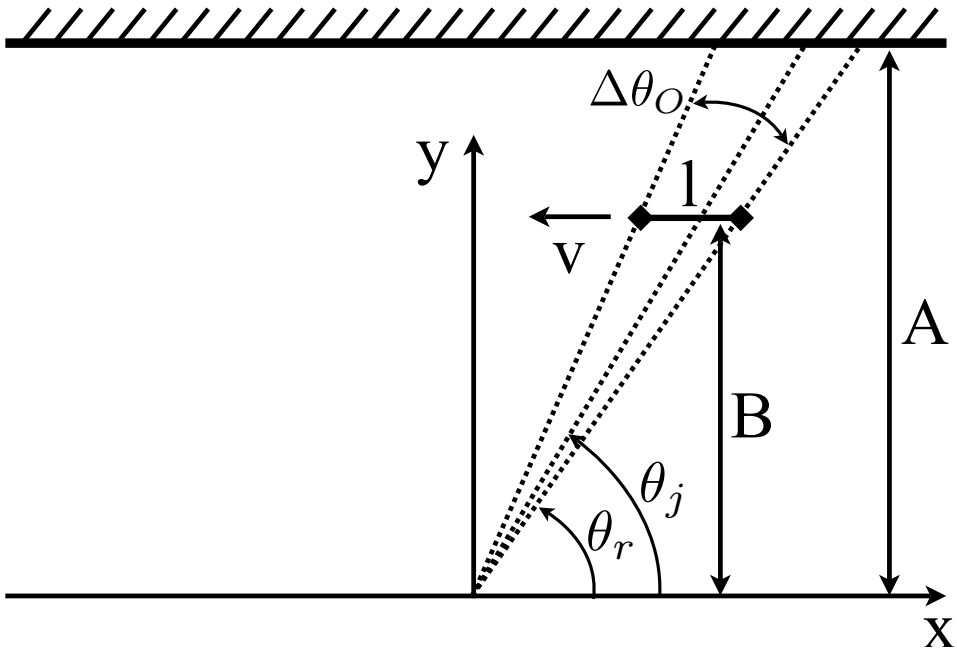



Figure 3.17: Schematics of moving object effect analysis.

A schematics for the analysis of moving objects on the final cost function is presented in Figure 3.17. The analysis considers a single object of size $l$, moving horizontally from right to left, at a velocity $V$, and a vertical distance $B$ from the laser scanner's origin. A horizontal static wall serves as the background, at a distance $A$ from the laser scanner's origin. The analysis assumes an infinitely fast laser scanner frequency (*i.e.*, the laser scans instantaneously), and a continuously successful scan matching, when excluding the moving object. The values of $A$ and $B$ are kept fairly close (200 $mm$ difference), so that the added contributions will not be filtered out by the threshold $T_E$ (see Sub-Section 3.1.1).

The analysis examines how many laser beams hit the object, and the added contribution of each beam to the cost is calculated. Since the beams hit the object and not the background wall - the contribution of those beams is, in fact, an undesired side effect of the object occluding the static background wall.

For simplicity, the analysis starts when the moving object is located at $x|_{(t=0)} = B$, and therefore the ray from the laser's origin to the object's right-most point is

$\theta_O = 45°$. The angle formed between the two lines from the laser's origin to both object's end points is given in Eq. (3.34):

$$\begin{aligned} \Delta\theta_O &= tan^{-1}\left(\frac{B}{x(t)-l}\right) - tan^{-1}\left(\frac{B}{x(t)}\right) \\ x(t) &= x|_{t=0} - Vt \end{aligned} \tag{3.34}$$

where $x(t)$ is the 'x'-location of the object's right-most point. This angle defines the number of laser beams that hit the object, based on the scanner's angular resolution $\Delta\theta$. The cost contribution for the $j^{th}$ beam that hits the object is therefore given by Eq. (3.35):

$$\begin{aligned} F_j &= \frac{A-B}{sin(\theta_j)} \\ \theta_j &= \theta_r - (j-1)\Delta\theta \end{aligned} \tag{3.35}$$

where $\theta_j$ is the angle of the current beam, $\theta_r$ is the angle of the right-most beam that hits the moving object, and $\Delta\theta$ is the laser scanner's angular resolution. The summation of these contributions, normalized by the total number of cost-contributing points from the Reference scan $n_c$, yields the total undesired contribution of the moving object to the cost function. A value of $n_c = 1080$ was used in this analysis, to reflect the scanner that was used in this work. The total contribution $\Delta f$ is expressed in Eq. (3.36):

$$\Delta f = \frac{1}{n_c}\sum_{j=1}^{n_O} F_j \tag{3.36}$$

where $n_O$ is the number of beams that hit the moving object.

As the object moves to the left, the number of laser beams that hit the object changes, as well as their respectable contributions. The total undesired contribution $\Delta f$ is plotted in Figure 3.18, as a function of the object's position. The figure shows the effect of the object's distance from the scanner's origin with the static wall located 200 $mm$ above the object horizontal trajectory. The velocity in this case is $V = 500\ mm/s$, with an object size of $l = 1000\ mm$. The jagged edges are due to the sudden changes in the number of laser beams that hits the object. The values of $A$ and $B$ are between 5000 $mm$ and 10000 $mm$, and as expected, the closer the objects is to the scanner - the bigger the undesired contribution becomes. In fact, with the current laser parameters, when the object is at a distance of approximately 5 $m$, the undesired contribution may be large enough to cause the scan matching to fail (with a threshold of $T_F = 10$).

For comparison, the red lines represent the cost contribution magnitude of the laser scanner's noise in measuring similar distances with the same number of points, assuming maximum laser noise (each red line style corresponds to the equivalent in black). As evident, the moving object typically has a considerably larger effect on the final cost, as compared with the laser scanner measurement noise.

The above procedure was repeated for a wall distance of A=10000 $mm$, but with

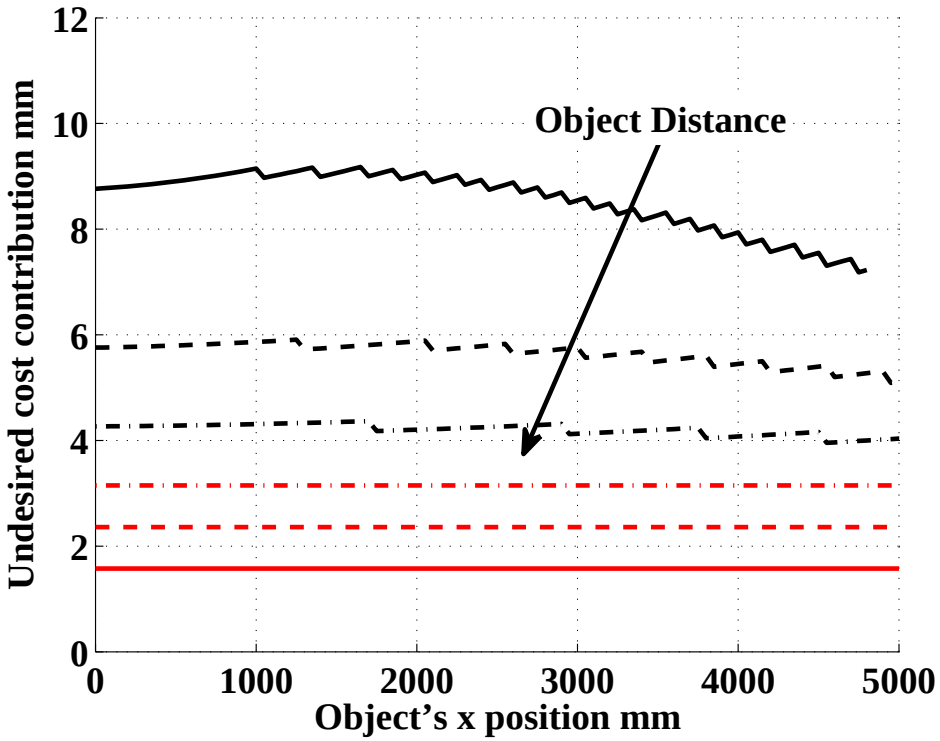



Figure 3.18: Effect of moving object distance from laser scanner's origin.

varying object size, Figure 3.19. As expected larger objects cause a higher undesired contribution to the cost, with an object size of $l = 2000\ mm$ having a relatively large undesired contribution, that may fail the scan matching, as discussed above.

This analysis shows that in some cases, given an object that is captured by a relatively low number of laser beams, where each beam's contribution to the final cost is also relatively small, the scan matching process may succeed, although the environment is not entirely static. Furthermore, if a successful scan matching occurs, and the moving object is updated into the map, it may be removed later by the isolated point filter, as it's appearance may very well be represented by a group of isolated pixels in the occupancy grid. The complete algorithm presented in this thesis, may therefore be capable of coping with mildly dynamic environments.

The red line in Figure 3.19 is presented for comparison, and shows the maximum increase in the final cost due to the laser scanner's noise, with the same number of points, at the same distance of 10000 $mm$ (identical to the dot-dash red line in Figure 3.18). It shows that a moving object size contribution typically has a stronger contribution to the final cost. As mentioned above, the algorithm may be able to handle some smaller moving objects, which don't cover a lot of the laser scanner's field of view.

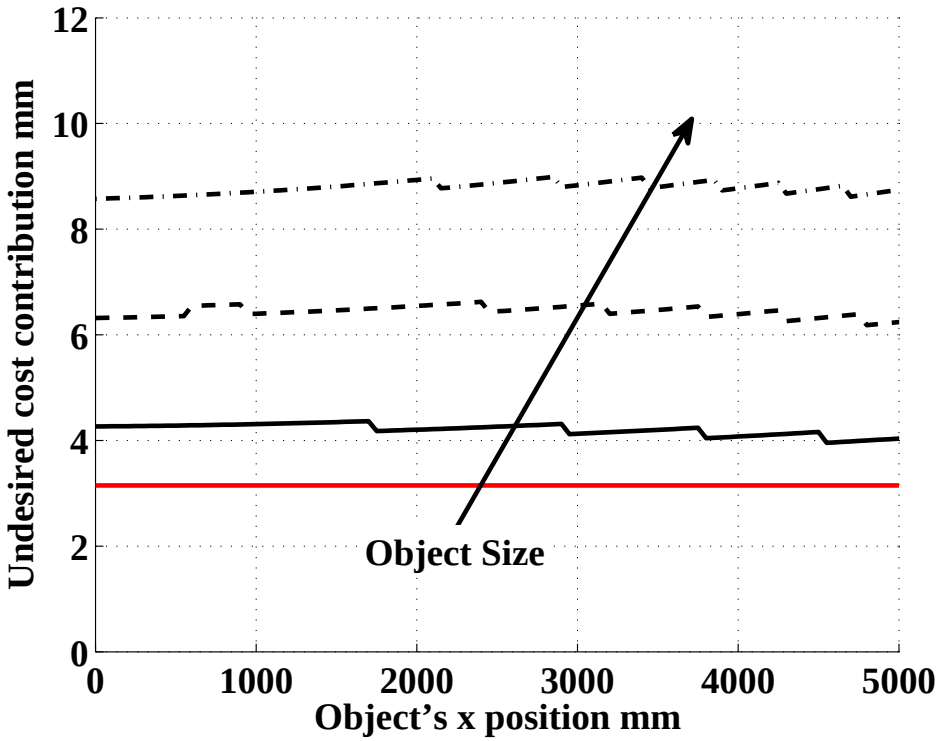



Figure 3.19: Effect of moving object size.

### 3.5.4 Object Detection Limitations

Several lower-bounds on detectable obstacle sizes are presented in Figure 3.20, using the properties of the Hokuyo UTM30LX laser scanner. The vertical axis shows obstacle size while the horizontal axis shows the range to the obstacle. The two green vertical lines represent the minimum and the maximum range of measurements that the scan matching algorithm considers. The maximum range is governed by the laser scanner used, while the minimum range is decided based on the platform size (400 $mm$ is approximately half the size of the aerial platform).

The resolution of the occupancy grid sets another lower bound, since no object that is smaller than that resolution can be adequately represented, and thus would not be represented in any virtual scan as well. The orange line shows this lower bound, which naturally does not change with the range to the obstacle.

The black line represents the laser's beam width at various ranges or $Rd\theta$. This was estimated experimentally by measuring obstacles, and estimating the beam width using the detectable area of a clear obstacle edge. It was found that for the UTM30LX, the beam width is approximately half of the angular resolution or 0.1°. Since the beam has a certain width - obstacles with features smaller than the beam width times the range would not be accurately scanned.

The purple line represents the laser scanners angular resolution, which for the UTM30LX is $\Delta\theta = 0.25°$. Any object smaller than $R\Delta\theta$ ($R$ is the range to object) may not be picked up by the laser and thus may not be adequately represented in

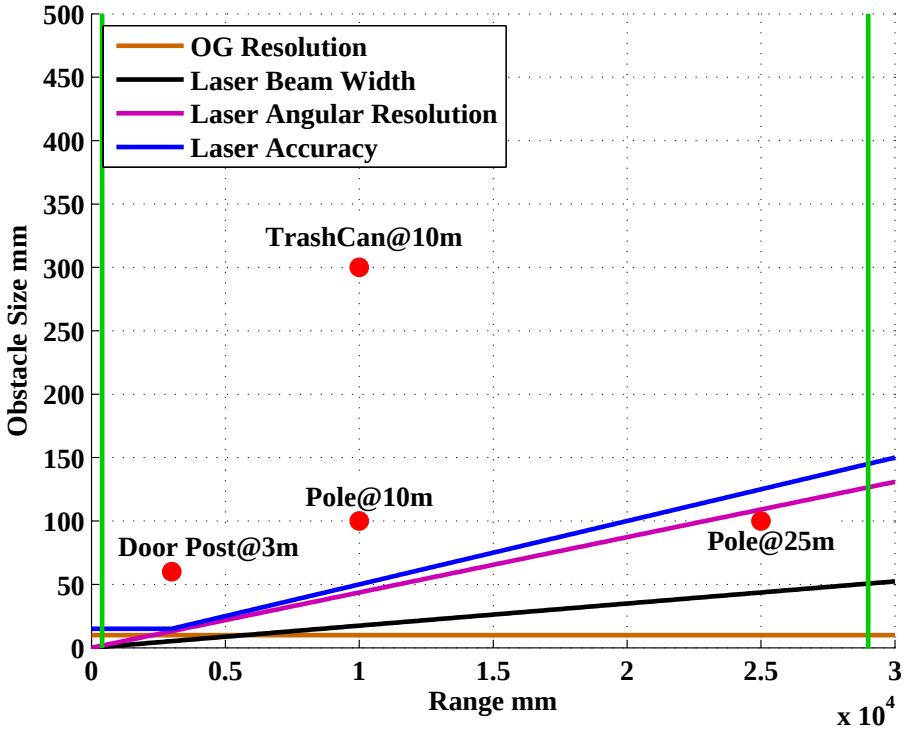



Figure 3.20: Detectable obstacle size bounds.

the occupancy grid. A typical example may be a wire or a slender pole that may not be captured by the scanner at a distance due to the low angular resolution.

The last bound is set by the laser scanner's accuracy. The laser noise may cause a misrepresentation of a relatively small obstacle with a size that is smaller than the intensity of the noise. As the laser noise increases with the measured range - this phenomena is more pronounced at large ranges. The noise model that was used for the UTM30LX employs a constant noise level of 15 $mm$ up to 3 $m$, and a 0.5% noise level for $R > 3\,m$. This was derived from the laser scanners accuracy experiments (see Sub-Section 4.1.2). For the UTM30LX, this lower bound appears to be the largest one.

Figure 3.20 shows several red dots that represent typical size objects such as a trash can, pole, and a door post. All are objects that exist in the typical office like environment. Any obstacle that is placed above the blue line (lower bound), and within the min/max range of the laser scanner (two vertical green lines) would be picked up and properly represented by the SLAM algorithm. The relatively large trash can is positioned well above the lower bounds and so it is likely to be accurately represented (and therefore used) by the SLAM algorithm. However, the relatively small pole may not be accurately picked up from a long range due to laser noise, but would be accurately mapped when the laser scanner is close to it. A door post is considered to be a finer detail object, and therefore would require the laser to be even closer in order to be accurately represented. It was found that with a resolution of 1 $cm^2$

per occupancy grid cell, even the smaller door post features, approximately 1.5 *cm* in size, were picked up by the laser from close range, and had some representation in the resulting map.

## 3.6 Proposed Accuracy Metrics

The discussion presented in Section 1.1.1 suggests that targeted flight requires precise SLAM capability. The proposed algorithm is intended to be used for targeted flight operations, and as such is claimed to provide highly accurate maps and position estimates. Quantifying the accuracy obtained by the algorithm is therefore of great importance.

Comparing scan matching algorithms based on individual scenes, (such as that presented in Section 5.1) can provide useful information. However, a given algorithm's overall performance may vary substantially between different scan scenes, different test scenarios, and different laser sensors (varying number of points, laser sensor noise, etc.). For this reason, quantitative as well as qualitative comparison of maps created by laser odometry are presented, using several algorithms, employed on different datasets.

The maps are comprised of multiple laser scans, and therefore represent a more challenging objective for performance evaluation, as compared to individual scene matching. The overall error over a complete traveled path is cumulative. Therefore, evaluating an algorithm over a series of interdependent scan matching scenes is considered to be far more challenging and rigorous, than evaluations based on a small set of isolated scenes. A large set of scenes contains a far greater variety of different shaped objects, scanned from multiple different viewing angles and ranges. The accumulated drift over a large set of scans is typically larger than the error in a single scan matching operation. Therefore, the proposed algorithm evaluation is based on errors accumulated over a certain traveled distance.

Below is the description of the metrics used for the evaluation. The first metric is simply based on measured lengths from the scenario, and the second metric was developed as part of this thesis and compares a SLAM map (in the form of an occupancy grid), with a true hand measured map.

### 3.6.1 Measured Lengths Comparison

A simple way to compare a map with the true environment is to compare measured lengths. To be effective, this practice needs to be employed on more than one dimension, to avoid possible biased spurious results (as was done by Nguyen et al. [4], where only a single measurement was used for the evaluation).

For these metrics, several distance measurements between selected points, were used to evaluate map accuracy in a benchmark scenario (Martin Hall, at the University of Maryland, see Sub-Section 4.3.1, specifically Figure 4.7, where these points are marked with capital letters). Segment lengths in the occupancy were subsequently extracted from the resulting maps.

Note that the occupancy grid has a finite resolution, stemming from its inherent cell resolution as well as the number of occupied cells that were captured in the area of the chosen point. The manual extraction involved minimal human judgement, as the surrounding area around each point typically contained sufficient information to infer the location of the evaluated point.

### 3.6.2 Average Cell Distance

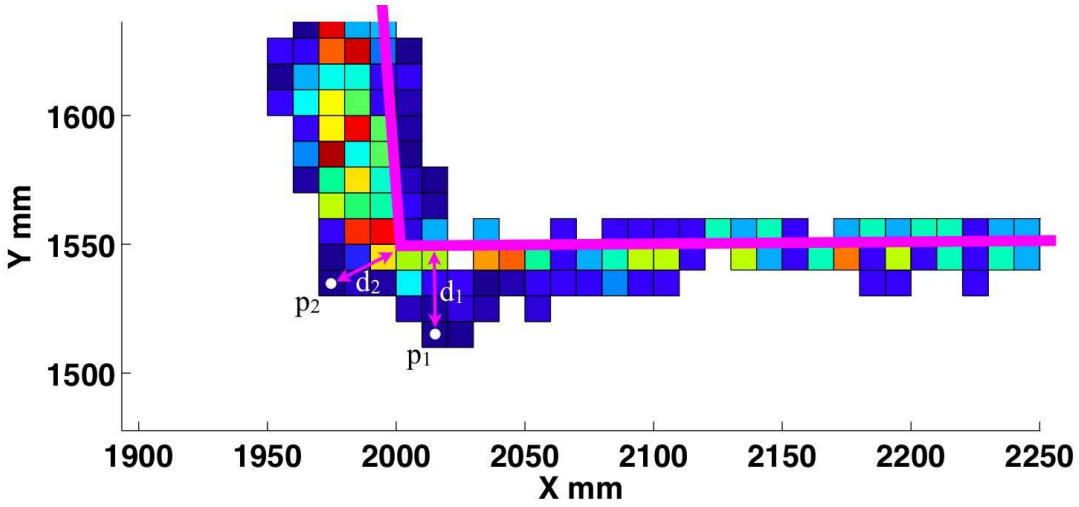



Figure 3.21: Schematics of the proposed map metric. Occupied cells are color coded by occupancy level (red-high, blue-low), and the two purple lines represent two segments of the hand measured true map. Representative cell centers are shown as white points $p_1$ and $p_2$, with their respective metric distance contributions $d_1$ and $d_2$, shown in purple arrows.

An additional metric is proposed in this thesis. For the calculation of the proposed metric, each occupied cell is matched with a segment from a hand measured map, by calculating the distance from its center to the matched wall (see point $p_1$ in Figure 3.21). The coordinates of the intersection point between the perpendicular line and the segment itself, must fall between the matched segment's end point coordinates both in the 'x' and the 'y' axes. For occupied cells that fall outside the boundaries of all segments, the distance to the nearest corner is calculated (for example $p_2$ in Figure 3.21). The distance is weighted by the cell occupancy level (normalized to unity), and all contributions are then added to a total cost, given by Eq. (3.37):

$$C_W = \left( \frac{1}{n_o} \sum_{i=1}^{n_o} D_i W_i \right) \tag{3.37}$$

where $C_W$ is the weighted cost, $n_o$ is the total number of occupied cells, $D_i$ and $W_i$ are the distance and occupancy weight of the $i^{th}$ cell, respectively. Note that the final presented value is without the cell's occupancy weight, which in fact represents

the average distance of all occupied cells from their respective associated walls (in $mm$). The minimizing process is done via exhaustive search in $x$, $y$, and $\psi$, to assure a global minimum is achieved.

### 3.6.3 Loop Closure Seamlessness

This metric is more qualitative by nature and relies on manual inspection of the loop closure area. In scenarios where the traveled path returns to the start position area, one may examine the alignment of the walls from the beginning and end of the traveled path. A good alignment suggests a low drift, while large gaps between similar walls may even be used to estimate the drift quantitatively, by measuring the distance between the two map walls.

## 3.7 Summary - Novel Algorithms

The novel algorithms presented in this chapter make use of some of the the basic robotics tools presented in Chapter 2. The PB-PSM algorithm was described in great detail, including several methods for cost rewarding that were evaluated prior to choosing the perimeter matching term, which is based on maximizing overlap between the two scans. The combined SLAM algorithm was described in detail, including it's coupling to the A* path planning algorithm. Various assumptions and limitations were presented, discussed, and analyzed using representative model problems, with emphasis on the effect of scan frequency on the measured objects, and the effect of a partially dynamic environment on the final cost. The accuracy metrics used later in this thesis were also presented.

# Chapter 4

# Experimental Setup

## 4.1 Laser Range Scanners

This section describes the sensors used in this research including calibration and individual sensor tests. This research makes use of two, 2D laser range scanners made by Hokuyo [82]. The scanners operate on the principle of measuring the phase difference of the light reflected from the target object. These scanners are relatively lightweight, and thus are suitable for robotic MAV research. This section depicts the details about the sensors, characterization testing, and experimental data that is required by the algorithm for sensor modeling.

### 4.1.1 Hokuyo URG 04LX-UG01


Figure 4.1: Hokuyo laser range scanner: URG 04LX-UG01.

The URG-04LX-UG01 (Figure 4.1) is a relatively short range laser scanner. Table 4.1 shows the sensor's specification as provided by the manufacturer [83].

Table 4.1: Hokuyo URG-04LX-UG01 manufacturer specification.

| | |
|---|---|
| Range | 5 $m$ |
| Field Of View | 240° |
| Spacial Resolution | 1 $mm$ |
| Angular resolution | 0.35° |
| Number of points | 683 |
| Accuracy | $< 3\%$ (above 1 $m$) |
| Scan Frequency | 10 $Hz$ |
| Required Power | 2.5 $W$ |
| Weight | 160 $grams$ |

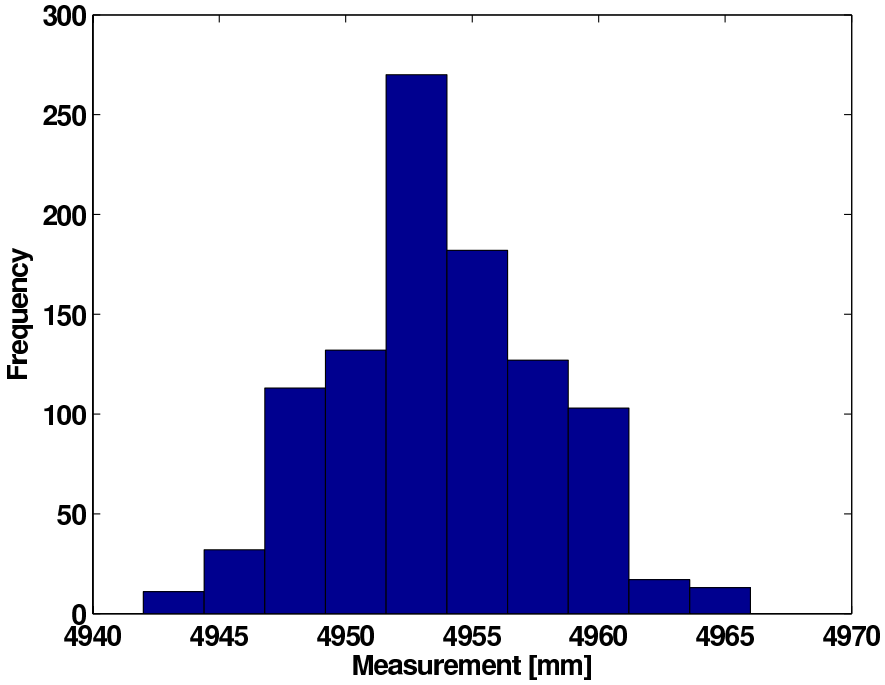



Figure 4.2: A single distance measurement statistical distribution.

The laser sensor measurements all exhibited a Gaussian distribution. An example is presented in Figure 4.2, showing a histogram of 1000 laser measurements of a 5 $m$ range. The manual measurement error is of the order of 10 $mm$ (mainly due to human optical measurements error, possible slight curvature of the measured object, and small angle misalignments).

Figure 4.3: Hokuyo laser range scanner: UTM-30LX.

### 4.1.2 Hokuyo UTM-30LX

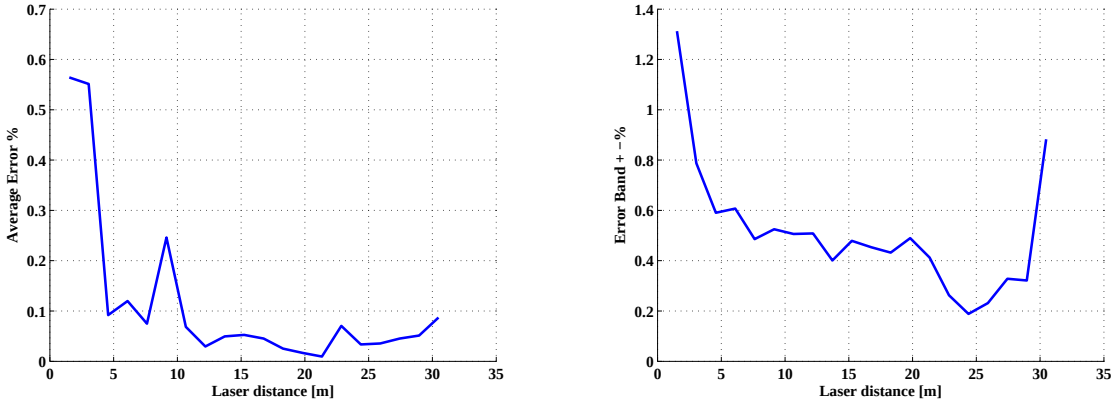
The UTM-30LX is also a laser scanner produced by Hokuyo (Figure 4.3) with a higher range of 30 $m$. Table 4.2 shows the manufacturer's specifications [76] (experimentally verified). The UTM-30LX also exhibited a Gaussian distribution, similar to the one shown in Figure 4.2.

Table 4.2: Hokuyo UTM-30LX manufacturer specification.

| | |
|---|---|
| Range | 30 $m$ |
| Field Of View | 270° |
| Spacial Resolution | 1 $mm$ |
| Angular resolution | 0.25° |
| Number of points | 1081 |
| Accuracy | $< 1\%$ (above 1 $m$) |
| Scan Frequency | 40 $Hz$ |
| Required Power | $7 - 8$ $W$ |
| Weight | 220 $grams$ |

The laser scanner accuracy (noise) characteristics were verified experimentally. A single object was measured at 20 distances up to the laser's maximum range. For each distance, the laser recorded 1000 range measurements, and the mean and total error band were extracted. Figure 4.4 shows the experimental error results in percent of the manually measured distance.

Figure 4.4 shows that the mean error is close to zero across most of the measured distance range. The laser noise was therefore modeled as Gaussian with zero mean. The standard deviation can be extracted from the error band since for a Gaussian




(a) (b)

Figure 4.4: UTM-30LX noise characteristics (Experimental). (a) Mean error in percents vs. measured distance. (b) Total error band, in percents vs. the measured distance.

distribution, the error band represents $\pm 3\sigma$. And so the standard deviation can approximated as:

$$\sigma = \frac{0.005 R_i}{3} \tag{4.1}$$

where $R_i$ is the $i^{th}$ laser range measurement.

## 4.2 Platforms

Several platforms were used in this research, to show the robustness of the algorithm and the independence of the results' quality and accuracy of the platform used to mobilize the laser scanner. The research was initiated with a simple wheeled platform, investigating various aspects of the algorithm's performance, followed by a more involved ground platform – a walking person, and finally an aerial platform, in the form of a single rotor helicopter.

### 4.2.1 Wheeled Platform

The ground platform is a simple wooden cart, on which the laser was mounted. The wheeled cart is approximately 50 *cm* by 40 *cm* in length and width, and 55 *cm* in height. It was manually driven through the corridors while the laser records measurements at given time intervals (scan rate). Note that the laser in this case was kept in motion *while* taking the scans, so the presented algorithm is examined for application on a moving platform. The cart and laser are shown in Figure 4.5

Figure 4.5: Cart and laser sensor.

### 4.2.2 Human Platform

Experiments were also carried out with the laser scanner being hand-held by a person, walking at various speeds. This introduces some pitch and roll motions to the laser sensor, and the general motion is naturally less stable, as compared with the wheeled cart. These experiments were used to better simulate the conditions that will be experienced by the laser on board the helicopter MAV.

### 4.2.3 Aerial Platform

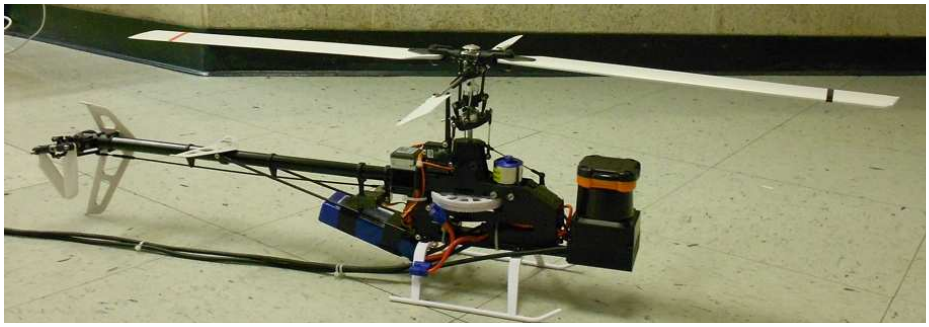


Figure 4.6: Blade 450 helicopter. Laser scanner mounted at the front.

Finally, for the aerial platform, an off-the-shelf single rotor remote controlled helicopter, Blade 450 (Blade Helicopters) was used (Figure 4.6). It features a main rotor diameter of 720 *mm*, weighing 760 *grams*. The 2D laser scanner UTM-30LX, was mounted at the front of the helicopter, and a tethered cable supplied power to

the laser as well as downloaded information from the laser to an off-board laptop, carrying out the SLAM algorithm.

## 4.3 Scenarios

### 4.3.1 Martin Hall, UMD

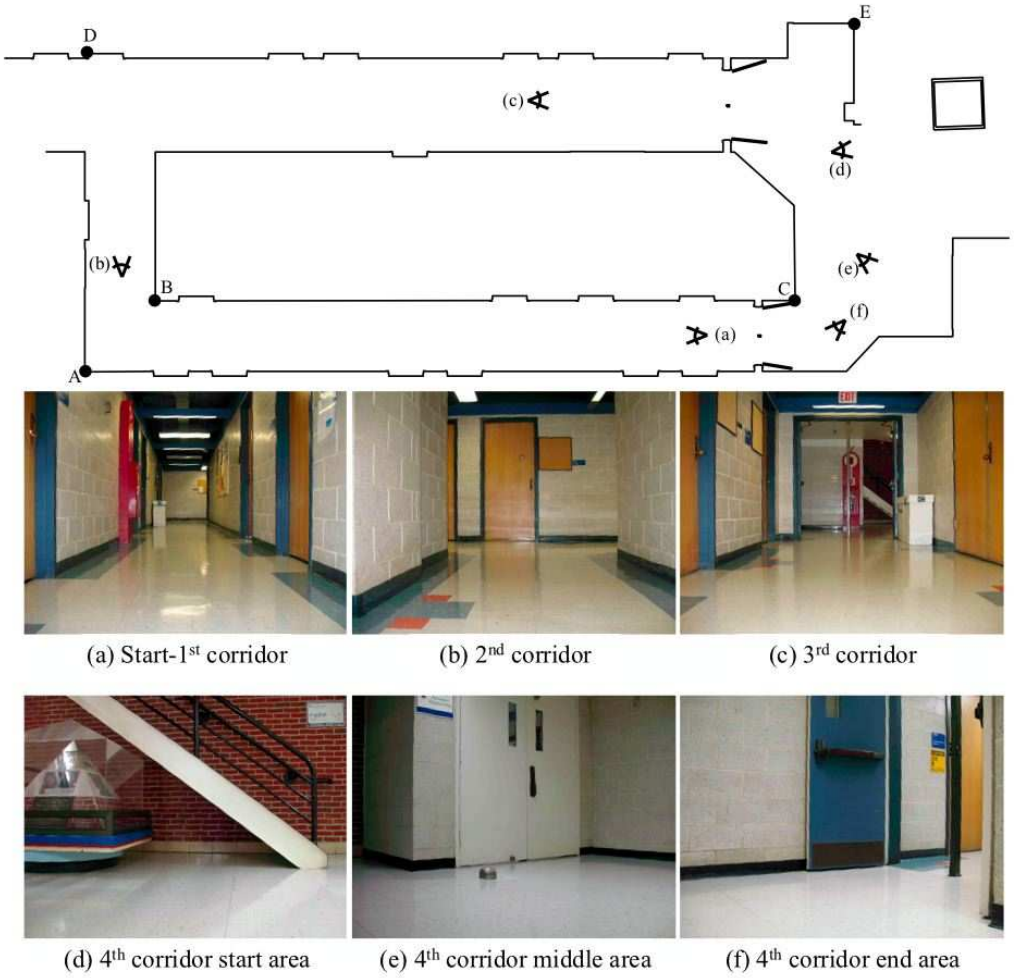



Figure 4.7: Martin Hall environment layout. Pictures' points of view are marked with their respective letters (lower case, in brackets). Selected points are marked with dots and respective capital letters, to be used later for measurements.

A primary indoor scenario in this work was the $3^{rd}$ floor in Martin Hall, at the University of Maryland. It was selected because of its availability during night time and weekends. This scenario was the focus of several experiments, including metric map benchmarking. For this reason, it was also hand measured in detail, with high accuracy (any feature larger than 1 *cm* was mapped). This environment was also

utilized for initial tests on the aerial platform. However, the limited open space prevented more advanced tests, and so other environments were utilized as well.

The environment is presented in a sequence of pictures presented in Figure 4.7, along with its 2D layout (the hand measured map), also showing the locations from which the pictures were taken (marked by lower case letters). The drawing also includes five representative points (marked by dots and capital letters) that will later be used for comparing selected measurements. The scenario includes corridors of different width, with several doorsteps, rectangular trash cans, two thin poles, and several access doors; some were kept closed (Figure 4.7(e)), and some were kept wide open at some angle to the surrounding walls (Figure 4.7(f)). The overall hand measured map accuracy is estimated to be approximately 2 *cm*, represented by over 350 straight segments.

### 4.3.2 Kim Engineering Building, UMD

The Kim Engineering building is located at the University of Maryland. The building was used for several tests, and details about the scenarios involved are given below.

#### Indoors, Ground Floor

Figure 4.8: Kim Engineering Building, ground floor rotunda.

The ground floor rotunda, inside the Kim Engineering Building, is shown in Figure 4.8. The scenario features multiple glass walls (which are not captured well by laser scanners), a large staircase in the middle of the hall, and several non-fixed and

non-2D obstacles (chairs, tables, etc.). This scenario was chosen due to its readily available size, allowing for controlled helicopter flights.

#### Outdoors, Front of Building

Figure 4.9: Kim Engineering Building, outdoors, front view.

The Kim building was also utilized as an outdoor scenario. The Front of the building is shown in Figure 4.9. It is characterized by some repeating column structure on the left side, several trees, and the front entrance to the building which features glass doors.

#### Outdoors, Back of Building

Another scenario that featured the Kim building is shown in Figure 4.10. This scenario mainly features a large round gas tank, several large trash cans and air conditioning units, and several poles, which are only captured by the laser scanner, when in close proximity due to its angular resolution).

### 4.3.3 Physics Building, UMD

The second floor at the Physics Building at the University of Maryland was also utilized, as it provides several options for closed loop trajectories. The floor plan is presented in Figure 4.11. The scenario is a typical office like environment, featuring longer corridors, as compared to the Martin Hall scenario. The main reason for choosing this scenario is to allow a closed loop course to be carried out with the aerial

Figure 4.10: Kim Engineering Building, back view.

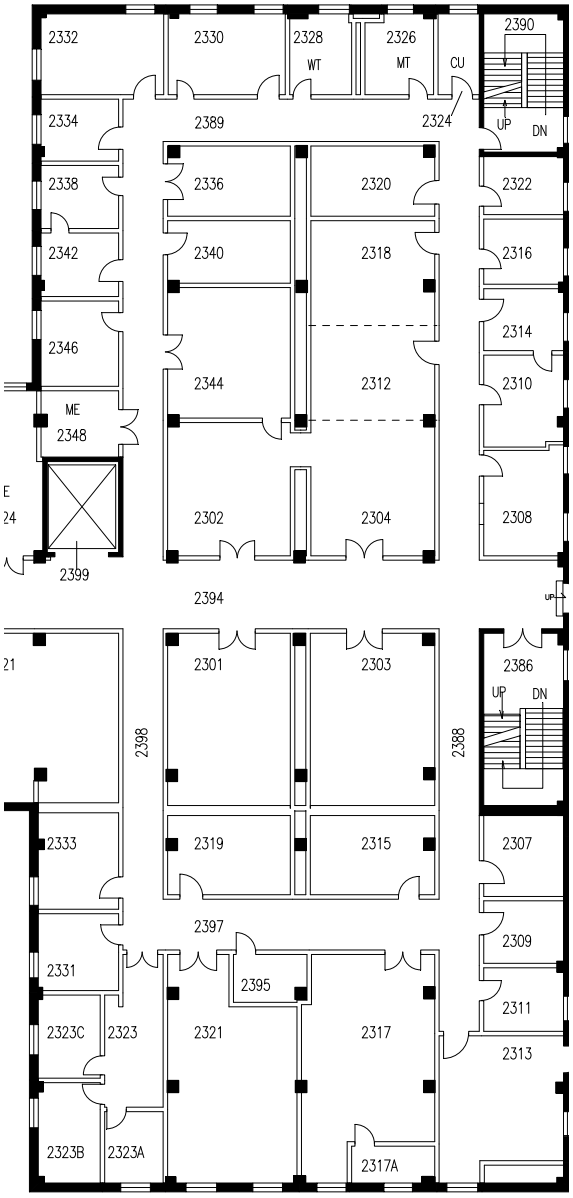



Figure 4.11: Floor plan of the Physics Building, second floor, UMD.

platform, since the Martin Hall scenario did not provide sufficient clearance for the Blade 450 helicopter (Sub-Section 4.2.3).

### 4.3.4 Northwestern Highschool, Maryland

Another outdoor scenario is part of NorthWestern High School, in Hyattsville, Maryland. The layout of the scenario is presented in Figure 4.12, and four representative pictures are shown in Figure 4.13. The area features several structures, outer

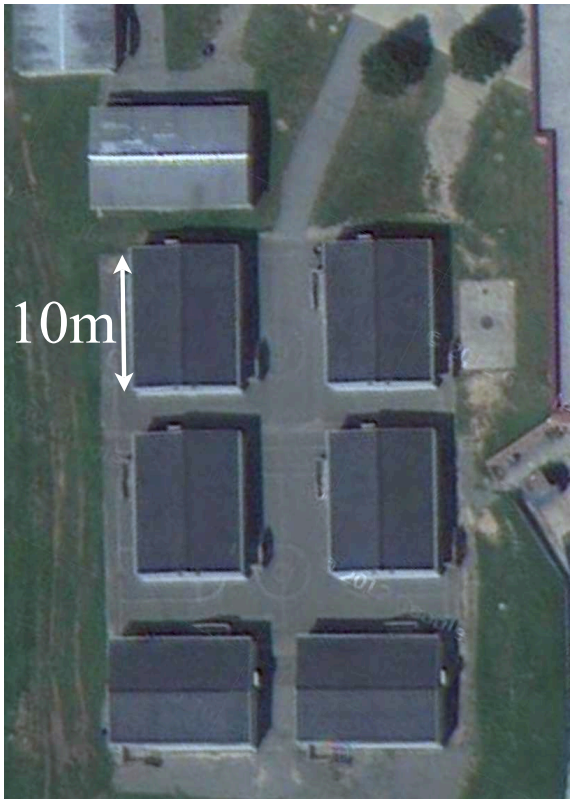



Figure 4.12: Northwestern High School scenario, Hyattsville, Maryland. Top view.

walls, and trees. The structures do have some non-2D features, such as staircases and air conditioning units, attached at various heights.

### 4.3.5 Greenbelt Park, Maryland

Testing was also conducted in a non-urban type environment, by collecting laser data in the forest of Greenbelt Park, located in Greenbelt, Maryland, shown in Figure 4.14. This forest environment significantly differs from the urban type environments, as the only features are tree trunks with relatively similar diameters. This environment is relatively sparse in features, and more susceptible to false matches as the structure of the obstacles is almost identical.

Figure 4.13: Representative pictures from Northwestern High School, outdoors scenario.

Figure 4.14: Greenbelt Park. Outdoors scenario featuring a cluttered environment with indistinguishable features.

# Chapter 5

# Experimental Results

This chapter first presents validation of the algorithm capabilities on all three platforms. The accuracy achieved by the platforms is compared, and evaluated. A detailed quantitative analysis of the algorithm's performance is presented, including comparison to previously published algorithms. The SLAM algorithm is later coupled to an A* path planning algorithm for the purpose of targeted flight experiments.

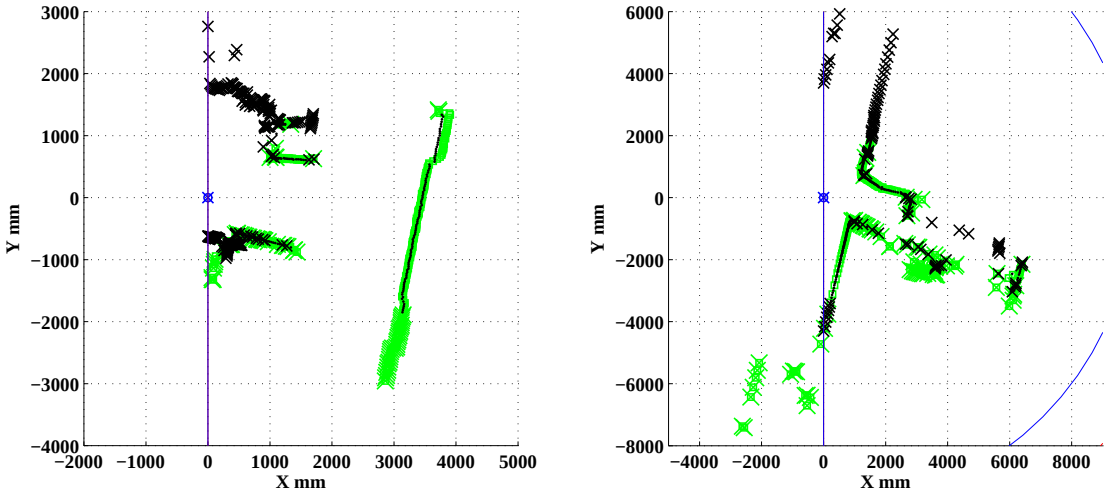
## 5.1 Single Scene Matching

This section presents baseline results for Perimeter Based Polar Scan Matching (PB-PSM) performance, when employed on individual scenarios. The results focus on accuracy and comparison to previously published algorithms, employed on the same scene datasets. The numerical convergence pattern is also presented and discussed. Comparison between the performance of the PB-PSM algorithm, and that of several other algorithms, including Polar Scan Matching (PSM), Polar Scan Matching-Cartesian (PSM-C), as well as a several different versions of Iterative Closest Point (ICP) implementations is presented. The comparison against the various ICP realizations is important as ICP is currently the most common scan matching algorithm that is being used by several groups [3, 10, 58, 71]. Comparison based on overall map accuracy is presented later in Section 5.2.

Diosi and Kleeman [29] performed several experiments where they scanned different scenarios from various known, preset points of view and laser positions. Using those scans they created a useful bench mark dataset for different algorithms, with which they compared PSM, PSM-C and a classic implementation of the ICP algorithm [54].

Of the 10 scenes presented by Diosi and Kleeman, some were stated by them to be of higher quality. Two representative scenes were chosen (scenes 2 and 7), for which all the algorithms that are compared were able to show satisfactory solutions. The most challenging match is reported by Diosi and Kleeman as match number 3, with the largest total translation distance of 717 $mm$, and the largest rotation of 27°. The ground truth translation and rotation between the two scans for match number 3 is given by the following triplet: $[\Delta x, \Delta y, \Delta\psi] = [219.4\ mm, 683.3\ mm, -27°]$.

Therefore, match number 3 was also selected as the benchmark for the PB-PSM algorithm.




Scene 2 Scene 7

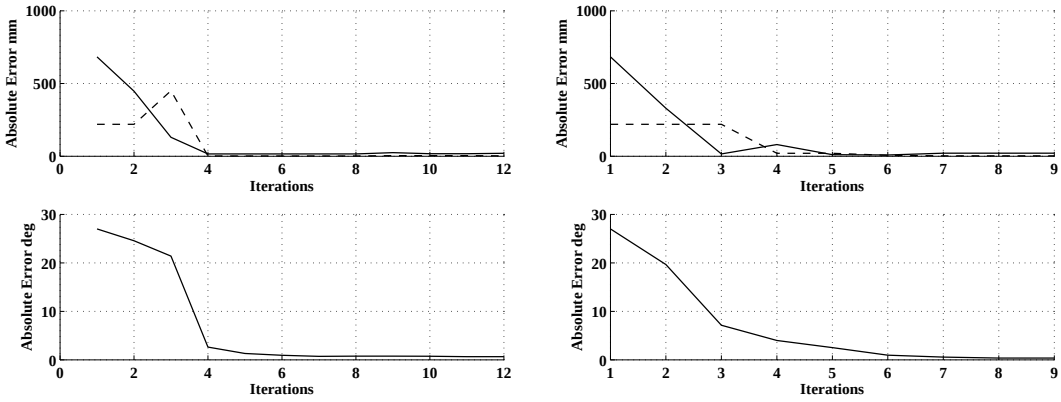
Figure 5.1: Two representative scan matching scenes. Green squares-Current Scan points (after roto-translation), black dots-Reference Scan, 'x' marks-eliminated points. The field of view considered was 180°, with a range of 10 $m$ (marked with a blue line, where visible).

Both scenes after being scan-matched are presented in Figure 5.1. Note that all points are presented, and points that are eliminated by one of the point filters are simply crossed out. The scenes contain out-of-range points, occluded points, mixed pixels, and several points that fall outside of the field of view after the roto-translation. As a result, all the filters discussed above had a contribution to a successful scan matching solution.

It is quite evident that the scan matching algorithm relies mostly on well defined features such as walls, corners, and objects of significant size (*i.e.* represented by more than one or two laser point). The relatively shapeless areas with clusters of points, are most likely highly cluttered areas, as also stated by Diosi and Kleeman [29]. These two scenes represent quite challenging cases, since the relatively large translation and rotation leaves few common features in both scenes. Nevertheless the matching appears successful (see accuracy estimates below).

### 5.1.1 Convergence Pattern

The convergence criterion that was set for the PB-PSM scan matching algorithm requires the following between two subsequent iterations: $|x^n - x^{n-1}| < 1\ mm$, $|y^n - y^{n-1}| < 1\ mm$, and $|\psi^n - \psi^{n-1}| < 0.01°$, where $n$ here represents the iteration's number. The convergence requirement on $\psi$ is considered to be extremely strict, as azimuth estimation is significantly more important than the estimation of




Scene 2 Scene 7

Figure 5.2: Convergence plots for both scenes. Upper figures: Translation error (in *mm*, solid lines - 'x' error, dashed lines - 'y' error). Lower figures - Rotation error (in degrees). Convergence criterion: 1 *mm* for translation (for each direction separately), and 0.01° for rotation.

plane translation. An error in rotation causes an immediate inconsistency in the resulting map, while a translation error only gradually degrades the map. Justification for the relatively tight convergence criterion is shown below. Figure 5.2 presents convergence plots for both scenes using the above mentioned convergence criteria. Both scenes converged after approximately 9 to 12 iterations. The number of iterations is significantly reduced when less challenging scenes are scan matched, and typically the algorithm converges within 5 to 10 iterations.

The convergence pattern for the rotation appears quite rapid and monotonous. The convergence for the translation appears to be quite rapid as well, with approximately 90% of the desired results already achieved after only 4 iterations. The planar convergence process may, in some cases, show a slight overshoot, which is immediately corrected in the following iteration. In all cases, the algorithm recovers from any increase in the error does within a single iteration, which is an important characteristic of the algorithm's convergence, and shows it's robustness.

### 5.1.2 Estimation Error

The comparison between the total translation and rotation error obtained using the PB-PSM algorithm, and several other algorithms is shown in Figure 5.3, for the two representative scenes. The algorithms compared, and the source of the presented data are listed below:

1. PSM [29].
2. PSM-C [29].
3. PB-PSM (current study).

4. PB-PSM without the Perimeter Matching term (current study). Since the PM term is excluded, this essentially represents PSM with exhaustive search for minimizing the cost function.

5. ICP [29].

6. ICP (current study, implementation by Bergstrom [84]), with 10% of the points allowed as outliers). This was done in order to compare another implementation against that used by Diosi and Kleeman [29], shown above.

7. ICP with exhaustive search, rather than least squares, which allows the use of the perimeter matching term. The cost function minimization process is identical to that of PB-PSM, but the data association follows the closest point metric, rather than the matching bearing metric.

8. ICP with exhaustive search, but without employing the perimeter matching term.

The comparison reveals two interesting topics. First, it shows the effectiveness of the perimeter matching term, especially in predicting rotation, when applied to both the PB-PSM and the ICP (using exhaustive search). Second, the PB-PSM algorithm is shown to be comparable to both PSM and PSM-C, while at the same time, it shows improvement when compared to both ICP implementations, particularly with regard to estimating rotation. Note that although the translation estimation for scene 7 appears quite similar between all 8 algorithms, there are significant differences in the rotation estimation error.

## 5.2 Mapping Accuracy

This section presents the evidence for the claimed accuracy, including metric evaluations and comparison to previously published algorithms.

### 5.2.1 Ground Platform Evaluation

The evaluation of the SLAM algorithm was extensively performed using the ground platform. The benchmark scenario was chosen to be the $3^{rd}$ floor in Martin Hall, at the University of Maryland. The environment is presented in a sequence of pictures presented in Figure 4.7, along with its 2D layout (the hand measured map). More details are described in Sub-Section 4.3.1.

A total of 12 similar experiments were conducted, all featuring a closed loop course, that starts at approximately at viewpoint ($a$) in Figure 4.7, facing left, and moving clockwise, until the traveled path has formed a closed loop. The metrics were employed on all 12 experiments, and thus the final values presented in this section are in fact a Root Mean Square (RMS) value of all 12 experiments. This prevents the possibility of drawing false conclusions based on a single experiment's success or failure.

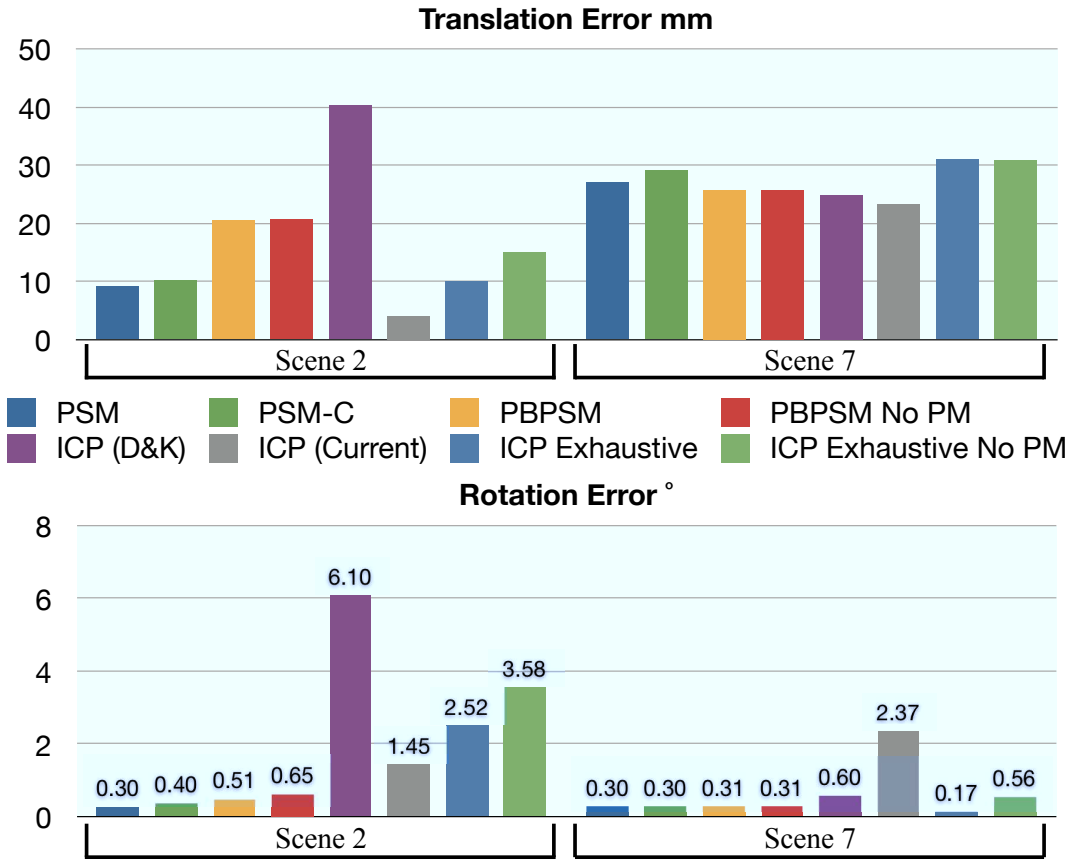



Figure 5.3: Comparison between PB-PSM, PSM, PSM-C, and several realizations of ICP. Legend shows the compared algorithm from left to right. Top figure: total translation error (*i.e.* $\sqrt{\varepsilon_x^2 + \varepsilon_y^2}$). Bottom figure: rotation error ($\varepsilon_\psi$ converted to degrees).

In each of the above mentioned experiments, a total of 100 laser scans were collected over approximately 50 $m$ of traveled path. The scan frequency was 2 $Hz$ (*i.e.*, 2 scans recorded per second), which implies average velocity of 1 $m/s$. An additional 6 experiments were conducted under the same conditions, except the door leading back to the start area was closed, and so the traveled path did not form a closed loop. This was done to avoid possible scan matching failures in areas with overlap between the start and the end of the course. This way, the final few scans do not contain any information from the start position area, which shows additional support for accuracy claims.

### Baseline Evaluation

An initial evaluation of the Martin Hall scenario (see Sub-Section 4.3.1) was performed by employing the proposed SLAM algorithm on a dataset of 200 laser scans, also collected at a rate of 2 scans per second, over the same course described above

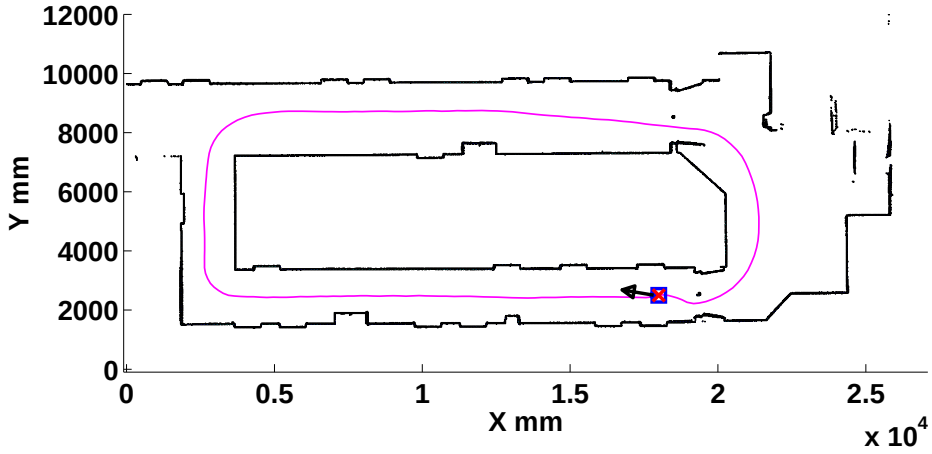
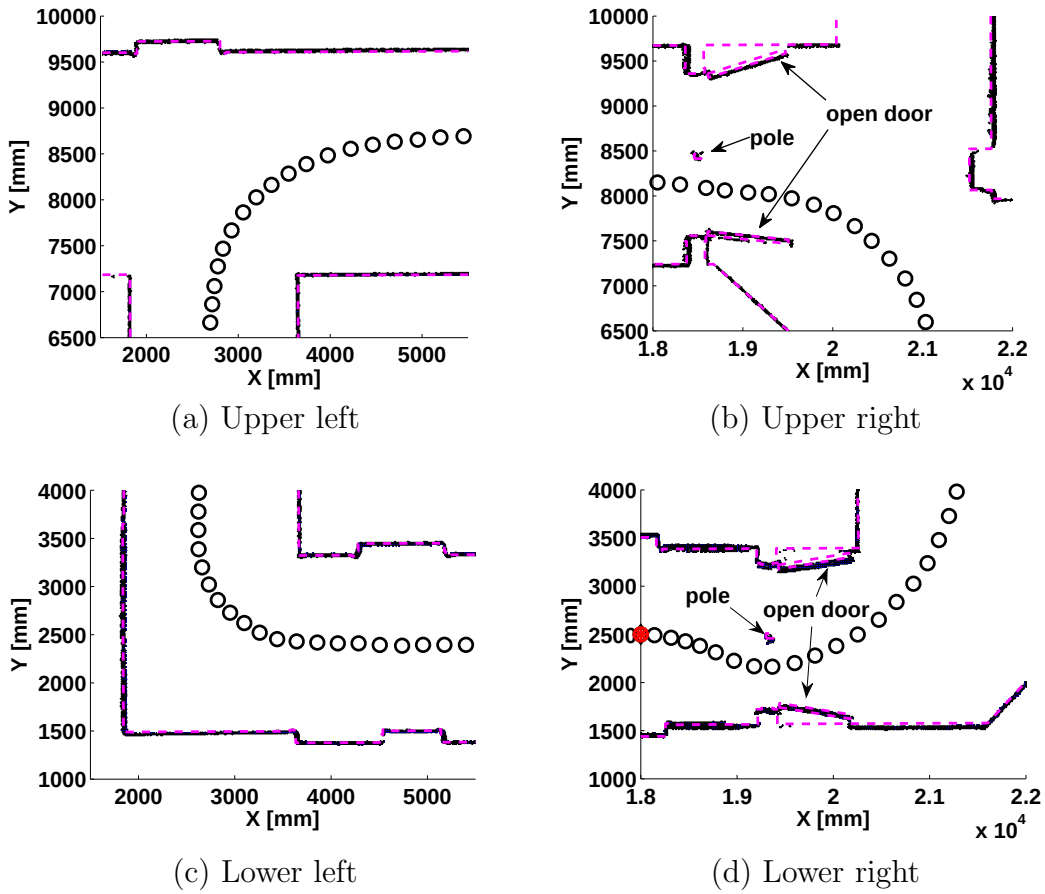



Figure 5.4: Closed loop hallway – complete map using 200 scans at 2 scans per second.

(but at a velocity of approximately 0.5 $m/s$, rather than $1m/s$). The resulting map is presented in Figure 5.4, showing a very crisp map. The closed loop course appears to be smooth with no occurrences of double walls or discontinuities. It is important to stress than this result is obtained using only the scan matching algorithm, in conjunction with the virtual scans. No loop-closure algorithm is required to achieve this level of accuracy.

A close-up of all four corners of the mapped area is presented in Figure 5.5, along with the true measured map presented on top of the virtual map using dashed purple lines. The close-ups show how accurate the mapping is, both when the platform experiences high turn rates (passing around corners), as well as when it travels in a straight motion down a long hallway. The challenge in accurately mapping the corners stems from the laser sensor yaw rate while performing sharp turns around corners. If the laser's scan speed is significantly faster compared to the platform's turn rate – the scan measurements are less affected by the platform's motion, and thus the laser signature of the environment is closer to static scan conditions (when the laser scanner does not move at all and the environment is completely static). For the results presented in this work, the platform's highest rotational velocity was 20 $deg/sec$, while the laser rotates at 14400 $deg/sec$ (see Sub-Section 5.3.3).

A detailed examination of the loop closure area is presented in Figure 5.6. The start of the loop closure area is considered to be when the virtual scan first picks up points from previously mapped occupancy grid areas. In the presented scenario the loop closure area includes approximately the last 20 steps of the traveled course. Figure 5.6 shows the loop closure area exhibiting an absolutely seamless loop closure with the occupancy grid resolution of 10 $mm$ by 10 $mm$. The previously mapped area is seamlessly merged along all the walls and doors that were picked up by the laser when the platform returns towards the start position. This highly accurate result is based on the success of the proposed SLAM algorithm to produce negligible errors in both position and map throughout the entire 45 $m$ course. As mentioned above, no loop closure algorithm was required to correct the map in any way.




(a) Upper left (b) Upper right

(c) Lower left (d) Lower right

Figure 5.5: Closed loop hallway close-up on all four corners, using 200 scans at 2 scans per second. True map in dashed line overlaid on top of the occupancy grid map.

The algorithm was then employed on another experiment in the same Martin Hall scenario, but using 100 laser scans, leading to a velocity of approximately 1 $m/s$. The resulting occupancy grid map is presented in Figure 5.7, along with the true hand measured map overlaid on top (solid grey lines, for clarity), and a total of 9 closeups showing the occupancy grid and the true map at various sections of the traveled course. The closeup insets are listed by the corridor number (with the lower corridor as #1, and counting clockwise, following Figure 4.7), and an accompanying letter for reference. The insets are a direct zoom-in crops of the same map.

Figure 5.7 shows that all the features appear to be accurately mapped with deviations that do not exceed 3 $cm$. The close-ups reveal how accurate the mapping is, even when the velocity is doubled, as compared with the previous case. A close examination of Figure 5.7 reveals that most walls have a thickness of no more than 3-4 occupancy grid cells ($3-4$ $cm$ using the chosen occupancy grid resolution), which results in a relatively crisp map.

Note that in Figure 5.7, inset $4a$ shows a typical example of what happens when the mapped scenario contains objects that do not reflect the laser beams with sufficient

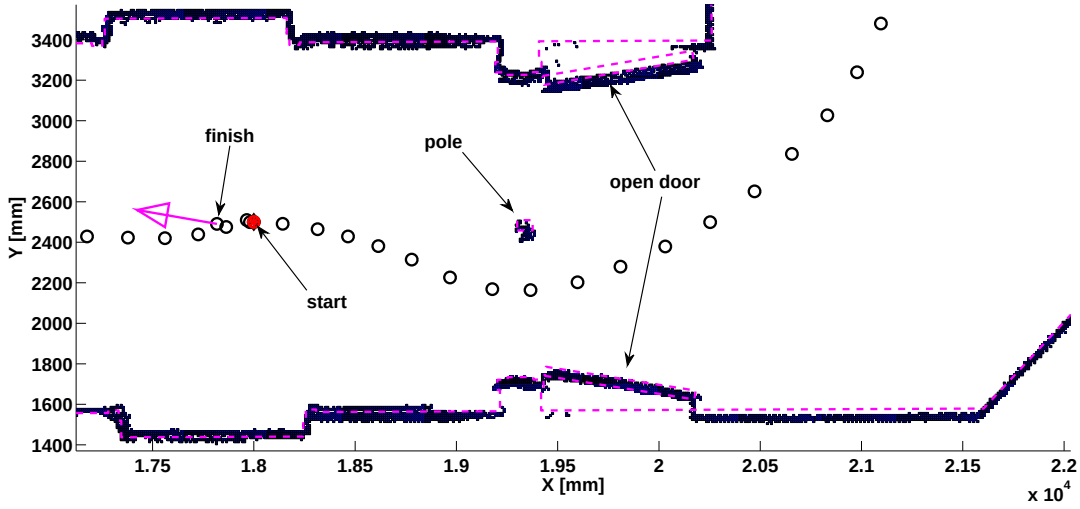



Figure 5.6: Closed loop hallway – close-up on loop closure area for the case of 200 scans, 2 scans per second (average velocity of 50 $cm/sec$). True map in dashed line overlaid on top of the occupancy grid map.

intensity, such as windows, and metallic surfaces. The mapped object in this case is a metallic fire extinguisher positioned behind a glass window (seen at the end of the $3^{rd}$ corridor in Figure 4.7(d)). The laser is designed to pick up surfaces that reflect its energy back in a way that allows for a good distance measurement. Thus, surfaces such as mirrors, windows, etc. yield inaccurate measurements, and may not be accurately mapped. This may sometimes result in a misrepresentation or a relatively small shift in their position in the map. However, the proposed algorithm is shown to be robust enough to handle a typical office environment, that may very well contain such objects.

Additional experiments were carried out using combinations of 1 and 2 scans per second and 100 to 300 total number of scans, all using the same scenario shown above. The traveled distances in all experiments was approximately 45 $m$, which results in a platform speed between 15 $cm/sec$ and 90 $cm/sec$, while yaw rates were as high as 40 $deg/sec$. All experiments provided the same result, with the exception of two cases that contained a few scan matching failures (cost functions that were too high above a set threshold). The total number of failures was below 5% of the total number of laser scans. Note that despite a few scan matching failures, the resulting map was quite accurate and rich with information to produce successful loop closure.

Following the above results, the total drift of the position estimates may be evaluated, based on the close up shown. Figure 5.7, and Figure 5.6 show that the largest distance between the true walls, and the occupancy grid walls was estimated to be less than 4 $cm$. This means that the total drift may be approximated as 0.1% of the traveled 45 $m$. This shows a considerable improvement to the capabilities of previously published algorithms (reported or analyzed), as shown in Sub-Section 1.2.9. The accuracy obtained by the proposed algorithm appears to be an order of magnitude

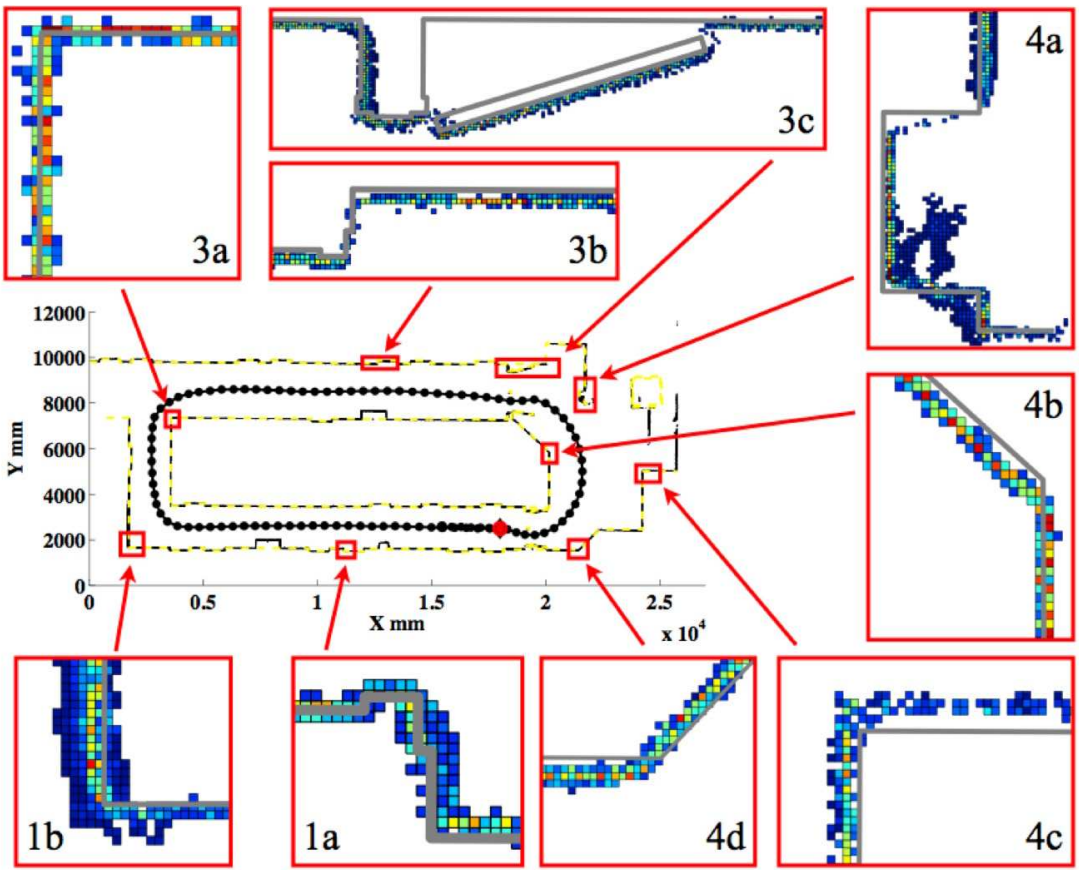



Figure 5.7: Ground platform results with insets showing 9 crops of the occupancy grid, with the true map overlaid in grey line (to distinguish from the occupancy grid colors).

better than the one obtained by the preciously published algorithms. This suggest that the PB-PSM algorithm may perform better when targeted motion missions are involved.

### Using Previously Mapped Areas

An additional experiment was carried out to examine how the algorithm performs when the platform has completed mapping of the scenario, but continues to move through the already mapped area. For this experiment, the platform collected 300 laser scans at a rate of 2 scans per second (traveling at a speed of approximately 50 $cm/sec$), while completing approximately 1.5 laps around the closed loop course. The resulting map is presented in Figure 5.8 with the true map in dashed lines, overlaid on top of the occupancy grid. As in the previous cases, one can see that the mapping is quite accurate, excluding two doorsteps along the top wall. Those two doorsteps were misaligned at first (due to bad scan matching, which was eliminated by the cost function threshold), but the algorithm was capable of recovering from this error, and complete the closed loop course.

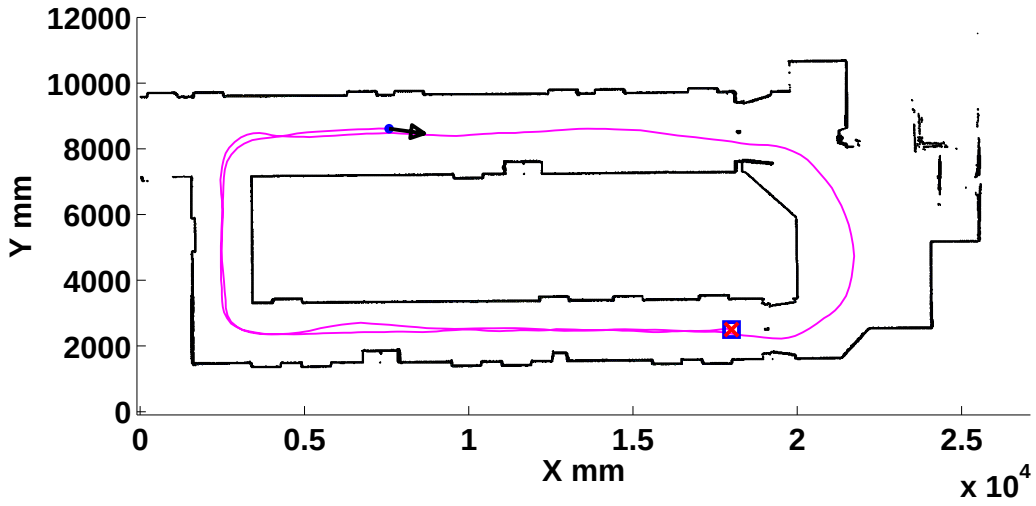



Figure 5.8: Closed loop hallway – complete map with 1.5 laps, using 300 scans at 2 scans per second, true map in dashed line on top of the occupancy grid map.

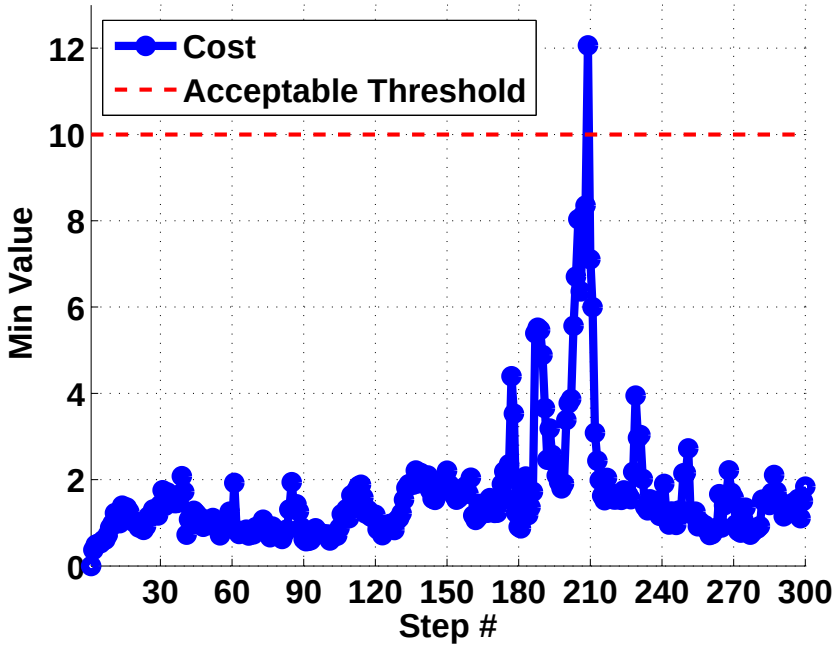



Figure 5.9: Cost function over 300 steps for the case of 1.5 laps, at a scan rate of 2 scans per second.

As the scan matching cost function provides some (although not accurate) measure of the quality of the scan matching algorithm, it can be used to study how loop closure affects estimated results, *i.e.*, study the algorithm's performance after returning to a previously mapped area. The scan matching cost values for this case are presented in Figure 5.9 for all 299 scan matching operations performed (as the first scan is registered directly into the occupancy grid). Note that each sudden increase in the cost typically reflects the discovery of a newly observed obstacle. After such discovery, the added scan data gradually reduces the cost as more observations are added to the occupancy grid and thus improves it's accuracy.

The first lap was completed after 213 scans have been taken (its cost is flagged in Figure 5.9). Cost values just before this point were relatively high as that area is characterized by a large transparent box, which has different reflectivity characteristics when scanned from different angles (note the square object located at $[25000, 8000]$ in Figure 5.8, which is visible in Figure 4.7(d)). However, it is evident from Figure 5.9 that the cost does not increase monotonically along the traveled course, *i.e.* estimation errors are bounded. After the loop has been closed, it returns back to the same level that was obtained when the map was created at the start of the first lap.

### Effect of Using Virtual Scans

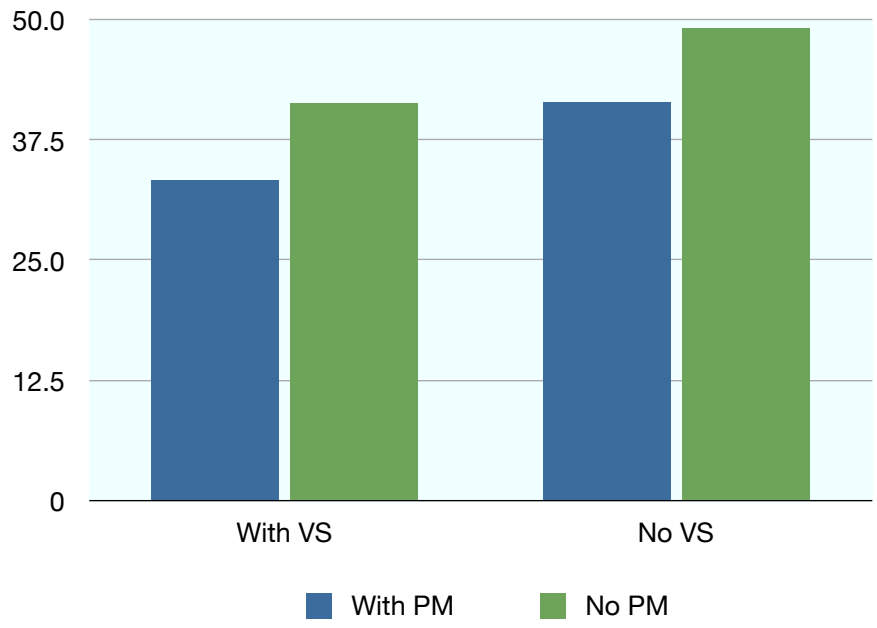



Figure 5.10: Effect of using virtual scans and perimeter matching.

The use of virtual scans is intended to improve the estimation accuracy. The virtual scan represents all the information collected thus far in previous laser scan

(stored in the occupancy grid). Therefore it is expected to yield better estimates as compared with using merely the previous laser scan. Figure 5.10 presents the map quality using the proposed metric, over the above mentioned 12 experiments (RMS value), evaluated with and without using the virtual scans. The figure shows a clear advantage for using virtual scans, as the map cost is reduced by approximately 19% and 16%, for the two cases, respectively.

### Effect of Using Perimeter Matching Term

The use of the perimeter matching term is expected to increase the accuracy, as the overlap between the scans will be maximized. Figure 5.10 (already presented above) presents results for the 12 similar experiments, and shows the advantage of using the perimeter matching term both when using virtual scans and when doing laser odometry (*i.e.*, not using virtual scans, but rather scan matching subsequent laser scans). The map cost is reduced by approximately 17% in both cases.

When using both the perimeter matching term and virtual scans, the overall map cost is improved by approximately 32%, which is a very significant, considering the total number of 12 experiments involved in this result.

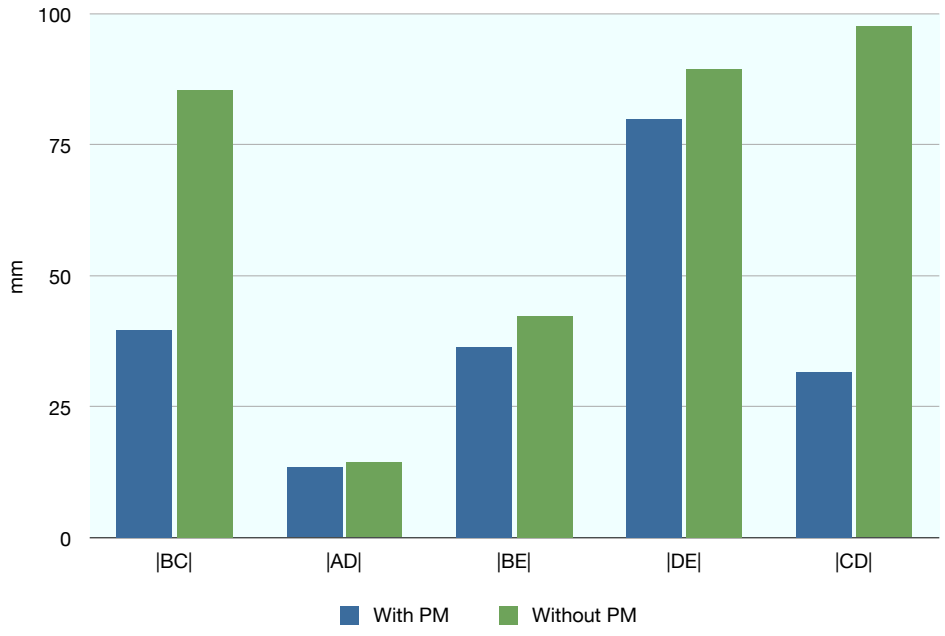



Figure 5.11: Root mean square of the error for several segment lengths in 12 identical experiments, comparing true measured lengths with lengths that are manually extracted from the occupancy grids.

In order to further present the effectiveness of the perimeter matching term, the following segment lengths: $|BC|, |AD|, |AE|, |BE|, |DE|$, and $|CD|$ (see Figure 4.7), were manually extracted from the hand measured map and compared against

the equivalent measurements taken from the resulting occupancy grids (see Sub-Section 3.6.1). Figure 5.11 presents the root mean square value of the error value in all 12 experiments, for each length, comparing the results with and without the perimeter matching term. Using of the perimeter matching term clearly shows a significant advantage for all the measured distance, except for the $|AD|$ segment, where only a minor advantage is shown. While most segments represent long traveled distances, the $|AD|$ segment is a measurement of a relatively short corridor length, and thus it seems to be less affected by the use of the perimeter matching term. In this case, virtual scans were not used, in order to prevent scan matching failure at the loop closure area (which would prevent some of the data from being inserted into the occupancy grid).

### Effect of Convergence Criterion

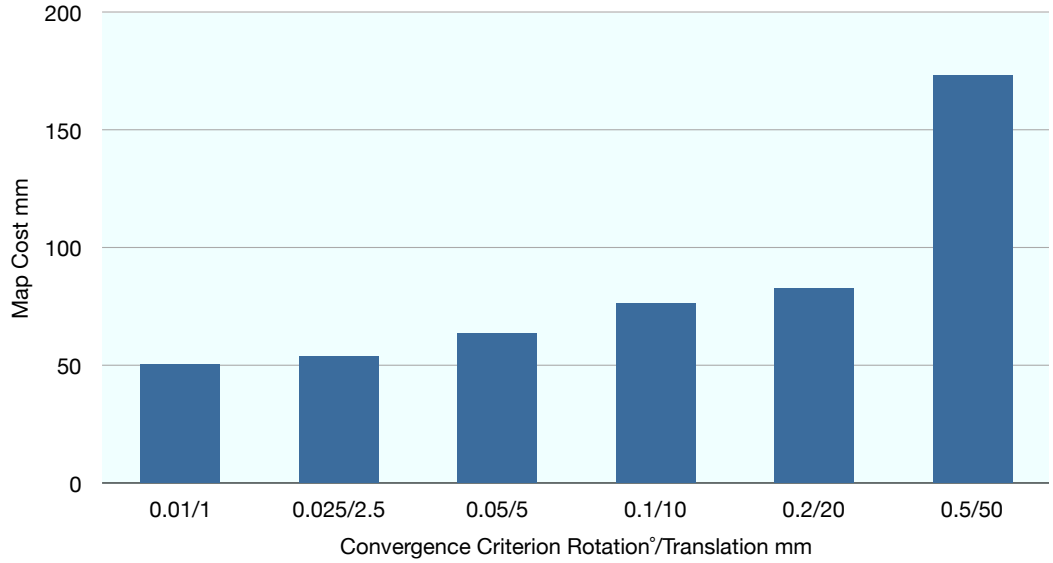



Figure 5.12: Effect of convergence criterions on the final map cost. The rotation convergence criterion is presented in degrees (on the left), while translation convergence criterion is presented in millimeters.

The effect of the convergence criterion was examined by employing the map metric described above on a set of maps created with varying convergence requirements (laser odometry only). Figure 5.12 shows the resulting map cost with the translation convergence criterion varied from 1 $mm$ to 50 $mm$, and the rotation convergence varied from 0.01° to 0.5°. In this case, the experiment used here contained 200 laser scans along the trajectory. The map cost, which represents the average distance of all occupied cells from their associated walls grows significantly, as the convergence criterion is loosened. This shows the importance of using relatively tight convergence requirements. The criterions are both gradually increased, as a tight requirement for only one criterion may compensate for a relatively loose criterion of the other.

### Non Closed Loop Experiments

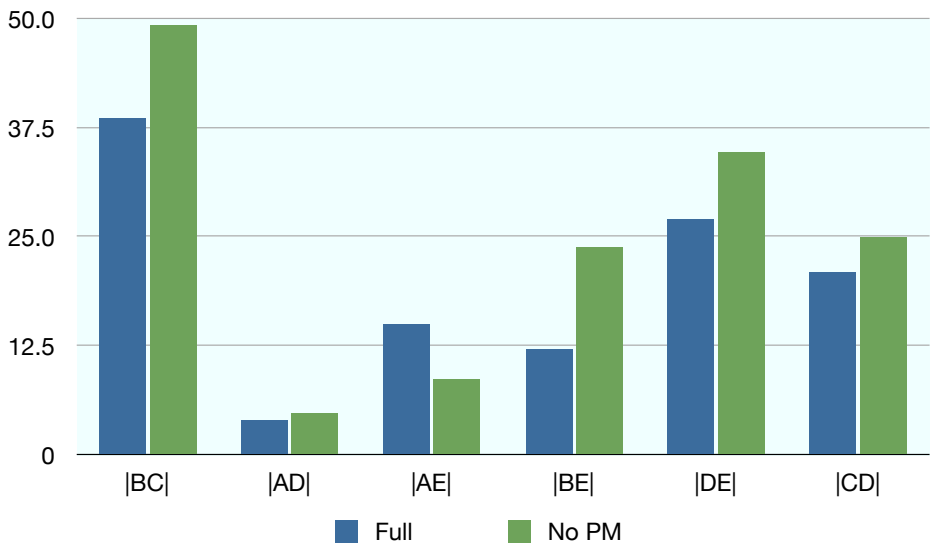



Figure 5.13: Root mean square of the error for several segment lengths in 6 identical experiments, with closed doors (no closed loop course), comparing true measured lengths with lengths that are manually extracted from the occupancy grids.

The comparison of measured lengths was also carried out for the 6 experiments that were conducted with the doors closed, which prevented the existence of a closed loop course. Figure 5.13 presents the effect of using the perimeter matching term in this case. As in the previous closed loop course example, the use of the perimeter matching term appears to significantly benefit the accuracy of the measured lengths. The exception in this case is for the length $|AE|$, however, the difference in that case is merely 6 $mm$, which is relatively small, and even below the occupancy grid resolution.

### Benchmarking Against Other Algorithms

The use of this metric also allows us to quantitatively compare the performance of the PBPSM with that of other algorithms, employed on identical complete laser datasets. Since some algorithms and their derivatives considerably failed when attempting the experiments with only 100 scans, an experiment that could be successfully completed with laser odometry by all the algorithms was chosen. The experiment chosen had 200 scans along the same traveled path. As the scan frequency was maintained on 2 $Hz$, this experiment was therefore performed at 0.5 $m/s$, which is half the speed of the previously presented 12 experiments. The higher number of scans (and respectively slower velocity) results in higher similarities between the subsequent scans.

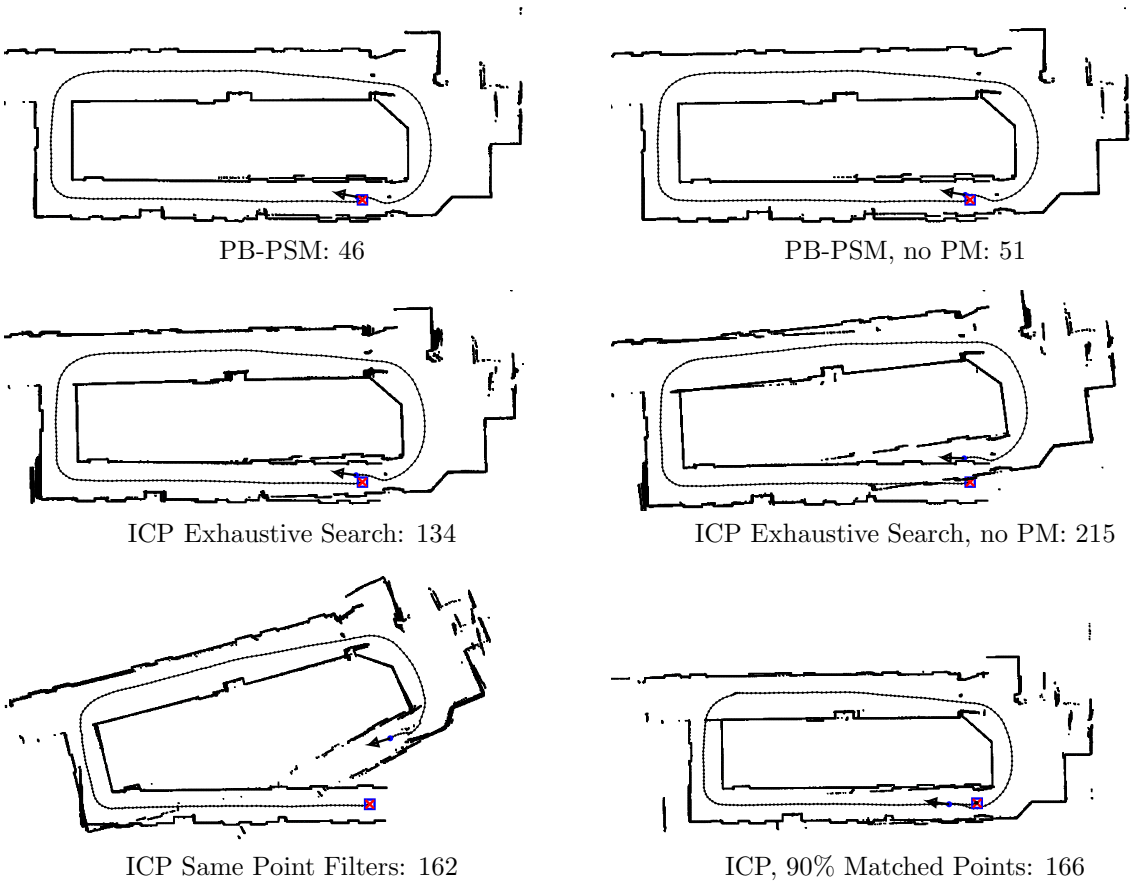



Figure 5.14: Benchmark scenario, algorithm comparison. The respective map score is calculated by the proposed map metric.

Figure 5.14 presents six maps, employing different algorithms on this dataset. All the maps were evaluated using the proposed map metric, and the final value is given below each map. The PBPSM value is shown with and without the use of the perimeter matching term to be 46 and 51, respectively, showing again the advantage of using it to reward the cost function.

The cost function detailed above, was also built using the "closest point" rule for data association (rather than PB-PSM's "matching bearing"), making it a derivative of the classic ICP algorithm. This allowed for using the perimeter matching term, while using the same algorithm for the function minimization by adaptive direct search. The resulting maps with and without employing the perimeter matching term are presented in Figure 5.14, and the associated map score of 134 and 215 show the advantage of using the rewarding term in this case as well. Note that since the "closest point" data association rule has a higher complexity than $O(n)$, the realization in this case is considerably slower than PB-PSM, and does not allow real time capability.

A comparison is also made against the classic implementation of the ICP algorithm, where the solution is obtained using least squares. Two options were tested, the first was using the same point filters as those used by PB-PSM, and the second

was that the ICP will consider only the best 90% of the matched points for the least squares solution. The second option allows the ICP algorithm to eliminate outliers based on their matching contribution. The resulting maps, presented in Figure 5.14, yielded a map score of 162 when using the same filters as in PB-PSM, and 166 when using 90% of the best matched points.

Figure 5.14 also allows a basic qualitative assessment of the resulting maps, showing that the map created by PB-PSM appears to be the best result. Moreover, the use of the perimeter matching cost regarding term in either of the two data association rules, is shown to significantly benefit the map quality. Lastly, the two results using both ICP algorithm variants result in a relatively poor map, as compared to PB-PSM.

This comparison shows the advantages of the PB-PSM algorithm over other scan matching methods, in both a qualitative and a quantitative way. The main feature of the PB-PSM algorithm, namely, the perimeter matching term, is shown again to be effective in achieving a better overall result. Another key features in PB-PSM is the cost minimization using adaptive direct search, which is shown to be important. However, a cost function with a higher computational complexity than $O(n)$ may prohibit this approach due to the relatively high number of function evaluations involved.

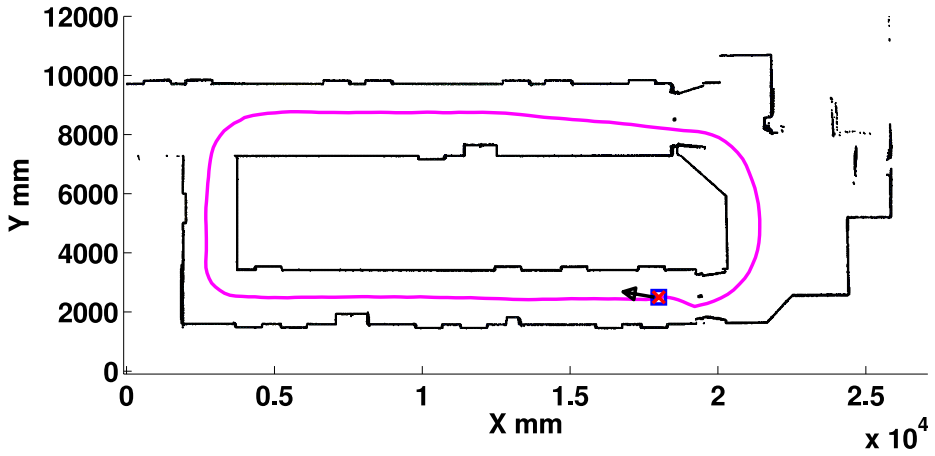



Figure 5.15: Benchmark scenario. Using exhaustive search, and perimeter matching term, but with the closest point rule for data association (as in ICP).

The ICP data association, along with the exhaustive cost minimization, the perimeter matching term, and the use of virtual scans were combined and tested on the same benchmark scenario dataset. The mapping result is presented in Figure 5.15, where the map appears identical to the accurate map obtained by the full PB-PSM algorithm. The map score in this case was 28, showing remarkable similarity to the score of 26 achieved by the PB-PSM algorithm. This result further proves that the accuracy achieved by the PB-PSM algorithm does not depend on the data association technique, but rather stems from the innovative features in the algorithm, identified and analyzed above.

#### Qualitative Map Comparison Using Existing Datasets

An additional qualitative comparison using laser odometry is presented using Diosi and Kleeman's [29] results for mapping the first room from their dataset. Diosi and Kleeman present the results of aligning the collected scans using only their PSM algorithm for laser odometry (*i.e.* not using their EKF-based SLAM algorithm). They report the results using raw odometry, PSM, PSM-C, and their own relization of ICP. Their results are shown in Figure 5.16(a). They conclude that all scan matching techniques were outperformed by simply using the odometry for aligning sequential scans.

However, when aligning the scans using results from PB-PSM, the map, shown in Figure 5.16(b), appears better than that obtained by odometry only. Some of the walls appear to be more crisp, which implies a more accurate match as several scans overlap on each other without causing a spread of the laser points. This demonstrates the higher-accuracy that can be achieved using PB-PSM as compared with the other three methods presented by Diosi and Kleeman, or raw wheel odometry alone.

### 5.2.2 Human Platform Evaluation

The algorithm was also tested on a human platform, where the laser was hand-carried through the environment by a human operator. Needless to say, the motion of a human is less structured than that of a ground platform, as some roll and pitch angles are naturally introduced, which in turn violate the two dimensionality assumption. This poses an additional difficulty for a scan matching algorithm.

Figure 5.17 presents results for mapping corridor number one (seen in Figure 4.7) with a human operator. In this case, the laser was moved through the corridor three times. Figure 5.17 shows the map that is constructed using the PB-PSM algorithm, both without and with using virtual scans (top and bottom figures, respectively). An apparent result is the importance of using virtual scans for the SLAM process, at it is quite clear that the map obtained with laser odometry contains significant drift, while the map obtained when using the virtual scans appears accurate.

Moreover, the mapping of the corridor across all the passes made, when using the virtual scans, appear to be drift-free and creates a single coherent map. Comparison to the true, hand measured map revealed the same maximum drift of 0.1% of the traveled distance (approximately 47 $m$ in this experiment), similar to the case of the ground platform (see Sub-Section 5.2.1). The human operator experiments were considered as a proof of concept for aerial platform experiments.

#### Closed Loop, Physics Building

Generally, all the scenarios attempted in this thesis, were initially explored with a walking person, preceding equivalent experiments with the aerial platform. The Physic Building scenario Sub-Section 4.3.3 was therefore initially attempted with a walking person (the aerial platform experiment is presented next). The resulting map for a closed loop course is presented in Figure 5.18, where one may see a seamless loop

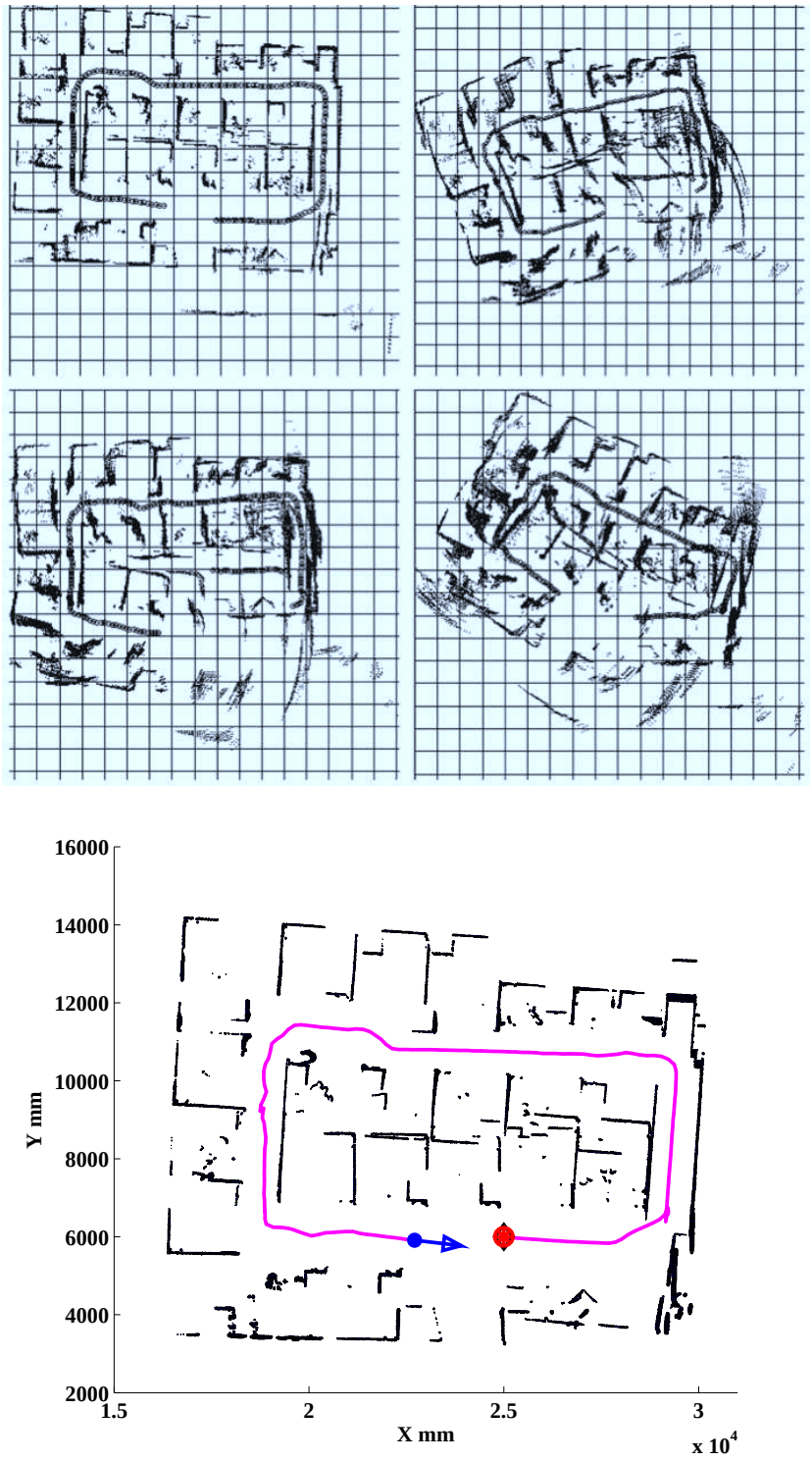
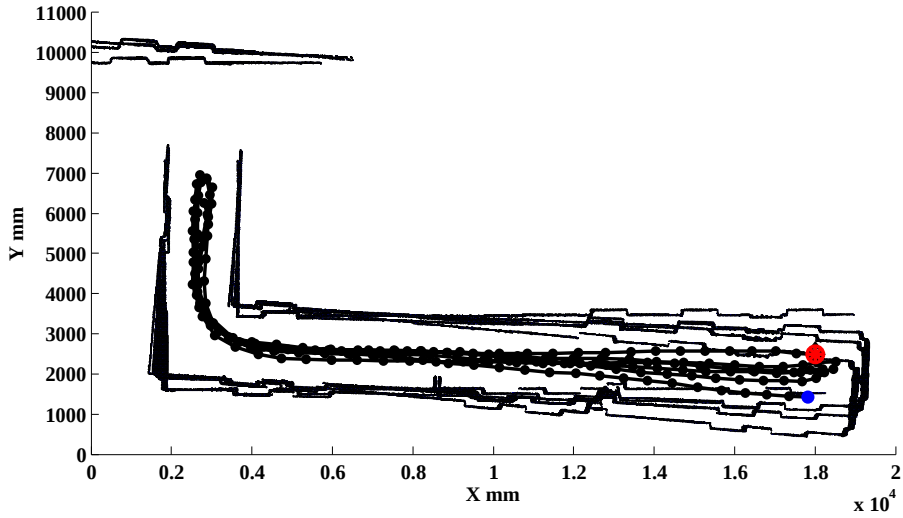
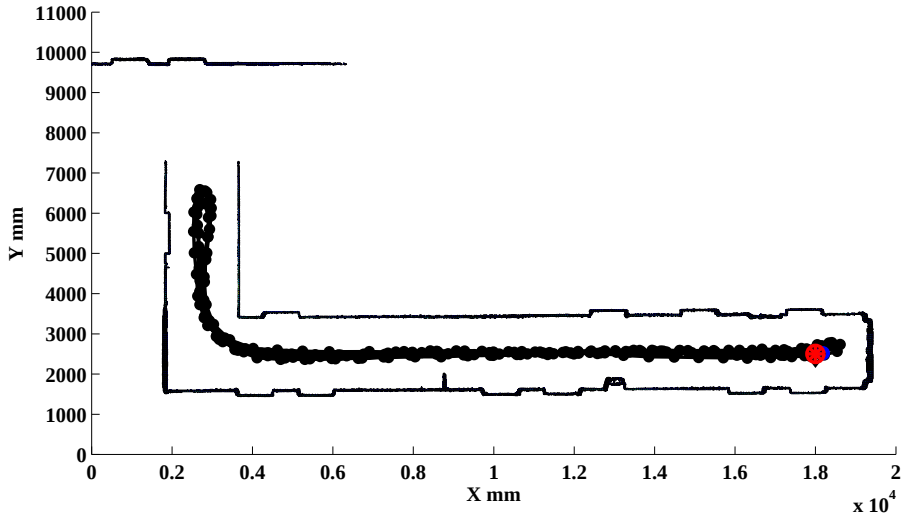



Figure 5.16: Diosi and Keelam's dataset, first mapped room. Comparing results for sequential laser-to-laser scan matching. Top left: Odometry only, Top right: PSM, Bottom left: PSM-C, Bottom right: ICP (Results by Diosi and Kleeman). Bottom: PB-PSM. Start point is marked by a large asterisk and a red circle, end point is marked by a smaller blue point, with an arrow pointing towards the final azimuth.




(a) PB-PSM algorithm only (scan-to-scan matching)




(b) PB-PSM with the SLAM algorithm

Figure 5.17: Results using a walking person. Red point on the right marks the starting point. Three passes are made through the corridor.

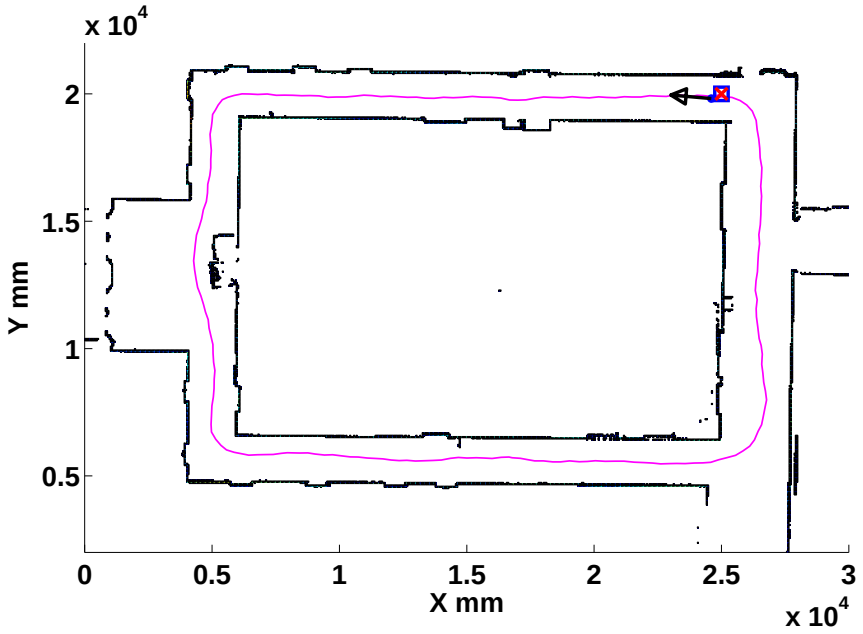



Figure 5.18: Physics Building, closed loop course with a human platform.

closure, after the person traveled approximately 70 $m$. This experiment shows that the algorithm's capabilities and cumulative accuracy were achieved with the walking platform as well, in a different scenario than Martin Hall.

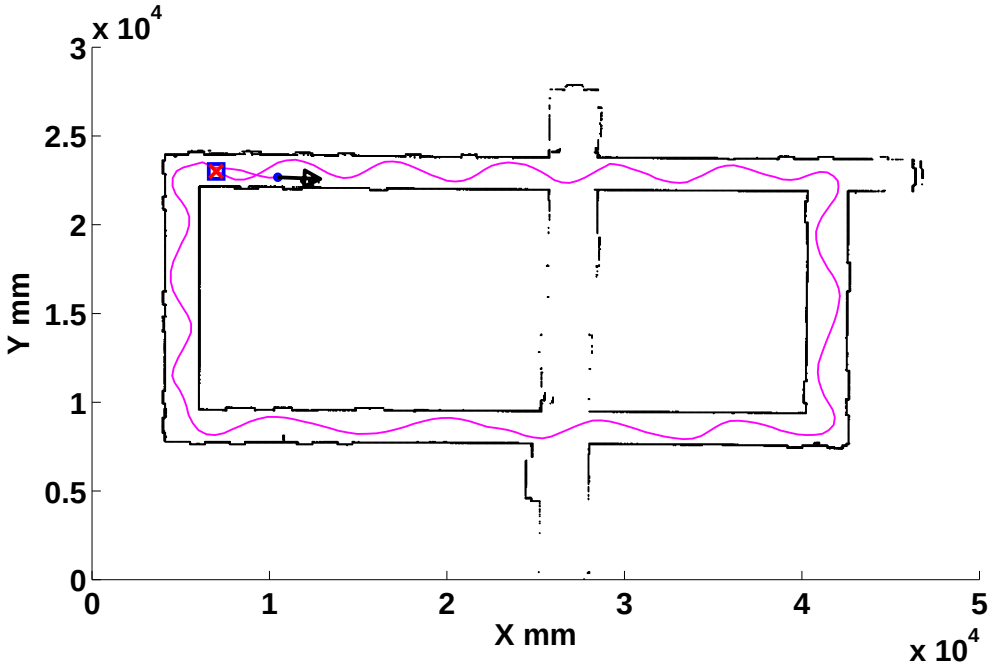



Figure 5.19: Physics Building, challenging motion pattern with a human platform, for a total traveled distance of over 110 $m$.

An additional experiment in the same environment was conducted to further show the algorithm's robustness. In this experiment, the traveled path was a slalom walking pattern, causing a cumulative traveled distance of over 110 $m$. The resulting map is presented in Figure 5.19, and appears to be quite accurate. The walking pattern posed a great challenge to the SLAM algorithm, changing its heading significantly between steps, and throughout the course in general.

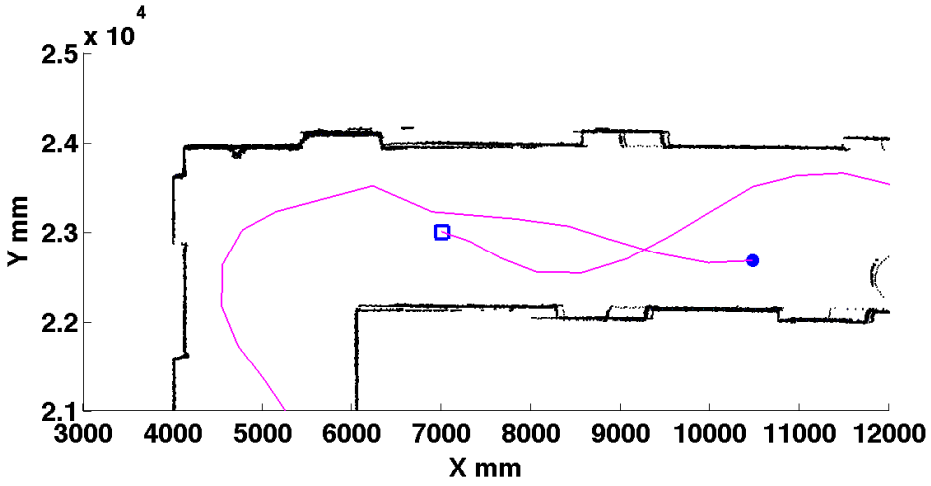



Figure 5.20: Physics Building, challenging motion pattern with a human platform, close up.

However, there are some occurrences of double-walls, since at this distance, the algorithm accumulates more drift. The loop closure area is shown in more detail in Figure 5.20. One may see the small gap that was formed due to the accumulated drift. Nevertheless, the gap is smaller than 10 $cm$, and as such is still considered relatively small, and as shown above, it is still approximately 0.1% of the total traveled distance (see Sub-Section 5.2.1).

The reader is referred to additional experiments with a human platform, conducted in outdoor scenarios, as a pre-validation for the aerial platform, in Section 5.5. Moreover, some outdoor scenarios further tested the capability of a seamless loop closure.

### 5.2.3 Aerial Platform Evaluation

The next step was to test the algorithm performance using an aerial platform. The aerial platform of choice is presented in Sub-Section 4.2.3.

#### Flight Down a Corridor

The Martin Hall scenario was first attempted with the aerial platform. Several flights were made in the upper corridor (see Figure 4.7), mainly due to its larger width, which allowed safe flight of the helicopter. Single pass as well as multiple passes through the corridor were attempted, to explore different traveled distances.

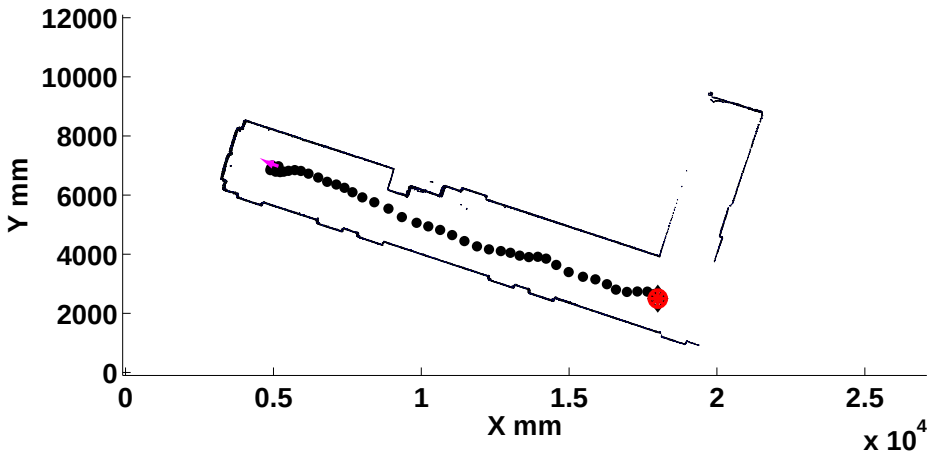



Figure 5.21: Corridor mapping – resulting map using 50 scans at 2 scans per second, pixels represent occupied grid cells, while points represent position estimates.

The map result of a single pass through the corridor is shown in Figure 5.21. The total flight distance was 17 $m$, and the helicopter was flown at an altitude of approximately 0.5 $m$. The occupancy grid is presented along with the MAVs positions estimates, represented by black dots. The starting point is marked using a red circle, while the purple arrow shows the azimuth at the last position estimate. The map appears to be crisp, and sharp, similar to previous results obtained using both the ground and the human platforms.

In order to further show the accuracy obtained by the algorithm with the helicopter platform, the Martin Hall scenario was attempted with the helicopter moving to the end of the hall and back towards its starting position. When the helicopter is flown backwards, the new scans are matched against virtual scans that contain all the information acquired thus far in the occupancy grid. Thus the scan matching quality is very robust as most of the laser scan can be matched versus an already existing map. The resulting map is presented in Figure 5.22, with the true map overlaid on top of the occupancy grid, marked using yellow dashed lines. The figure also includes 4 closeup insets for detailed comparison to the hand measured true map.

In some cases, the helicoptered underwent relatively high roll angles, and thus the laser scanner picked up readings from the floor. Generally, these scans cannot be matched against data from the occupancy grid, and thus are discarded from the scan matching process using one of the above described point filers. Nevertheless, the algorithm was robust enough to have an accurate enough match, associated with an acceptable cost value (see Sub-Section 3.5.1). In extreme cases, a floor scan may lead to a scan matching failure, as described in Sub-Section 5.3.5.

Since the height of the helicopter is not constant, the laser scanner can pick up obstacles of different depth, that are located in different heights. If the overall scan still contains enough information to preform successful scan matching - the algorithm would incorporate the entire scan into the map, which would result in both depths of

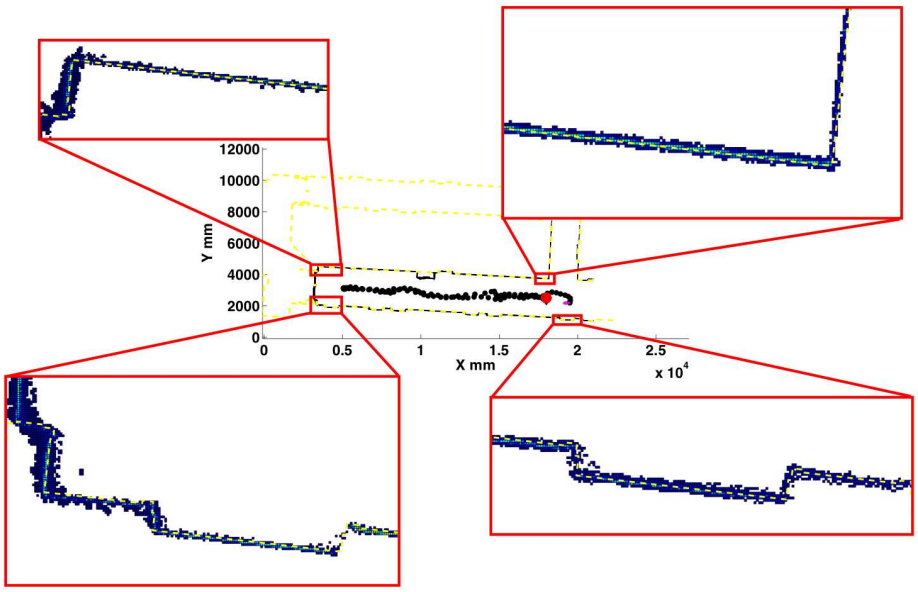



Figure 5.22: Corridor mapping – MAV was flown to the end of the hall and back - comparison between the map generated by the algorithm and the true hand measured map.

the obstacle being mapped. An example can be seen in Figure 5.22, at approximately $(x, y) = (10000, 5000)$, where the boxes next to the wall are mapped with two different depth. This can serve as an additional proof of the algorithm robustness.

The comparison between the generated map and the hand-measured true map, for the case of mapping the corridor back and forth, is also presented in Figure 5.22, where the dashed yellow line represents the true map. The overall match between the occupancy grid and the true map appears to be very good. An even closer look at four locations is given by four close-up insets. The close-ups show several fine details that are also accurately mapped by the algorithm. Note that every pixel in this figure represents a square of area 1 $cm^2$. These close-ups reveal an accuracy of approximately $2 - 3$ $cm$. Hence, out of approximately 34 meters traveled by the helicopter, the accuracy appears to be within less than 0.1% of the traveled distance. This level of accuracy is identical to the accuracy obtained using a ground platform, shown in Sub-Section 5.2.1.

In order to further test the mapping capability of the algorithm in the same corridor environment, the helicopter was flown through the hallway for a total of 8 passes. As evident in Figure 5.23, the algorithm proposed herein does not exhibit any significant drift, despite not using a map smoothing algorithm. These results stem from the integration of the evolving map in the scan matching process. The total length of the traveled path was approximately 100 $m$. Note that the only limiting factor was battery endurance, allowing less than 10 minute flights per charge.

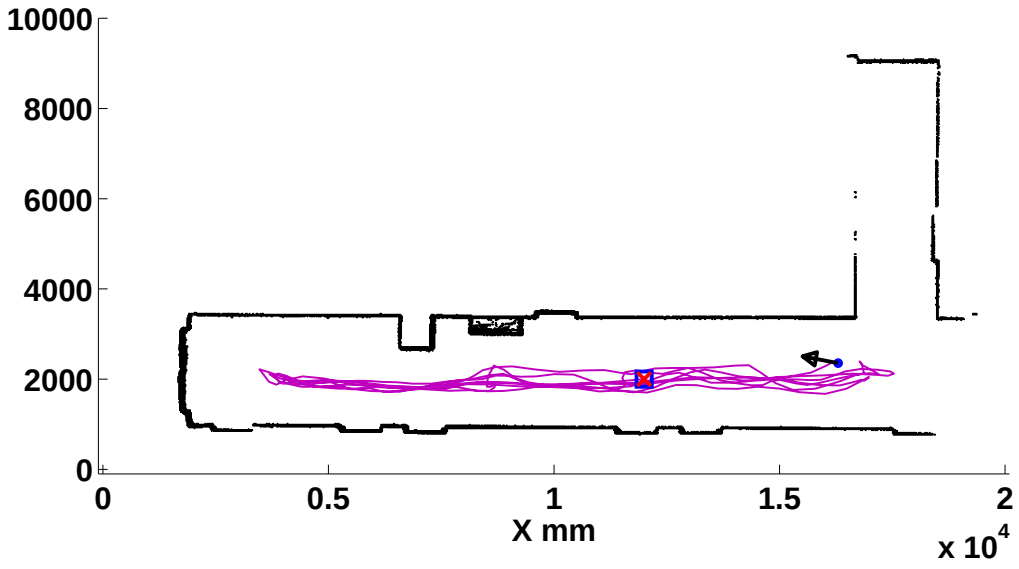
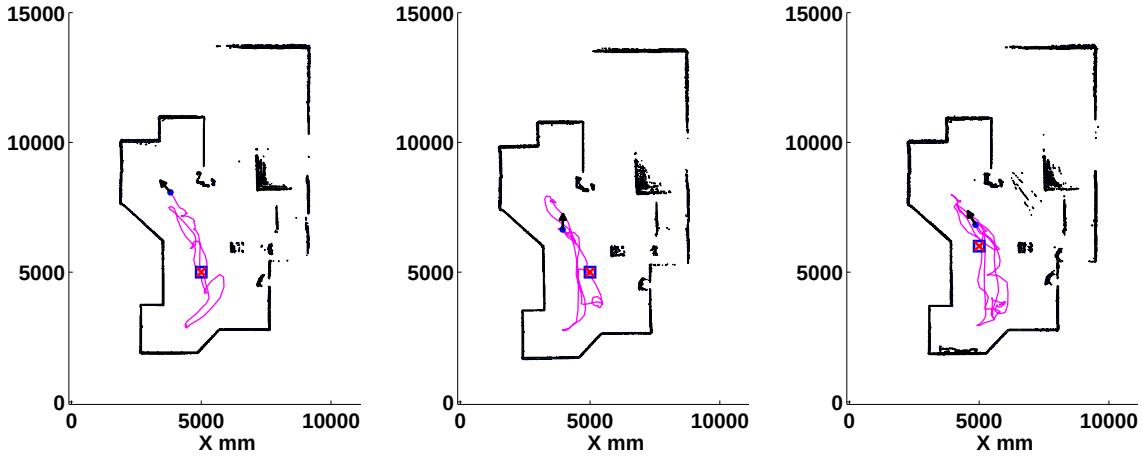



Figure 5.23: Corridor mapping – MAV was flown for a total of 8 passes through the corridor.




(a) 19 meters traveled (b) 23 meters traveled (c) 32 meters traveled

Figure 5.24: Three different runs of the same room exploration scenario.

Another mapping scenario using the proposed algorithm is presented in Figure 5.24. In this case, the helicopter is flown three times in completely random patterns, with increasing flight course length, while the surroundings are mapped. This scenario pictures can be seen in pictures $(d-f)$, in Figure 4.7, which show the right hand side of the floor-plan schematics (except all doors were closed for this scenario). The helicopter path in this example is marked with a solid purple line, initial position is marked with a large red point, and the final position is marked by a smaller blue point with an arrow pointing toward the final helicopter azimuth.

As in the previous scenarios, the map appears to be crisp and sharp, despite several objects that violate the 2D assumption, such as the staircase, or the box display (as shown in Figure 4.7 (d)). The item in the center of the room at $(x, y) = (16000, 4000)$ is the wheeled hardware cart used for the off board laptop. The pilot stands at approximately $(x, y) = (15500, 2500)$, and the author is located at $(x, y) = (16500, 2500)$ (recording laser data). The two people were standing still throughout the experiment, so as to maintain an approximately static environment. All three flights yielded identical maps within the above stated accuracy.

### Flight in a Closed-Loop Course

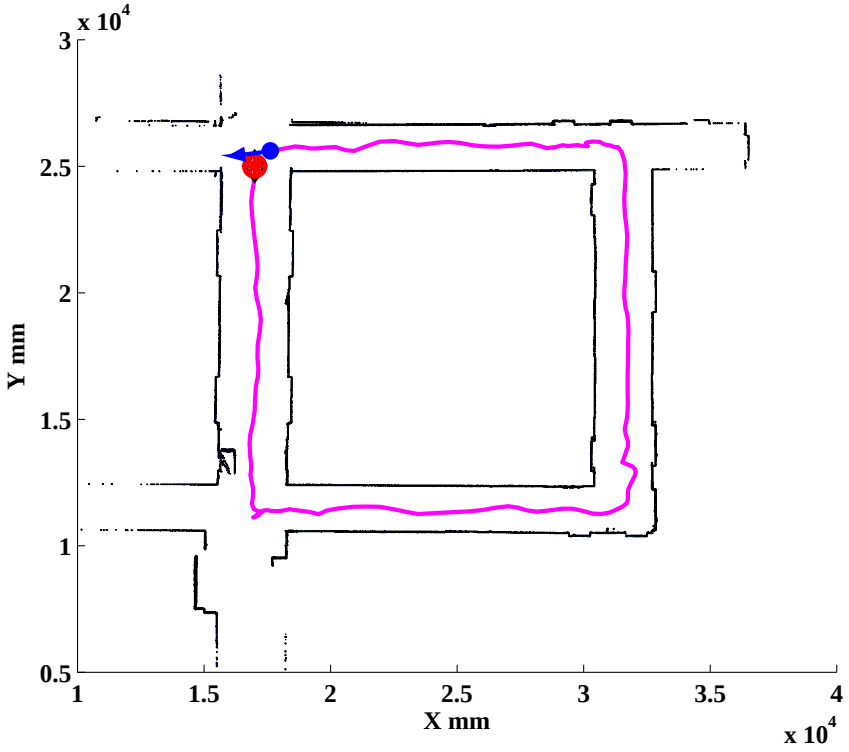



Figure 5.25: Mapping results on closed loop course. Helicopter path is marked with a purple solid line. Total course length is approximately 60 $m$.

In order to further test the accuracy of the algorithm, the helicopter was flown

through a closed loop course, located in the Physics Building, at the University of Maryland (scenario shown in Sub-Section 4.3.3). As in the case of the wheeled cart, the advantage of a closed loop course is that a seamless overlap between the start and finish areas implies a relatively low accumulated drift. This, in turn, allows for a successful scan matching between the latest laser scan from the end of the flight, and the virtual scan that includes parts from the previously mapped area, when the flight course started. Figure 5.25 shows the resulting map of the attempted closed loop course. The resulting map appears to be crisp, as the previous maps were, and the area where the loop is closed appears to be seamless. The course length is approximately 60 $m$, which is relatively long. This result is a strong testament as to the robustness and the accuracy of the proposed algorithm, on the most challenging platform of all three.

## 5.3 Algorithm Limitations

There is a large variety of scenario types where SLAM is required to work. However, in some cases, the algorithm may fail due to various reasons. Therefore, it is beneficial to have certain guidelines for predicting the algorithm's success.

### 5.3.1 Effect of Laser Measurement Noise

The effect of the laser measurement noise was examined by adding noise to an existing dataset. The noise characteristics is a random Gaussian noise:

$$\Delta r \sim N(0, \sigma^2) \tag{5.1}$$

where $\Delta r$ is the range noise that is added to the baseline range measurement for each beam at every laser scan (in $mm$). The noise for each laser beam is independent. These characteristics are identical to that of the laser's baseline noise (see Sub-Section 4.1.2), where $\sigma$, the standard deviation, can be extracted using the noise intensity (seeEq. (4.1)). The noise intensity (in %) was increased until the algorithm completely failed without being able to recover (in some cases, the scan matching algorithm may fail in some scenes but may be able to recover in later scenes so the SLAM process can continue). This examination was performed on a dataset collected by the single rotor helicopter aerial platform, mapping a single corridor (one sweep), in Martin Hall at the University of Maryland.

The results are presented in Figure 5.26, for the baseline laser noise and four additional simulated levels of noise. The threshold for the cost function that determines scan matching failure is set to 10 (marked by a dashed red line). The baseline laser noise case does show a single case of scan matching failure (at step number 25) but the algorithm was able to recover and the following scenes were scan matched with a sufficiently low cost. However, increasing the laser noise results in an increase in the final cost function values for all scenes, as expected, and the number of eliminated scenes increases as well. Finally, for the case of 3% noise intensity (brown line)

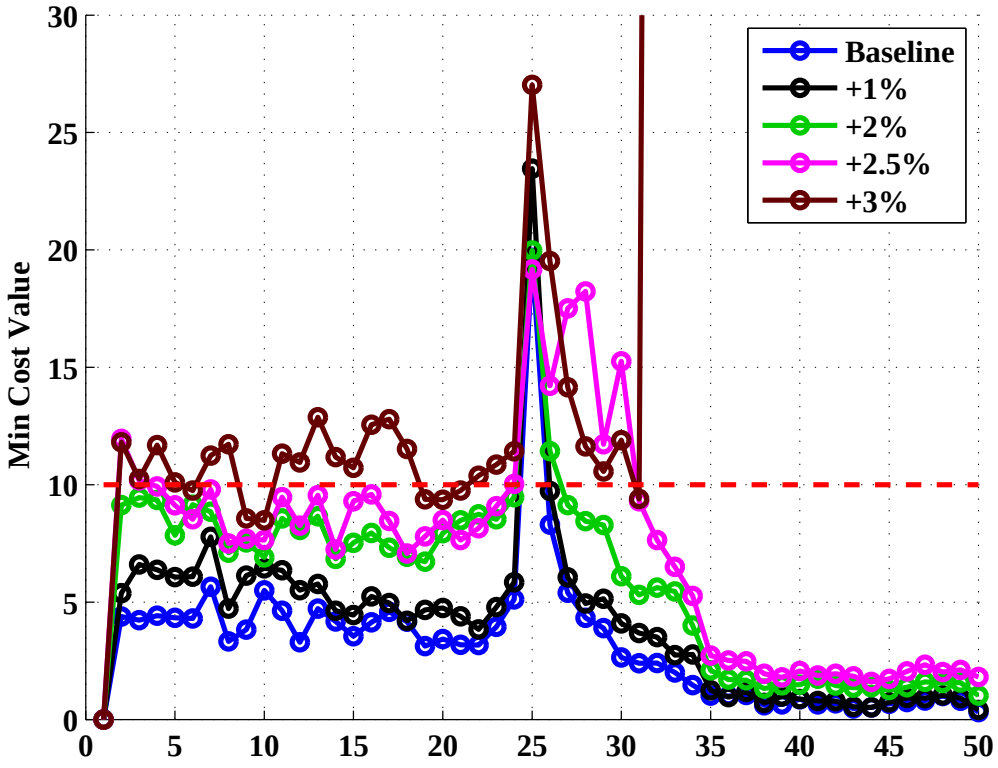


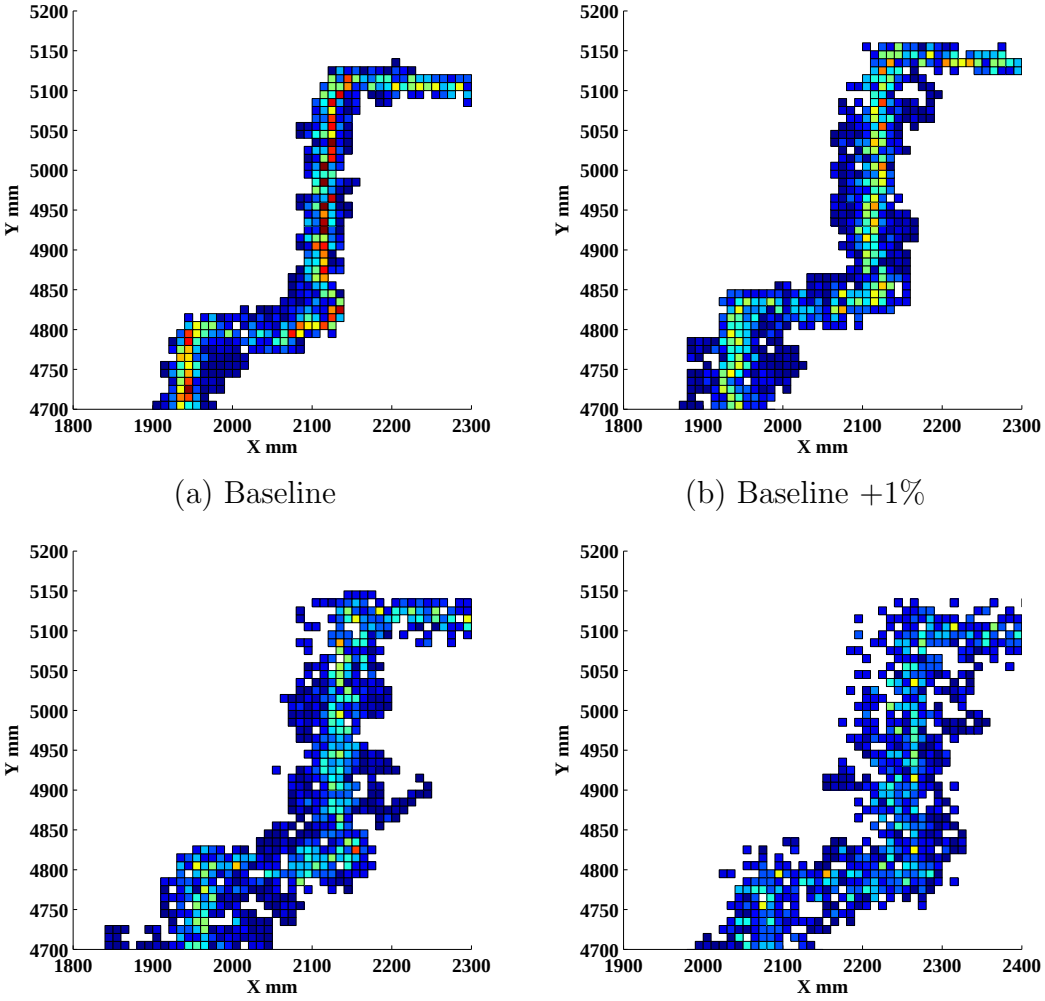

Figure 5.26: Effect of laser noise on cost. Showing the final cost value for case of an aerial platform flying down a corridor. The case was run with the baseline laser noise, and four values of added noise.

the algorithm exhibits a failure at step 31 from which is was unable to recover (as indicated by cost value rising outside the figure scope).

To show the effect of noise on the occupancy grid map quality, four close-ups on a part of the mapped corridor are shown in Figure 5.27 (note: the same colorbar is used in all four figures, where the warmest color represents an occupancy of 10). The increase in laser noise continuously deteriorates the sharpness of the map, as walls appear more spread out, and less crisp. The wall thickness increases from 5 to 6 cells, to over 10 cells, which appear blurry. For high levels of noise - the virtual scan algorithm would have a difficulty to find well defined objects due to the increased wall thickness. Moreover, the location of the vertical wall changes as the noise is increased (note that the scale is different on the last figure as the wall "moves" to the right).

### 5.3.2 Effect of Virtual Scan Resolution

The number of virtual points used, denoted by $n_V$, does not necessarily has to equal the number of laser points. On the contrary - using more virtual points may help represent the environment better in the virtual scan, as it may capture more




(a) Baseline

(b) Baseline +1%

(c) Baseline +2%

(d) Baseline +3%

Figure 5.27: Effect of noise on map quality. A single feature from a corridor map. Map resolution is 1 $cm^2$. (a) Baseline laser data (noise level of approximately 1%). (b) Noise level increased by 1%. (c) Noise level increased by 2%. (d) Noise level increased by 3%.

features, including relatively small features that may be missed due to low virtual scan resolution. The cost function searches for laser point matches for each virtual point, and so there is no restriction on the number of virtual points that are used. However, one must keep in mind the cost function complexity is linear with $n_V$, and so more virtual points would result in higher computational requirements.

The effect of the virtual scan angular resolution, which for a fixed field of view is controlled by $n_V$, can be shown by decreasing number of virtual rays (keeping all else constant) until the algorithm fails without recovery. For this purpose, a case of an aerial platform performing two sweeps up and down a corridor was utilized (100

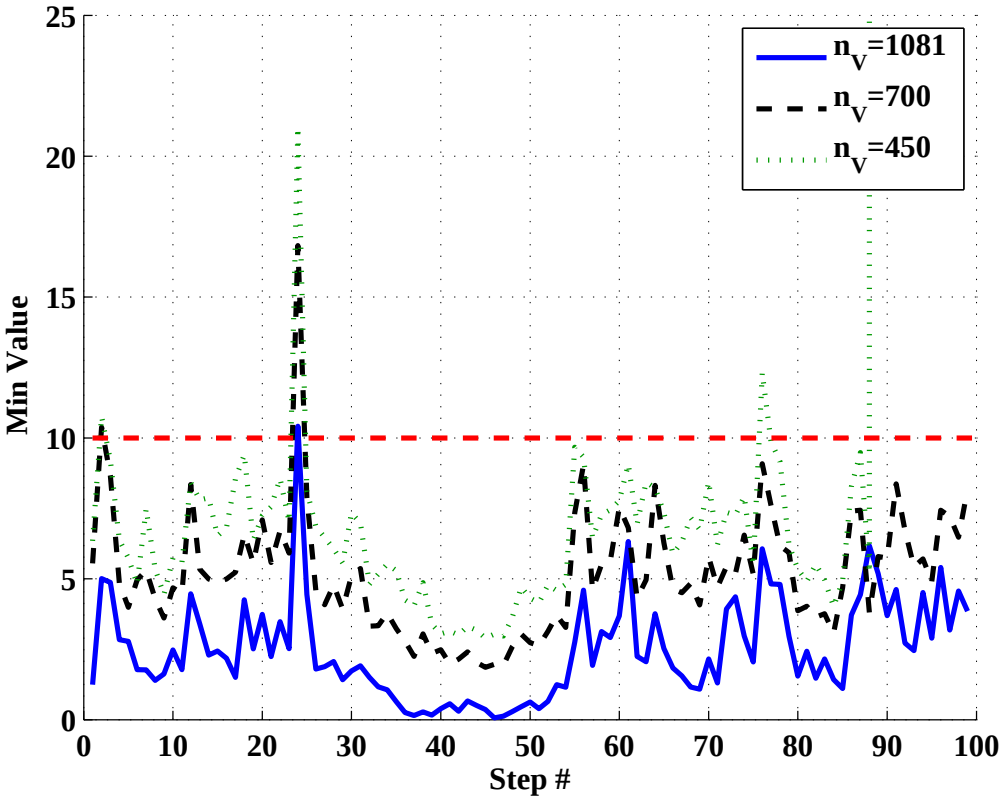



Figure 5.28: Effect of virtual scan resolution - plotting the final cost for the case of an aerial platform while reducing the number of virtual rays $n_V$ until the algorithm completely fails without recovery.

scenes in total). The final cost results for three values of decreasing $n_V$ are shown in Figure 5.28. The highest value is the same as the number of laser rays - 1081, and except for scan 24, all the scenes result in a cost that is lower than the set threshold of 10. Decreasing the $n_V$ result in an increases in the cost across all scenes.

For $n_V = 700$, the relative behavior of the cost (the shape of the graph) remains identical, as the laser scenes are identical, and the virtual scenes maintain some similarity. However, the cost is higher as the resolution used to represent the virtual map is lower and thus the fidelity of object representation deteriorates, and so the possible scan matching accuracy is reduced. In this case, it was found that $n_V = 450$ caused an unrecoverable error in the scan matching algorithm at step 89. Moreover, the shape of the graph appears to be different in several steps, resulting from significant differences in the way the virtual map is captured by the virtual scan.

Ideally more virtual points are desired. However, in some applications, when the environment is characterized by low clutter, a relatively low number of virtual rays may be used successfully.

### 5.3.3 Effect of Laser Scanner Parameters

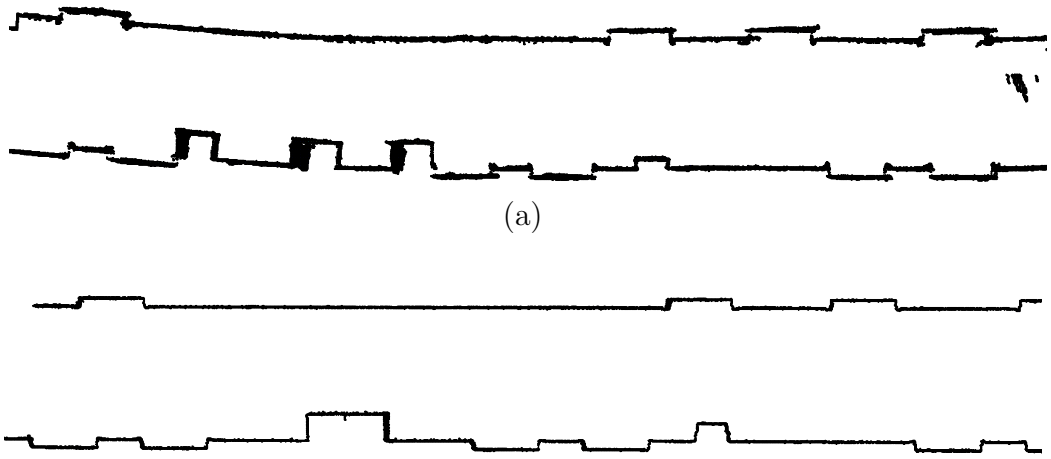
The algorithm's sensitivity to the laser scanner's specifications was examined by employing the algorithm on two laser scanners (Sub-Section 4.1), denoted as "higher end" and "lower end", while also isolating the characteristics that has the most impact on the results.

The range of the URG-04LX is quite low, and thus experiments in typical office like environments are challenging due to sparse amount of features that are within the laser's range. Hence, in the experiments conducted using the lower end laser scanner three boxes were placed along one wall in the area that was sparse in features.

A map for a single corridor is presented in Figure 5.29, using both laser scanners. Note that although it cannot be seen - the maps are in fact a collection of pixels which represent the occupied cells in the occupancy grid, color-coded with the relative occupancy. In this figure, each pixel represents a square with an area of 10 $mm^2$. The scenario is a hallway in the $3^{rd}$ floor of Martin Hall at the University of Maryland, approximately 15 $m$ in length. It is composed of walls, doors, door-posts, cabinets and several trash cans. Note: the number of virtual rays $N_V$ was unchanged in all cases.

It is quite clear that the map produced by the algorithm using the lower end sensor appears bent, and compared to the map produced using the higher end sensor it has thicker walls. One can also note some inaccuracies in the intersection of several door post features. Moreover, the distances between some of the doors appear to be different in both maps.

When comparing to the hand measured map, the map results using the higher end laser was found to be accurate to within 2 $cm$, which is considered a high quality map. In order to perform a fair comparison between the sensors, and isolate the cause for the bent corridor result, the higher end scanner was artificially limited to reflect the properties of the lower end scanner where possible. The dataset that is used is the same one that created Figure 5.29 (b).



(a)

(b)

Figure 5.29: Single corridor map. Typical office-like environment, mapped with two sensors: (a) Lower end URG-04LX-UG01 (b) Higher end UTM-30LX.

Table 5.1: Hokuyo UTM-30LX manufacturer specification.

| Scanner | URG04LX-UG01 | UTM30LX | Modified UTM30LX |
|---|---|---|---|
| FOV | 240° | 270° | 240° |
| Number of points | 683 | 1081 | 960 |
| Accuracy | $< 3\%$ | $< 1\%$ | $< 3\%$ |
| Range | 5 $m$ | 30 $m$ | 5 $m$ |
| Scan Frequency | 10 $Hz$ | 40 $Hz$ | 40 $Hz$ |

Table 5.1 shows a comparison between the two baseline sensors and the artificially limited UTM30LX scanner. The FOV was reduced to 240°, the range was decreased to 5 $m$, laser measurement noise was added to reflect the lower quality laser's 3% noise level. The number of points changes only due to the reduction in FOV, and thus the azimuthal resolution cannot be precisely matched with the lower end laser's. The data were collected with the laser scanner's rotating at $40Hz$ (which is not a user-controlled property), and therefore this is the main difference between the sensors.

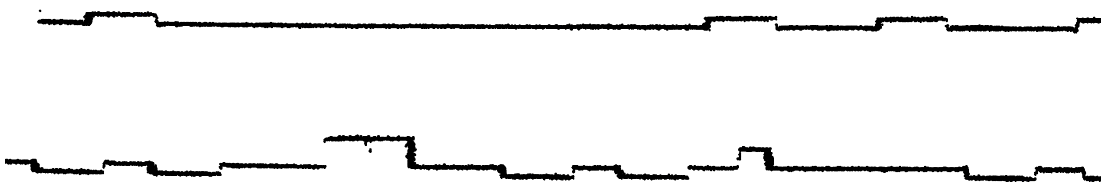


Figure 5.30: Single corridor map - using the simulated modified UTM30LX scanner.

The map results for the artificially limited sensor is presented in Figure 5.30. The walls appear thicker, as a result of the increased sensor noise (same as in Figure 5.29(a)); however, the overall quality of the map appears to be quite good, and resembles closely the map of Figure 5.29 (b). It can therefore be determined that the key difference between the two sensors (other than the range), that controls mapping quality is the Scan Frequency (the rotational velocity of the laser's internal mirror).

The physical reasoning for this result is quite simple. Since the platform moves while the laser is scanning the environment, the faster the laser mirror spins - the less effect the motion has on the scan result. A faster moving platform would results in a distorted scan of the surroundings and thus scan matching several distorted scans may result in a distorted and bent map. The effect is similar to the well known "Doppler Effect". An infinitely high scan frequency would results in a scan that is similar to that obtained when the laser is static. However, as the ratio between the platform's velocity and the laser scan frequency increases, the scan fidelity with respect to the surroundings deteriorates.

### 5.3.4 Effect of Occupancy Grid Resolution

The resolution of the occupancy grid determines the memory requirements of the algorithm. Compared to the memory space required to store the occupancy grid, the memory required to run the SLAM algorithm itself is negligible as it is $O(n)$, where $n = max(n_V, n_L)$. The memory requirement of the occupancy grid depends on the size of the mapped scenario. Most scenarios in this research were of the order of 30 $m$ by 15 $m$. Therefore, the memory space required to hold an occupancy grid with a 10 $mm$ by 10 $mm$ is of the order of $5Mb$, which is well within the capabilities of modern digital memory capacity.

Since the occupancy grid holds a single number per cell (a float), the memory space required to hold the occupancy grid data is $A_{grid}/A_{cell}$, which for typical cases in this

research is $O(10^6)$ or approximately $O(1MB)$. This complexity may be considered as a disadvantage as compared to feature based maps, which only store the locations of detected features, and thus have considerably lower space requirements. It is therefore desired to examine the algorithm's performance with lower resolutions so as to allow operation in larger areas, and lower memory requirement in general.

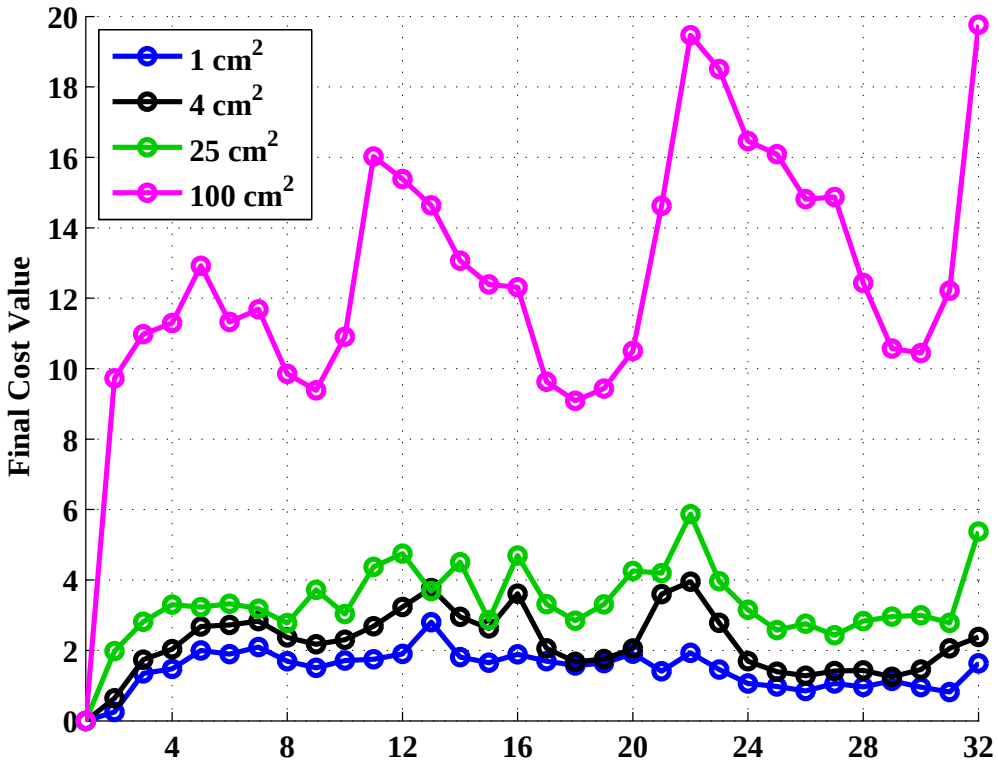



Figure 5.31: Effect of occupancy grid resolution on the final cost.

Figure 5.31 presents the final cost over 32 scenes that form a single corridor (dataset obtained using a ground platform). The algorithm was tested on four occupancy grid resolution values of 1 $cm^2$, 4 $cm^2$, 25 $cm^2$, and 100 $cm^2$, while all other parameters were kept constant (including the perimeter matching acceptance threshold). As expected, the final cost values increase with the reduction in OG resolution. The difference between 1 $cm^2$, 4 $cm^2$, and 25 $cm^2$ appears to be quite small, and considering the threshold for final cost acceptance - all three OG resolution values result in acceptable cost values.

However, the final cost values using an OG resolution of 100 $cm^2$ appears to be significantly higher than the rest. This is the results of an internal parameter in the algorithm where for relatively short ranges - the acceptance of a match into the perimeter matching term is performed based on a prescribed value rather than the laser noise model. In the above presented experiments, the threshold was set at 50. Thus, the relatively poor resolution of 100 $cm^2$ caused a large amount of points to

be excluded from the perimeter matching term, thus increasing the final cost value significantly.

The resulting maps using the different OG resolution values are presented in Figure 5.32, along with close-ups on the lower-left corner of the mapped corridor in Figure 5.33. All the maps appear to feature straight walls, however, when using coarse resolutions, several fine details can no longer be adequately represented (especially the door post details). When examining the close-ups, one may notice that the position of both the vertical and horizontal walls remain unchanged at 1850, and 1630 (within the capabilities of the different resolutions). These results are impressive as the resolution was decreased by two orders of magnitude from 1 $cm^2$ to 100 $cm^2$ (and as a result so did the error in object representation). It serves to show that the algorithm is more robust to OG resolution that it is to laser sensor noise, where wall positions tend to change significantly with increasing laser noise (see Sub-Section 5.3.1).

### 5.3.5 Failure Modes

Several algorithmic failure modes were detected during the course of this work. These are described and explained below. Additionally, suggestions for mitigating these failure modes are proposed and discussed. These failure modes are not part of the algorithm limitations described in Sub-Section 3.5.4.

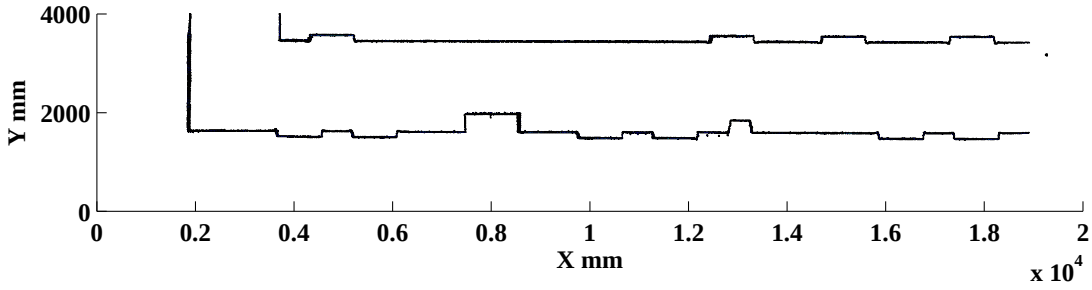
#### Repetitive Structure

Some scenarios may include repetitive structure that may result in the cost function having multiple minima. An example of such a scenario is given in Figure 5.34. Since the box-like structure is repeated, in cases where the search grid for cost minimization is large enough to cover the repeated structure, more than one minimum point may be found. This may lead to a possible failure when the wrong match may have a slightly lower cost, and thus will be mistaken for the right solution. A possible method to mitigate this failure mode may be to have a better initial guess based on additional platform sensors. This may allow the reduction of the search grid size and thereby reduce the chances of multiple possible matches with the repeated structure.

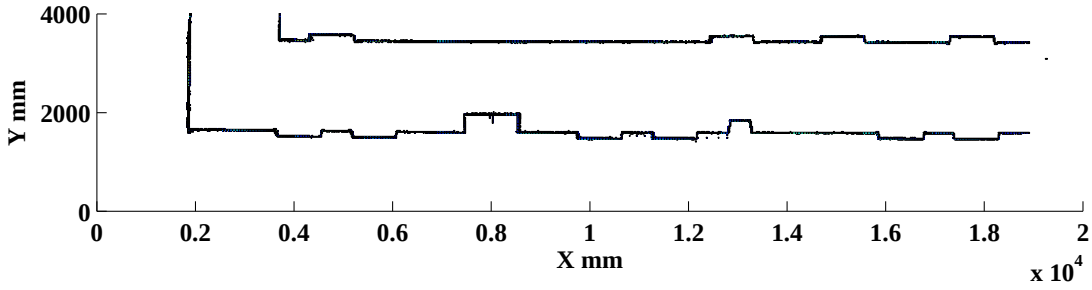
#### Insufficient Features

Insufficient features in a scan may result in a failure of the scan matching process. The first example involves a scan of two long corridors, shown in Figure 5.35. The scan has two main parallel features along one dimension. These features may be used for position estimation using scan matching along that dimension. However, the scan has almost no features in a direction that is different than that of the two main walls. This may lead to a failure to find a unique scan matching solution, as the two main features may be aligned in multiple location along their principal direction.

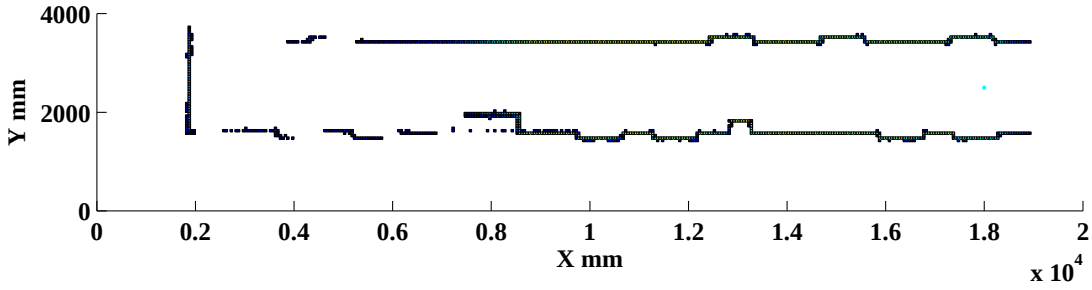
The second example involves relatively fast platform motion, where two subsequent laser scans are taken while the platform has traveled a relatively large distance, resulting in little overlap between the scans. Such a case may occur when turning



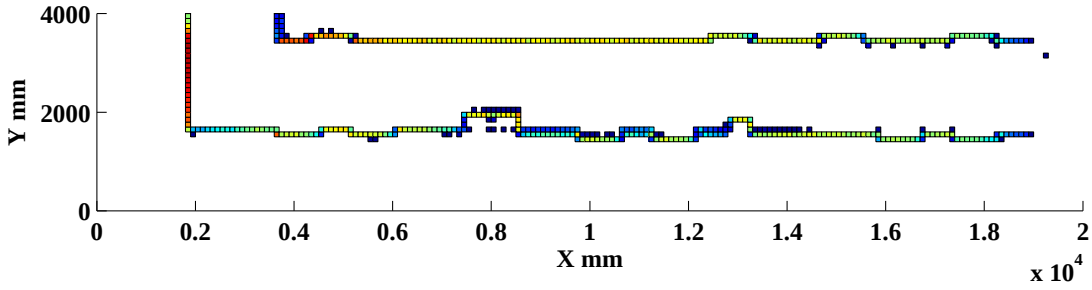

(a) 1 $cm^2$




(b) 4 $cm^2$

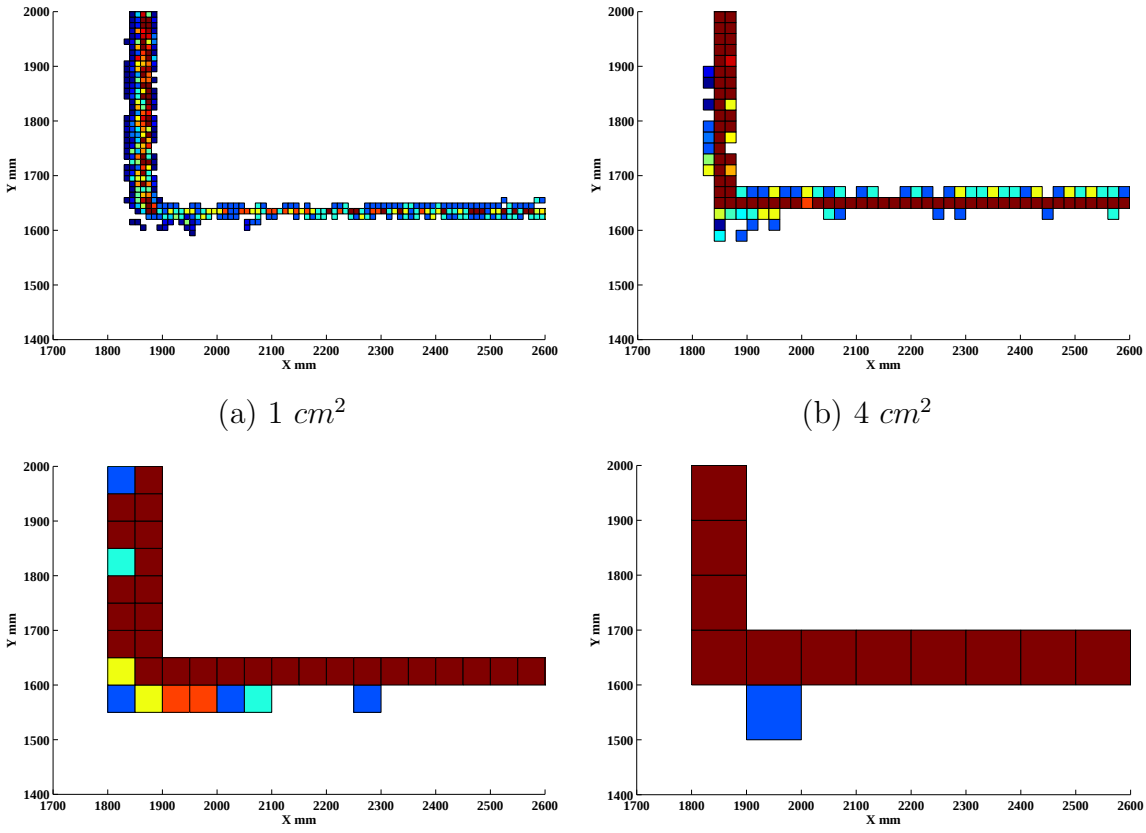



(c) 25 $cm^2$




(d) 100 $cm^2$

Figure 5.32: A part of a ground platform experiment, showing a single corridor, run with increasing occupancy grid resolution.




(a) 1 $cm^2$

(b) 4 $cm^2$

(c) 25 $cm^2$

(d) 100 $cm^2$

Figure 5.33: A part of a ground platform experiment, showing closeups on a single corridor map, run with increasing occupancy grid resolution.

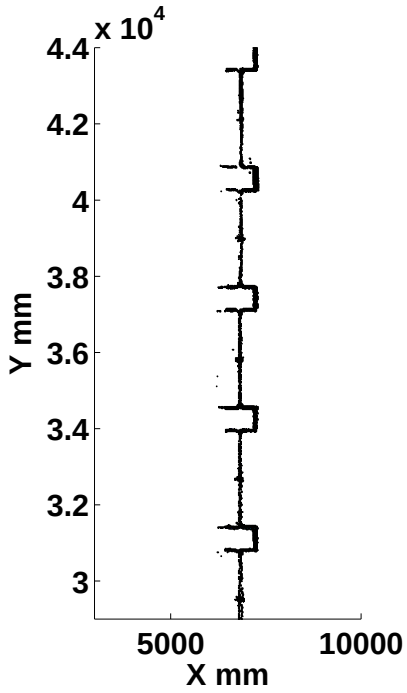



Figure 5.34: Example of a repetitive structure.

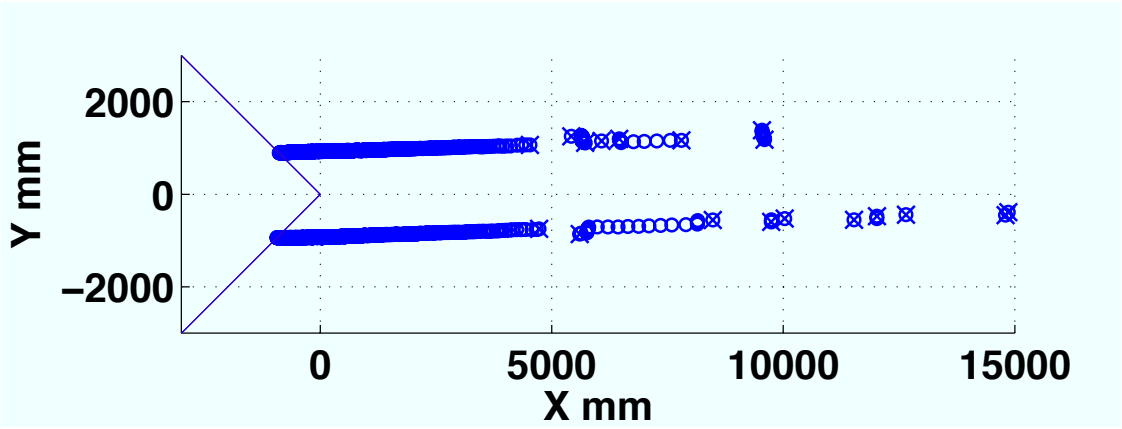



Figure 5.35: Failure mode for a scan with only two long corridors. The scan has features along one dimension, but almost no features along the other dimension, making it less likely to have a unique scan matching solution.

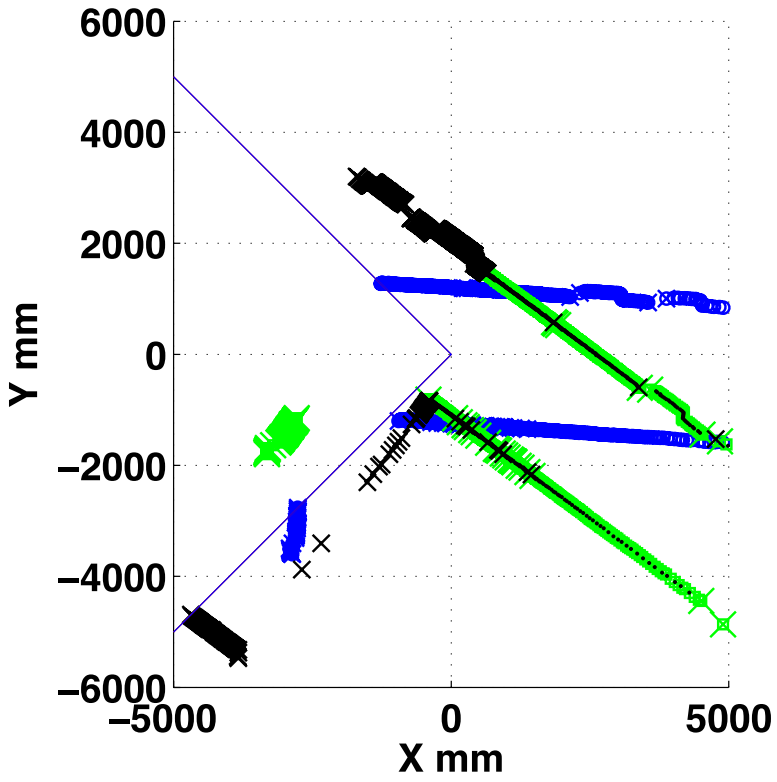



Figure 5.36: Moving too fast, causing a reduction in usable scan overlap, which may eventually lead to a scan matching failure.

around corners, as shown in Figure 5.36. The new laser scan, taken after the corner has been passed, contains a lot of new information that is not yet contained in the map. Thus the virtual scan and the new laser scan have little overlap, causing a failure in the scan matching process.

In the third example, a scan scenario would contain very few well defined features. Figure 5.37, presents an example of such a scenario, where a laser scan contains a single well defined wall on one side, but a completely unstructured scan of a large outdoor bush on the other side. Every scan of the bush would appear different, as the bush has no well defined structure to it, and thus different clusters of points will be shown in each scan. Subsequently, scan matching would not produce usable solutions. Unfortunately, an environment with not enough well defined features in it would eventually cause the algorithm to fail.

### Significantly Non-2D Scenes

The algorithm is based on a 2D laser scanner, and as such it primarily assumed a 2D environment. Nevertheless, the algorithm can process scenarios with some non-2D obstacles, to a certain extent, as shown and discussed in the examples above. However, a scenario which is comprised mostly out of non-2D objects may cause a

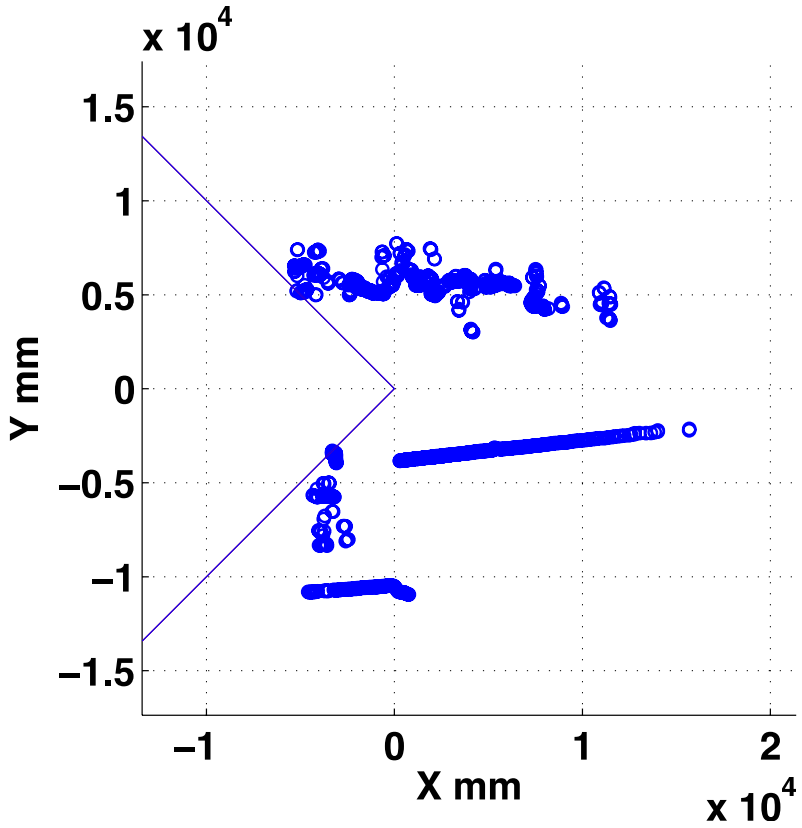



Figure 5.37: A partially featureless laser scan. There is only a single wall (right hand side of the scan), while the rest is a scan of a large bush.

failure of the algorithm. Examples include scenarios with sloped surfaces (as each scan may capture a different level), certain vegetation (as in Figure 5.37), multiple objects of different heights, etc.

### High Pitch/Roll Angles

During the experiments, a possible 2D assumption violation was caused by the pitch and/or roll motions of the platform. These motions change the measured objects, as the laser may pick up different objects from different levels of the environment.

The algorithm has two features that try to overcome such violations. The first is the occupancy grid resolution itself: a typical laser beam with $r = 5000\ mm$, allows pitch/roll angles of up to 3°, before it's measurement is registered to a different cell. However, even if some pitch/roll angles are larger than 3°, the measurements is still registered to nearby cells. Hence large pitch and roll angles would not necessarily cause a failure of the algorithm. It was found that for the above presented scenarios, angles of up to 10° could be sustained in some cases (as long as the pitch/roll angle do not significantly change the scanned environment's shape).

The second feature is the elimination of contributions for which $F_i > T_E$, where

$T_E$ is the elimination threshold explained above. In some cases different obstacles may be scanned, as explained above. In such a case, the scan matching algorithm attempts to match two different scans that have too little overlap of identical features. In most cases, eliminating some of the points that represent two different obstacles (*e.g.*, boxes, pillars, etc.), can help reduce the cost function to an acceptable value. The threshold set was $T_E = 1000\ mm$, which represents a typical range difference in most real life scenarios.

## 5.4 Results Using Existing Datasets

### 5.4.1 Comparison With Existing Full Scale Datasets

The algorithm was also successfully tested on two previously published full scale datasets. Both laser scan datasets were collected using a wheeled ground platform (whether autonomous or driven by a human operator), in a typical office-like environment over a relatively long traveled distance.

As detailed below, the two datasets made use of significantly different laser scanners, varying both Field Of View (FOV), range, accuracy, and the number of laser points per scan. The first dataset was collected by Diosi and Kleeman [29] using a 2D laser range scanner producing 361 points with a FOV of 180° (and thus an angular resolution of 0.5°), recording data at 30 $Hz$. The second dataset was collected by Andrew Howard [85], with a laser scanner producing 180 points, with a FOV of 180° (and therefore, an angular resolution of 1°).

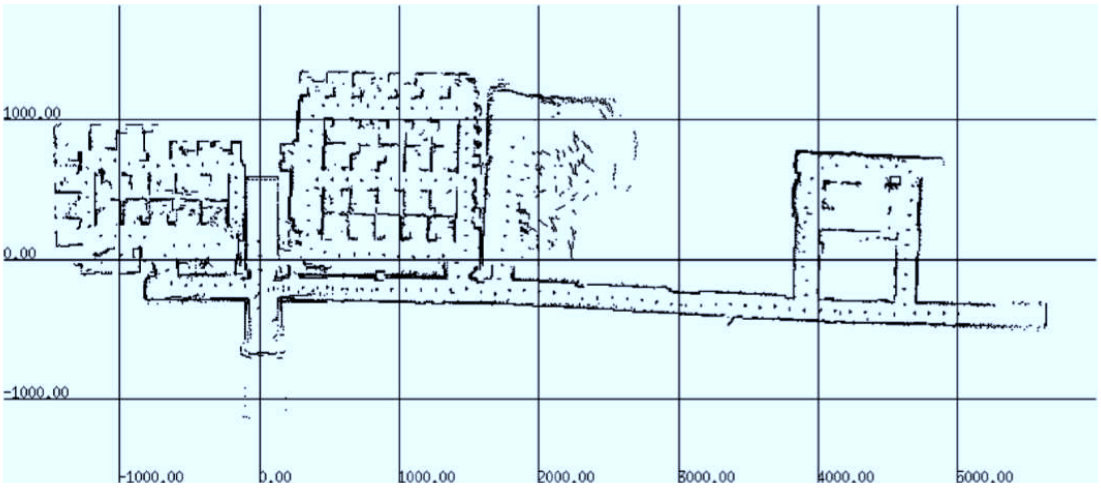
It is important to note that other than laser sensor parameters, none of the algorithm parameters stated above were altered, in order to use these previously published dataset. This is with the exception of increasing the final accepted cost value from $T_F = 10$ to $T_F = 15$, to allow for some anomalies caused by moving objects in both scenarios.

#### Monash University Database

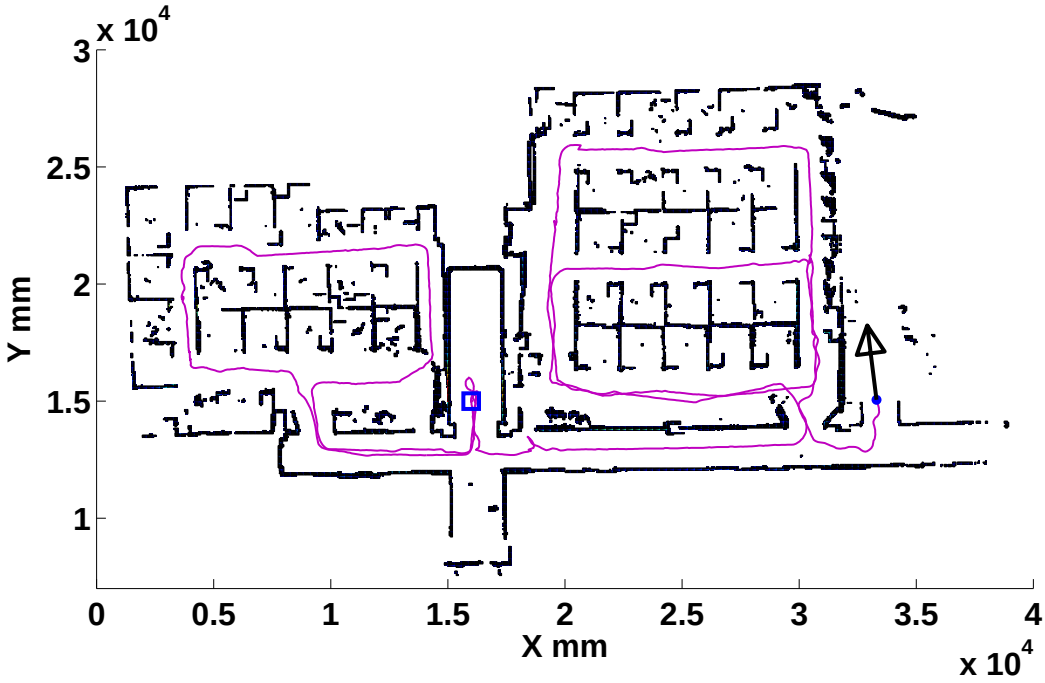
The first dataset was collected by Diosi and Kleeman [29] to test the performance of their scan matching algorithm within their own EKF-based SLAM algorithm. Their results are presented in Figure 5.38 (a). The experiment was conducted in an office like environment, using a ground platform and a 2D laser range scanner producing 361 points with a FOV of 180°, recording data at 30 $Hz$.

This dataset is described by Diosi and Kleeman as less than ideal, as the laser sensor picked up more than 10 people walking in its FOV, and several doors were opened and closed during the traversal of the environment. Moreover, according to them, the use of a loop closure algorithm is challenging by the relatively large amount of repetitive structures (the office cubicles in both the first and second rooms), as the algorithm may produce false positive loop closure events.

Nevertheless, the current algorithm was employed on the same dataset, and produced highly accurate mapping results as seen in Figure 5.38. The map appears to be crisp with a significantly less occurrences of "double walls", as compared with




(a) Results by Diosi and Kleeman, using EKF-SLAM, PSM, and loop closure algorithm (grid lines spacing is 10 $m$, dimensions on the left axis in $cm$).




(b) Results using PB-PSM scan matching only (units in $mm$). Start and end points are marked by a blue square and dot, respectively. Final heading arrow shown in black. Traveled path is drawn as a purple line.

Figure 5.38: Scenario exploration by Diosi and Kleeman, using their Monash University's dataset [29].

the results of Diosi and Kleeman. It is noteworthy that PB-PSM has no issues with repetitive structures such as the cubicles that populate the $1^{st}$ room. Issues such as the dynamic environment were easily solved by increasing the set threshold for the overall successful scan-matching to a value of $T_F = 15$. The robotic platform is said to trip twice over a 1.5 *cm* cable protector on the floor, which also contributed to the low quality of the laser scan dataset.

However, the algorithm failed without recovery when the $3^{rd}$ room was reached. The laser scans in that room became quite sparse with very few points that can be successfully utilized for scan matching. The results by Diosi and Kleeman also show very poor mapping quality in the $3^{rd}$ room, with almost no usable mapped features.

#### University of Southern California Database

The second dataset was generated by Andrew Howard, and was obtained through the online Radish repository [85]. The experiment was conducted using four autonomous robots individually exploring a closed area (with some human supervision). Several moving objects (people and/or robots) were captured in the dataset, thus rendering it quite challenging to obtain good results. A more detailed description of the experiment and the algorithms used to obtain the results can be found in the work by Howard et al. [86]. In this dataset, a laser sensor with 180 points was utilized, with a FOV of 180°.

The map obtained by Andrew Howard is presented in Figure 5.39 (a), along with the map generated by the present PB-PSM algorithm in Figure 5.39 (b), using the dataset from robot #2. The mapping results appear to be in excellent agreement. Note that in this case the amount of points in each scan is significantly smaller, especially when it's compared with the in-house results with the Hokuyo UTM-30LX, which boasts 1081 points per scan.

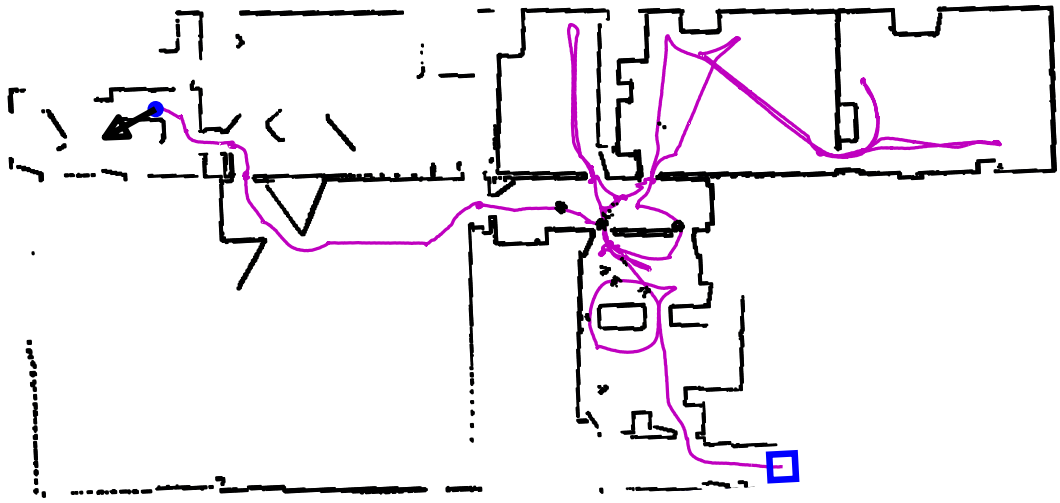
## 5.5 Outdoor Experiments

The algorithm was tested on several outdoor scenarios. Outdoor scenarios significantly differ from indoor office type environments. These scenarios typically contain more non-2D objects. Moreover, the laser sensor range is more limited in outdoor environments, as the reflected laser energy is diminished by the outdoor lighting. Several experiments had to be conducted at night, to overcome the limited range problem, and capture enough obstacles. This section describes the results obtained in several outdoors scenarios. The scenarios themselves are described in Section 4.3. Data from all the outdoor scenarios was collected using the human platform (walking person).

### 5.5.1 Kim Engineering, UMD: Front area

The algorithm was employed on the front area of the Kim Building at the University of Maryland, described and shown in Sub-Section 4.3.2. The resulting map is

(a) Results by Andrew Howard et al. Occupancy grid color coded with grayscale for occupancy values between 0 and 1 (image copied from the online Radish dataset repository [85].



(b) Results using present PB-PSM scan matching only. Start point is marked by a large blue square, end point is marked by a blue dot, with the final heading arrow in black. The traversed path is drawn as a purple line.

Figure 5.39: Scenario exploration. Andrew Howard's database [85].

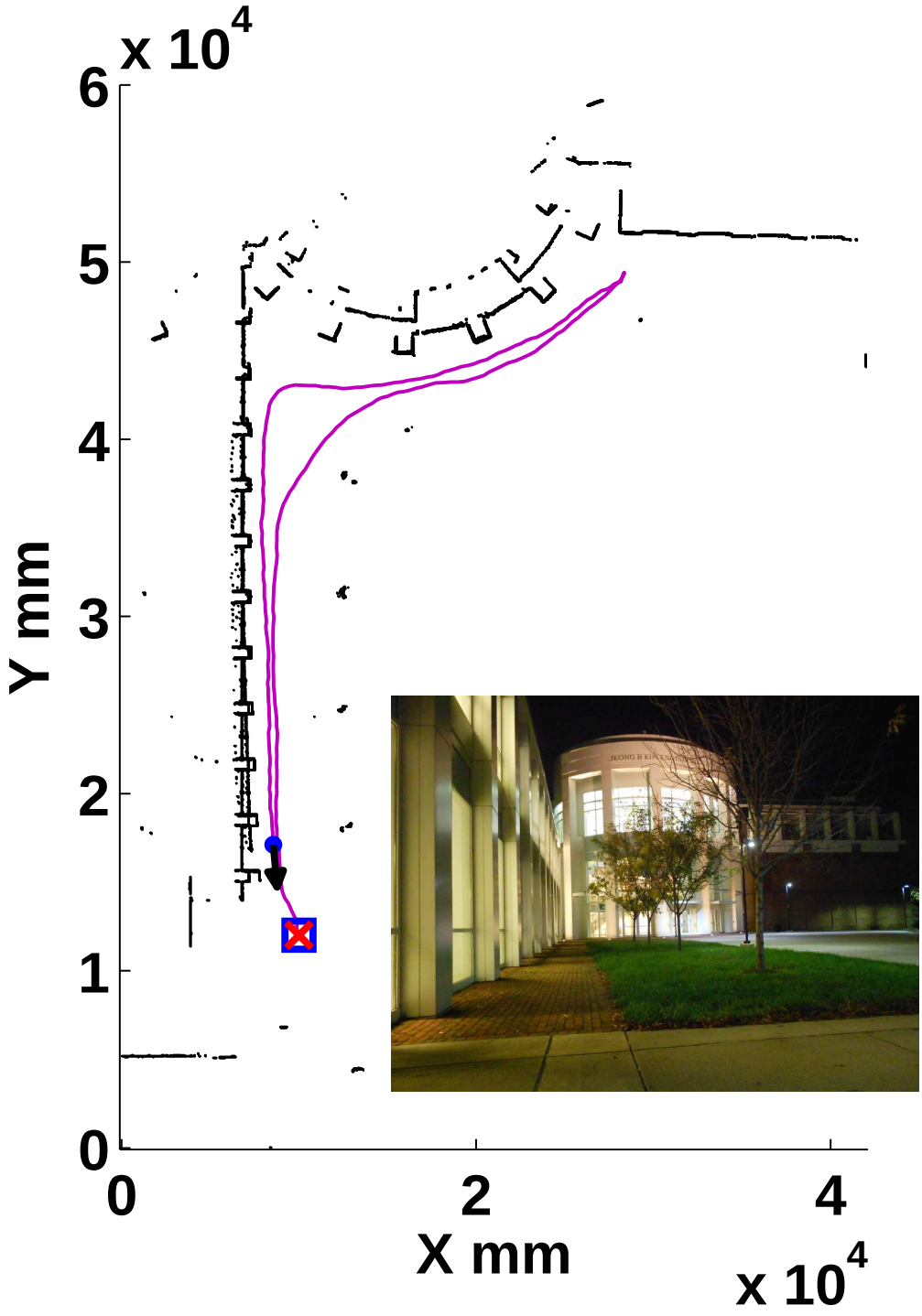



Figure 5.40: Kim Engineering Building, front area map.

presented in Figure 5.40, where the start position is marked with a blue square with a red 'x' mark inside of it, and the final position is marked by a blue dot, with the final azimuth shown by the black arrow. The traveled path is shown using a purple line, connecting all the position estimates. The total traveled distance was approximately 100 $m$.

The algorithm was successful in mapping this scenario, despite the many challenges, including the glass walls and the repetitive features along the left wall. The mapping results even show that the algorithm captured the relatively thin trees located in the area (which show up as clusters of points).

### 5.5.2 Kim Engineering, UMD: Back area

The algorithm was successfully employed on the back area of the Kim Building at the University of Maryland, described and shown in Sub-Section 4.3.2. The mapped area is presented in Figure 5.41 (same markings as above for the start/end positions, final azimuth and traveled path). The total traveled distance was approximately 70 $m$.

As seen in Figure 5.41, the mapped area appears crisp, and show many of the features that can be see in the pictures. These include the small yellow poles around the large gas tank. The experiment was stopped when facing two parallel walls, that were longer than the laser's range, as it prohibited a unique scan matching solution (see the failure mode described in Sub-Section 5.3.5).

### 5.5.3 Greenbelt Park, MD

The scenario of Greenbelt Park is considered to be a highly cluttered, and challenging environment, mainly because of the multiple objects (tree trunks and branches), and the lack of large coherent structures (such as walls). Moreover, all the trees have the same round shape, with very small diameter differences (which may pose a significant difficulty for appearance based loop closure algorithms).

The resulting map is presented in Figure 5.42, where several trees may be seen as clusters of occupied cells. Some some trees show their round shape, others were only partially captured, and therefore show only a small part of their frontal area. The traveled path in this case was created as part of an obstacle avoidance experiment (hence the slalom shape).

### 5.5.4 Northwestern High School, MD

The Northwestern High school scenario was chosen because it provides an opportunity to inspect a closed loop course, traveled between several structures (same as the case of the Martin Hall scenario Sub-Section 4.3.1). Pictures of the scenario may be seen in Sub-Section 4.3.4. In addition to the structures, several trees and bushes were also captured by the laser scanner during the experiments of this dataset.

The resulting map of the Northwestern High scenario, obtained by a closed loop course is presented in Figure 5.43. The start and end positions nearly overlap each

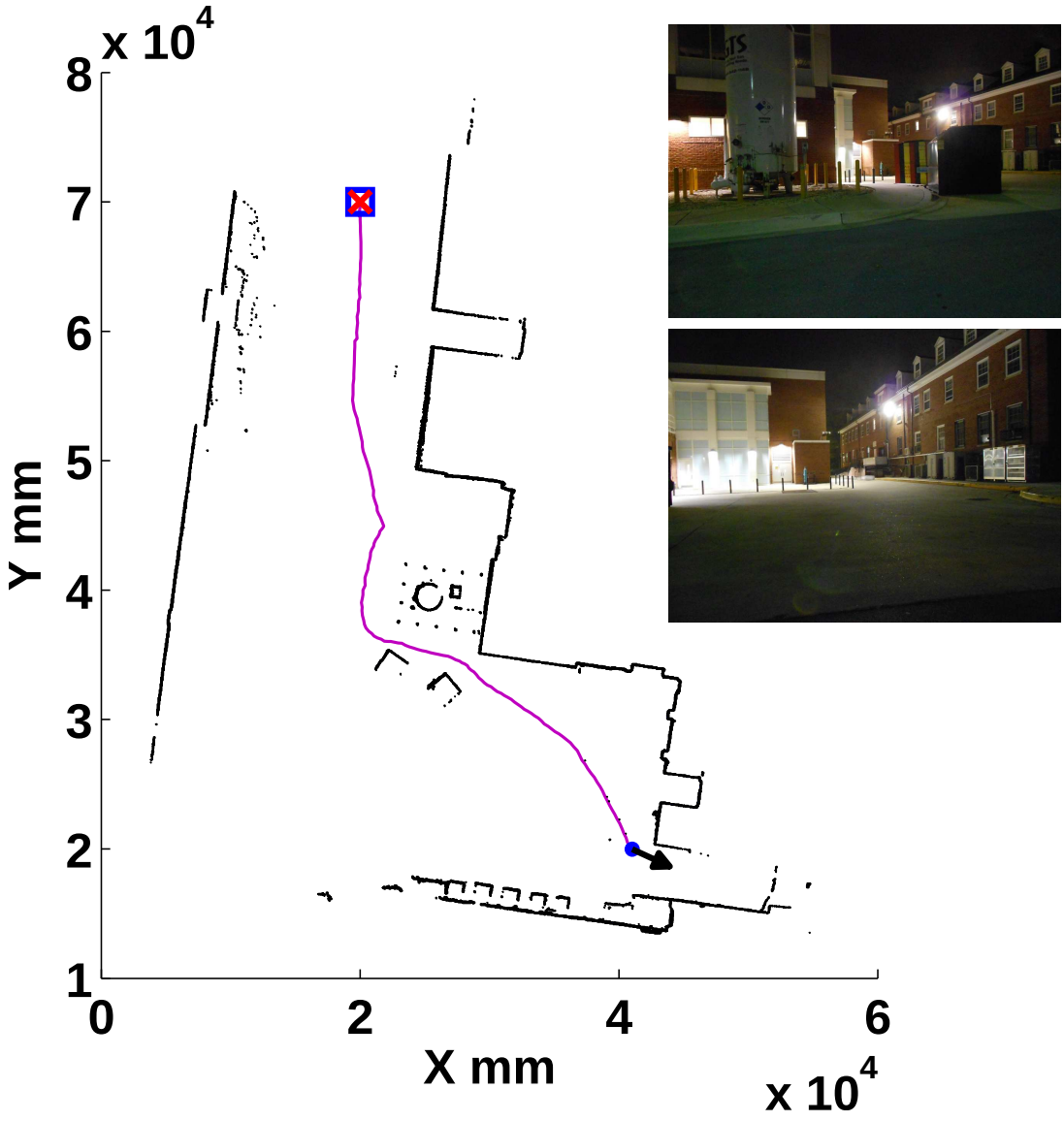



Figure 5.41: Kim Engineering Building, map of the back area.

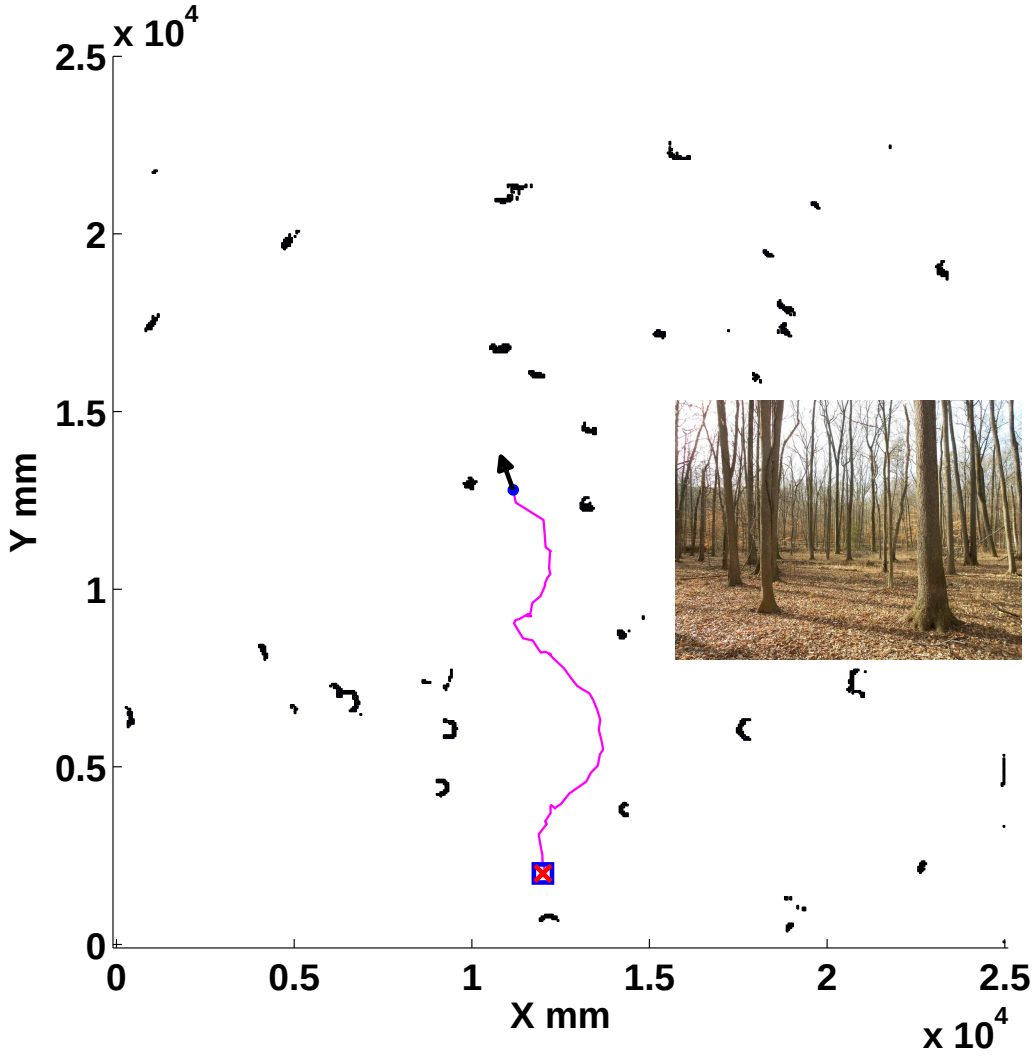



Figure 5.42: Greenbelt Park - forest environment.

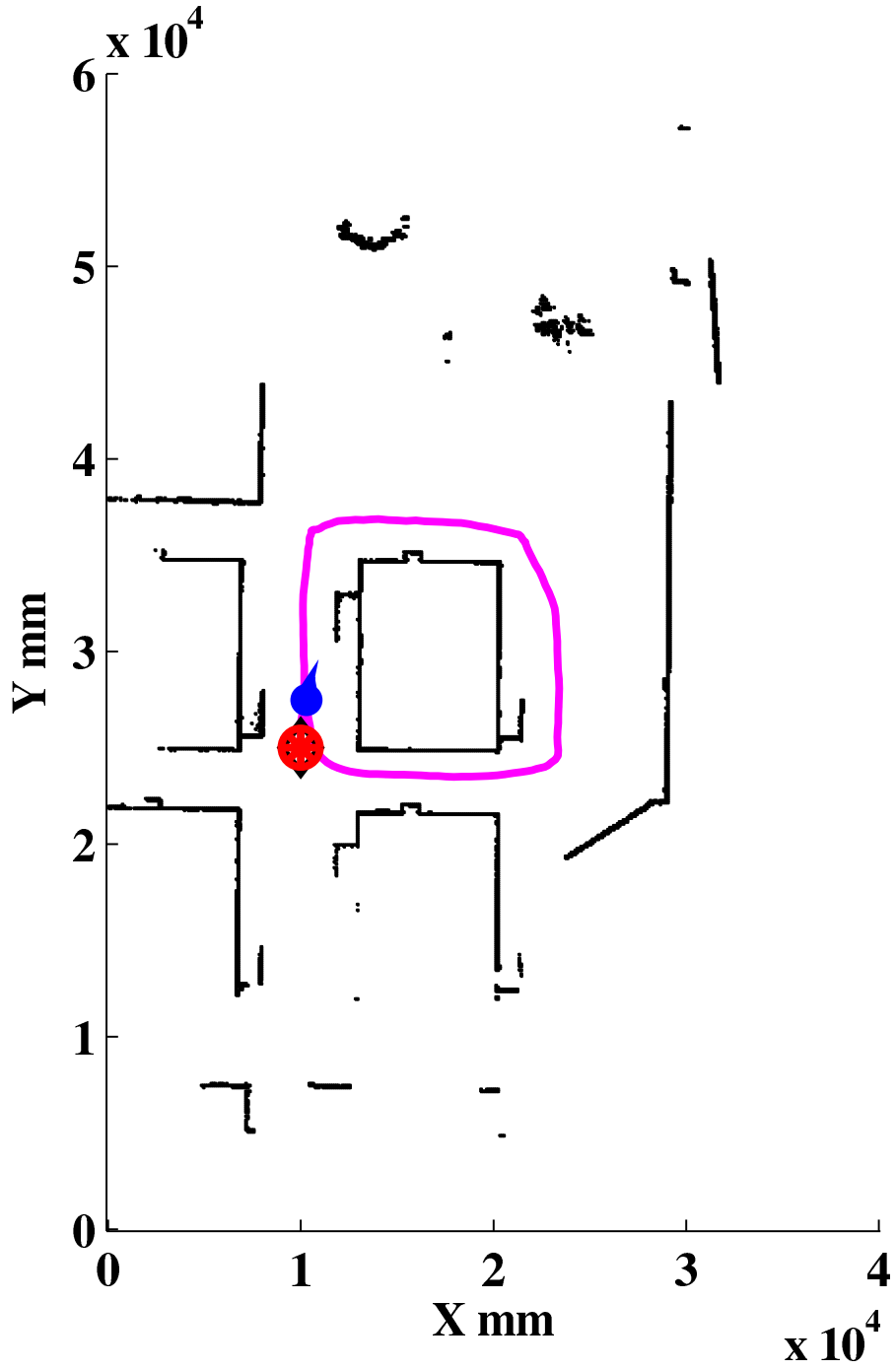



Figure 5.43: Map of Northwestern High School, closed loop course using a human platform

other, and the traveled path is shown as a purple line (clockwise traveled direction). All the walls appear to be crisp and well defined, and obstacles in the overlap area between the beginning and the ending of the course overlap seamlessly. No apparent drift is seen in the map.

### Varying Speed

This scenario was attempted several times using, collecting data at different velocities. Figure 5.44 presents four different mapping results, with the platform traveling at increasing velocities. As evident in the figure, all the resulting maps appear similar. The maximum speed is 100 $cm/s$, which is similar to the maximum speed attempted in the Martin Hall scenario. At this speed, the laser still rotates fast enough so the limitation on the platform velocity is not yet met (*i.e.* the laser scan remains a good approximation for a static scan of the environment. See Sub-Section 5.3.3 for more details).

### Path Independency

The last experiment with a human platform was conducted to show the independence of the results of the traveled path. In some cases, SLAM algorithms were coupled with the path planner to maximize the mapping output quality. However, in scan matching based algorithms, such as the one proposed in this thesis, there is no need for such a coupling.

Figure 5.45 presents two map results of the Northwestern High School scenario, obtained using fundamentally different traveled paths. The first path may be described as a very "Structured" traveled pattern, while the other path may be described as "Unstructured", showing erratic motion patterns which also contain two complete 360° spins. Despite the significant difference between the two paths, the maps appear very similar, and the "Unstructured" path still maintained a seamless overlap between the start and the end of mapped area.

## 5.6 Path Planning and Obstacle Avoidance

The path planning experiments present one application of the proposed SLAM algorithm in a targeted flight mission. This application also requires the SLAM algorithm to perform in real time, providing both map and position estimates for use by the path planner. The results presented in this section are in the form of evolving maps and traveled paths, but include the planned path as well. The algorithm works sequentially, a laser scan, followed by a virtual scan, scan matching, map update and path re-planning. No delay is introduced to maintain a constant SLAM estimation rate. The average time per a complete step was approximately 1 second. The primary focus is on targeted flight, and successfully navigating an MAV to a pre-defined goal position. The resolution of the OG was set as 10 $mm \times 10$ $mm$. The map that is used by the A* algorithm is merely a coarse version of the fine occupancy grid, generated by the SLAM algorithm. The A* OG resolution was set as 1000 $mm \times 1000$ $mm$.

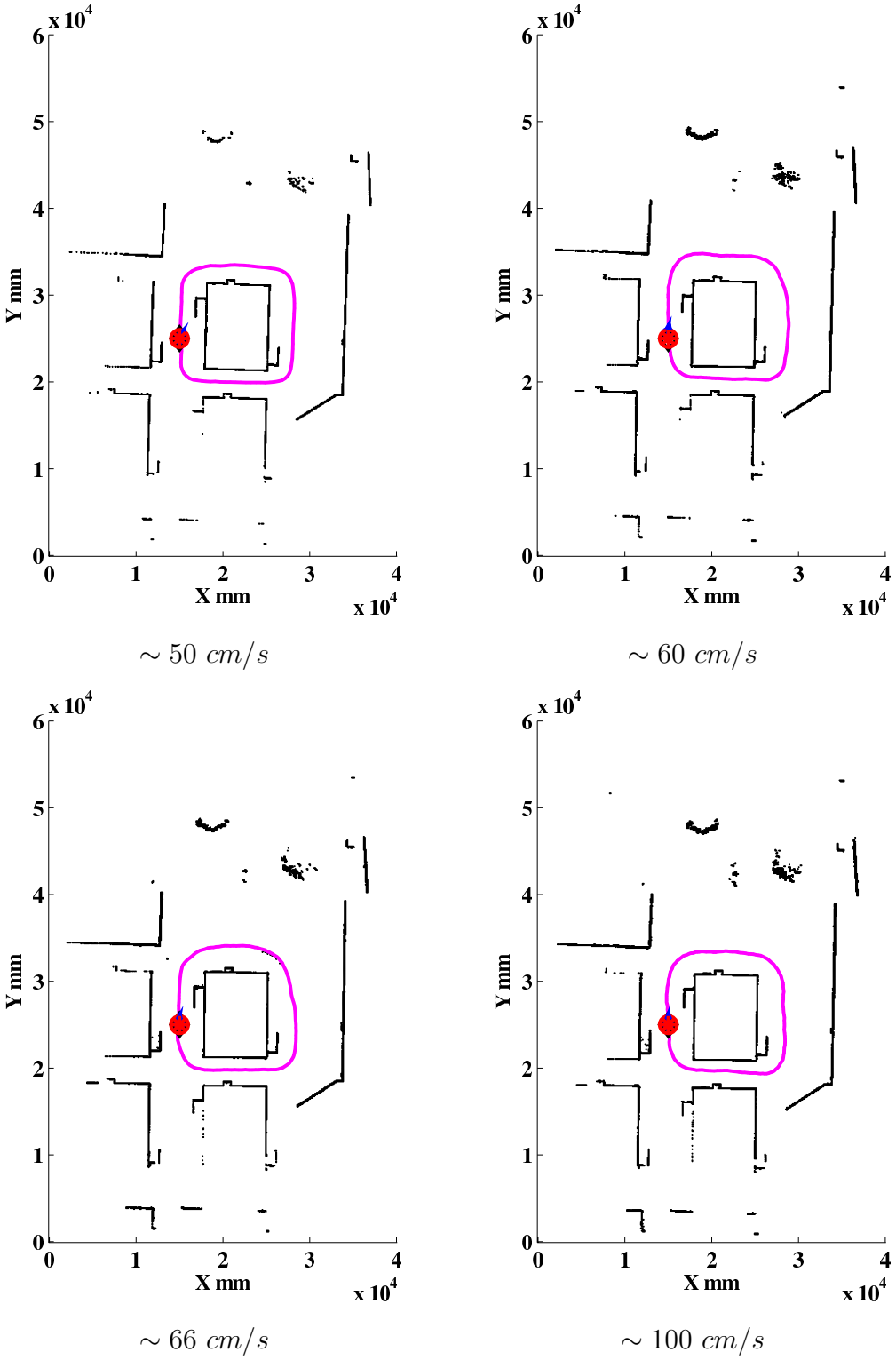
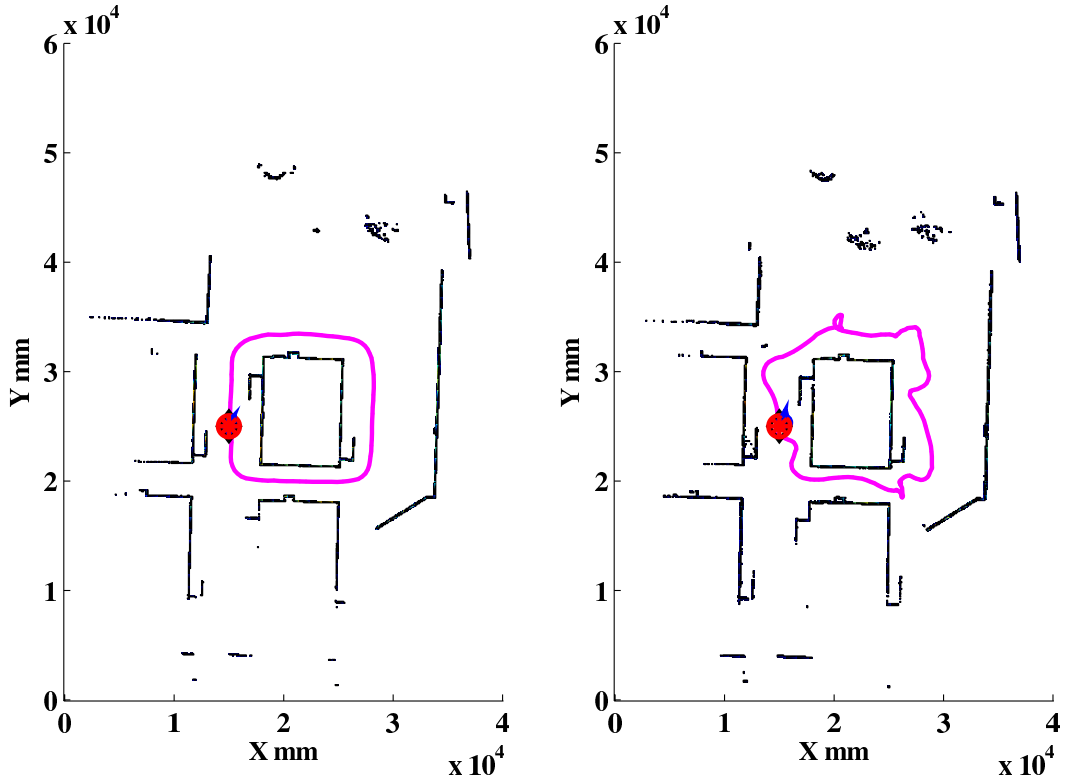



Figure 5.44: Mapping insensitivity to motion velocity. Showing the mapping results of several experiments, conducted in the same Northwestern High scenario, at various speeds. All the results show similar maps.




"Structured" motion

"Unstructured" motion

Figure 5.45: Mapping insensitivity to traveled path. Showing similar map obtained with "structured" motion, and a very "unstructured" motion.

### 5.6.1 Targeted Flight - Outdoors

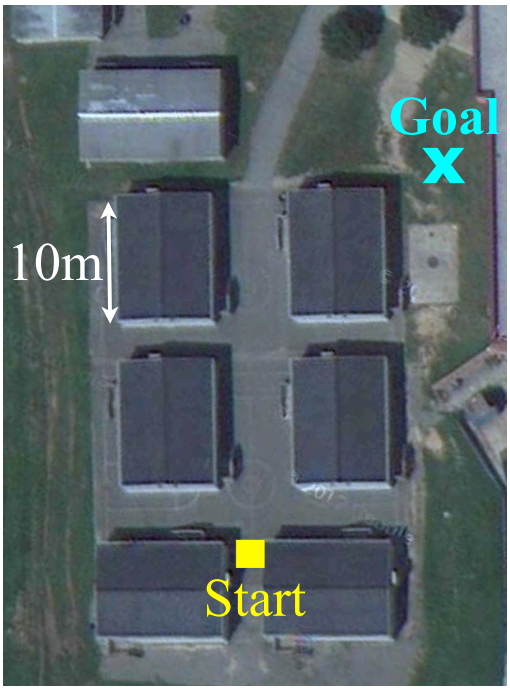



Figure 5.46: Start and goal position in the Northwestern High School scenario.

The first targeted flight experiment was performed outdoors, at Northwestern High School, in Hyattsville, Maryland. The layout of the scenario is presented in Figure 5.46, and four representative pictures are shown in Figure 4.13. The area chosen features several structures, outer walls, and trees. It allowed for easy flight of the helicopter, as all passages were wide enough (unlike the Marin Hall scenario).

The pilot was not given any information as for the goal location or the scenario layout neither prior, nor during the experiment. Once airborne, the pilot was only shown the helicopter's heading arrow on a screen, which he had to maintain pointing "North" (up), in order to follow the desired heading direction. The pilot was asked to continuously correct the helicopter's heading accordingly, as best he can, while also moving the helicopter in that direction. The final approach towards the goal was achieved by reducing the size of the arrow to reflect the actual distance from the goal. The experiment was completed when the helicopter's position was within 50 *cm* of the desired goal position.

The algorithm itself is also only given the initial and goal position, in it's own coordinates. As the helicopter moves, the SLAM algorithm generates the map and position estimates, while the A* is invoked to recalculate the optimal path every three steps, or if the path becomes obstructed (which is checked every step). The heading information presented to the pilot is updated based on the most current information at each step.

Figure 5.47 presents several snapshots of the fine occupancy grid, and the A* occupancy grid, showing the evolution of these maps at key points along the flight

trajectory. The initial and final positions are marked by a blue '□' and a red 'x' respectively, the helicopter's flight path is shown in black with dots showing position estimates along the flight path, while the A* path is seen as a series of straight, purple lines. The fine occupancy grid, is presented by simply drawing each occupied cell, (occupancy level color coding not clearly seen due to resolution). The A* path is seen updated as the map evolves, and the helicopter is flown towards the goal.

The final frame of Figure 5.47 is enlarged, and presented again in Figure 5.48. It shows the helicopter reaching the goal, and the resulting final map that was generated. The experiment was successfully repeated several times, in the same environment but with different goal positions, under the same conditions discussed above. The pilot reported adequate workload involved in both flying the helicopter and correcting its heading based on the instructions from the algorithm.

### 5.6.2 Path Planning With Artificial Obstacle Avoidance

The second targeted flight experiment was performed indoors, in the Kim Building, at the University of Maryland. A representative picture of the scenario is shown in Figure 4.8. The scenario provided sufficient open space to fly the helicopter, and features multiple objects such as chairs, tables, and a large set of spiral stairs (all area challenging non-2D objects). In addition, the area is surrounded by glass windows, which pose additional difficulty to map with a laser scanner (due to poor reflectivity of the lasers energy).

As in the outdoors experiment, the pilot was not given any information as for the goal location, and is only provided with a heading arrow. The objective in this scenario was to eliminate the pilot's judgment with respect to the path planning (as well as local obstacle avoidance), by adding objects into the A* occupancy grid. The artificial objects are represented by artificially occupied cells. These artificial obstacles are not visible to the pilot at any point prior or during the experiment. Naturally, the SLAM algorithm only considers the real obstacles, which are scanned by the laser.

The use of artificial obstacles allows us to decouple the pilot's judgement from the obstacle avoidance. Since the pilot does not see the obstacles in any way - we can conclude that he flies the helicopter based only on the instructions from the algorithm. The goal position in this scenario is placed 15 $m$ meters ahead of the initial position. However, due to the presence of the artificial obstacles, the navigation task is quite involved, mainly due to the proximity of the obstacles to each other.

In the present scenario, the algorithm itself is given the initial and goal positions, and several cells in the coarse A* occupancy grid are artificially occupied to represent the artificial obstacles, considered by the A* path planner. Figure 5.49 present several snapshots of the fine occupancy grid, and the A* occupancy grid, showing the evolution of these two occupancy grid maps at key points along the flight trajectory (identical information and symbols as in Figure 5.47). The artificial obstacles are presented by drawing the artificially occupied A* cell in red. As in the outdoor scenario, the A* path is seen updated as the map evolves, and the helicopter is flown towards the goal.

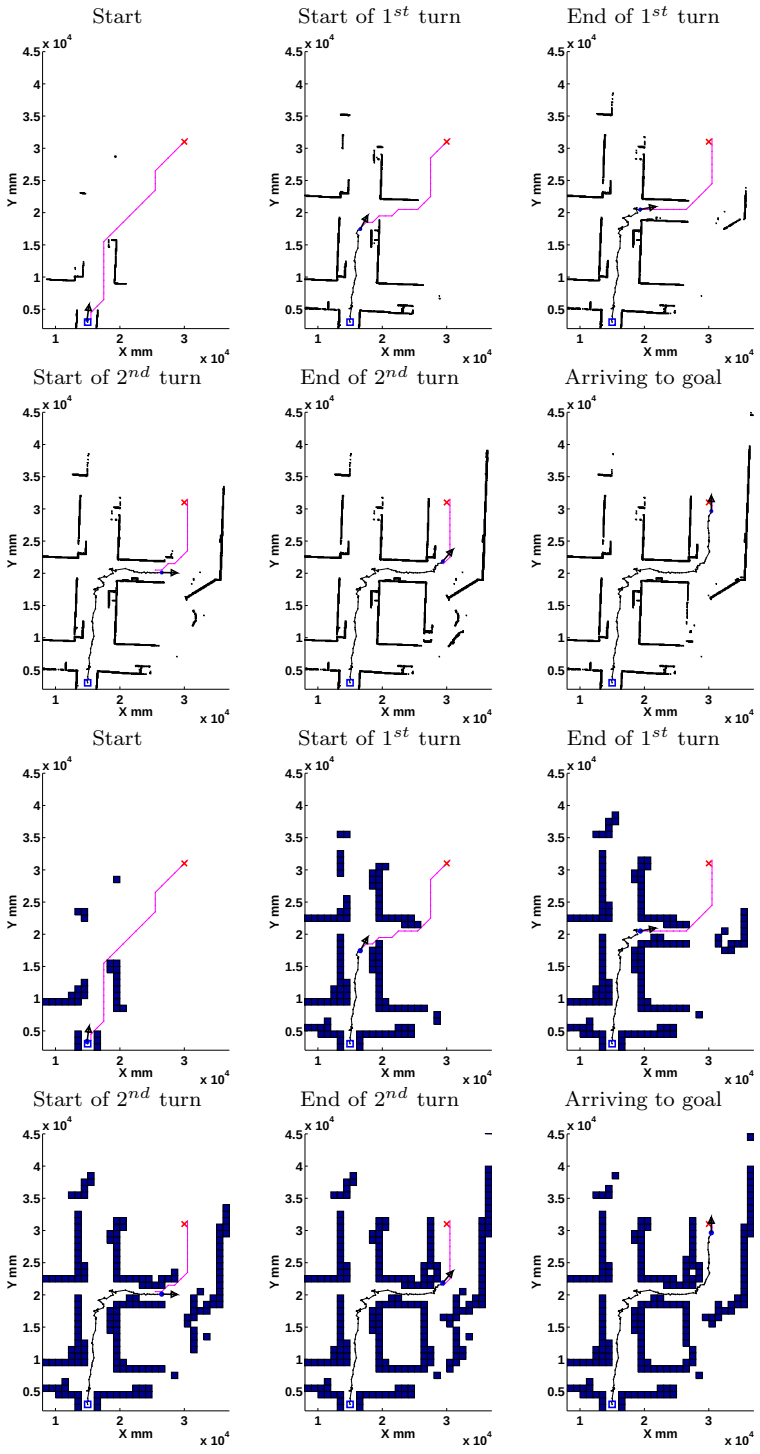



Figure 5.47: Outdoor experiment, evolution of the occupancy grid and the A* occupancy grid, with the helicopter planned and executed path. Top and bottom 6 figures show the fine and coarse occupancy grid evolution, respectively. Helicopter and A* path are shown as black and purple lines, respectively.

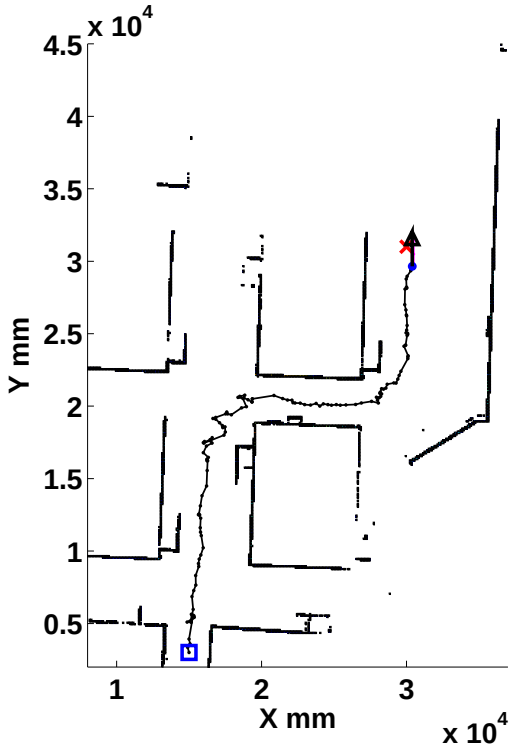



Figure 5.48: Outdoor experiment, final step - arrival to goal.

Figure 5.50 presents three insets of the final occupancy grid map, focusing at the three artificial obstacles. The insets all show the path of the helicopter, avoiding all three artificial obstacles. Note that in this case, the helicopter's size was intentionally not considered, thus allowing close proximity to the artificial obstacles, in order to test the limits of the algorithm. Generally, a minimum safe distance is maintained, based on the MAV size (by employing close proximity rules in the A* algorithm).

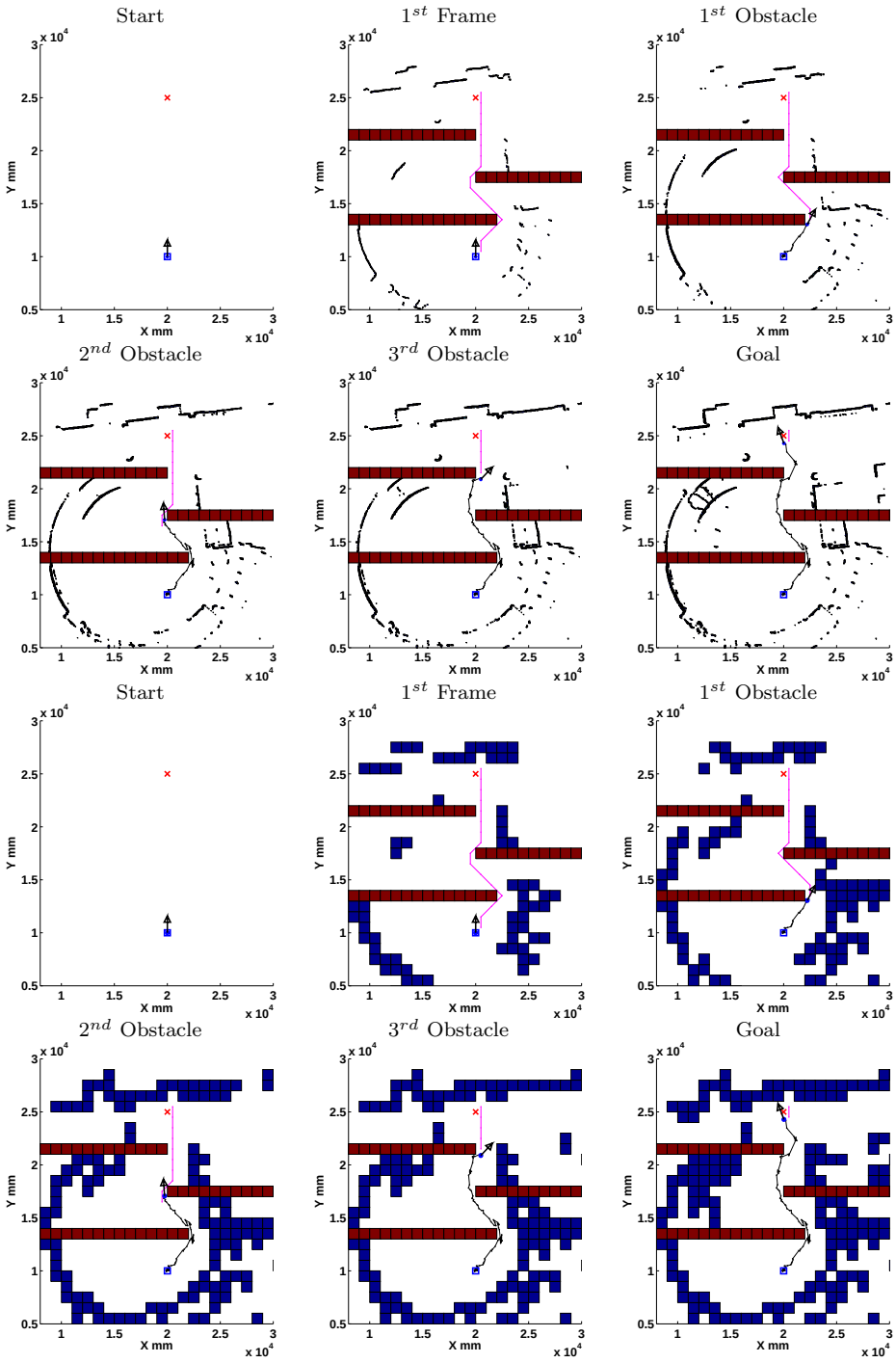



Figure 5.49: Indoor experiment, evolution of the occupancy grid and the A* occupancy grid, with the helicopter planned and executed path. Top and bottom 6 figures show the fine and coarse occupancy grid evolution, respectively. Helicopter and A* path are shown as black and purple lines, respectively. Artificial obstacles are shown as red squares.

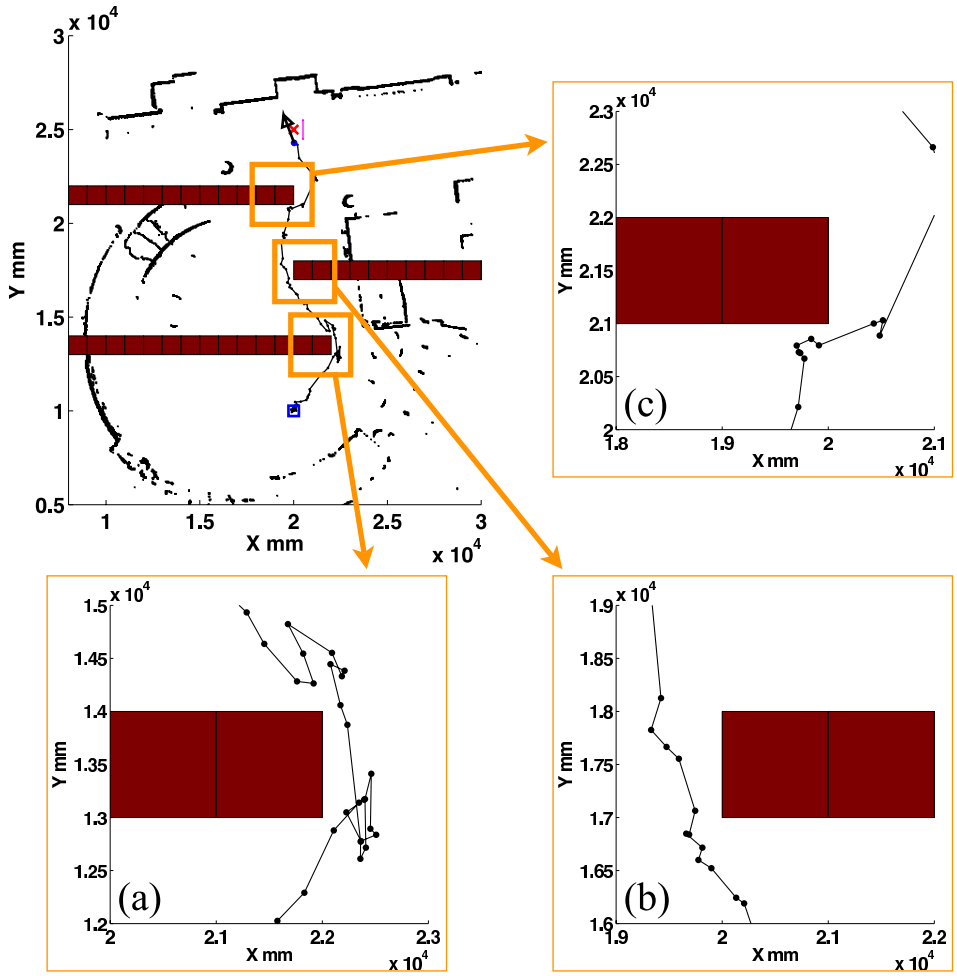



Figure 5.50: Zoom in for the final map and flight path, showing the helicopter's accurate position estimates, avoiding the artificial obstacles (without consideration for the helicopter's size).

# Chapter 6

# Conclusion

## 6.1 Summary

### SLAM Algorithm

This thesis presented a refined SLAM algorithm, based on a scan matching approach. The algorithm is independent of the platform's dynamic model, and is designed to work in primarily two dimensional environments.

The SLAM algorithm was tested on 9 environments, both indoor and outdoor. Three different platforms were used, wheeled ground platform, a walking person, and an aerial platform. Successful results were achieved on all three platforms. Repeatability and robustness were shown using a set of 12 repeatable experiments on a benchmark scenario. The algorithm was also successfully tested using previously published datasets, which were collected using different laser scanners.

### Scan Matching Algorithm

A refined scan matching algorithm, Perimeter Based Polar Scan Matching (PB-PSM), was presented, developed for scan matching 2D laser scans. The scan matching algorithm exploits the polar coordinate representation of a set of laser range measurement, as they are output from a 2D laser range scanner. PB-PSM differs from previously published algorithms in three main aspects: using adaptive direct search for the cost minimization process, introduction of a perimeter matching term in the scan matching cost function (maximizing overlap between the matched scans), and employing relatively tight convergence requirements.

The algorithm is shown to have superior performance as compared with previously published algorithms, with relatively high accuracy. The algorithm was tested using various laser sensors, and several different environments, both indoor and outdoor. PB-PSM was proven to be very useful for both laser odometry and SLAM applications.

### Accuracy

A literature review of previously published work about SLAM algorithm revealed a large gap in the discussion about accuracy, as well as a lack of tools that enable quantitative comparison between SLAM algorithms. The importance of accuracy achieved by SLAM algorithms was therefore thoroughly discussed in this thesis, and uniquely quantified in large scale scenarios. Several metrics were used for accuracy quantification, in a benchmark scenario, including comparison of several measured distances, and a metric for matching a complete occupancy grid. The newly formulated metric provides an average occupied cell distance from the true map.

Typically, SLAM results would be compared to an overhead layout of the scenario by means of the naked eye, in a qualitative manner. Moreover, the overhead layout used for comparison in most cases is not the measured true map, but rather an approximate map (blueprints, satellite images, pictures, or GPS data which contains considerable error in some cases). Quantitative estimates of accumulated drift were rarely provided, and only when using simulation environments or small scale scenarios, where a motion capture system could be set up. Moreover robustness was not shown, as most research works focus on very few scenarios (typically one), and repeatability is often ignored.

The current algorithm's performance was analyzed in great detail, showing its robustness and repeatability over 12 repeated experiments in a benchmark office-like scenario. The contribution of the various algorithmic elements was assessed using a newly developed metric, which allows quantitative comparison of SLAM-generated maps to a true map of the experimental scenario. For this reason, the RMS of the accuracy metrics employed on all 12 experiments was used as a baseline for comparison.

The proposed algorithm was favorably benchmarked against previously published Iterative Closest Point (ICP) scan matching algorithms, showing a significant accuracy advantage. It was shown that the algorithm's innovations are in fact key to the final map accuracy (independent of the data association technique). Additional qualitative evaluation are presented using visual examinations, as was done in previous published research.

### Targeted Flight

The SLAM algorithm was then coupled to an A* path planner, and experimental validation for the entire navigation system was presented, using pilot in the loop experiments. The pilot was capable of successfully navigating between obstacles, following heading information only, towards a pre-defined goal position. To eliminate the pilot's judgement from the obstacle avoidance task, invisible obstacles that the pilot cannot see were introduced. These artificial obstacles were considered only by the path planning algorithm. The algorithm successfully instructed the pilot to navigate the helicopter safely to the goal, while avoiding all obstacles.

## 6.2 Conclusions

### 6.2.1 Major Conclusions

i. General conclusions

(a) It was shown in this thesis that some operational scenarios may not provide any opportunity for revisiting previously explored areas. Therefore, in those scenarios, successful operation hinges upon the SLAM algorithm's accuracy. Algorithms that depend on loop closure opportunities may not be utilized. Specifically, the importance of SLAM accuracy for targeted flight missions was presented and discussed.

(b) The proposed SLAM algorithm may be used for successful targeted flight missions, where the goal is defined relative to the initial position. Using the structure of the proposed SLAM algorithm, the path planner is completely decoupled from the SLAM algorithm, making it easy to use with many different path planners (Section 5.6).

(c) The drift obtained by the suggested SLAM algorithm was quantified at approximately 0.1% of the traveled distance length (Sub-Section 5.2.1). This is a considerably lower drift as compared with previously published algorithms.

(d) The proposed SLAM algorithm may be employed on various platforms including both ground (wheeled and walking) and aerial. The accuracy achieved with all platforms was shown to be fairly comparable.

(e) Using the newly developed metric, presented in this thesis, accuracy of SLAM algorithms may be quantitatively compared, when employed on the same benchmark scenario. The metric particularly fits occupancy grid based maps, which are currently the most common form of maps. This metric may also be used for quantitatively evaluating loop closure algorithms, by employing the metric on the optimized maps. Quantitative accuracy estimates may greatly complement the qualitative comparison that is typically presented in the literature.

(f) Single sensor SLAM was demonstrated throughout this thesis, when using the proposed algorithm. Although an Inertial Measurement Unit (IMU) may significantly reduce the computational requirements, it is not an essential sensor, and SLAM results with a laser scanner as the sole sensor have been demonstrated to be highly accurate. The IMU data can be used to provide a good initial guess for the scan matching, reducing the number of iteration required for convergence.

ii. Adaptive direct search

(a) For scan matching algorithms, the use of adaptive direct search has been shown to be superior in terms of overall scan matching accuracy. Adaptive

direct search had a very significant impact on achieving the most accurate solution for scan matching, with improved robustness to the common phenomenon of local minima. Gradient search based methods for minimizing the scan matching cost function may not be a good candidate for achieving the best global minima (Figure 5.14). It was also shown that the adaptive direct search is, in fact, responsible for the achieved accuracy, regardless of the method of data association between the two matched scans (Figure 5.15).

(b) Using adaptive direct search, the tight convergence criterion requirements of $0.01^\circ$ for rotation and 1 $mm$ for translation, were shown to have a strong effect on the overall performance (Figure 5.12).

(c) The adaptive direct search requires considerably more cost function evaluations, and therefore a cost function with a complexity of $O(n)$ (where $n$ is the number of points in the reference scan), is required. The polar scan matching approach allows for such a cost function, while other scan matching algorithms have a cost function with a higher complexity of $O(kn)$ or $O(n^2)$ (in this case $k$ is the size of a search window used in the cost function).

iii. The perimeter matching term was shown to be a key in improving the robustness of the scan matching algorithm. It contributed greatly to overall accuracy since small objects that were not over-shadowed by large objects. The maximum overlap between the matched scans was driving to matching process (Figures 5.10-5.11, and Figure 5.13).

iv. Using virtual scans rather than the more common laser odometry was shown to have a significant positive effect on overall SLAM accuracy (Figures 5.10 and 5.17).

v. When attempting to match single laser scenes, it was shown that PB-PSM outperforms several previously published scan matching methods, especially with respect to rotation estimation (Figure 5.3).

vi. For laser odometry on complete datasets, accuracy achieved by PB-PSM was proven to be superior as compared with the PSM, PSM-C, and ICP algorithms employed on complete datasets (Figure 5.16).

### 6.2.2 Additional Conclusions

i. The convergence of PB-PSM was shown to be quite fast, as most scenarios converged within approximately 10 iterations (Figure 5.2). This may be improved with the introduction of a better initial guess.

ii. It was shown that accuracy can be improved when using datasets with more scans at a higher rate, as this reduces the error introduced into the laser scan by platform's motion (Figure 5.4). The accuracy using such datasets was shown to

be very high, with the true map of the scenario overlapping the generated map with no more than $3-4$ $cm$ of error, for a 50 $m$ traveled path (Figures 5.5-5.7).

iii. Use of previously mapped areas was demonstrated to be beneficial for map accuracy. When returning to an area that was already mapped, the algorithm was able to use the existing map to correct itself, and lower the cost function (Figures 5.8-5.9, and Figure 5.24).

iv. The proposed SLAM algorithm may be employed over long traveled distances, with very minimal error, while maintaining the relatively small drift (Figures 5.18-5.20). The relatively high accuracy may be used to provide better initial guesse for loop closure algorithms for further map optimizations.

v. The proposed algorithm limitations stem mainly from the capabilities of the laser scanner (range, frequency, noise, spatial, and angular resolution, see Figures 3.20-5.26, and Figures 5.29-5.30), the occupancy grid resolution (Figures 5.31-5.33), and the virtual scan resolution (Figure 5.28).

vi. The failure modes of the algorithm include repetitive structure, where the scan matching solution may be ambiguous (Figure 5.34), dynamic environments that introduce objects that may not be used for accurate mapping (Figure 3.16), areas with insufficient features for scan matching (Figures 5.35-5.37), areas that feature many non-2D objects (Figure 5.37), as well as high pitch and roll situations where the scanned objects may contain parts of the floor or ceiling.

vii. Mapping a closed loop course without the aid of a loop closure algorithm was shown using a wheeled ground platform (throughout Sub-Section 5.2.1), a walking person (Figure 5.18), and a helicopter platform (Figure 5.25). All results presented a very accurate and seamless loop closure area.

viii. The proposed SLAM algorithm was extensively tested on multiple datasets, including previously published datasets by other authors, using significantly different laser scanners, showing improved performance Figure 5.38 or comparable performance (Figure 5.39).

ix. Extensive testing was performed in multiple outdoor scenarios, featuring both typical urban buildings, as well as forest environments, both considered to be highly unstructured, and feature many non-2D objects (Section 5.5).

x. Mapping results were shown to be independent of the platform's speed, within the limitations presented in Sub-Section 5.3.5 (shown in Sub-Section 5.2.1, as well as in Figure 5.44). Moreover, the path traveled by the platform does not affect the mapping result (Figure 5.45).

## 6.3 Future Work

The following is a list of possible future work areas:

- To be fully autonomous, the MAV is also required to calculate the control commands that will lead to following the desired trajectory, taking into account its maneuverability limitations. For this task, a command calculation block needs to complement the algorithms presented in this work.

- The scan matching method can become considerably faster if a good initial guess can be provided. This will allow the function minimization to be performed on search grids that are smaller than the typical platform step size. This will, in turn, reduce the number of iterations and subsequently cost function calls. A good initial guess may be achieved with an accelerometer/gyro data integration, that will give an initial pose estimated.

- The limitation on pitch/roll angles may be addressed with the introduction of gyro data. The pitch/roll angle can be measured by the gyro, and combined with height measurements, which may be achieved using several downward-deflected beams from the laser scanner, one maybe able to figure out if the laser scan contains floor scans and should be excluded from the scan matching process. This practice has been used in the work by Shen et al. [3].

- The current algorithm was developed in 2D. However, the concepts presented here may easily be applied in 3D, when a point cloud is the incoming dataset. The adaptive direct search grid may be constructed in 3D, and the perimeter matching terms will turn into an "area" matching term, as it will now consider a matched area as the basis for cost rewarding.

# Bibliography


[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part I the essential algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. 1. The MIT Press, 2005.

[3] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. pages 20–25, 2011. IEEE International Conference on Robotics and Automation (ICRA), 9-13 May.

[4] V. Nguyen, A. Harati, and R. Siegwart. A lightweight slam algorithm using orthogonal planes for indoor mobile robotics. 2007. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, 29 October - 2 November, San Diego, California, USA.

[5] O Garcia-Favrot and M Parent. Laser scanner based slam in real road and traffic environment. 2009. In IEEE International Conference Robotics and Automation (ICRA09). Workshop on Safe navigation in open and dynamic environments Application to autonomous vehicles.

[6] A Bachrach, S Prentice, R He, and N Roy. RANGE–Robust autonomous navigation in GPS?denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.

[7] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate SLAM with Rao–Blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.

[8] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. 2003. IEEE/RSJ International Conference on Intelligent Robots and Systems, 27-31 October, Las Vegas, NV, USA.

[9] Cyrill Stachniss, Giorgio Grisetti, Dirk Hähnel, and Wolfram Burgard. Improved rao-blackwellized mapping by adaptive sampling and active loop-closure. 2004. In Proc. of the Workshop on Self-Organization of AdaptiVE behavior (SOAVE).

[10] A. Segal, D. Hähnel, and S. Thrun. Generalized-icp. volume 25, 2009. Proc. of Robotics: Science and Systems (RSS).

[11] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[12] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. pages 3562–3568, 2006. IEEE/RSJ International Conference on Intelligent Robots and Systems.

[13] Ye Cang and Borenstein Johann. A novel filter for terrain mapping with laser rangefinders. *IEEE Transactions on Robotics*, 20(5):913–921, 2004.

[14] T. Bailey, J. Nieto, and E. Nebot. Consistency of the fastslam algorithm. pages 424–427, 2006. IEEE International Conference on Intelligent Robotics and Automation.

[15] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. Autonomous navigation and exploration of a quadrotor helicopter in gps-denied indoor environments. 2009. In IARC First Symposium on Indoor Flight Issues.

[16] K. Lee, S. Wijesoma, and J. Guzmán. A constrained slam approach to robust and accurate localisation of autonomous ground vehicles. *Robotics and Autonomous Systems*, 55(7):527–540, 2007.

[17] Wayne Johnson. *Helicopter Theory*. Courier Dover Publications, 2012.

[18] S Thrun, M Diel, and D Hähnel. Scan alignment and 3-D surface modeling with a helicopter platform. *The $4^{th}$ International Conference on Field and Service Robots*, 2003.

[19] B Steder, G Grisetti, C Stachniss, and W Robotics IEEE Transactions on Burgard. Visual SLAM for Flying Vehicles. *Robotics, IEEE Transactions on*, 24(5), 2008.

[20] LA Clemente, AJ Davison, I Reid, J Neira, and JD Tardós. Mapping large loops with a single hand-held camera. *Robotics: Science and Systems*, 2007.

[21] Jari Saarinen, Roman Mazl, Miroslav Kulich, Jussi Suomela, Libor Preucil, and Aarne Halme. Methods for personal localisation and mapping. 2004. $5^{th}$ IFAC/EURON Symposium on Intelligent Autonomous Vehicles Instituto Superior Tcnico, July 5-7, Lisboa, Portugal.

[22] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6d slam with an application in autonomous mine mapping. *Robotics and Automation*, 2:1998–2003, 2004.

[23] Lionel Heng, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing. pages 2472–2477, 2011. IEEE International Conference on Robotics and Automation (ICRA).

[24] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-Based State Estimation for Autonomous Rotorcraft MAVs in Complex Environments. In *IEEE International Conference on Robotics and Automation (ICRA).*, 2013.

[25] Chen Friedman, Inderjit Chopra, Svetlana Potyagaylo, Omri Rand, and Yaron Kanza. Towards model-free slam using a single laser range scanner for helicopter mav. 2011. AIAA Guidance Navigation and Controls Conference, Portland, Oregon, August 8-11.

[26] T. Bailey and E. Nebot. Localisation in large-scale environments. *Robotics and Autonomous Systems*, 37:261–281, 2001.

[27] C. Brenneke and B. Wagner. A scan based navigation system for autonomous operation of mobile robots in man-made environments. 2003. International Conference of systems engineering (ICSE), Coventry, Great Britain, September.

[28] Soonyong Park and Sung-Kee Park. Spectral scan matching and its application to global localization for mobile robots. 2010. IEEE International Conference on Robotics and Automation Anchorage Convention District May 3-8, 2010, Anchorage, Alaska, USA.

[29] Albert Diosi and Lindsay Kleeman. Fast laser scan matching using polar coordinates. *Int J Robot Res*, 26(10):1125–1153, 2007.

[30] Kin Leong Ho and Paul Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286, 2007.

[31] A. Bachrach, A. S. Huang, P. Henry D. Maturana, M. Krainin, D. Fox, and N. Roy. Visual navigation for micro air vehicles. *(workshop manuscript)*, 2011.

[32] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[33] P. Nunez, R. Vazquez-Martin, A. Bandera, and F. Sandoval. Fast laser scan matching approach based on adaptive curvature estimation for mobile robots. *Robotica*, 27:469–479, 2009.

[34] G.A. Borges and M.J. Aldon. Line extraction in 2d range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40(3):267–297, 2004.

[35] J. S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. 1999. IEEE International Symposium on Computational Intelligence in Robotics and Automation 1999 CIRA '99 Proceedings 1999.

[36] Kurt Konolige. Large-scale map-making. 2004. Proceedings Of The National Conference On Artificial Intelligence.

[37] J Nieto, T Bailey, and E Nebot. Scan-SLAM: Combining EKF-SLAM and scan correlation. In *Field and Service Robotics (FSR)*, 2005. 167–178.

[38] R Sim and N Roy. Global a-optimal robot exploration in slam. *IEEE International Conference on Robotics and Automation*, 1:661, 2005.

[39] J Nieto, T Bailey, and E Nebot. Recursive scan-matching SLAM. *Robotics and Autonomous Systems*, 55(1):39–49, 2007.

[40] P Newman, D Cole, and K Ho. Outdoor SLAM using visual appearance and laser ranging. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187, 2006.

[41] Sebastian Thrun, Mark Diel, and Dirk Hähnel. Scan alignment and 3-d surface modeling with a helicopter platform. In *Field and Service Robotics*, pages 287–297. Springer, 2006.

[42] Ruijie He, S Prentice, and N Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *IEEE International Conference on Robotics and Automation (ICRA).*, 2008.

[43] T Kollar and N Roy. Efficient optimization of information-theoretic exploration in SLAM. *AAAI*, pages 1369–1375, 2008.

[44] Jorge Artieda, José M Sebastian, Pascual Campoy, Juan F Correa, Iván F Mondragón, Carol Martínez, and Miguel Olivares. Visual 3-D SLAM from UAVs. *Journal of Intelligent and Robotic Systems*, 55(4-5):299–321, 2009.

[45] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[46] F Kendoul, I Fantoni, and K Nonami. Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles. *Robotics and Autonomous Systems*, 57(6-7):591–602, 2009.

[47] S Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001.

[48] E Brunskill and N Roy. Slam using incremental probabilistic pca and dimensionality reduction. *IEEE International Conference on Robotics and Automation*, 1:342, 2005.

[49] X Ji, H Zhang, D Hai, and Z Zheng. An Incremental SLAM Algorithm with Inter-calibration between State Estimation and Data. *RoboCup 2008: Robot Soccer World Cup XII*, 2009.

[50] R Valencia, J Valls Miró, and G Dissanayake. Active pose SLAM. *Robots and Systems*, 2012.

[51] A Walcott-Bryant, M Kaess, H Johannsson, and Leonard J. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2012.

[52] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 2:1899–1906, 2003.

[53] E Olson, J Leonard, and S Teller. Fast iterative alignment of pose graphs with poor initial estimates. 2006. International Conference on Robotics and Automation.

[54] J.S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. pages 61–67, 1996. Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots.

[55] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[56] T. Röfer. Using histogram correlation to create consistent laser scan maps. pages 625–630, 2002. Proceedings of the IEEE International Conference on Robotics Systems (IROS-2002). EPFL, Luasanne, Switzerland.

[57] X Yuan, CX Zhao, and ZM Tang. Lidar scan-matching for mobile robot localization. *Information Technology Journal*, 9(1):27–33, 2010.

[58] A. Censi. An ICP variant using a point-to-line metric. pages 19–25, 2008. IEEE International Conference on Robotics and Automation (ICRA), 2008.

[59] M. Tomono. A scan matching method using euclidean invariant signature for global localization and map building. volume 1, pages 886–871, 2004. ICRA 2004, Piscataway, NJ.

[60] I. J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[61] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[62] Banerji Debajyoti, Ray Ranjit, Basu Jhankar, and Basak Indrajit. Autonomous Navigation by Robust Scan Matching Technique. *INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND CREATIVE ENGINEERING*, 2(10), 2012.

[63] Koenig Sven and Maxim Likhachev. D* lite. pages 476–483, 2002. AAAI/IAAI.

[64] L. Kavraki, P. Svestka, J. Latombe, , and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[65] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998. Technical Report 98-11Computer Science Dept., Iowa State University.

[66] Svetlana Potyagaylo and Omri Rand. Planning and operational algorithms for autonomous helicopter. 2009. American Helicopter Society 65th Annual Forum, Grapevine, Texas, May 27-29.

[67] M. Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT, 1996.

[68] Hart Peter E., Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. pages 100–107, 1968. IEEE Transactions on Systems Science and Cybernetics.

[69] N Roy, G Gordon, and S Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23(1):1–40, 2005.

[70] T Kollar and N Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27(2):175, 2008.

[71] E. B. Olson. Real-time correlative scan matching. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009.

[72] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[73] Bernt Schiele and James L. Crowley. A comparison of position estimation techniques using occupancy grids. *Robotics and autonomous systems*, 12(3-4):163–161, 1994.

[74] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

[75] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

[76] Hokuyo Automatic Co. LTD. Hokuyo UTM-30LX laser range finder, 2005. http://www.hokuyo-aut.jp/02sensor/07scanner/download/index.html.

[77] Y Okubo, C Ye, and J Borenstein. Characterization of the hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation. 2010. SPIE Defense, Security + Sensing, Unmanned Systems Technology XI.

[78] A. Diosi. *Laser Range Finder and Advanced Sonar Based Simultaneous Localization and Mapping for Mobile Robots*. PhD thesis, Monash University, Australia, 2005.

[79] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. 2010. Pearson Education, $3^{rd}$.

[80] Amit Patel. *Amit?s thoughts on path-finding and A-star*. http://theory.stanford.edu/amitp/GameProgramming/, 2003.

[81] Daniel Rolf Wichmann. Automated Route Finding on Digital Terrains. Technical Report COMPSCI 780 Project Report, Graphics Group, Department of Computer Science, University of Auckland, New Zealand, February 2004.

[82] Hokuyo Automatic Co. LTD. Hokuyo laser range finder, 2005. http://www.hokuyo-aut.jp/02sensor/.

[83] Hokuyo Automatic Co. LTD. Hokuyo URG-04LX-UG01 laser range finder, 2005. http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/URG-04LX_UG01_spec.pdf.

[84] P. Bergstrm. ICP implementation for Matlab. 2007. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=12627&objectType=FILE.

[85] A. Howard. Laser scans dataset collected at fort ap hill, as part of the darpa/ipto sdr project. Technical report, 2004. http://cres.usc.edu/radishrepository/view-one.php?name=ap_hill_07b.

[86] A. Howard, L.E. Parker, and G.S. Sukhatme. Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection. *International Journal of Robotics Research*, 25(5-6):431–447, 2006.