TOWARDS 3D MAPPING AND AUTONOMOUS EXPLORATION FOR UNDERWATER ROBOTS IN CLUTTERED ENVIRONMENTS

by

Jinkun Wang

A DISSERTATION

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Jinkun Wang, Candidate

ADVISORY COMMITTEE

Brendan Englot, Chairman Date

Enrique Dunn Date

Yi Guo Date

Mehmet Kurt Date

Damiano Zanotto Date

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2019

TOWARDS 3D MAPPING AND AUTONOMOUS EXPLORATION FOR UNDERWATER ROBOTS IN CLUTTERED ENVIRONMENTS

ABSTRACT

Over the last two decades, the application of autonomous underwater vehicles has proliferated across challenging perceptual tasks such as pipeline and ship hull inspection, bathymetric survey and structure mapping. Although a camera can capture fine details of the underwater environment, its capability is often limited by the turbidity of the water. However, a viable alternative, sonar, is plagued by low signal-to-noise ratio, low resolution and the lack of elevation information in a sonar measurement. Therefore, accurate, reliable state estimation and mapping, and specifically, simultaneous localization and mapping (SLAM), are essential prerequisites for achieving such autonomy. Meanwhile, the capability of autonomous exploration without human intervention is potentially impactful for underwater robots in scenarios where teleoperation is limited or infeasible due to constrained communication in cluttered, previously unmapped subsea environments.

In this work, we accomplish the objective of autonomous exploration by solving three major problems faced by underwater robots navigating in cluttered marine environments: mapping, SLAM and exploration. First, building representative 3D maps of surrounding environments is fundamental for applications such as safe navigation and environmental inspection. We propose an efficient Gaussian process occupancy mapping algorithm that is capable of producing rich yet accurate maps utilizing sparse measurements from a mechanical scanning sonar. We further incorporate a Gaussian process random field into a factor graph, yielding a 3D terrain map from imaging sonar in the absence of elevation angles. Secondly, a robust data association

technique based on joint compatibility branch and bound is proposed to address the ambiguity during landmark matching in feature-sparse subsea environments. Work on the development of SLAM for our inspection-class underwater robot is described. Thirdly, we present a novel expectation-maximization (EM) exploration algorithm, taking into account the robot's mapping rate, map uncertainty, and state estimation uncertainty. We validate our proposed algorithm on a proxy ground vehicle equipped with a short-range LiDAR, leveraging segment-aided pose SLAM adapted for better active localization. Lastly, the real-time applicability of the framework is also demonstrated in a previously unmapped underwater environment, which is supported by a robust SLAM framework using 2D imaging sonar.

Author: Jinkun Wang

Advisor: Brendan Englot

Date: November 25, 2019

Department: Mechanical Engineering

Degree: Doctor of Philosophy

Acknowledgments

Two roads diverged in front of me as a third-year undergraduate student six years ago and I took the one that led me through five years of Ph.D. study. Although the journey was challenging, looking back at my decision now I feel happy and fulfilled. During the time I spent in the Robust Field Autonomy Lab (RFAL) at Stevens, numerous people have helped me. I cannot begin to express my thanks to my advisor, Dr. Brendan Englot. Dr. Englot is a great mentor in both research and life. Under his guidance, I felt confident in robotics research and was able to publish papers in top-tier conferences from an undergraduate who knew nothing about robotics. As a foreign student, I always felt relaxed and comfortable in the lab thanks to Dr. Englot's great care of us.

I would like to express my deepest appreciation to my committee members: Dr. Enrique Dunn, Dr. Yi Guo, Dr. Mehmet Kurt, and Dr. Damiano Zanotto. I'm extremely grateful for your suggestions and comments on my dissertation, and I enjoyed the proposal and defense with you.

I would like to extend my sincere thanks to all RFAL members: Shi Bai, Dong Cui, Fanfei Chen, Kevin Doherty, Dengwei Gao, Yewei Huang, John Martin, Jake McConnell, Sumukh Patil, Erik Pearson, Paul Szenher and Tixiao Shan. I have learned a lot from the discussions with you and my research work couldn't be done without your feedback. I also wish to thank Prof. Carolyn Hunter at U.S. Merchant Marine Academy and all Summer Research Institute (SRI) Program participants, who helped me a lot in field experiments.

I gratefully acknowledge the sponsorship that supported my doctoral study: American Bureau of Shipping Scholarship, NSF and Schlumberger-Doll Research. Special thanks to Stephane Vannuffelen and Timothy Osedach from Schlumberger,

and I had the great pleasure of working with you on the BlueROV project!

I want to extend my thanks to Wei Cui, Houzhu Ding, Zhongjing Ren, Xiaoyu Su, Xiangyu Xu, Chenhui Zhao, and Jiaxing Zhou. It was a wonderful time to enjoy American life with you. Finally, the completion of my dissertation would not be possible without the support of my parents. I feel regret for not visiting you frequently during these years, and I will give my best in the future.

Jinkun Wang (王瑾琨)

December 1, 2019

Broomfield, CO

Table of Contents

List of Tables

List of Figures

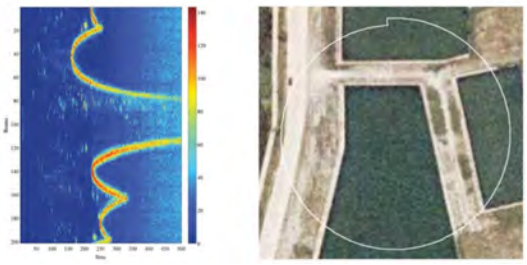
Chapter 1
Introduction

In recent years, robots, such as manipulators, legged robots and quadrotors, are under active development to bring autonomy into industry or even our daily life. Their development has been advanced dramatically with the help of computer vision, machine learning and reinforcement learning. In December 2018, Waymo was announced as the first company to commercialize a fully autonomous taxi service in the U.S. However, underwater robots haven't achieved the same levels of autonomy as ground vehicles. More than 80% of the oceans, which cover 71% of the area of the Earth, haven't been explored and mapped as throughly as Earth's surfaces. Therefore, oceans remain as a frontier for robotics research. Underwater robots that are equipped with similar autonomy as ground vehicles will lead to more discovery of valuable resources and energy in the depths of the ocean. Also, constructing accurate, descriptive maps of underwater structures and terrain is of great interest for security and environmental monitoring, and for port and harbor infrastructure inspection, where regular observation by divers is currently required.

The challenges we are faced with in underwater robots are rooted in the limited sensing capability as well as high and unpredictable disturbances. Electromagnetic signals are heavily damped underwater, thus not propagating far. Thus, it rules out the applicability of a variety of sensors relying on electromagnetic waves including camera, GPS, laser range-finder, etc. Acoustic waves are the canonical alternative in positioning and perception systems. For instance, with a Doppler Velocity Log (DVL), we are able to measure the relative velocity with respect to the sea bottom by measuring the Doppler shift of the sound wave; with an imaging sonar, we know

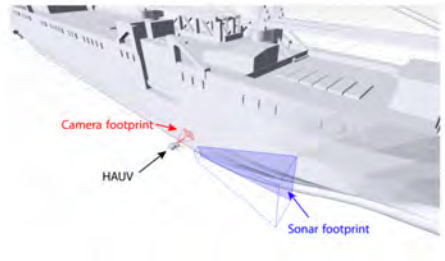
whether obstacles exist in the sensor's field of view by measuring the intensity of the reflected sound wave. However, the resolution of acoustic sensors is severely limited by the wavelength of sound, which is typically larger by 2 to 3 orders of magnitude than that of light. What's more, underwater sound propagation is affected by a number of factors: loss from the spherical spread of sound waves, attenuation due to absorption, refraction and scattering. Apart from the noise and low resolution of sensors, disturbances from the water also pose a great challenge for accurate state estimation and stable attitude control, especially for applications in environments with high currents. The development of underwater robotics is also restricted by the computation resource an embedded computer can provide. Though we can transfer computation to a top-side computer, the bandwidth of transmission is limited.

Many underwater robot platforms have performed mapping tasks, or achieved full autonomy in complex outdoor environments. For an autonomous underwater vehicle (AUV) equipped with a DVL and a single-beam scanning sonar, a navigation system has been developed to produce 2D maps of man-made, structured environments [7], [8], [3]. Inspection of ship hulls and marine structures was studied and approached by simultaneous localization and mapping, navigation and planning algorithms [4], [9], [10]. An approach to endow an AUV with the capabilities to move through unexplored environments was presented with specific focus on planning of feasible and safe paths in underwater environments, [11], [12], [6], [13], [5]. The real experiments mentioned above are depicted in Fig. 1.1. The novelty of this work is increasing underwater mapping capability (from sparse to dense maps and from 2D to 3D maps) and taking into account uncertainty in exploring an unknown environment.

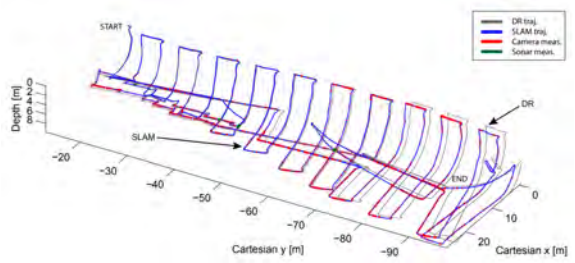


(a) Raw sonar data of an abandoned marina in polar frames [3]

(b) The resulting map and trajectory from SLAM together with satellite image [3]

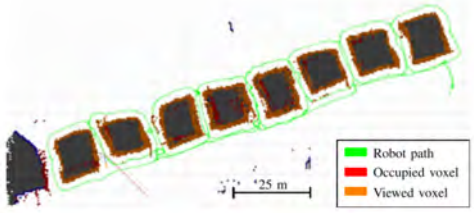


(c) Depiction of the sensor field of view for the imaging sonar and monocular camera during open-area, hull-locked inspection [4]



(d) SLAM navigation results on the SS Curtiss [4]

(e) Survey areas consisting of a series of concrete blocks [5]



(f) Real survey trajectory using cameras on AUVs [6]

Figure 1.1: Examples of underwater mapping and navigation experiments.

1.1 Problem Statement

Our objective is to achieve full autonomy on underwater vehicles, which are equipped with perception and navigation sensors, in complex and unknown subsea environments. To successfully reach the objective, simultaneous localization and mapping (SLAM) is a key component that must be adapted for noisy, featureless underwater environments. As a result of sparse measurements from acoustic sensors, descriptive and rich maps are of great value for both inspection and navigation. Finally, the robot should be able to explore unknown environments without human intervention, while estimating its localization and inferring the surroundings using SLAM. More importantly, the exploration should take into account uncertainty associated with localization and mapping, because underwater vehicles are prone to drift.

1.2 Overview and Contributions

The dissertation is structured as follows. An autonomous system integrates state estimation, mapping and navigation. In Chapter 2, we present an introduction of essential components in autonomous robots: SLAM, occupancy mapping and exploration algorithms. We provide a review of current methodologies.

In Chapter 3, we discuss the mapping problem using sonar in underwater environments. Specifically, we tackle two problems in the sonar measurements, sparsity which we encounter using a single-beam sonar and loss of elevation angles which we encounter using a 2D imaging sonar. In Chapter 3.3, we present a novel algorithm to produce descriptive online 3D occupancy maps using Gaussian processes (GPs). GP regression and classification have met with recent success in their application to robot mapping, as GPs are capable of expressing rich correlation among map cells and sensor data. However, the cubic computational complexity has limited its application to

large-scale mapping and online use. We address this issue first by proposing test-data octrees within blocks of the map that prune away nodes of the same state, condensing the number of test data used in a regression, in addition to allowing fast data retrieval. We also propose a nested Bayesian committee machine which, after new sensor data is partitioned among several GP regressions, fuses the result and updates the map with greatly reduced complexity. Finally, by adjusting the range of influence of the training data and tuning a variance threshold implemented in our method's binary classification step, we are able to control the richness of inference achieved by GPs and its tradeoff with classification accuracy. In Chapter 3.4, we propose a novel approach for underwater simultaneous localization and mapping using a multibeam imaging sonar for 3D terrain mapping tasks. The high levels of noise and the absence of elevation angle information in sonar images present major challenges for data association and accurate 3D mapping. Instead of repeatedly projecting extracted features into Euclidean space, we apply optical flow within bearing-range images for tracking extracted features. To deal with degenerate cases, such as when tracking is interrupted by noise, we model the subsea terrain as a Gaussian Process random field on a Chow—Liu tree. Terrain factors are incorporated into the factor graph, aimed at smoothing the terrain elevation estimate.

In Chapter 4, two related works on SLAM are presented. In Chapter 4.1, we first present a novel approach to perform underwater simultaneous localization and mapping (SLAM) using a small inspection-class remotely operated vehicle (ROV) equipped with a single-beam scanning sonar, amidst high levels of noise present in the sonar data, and in the absence of inertial/odometry measurements. Features are extracted from hierarchically grouped clusters of sonar returns, data association is performed via the iterative joint compatibility test, and the vehicle's trajectory and map are estimated using incremental smoothing and mapping (iSAM). The resulting

point clouds derived from the ROV's sonar are used to produce Gaussian process occupancy maps, which interpolate among gaps in the acoustic range data to produce descriptive 3D maps of submerged structures. In Chapter 4.2, we study the ambiguous data association problem confronting SLAM, specifically for the autonomous exploration of environments lacking rich features. In such environments, a single false positive assignment might lead to catastrophic failure, which even robust back-ends may be unable to resolve. Inspired by multiple hypothesis tracking, we present a novel approach to effectively manage multiple hypotheses (MH) of data association inherited from traditional joint compatibility branch and bound (JCBB), which entails the generation, ordering and elimination of hypotheses.

Before introducing autonomous exploration on underwater vehicles, we first present the derivation and implementation in Chapter 5 on our ground vehicle as a proof of concept. We consider the problem of autonomous mobile robot exploration in an unknown environment for the purpose of building an accurate map efficiently. Most literature on this subject is focused on the combination of a variety of utility functions, such as curbing robot pose uncertainty and the entropy of occupancy grid maps. However, the effect of uncertain poses is typically not well incorporated to penalize poor localization, which ultimately leads to an inaccurate map. Instead, we explicitly model unknown landmarks as latent variables, and predict their expected uncertainty, incorporating this into a utility function that is used together with sampling-based motion planning to produce informative and low-uncertainty motion primitives. We propose an iterative expectation-maximization algorithm to perform the planning process driving a robot's step-by-step exploration of an unknown environment.

In Chapter 6, we conclude the dissertation by presenting the exploration framework in underwater environments. A robust SLAM solution is proposed to provide fundamental functions that an exploration algorithm needs. We employ a keyframe-

based approach to performing SLAM optimization and at each keyframe feature points are extracted and matched using improved ICP to provide constraints between poses. A local occupancy grid map is built in the local keyframe and the global map is computed and updated via an efficient submap method. Using the estimated trajectory and occupancy map, we demonstrate our proposed exploration algorithm in Chapter 5 in real and simulated underwater environments using 2D imaging sonar. We provide conclusions and future works in Chapter 7.

In summary, the main contributions of this dissertation are as follows:

- Novel algorithms for computationally efficient 3D Gaussian process occupancy mapping [14] and accurate imaging sonar-based terrain mapping [15].
- An approach for landmark-based SLAM with a single-beam scanning sonar [16] and a related solution for multi-hypothesis joint compatibility branch-and-bound data association [17].
- A novel framework for autonomous exploration that uses virtual landmarks to enforce accurate map-building under localization uncertainty [18], and its application using pose SLAM aboard an unmanned ground vehicle [19].
- A scan-matching based SLAM framework that allows the adaptation of the above framework to the autonomous exploration of underwater environments, permitting one of the first demonstrations of underwater active SLAM in an unstructured, outdoor environment.

Chapter 2
Background

2.1 Simultaneous Localization and Mapping (SLAM)

Simultaneous localization and mapping (SLAM) describes the problem of localizing a robot while simultaneously mapping the environment, provided a variety of sensor measurements, e.g. odometry measurement from an odometer, linear accelerations and angular velocities from an inertial measurement unit (IMU), and range to objects from laser range-finders. For either an unmanned ground vehicle (UGV) or a remotely operated underwater vehicle (ROV), we denote the state of the robot at the i-th time step by $\mathbf{x}_i$, with $i \in \{0, ..., N\}$. We are interested in recovering 6-D pose, position and orientation, of the robot, thus $\mathbf{x}_i \triangleq [R_i, \mathbf{p}_i] \in \mathrm{SE}(3)$, where the rotation matrix R_i belongs to 3D Special orthogonal Group, i.e., $R_i \in \mathrm{SO}3$, and position vector belongs to 3D vector space, i.e., $\mathbf{p}_i \in \mathbb{R}^3$. In general, in order to incorporate map of the surrouding environment into the state space, we represent an object, which can be preceived as point cloud, features in camera images, as a 3D point-like feature, denoted as $\mathbf{l}_j \in \mathbb{R}^3$ with $j \in \{1, ..., M\}$. The measurement is represented as $\mathbf{z}_k, k \in \{1, ..., K\}$, along with a pair of match (i_j, j_k) related to the corresponding landmark l_{j_k} observed at step $\mathbf{x}_{i_k}$. Given the set of measurements $\mathcal{Z} = \{\mathbf{z}_k\}$, the objective is to recover robot's trajectory $\mathcal{X} = \mathbf{x}_i$ and the landmark map $\mathcal{L} = \{\mathbf{l}_j\}$.

The problem can be formulated as a belief net, or Bayesian network, which is a directed acyclic graph encoding the conditional independence structure of all the random variables through factorization. The joint probability distribution with

respect to such network can be expressed as

$$P(\mathcal{X}, \mathcal{L}, \mathcal{Z}) \propto P(\mathbf{x}_0) \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \prod_{k=1}^{K} P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k}).$$

In the above equation, $P(\mathbf{x}_0)$ is a prior on the initial state, $P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)$ is the motion model parameterized by a control input $\mathbf{u}_i$, $P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k})$ is the measurement model, and we have dropped the non-informative prior on landmarks since we do not have knowledge of the map in advance. As is standard in the SLAM literature, both the motion model and the measurement model are assumed to be corrupted with zero-mean Gaussian noise. More specifically, the motion model is defined by

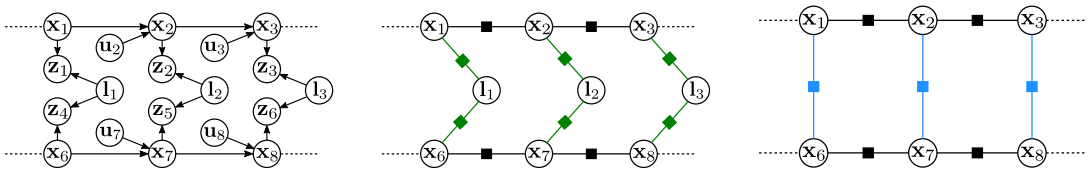
$$\begin{aligned} &\mathbf{x}_i = \mathbf{f}_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + w_i, \;\; w_i \sim \mathcal{N}(\mathbf{0}, \Lambda_i), \\ &P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \propto \exp\big(-\frac{1}{2}\|\mathbf{f}_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\Lambda_i}\big), \end{aligned} \tag{2.1}$$

where $\mathbf{f}_i$ is the state transition function, and the measurement model is defined by

$$\begin{aligned} &\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + v_k, \;\; v_k \sim \mathcal{N}(\mathbf{0}, \Sigma_k), \\ &P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \propto \exp\big(-\frac{1}{2}\|\mathbf{h}_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k\|^2_{\Sigma_k}\big), \end{aligned} \tag{2.2}$$

where $\mathbf{h}_k$ is landmark measurement function, and $\|\mathbf{e}\|_\Sigma \triangleq \mathbf{e}^T\Sigma^{-1}\mathbf{e}$ is the squared Mahalanobis distance.

The formulation using belief net is natural to model the SLAM problem, but factor graphs have better connection to the optimization process. A factor graph is a bipartite graph consisting of factor nodes $\phi_i \in \mathcal{F}$ and variable nodes $\boldsymbol{\theta}_j \in \boldsymbol{\Theta}$. Edges in a factor graph are always between factor nodes and variable nodes, and a factor graph factorizes the SLAM probabilistic function as



(a) Bayesian belief network of a landmark SLAM problem

(b) Factor graph of a landmark SLAM problem

(c) Factor graph of a pose SLAM problem

Figure 2.1: Different representations of SLAM problems.

$$P(\boldsymbol{\Theta}) \propto \prod_i \phi_i(\boldsymbol{\Theta}_i), \tag{2.3}$$

where the factors can be derived from motion and measurement models,

$$\begin{aligned} \phi_i(\mathbf{x}_{i-1}, \mathbf{x}_i) &\propto P(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i), \\ \phi_i(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) &\propto P(\mathbf{z}_k | \mathbf{x}_{i_k}, \mathbf{l}_{j_k}). \end{aligned} \tag{2.4}$$

The optimization of a factor graph follows the same process as above,

$$\begin{aligned} \boldsymbol{\Theta}^* &= \arg\max_{\boldsymbol{\Theta}} \prod_i \phi_i(\boldsymbol{\Theta}_i) \\ &= -\arg\min_{\boldsymbol{\Theta}} \sum \log \phi_i(\boldsymbol{\Theta}_i), \end{aligned} \tag{2.5}$$

which leads to a nonlinear least-squares problem,

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \left\{ \sum_{i=1}^{N} \left\| \mathbf{f}_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i \right\|^2_{\Lambda_i} + \sum_{k=1}^{K} \left\| \mathbf{h}_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k \right\|^2_{\Sigma_k} \right\}. \tag{2.6}$$

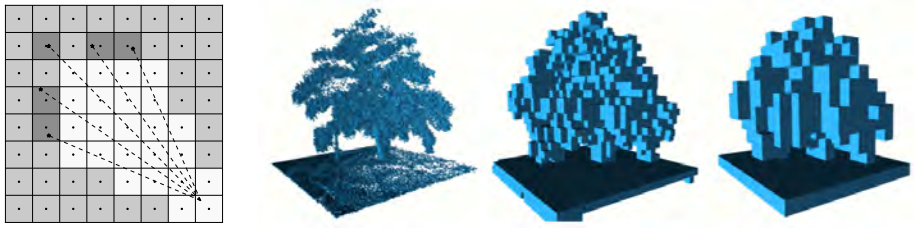
There are a variety of optimization algorithms for the nonlinear equations (e.g., Gauss-Newton, Levenberg-Marquardt), in which the estimate is iteratively improved by updating rules based on gradients. Although the system typically contains a gigantic Jacobian matrix after linearization, we can exploit the sparse structure

rooted in the SLAM problem, where majority of constraints are sequential and loop-closures don't occur frequently. Dellaert and Kaess [20] provide more detailed review of modern sparse graph SLAM solutions, and g2o [21] [22], ceres [23], gtsam [24] are open-sourced softwares for nonlinear least-squares problems.

2.2 Occupancy Mapping

Constructing maps from range sensor measurements is one of the fundamental tasks of robotic perception. Many applications such as localization and autonomous navigation (obstacle avoidance and path planning) depend on reliable and accurate map representations of the environment. In particular, descriptive maps of underwater structures and terrain is of great interest for environmental monitoring, and for port and harbor infrastructure inspection, where regular observation by divers is currently required. Besides accuracy, efficiency is another key challenge in robotic mapping. Robots equipped with volumetric laser scanners or depth cameras may generate millions of points in a single scan, which requires the efficient implementation of mapping algorithms in order to run in real-time applications.

We address the problem of generating an occupancy map from sensor observations in a static environment under the assumption that poses of a robot are known, which is viable if the mapping is accompanied by state estimation, e.g., Kalman filter, pose SLAM. We treat the occupancy status as a random variable, describing the probability of a location being occupied by obstacles. The occupancy mapping is defined as a function from cell location $\mathbf{x}_i \in \mathbb{R}^3$ to occupancy probability $m_i \in [0, 1]$, $f : \mathbb{R}^3 \rightarrow [0, 1]$. The occupancy can be classified into *occupied* with obstacles ($p(m_i) > p_{\text{occupied}}$), *free* of obstacles ($p(m_i) < p_{\text{free}}$), and *unknown* status ($p(m_i) = 0.5$) which we generally assume as priors. The goal is to infer the occupancy



(a) 2D OGM (b) Multi-resolution 3D Octomaps [25]

Figure 2.2: Demonstration of occupancy grid mapping. (a) 2D occupancy grid mapping from one sensor measurement consisting of 5 sparse scan lines. The map is colored based on occupancy probability (darker represents occupied). (b) 3D Octomap is capable of providing a multi-resolution representations for different applications.

probability from sensor observations and robot trajectory,

$$p(m_i|\mathcal{Z}, \mathcal{X}) \tag{2.7}$$

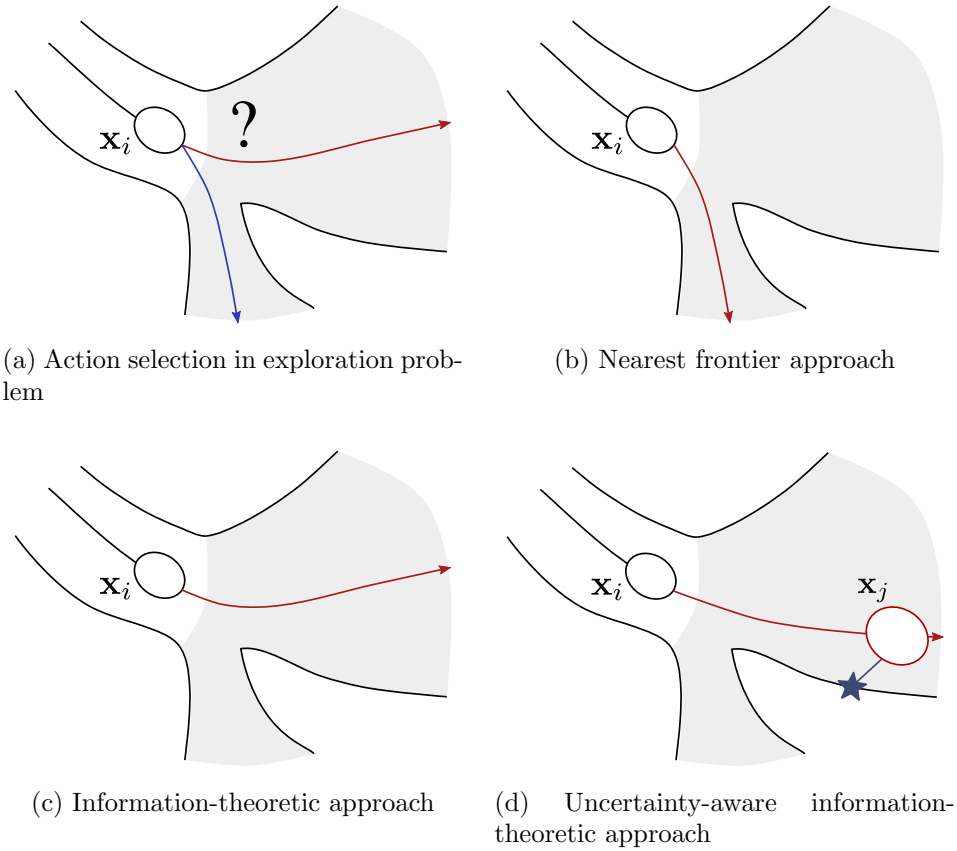
Occupancy grid mapping (OGM) [26], which represents the environment by grid cells of equal size, has been widely used in exploration and navigation tasks (Fig. 2.2a). It captures the locations of obstacles in the environment by maintaining a probability of occupancy for each cell which is updated independently and incrementally using Bayes filtering. OctoMaps [25] reduce the memory usage in occupancy grid mapping by organizing cells in an efficient data structure based on octrees. Another benefit of OctoMaps is that they can produce maps of variable resolution from the same underlying data (Fig. 2.2b). Despite the benefits, the above methods rely on the assumption that all grid cells are statistically independent, and sensor data is only correlated with grid cells directly intersected by range beams. As a result of this strict assumption, sparse sensor measurements will yield discontinuous occupancy maps between adjacent sensor views or scan lines, which may pose a threat for navigation tasks if path planners deem the gaps to be unoccupied. This has been highlighted previously as a problem that poses difficulties for robot exploration [27].

A discontinuous mapping result is shown in Fig. 2.2a, and we can observe cells that intersect with scan line are considered as occupied, yet the gap between them remains unknown.

To overcome the limitation, Gaussian processes (GPs), which is a nonparametric Bayesian learning technique used for regression and classification, were introduced into occupancy mapping [28] [29]. This demonstrated that GPs could be used to achieve rich and reliable probabilistic inference about unobserved areas; gaps in the sensor data will be assigned a probability of occupancy that is correlated with neighboring areas covered by the sensor. The approach offers greater flexibility than previous efforts to capture dependency that are not suitable for online use [30] or cannot feasibly be extended to 3D mapping problems [31] [32]. However, the major drawback of GP regression is the high computational complexity of $\mathcal{O}(N^3 + N^2M)$, where N is the number of training data and M is the number of test data. This high computational cost limits its scalability to large datasets.

Other approaches have been proposed, including Hilbert mapping [33], which uses a logistic regression classifier in conjunction with approximate kernels to achieve comparable performance to standard GP occupancy mapping in less time. A version of Hilbert maps which incrementally constructs a 3D Hilbert map online [34], somewhat mitigating the difficulty in approximating kernel computation for large 3D environments is proposed, but this method provides only occupancy probability predictions without associated variances, which may provide deeper information about prediction uncertainty. Bayesian generalized kernel has been proposed [35], leveraging recent advances in test-data structures for mapping, sparse kernels, and Bayesian nonparametric inference, which achieves cheaper computation and comparable accuracy

.



(a) Action selection in exploration problem

(b) Nearest frontier approach

(c) Information-theoretic approach

(d) Uncertainty-aware information-theoretic approach

Figure 2.3: Three classic approaches to selecting actions during exploration. In the map, mapped and unmapped areas are colored by white and gray respectively, and the trajectory and current location of the robot is denoted as a line ending with error ellipse. Existing landmark is represented as a blue star.

2.3 Autonomous Exploration

We consider the autonomous mobile robot exploration problem in unknown environments, where the robot's objective guiding exploration is to produce a map of its surroundings accurately and efficiently. In general, the autonomous exploration problem for mobile robots comprises three stages:

1. The robot identifies candidate locations to explore or paths to follow;

2. A utility function is evaluated for every candidate and the optimal one is se-

lected;

3. The robot executes the optimal action and updates its current knowledge of the environment.

The first task is commonly achieved by enumerating frontier locations, or by employing sampling-based methods such as rapidly-exploring random tree (RRT) and probabilistic roadmap (PRM), and the third task is simply performed by feedback controllers. The remaining question is how to properly formulate a utility function that effectively captures the exploration-exploitation dilemma, i.e., a balancing of visiting unknown areas to reduce map uncertainty and revisiting known areas to seek better localization. Here we make the following assumptions for simplicity:

1. There are a certain number of static landmarks, or features, objects, in the environment which can be used for localization.

2. The movement of the robot is confined in a limited space that is known in advance.

Without considering the robot's localization uncertainty, the problem has been approached by following the nearest frontier [36], choosing sensing actions to maximize mutual information (MI) [37, 38, 39, 40], using Cauchy-Schwarz quadratic mutual information (CSQMI) to reduce computation time [41], by combining global planning with local motion primitives and also refining a trajectory using optimization methods [42], and by exploring on continuous Gaussian process frontier maps [43]. The simplified problem, planning with a priori maps, has also been discussed to actively minimize the uncertainty of known landmarks [44, 45, 46]. Most of the existing research on exploration in unknown environments takes advantage of occupancy grid

maps, and this paradigm has succeeded in complex applications, including real-time 3D exploration and structure mapping with micro aerial vehicles [47, 48].

If we take into account uncertainty of robot pose or map, an integrated exploration strategy was proposed to combine different utilities [49, 50], e.g., the utility of information gain over the occupancy grid maps, the utility of travel distance to the goal, and the utility of localizability, which incorporates the uncertainty of robot poses and landmarks. Recently, an approach of improving features uncertainty while maintaining informative actions was implemented on aerial vehicles [48]. However, these methods ignore the correlation between localization and information gain. High-uncertainty poses are likely to result in inaccurate occupancy grid maps, limiting the usefulness of the information gained by exploring unknown regions.

Chapter 3

Underwater 3D Mapping using Sonar

Over the last two decades, the application of autonomous underwater vehicles (AUVs) has proliferated across challenging perceptual tasks such as pipeline and ship hull inspection, bathymetric survey, and structure mapping. Many efforts have been devoted to achieving the autonomy of underwater vehicles for such tasks, and an accurate representation of the environment is an essential prerequisite for such autonomy. To acquire such a representation of the environment, optical sensors (cameras) or acoustic sensors (sonars) are typically utilized. Although a camera can capture fine details of the underwater environment, its capability is often limited by the turbidity of the water. Additionally, illumination changes may make captured data unreliable. On the other hand, sonar (an acronym for sound navigation and ranging) will function in water that has high turbidity, offering long-range visibility and a wide aperture.

Sonar takes measurements by emitting pulses of sound and listening for echoes that are reflected back from submerged objects. However, the longer wavelength of sound in the water compared to that of light in the air greatly limits the angular resolution $\delta\beta$ of a sonar device, which can be computed as λ/L, where λ, L denote wavelength and receiver size. Additionally, the sound propagation is influenced by a number of factors, such as attenuation due to water absorption, refraction due to variable density, and scattering from the sea surface and seafloor. Another hurdle to overcome in order to apply an occupancy mapping algorithm is the loss of elevation information in the sonar measurement, which will be discussed in Sec. 3.2.

In this chapter, we first provide a literature review of the sonar mapping problem, then introduce the sonar sensor model, and two proposed approaches to perform-

ing descriptive 3D mapping under sparse sonar measurements are presented. While the first solution employs a laser-like sensor model, which naively assumes zero elevation for range measurements, the sonar is accurately modeled to conduct localization and terrain mapping in the second solution.

3.1 Related Works

One of the earliest works to produce underwater 3D maps employed a particle filter in combination with an occupancy grid map in submerged tunnels [51], using a robot equipped with a Doppler velocity log (DVL) and multiple sonar arrays. For a robot equipped with a DVL and a single-beam scanning sonar, the Hough transform was used to extract line features to produce 2D maps of man-made, structured environments [3]. Some works have used cameras to collect additional observations given sufficient illumination; for example, a camera and multi-beam profiling sonar are combined for ship hull inspection [4].

Among much of the underwater simultaneous localization and mapping literature (SLAM), mapping is limited to 2D representations of the environment [3]. However, constructing a 3D representation of the underwater environment with sonar is challenging due to its physical limitations. Sonar is plagued by high levels of noise and low resolution, and the lack of an elevation angle in a sonar measurement is another major obstacle to achieving accurate 3D mapping. In [52], two sonars mounted orthogonally on a torpedo-shaped AUV are used to support scan-matching within a SLAM framework, and the vertical sonar with a narrow beam is used for 3D mapping. In [9], point features are extracted and registered to those from other sonar images, and the pairwise transformation between images is used to constrain the vehicle. However, this work assumes that an imaging sonar is aligned with the terrain

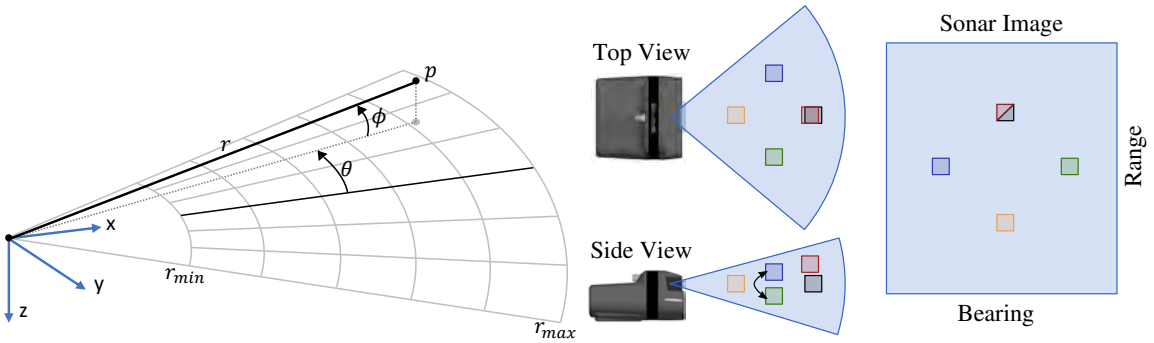
surface, otherwise the transformation is erroneous. The detailed model of a ship hull can be generated by configuring a multibeam sonar in profiling mode with narrower vertical beam-width [4]. Elevation recovery from acoustic shadows is proposed in [53], but the application is limited when shadows do not exist.

Acoustic structure from motion (ASFM) [54] is proposed to address the ambiguity of the elevation angles associated with an imaging sonar's returns from the surrounding environment. The influence of different robot motion primitives on this degeneracy is discussed in [54], [55]. Data association in ASFM based on reprojection error is proposed in [56]. Non-parametric and semi-parametric factors are introduced to handle under-constrained landmarks in [57]. Degeneracy is determined by examining the eigenvalues of the Jacobian matrix of a specific landmark. However, under-constrained landmarks are beneficial only for localization, and elevation angles can't be accurately estimated from graph optimization. The state-of-the-art ASFMs still lack a robust data association method, and a general motion primitives is required for accurate mapping.

3.2 Sonar Sensor Model

Given a feature $\mathbf{l}^s = [x, y, z]^T$ in the sensor frame represented in Cartesian coordinates, we can describe it as $\mathbf{s} = [r, \theta, \phi]^T$ in spherical coordinates, where r is the range to the sensor origin, θ is the azimuth and ϕ is the elevation angle (see Fig. 3.1). The conversion between these two forms can be expressed with the following equations:

$$\mathbf{l}^s = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\phi\cos\theta \\ r\cos\phi\sin\theta \\ r\sin\phi \end{bmatrix} \tag{3.1}$$



(a) 3D Sonar geometry (b) Example of sonar image

Figure 3.1: Imaging sonar model. A feature p can be represented as $[r, \theta, \phi]^T$ in a spherical coordinate frame. Note that the range r and the azimuth angle θ of p can be directly derived from measurements, while the elevation angle ϕ is lost in the 2D sonar image.

$$\mathbf{s} = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = h(\mathbf{l}^s) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan 2(y, x) \\ \arctan 2(z, \sqrt{x^2 + y^2}) \end{bmatrix}. \tag{3.2}$$

Though the range r and the azimuth angle θ of feature $\mathbf{l}^s$ can be directly derived from the 2D sonar image, the elevation angle $\phi = h_\phi(\mathbf{l}^s)$ is lost due to the wide vertical aperture. In other words, the mapping of a 3D world into a 2D sonar image eliminates the elevation information, which results in the ambiguity of features appearing along any $|\phi| \leq \phi_{\max}$ arc. Although the vertical aperture is significantly reduced by leveraging a lens or using a profiling sonar, a large field of view in elevation angle is often beneficial for a robot's situational awareness.

An example of sonar image is shown in Fig. 3.1b. Due to the loss of elevation, objects that have same bearing and range values will appear at the same location in the sonar image, even with at different heights (see red and black objects). Also, swapping blue and green objects has no impact on the sonar image.

3.3 Gaussian Process Occupancy Mapping

Sonar measurements are sparse, especially for a scanning sonar with fewer beams. To overcome the limitation of traditional occupancy grid mapping under sparse measurements, we first propose to use Gaussian Process occupancy mapping for rich inference over unobserved space, and some improvements are made to enable near real-time computation.

Briefly speaking, GP occupancy mapping collects sensor observations and the corresponding labels (*free* or *occupied*) as training data; the map cells comprise test data, which are related to the training data using covariance functions. After a regression is performed, we obtain a cell's probability of occupancy by "squashing" regression outputs into occupancy probabilities using binary classification functions. This procedure is detailed below. Let us consider the noisy observation model,

$$y = f(\mathbf{x}) + \epsilon, \tag{3.3}$$

where $\mathbf{x}$ is the input vector, y is the observed target value of the latent function value $f(\mathbf{x})$ added with noise $\epsilon \in \mathcal{N}(0, \sigma_n^2)$. A Gaussian process is defined as a distribution over functions [58],

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.4}$$

with mean function $m(\mathbf{x})$ which we take as zero and covariance function $k(\mathbf{x}, \mathbf{x}')$. Given n training points $X = \{\mathbf{x}_i\}_{i=1}^n$ and observation vector $\mathbf{y}$, a Gaussian process predicts the latent values $\mathbf{f}_*$ at m test points $X_* = \{\mathbf{x}_{*i}\}_{i=1}^m$ to be

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)). \tag{3.5}$$

Here we use a compact notation to denote covariance matrices: $K = K(X, X), K_* = K(X, X_*), K_{**} = K(X_*, X_*)$. The mean and covariance of the Gaussian process can be written as

$$\bar{\mathbf{f}}_* = K_*^T (K + \sigma_n^2 I)^{-1} \mathbf{y}, \tag{3.6}$$

$$cov(\mathbf{f}_*) = K_{**} - K_*^T (K + \sigma_n^2 I)^{-1} K_*. \tag{3.7}$$

In the case of 3D occupancy mapping, we have a set of training input data X, where $\mathbf{x}_i \in \mathbb{R}^3$ is an *occupied* point in the environment observed by the sensor or a *free* point sampled along sensor beams, and an accompanying set of training target data Y, where $\mathbf{y}_i = 1$ or -1 for *occupied* and *free* points respectively.

3.3.1 Covariance Functions

Covariance functions, or kernel functions, define the similarity between a pair of points: if they are close to each other, they tend to have the same target value [58]. A common choice for GPs is to use a squared exponential covariance function [58], and a sparse covariance function was introduced in [59] that achieves comparable smoothness, while reducing to zero correleration when two points are a specified distance apart. However, such functions are often too smooth to capture the sharp variations in occupancy that typically occur in real-world mapping scenarios. To address this issue, the Matérn covariance function has been applied successfully in GP occupancy mapping [60] [43]. The Matérn covariance function, with its smoothness parameter set to $\nu = 3/2$, is defined as

$$k(r) = \sigma_f^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(\frac{-\sqrt{3}r}{l}\right), \tag{3.8}$$

where $r = ||\mathbf{x} - \mathbf{x}'||$, and the two hyperparameters σ_f^2 and l are prior signal variance and characteristic length-scale. It's worth noting that the correlation drops quickly as the distance between points increases. Based on this property, in this paper we approximate Gaussian process regression by using a subset of the training data; specifically, we only utilize nearby training data to predict the value of a given test point.

3.3.2 Binary Classification Functions

In order to "squash" the output of GP regression into an occupancy probability in the range of $[0, 1]$, we adopt a logistic regression model that leverages both mean μ_i and variance σ_i^2 at every test point [43],

$$p(y = 1|\mathbf{x}_i) = \frac{1}{1 + \exp(-\gamma\omega_i)}, \tag{3.9}$$

where $\omega_i = \sigma_{min}^2 \mu_i / \sigma_i^2$ is the weighted mean, σ_{min}^2 is the minimum variance, and γ is a positive constant. A classification is performed by combining the resulting occupancy probability from (7) and variance of the predicton from (5),

$$state = \begin{cases} free & p < p_{\text{free}}, \sigma_i^2 < \sigma_t^2 \\ occupied & p > p_{\text{occupied}}, \sigma_i^2 < \sigma_t^2 \\ unknown & otherwise \end{cases} \tag{3.10}$$

where σ_t^2 is the variance threshold, which expresses the confidence we have in the predicted occupancy probability.

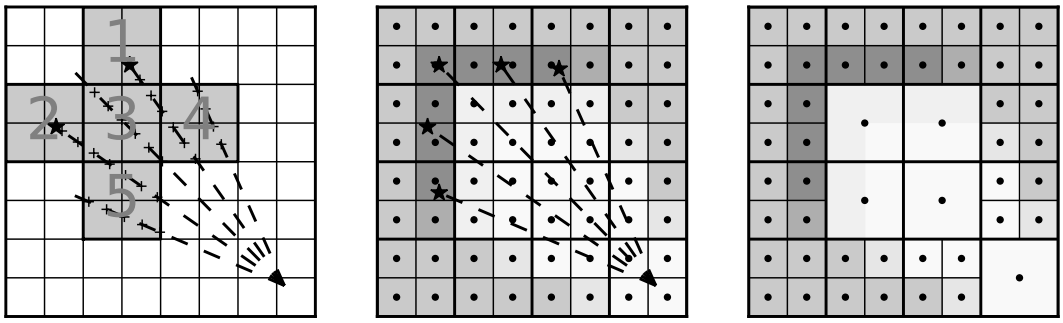


Figure 3.2: 2D demonstration of our approach with an example laser scan, where dots represent test points and the stars and crosses along scan lines represent training data (hits and free points, respectively). *Left*: Setup for predicting the occupancy probability of Block #3. The extended block consists of 5 blocks for which separate regressions are performed using the data contained in each; the resulting predictions of Block #3 from each regression are fused via BCM. *Middle*: GP occupancy mapping with test-data octrees before pruning. There are 16 depth-two octrees which are separated by thick grid lines in the map. *Right*: Final map after pruning.

3.3.3 Spatial Partitioning with Test-Data Octrees

Our choice of data structures for occupancy mapping will have important implications for the method's time and memory complexity. By representing the environment using equally sized grid cells, we can achieve constant access and write time. However, the downside is that memory consumption grows cubically in the dimension of the environment. An alternative is to use octrees, data structures which recursively partition a space into groups of smaller, equally sized nodes such that every node in the tree has eight children. Octrees are a memory efficient approach for representing the entire 3D environment because the partitioning process need only be implemented in spaces containing observed structures, yielding a multi-resolution map. However, as a result of its tree structure, octrees suffer from an access complexity of $O(log(d))$, where d is the maximum depth of the tree.

Octrees have been applied successfully to 3D occupancy grid mapping in the OctoMap framework [25]. By casting rays in the octree according to a range sensor's

observations, OctoMap incrementally updates occupancy probabilities of the cells that rays pass through. In an OctoMap, inserting or partitioning cells occurs in the vicinity of sensor beams only, which conforms to the philosophy of octrees: partitioning as needed. Unlike occupancy grid mapping, GP occupancy mapping attempts to predict the contents of many spaces where sensor data has not directly landed, and without knowing where the boundaries in occupancy will lie, we must deploy a fine discretization everywhere. In this case, a conventional octree is inappropriate for storing map information; it offers advantages neither in space nor in time considering that all leaf nodes need to be partitioned and initialized in the beginning.

Therefore we present a new data structure called *test-data octrees*, which are pruned to lower resolution over time instead of branching to a higher resolution. When all sibling nodes within the outermost layer of an octree attain the same designation as either *free* or *occupied*, these nodes are pruned from the tree and only their parent remains. This is the case not only for representing obstacles in the map, but also for the test data used repeatedly in Gaussian process regressions from one robot measurement to the next. In instances where the octree is pruned, eight test data points, each located at the centroid of its respective grid cell, are reduced to one point located at the centroid of the parent cell. This is illustrated in a two-dimensional example at the bottom of Figure 3.2 using quadtrees. In addition to reducing the size of the test data, the map's memory usage can be reduced by pruning nodes with the same state (*free* or *occupied*). In Figure 3.2, 20 free nodes are pruned, which means that 23% of the memory in use can be freed. Among other general advantages are the fact that we can initialize octrees with pointers, and thus octree cells can be dynamically allocated as a robot explores an unknown environment, and we can retrieve test points and update their states in near-constant time if a tree has a small number of layers.

3.3.4 Nested Updates with BCMs

Algorithm 1: GPOctoMap-NBCM

$\mathcal{M} \leftarrow \{\mathcal{B}_i = \emptyset\}, i = 1, \ldots, K$;
$\mathcal{G} \leftarrow \{g_i = \emptyset\}, i = 1, \ldots, K$;
while $!MappingComplete$ **do**
  $\mathcal{Z}_n \leftarrow SensorHits()$;
  $\mathcal{M}_n \leftarrow \emptyset$; `// local blocks need to be updated`
  for $\mathcal{B}_i \in \mathcal{M}$ **do**
    if $Intersect(\mathcal{B}_i, \mathcal{Z}_n)$ **then**
      $\mathcal{M}_n \leftarrow \mathcal{M}_n \cup \mathcal{B}_i$;
  for $\mathcal{B}_i \in \mathcal{M}_n$ **do** `// training in blocks`
    if $IsEmptyOctree(\mathcal{B}_i)$ **then**
      $\mathcal{B}_i \leftarrow CreateOctreeInBlock(\mathcal{B}_i)$;
    $X, \mathbf{y} \leftarrow GetTrainingPoints(\mathcal{Z}_n)$;
    $g_i \leftarrow GPRegression(X, \mathbf{y})$;
  for $\mathcal{B}_i \in \mathcal{M}_n$ **do** `// BCM fusion in extended blocks`
    for $\mathcal{B}_e \in FindExtendedBlocks(\mathcal{B}_i)$ **do**
      $X_* \leftarrow GetOctreeLeafNodes(\mathcal{B}_e)$;
      $\mathbf{f}_*, \mathbf{var}_* \leftarrow GPPredict(g_e, X_*)$;
      $\mathcal{B}_i \leftarrow BCMUpdate(\mathcal{B}_e, \mathbf{f}_*, \mathbf{var}_*)$;
  for $\mathcal{B}_i \in \mathcal{M}_n$ **do**
    $PruneNodes(\mathcal{B}_i)$;
$OccupancyProbs \leftarrow \emptyset$;
for $\mathcal{B}_i \in \mathcal{M}$ **do**
  if $!IsEmptyOctree(\mathcal{B}_i)$ **then**
    $\mathbf{x}_*, \mathbf{f}_*, \mathbf{var} \leftarrow GetOctreeLeafNodes(\mathcal{B}_i)$;
    $\{\mathbf{x}_*, \mathbf{p}\} \leftarrow BinaryClassification(\mathbf{f}_*, \mathbf{var}_*)$;
    $OccupancyProbs \leftarrow OccupancyProbs \cup \{\mathbf{x}_*, \mathbf{p}\}$;
return $OccupancyProbs$;

Using a Bayesian Committee Machine (BCM) [61], a separate GP regression may be performed over each newly arrived sensor observation, and the resulting estimates can be fused, either in batch or one new observation at a time. Suppose the entire training space D is split into K data sets such that $D_i = \{X_i, \mathbf{y}_i\}$, for

$i = 1, \ldots, K$. The formulation of the BCM may then be expressed as:

$$m(\mathbf{f}_*|D) = cov(\mathbf{f}_*|D)^{-1} \sum_{i=1}^{K} cov^{-1}(\mathbf{f}_*|D_i) m(\mathbf{f}_*|D_i), \tag{3.11}$$

$$cov^{-1}(\mathbf{f}_*|D) = -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} cov^{-1}(\mathbf{f}_*|D_i). \tag{3.12}$$

We will approximate the resulting covariance matrices with variance matrices, removing off-diagonal indices as proposed in a prior application of this method [62]. This prevents the potential for cubic complexity in the number of test data, while still allowing rich correlation to be expressed relating a subset of training data to every point of the test data.

In addition to leveraging the BCM to update the map one measurement at a time, as employed in [62] and [43], the occupancy information revealed by a new measurement is recovered in a modular manner, over a series of several GP regressions. First, we determine how many distinct, non-overlapping sets of test data will be updated with training data from the new measurement. In our *conservative* approach to GP occupancy mapping, in which accurate classification is the goal, the test data is drawn separately from every *block* intersected by one or more range beams. Blocks are comprised of several occupancy grid cells, and are maintained at the same resolution as the parent cells in the outermost layer of our test-data octrees. For every block of test data, the corresponding training data is comprised of all portions of the new measurement's range beams that pass through the block's *extended block*, a larger surrounding region that allows more distant training data to influence a block's test data. The notion of using an extended block to derive the training data applied to an individual block's test data was first suggested in [62].

We define an extended block to be the set of neighboring blocks with faces

adjacent to the block containing the test data of interest. An example of this definition is given in Figure 3.2. In our *aggressive* approach to GP occupancy mapping, we perform a GP regression for the test data in every block for which range beams pass through some portion of the *extended* block. This extends the influence of a new measurement to test data in neighborhoods that were not necessarily observed by the sensor. Our aggressive approach also implements a higher variance threshold in Eq. 3.10 than the conservative approach.

Although the size of a given set of training data is reduced significantly by partitioning the space into blocks and confining training points from a new measurement to extended blocks, the result may often be unsuitable for real-time mapping applications (see [62] or Table 3.1). In the results to follow in Table 3.1, the parameterization labeled BCM-NP can be regarded as Kim's method [62] with different covariance and logistic functions to provide a one-to-one comparison with our proposed method. The problem can be further divided-and-conquered, however, by applying a BCM to a single block's regression, and splitting apart the training data that lies in an extended block. A separate GP can be trained for each subset of the training data, using a BCM to fuse the results and predict occupancy probabilities in each block. This second-layer BCM is formulated according to the following equations:

$$m(\mathbf{f}_*|D_i) = cov(\mathbf{f}_*|D_i)^{-1} \sum_{j=1}^{E} cov^{-1}(\mathbf{f}_*|D_{ij}) m(\mathbf{f}_*|D_{ij}),$$

$$cov^{-1}(\mathbf{f}_*|D_i) = -(E-1)\sigma_f^{-2} + \sum_{j=1}^{E} cov^{-1}(\mathbf{f}_*|D_{ij}),$$

where E is the number of segments into which an extended block's training data is partitioned (3.2). At this level, we do not further partition the test data because we want the diverse neighborhood of training data in the extended block to be applied

to all test data. The BCM approximation, as a transductive learning method, will weigh the training data appropriately according to their proximity and relevance to the test data [63].

By performing separate regressions for test data segregated into *blocks*, whose respective training data are segregated into *extended blocks*, the computational complexity of a regression decreases from $O(N^3+N^2M)$ to $O(N^3/K^2+N^2M/K^2)$, where K is the number of blocks. By dividing each extended block's training data into E segments, and fusing the result with a BCM, the complexity is further reduced to $O(N^3/(K^2E^2)+N^2M/(K^2E))$. It should also be noted that this procedure affords the option of parallelizing the computation of the many small, separate GP regressions required to update the map with a new sensor observation. The full procedure proposed in this paper, including both the use of test-data octrees and the nested application of BCMs to the map update, is summarized in Algorithm 1.

3.3.5 Experiments on UGVs

In this section, we demonstrate our method using simulated experiments, which were conducted in Gazebo where the ground truth for both maps and robot poses were known precisely, and on three large-scale real datasets: Freiburg-079 Corridor[1], Freiburg Campus[1], and SpaceBot Arena[2]. We sub-sampled the raw data using VoxelGrid filtering in Point Cloud Library (PCL) [64] with resolution $0.1m$. As for OctoMaps, we used the original point cloud without sub-sampling.

As a consequence of these limited observations, the simulated robot did not obtain a complete scan of the environment. When a laser beam swept over flat surfaces, e.g., walls and floors, gaps between beam returns increased. As a result, the

[1] `http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/`
[2] `http://www.ais.uni-bonn.de/mav_mapping/`

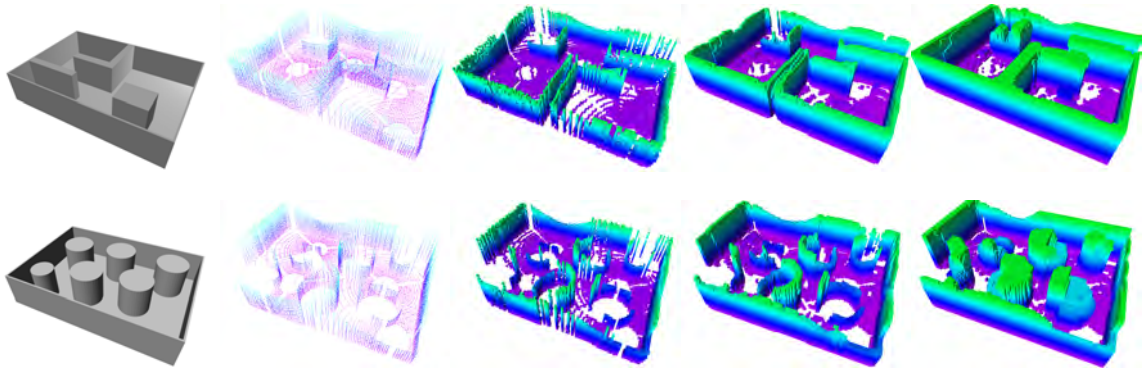


Figure 3.3: Mapping results in simulated "structured" (*top row*) and "unstructured" (*bottom row*) environments. From *left* to *right*: ground truths, accumulated raw point clouds, OctoMap, conservative GP mapping, and aggressive GP mapping. All maps are generated with $0.1m$ resolution and cells with occupancy probability > 0.7 are visualized with colors indicating cell height.

raw data contained a substantial number of gaps (second column in Fig. 3.3). The figures in the third column of Fig. 3.3 depict conventional occupancy grid mapping with OctoMap; free-space gaps remain in the final result while the predicted occupied cells precisely correspond to laser hits in the raw data, which is of limited utility for further tasks. In contrast, our proposed method reasonably filled in the volumes between adjacent laser scans, which resulted in smooth surfaces (fourth column in Fig. 3.3). This spatial inference capability was enhanced in our "aggressive" mapping results (fifth column of Fig. 3.3): our method was largely successful in predicting occupancy in the regions devoid of measurement.

We further analyze the accuracy of our proposed method. With full knowledge of ground truth, the accuracy of the maps can be inspected using Receiver Operating Characteristic (ROC) curves. ROC curves in Fig. 3.4 show that our conservative method GPOctoMap-NBCM outperforms OctoMap in two ways: 1) GPOctoMap-NBCM has fewer incorrectly classified negatives (occupied cells), which yields a lower false positive rate in the unstructured environment; 2) while maintaining a low false

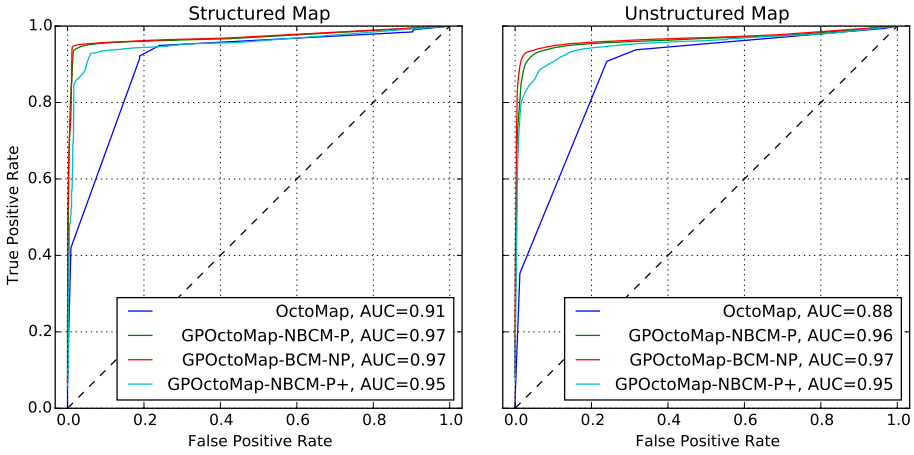


Figure 3.4: Receiver operating characteristic (ROC) curves for occupancy mapping with simulated data in the "structured" and "unstructured" environments depicted in Figure 3.3. *Free* is considered to be a positive label. See Table 3.1 for the meanings of the labels in the plot legends.

positive rate, GPOctoMap-NBCM is able to achieve a higher true positive rate, or *recall*, which indicates that GPs can correctly predict more positives (free cells). With regards to aggressive mapping, we see an increase in the false positive rate and a decrease in the true positive rate, which can be explained by the fact that its predictions are not entirely correct. GP occupancy mapping without test-data octrees and nested BCMs marginally outperforms our "conservative" approach, but at the cost of requiring more than an order of magnitude of additional computation time, as detailed in Table 3.1.

For the real datasets, as shown in Fig. 3.5, GPOctoMap-NBCM improved upon OctoMap's coverage by successfully inferring the contents of gaps to a large extent. It's worth noting that our "aggressive" GP mapping is able to predict most of the ceiling in the FR-079 dataset. The richness of inference signals at least two potential applications of GPOctoMap-NBCM. First, it is able to build a map with improved quality using only a subset of sensor measurements, which means a cheap and low-resolution sensor can be used for mapping. Second, the time spent on scanning

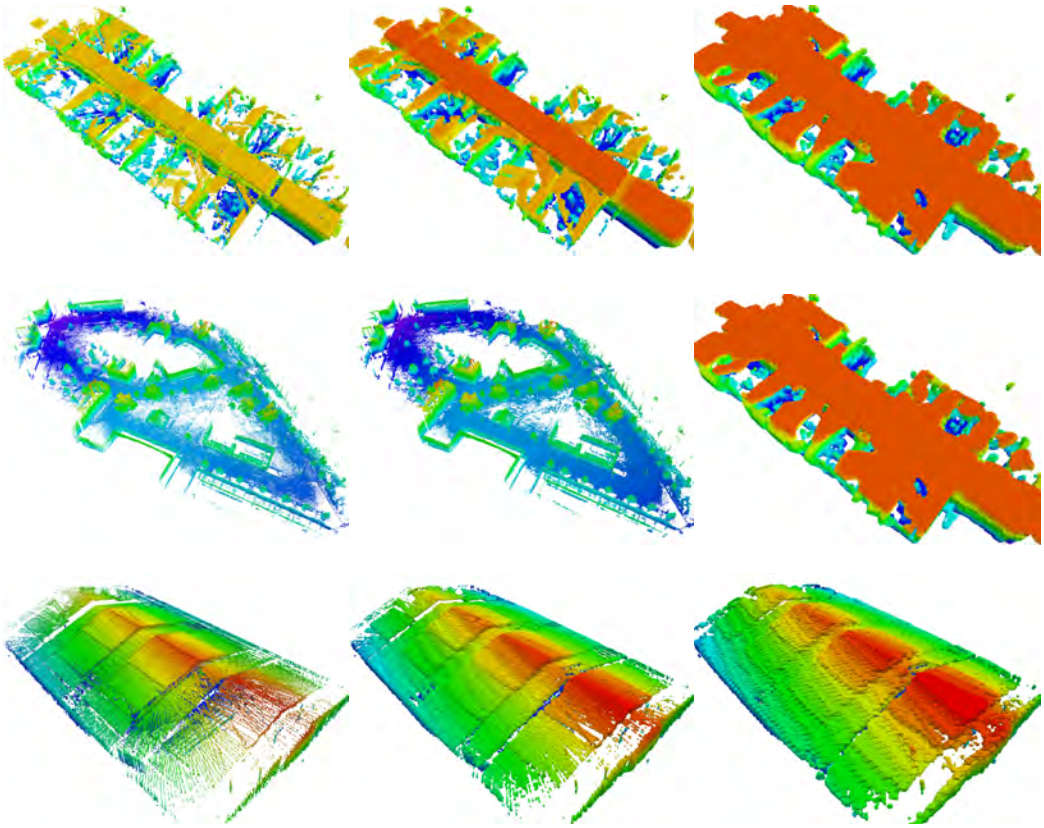


Figure 3.5: 3D maps built with OctoMap (*first column*) and GPOctoMap tuned 1) conservatively (*second column*) and 2) aggressively (*third column*). From *top* to *bottom*: Freiburg-079 corridor dataset, Freiburg campus dataset and Spacebot Arena dataset. All maps are built at $0.1m$ resolution and cells are colored corresponding to cell height.

Dataset (m^3)	Scans	Points/Scan	Method	Time (s)	Time/Scan (s)
Sim Structured Data $10.0 \times 7.0 \times 2.0$	12	3500	BCM-NP	6.57	0.55
			NBCM-P	0.44	0.04
			NBCM-P+	0.37	0.03
			OctoMap	0.20	0.02
FR-079 $43.8 \times 18.2 \times 3.3$	66	89445	BCM-NP	724.10	10.97
			NBCM-P	10.96	0.17
			NBCM-P+	9.57	0.15
			OctoMap	6.71	0.10
FR Campus $292.0 \times 167.0 \times 28.0$	81	247817	BCM-NP	N/A	N/A
			NBCM-P	329.95	4.07
			NBCM-P+	294.62	3.64
			OctoMap	242.88	3.00
Spacebot Arena $72.3 \times 71.4 \times 12.9$	8	296734	BCM-NP	1189.42	148.68
			NBCM-P	33.06	4.13
			NBCM-P+	27.35	3.42
			OctoMap	33.65	4.21

Table 3.1: Benchmarking of experimental results. Sub-sampled data are used for GP occupancy mapping whereas OctoMap uses original full-density data. In the table, "NBCM" refers to our proposed nested BCM, which includes BCMs applied within an extended block and also to sequential observations; "BCM" only applies fusion to sequential observations. Methods having "P" use test-data octrees to prune nodes with the same state; "NP" means no pruning. We also explore the "aggressive" variant of our GP mapping formulation, which is denoted as "+".

can be reduced without the need to cover the whole space. In addition to being superior in runtime to prior variants of GP occupancy mapping (represented by the BCM-NP parameterization), the runtime of GPOctoMap-NBCM is also comparable to OctoMap's, as shown in Table 3.1. In the spacebot arena dataset, GPOctoMap-NBCM even exceeds OctoMap in runtime because there exists a large volume of free space, which dramatically reduces the number of test data through octree pruning.

3.4 Terrain Mapping with a Multi-Beam Imaging Sonar

Imaging sonar significantly improves the perception capability compared to a scanning sonar; it has a wide horizontal aperture with hundreds of sensing beams, and it's able to acquire measurements at a higher rate. However, imaging sonar still suffers from the loss of elevation information due to a wide vertical aperture. Although 3D locations can't be deduced from a single image, which is composed of two types of information, bearings and ranges, it's feasible to estimate feature positions from multiple views of the scene. In this section, we show that by enforcing a terrain model on the mapped environment an accurate 3D map is achieved through SLAM optimization in which the data association is provided by optical flow.

We formulate feature-based SLAM as a least-squares problem using the same notation as in Sec. 2.1. We assume a Gaussian measurement model between robot state $\mathbf{x}_i \in \mathcal{X}$ and feature position $\mathbf{l}_j \in \mathcal{L}$, and we assume that the process model given an input $\mathbf{u}_i \in \mathcal{U}$ is as follows:

$$\mathbf{x}_i = f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}_i, \;\; \mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \Lambda_i), \tag{3.13}$$

$$\mathbf{z}_k = h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + \mathbf{v}_k, \;\; \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \Gamma_k), \tag{3.14}$$

where i_k, j_k denote the associated state and feature corresponding to the k-th measurement (see Sec. II.C). The evolution of state is modeled by f using measurements from navigation sensors, including an inertial measurement unit (IMU) and Doppler velocity log (DVL). The observation is predicted in h by transforming features from the global frame to the sonar frame parameterized in spherical coordinates, or formally, $h(\mathbf{x}, \mathbf{l}) = h(\mathbf{l}^s)$ in Eq. 3.2. The estimate is obtained by solving the nonlinear

least-squares equation,

$$\mathcal{X}^*, \mathcal{L}^* = \arg\min_{\mathcal{X},\mathcal{L}} \sum_i ||\mathbf{x}_i - f_i(\mathbf{x}_{i-1}, \mathbf{u}_i)||^2_{\Lambda_i} + \sum_k ||\mathbf{z}_k - h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})||^2_{\Gamma_k}.$$

In previous work on 3D SLAM using imaging sonar, no constraints are imposed on the elevation angle, and its ambiguity is clarified by solving the least-squares equations, which may contain measurements of the same feature from a variety of perspectives. However, in practice, the nonlinear system regularly becomes ill-posed without a proper initial estimate and sufficient sensor motion [54, 55]. Advanced multi-beam sonars feature a narrow vertical aperture (12° in our experiments), and given the mounting configuration shown in Fig. 3.11a, elevation angles of observed features are distributed symmetrically around zero. Therefore in this work, an imaginary elevation angle $\tilde{\phi}$ is incorporated into the measurement vector $\mathbf{z} = [r, \theta, \tilde{\phi}]^T$, which is varied based on the predicted elevation $h_\phi(\mathbf{l}^s)$,

$$\tilde{\phi} = \begin{cases} \hat{\phi}, & : |h_\phi(\mathbf{l}^s)| \leq \phi_{\max} \\ 0, & : |h_\phi(\mathbf{l}^s)| > \phi_{\max}. \end{cases} \tag{3.15}$$

The measurement noise covariance matrix is simplified to $\Gamma = \text{diag}([\sigma_r^2, \sigma_\theta^2, \sigma_{\tilde{\phi}}^2])$, with the standard deviation of elevation noise set to half the vertical aperture, and $\phi_{\max} = 3\sigma_\phi$.

3.4.1 Feature Tracking

As a variety of noise exists in sonar images, such as Gaussian, impulse and speckle noise, we rely on the A-KAZE feature detector [65] as in previous work on this subject [57], which is designed to describe features at different smoothing scales while

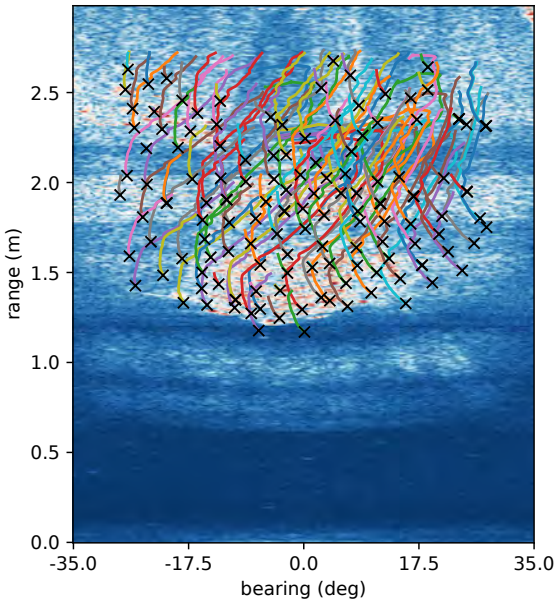


Figure 3.6: Feature tracking example in sonar frame Fig. 3.12. Extracted A-KAZE features are labeled as black ×, and the corresponding feature tracks are shown in colored lines.

retaining details. An example of extracted A-KAZE features is shown in Fig. 3.6.

The correspondence (i_k, j_k) between features detected in different sonar frames has been solved through data association techniques [56, 10] and point cloud registration [9]. Data association requires the computation of feature uncertainty in 3D space, which is computationally expensive in the presence of dense features. The ambiguity of elevation angles also presents a challenge.

In this work, our ROV moves at a relatively slow speed, and we seek to match features by tracking based on optical flow. Optical flow, specifically the Lucas-Kanade method [66], is developed on two assumptions: (1) feature intensities do not change between consecutive frames, and (2) neighboring pixels have similar motion. Although acoustic returns from objects at different elevation angles have different intensities, the assumptions hold well in practice with sonar images acquired at high frequency and with slow motion.

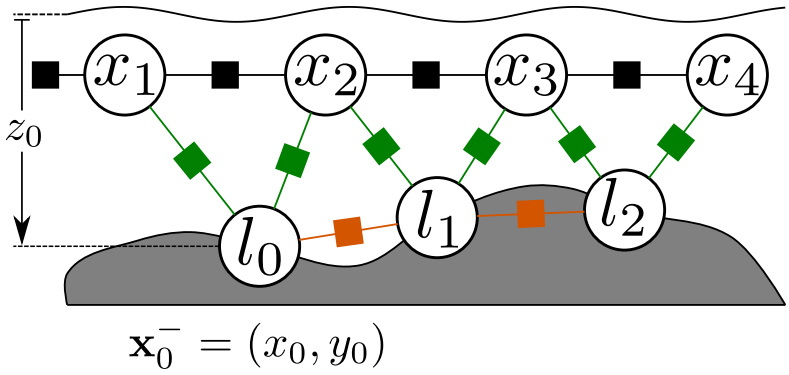
The feature tracking works as follows. We extract A-KAZE features that are used for tracking in the initial frame, and new features are introduced if they are not in close proximity to current features. Tracking of a feature stops when the minimum eigenvalue criterion isn't satisfied in the Lucas-Kanade method, or the distance between descriptors computed at previous and current feature locations is larger than a designated threshold. The tracking history of A-KAZE features is visualized in Fig. 3.6. Feature motions are consistent with the trajectory, but tracking is error-prone when the range to a feature is larger than 2.0 meters. One reason is that sonar has limited angular resolution, causing sonar imagery further away from the origin to be blurrier along the bearing-axis. There are also a few short feature tracks that don't start from the top of an image due to tracking failure. To some extent, estimation error is attributed to the insufficient measurement constraints from short tracks.

3.4.2 Gaussian Process Terrain Models

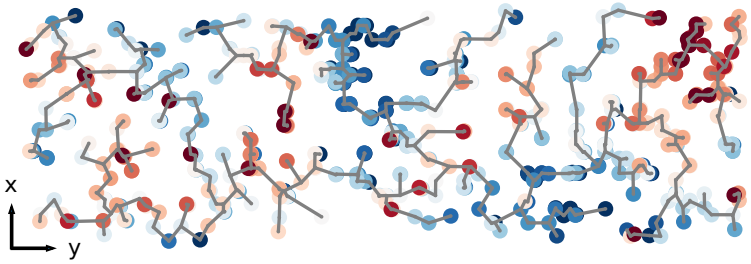
In this section, we discuss the Gaussian Process terrain model and the approximation methods required for it to be integrated into a factor graph.

We frame the mapping part of SLAM as a terrain modeling problem. Let $\mathbf{x}^- = [x_i, y_i]^T$ be the $x-y$ location of feature $\mathbf{l}_i$, let $m_i = z_i$ be the height of the feature, which is defined as the vertical distance from the water surface to the feature in Fig. 5.4, and let $X^-_{M\times 2}$ and $\mathbf{m}_{M\times 1}$ be the 2D location matrix and height vector for feature set $\mathcal{L}$. The terrain model maps 2D location to height $m = g(\mathbf{x}^-)$.

Gaussian Processes (GPs) are a non-parametric approach to learn a latent function enforcing correlation among input data. We use Gaussian Process regression to model the terrain data [67], and thus feature heights are spatially dependent in



(a) Factor graph with terrain factors



(b) Terrain factor construction using CLT

Figure 3.7: Terrain factors (orange) connecting two landmark nodes are constructed on the edges of a Chow–Liu tree. The tree is built with projected 2D features $\{\mathbf{x}_i^-\}$, and nodes are colored by optimized terrain height z_i for visualization.

spite of the fact that they are independently observed and tracked. Mathematically,

$$g(\mathbf{x}^-) \sim \mathcal{GP}(0, k(\mathbf{x}^-, \mathbf{x}^{-\prime})), \tag{3.16}$$

in which the mean function is zero, and $k(\cdot, \cdot)$ is the covariance function that defines the similarity between a pair of height variables. Under the GP assumption, the aggregated height vector is distributed as a joint Gaussian distribution with dense covariance matrix,

$$\mathbf{m} \sim \mathcal{N}(\mathbf{0}, K(X^-, X^-)). \tag{3.17}$$

The predicted mean at any location given existing observations is expressed as

$$\mathbb{E}[\mathbf{m}_*] = K(X_*^-, X^-)K^{-1}(X^-, X^-)\mathbf{m}. \tag{3.18}$$

GPs bring several benefits to the terrain reconstruction problem. Under-constrained landmarks can be handled more effectively, whether this stems from the fact that feature detection is less accurate on sonar images with low signal-to-noise ratio, or from an overly simplistic motion a robot has executed. The addition of a GP model imposes constraints on the shape of the terrain formed by the observed features. The impact of a GP terrain model is depicted in Fig. 3.8, where 20 samples are drawn from the distributions $\prod \mathcal{N}(\phi_i|\tilde{\phi}_i, \sigma_\phi^2)$ and $\prod \mathcal{N}(\phi_i|\tilde{\phi}_i, \sigma_\phi^2)\mathcal{N}(\mathbf{m}|\mathbf{0}, K(X^-, X^-))$. Here, we only consider range and elevation angle, and we use a feature's true elevation as a mean value. In addition to the above advantages, features are sparse in underwater environments, thus leaving gaps between 3D points. GP regression enables rich and reasonable inference in regions without measurements.

3.4.3 Terrain Factors

The GP model is essentially a giant factor involving all landmark nodes, and the optimization cost is prohibitive. In order to implement correlated terrain factors while maintaining sparsity of the factor graph, we approximate the full GP model with a Gaussian Process random field [68] defined on a tree structure. A product of conditional distributions is implied from a tree structure,

$$p(\mathbf{m}) \approx q_{\text{tree}}(\mathbf{m}) = p(m_{\text{root}}) \prod_{i \neq \text{root}} p(m_i|m_{\pi_i}), \tag{3.19}$$

where π_i is the parent of node i.

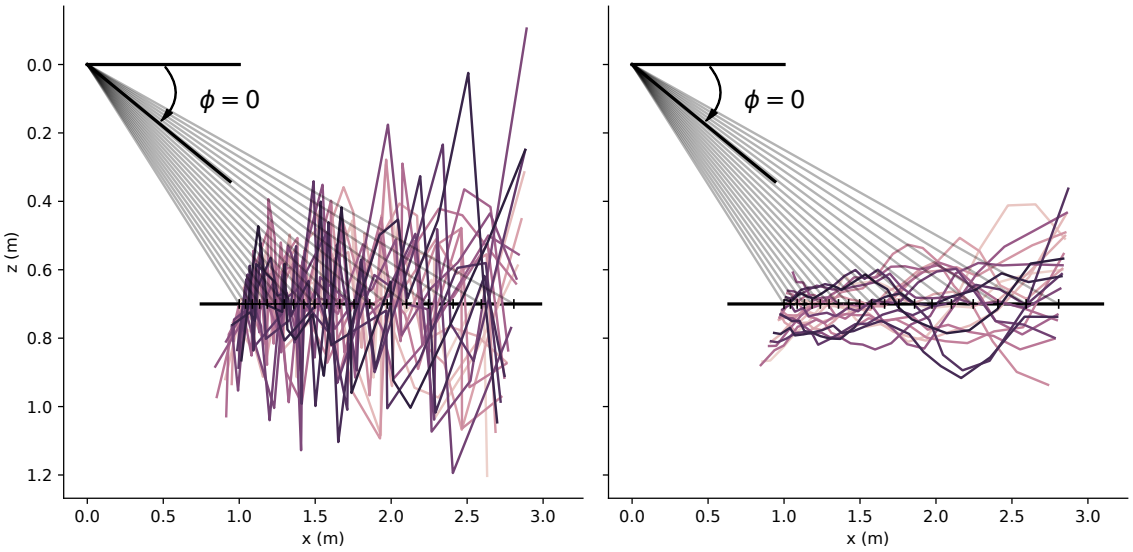


Figure 3.8: Terrain height estimates resulting from 20 noise-corrupted measurements of the 20 features in the sonar's field of view, both without (left) and with (right) a GP terrain model. The actual (flat) ground is marked in black while terrain estimates are colored. The zero-elevation plane of the sonar is represented as a black line.

Given the assumption that $[m_i, m_{\pi_i}]^T$ are distributed as a Gaussian process, we can formulate the joint distribution and conditional distribution as

$$\begin{bmatrix} m_i \\ m_{\pi_i} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{ii} & k_{i\pi_i} \\ k_{\pi_i i} & k_{\pi_i \pi_i} \end{bmatrix} \right), \tag{3.20}$$

$$\begin{aligned} m_i | m_{\pi_i} &\sim \mathcal{N}(\mu_{i|\pi_i}, \Sigma_{i|\pi_i}) \\ &= \mathcal{N}(k_{i\pi_i} k_{\pi_i \pi_i}^{-1} m_{\pi_i}, k_{ii} - k_{i\pi_i}^2 k_{\pi_i \pi_i}^{-1}), \end{aligned} \tag{3.21}$$

where k_{ij} is defined as $k(\mathbf{x}_i^-, \mathbf{x}_j^-)$. Accordingly, the terrain constraint is expressed as

$$m_i = \mu_{i|\pi_i} + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \Sigma_{i|\pi_i}). \tag{3.22}$$

The terrain factors are error functions between two elevation variables, which in turn

are incorporated into the least-squares minimization in Eq. 3.15,

$$||m_{i=\text{root}}||^2_{k_{ii}} + \sum_{i \neq \text{root}} ||m_i - \mu_{i|\pi_i}||^2_{\Sigma_{i|\pi_i}}. \tag{3.23}$$

3.4.4 Implementation Details

The tree structure relating a set of features is determined using the Chow-Liu algorithm [69, 70], which constructs the Chow–Liu tree (CLT) by minimizing the Kullback–Leibler divergence between the joint Gaussian distribution and the approximated distribution. In essence, the problem is to find the maximum mutual information spanning tree on a graph, which contains edges connecting any two 2D points with weights defined by the mutual information between two random elevation variables,

$$I(m_i, m_j) = -\frac{1}{2}\log(1 - \rho^2), \tag{3.24}$$

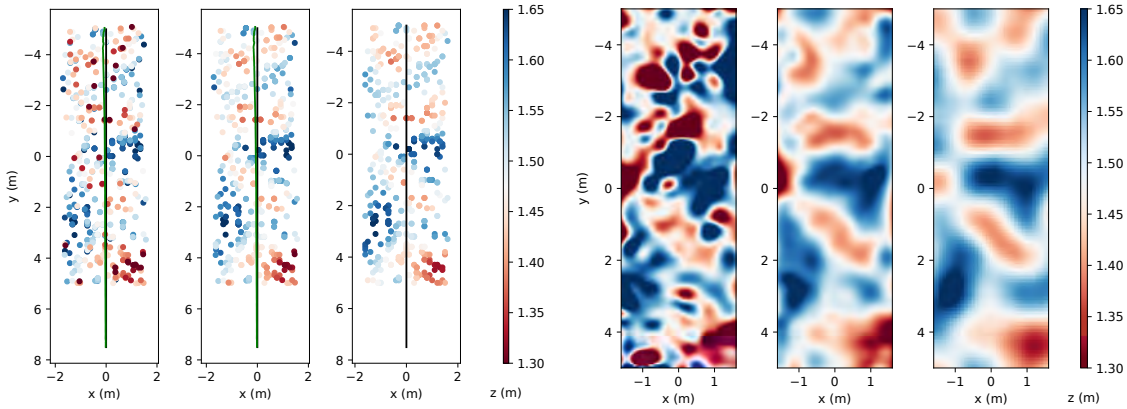
where $\rho = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}}$ is the correlation coefficient. The mutual information is monotonically decreasing with respect to the Euclidean distance between two 2D points if we use a stationary covariance function. As a consequence, the maximum mutual information spanning tree is equivalent to the Euclidean minimum spanning tree, and searching can be limited to edges in a Delaunay triangulation of 2D points.

The terrain factor requires the 2D location $\mathbf{x}^-$ of a feature to compute the conditional distribution in Eq. 3.21. Since our vehicle's motion is mostly along the x-axis (forward), ambiguity in feature position is likely to appear along the z-axis [54]. Take the sensor configuration shown in Fig. 3.11a as an example. The spread due to elevation change has less impact on the projected $x - y$ plane than it does in z:

$$\frac{\Delta z}{\Delta xy} = \frac{r\sin(20^\circ + 6^\circ) - r\sin(20^\circ - 6^\circ)}{r\sin(20^\circ - 6^\circ) - r\cos(20^\circ + 6^\circ)} \approx 2.75. \tag{3.25}$$

Therefore, we optimize the trajectory and feature positions without introducing terrain constraints, and then terrain factors are constructed upon estimating the 2D feature locations X^-, followed by further optimization of the whole system.

3.4.5 Experiments in Simulation



(a) Top-down view of feature estimate

(b) GP terrain maps using feature points

Figure 3.9: Simulation results of terrain reconstruction. From left to right: algorithm without terrain factors, with terrain factors, and ground truth. The trajectory estimate and ground truth are represented by green and black lines, respectively. Features and maps are colored according to depth (red represents higher terrain elevation).

The simulation environment is designed to emulate our subsequent real experiment in a towing tank, and is aimed at evaluating our algorithms quantitatively. The vehicle follows a straight-line trajectory ($z = 1.0$ m) as shown by the black line in Fig. 3.9a from $y = 7.5$ m to $y = -5$ m over 60 seconds. A random terrain is generated with average depth at $z = 1.5$ m. We assume a limited number of features on the terrain are trackable, 200 of which are in the field of view. Vehicle poses are uniformly sampled at intervals of 0.2s, and at each pose, sonar measurements are simulated. The sonar has a field of view of $r = [0\text{ m}, 3\text{ m}]$, $\theta = [-35°, 35°]$ and $\phi = [-15°, 15°]$, and Gaussian noise is added to range and bearing measurements $\sigma_r = 0.0025$ m, $\sigma_\theta = 0.01$

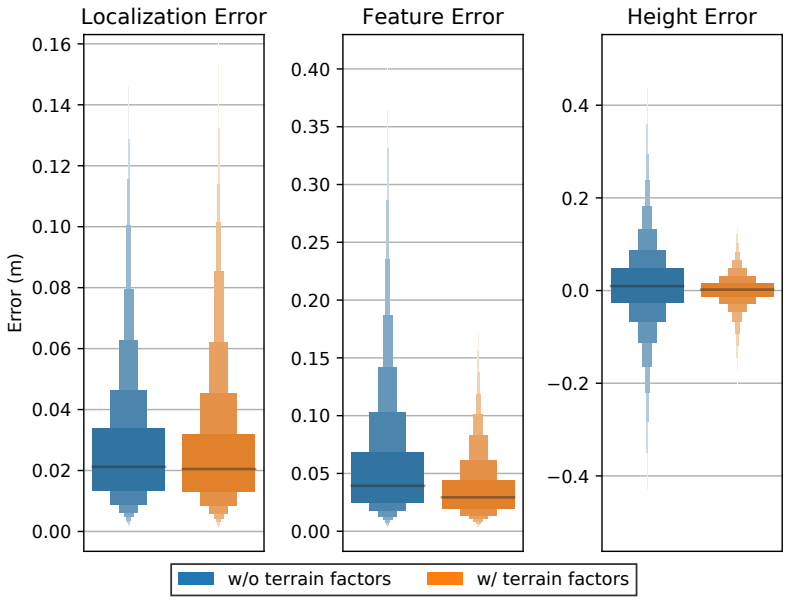
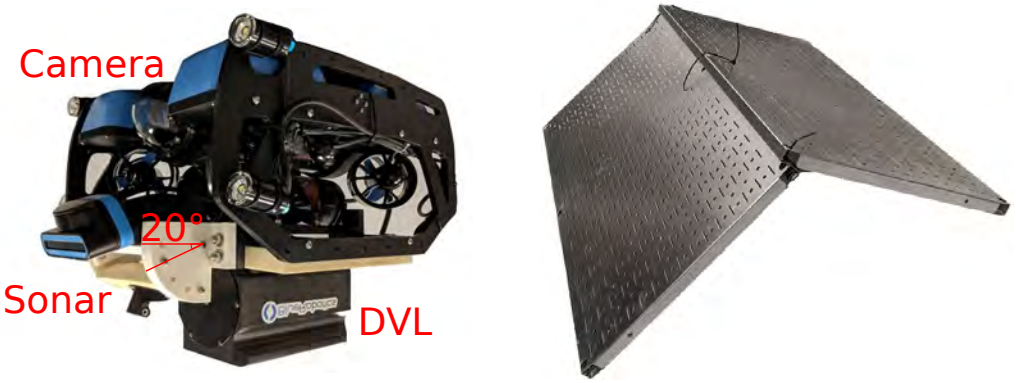


Figure 3.10: Results from the simulation of Fig. 3.9. Box plots with quantiles are used for visualizing error distributions with outliers.

rad. We also introduce randomness into feature tracking across consecutive frames. Consider two observations of the same feature at step i and $i+1$, i_p and $[i+1]_q$, each being assigned to landmarks j_p and j_q respectively. We assume that a feature can be successfully tracked, i.e., it is assigned to the same label in the current frame as it is in the previous frame, with probability 0.95, and thus $P(j_p = j_q) = 0.95$.

The experiments are repeated for 50 independent trials to evaluate performance, and one example trial is visualized in Fig. 3.9. We analyze three types of error of the SLAM result. First, localization error is computed as the Euclidean distance between estimated vehicle position and ground truth position. Secondly, we compare the mapping error as the distance between estimated feature position and ground truth position. Thirdly, we perform Gaussian process regression using estimated feature points $(X^-, \mathbf{m})$ as training data to produce a height map as shown in Fig. 3.9; the height error is composed of the difference between prediction and ground truth at every location.



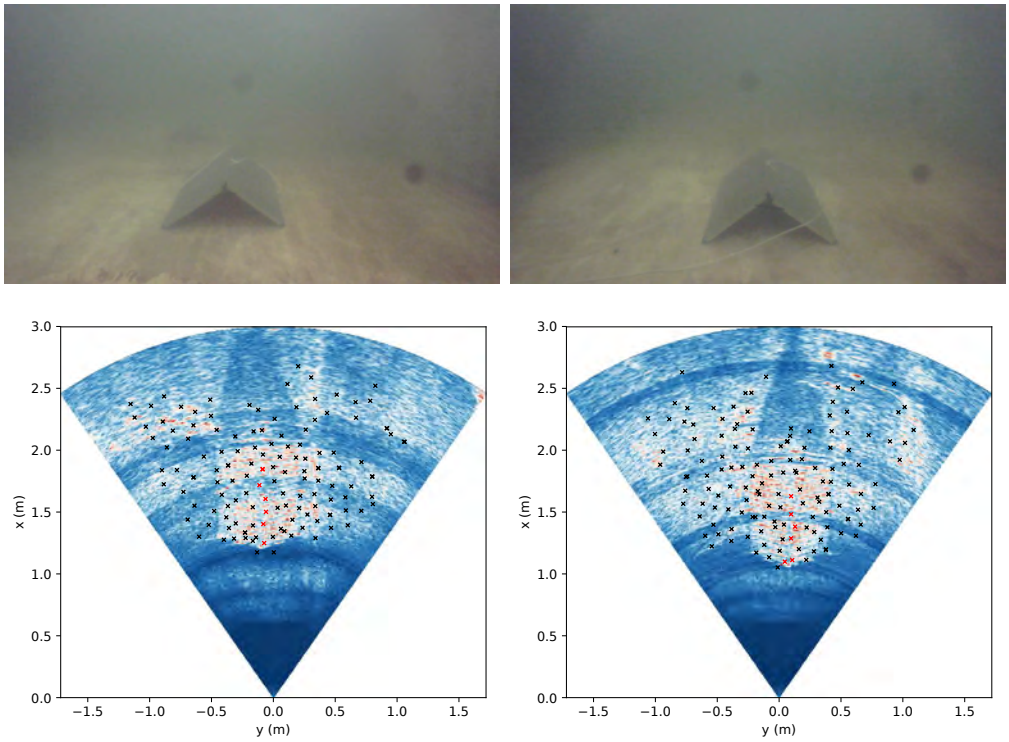
(a) BlueROV2 system overview (b) A pegboard ramp

Figure 3.11: Our experimental platform BlueROV is equipped with DVL and multi-beam imaging sonar. The sonar mounted at 20 degrees downward is used to observe objects on the floor, including a pegboard ramp.

All results are presented in Fig. 3.10 using box plots to provide more information on the error distribution. It is clear that all types of error exhibit longer tails without adding terrain factors. GP regression in post-processing aids the terrain height estimate when feature error is small enough, however, it is not helpful when dealing with abnormalities. Overall though, smoothness constraints significantly reduce elevation error when applied in the form of terrain factors during graph optimization.

3.4.6 Experiments on ROVs

Similar data was gathered using the BlueROV2, with additional sensors (see Fig. 3.11a). IMU data from a VectorNav VN-100, body velocity from an RTI SeaPilot DVL (600 MHz), and depth from a Bar30 pressure sensor on the BlueROV2 were used for constructing odometry factors in GTSAM. We used the Oculus M750d multi-beam imaging sonar, operated in 1.2MHz mode with a field of view of maximum range 3 meters, horizontal aperture 70° and vertical aperture 12°. The sonar image updates at 10Hz and has range resolution of 0.005 m and an angular resolution of 0.0024 rad.

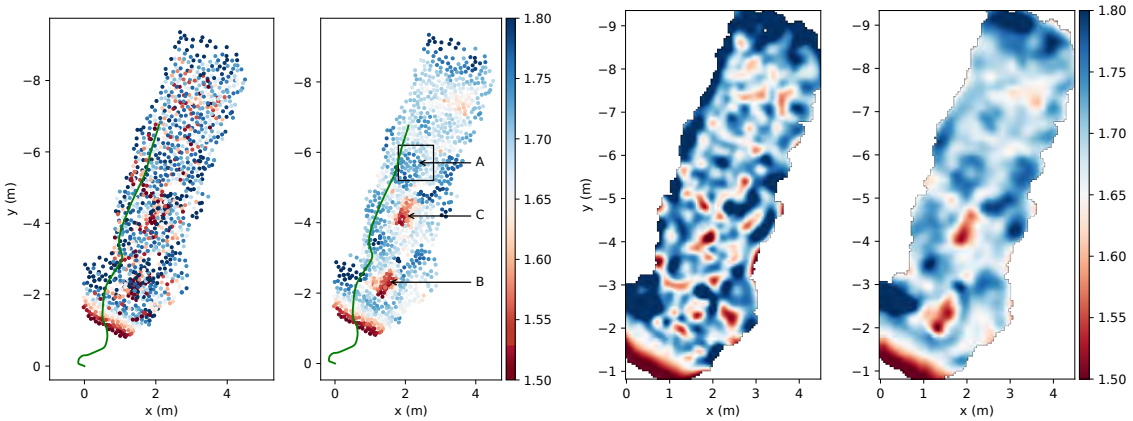


(a) Sample near location B

(b) Sample near location C

Figure 3.12: Two representative camera and sonar images near location B/C (Fig. 3.13a) where one of pegboards is in the field of view. In the sonar images (colored by intensity), A-KAZE features are marked with ×, and red features highlight the peak heights of the pegboards, which are used in Table 1.

The experiment was conducted in the Stevens towing tank. We placed two pegboard "ramps" at the bottom of the tank. Illustrative representations of the pegboards are shown in Fig. 3.11b. Each pegboard is 80 cm × 40 cm, and two pegboards form a ramp with height 20 cm. During the experiment, the vehicle is driven at a fixed depth of 1m and an approximate speed of 0.15m/s. The trajectory of the whole operation, which has a length of 8 meters over 1 minute, is shown in Fig. 3.13a. The two pegboards are marked as B and C respectively in Fig. 3.13a. The trajectory avoids regions directly above the pegboards to ensure successful DVL bottom tracking.



(a) Top-down view of feature estimate (b) GP terrain maps using feature points

Figure 3.13: Experimental results in water tank. From left to right: algorithm without terrain factors, with terrain factors. See Fig. 3.9 for notations. The locations of two pegboards are marked by B/C, and a sample region of the tank floor is marked A. Pegboards are visible in GP terrain maps, but outliers are pervasive around pegboards without adding terrain constraints.

From the feature point cloud and predicted height map (Fig. 3.13), two protruding regions (B and C) are visible using the proposed terrain factors, whereas treating features independently produces more erroneous estimates. However, terrain heights near the origin and the top boundary of the map gradually drift away from the ground truth. These regions are observed from a limited span of elevation angles, and as a consequence, those under-constrained features stay near the initial estimate with zero-elevation. Although terrain factors build pairwise connections between two feature nodes, they cannot prevent height from drifting due to degeneracy, provided that the observed terrain surface is smooth.

With regard to quantitative analysis, we only inspect the height of the ramp above the tank floor. Features that are likely to lie on the ramp peaks are manually selected in the sonar images as depicted in Fig. 3.12 using red $\times$ marks. The depth of the tank floor is estimated as the average height of features in region A. From

depth (m)	w/o terr. factors	w/ terr. factors
A	1.708 ± 0.084	1.711 ± 0.028
B	1.539 ± 0.059	1.542 ± 0.024
C	1.558 ± 0.057	1.549 ± 0.026

Table 3.2: Depth with 1 standard deviation of three regions marked in Fig. 3.13a without/with terrain factors. The height of the ramp peak at B/C is calculated by the difference compared to ground A.

the result in Table I, we can see that height estimates have lower variance when terrain factors are incorporated into the optimization, and the estimated height of the ramps is slightly closer to the actual geometry (0.2 m). However, both algorithms underestimate their true height, which may be caused by inaccurate calibration of factors including sensor displacement and speed of sound.

3.5 Conclusions

We present two applications involving mapping using sonar in underwater environments. We employ Gaussian processes to produce rich inference using sparse sonar measurements over unobserved gaps. But this application is limited to 2.5D environments, assuming a planar sonar sensor model. For the second application using a multi-beam imaging sonar for terrain mapping, we offer two improvements. First, data association is performed via optical flow tracking, which is more robust to noise and the absence of elevation angle. Second, a degenerate system is partly solved by adding terrain constraints connecting feature pairs, constraining their height values to be similar. The effectiveness of terrain factors is validated in both simulation and experiment. Outliers among the resulting feature estimates are reduced, and thus GP terrain maps are more accurate.

Looking ahead at areas for improvement, better initialization estimates can

potentially be provided using the linear triangulation proposed in [55]. In addition, the real-time viability of our SLAM framework is impeded by adding terrain factors, since the construction of a CLT requires full knowledge of a feature's 2D location. This also introduces a large number of additional constraints into the factor graph, making optimization computationally costly and challenging for real-time applications on embedded platforms. Although the offline SLAM solution requires about 7 seconds of computation on a laptop equipped with an Intel Core i7-4810MQ @ 2.80 Ghz $\times$ 8, the setup of our current experiment, without loop closures, is idealistic. Future work involves exploring its application at larger scales.

Chapter 4

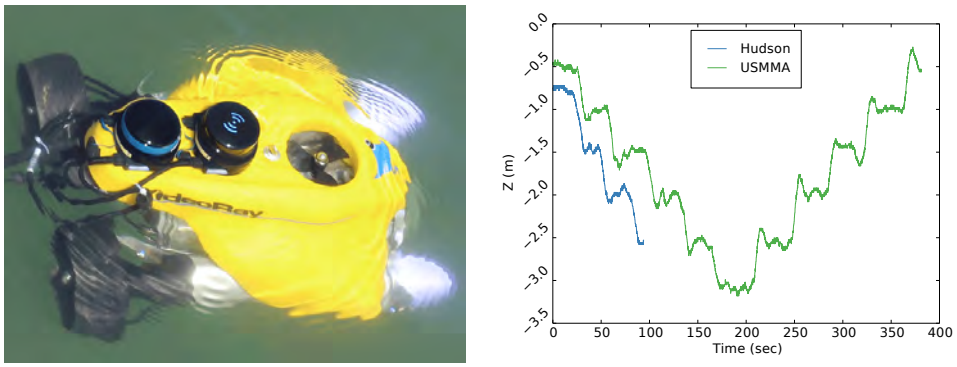
Underwater SLAM and Data Association

4.1 Submerged Structures Mapping with a Single-Beam Scanning Sonar

In the section, we discuss an approach to performing 3D occupancy mapping while treating sonar measurements in the same manner as a laser range-finder. Formally, we let $\phi = 0$, thus detected objects are on the zero-elevation imaging plane. We apply GPOctoMap presented above for mapping submerged structures using VideoRay ROV (Fig. 4.1a) equipped with a single-beam scanning sonar (Tritech Micron). An example of the scanning sonar return is illustrated in Fig. 4.2a, which is accumulated from 8.5 seconds of measurements from a corrugated seawall. We first briefly describe a processing pipeline that is applied to sonar images, to extract points that are likely to represent true range returns from submerged structures, and a hierarchical clustering method that subsequently divides the extracted points into features that are used in feature-based SLAM. GPOctoMap is applied to the filtered sonar data after the estimated trajectory is obtained.

4.1.1 Sonar Processing Pipeline

A sonar beam is composed of a sequence of intensity values, where a high intensity typically represents a return from subsea terrain or a submerged structure, however a number of factors may contribute to variations in signal intensity: changes in material properties, the strength of the transmitted signal, receiver sensitivity, and the distance to a target [4]. To address the limitations of applying a single threshold throughout a scan when faced with variable intensities, we propose a cluster-based



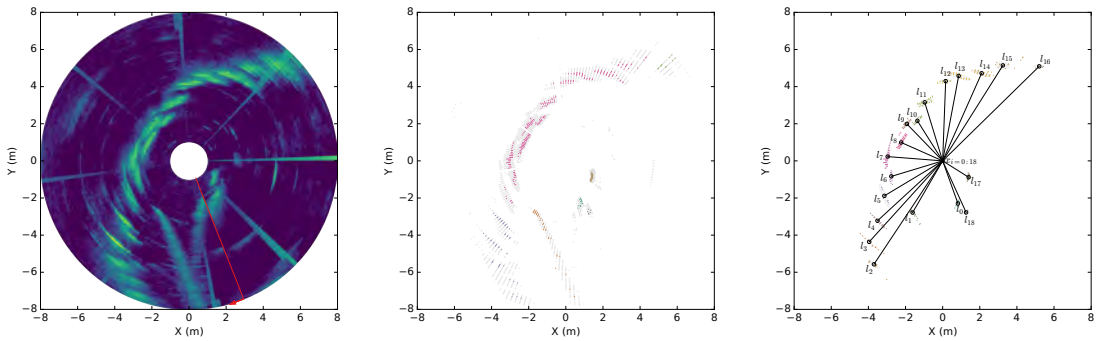
(a) (b)

Figure 4.1: The VideoRay was commanded to incrementally dive while collecting sonar scans. (a) VideoRay Pro4 ROV with a Tritech Micron scanning sonar. (b) Transitions in depth during the course of two experiments at different locations.

adaptive thresholding technique. First, a conservatively high amplitude threshold is applied to the entire image. The resulting points are subsequently clustered using the density-based spatial clustering of applications with noise (DBSCAN) algorithm [71], for which an appropriate number of clusters is designated automatically. Each cluster is then filtered individually using an amplitude threshold selected locally, chosen to be a designated number of standard deviations from the mean amplitude value within each cluster. DBSCAN also facilitates outlier detection, as clusters containing very few points can be eliminated from a filtered scan. A representative outcome of these steps is depicted in Fig. 4.2b. The initial filtering result is represented with light gray points, and the points remaining after adaptive thresholding on each cluster are colored based on the cluster label.

However, points from two objects may sometimes be recognized as one cluster, resulting in a substantial amount of connections, which is unfavorable for extracting representative features. We address this problem by using the hierarchical clustering methodology of ordering points to identfiy the clustering structure (OPTICS) [72]. OPTICS abandons the assumption that there exists a global parameter setting to

describe the cluster structure, and exploits the varying density across clusters as the criteria for ordering points to obtain a reachability plot. Automatic extraction of clusters [73] follows, and a representative example of the result is shown in Fig. 4.2c.



(a) (b) (c)

Figure 4.2: Representative results of the proposed sonar processing pipeline, applied to data collected in close proximity to a corrugated seawall with three angled sections. (a) Colored intensities of one 360° sonar scan (lighter color represents stronger intensity) with 200 beams that were collected within 8.5 seconds; the red line denotes the beginning of the full scan. (b) Adaptive thresholding of filtered points (light gray) locally within individual clusters, which are indicated using different colors. (c) Factor graph with variable nodes, extracted point landmarks and robot poses.

4.1.2 SLAM with Scanning Sonar

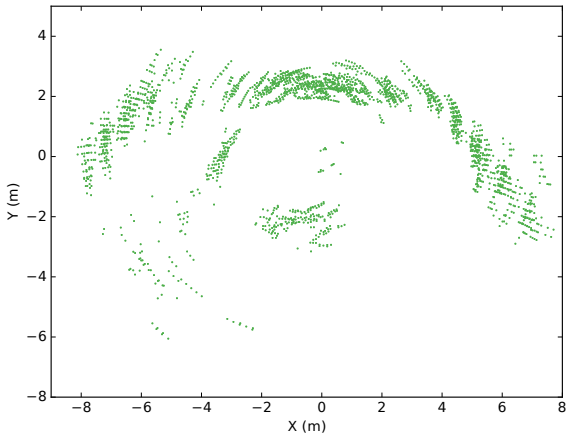


Figure 4.4: Misaligned points resulting from ICP registration among 10 full sonar scans.

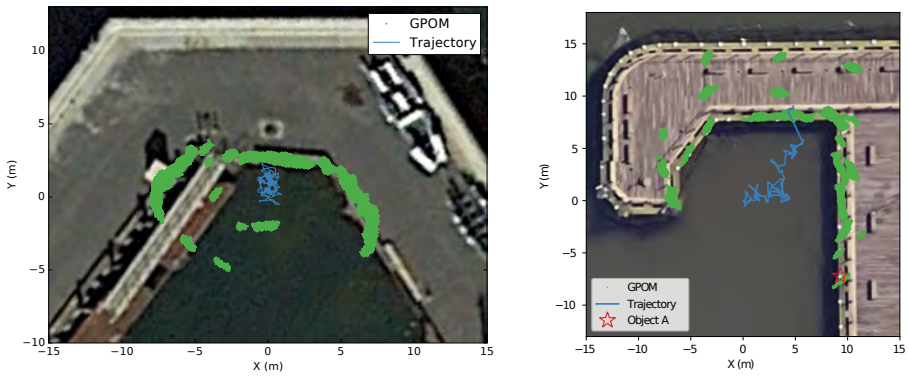


Figure 4.3: Two experimental results using the proposed methodology: United States Merchant Marine Academy, Hague Basin (*left*) and Hudson River Park, Pier 84 (*right*). Estimated trajectories and maps are overlaid with satellite images.

We next discuss the incremental SLAM approach that follows our extraction of features from sonar scans. Considering the drift of a robot under currents, registration methods which assume the robot holds a fixed position while collecting a 360° sonar scan will perform poorly, as shown in Fig. 4.4. First we discuss the application of incremental smoothing and mapping (iSAM2) [74] to provide an estimate of the robot's trajectory. Then, we discuss the front-end algorithms that address the problem of data association.

It is of course desirable to produce estimates of a robot's trajectory that are consistent with the observation of landmarks. *Loop-closure*, when a robot sees objects that it has seen before, may occur frequently as a sonar is scanning, and will introduce constraints into the state estimation process linking landmarks and robot poses. Thus we can determine the most likely map of observed landmarks by considering the many possible pose histories of the vehicle.

Mathematically we adopt the SLAM problem as a smoothing problem on a probabilistic graphical model, a *factor graph*. A factor graph is a bipartite graph $G = (\mathscr{F}, \Theta, \mathscr{E})$ comprised of two types of nodes, *factor nodes* and *variable nodes*.

Landmarks $\mathbf{l}_i \in \Theta$ and robot poses $\mathbf{x}_i \in \Theta$ are two kinds of variables nodes in the SLAM problem (see Fig. 4.2c). A factor node, $f_i \in \mathscr{F}$, which derives from a landmark measurement or from the process model

$$\mathbf{x}_i = g_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}, \ \ \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Lambda_i), \tag{4.1}$$

is a function of the variables in Θ. The relationships between factor nodes and variables are defined in edges $e_{ij} \in \mathscr{E}$. Following from the assumption that both the measurement model and process model are Gaussian, a factor node can be formulated from one observation as

$$f_k(\Theta_k = \{\mathbf{x}_k, \mathbf{l}_k\}) \propto \exp(-\frac{1}{2}||h_i(\mathbf{x}_k, \mathbf{l}_k) - \mathbf{z}_k||^2_{\Gamma_k}). \tag{4.2}$$

or from one control cycle as

$$f_i(\Theta_i = \{\mathbf{x}_i, \mathbf{x}_{i-1}\}) \propto \exp(-\frac{1}{2}||g_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i||^2_{\Lambda_i}), \tag{4.3}$$

Generally, a factor graph defines the factorization of a function $f(\Theta)$ as

$$f(\Theta) = \prod_i f_i(\Theta_i). \tag{4.4}$$

The objective is to find the optimal variables Θ^* that maximize Eq. 4.4, which can

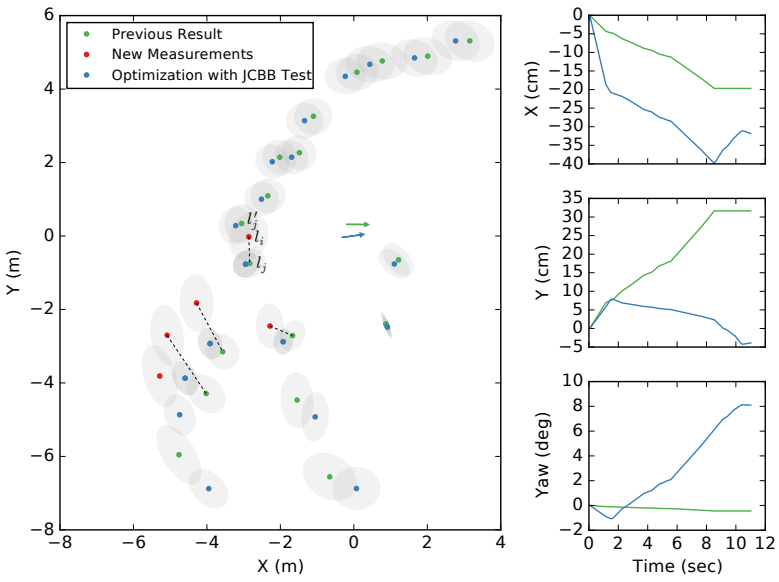


Figure 4.5: A result of iterative data association via JCBB. Four possible landmark pairings (dashed lines) are selected, then optimization on the factor graph is performed to obtain estimates of the vehicle trajectory and landmark locations. Final states of the vehicle are denoted by arrows.

be done through a nonlinear least-squares estimator

$$\begin{aligned}\Theta^* &= \underset{\Theta}{\operatorname{argmax}}\, f(\Theta) = \underset{\Theta}{\operatorname{argmin}} -\log(f(\Theta)) \\ &= \underset{\Theta}{\operatorname{argmin}} \sum_i ||g_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i||^2_{\Lambda_i} \\ &\quad + \sum_k ||h_k(\mathbf{x}_k, \mathbf{l}_k) - \mathbf{z}_k||^2_{\Gamma_k}\end{aligned} \tag{4.5}$$

We utilize the state-of-the-art SLAM algorithm iSAM2, which efficiently produces online estimates of landmark locations and a robot's trajectory as measurments are gathered.

4.1.3 Iterative Data Association

While the back-end of SLAM systems performs optimization on a factor graph, the front-end is responsible for constructing the factor graph, in which searching for landmark correspondences after revisiting previously mapped areas is a key component.

The Individual Compatibility Nearest Neighbor (ICNN) algorithm [75] solves the problem by matching each measurement with its gated nearest neighbor. Mathematically, a pair comprised of measurement $\mathbf{z}_i$ and landmark $\mathbf{l}_{j_i}$ is *compatible*, given the landmark's predicted measurement $\hat{\mathbf{z}}_{j_i}$, if the following condition is satisfied,

$$d^2_{ij_i} = \nu^T_{ij_i} S^{-1}_{ij_i} \nu_{ij_i} < \chi^2_{2,\alpha}, \tag{4.6}$$

where $\nu_{ij_i} = \mathbf{z}_i - \hat{\mathbf{z}}_{j_i}$ is the innovation, S_{ij_i} is its covariance, and α is the confidence level. However, the gated nearest neighbor approach ignores the fact that the innovations are correlated, and specifically that a set of pairings might not be *jointly* compatible.

To overcome this limitation, we adopt Joint Compatibility Branch and Bound (JCBB) [75] to verify a hypothesis of pairings. Similarly, the selection criterion is expressed as

$$D^2_{\mathcal{H}_i} = \mathbf{h}^T_{\mathcal{H}_i} S^{-1}_{\mathcal{H}_i} \mathbf{h}_{\mathcal{H}_i} < \chi^2_{d,\alpha}, \tag{4.7}$$

where $\mathcal{H}_i = \{j_1, j_2, ..., j_i\}$ represents the correspondences for each measurement in the current batch ($j_i = 0$ means an unseen landmark), $\mathbf{h}_{\mathcal{H}_i} = [v_{1j_1}, v_{2j_2}, ..., v_{ij_i}]^T$ is the joint innovation, and $d = \dim(\mathbf{h}_{\mathcal{H}_i})$. $D_{\mathcal{H}_i}$ is calculated for every hypothesis $\mathcal{H}_i$ that is established by traversing the interpretation tree in order to find the jointly compatible hypothesis with the most non-null pairings. The joint compatibility is monotonically non-decreasing, and as a consequence, the search of the tree is bounded.

Inspired by the iterative closest point (ICP) algorithm, we perform JCBB iteratively, each step of which is followed by optimization on a factor graph with constraints derived from the resulting data associations. These two interleaved steps are repeated until the process converges, and no new associations are found. This iterative process allows a larger number of jointly compatible correspondences to be retrieved, which decreases the probability that a spurious pairing exists for a given hypothesis [75]. Furthermore, a filtering procedure can be implemented to remove dynamic obstacles or spurious measurements that are rarely seen among pairings.

In most feature-based SLAM systems, the sensor is capable of observing a set of multiple landmarks simultaneously, in which case several landmarks may correspond to the same robot pose. However, with a scanning sonar and strong currents, a robot will sequentially observe landmarks such that each of them is associated with a different pose. We allow a designated number of these recently observed landmarks to remain "active" such that their associations can be changed. The outcome of one sequence of iterative data association is illustrated in Fig. 4.5. In this example, we wish to match our five most recently observed features with existing landmarks. According to ICNN, we will end up with the following pairings: the uppermost measurement l_i is paired with the landmark l'_j, and the other three remain the same. However, there is high risk that the first pairing is incorrect, because of the inconsistent rotation as a result of the first pairing and the others. In contrast, four pairings are established through JCBB which are simultaneously acceptable. The locations of landmarks and the trajectory of the vehicle are then corrected to maintain a consistent map.

4.1.4 Experiments on ROVs

Two experiments were carried out in cluttered shallow-water environments, using the VideoRay Pro4 ROV equipped with a Tritech Micron scanning sonar (Fig. 4.1a):

1) U.S. Merchant Marine Academy (USMMA) in King's Point, NY, 2) Pier 84 in Hudson River Park, NY. The ROV also has an accelerometer and compass onboard, however these measurements were not utilized due to a prohibitive amount of noise, especially the compass heading noise induced by interference from rotating thrusters. We commanded the ROV to incrementally dive and hold a fixed heading while collecting sonar data (Fig. 4.1b). In both experiments, we assumed the vehicle held a fixed translational position during its navigation. Despite the lack of inputs to the system, the vehicle exhibited a large amount of translational movement as a consequence of disturbances from the tether, currents, and wind-induced waves. Therefore, the objective is to recover the free movement of the ROV and to build a descriptive 3D map of its surrounding environments. The mapping results were overlaid with satellite images from Google earth to demonstrate the performance in Fig. 4.3.

While the centroid of extracted clusters after OPTICS is used as features in SLAM, points in clusters are treated as measurements from laser range-finder to perform mapping using GPOctoMap. The resulting point clouds used to populate 3D occupancy maps are shown on the left in Fig. 4.6. A standard occupancy grid map would remain quite sparse, containing many gaps that pose challenges for reasoning about motion planning and collision avoidance. However, the sonar-derived point cloud was used as training data for a GP regression in which the occupancy of the full map contents was predicted. As shown in Figs. 4.6, GP occupancy maps leveraged predictive inference to close gaps in many of the surrounding obstacles and produce continuous 3D maps that might serve as a tool for further decision-making about exploring these environments.

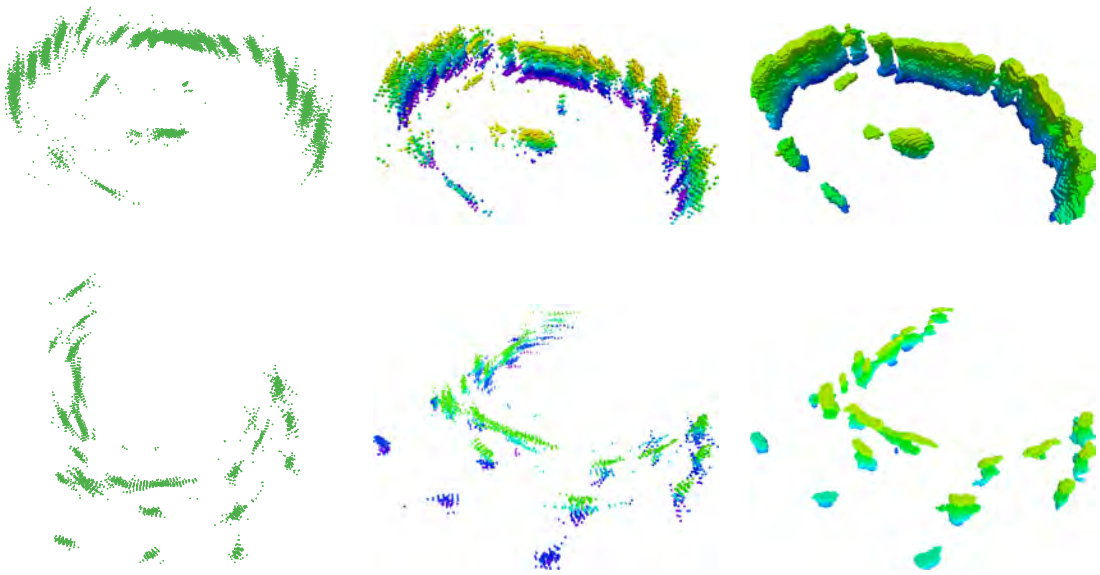


Figure 4.6: Comparison of map results from raw point clouds (*left*), OctoMap (*center*) and GP occupancy maps (*right*) using data collected at USMMA (*top*) and Pier 84 (*bottom*). The raw point clouds are visualized in the horizontal plane, from above, and the occupancy maps are shown from an isometric view.

4.2 The Data Association Problem

4.2.1 Problem Definition

We first give the definition of data association that will be assumed throughout this paper. A robot navigating throughout the environment repeatedly collects observations from a set of m features. Let $\{\mathbf{z}_1, ..., \mathbf{z}_m\}$ be the set of measurements of the landmarks $\{\mathbf{l}_1, ..., \mathbf{l}_m\}$ at state $\mathbf{x}$. The data association problem is to determine the hypothesis $H \triangleq \{j_1, ..., j_m\}$, each of which associates one measurement $\mathbf{z}_i$ with one non-repeated landmark $\mathbf{l}_{j_i}$. A measurement from a new landmark, or a null-pairing, is denoted as $j_i = 0$. Subsequently, the back-end of SLAM updates the system's estimate of the robot state and landmark locations, incorporating this set of new measurements.

The selection criterion is defined by the Mahalanobis distance between actual

measurements and predicted measurements given a noisy observation model. Mathematically, the joint measurement model under hypothesis H is

$$\mathbf{z}_H = \mathbf{h}_H(\mathbf{x}, \mathbf{l}_H) + \mathbf{v}, \tag{4.8}$$

where $\mathbf{h}_H \triangleq [\mathbf{h}_{1j_i}, ..., \mathbf{h}_{mj_m}]^T$ is the collection of independent measurement models with zero-mean Gaussian noise $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_H)$, and $\mathbf{l}_H \triangleq [\mathbf{l}_{1j_1}, ..., \mathbf{l}_{mj_m}]^T$ is the tested landmark vector corresponding to $\mathbf{z}_H = \{\mathbf{z}_1, ..., \mathbf{z}_m\}$. The distance d_H^2 is given by

$$\mathbf{e}_H = \mathbf{z}_H - \mathbf{h}_H(\hat{\mathbf{x}}, \hat{\mathbf{l}}_H) \tag{4.9}$$

$$\mathbf{C}_H = \mathbf{H}_H \hat{\mathbf{P}}_H \mathbf{H}_H^T + \mathbf{R}_H \tag{4.10}$$

$$d_H^2 = \mathbf{e}_H^T \mathbf{C}_H^{-1} \mathbf{e}_H, \tag{4.11}$$

where $\mathbf{H}_H$ is the Jacobian matrix with respect to robot pose and landmark positions, and $\hat{\mathbf{P}}_H$ is the joint covariance of the estimates. One set of pairings can be accepted, or is *jointly compatible*, if the predicted measurements lie in the validation gate based on the chi-squared distribution,

$$\text{jc}(H) \text{ if } d_H^2 \leq \chi_{d,\alpha}^2, \tag{4.12}$$

where $d \triangleq \dim(\mathbf{h}_H)$, and α is the confidence level.

4.2.2 Joint Compatibility Branch and Bound

In general, only one "optimal" solution is derived from data association algorithms, and the problem is challenging due to the exponential growth of the *interpretation tree* [76]. Therefore, obtaining the hypothesis with the minimum Mahalanobis distance is

computationally intractable.

Joint compatibility branch and bound (JCBB) [75] instead searches for the hypothesis which has the maximum number of non-null pairings $N(H) = \sum_{i=1}^{m} \mathbb{I}_{j_i \neq 0}$,

$$H^* = \underset{H}{\operatorname{argmax}}\, N(H), \text{ s.t. } \mathrm{jc}(H). \tag{4.13}$$

The underlying idea is that the probability of accepting a spurious pairing decreases as we increase the number of jointly compatible non-null pairings. With this redefinition, the combinatorial problem becomes tractable using a branch and bound algorithm. While generating the interpretation tree, we obtain an incomplete hypothesis $H_i \triangleq \{j_1, ..., j_{i<m}\}$ at every leaf. Given the best H^* so far, the branching of a leaf, or insertion of a new pairing $H_{i+1} = H_i \cup \{j_{i+1}\}$, is bounded by the lower bound $\underline{\mathrm{N}}(H^*)$. More specifically, H_{i+1} will be considered as a candidate only if $\overline{N}(H_{i+1}) > \underline{\mathrm{N}}(H^*) \wedge \mathrm{jc}(H_{i+1})$. The upper bound of the number of non-null pairings $\overline{N}$ can be estimated by assuming all future independently compatible pairings are also jointly compatible. Thus a good lower bound will significantly reduce the computation involved in the compatibility check.

4.2.3 Multiple Hypothesis JCBB

The myopic pairings derived from JCBB can be inaccurate. For example, when faced with the situation in the top left of Fig. 4.10, the solution that greedily incorporates more pairings with existing landmarks will distort the robot's state estimate. The assumptions of JCBB are typically capable of rejecting false positive pairings with a dense feature cloud, such as that from a camera image. However, these assumptions do not hold when dealing with fewer features. In addition, errors can result from the linearization of priors [77] and from the fact that maximum a posteriori estimation

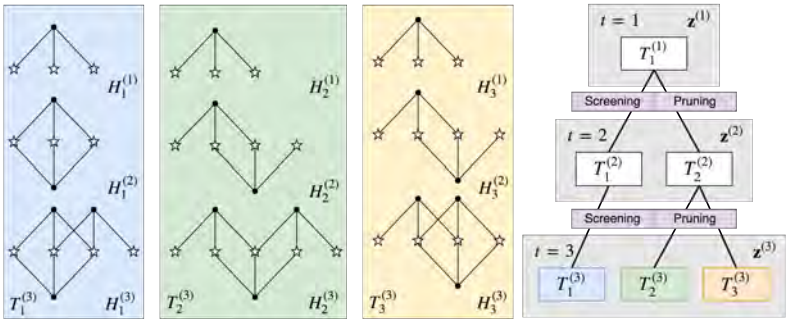


Figure 4.7: A simple example of MHJCBB. The left three panels visualize three tracks resulting from ambiguous data associations after taking three observations (from top to bottom). At each time instant, tracks diverge, taking into account possible hypotheses. The formation and ordering of tracks are discussed in Sec. III.B. Redundant hypotheses are screened and the resulting tracks are pruned to keep the K-best solutions (Sec. III.C).

converges to local minima.

In this section, we introduce our proposed approach to addressing the limitations of JCBB. Similar to multiple hypothesis tracking (MHT) [78], the key idea is to defer decision-making about data association and ultimately to pick the association "track" with the maximum number of non-null pairings. Consequently, the need for interpretation at a single ambiguous moment is avoided by accumulating measurements over a longer time horizon.

Let $T^{(t-1)}$ be the hypothesis track associated with measurements $Z^{(t-1)}$ over the trajectory $X^{(t-1)}$,

$$T^{(t-1)} \triangleq \{H_1, ..., H_{t-1}\},$$
$$Z^{(t-1)} \triangleq \{\mathbf{z}_{H^{(1)}}, ..., \mathbf{z}_{H^{(t-1)}}\},$$
$$X^{(t-1)} \triangleq \{\mathbf{x}_1, ..., \mathbf{x}_{t-1}\}.$$

Here, we use the superscript to indicate a time step and the subscript for indexing, e.g., $T_i^{(j)}$ means the ith track at time j. In the notation that follows, the superscript

will occasionally be omitted for the sake of brevity.

We seek to resolve ambiguity by maintaining the K-best solutions of the data association problem, which are represented as $\{T_1^{(t-1)}, ..., T_{k\leq K}^{(t-1)}\}$. Given the latest measurements, the goal is to correctly populate a compact set of the most probable association histories $\{T_k^{(t)}\}$ after pruning. However, without intentionally planning to eliminate ambiguity, there is no guarantee that all tracks will eventually be reduced to one.

4.2.4 Hypothesis Orderings

A good measure is needed to select the K-best solutions, and a common approach is to leverage the joint probability model [79], or the sum of residual error d_T^2 introduced by measurement and odometry constraints in its logarithmic form [80],

$$\begin{aligned} d_T^2 = &\sum_t ||\mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) - \mathbf{x}_t||_{\mathbf{Q}_t}^2 \\ &+ ||\mathbf{h}_{H_t}(\mathbf{x}_t, \mathbf{l}_{H_t}) - \mathbf{z}_{H_t}||_{\mathbf{R}_{\mathbf{H}_t}}^2, \end{aligned}$$

where $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}$ is the process model with zero-mean Gaussian noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$. Unlike d_H^2, the posterior residual error d_T^2 associated with a track of associations is computed after optimization.

However, a small d_T^2 doesn't necessarily indicate a good mapping result, because the optimization may easily become trapped in local minima [81]. For instance, consider two mapping results produced from the same trajectory (Fig. 4.8), using JCBB and using known data associations respectively. The former result, which is severely distorted, turns out to have a smaller posterior error. Additionally, a data association algorithm that assumes all measurements to be from new landmarks can

drive the d_T^2 error to zero.

Therefore, we propose two total orderings for intermediate hypothesis H and association track T. After new measurements are introduced into the least-squares problem, the first ordering is used for sorting association tracks. Inspired by the objective function used in JCBB, an association track is measured in terms of the total number of non-null pairings over the entire trajectory,

$$N(T) = \sum_t N(H_t). \tag{4.14}$$

Thus, the tracks are sorted in the following way,

$$\begin{aligned} T_i < T_j \;\Rightarrow\; & N(T_i) < N(T_j) \vee \\ & N(T_i) = N(T_j) \wedge d_{T_i}^2 > d_{T_j}^2. \end{aligned} \tag{4.15}$$

By maximizing the overall non-null pairings, populating new landmarks is discouraged. However, tracks that have false associations are likely to diverge from the correct trajectory, and so maintaining maximum non-null pairings is less probable. In the example in Fig. 1, there are three tracks with $N(T_1) = 0 + 3 + 2, N(T_2) = 0 + 2 + 2, N(T_3) = 0 + 2 + 3$.

Based on the above definition, we could follow the same search procedure as in JCBB, using the lower bound computed by $N(T^{*(t-1)}) + N(H^{*(t)})$, where $H^{*(t)}$ is generated from the track $T^{*(t-1)}$. However, if we consider a loop-closure observing existing landmarks, true tracks with even smaller $N(T)$ will produce more non-null pairings in the short-term. Thus, the second ordering is designed by valuing short-term hypotheses, solely incorporating $N(H)$.

The conventional JCBB method is used to search the interpretation forest con-

sisting of interpretation trees rooted at multiple tracks. In the example shown in Fig. 1 at time $t = 2$, two hypotheses $\{H_1^{(2)}, H_2^{(2)} = H_3^{(2)}\}$ from T_1 are jointly compatible, which consequently form two tracks $\{T_1^{(2)}, T_2^{(2)}\}$. Similarly, at $t = 3$, one $\{H_1^{(3)}\}$ from $T_1^{(2)}$ and two $\{H_2^{(3)}, H_3^{(3)}\}$ from $T_2^{(2)}$ split into three tracks $\{T_1^{(3)}, T_2^{(3)}, T_3^{(3)}\}$. A priority queue with maximum size K is maintained to house jointly compatible hypotheses from any track that could be one of the K-best. The priority is defined using the following total ordering:

$$\begin{aligned} H_i < H_j \;\Rightarrow\; & N(H_i) < N(H_j) \vee \\ & N(H_i) = N(H_j) \wedge d^2_{H_i} > d^2_{H_j}. \end{aligned} \tag{4.16}$$

In this way, the last entry in the queue serves as the lower bound while traversing the forest. A hypothesis will be inserted into the priority queue as long as it is superior to the lower bound either in the number of non-null pairings or in the chi-squared distance, both of which are non-decreasing with respect to node level. It is also worth noting that until now we do not distinguish hypotheses from different association tracks, thus the outcome is comprised of the i-th best H_{ki} from association track T_k.

4.2.5 Hypothesis Elimination

Hypotheses and association tracks grow exponentially (due to the fact that we must perform JCBB over K association tracks), so poor management of the various hypotheses will increase the computational burden, and more importantly, it will neglect correct associations. This happens in situations where hypotheses with more false positives have occupied the majority of spots in the priority queue, leaving no space for correct hypotheses with more true negatives. Thus, we propose a number of techniques to eliminate unlikely associations that fall into two categories: screening

and pruning (similar to concepts in [82]).

Screening refers to eliminating unnecessary branches before generation. Suppose a jointly compatible hypothesis is $H = \{j_1 = 1, j_2 = 2\}$. Then, there are three trivial hypotheses that are also jointly compatible: $\{j_1 = 1, j_2 = 0\}$, $\{j_1 = 0, j_2 = 2\}$, and $\{j_1 = 0, j_2 = 0\}$, which become redundant if they result in little to no ambiguity during the ensuing data association. Thus we formulate two screening rules by examining the posterior estimate as follows:

$$\text{delete } H_{ki} \text{ if } H_{ki} < H_{kj}$$
$$\wedge\ ||\hat{\mathbf{x}}_{t,H_{ki}} - \hat{\mathbf{x}}_{t,H_{kj}})||^2_{\mathbf{P}} < \alpha_1, \quad \text{(Rule 1)}$$

$$\text{delete } H_i \text{ if } N(H_i) = 0$$
$$\wedge\ (\exists k\ (N(H_{kj}) \geq 1 \wedge ||\hat{\mathbf{x}}_{t,H_i} - \hat{\mathbf{x}}_{t,H_{kj}}||^2_{\mathbf{P}} < \alpha_1)), \quad \text{(Rule 2)}$$

where $\hat{\mathbf{x}}_H$ is the updated pose estimate and $\mathbf{P}$ is the updated covariance estimate from any of the two hypotheses. It is worth noting that in practice, we use an extended Kalman filter (EKF) update step to predict the posterior pose and covariance estimates after incorporating a new hypothesis H, instead of performing the full least-squares optimization that is applied elsewhere. The first rule selectively discards any hypothesis that is close to a high-ranking hypothesis derived from the same association track T_k, and the second rule ignores the all-null hypothesis if the posterior state is in the proximity of any hypothesis with non-null pairings. Both rules restrain the addition of new landmarks.

After the generation of hypotheses, *pruning* is used for the elimination of redundant tracks, since all states are updated with new measurements. The rules are

as follows:

$$\text{delete } T_i \text{ if } d^2_{T_i} \text{ is an outlier} \vee d^2_{T_i} > \alpha_2, \tag{Rule 3}$$

$$\text{delete } T_i \text{ if } N(T_j) - N(T_i) > \alpha_3, \tag{Rule 4}$$

$$\text{delete } T_i \text{ if } T_i < T_j \tag{Rule 5}$$
$$\wedge \ (\forall n \leq t \ (||\mathbf{x}_{n,T_i} - \mathbf{x}_{n,T_j}||^2_{\mathbf{P}} < \alpha_4).$$

The third and fourth rules delete unlikely tracks with regard to the chi-squared error and the number of landmarks that are observed at least twice. However, the fourth pruning rule is not executed frequently, considering that the observation of existing landmarks doesn't occur until loop-closure (observing the same landmarks at consecutive poses usually will not eliminate ambiguity). Therefore, we consider tracks with many new landmarks to be less probable after a certain number of re-observations are performed. The fifth rule inspects the closeness of two estimated trajectories by calculating the maximum Mahalanobis distance between two poses at every step, and we consider the track of the highest order to represent the others in its close proximity.

4.2.6 Traversal Order

The branch and bound algorithm relies on the quick computation of lower bounds such that suboptimal branches are not expanded. Typically, it is solved by using a depth-first search (DFS) strategy to obtain one jointly compatible hypothesis at the leaf node. However, in the context of multiple hypothesis data association, the K-best solutions are generally distributed across different "track trees". Thus exhaustive traversal of a forest, tree by tree, is less efficient. In [83], a novel traversal strategy, mixed stacked depth-breadth first (MSDBF) search, was proposed to speed up the search for an optimal solution.

Algorithm 2: MHJCBB

Result: Return best estimates $\hat{X}, \hat{L}$

Given measurements Z, priors $T^{(1)}$;
for $\{\mathbf{z}_i^{(t)}\}$ **do**
  Priority queue $PQ \leftarrow \emptyset$ ordered by Eq. 4.16;
  for *MSDBF* $H_{ki}^{(t)}$ **do**
    if $H_{ki}^{(t)} > PQ.top()$ and jc(H) **then**
      Delete or insert $H_{ki}^{(t)}$ to PQ by rules 1-2;
    end
  end
  for $H_{ki}^{(t)} \in PQ$ **do**
    Add $\{\mathbf{z}_i^{(t)}\}, H_{ki}^{(t)}$ to $T_k^{(t)}$;
    Optimize states $\hat{X}^{(t)}_{T_k^{(t)}}$, $\hat{L}^{(t)}_{T_k^{(t)}}$;
    Delete $T_k^{(t)}$ by rules 3-5;
  end
end
Return $\hat{X}^{(t)}_{T_1^{(t)}}, \hat{L}^{(t)}_{T_1^{(t)}}$ with $T_1^{(t)}$ ordered by Eq. 4.15;

Firstly, a non-recursive traversal with stacks replaces function recursion. After visiting the first leaf, instead of visiting its siblings, mixed depth-breadth visits the first leaf traced back to the other children from the root by using a sequence of stacks (see [83] for implementation details). In principle, leaves in a tree or in a forest are explored in parallel. For example, two binary trees with two levels each will have leaves $\{l_1^1, l_2^1, l_3^1, l_4^1, l_1^2, l_2^2, l_3^2, l_4^2\}$, which will be explored in MSDBF in the order $(l_1^1, l_1^2, l_3^1, l_3^2, l_2^1, l_2^2, l_4^1, l_4^2)$. In this manner, the number of visited nodes is reduced, and if the computation time is limited, stopping the search early has a less detrimental effect on the outcome of MSDBF, as shown in the experiment to follow.

We describe the MHJCBB data association procedure as a whole in Algorithm 1. The process proceeds whenever new measurements arrive by populating association hypotheses (the first inner loop) and appending them to current tracks (the second

Max Nodes	Noise:	1	2	3	4	5
30	DFS	-0.03	**+0.46**	+0.39	+0.40	+0.59
	MSDBF	**-0.04**	+0.57	**+0.16**	**+0.12**	**+0.39**
50	DFS	-0.02	**+0.14**	+0.20	+0.19	+0.43
	MSDBF	**-0.04**	+0.22	**+0.18**	**+0.03**	**+0.24**
70	DFS	+0.03	**-0.03**	**+0.06**	+0.19	+0.33
	MSDBF	**-0.03**	+0.05	+0.15	**+0.12**	**+0.24**

Table 4.1: The increase/decrease in landmark error when limiting the max. number of visited nodes is compared over different node limits using MHJCBB($K = 10$) (baselines without restricting visited nodes are set to zero). Mixed depth-breadth first search generally obtains optimal hypotheses faster than depth first search.

inner loop). The MSDBF search is used to iterate over leaves in the interpretation tree rooted at its corresponding track, and the search is bounded by the least optimal case in PQ using hypothesis ordering 4.16 (or it ignores the bound if PQ hasn't reach its capacity). Given a jointly compatible hypothesis, Rules 1-2 are used to determine whether it will result in uncertain state estimation. Hypothesis generation is followed by optimization of the entire trajectory and map using a smoothing and mapping method, and unnecessary tracks are removed following Rules 3-5. The algorithm outputs an estimate selected from the remaining tracks using "track ordering" per Equation 4.15.

4.2.7 SLAM Over a Predefined Trajectory

Our simulation of a predefined trajectory employs an environment with 60 uniformly distributed random point features (Fig. 4.8). The robot is equipped with a sensor with a limited field of view (5m, 120°) that is capable of measuring the relative range and bearing to a landmark. Zero mean Gaussian noise is added to both process and measurement models. The robot is commanded to travel in four square patterns ($15m \times 15m$). To demonstrate association error, we gradually multiplied the standard

deviation (0.05° for rotation and bearing measurement, $0.05m$ for translation and range measurement) by factors of $1, 2, 3, 4, 5$ (this factor is the "noise level" plotted along the x-axes of Fig. 2). All methods included 50 trials with different random seeds.

The outcomes across one representative trial are illustrated in Fig. 4.8. JCBB generates a misleading map as a result of a few incorrect data associations, whereas by using MHJCBB, tracks diverge when ambiguity occurs and converge to one track as soon as the robot observes enough existing landmarks. The average performance is shown in Fig. 4.9 by analyzing root-mean-square error with respect to true landmark positions. Given a mapping from ground truth to landmark estimates ($\mathbf{l}_i \to \mathbf{z}_j \to \hat{\mathbf{l}}_k$), we define $\hat{L}(\mathbf{l}_i) = \{\hat{\mathbf{l}}_k | \mathbf{l}_i \to \cdot \to \hat{\mathbf{l}}_k\}$ to be the set of estimates corresponding to the same landmark. Then we calculate the error as follows,

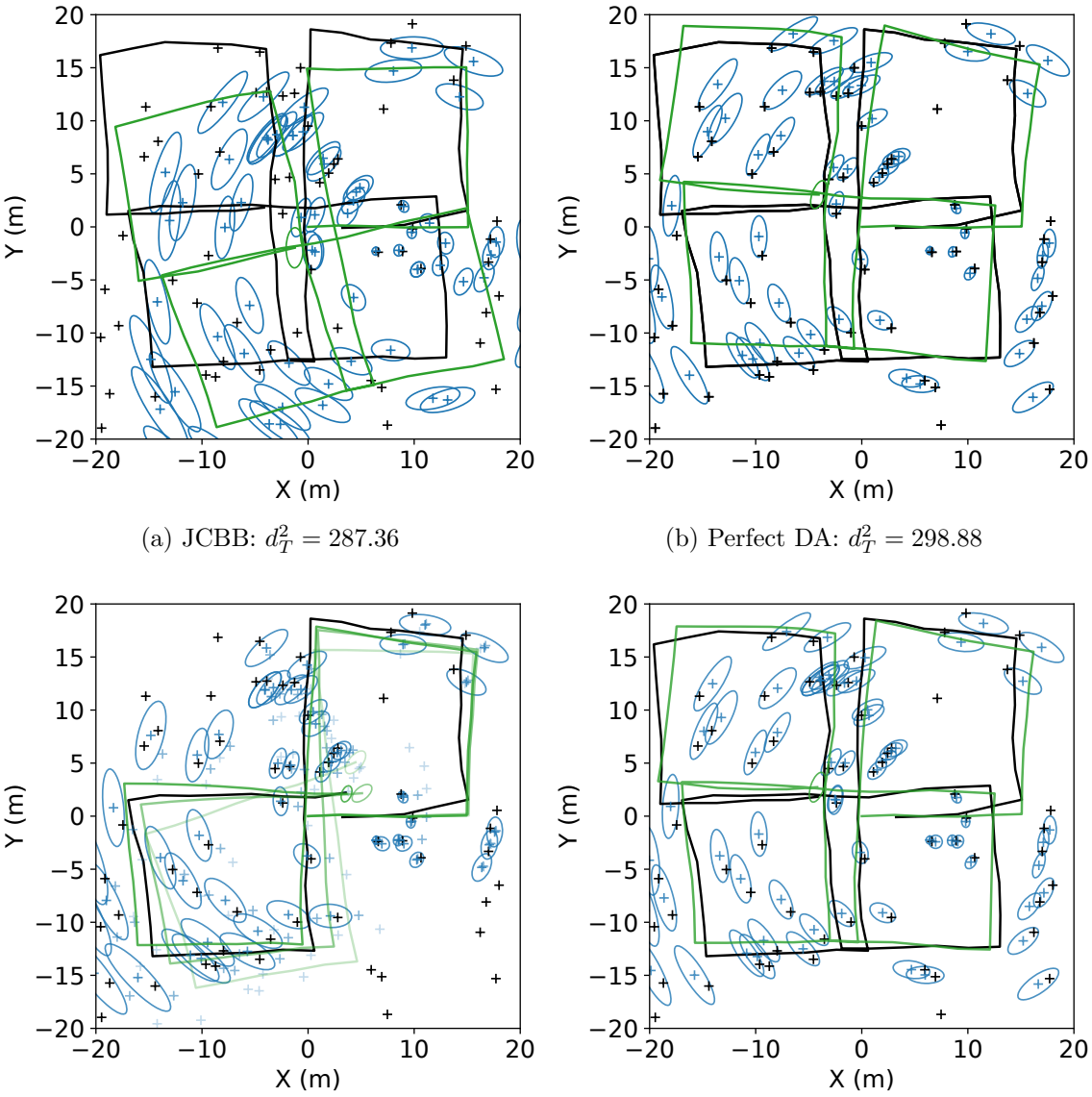
$$\mathrm{RMSE}(\hat{L}) = \sqrt{\frac{1}{|L|} \sum_{\mathbf{l}_i \in L} \mathrm{MSE}(\mathbf{l}_i)} \tag{4.17}$$

$$\mathrm{MSE}(\mathbf{l}_i) = \frac{1}{|\hat{L}(\mathbf{l}_i)|} \sum_{\hat{\mathbf{l}}_k \in \hat{L}(\mathbf{l}_i)} ||\mathbf{l}_i - \hat{\mathbf{l}}_k||_2^2. \tag{4.18}$$

It is clear that landmark error rises dramatically as we increase the noise level, and our proposed method recovers accurate landmark positions under high uncertainty. However, although allowing more estimation tracks improves performance, especially in extremely uncertain environments, the performance drop compared with SLAM under perfect knowledge of data association is still substantial. This can be explained by the fact that the error introduced by an incorrect pairing is exacerbated by increased noise, thus more hypotheses even with false positive pairings are considered jointly compatible. Consequently, even if we increase the quantity of hypotheses, the subtle differences in the posterior error makes the hypothesis pruning

process error-prone.

In addition, we show the number of visited nodes in the interpretation forest during the entire trajectory, along with all independently and jointly compatible nodes, in Fig. 4.9(b). As mentioned above, the numbers of visited nodes along with jointly compatible nodes undergo an approximately linear increase as the noise level increases. However, the required computation is still demanding in contrast to single hypothesis JCBB, which visits fewer than 250 nodes (this JCBB visit count is not shown in Fig. 2 due to the scaling of the plot). Mixed depth-breadth first search aids MHJCBB by reducing the number of joint compatibility checks required. Further investigation shows that when limiting the maximum number of nodes the search is allowed to visit, using MSDBF search to spread out visited leaves among multiple interpretation trees yields less accuracy loss, as shown in Table 1.



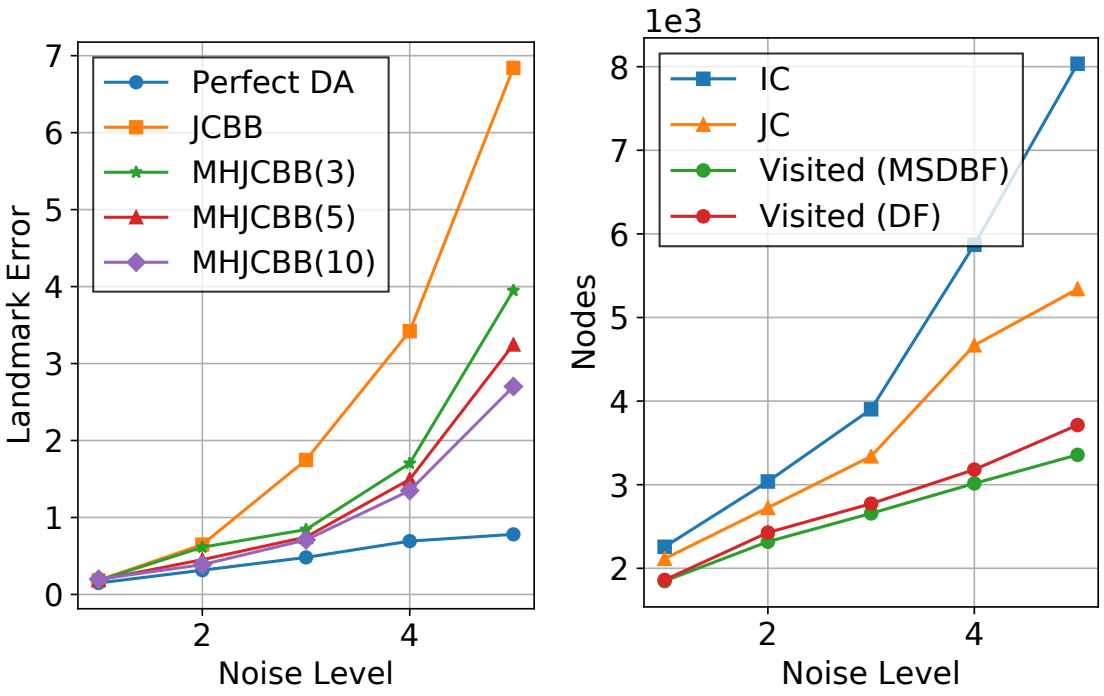
(a) JCBB: $d_T^2 = 287.36$

(b) Perfect DA: $d_T^2 = 298.88$

(c) MHJCBB($K = 5$): Step 43

(d) MHJCBB($K = 5$): $d_T^2 = 291.04$

Figure 4.8: Mapping examples in a simulated environment (noise level = 4) using (a) JCBB, (b) known data associations, and (c-d) MHJCBB($K = 5$). JCBB distorts the trajectory but iSAM2 reports a relatively small error. Tracks using MHJCBB diverge when the robot revisits its start location, but they converge to the optimal trajectory after collecting enough observations. **Black**: ground truth, **green**: multiple estimated trajectories, **blue**: multiple estimated landmarks and error ellipses (2 std. deviations). Darker-colored trajectories and landmarks in (c) indicate the favored hypotheses.



(a) (b)

Figure 4.9: Mean landmark error and interpretation forest node counts (using MHJCBB(10)) are plotted with respect to the noise level over 50 trials of the example shown in Fig. 4.8 with randomized landmark locations. MHJCBB effectively manages errors at low noise levels, and MSDBF reduces the number of forest nodes visited (IC: total independently compatible node count, JC: total jointly compatible node count, Visited: nodes actually visited during the search).

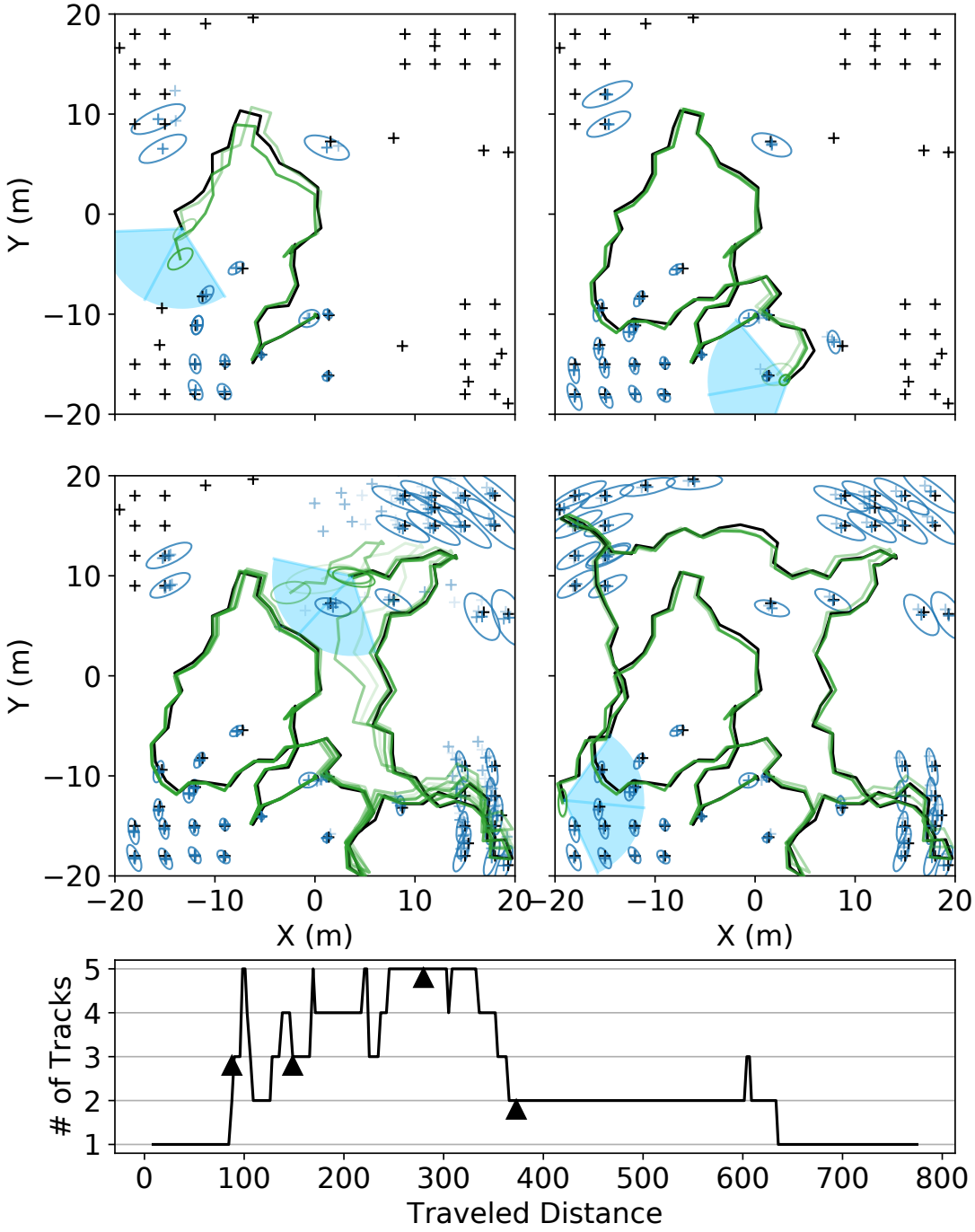


Figure 4.10: Four representative steps from EM exploration using our proposed MHJCBB($K = 5$) algorithm. The estimated trajectories (*lines*) and maps (*crosses*) correspond to the four instants along the robot's trajectory that are marked in the plot at bottom, which shows the evolving number of tracks.

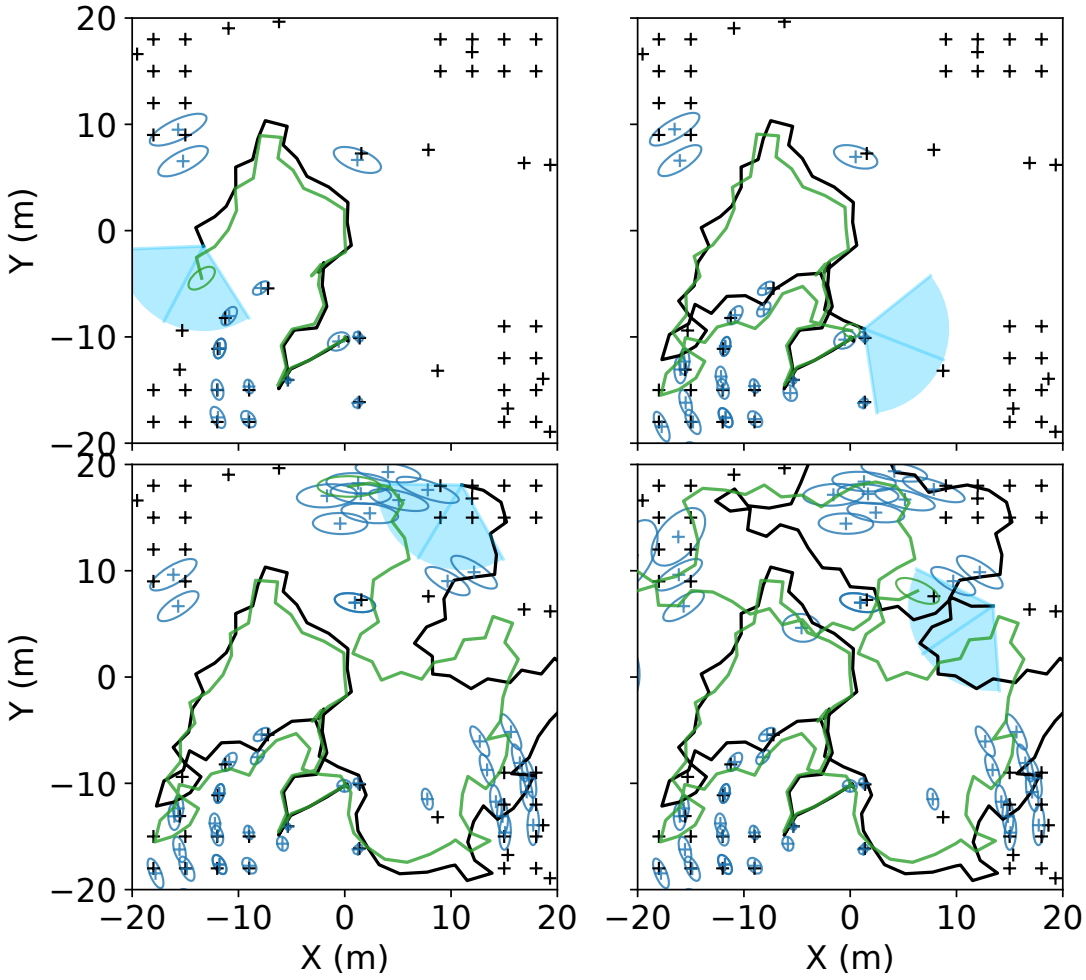
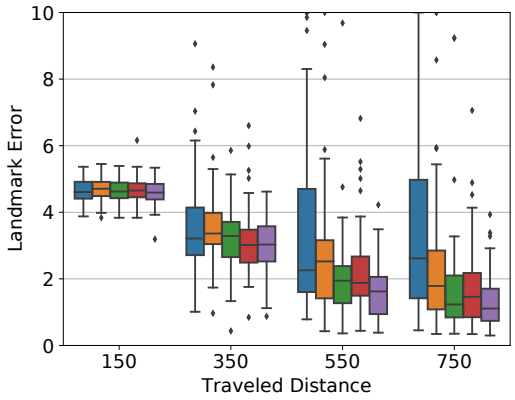
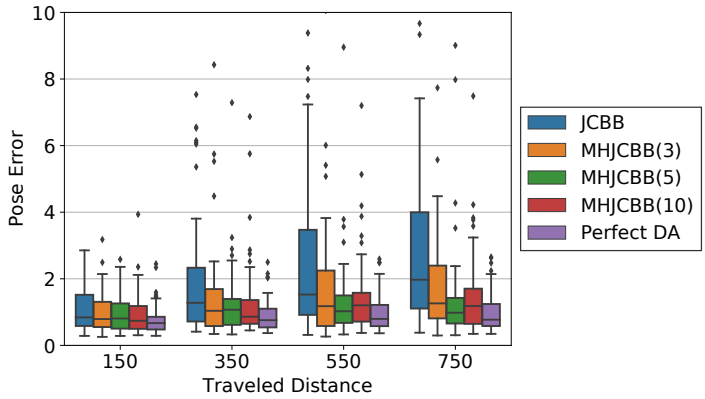


Figure 4.11: Snapshots at the same travel distances shown in Fig. 4.10 are now shown using a single-hypothesis JCBB algorithm to support EM exploration. In the first panel, an erroneous association with an existing landmark corrupts state estimation, and hinders the subsequent exploration process.



(a)



(b)

Figure 4.12: Results showing landmark errors and robot pose errors with respect to distance traveled over 50 trials of EM exploration. Errors accumulate as the robot explores using JCBB, and in contrast, MHJCBB with different maximum numbers of hypotheses ($K = 3, 5, 10$) produces comparable results to the trials with perfect knowledge of landmark associations.

Chapter 5

Autonomous Exploration in Virtual Maps on UGVs

We address the problem of autonomous exploration for a range-sensing mobile robot in an initially unknown environment. Our robot performs SLAM and constructs an occupancy grid map as it explores. We assume a bounded 2D space $V \subset \mathbb{R}^2$ where all discretized cells $\mathbf{m}_i$ are initialized as *unknown* $P(m_i = 1) = 0.5$. A *frontier* is defined as the boundary where free space meets unmapped space. The exploration is considered complete if no frontier can be detected. However, highly uncertain poses are likely to result in complete, yet inaccurate occupancy grid maps, limiting the usefulness of information gained by exploring unknown space. Assuming the environment contains individual landmarks $L = \{\mathbf{l}_k\}$, apart from discovering more landmarks, minimizing the estimation error is equally crucial.

5.1 EM Exploration

Let $\mathcal{L} = \{\mathbf{l}_i\}$ be the set of landmarks in the environment. We have an estimate for each of them $\mathbf{l}_i \in \mathbb{R}^2$, which is distributed as $\mathcal{N}(\hat{\mathbf{l}}_i, \Sigma_{\mathbf{l}_i})$. Naturally, an exploration strategy considering mapping uncertainty has the definition of the utility function as follows,

$$U = \sum_{\mathbf{l}_i \in \mathcal{L}} \phi(\Sigma_{\mathbf{l}_i}), \tag{5.1}$$

where $\phi : \Sigma \to \mathbb{R}$ represents the uncertainty criterion for covariance matrices. In addition to the uncertainty of landmarks, it is beneficial to add a cost-to-go $C(\mathbf{a})$ to favor a shorter path [84]. Thus the optimal action sequence is the one minimizing the

uncertainty with a weighted smaller cost:

$$\mathbf{a}^* = \underset{\mathbf{a}}{\operatorname{argmin}}\ U + \alpha C. \tag{5.2}$$

Since there are landmarks that haven't been observed yet, in most works, the common strategy is to omit the values of unknown landmarks and replace them with the entropy of occupancy grid maps. In this paper, we introduce the concept of *virtual landmarks* and treat the landmarks as latent variables. The objective is then to minimize the uncertainty of possible landmarks that would be observed when following the planned path. This uncertainty is defined

$$U \approx \sum_{\mathbf{v}_k \in \mathcal{V}} \phi(\Sigma_{\mathbf{v}_k}), \tag{5.3}$$

where $\mathcal{V} = \{\mathbf{v}_k\}$ are *virtual landmarks*, which represent all the possible locations of actual landmarks. They won't be incorporated into a robot's SLAM optimization unless they are revealed to be true landmarks, but the potential to reduce their uncertainty is considered throughout the course of autonomous exploration. Intuitively, the approximation computes the expected uncertainty of actual landmarks after taking into account the observations collected while following a candidate path. The function provides a trade-off between exploration and localization internally: exploration will decrease the uncertainty of virtual landmarks, which are initialized to have large covariance matrices, but poor-quality localization will lead to higher cost, since the covariance matrices of landmarks are estimated based on the robot poses that can observe them.

In the formulation of the SLAM problem as a *belief net* [80], the solution is

obtained by maximizing the joint probability distribution,

$$X^*, L^* = \underset{X,L}{\operatorname{argmax}} \ \log P(X, L, Z), \tag{5.4}$$

where X, L, Z are robot poses, landmarks, and measurements respectively. During exploration, we are confronted with unknown landmarks that haven't been observed yet. Therefore, we introduce the concept of *virtual landmarks* V as latent variables, which describe potential landmark positions that would be observed when following the planned path. Then the objective is to maximize the following marginal model,

$$\begin{aligned} X^* &= \underset{X}{\operatorname{argmax}} \ \log P(X, Z) \\ &= \underset{X}{\operatorname{argmax}} \ \log \sum_{V} P(X, Z, V). \end{aligned} \tag{5.5}$$

The above equation involves unobserved variables, which can be approached intuitively using an expectation-maximization (EM) algorithm as follows,

$$\text{E-step:} \quad q(V) = p(V|X^{\text{old}}, Z) \tag{5.6}$$

$$\text{M-step:} \quad X^{\text{new}} = \arg\max_{X} \mathbb{E}_{q(V)}[\log P(X, V, Z)]. \tag{5.7}$$

In the E-step, latent virtual landmarks are computed based on the current estimate of the trajectory and the history of measurements. In the M-step, a new trajectory is selected such that the expected value of joint probability, given the virtual landmark distributions, is maximized. The iterative algorithm alternates between the E-step and M-step, but each iteration is accomplished by the execution of actions and the collection of measurements.

The equation above poses a challenge for efficient solution due to the exponen-

tial growth of potential virtual landmark configurations with respect to the number of virtual landmarks. Inspired by classification EM algorithms, an alternative solution would add a classification step (C-step) before the M-step to provide the maximum posterior probability estimate of the virtual landmark distributions,

$$\text{C-step:} \quad V^* = \underset{V}{\operatorname{argmax}} \; p(V|X^{\text{old}}, Z) \tag{5.8}$$

$$\text{M-step:} \quad X^{\text{new}} = \arg\max_{X} \; \log P(X, V^*, Z). \tag{5.9}$$

If we further assume measurements are assigned to maximize the likelihood

$$Z = \underset{Z}{\operatorname{argmax}}\, h(X, V),$$

then the joint distribution can be expressed as a multivariate Gaussian centered at the proposed poses and landmark positions, and the covariance can be approximated by the information matrix inverse,

$$P(X, V, Z) \sim \mathcal{N}\left(\begin{bmatrix} X \\ V \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XV} \\ \Sigma_{VX} & \Sigma_{VV} \end{bmatrix} \right). \tag{5.10}$$

The solution of Eq. 5.9 is equivalent to evaluating the log-determinant of the covariance matrix,

$$\underset{X}{\operatorname{argmax}} \; \log P(X, V^*, Z) = \underset{X}{\operatorname{argmin}} \; \log\det(\Sigma). \tag{5.11}$$

This implies that the performance metric for our proposed exploration is consistent with the D-optimality criterion in active SLAM [85], except that the subjects considered include unobserved landmarks.

Since we are more interested in the uncertainty of the virtual landmarks and

the most recent pose $\mathbf{x}_{T+N}$ at step T with planning horizon N, we can marginalize out irrelevant poses in Σ_{XX}, ending up with $\Sigma_{\mathbf{x}_{T+N}}$. Typically, there exist thousands of virtual landmarks, thus approximation of Σ_{VV} is critical for real-time applications. Combined with pose simplification, we can obtain that, for a positive definite covariance matrix,

$$\log\det(\Sigma) < \log\det(\Sigma_{\mathbf{x}_{T+N}}) + \sum_{k} \log\det(\Sigma_{\mathbf{v}_k}), \tag{5.12}$$

where Σ_{V_k} is the diagonal block involving the kth virtual landmark in Σ_{VV}. This approximation is reasonable considering an overestimate of information using the introduced virtual landmarks (see Sec. 5.3.1).

In the next two sections, we show an approximative estimate of $\Sigma_{\mathbf{v}_k}$. The computation is illustrated in Figure 5.1. At the beginning, existing pose estimate and covariance (top left) can be obtained from SLAM, and candidate actions (top right) are generated from path library which will be discussed later. First, given one candidate path and predicted measurements, the covaraince of poses in the entire trajecotyr is updated, i.e., $\Sigma_{\mathbf{x}_{1:T+N}}$ (bottom left). Second, provided the predicted pose uncertainty, we approximate the covariance of virtual landmarks independently without the full pose covariance matrix (bottom right).

5.1.1 Belief Propagation on Candidate Actions

The SLAM problem defined in Equation 2.6 can be approximated by performing linearization and solved by iteration through the linear form given by

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{2}\|A\boldsymbol{\delta} - \mathbf{b}\|^2, \tag{5.13}$$

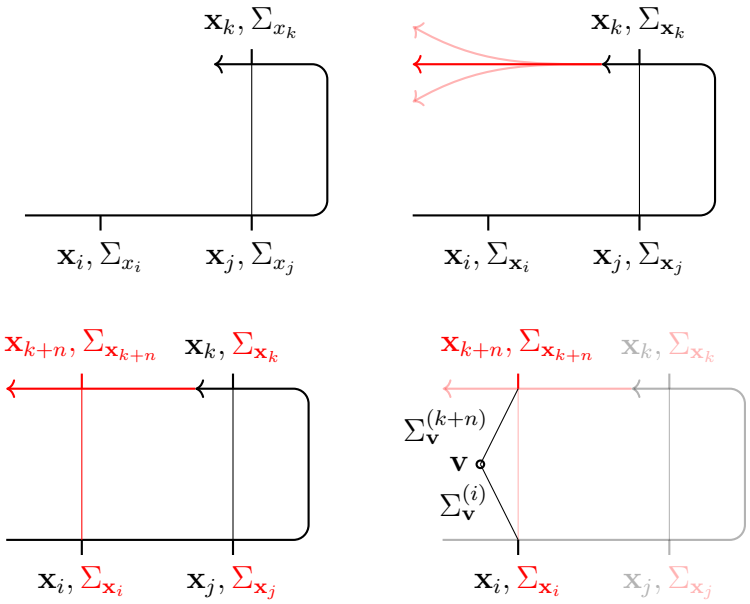


Figure 5.1: Belief propagation of candidate actions and virtual landmarks in EM-exploration.

where the matrix A represents Jacobian matrix and the vector b represents measurement residuals. The incremental update $\boldsymbol{\delta}$ in the above equation is obtained by solving the linear system,

$$A^\top A\boldsymbol{\delta} = A^\top \mathbf{b}. \tag{5.14}$$

It can also be formulated as $\Lambda\boldsymbol{\delta} = \boldsymbol{\eta}$ where $\Lambda = A^\top A$ is refered as information matrix. In general, the covariance matrix is obtained as the inverse of the information matrix

$$\Sigma = \Lambda^{-1} = (A^\top A)^{-1}. \tag{5.15}$$

As shown in [86], the recovery of block-diagonal entries corresponding to pose covariances can be implemented efficiently. As a result, we only concern the problem of updating uncertainty upon new measurements assumping $\Sigma_{\mathbf{x}_i}$ are given. Figure 5.1 shows a scenerio (from top right to lower left) where belief about current trajectory has

been obtained (black) and updates (red) are to be investigated if new measurements are available.

The covariance recovery can be performed in three steps. We use Figure 5.1 and Equation 5.19 as an example, where k is the most recent pose. First, we calculate disgonal entries in the covariance matrix (blue). Second, we ignore loop closure measurements and propogate covariance using only odometry measurements which could be from wheel odometry or sequential scan matching. The open-loop covariance recovery is given by

$$\Sigma_{\mathbf{x}_{i,k+t}} = \Sigma_{\mathbf{x}_{i,k+bauschbiott-1}} F_{k+t}^{\top}, \tag{5.16}$$

where $F_{k+t} = \frac{\partial \mathbf{f}_{k+t}}{\partial \mathbf{x}_{k+t}}$ is the Jacobian matrix of $\mathbf{f}_{k+t}$ with respect to pose $\mathbf{x}_{\mathbf{x}_{i,k-1}}$. The equation is applied recursively and the initial value $\Sigma_{\mathbf{x}_{i,k}}$ can be calculated by

$$\Lambda\Sigma_{\mathbf{x}_{\cdot,k}} = I_k, \quad R\Sigma_{\mathbf{x}_{\cdot,k}} = R^{\top} I_k \tag{5.17}$$

where $\Sigma_{\mathbf{x}_{\cdot,k}}$ represents the cross-covariance between poses and current pose (green column k) and I_k is a sparse block column matrix with an identity block only at the position corresponding to pose k. The solution of Equation5.17 is obtained by Cholesky decomposition on the right (the R is available immedidately after incremental update in iSAM2 [74]), whose computational complexity, for sparse R with N_{nz} nonzero elements, is $O(N_{nz})$.

Finally, we use Woodbury formula to update covariance matrix as shown in Figure 5.2 [87],

$$\Sigma' = \Sigma + \Delta\Sigma, \Delta\Sigma = -\Sigma A_u^{\top}(I + A_u \Sigma A_u^{\top})^{-1} A_u \Sigma, \tag{5.18}$$

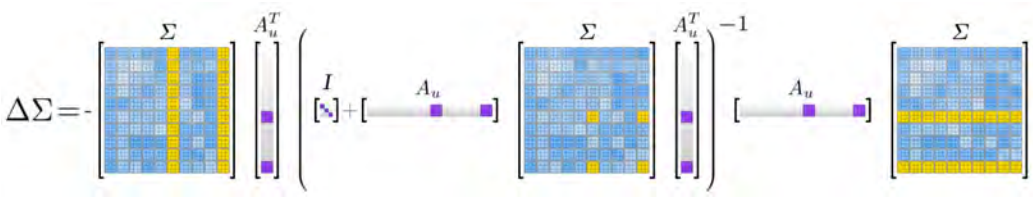


Figure 5.2: Efficient calculation of pose covariance on candidate actions.

where A_u is Jacobian matrix with each block row corresponding to one loop closure measurement. Using the above formula results in a much efficient update as it avoids to invert a large dense matrix $(A^\top A + A_u^\top A_u)^{-1}$. What we need involves recovering block column if the pose apprears in the measurements (green column i) and a matrix inversion of a relatively small matrix with the number of block row equal to the number of loop closure measurements.

$$\begin{bmatrix}
\Sigma_{\mathbf{x}_0} & & \Sigma_{\mathbf{x}_{0i}} & & & \Sigma_{\mathbf{x}_{0k}} & \cdots & \Sigma_{\mathbf{x}_{0,k+n}} \\
& \ddots & \vdots & & & \vdots & \cdots & \vdots \\
& & \Sigma_{\mathbf{x}_i} & & & \Sigma_{\mathbf{x}_{ik}} & \cdots & \Sigma_{\mathbf{x}_{i,k+n}} \\
& & & \ddots & & \vdots & \cdots & \vdots \\
& & & & \Sigma_{\mathbf{x}_j} & \Sigma_{\mathbf{x}_{jk}} & \cdots & \Sigma_{\mathbf{x}_{j,k+n}} \\
& & & & & \ddots & \cdots & \vdots \\
& & & & & \Sigma_{\mathbf{x}_k} & \cdots & \Sigma_{\mathbf{x}_{k,k+n}} \\
& & & & & & \ddots & \vdots \\
& & & & & & & \Sigma_{\mathbf{x}_{k+n}}
\end{bmatrix} \tag{5.19}$$

5.1.2 Belief Propagation on Virtual Landmarks

Assume the robot following a certain path is able to take measurements from landmarks in the surrounding environment. Let $\{\mathbf{x}_i \in \mathbf{SE}(2)\}$ be the robot poses that observe the same landmark $\mathbf{l} \in \mathbb{R}^2$. In the following derivation we don't distinguish

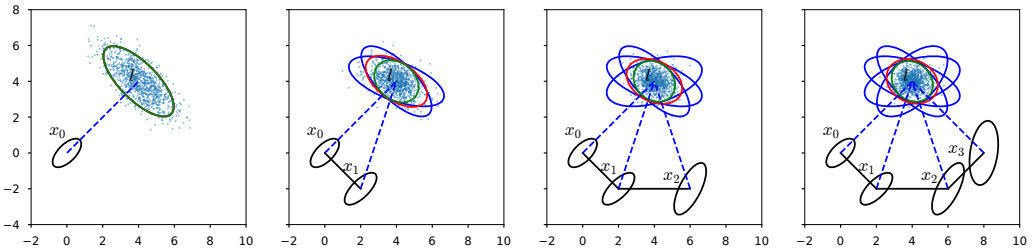


Figure 5.3: Covariance intersection is used to compute an upper bound (red ellipses) of the actual landmark covariance (green ellipses). Samples from the variable representing nearest neighbor are indicated as blue dots.

between virtual landmark $\mathbf{v}$ and actual landmark $\mathbf{l}$. The measurement $\mathbf{z}_i$ can be obtained from the following sensor model

$$\mathbf{z}_i = \tilde{\mathbf{z}}_i + \mathbf{v}_i = h_i(\mathbf{x}_i, \mathbf{l}) + \mathbf{v}_i, \;\; \mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i). \tag{5.20}$$

We further assume that the sensor model is invertible, i.e., we are able to predict landmark position given robot pose and measurement. Mathematically, the Jacobian matrix has full rank, $\text{rank}(\frac{\partial h_i}{\partial \mathbf{l}}) = 2$. One example of such models is bearing-range measurement which is commonly used in sonars and laser range-finders. In the following, we will use the inverse sensor model for convenience,

$$\mathbf{l} = h_i^{-1}(\mathbf{x}_i, \tilde{\mathbf{z}}_i) = h_i^{-1}(\mathbf{x}_i, \mathbf{z}_i - \mathbf{v}_i), \;\; \mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i). \tag{5.21}$$

The Jacobian matrices of the inverse sensor model are represented as $\mathbf{H} = \frac{\partial h^{-1}}{\partial \mathbf{x}}, \mathbf{G} = \frac{\partial h^{-1}}{\partial \tilde{\mathbf{z}}}$. Given the covariance matrices of poses $\{\mathbf{\Sigma}_i | \mathbf{\Sigma}_i \succ 0\}$, we intend to provide a consistent estimate of the landmark's covariance without the computation of $\{\mathbf{\Sigma}_{ij} | i \neq j\}$ as demonstrated in the bottom right in Figure 5.1.

Suppose we obtain two measurements of the same landmark at two distinct poses $\mathbf{x}_i, \mathbf{x}_j$, resulting in a joint distribution

$$\begin{bmatrix} \mathbf{l}_i \\ \mathbf{l}_j \end{bmatrix} = \begin{bmatrix} h^{-1}(\mathbf{x}_i, \mathbf{z}_i) \\ h^{-1}(\mathbf{x}_j, \mathbf{z}_j) \end{bmatrix}. \tag{5.22}$$

$$\begin{aligned} \mathrm{cov}\Big(\begin{bmatrix} \mathbf{l}_i \\ \mathbf{l}_j \end{bmatrix} \Big) &= \begin{bmatrix} \mathbf{\Sigma}_i^{\mathbf{l}} & \mathbf{\Sigma}_{ij}^{\mathbf{l}} \\ \mathbf{\Sigma}_{ij}^{\mathbf{l}\top} & \mathbf{\Sigma}_j^{\mathbf{l}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{H}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_j \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_i & \mathbf{\Sigma}_{ij} \\ \mathbf{\Sigma}_{ij}^{\top} & \mathbf{\Sigma}_j \end{bmatrix} \begin{bmatrix} \mathbf{H}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_j \end{bmatrix}^{\top} \\ &+ \begin{bmatrix} \mathbf{G}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_j \end{bmatrix} \begin{bmatrix} \mathbf{R}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_j \end{bmatrix} \begin{bmatrix} \mathbf{G}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_j \end{bmatrix}^{\top} \end{aligned}$$

Let $\mathbf{P}_{i1} = \mathbf{H}_i \mathbf{\Sigma}_i \mathbf{H}_i^{\top} \succ \mathbf{0}, \mathbf{P}_{i2} = \mathbf{G}_i \mathbf{R}_i \mathbf{G}_i^{\top} \succ \mathbf{0}, \mathbf{P}_{ij} = \mathbf{H}_i \mathbf{\Sigma}_{ij} \mathbf{H}_j^{\top}$.

$$\mathrm{cov}\Big(\begin{bmatrix} \mathbf{l}_i \\ \mathbf{l}_j \end{bmatrix} \Big) = \begin{bmatrix} \mathbf{\Sigma}_i^{\mathbf{l}} & \mathbf{\Sigma}_{ij}^{\mathbf{l}} \\ \mathbf{\Sigma}_{ij}^{\mathbf{l}\top} & \mathbf{\Sigma}_j^{\mathbf{l}} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{i1} + \mathbf{P}_{i2} & \mathbf{P}_{ij} \\ \mathbf{P}_{ij}^{\top} & \mathbf{P}_{j1} + \mathbf{P}_{j2} \end{bmatrix}$$

We obtain two covariance estimates independently from two poses and we will use split covariance intersection (SCI) [88] to calculate an upper bound of the actual landmark covariance as follows

$$\hat{\mathbf{\Sigma}}^{\mathbf{l}} = (\frac{1}{\omega}\mathbf{P}_{i1} + \mathbf{P}_{i2})^{-1} + (\frac{1}{1-\omega}\mathbf{P}_{j1} + \mathbf{P}_{j2})^{-1}, \omega \in [0, 1], \tag{5.23}$$

where ω can be optimized via

$$\omega^* = \underset{\omega \in (0,1)}{\mathrm{argmin}} \det(\hat{\mathbf{\Sigma}}^{\mathbf{l}}). \tag{5.24}$$

Readers are refered to [88] for proof of SCI. But the core idea is that if we construct an optimal linear, unbiased estimator $\hat{\mathbf{l}} = \mathbf{K}_i\mathbf{l}_i + \mathbf{K}_j\mathbf{l}_j(\mathbf{K}_i + \mathbf{K}_j = \mathbf{I})$ and it can be proved that $\hat{\mathbf{\Sigma}}^{\mathbf{l}} \succeq \mathbb{E}[(\hat{\mathbf{l}} - \mathbf{l})(\hat{\mathbf{l}} - \mathbf{l})^\top]$. Therefore we are able to further approximate Equation 5.12 by

$$\log\det(\Sigma) < \log\det(\Sigma_{\mathbf{x}_{T+N}}) + \sum_k \log\det(\hat{\Sigma}_{\mathbf{v}_k}). \tag{5.25}$$

We demonstrate the process of incremental split covariane intersection in Figure 5.3. At each step a covariance estimate calculated from the measurement at this specific step is indicated as dark blue ellipse. After the first step, we are able to fuse them using covariance intersection using Equation 5.23 as shown in red ellipses. It's evident that the resulted ellipsoid (red) contains that from SLAM (green).

From the above derivation an upper bound of a (virtual) landmark is calculated. However it's worth investigating its meaning in the context of a pose SLAM where no landmark is incorporated into optimization. To simplify the problem, we assume feature ponts are measured and constraints between poses are constructed through iterative closest point. Thus the projection of measurements is exactly the same as that in the landmark scenario and each projected feature point is distributed as a Gaussian centered as the actual location. We visualize its distribution using samples in the leftmost plot in Figure 5.3. In ICP, a point is associated to its nearest neighbor and the matched feature is given by

$$\tilde{\mathbf{l}} = \underset{\mathbf{l}_i}{\operatorname{argmin}} \|\mathbf{l}_i - \mathbf{l}\|_2. \tag{5.26}$$

Now it's trivial to show that

$$\mathbf{\Sigma}^{\mathbf{l}} \succeq \mathbb{E}[(\hat{\mathbf{l}} - \mathbf{l})(\hat{\mathbf{l}} - \mathbf{l})^\top] \succeq \mathbb{E}[(\tilde{\mathbf{l}} - \mathbf{l})(\tilde{\mathbf{l}} - \mathbf{l})^\top]. \tag{5.27}$$

The distribution of $\tilde{\mathbf{l}}$ is visualized at every step in Figure 5.3. While it's expected to minimize the (virtual) landmark covariance after taking an action in landmark-based SLAM, we essentially minimize the closeness of measurements in ICP-based pose SLAM. Because association error is one major cause of ICP error, tightly clustered target points greatly contribute to registration performance.

5.2 Segmant-aided SLAM

Although our EM exploration algorithm does not rely on any specific SLAM framework, a fundamental requirement is the need to predict the resulting uncertainty of future robot poses if a sequence of sensing actions is executed. Typically during exploration, loop closure constraints, such as the re-observation of a landmark, are desired with some regularity for better localization. However, not all real-world exploration problems can reason scalably about individual landmarks. Thus we propose to use segment-aided LiDAR mapping [89] to support decision-making in exploration, which optimizes a pose graph but is capable of detecting and handling re-observation as in landmark-based SLAM. In this section, we first present the segment-aided SLAM framework, then in the next sections, we elaborate the implementation of proposed EM exploration algorithm.

The diagram of our proposed SLAM system is shown in Fig. 5.4. The backbone of the pose graph is composed of two sequential factors. The *odometry* factor ($\mathbf{f}^O$) defines the relative motion constraint between two consecutive poses from persistent odometry measurements. Besides odometry, when the robot is equipped with a 3D LiDAR, *sequential scan matching* ($\mathbf{f}^{\text{SSM}}$) also provides a relative transformation by aligning point clouds observed at two positions. The essential component of graph SLAM to ensure accurate estimation is loop closure, which is incorporated in two

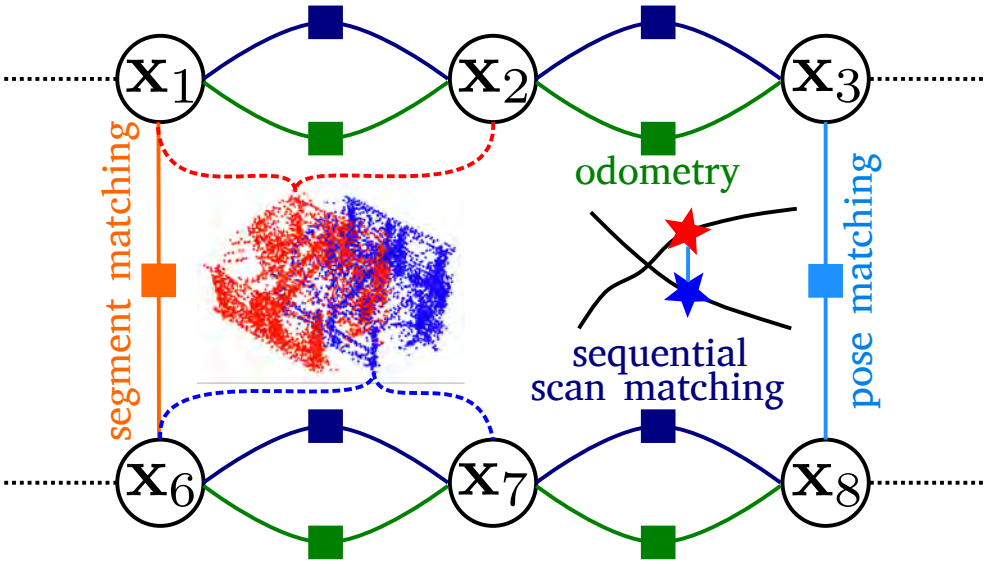


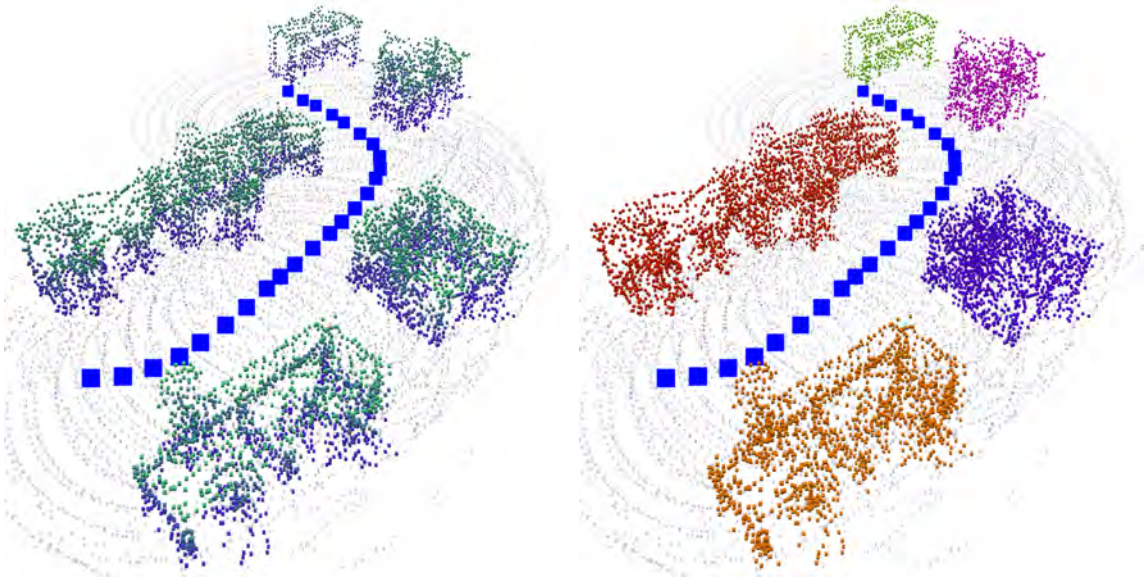
Figure 5.4: Overview of the segment-aided LiDAR mapping used for exploration. The factor graph includes sequential factors from odometry (green) and sequential scan matching (blue), and place recognition factors from segment matching (orange) and pose matching (cyan).

ways. First, when the current position of the robot is in the vicinity of a previously visited position, *pose matching* ($\mathbf{f}^{\mathrm{PM}}$) is performed by matching two point clouds accumulated around these two positions. Secondly, *segment matching* ($\mathbf{f}^{\mathrm{SM}}$) is utilized for loop-closure, and details on segmentation and segment association are elaborated below. Overall, the factor graph can be expressed as

$$\mathbf{f}(\boldsymbol{\Theta}) = \mathbf{f}^{0}(\boldsymbol{\Theta}_0) \prod_i \mathbf{f}_i^{\mathrm{O}}(\boldsymbol{\Theta}_i) \prod_j \mathbf{f}_j^{\mathrm{SSM}}(\boldsymbol{\Theta}_j) \qquad \text{(sequential)}$$

$$\prod_p \mathbf{f}_p^{\mathrm{PM}}(\boldsymbol{\Theta}_p) \prod_q \mathbf{f}_q^{\mathrm{SM}}(\boldsymbol{\Theta}_q), \qquad \text{(loop closures)}$$

where variables $\boldsymbol{\Theta}$ contain 6-DOF robot poses, and every factor $\mathbf{f}_i(\boldsymbol{\Theta}_i)$ defines a constraint model on a set of variables $\boldsymbol{\Theta}_i$. The optimization of a factor graph leads to a nonlinear least-squares problem, which can be solved efficiently using iSAM2



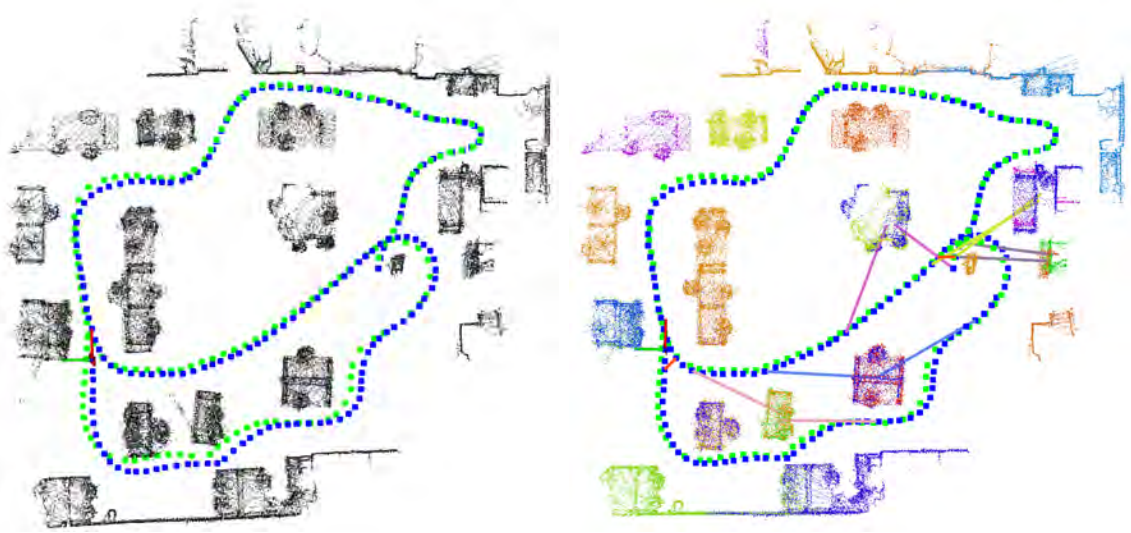
(a) Colored by height (b) Colored by segment

Figure 5.5: A segmented point cloud after ground removal.

[74]. Sequential scan matching is performed using the iterative closest point (ICP) algorithm.

ICP-based scan matching can easily get trapped in local minima, which results in erroneous sequential scan matching factors and loop-closure factors. To improve the robustness, we discard some scan matching results when the point cloud after ground removal has too few points to obtain a reliable transformation. Furthermore, an M-Estimator, specifically a Cauchy function, is applied to factors involving ICP in order to alleviate the consequences of erroneous constraints.

The segmentation and matching scheme follows *SegMatch* as proposed in [90], with some minor adjustments. We first remove the ground in a point cloud by fitting a plane using points appearing near ground level with the knowledge of sensor displacement and orientation. Euclidean cluster extraction is then performed on non-ground points on the i-th time-step to divide the points into clusters $\{C_1^i, C_2^i, ...\}$, and a voxel grid is constructed from each cluster denoted as $v(C)$. Two clusters detected



(a) No place recognition (b) Add SM and PM

Figure 5.6: Segment-aided LiDAR mapping. The ground truth trajectory (green) is obtained via LeGO-LOAM [91] using a full-range (100m) LiDAR point cloud, while our estimated trajectory (blue) relies on points with a 3m cutoff distance. The drift is corrected by adding *segment matching* (two lines with the same color connecting pose and segment), and *pose matching* (red lines connecting two poses).

in sequential frames are considered to be parts of the same segment when their corresponding voxel grids overlap by a certain number of voxels, or $v(C^i) \cap v(C^{i+1}) > v_{\text{thresh}}$. Let C_j^i be the cluster in the j-th segment observed at the i-th time-step, then a complete segment can be represented as $S_j = \{C_j^i, C_j^{i+1}, ..., C_j^{i+N_j}\}$, and all cluster points are anchored at the first step i given relative pose information via state estimation. Fig. 5.5 demonstrates a scene showing a segmented point cloud.

Feature descriptors (e.g., Ensemble of Shape Functions (ESF)) computed on a segment's point cloud provide one approach to determining if two segments are from the same object. Similarly, if we assume our state estimation has limited drift in an indoor environment, associated segments are likely to be spatially close to each other. Thus, S_j and S_k are matched by sequentially verifying the following distances (1) $\Delta c = |\mathbf{c}(S_j) - \mathbf{c}(S_k)|$ where $\mathbf{c}(S)$ is the segment centroid, (2) $\Delta f = |\mathbf{f}(S_j) - \mathbf{f}(S_k)|$

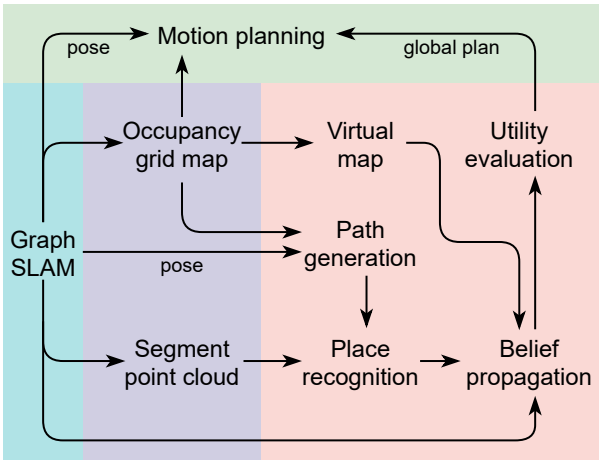


Figure 5.7: System overview of our proposed EM-exploration algorithm.

where $\mathbf{f}$ is a feature descriptor vector, and (3) $\Delta v = v(S_j) \cap v(S_k)$ where the voxel grid v is constructed using the entire cluster of points in a segment. A successful segment matching results in a SM factor describing point cloud registration over two segments. An example is given in Fig. 5.4: two segments, blue and red are tracked at poses $\mathbf{x}_1$ – $\mathbf{x}_2$ and $\mathbf{x}_6$ – $\mathbf{x}_7$ respectively, and segment matching produces a factor connecting $\mathbf{x}_1$ and $\mathbf{x}_6$.

5.3 Implementation Details

In this section, we present an implementation of the proposed EM exploration algorithm. The system architecture is illustrated in Fig. 5.7, and each module contributing to exploration is elaborated in the following subsections.

5.3.1 Virtual Map

How can we predict unobserved landmarks without prior knowledge of the characteristics of an environment? We can approach this question by making a conservative assumption that any location which hasn't been mapped yet has a virtual landmark.

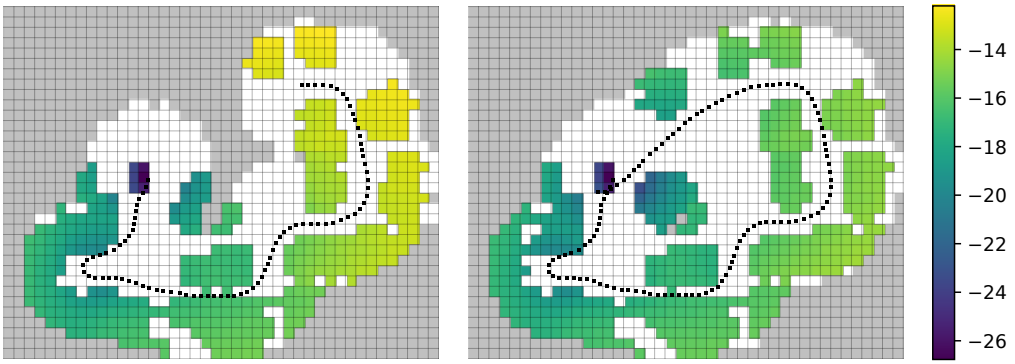


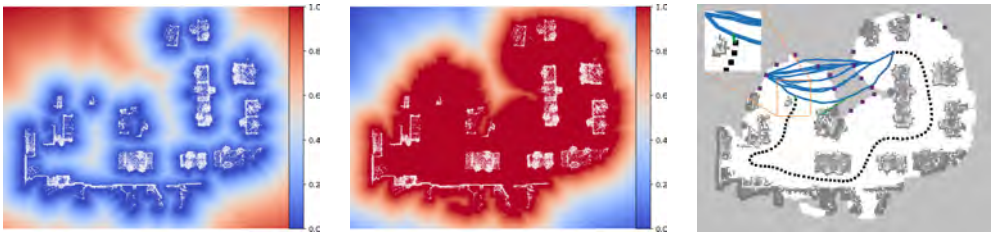
Figure 5.8: Virtual landmarks, shown as grid cells with 0.5m resolution. Their uncertainty (measured by covariance log-determinant) is reduced after closing a loop. Unmapped virtual landmarks (gray) have large initial uncertainty outside the spectrum of values depicted.

Therefore, the probability that a location is potentially occupied with a landmark is strongly related to the traditional occupancy grid map. Let $P(m_i)$ denote the occupancy of a discretized cell, then we define a virtual map V consisting of virtual landmarks with probability

$$P(v_i = 1) = \begin{cases} 1, & \text{if } P(m_i = 1) \geq 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{5.28}$$

Under circumstances with severe drift, the expected map could be utilized to calculate a weighted average of occupancy grid maps from all trajectory hypotheses, as in our previous work [17]. In Fig. 5.8, we whiten the grid cells that, once observed, no longer possess virtual landmarks.

In its definition, existing landmarks have been incorporated into the virtual map, which is essential because minimizing the uncertainty of observed landmarks is also desired. What distinguishes an occupancy grid map from the virtual map is the treatment of unknown space, which consequentially determines what metric



(a) Localization cost $\lambda_l(\mathbf{x})$ (b) Exploration cost $\lambda_e(\mathbf{x})$ (c) Alternative paths

Figure 5.9: Alternative paths to a destination are generated from the shortest path on a variety of cost maps by varying weights in a weighted sum of distance, localization and exploration cost. In (c), the alternative paths to a single frontier location are visualized in blue for clarity. Potential segment matching and pose matching are denoted by green lines.

we use for exploration. In occupancy grid mapping, in order to enable exploration towards unknown space, Shannon entropy is typically leveraged to optimize a path that provides maximum information gain. However, another metric, such as an optimality criterion defined over the covariance matrix, is required to take into account mapping and localization uncertainty. In contrast, virtual landmarks are initialized to have high uncertainty, which is reduced by taking measurements of them, resulting in remarkable information gain. The same metric can be further optimized by improving localization through loop closures, thus unifying the utility measure used in both exploration and localization.

5.3.2 Path Generation

In the M-step, given the distribution of virtual landmarks, path candidates are generated and evaluated using our proposed utility function (Sec. 5.3.5). The global paths that are to be followed over a long span of time must take into account two types of actions, exploration and place-revisiting [84]. Exploration actions normally have destinations near frontier locations where mapped cells meet unknown cells, and to reduce localization uncertainty, place-revisiting actions travel back to locations the

robot has visited, or where it's able to observe a previously observed map segment. The prevalence of these locations requires us to examine a large number of free grid cells in order to obtain a near-optimal solution.

Therefore, we break the path generation problem into two steps. First, frontiers that are reachable from the robot's current location are identified as destinations of our path primitives. Next, we refine the path to a frontier location instead of taking the shortest one using multi-objective optimization for path planning. Let $d(\mathbf{x}_i, \mathbf{x}_j)$ be the distance cost between two poses, let $d_{\text{occupied}}(\mathbf{x}), d_{\text{observed}}(\mathbf{x})$ be the distance costs from a pose to the nearest occupied cell and nearest observed cell respectively, and let r_{sensor} be the maximum sensor range. Then we define (1) the cost for localization as $\lambda_l(\mathbf{x}_j)d(\mathbf{x}_i, \mathbf{x}_j)$ where $\lambda_l(\mathbf{x}_j) = 1 - \exp(-a_l d_{\text{occupied}}/r_{\text{sensor}})$, and (2) the cost to explore as $\lambda_e(\mathbf{x}_j)d(\mathbf{x}_i, \mathbf{x}_j)$ where $\lambda_e(\mathbf{x}_j) = \exp(-a_e d_{\text{observed}}/r_{\text{sensor}})$. The cost maps favoring localization and exploration are illustrated in Figs. 5.9a and 5.9b.

At current step T, we wish to find a path that minimizes the following weighted sum,

$$\sum_{i=1}^{N} \big(w_0 + w_l\lambda_l(\mathbf{x}_{T+i}) + w_e\lambda_e(\mathbf{x}_{T+i})\big)d(\mathbf{x}_{T+i-1}, \mathbf{x}_{T+i}), \tag{5.29}$$

where w_0 is added to allow distance to dominate the cost when the other two terms are close to zero. Considering there is no single optimal path for the above objective function, we employ a simple weighted sum approach to explore Pareto optimal solutions [92], by varying weights $w_l \in [0, 1], w_e = 1 - w_l$ and leaving w_0 as constant. The resulting paths are further pruned to retain only those that are well-separated. We build a roadmap in the collision-free configuration space, and Dijkstra's algorithm is used to search for the shortest path with different cost functions. A representative example is shown in Fig. 5.9c, and it's evident that a few detoured paths are generated to explore unknown space, and to re-observe one map segment and acquire pose

matching (green lines).

5.3.3 Place Recognition

The place recognition module is designed to achieve accurate prediction of both pose matching factors $\{\tilde{\mathbf{f}}_p^{\mathrm{PM}}\}$ and segment matching factors $\{\tilde{\mathbf{f}}_q^{\mathrm{SM}}\}$ given a predefined trajectory. While predicted pose matching occurs in the same manner as in real SLAM, we don't resolve segment matching by predicting LiDAR measurements using a traditional ray-casting algorithm on a voxel grid associated with a segment $v(S)$. Instead, we approximate the measurement by performing a range search on the entire set of points in a segment, and if we are able to gather enough points within sensor range, we are confident that a cluster will be extracted from this segment. The predicted matching is also validated by subsequent poses, and a segment matching factor will be accepted if the process is successful without interruption following a designated number of sequential poses.

5.3.4 Belief Propagation

Belief propagation is concerned with evaluation of Eq. 5.25 given future sequential odometry factors $\{\tilde{\mathbf{f}}_i^{\mathrm{O}}\}$, and more importantly, loop closure candidates from place recognition $\{\tilde{\mathbf{f}}_p^{\mathrm{PM}}\}$, $\{\tilde{\mathbf{f}}_q^{\mathrm{SM}}\}$. Pose covariance recovery follows a standard update of iSAM2, and we use $\tilde{\Sigma}_{\mathbf{x}_i}$ to denote the covariance estimate after incorporating predicted future factors. Clearly, the segment-aided mapping framework doesn't possess landmarks, so we create an imaginary inverse sensor model $\mathbf{v}_k = h^{-1}(\mathbf{x}_i, \mathbf{z}_{i_k})$ that is able to compute the state of a virtual landmark from measurement z that is corrupted by zero-mean Gaussian noise with covariance Λ. The predicted error covariance for the kth virtual landmark from a measurement at the ith pose is $\Sigma_{\mathbf{v}_k}^{(i)} = A_k^i \tilde{\Sigma}_{\mathbf{x}_i} {A_k^i}^T + B_k^{i_k} \Lambda {B_k^{i_k}}^T$, and $A_k^i, B_k^{i_k}$ are Jacobian matrices of the inverse sen-

sor model with respect to pose and landmark. Eventually, we obtain $\hat{\Sigma}_{\mathbf{v}_k}$ by fusing all individual estimates from poses that can observe the kth landmark using SCI in Eq. 5.23. Unobserved andmarks will still have large initial covariance. We illustrate the uncertainty reduction due to loop closure in Fig. 5.8.

5.3.5 Utility Evaluation

As discussed in Sec.5.3.2, a variety of candidate paths are generated and we select the best one among them according to a utility function that maps a path to a scalar. In Eq.5.12, the log-determinant of the covariance matrix is derived from the M-step as the uncertainty metric. Since the estimated covariance has to be fused with a large initial covariance, the log-determinant, or D-optimality, is guaranteed to be monotonically non-increasing during the exploration process, which is consistent with the conclusion in [93]. In addition to uncertainty criteria, it is valuable to incorporate a cost-to-go term to establish a trade-off between traveling cost and uncertainty reduction [84]. Thus, our utility is finalized as,

$$\begin{aligned} U_{\mathrm{EM}}(X_{T:T+N}) = &- \log\det(\tilde{\Sigma}_{\mathbf{x}_{T+N}}) - \sum_k \log\det(\hat{\Sigma}_{\mathbf{v}_k}) \\ &- \alpha d(X_{T:T+N}), \end{aligned} \tag{5.30}$$

where α is the weight on path distance $d(X_{T:T+N})$. In our experiments, we adopt a linearly decaying weight function with respect to traveled distance, whose parameters are determined experimentally and applied consistently throughout our algorithm comparisons below. The selected path is executed immediately, but to account for deviation from the nominal trajectory during execution and inaccurate prediction after taking new measurements, the path is discarded when it's blocked by obstacles

or the robot has traveled a designated distance. The exploration planning process is repeated until no frontier is detected.

5.4 2D Experiments in Simulations

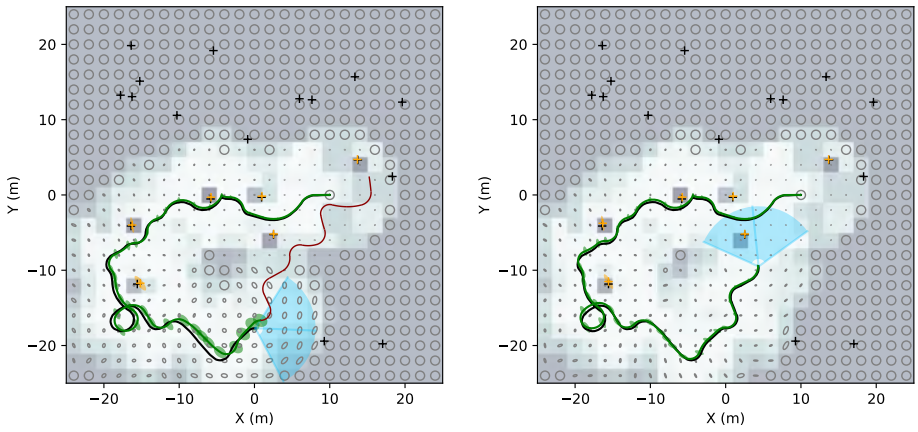


Figure 5.12: Gaussian error ellipses (0.5 standard deviations) of virtual landmarks. Left: The uncertainty of virtual landmarks at the bottom of the map isn't reduced significantly after exploration, as the robot hasn't met a loop-closure, which means the possible landmarks observed in the future would have high uncertainty. The proposed algorithm makes a decision to travel upwards in order to revisit a landmark for better localization. Right: The execution leads to an accurate estimate of the trajectory, and the error ellipses of virtual landmarks shrink significantly.

Table 5.1: Simulation Parameters

Bearing: stddev (deg)	0.5	Translational speed (m/s)	[0.5,1.0]
Range: stddev (m)	0.002	Rotational speed (rad/s)	[-0.5,0.5]
Rotation: stddev (deg)	0.2	Simulation step (s)	0.2
Translation: stddev (m)	0.01	Simulation duration (s)	[1.0,4.0]
Bearing FOV (deg)	120	Safe distance (m)	1.5
Range FOV (m)	[1.0,8.0]	Number of landmarks	20
Initial position: ([m, m])	[10.0,0.0]	Initial sigmas ([m, m, deg])	[0.05,0.05,0.01]

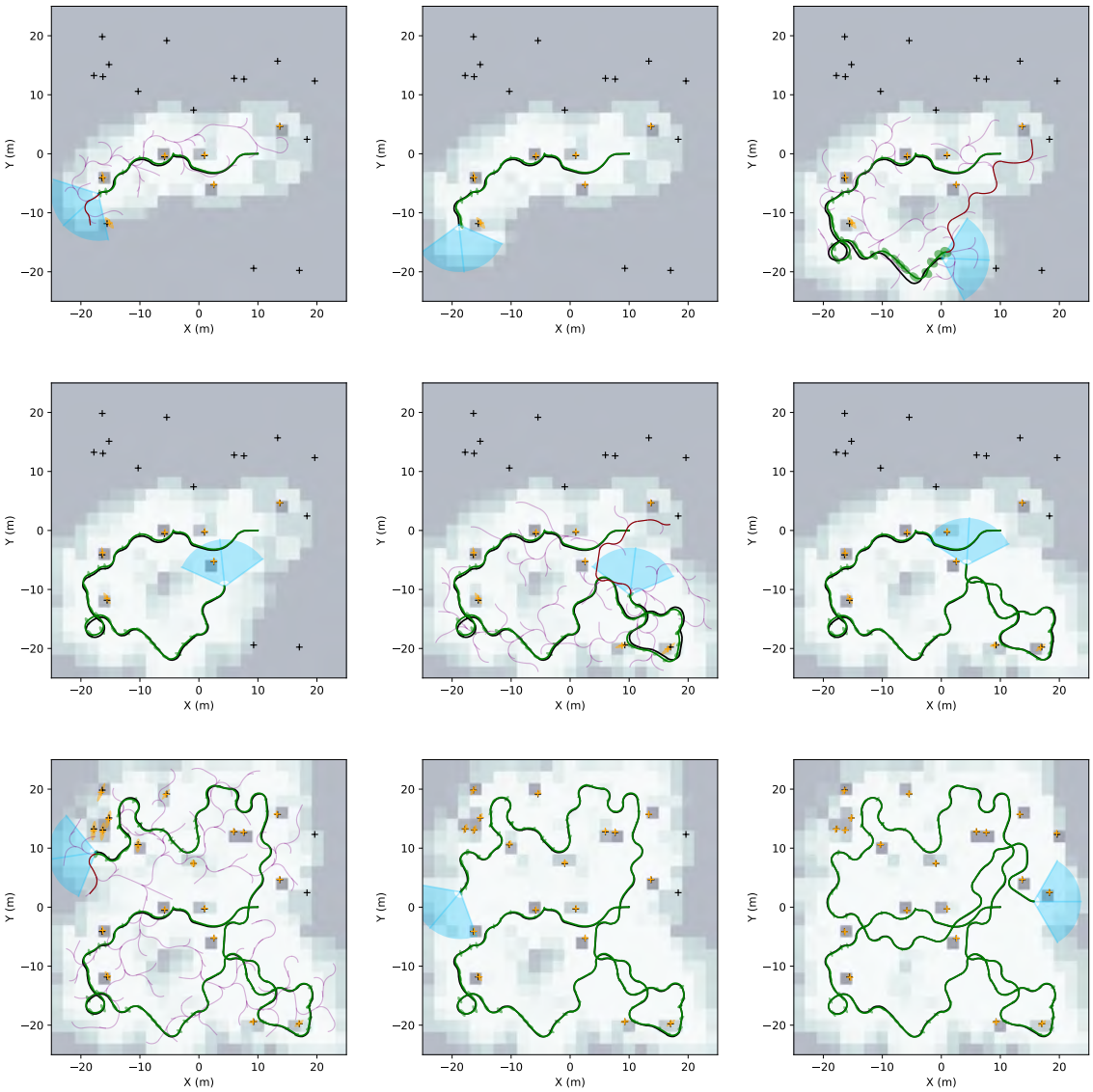


Figure 5.11: A few representative steps of planning and execution using our proposed algorithm, over advancing, but non-consecutive time instants during a single exploration process. Virtual landmarks are embedded in the underlying grid cell environment map with occupancy probabilities shown in grayscale color. Error ellipses (2 standard deviations) of observed landmarks (orange) and robot poses (green) are overlaid on top of the corresponding best estimates. An RRT rooted at the current robot pose is constructed (purple). The trajectory from the RRT offering the expected minimum cost is marked by a red line. From top left to bottom right, one figure showing the expected best trajectory is followed by a figure showing its execution, and the final figure simply represents the final step of the exploration task.

We analyze the performance of the proposed algorithm in a simulated environment (see Fig. 5.11 as an example). The simulation employs an environment with point features uniformly distributed in its inner region (the square $[-20\text{m}, 20\text{m}]$ in Fig. 5.11). The robot is equipped with a range sensor with a limited field of view (FOV) that is capable of measuring the relative range and bearing to a landmark. Zero-mean Gaussian noise is added to both measurements. Our mobile robot is configured as a Dubins car, which we assume has constant translational and rotational speed during one simulation period T. The assumed vehicle dynamics are as follows:

$$x_t = x_{t-1} + v_{T_1} dt \cos\theta_{t-1} \tag{5.31}$$

$$y_t = y_{t-1} + v_{T_1} dt \sin\theta_{t-1} \tag{5.32}$$

$$\theta_t = \theta_{t-1} + \omega_{T_1} dt, \tag{5.33}$$

where $v_{T_1} \in [v_{\min}, v_{\max}], \omega_{T_1} \in [\omega_{\min}, \omega_{\max}], t \in [T_{\min}, T_{\max}]$. Similarly, zero-mean Gaussian noise is added to the state propagation equations above. We also place a circumscribed obstacle radius around landmarks to avoid collision. The robot is initialized with low uncertainty to perform the exploration task. The configuration details of our simulation environment are listed in Table 5.1.

The RRT planner only takes samples within the free space, which is defined by an occupancy probability less than 0.4. To take into account the kinodynamic constraints, we construct a motion library in advance by forward simulating the vehicle with all possible combinations of simulation period, translational and rotational velocity. In the *Steer* function, the connectivity of nodes is checked through searching for the nearest end point in the motion library given a certain radius.

5.4.1 Comparison

We analyze the performance of our proposed algorithm by comparing it with two variants of entropy-based exploration algorithms.

1. SLAM-OG [49]. The utility function consists of a normalized localization component (SLAM) and a normalized occupancy grid (OG) mapping component:

$$U_{SLAM\_OG} = \alpha \frac{I_{SLAM}(\mathcal{X}, \mathcal{L}|\mathbf{a})}{I_{SLAM\max}} + (1-\alpha)\frac{H_{OG}(\mathbf{m}|\mathbf{a})}{H_{OG\max}}, \tag{5.34}$$

   where $I_{SLAM} \approx \sum_{i=1}^{|\mathcal{L}_o|} \sqrt{\det(\Sigma_{\mathbf{l}_i})}$, $H_{OG} = \sum_{\mathbf{m}_i \in \mathbf{m}} H(\mathbf{m}_i)$.

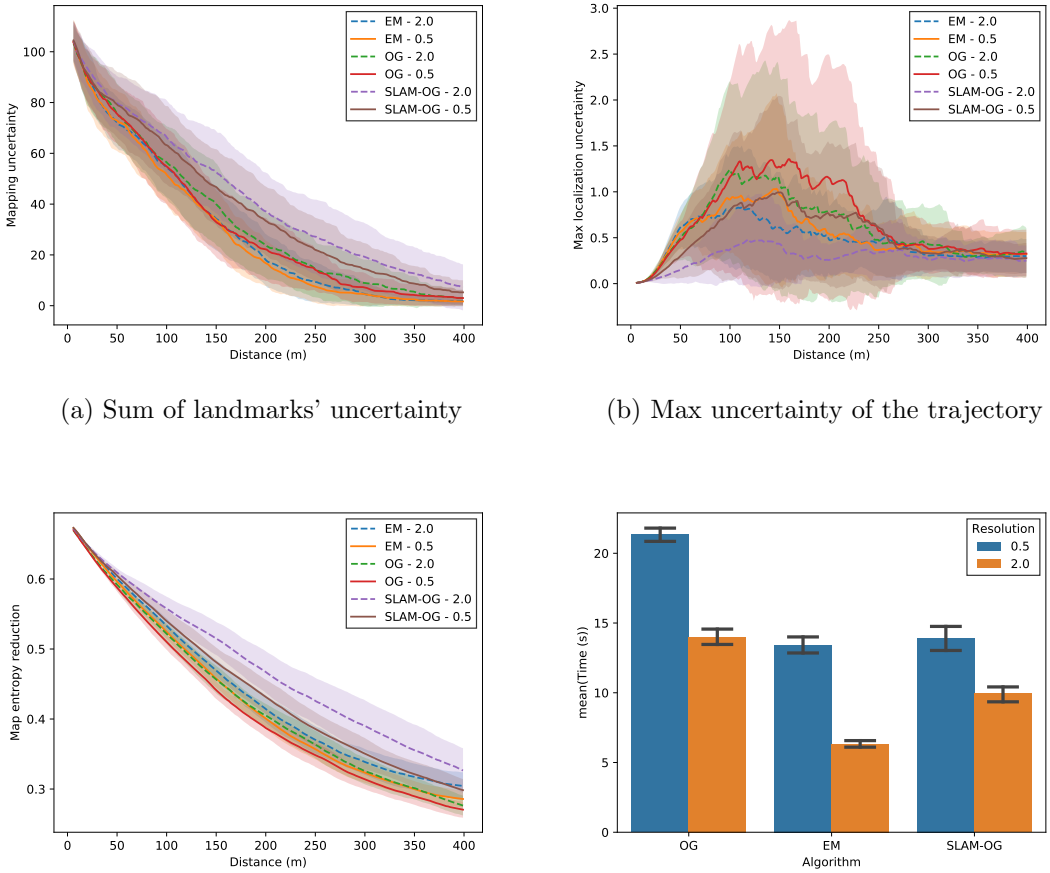
2. OG [94], [84]. The utility function computes the map entropy weighted by the likelihood of robot poses:

$$U_{OG} = \int_{\mathcal{X}} H(\mathbf{m}|\mathcal{X}, \mathcal{Z}) p(\mathcal{X}, \mathcal{L}_o|\mathcal{Z}) \tag{5.35}$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} H(\mathbf{m}^{[n]}|\mathcal{X}^{[n]}, \mathcal{Z}). \tag{5.36}$$

In the first utility function (SLAM-OG), the uncertainty of continuous pose variables is expressed using differential entropy, unlike occupancy grid maps with a discrete probability distribution. As a consequence, the scale of pose uncertainty is much smaller than that of map entropy [93]. The second utility function (OG) prioritizes grid cells that lie inside the robot's current sensor observation cone. In addition, the potential impact of loop closures on previously observed portions of the map (and their accuracy) is ignored, due to the utility function's emphasis on the entropy of occupancy maps.

5.4.2 Results



(a) Sum of landmarks' uncertainty

(b) Max uncertainty of the trajectory

(c) Average entropy reduction

(d) Planning time

Figure 5.13: The results of 50 exploration trials with the same randomly initialized landmarks for every algorithm.

The simulated experiments were conducted such that all the parameters are the same for all algorithms except for the weight of the distance cost. Here we used a distance weight with a constant rate of decay with respect to the area discovered, or $\alpha = w_{\max}\frac{\#\text{free cells}}{\#\text{total cells}}$. For each algorithm, the maximum weight $w_{\max}$ was chosen to achieve the best performance through exhaustive search. Landmarks were uniformly sampled in the environment and 50 environments were generated. We also tested

all algorithms using two levels of resolution for the virtual landmarks and occupancy grids: $2.0(m)$ and $0.5(m)$ to see the effect of using a high resolution map. One example of exploration using the EM algorithm with a low resolution is demonstrated in Fig. 5.11. The performance of the algorithms was averaged among the 50 experiments performed, as listed in Fig 5.13.

Fig. 5.13a shows the uncertainty reduction of landmarks, where the uncertainty is computed by $\sum_{\mathbf{l}\in\mathcal{L}} \mathrm{tr}(\Sigma_{\mathbf{l}})$, in which unobserved landmarks have initial estimates $\sigma_0 = 3.0$. On one hand, this metric takes into account localization and mapping uncertainty through observed landmarks; on the other hand, it also measures the exploration rate by incorporating unknown landmarks. Fig. 5.13b shows the evolution of localization error by evaluating $\max_{\mathbf{x}\in\mathcal{X}} \mathrm{tr}(\Sigma_{\mathbf{x}})$. We also compare different algorithms with respect to average map entropy ($\sum H(\mathbf{m}_i)/N_{\mathbf{m}}$) reduction, shown in Fig. 5.13c. The planning time is shown in Fig. 5.13d under Ubuntu 14.04 on an i7-6950X CPU, although these Python-based implementations are capable of futher optimization.

We have the following observations from the comparisons. The OG algorithm, weighing map entropy by localization uncertainty, has a comparable exploration rate with the proposed algorithm (EM) in the beginning (Fig. 5.13a, 5.13c). However, landmark uncertainty is reduced at a lower rate when most of the landmarks are observed (Fig. 5.13a). In addition, by exploring with the EM algorithm, we end up with more accurate feature-based maps - their curves are closer to zero in the final stage of Fig. 5.13a. The SLAM-OG algorithm places constant relative weights on SLAM and OG uncertainty, and thus its exploration is the most conservative, but it has the lowest localization error as shown in Fig. 5.13b. In contrast, the EM algorithm is able to maintain low trajectory uncertainty and it achieves a superior exploration rate until landmark uncertainty is close to zero. More importantly, the planning time

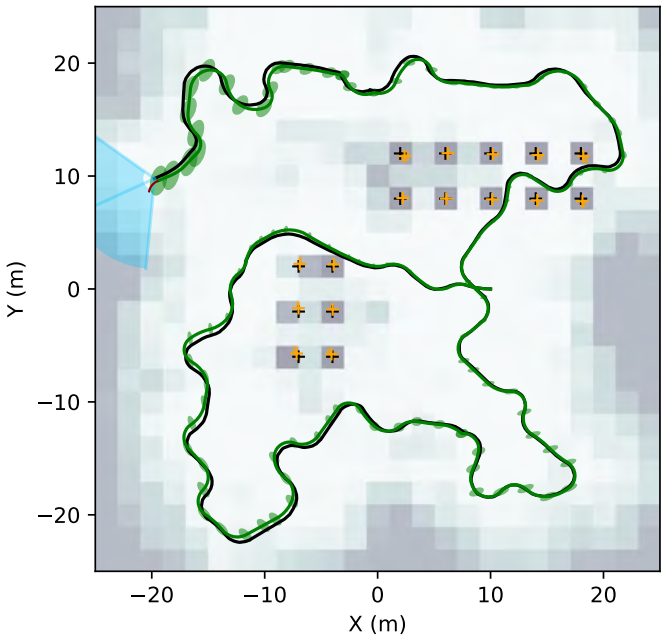


Figure 5.14: An exploration example in a structured environment.

of the EM algorithm is much cheaper because it doesn't need to calculate the map entropy for every candidate by sampling from the posterior trajectory distributions (Eq. 5.36). Overall, by using a finer resolution, the performance of all algorithms improves. However, for the EM algorithm, the small gains in performance do not appear to merit the sacrifice in computation time.

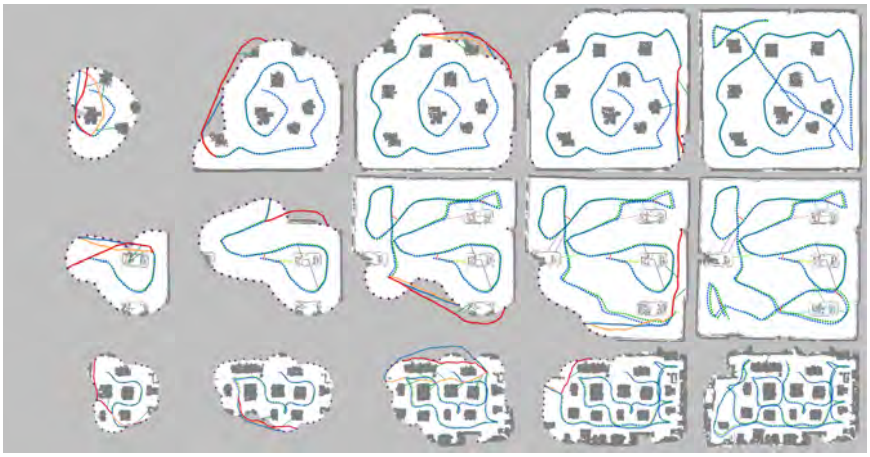


Figure 5.16: Three EM exploration examples in simulated office, parking lot and real-world lab environments (from top to bottom). Trajectory estimate and ground truth are plotted with blue/green dashed lines. We also query the optimal path using different algorithms at the same moment off-line, which are shown with solid lines: EM (red), RHEM (orange) and NBV (blue). Frontiers are represented as purple squares, and predicted place recognition constraints are shown as thin green lines. All examples required 8-9 min. of real-time operation.

5.5 3D Experiments using Pose SLAM

(a) Jackal w/ VLP-16 (b) Stevens ABS laboratory

Figure 5.15: Our UGV platform and test environment.

We analyze the performance of our proposed algorithm in two simulated environments where we can obtain ground truth data, and in a real environment. The robot is

intended to explore the environment given only a bounding box, and terminates when no unknown area is reachable. In all experiments, we use a Clearpath Jackal UGV (Fig. 5.15a), which is equipped with odometry and a VLP-16 LiDAR with 3-meter range, intended to emphasize uncertainty. We implement segment-aided SLAM and EM exploration in ROS, and the simulation is built upon Gazebo.

5.5.1 Comparison

To better assess the proposed algorithm, We compare it with two variants of next-best-view approaches. Throughout the comparison, we use the same path generation method, and the utility function is regarded as the only independent variable. Consequently, our implementations differ somewhat from the documented performance of the original algorithms. We also note that the following utility functions depend entirely on the occupancy grid map, not the virtual map.

Next-best-view (NBV): The NBV planner computes accumulated gain discounted exponentially by distance from start. The gain is defined with regard to the occupancy status of the visible volume at pose $\mathbf{x}_{T+i}$ detailed in [47], and thus

$$U_{\mathrm{NBV}}(X_{T:T+N}) = \sum_{i=1}^{N} \mathrm{Gain}(\mathbf{x}_{T+i}) \exp(-\lambda d(X_{T:T+i})). \tag{5.37}$$

Uncertainty-aware receding horizon exploration and mapping (RHEM): The RHEM planner [48] improves the result in NBV by taking into account vehicle and feature uncertainty. Specifically, we apply U_{NBV} to search for a goal point, then a nested utility function is evaluated on alternative paths to the designated goal point,

$$U^{(2)}_{\mathrm{RHEM}} = -\log\det(\tilde{\Sigma}_{\mathbf{x}_{T+N}}) - \sum_{k} \log\det(\hat{\Sigma}_{\mathbf{l}_k}), \tag{5.38}$$

where we switch to $\mathbf{l}_k$ in the above notation to indicate actual rather than virtual landmarks. Similarly, in a featureless environment, we resort to the same technique to acquire an upper bound of a virtual landmark's covariance, but the utility function in RHEM doesn't include the uncertainty of unknown spaces that have not yet been mapped.

5.5.2 Simulation Environments

Figure 5.17: Gazebo-based simulation environments explored by a simulated Jackal UGV. Results on the right show mapped volume, mapping error and localization error with respect to traveled distance over 30 trials starting from the origin of each map.

The simulated experiments were conducted in two environments ($10\,\text{m} \times 10\,\text{m}$), an office and parking lot, featuring different densities of objects for localization (see Fig. 5.17). All algorithms were executed over 30 trials, starting from the same location. We use three statistics with respect to traveled distance to analyze the performance:

1. Exploration progress is computed as the spatial volume considered as free or occupied in a 3D Octomap [25].

2. We measure the localization quality using root-mean-square error (RMSE) of

the entire trajectory,

$$E_{\text{Trajectory}} = \sqrt{\frac{1}{T}\sum_{i=0}^{T} ||\hat{\mathbf{x}}_i - \mathbf{x}_i||^2}. \tag{5.39}$$

3. Map error is computed as the RMSE of all points transformed based on the estimated trajectory,

$$E_{\text{Map}} = \sqrt{\frac{1}{M}\sum_{k=0}^{M} ||\hat{\mathbf{p}}_k - \mathbf{p}_k||^2}. \tag{5.40}$$

From the results presented in Fig. 5.17, it is evident that the EM algorithm maintains the highest exploration rate in terms of mapped volume, and more importantly it effectively modulates both trajectory and map error. The outcome can be reasoned from a few moments during EM exploration in Fig. 5.17. The NBV approach is solely based on unexplored volume, thus prone to erroneous state estimation. Aside from uncertainty, its utility is exponentially discounted by distance cost, which renders information obtained at a greater distance negligible. To remedy localization error, RHEM is inclined to take a detour to where the robot can reacquire previously mapped segments, but generally the improved path resides in known space. Therefore, improving one utility impairs the other one because of the separation of exploration and localization metrics. To better understand the difference in movements during exploration, we produce heatmaps of robot positions in the parking lot environment (see Fig. 5.18). We can see the robot was more attentive to places close to vehicles in the parking lot using the EM exploration; meanwhile, both RHEM and NBV left a large amount of footprints in the empty area, and the NBV planner had less interest in the right-half, less "informative" space.

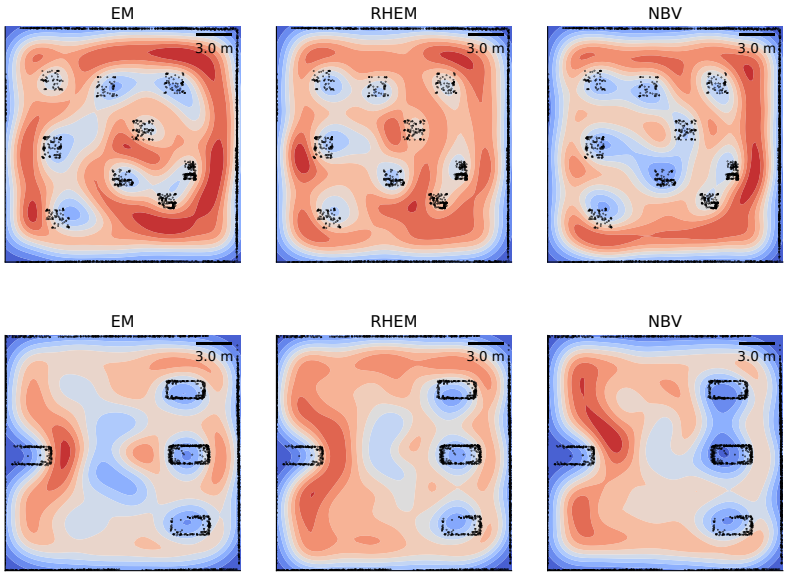


Figure 5.18: Heat map of robot positions using kernel density estimation over 30 experiments in the parking lot environment. The 3-meter scale indicates the sensor range simulated.

5.5.3 Real world environment

We also wish to demonstrate the applicability of our proposed algorithm to real robot platforms. The real-world experimental scenario was located at the Stevens ABS Engineering Center ($25\,\text{m} \times 15\,\text{m}$), similar to the simulated office, occupied with workbenches and chairs (Fig. 5.15b). All parameters are kept the same as in the simulated environments. Progressive instances of a representative execution trace are depicted in Fig. 5.16. We can observe that drift from the ground truth trajectory (obtained from LeGO-LOAM [91]) becomes more severe when the vehicle is further from the starting location. As a result, it traveled through the central region several times for better localization. Inspection of the resulting trajectory illustrates the balance between exploration and exploitation.

5.6 Conclusions

We have adapted an EM exploration algorithm to the widely generalizable context of pose SLAM, which exploits segmentation of point clouds to enhance place recognition. The proposed framework, comprising virtual map construction and computation of the covariance upper bound, offers an advantageous capability of forecasting future actions without requiring the explicit modeling of features in SLAM. The EM algorithm exhibits superior performance in exploration rate, localization and mapping accuracy in two simulated experiments. The computational complexity is discussed in our original work [18], but throughout our experiments, it is able to achieve near real-time querying, and all experiments required less than 10 minutes of real-time operation. However, our path primitives only consist of paths to frontier locations, thus uncertainty continues increasing as exploration progresses. A more appropriately designed routing scheme is desired. Future work also entails validation in larger and more complex real-world environments.

Chapter 6

Autonomous Exploration in Virtual Maps on ROVs

In this section we present the implementation of BlueROV underwater SLAM and exploration algorithms and real experiments in an underwater environment. As shown in Figure 6.1, most components are similar to those implemented on our UGV. Therefore we will focus on the SLAM component of the architecture.

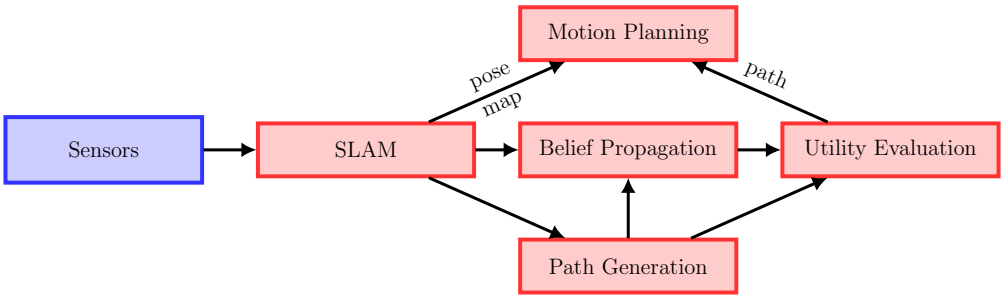


Figure 6.1: Diagram of the underwater exploration framework.

6.1 Keyframe-based SLAM

The underwater SLAM framework is built upon a keyframe-based pose graph. Each keyframe is chosen based on the following heuristic: a keyframe is inserted when the robot moves a certain distance or rotates a certain angle. As shown in Figure 6.2, keyframes are added at around 0.2 Hz. Considering our ROV is operated manually or autonomously at a low speed and its sonar has a large field of view, low-speed state estimation is sufficient in our experiments. The frontend of SLAM performs scan matching on features extracted from sonar images in keyframes. The scan matching relies on an initial pose estimate from dead reckoning, which fuses three sources of measurements: DVL, IMU and pressure sensors at 5 Hz. The constraints from the

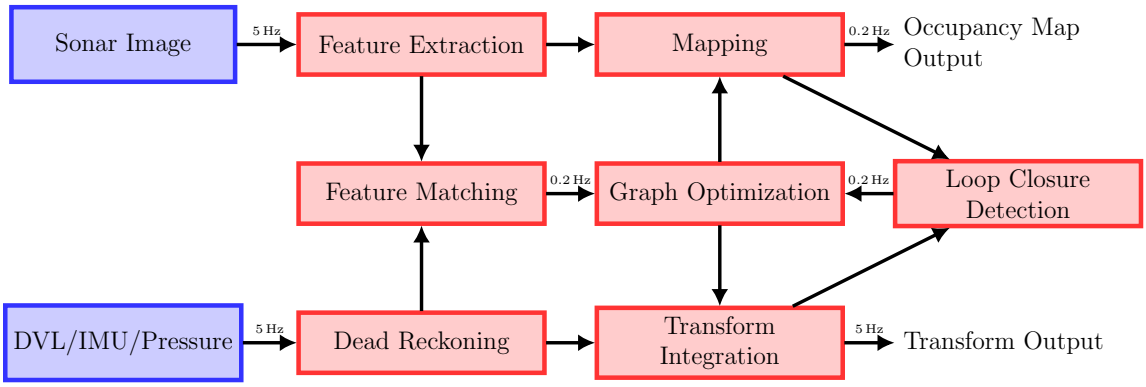


Figure 6.2: Diagram of the underwater SLAM framework.

frontend are incorporated in a pose-only factor graph. Graph optimization on the factor graph provides a low-frequency state estimate, which is integrated with dead reckoning to produce high-frequency estimate. Given the trajectory and feature points at keyframes, the purpose of the mapping module is to generate an occupancy map which is used in path planning. The drift is corrected by the constraints provided by loop closure detection, which serves to match current keyframe with the past history of keyframes.

6.1.1 Feature Extraction

The 2-d imaging sonar emits sound waves and the echo received by the sonar carries the characteristics of the imaging area, which is represented by $\{(r_i, b_j), 0 \leq i \leq N_r, 0 \leq j \leq N_b\}$ in polar coordinates. We assume the measurement corresponding to every beam is independent and each beam is processed separately. Given a beam of intensity measurement $\mathbf{z}_j = \{I_{ij} = I(r_i, b_j), 0 \leq i \leq N_r\}$, we're interested in identifying range bins that represent sound pulses returned by targets in the water. The extracted features provide better situational awareness in cluttered environments, helping improve localization, decision-making and obstacle avoidance. Examples of measurements from 2-d imaging sonar are shown in Figure 6.3.

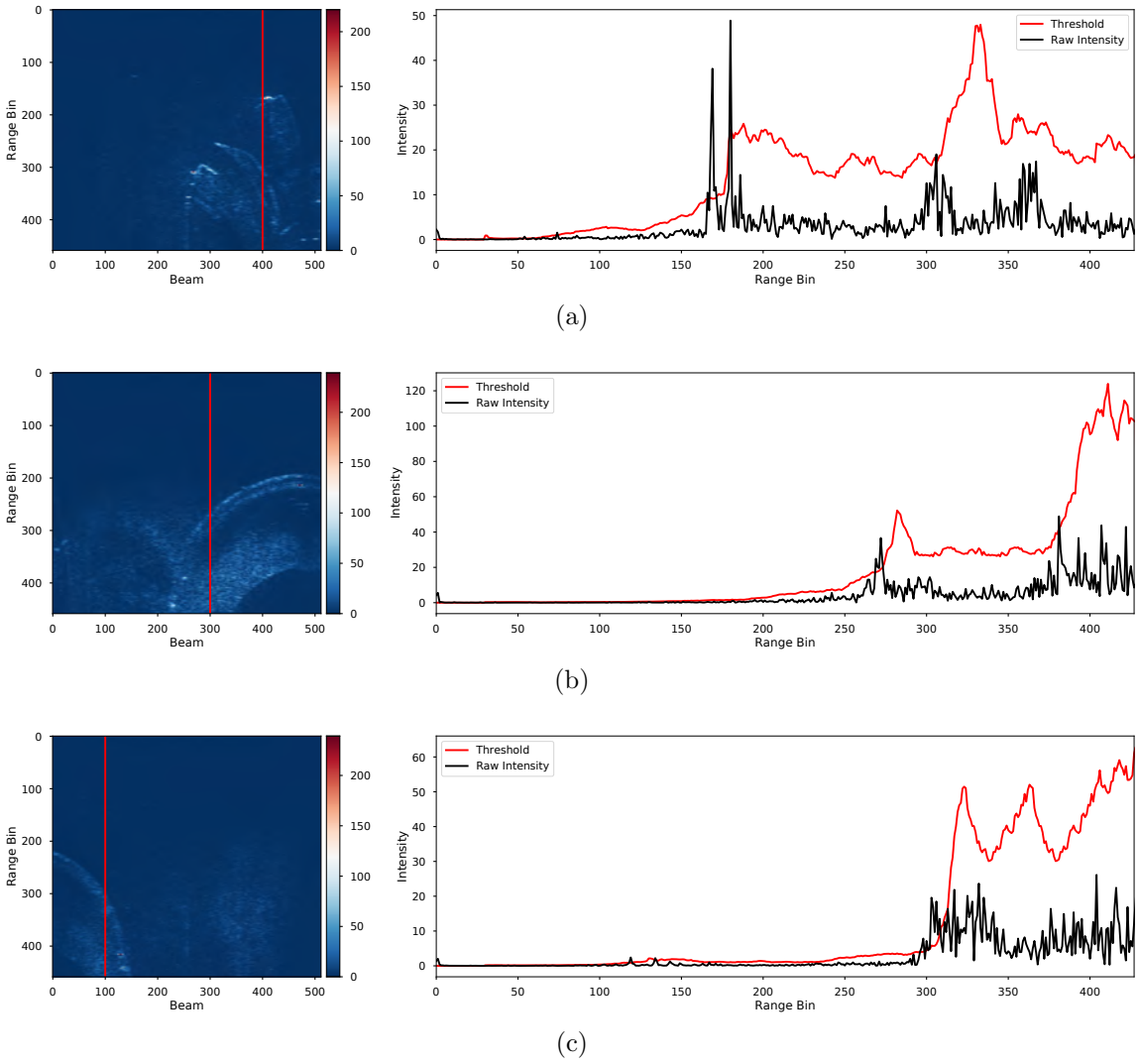


Figure 6.3: CFAR feature detection. Raw sonar images using Oculus M750d with 512 beams are shown at left. The adaptive threshold corresponding to the sonar beams marked with red lines at left using SO-CFAR is represented by the red plot lines on the right. Cells with intensity that is larger than the threshold are treated as targets.

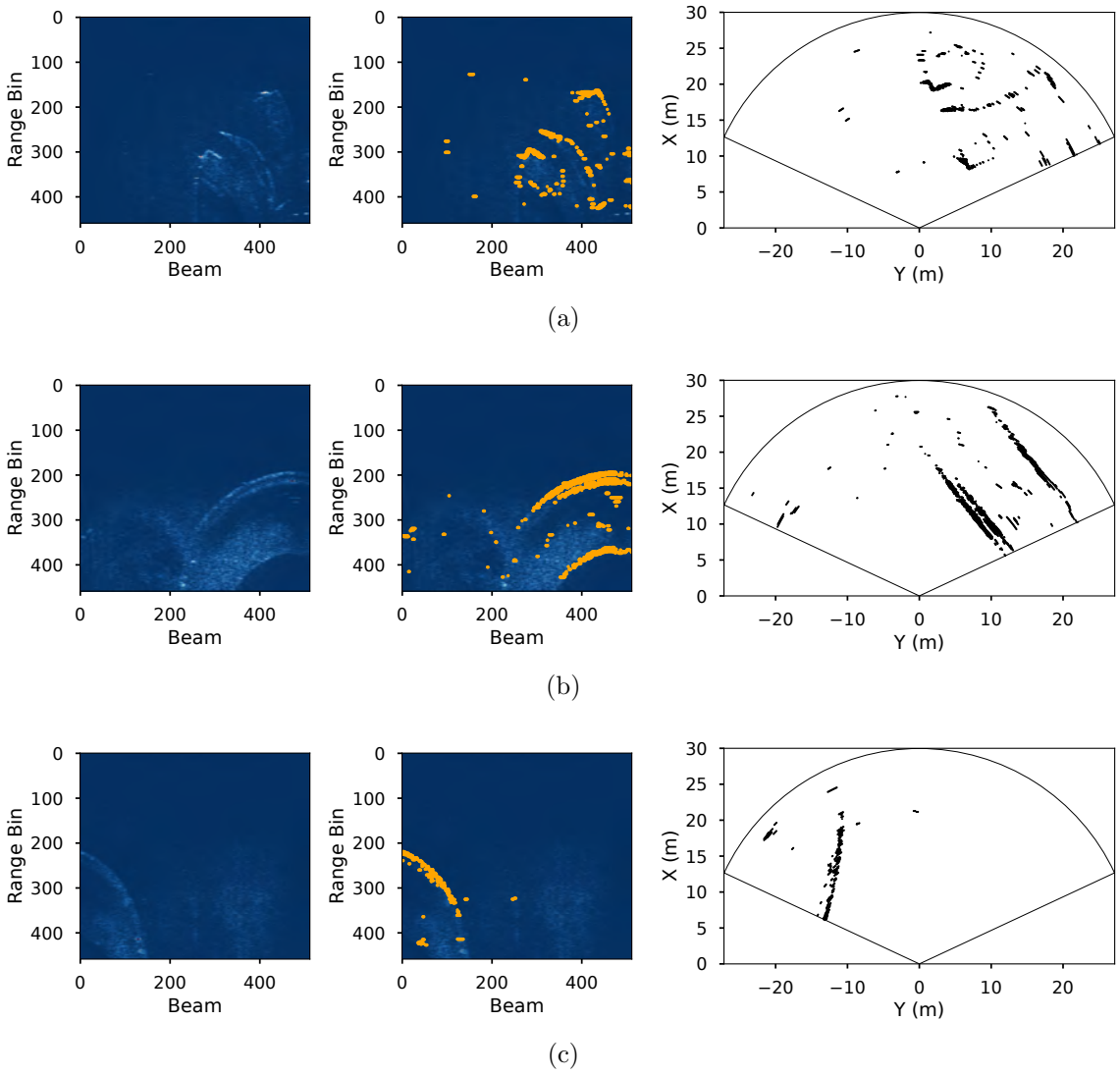


Figure 6.4: SOCA-CFAR feature detection. Similar to Figure 6.3, the left column shows raw data and the middle column visualizes detected features (orange points). Transformed points in Cartesian coordinates are shown on the right.

In order to extract features reliably in all kinds of environments with varying noise power, we employ a Constant False Alarm Rate detector, which initially was designed for radar, for 2-d imaging sonar. Below we use the notions in [95].

We assume a square-law detector on range bins and the intensity at each range bin is independently exponentially distributed with probability density function (*pdf*)

$$f(I_{ij}) = \frac{1}{2\lambda} \exp(-\frac{I_{ij}}{2\lambda}). \tag{6.1}$$

Under Neyman-Pearson hypothesis testing, consider the following hypothesis

$$H_0 : \lambda = \mu,$$
$$H_1 : \lambda = \mu(1 + S),$$

where μ denotes the total background clutter-plus-thermal noise power without targets in range bin (r_i, b_j), and $\mu(1 + S)$ denotes background with added signal with signal-to-noise-ratio (SNR) S in the presence of targets. A cell under test (CUT) is compared against test statistics using cells in a sliding window excluding CUT. Let $\tilde{T}_{ij}$ be the statistics derived from a sliding window (e.g., average intensity in the window), and let $T_{ij} = \tau\tilde{T}_{ij}$ be the test statistics. Then the decision rule is given by

$$I_{ij} \underset{H_0}{\overset{H_1}{\gtrless}} T_{ij}. \tag{6.2}$$

The false alarm rate, i.e., choosing the alternative hypothesis while the null hypothesis is true, is obtained by

$$P_{fa} = \mathbb{E}_{\tilde{T}_{ij}}[P(I_{ij} > \tau\tilde{T}_{ij})] = \int_{\tilde{T}_{ij}} \exp(-\frac{\tau\tilde{T}_{ij}}{2\lambda}) P(\tilde{T}_{ij}) d\tilde{T}_{ij}. \tag{6.3}$$

For a desired false alarm rate, the threshold multiplier τ can be calculated by solving the above equation. To achieve a CFAR property, τ is required to be independent of the distribution parameter λ.

A variety of CFAR detectors have been proposed and one of the most widely used ones is cell-averaging CFAR (CA-CFAR), where the test statistic is defined as the average intensity in a sliding window around the tested range bin. To tackle different operational conditions, e.g., single target, multi-target and clutter edge, smallest-of cell-averaging CFAR (SOCA-CFAR), greatest-of cell averaging CFAR (GOCA-CFAR) [96] and order-statistic (OS) CFAR [97]have been proposed. Based on our test with different experiments with imaging sonar, SO-CFAR provides a better trade-off between accuracy and computation cost.

Suppose the cells around CUT are $\{I_{i-N,j}, ..., I_{i-1,j}, I_{i+1,j}, ..., I_{i+N,j}\}$. Two average intensities computed from leading and trailing window are given by

$$\tilde{T}_{ij,<} = \frac{1}{N}\sum_{n=1}^{N} I_{i-n,j}, \tilde{T}_{ij,>} = \frac{1}{N}\sum_{n=1}^{N} I_{i+n,j}. \tag{6.4}$$

The test statistics in SOCA-CFAR is

$$T_{ij} = \tau \min(\tilde{T}_{ij,<}, \tilde{T}_{ij,>}), \tag{6.5}$$

where τ, given a specified false alarm rate, can be computed as follows [98],

$$P_{\text{fa}} = \big(2 + \frac{\tau}{N}\big)^{-N} \sum_{n=0}^{N-1} \binom{N-1+n}{n} \big(2 + \frac{\tau}{N}\big)^{-n}. \tag{6.6}$$

Features detected by SOCA-CFAR $\{(r_{n_i}, b_{n_j}) | 0 \leq n \leq N\}$ can be transformed to Cartesian coordinates, forming a 2D point cloud as $\{\mathbf{p}_n = (x_n, y_n) \in \mathbb{R}^2 | 0 \leq n \leq$

$N\}$ where

$$x_n = r_{n_i}\cos(b_{n_j}), \quad y_n = r_{n_i}\sin(b_{n_j}). \tag{6.7}$$

A few examples of sonar measurements taken at a marina are shown in Figure 6.3 and Figure 6.4. Figure 6.3 visualizes the decision threshold T_{ij} for beams highlighted by red lines and Figure 6.4 visualizes extracted feature points in polar and Cartesian coordinates.

6.1.2 Feature Matching

In this work, we assume the vehicle is operated in a planar environment and the sonar takes measurements solely from objects in the same plane. Therefore, both robot poses and relative measurements are elements in *special Euclidean group* $\mathrm{SE}(2) \doteq (\mathbf{R}, \mathbf{t}) : \mathbf{R} \in \mathrm{SO}(2), \mathbf{t} \in \mathbb{R}^2\}$. Let $\mathbf{x}_s \in \mathrm{SE}(2), \mathbf{x}_t \in \mathrm{SE}(2)$ be two 2D poses at which two sets of features, denoted as $\mathcal{P}_s, \mathcal{P}_t$, are observed respectively. The 2D points are represented in a local frame. Presumably, $\mathbf{x}_t$ and $\mathbf{x}_s$ are referred to as *target* pose and *source* pose, and $\mathcal{P}_t$ and $\mathcal{P}_s$ are referred to as *target* points and *source* points. Assuming the vehicle is operated in a planar environment and the sonar takes measurements solely from objects in the same plane, the relative transformation between two poses at which two sonar images are taken can be recovered using scan matching techniques. Specifically, we want to recover the relative transformation between two poses $\mathbf{x}_{ts} \in \mathrm{SE}(2)$ combining relative rotation $R_{ts} \in \mathrm{SO}(2)$ and relative translation $\mathbf{t}_{ts} \in \mathbb{R}^2$,

$$\mathbf{x}_s = \mathbf{x}_t \oplus \mathbf{x}_{ts}, \mathbf{x}_{ts} = \mathbf{x}_t \ominus \mathbf{x}_s = (\mathbf{R}_{ts}, \mathbf{t}_{ts}), \tag{6.8}$$

$$\mathbf{R}_{ts} = \mathbf{R}_t^\top \mathbf{R}_s, \mathbf{t}_{ts} = \mathbf{R}_t^\top(\mathbf{t}_s - \mathbf{t}_t)), \tag{6.9}$$

where we use $\oplus$ and $\ominus$ to denote the composition and inverse composition of two elements in SE(2).

Iterative Closest Point

The most widely used algorithm is iterative closest point (ICP), which minimizes the following cost function

$$J(\mathbf{x}_{ts}) = \sum_{\mathbf{p}_t \in \mathcal{P}_t, \mathbf{p}_s \in \mathcal{P}_s} ||\mathcal{T}_{ts}\mathbf{p}_s - \mathbf{p}_t||^2 \tag{6.10}$$

$$= \sum_{\mathbf{p}_t \in \mathcal{P}_t, \mathbf{p}_s \in \mathcal{P}_s} ||\mathbf{R}_{ts}\mathbf{p}_s + \mathbf{t}_{ts} - \mathbf{p}_t||^2, \tag{6.11}$$

where $\mathcal{T}\mathbf{p} = \mathbf{R}p + \mathbf{t}$ represents the action of $\mathbf{x} \in \mathrm{SE}(2)$ on $\mathbf{p} \in \mathbb{R}^2$. The correspondence between $\mathbf{p}_t$ and $\mathbf{p}_s$ in the above cost function is established by nearest neighbor search. However it's well known that ICP is plagued by the problem of local minima due to non-convexity. The final result heavily depends on the quality of the initial transformation. Typically the initial guess is estimated by dead reckoning, for instance, from IMU, wheel odometer, etc. The scan matching using ICP is impaired more significantly in underwater environments for three main reasons.

1. Features extracted from sonar images are sparse due to limited angular resolution. The imaging sonar in our experiments has angular resolution of 1°, which limits its tangential resolution at 30 m range no more than 0.5 m.

2. Measurements are noisy as a result of the low signal-to-noise-ratio of imaging sonar.

3. Dead-reckoning is typically poor without a navigation-grade IMU. The main cause of dead-reckoning error is the drift of heading estimation. For example,

the IMU in our experiments achieves heading accuracy of 2° RMS with a stable magnetic environment.

Examples of using ICP on sonar features in two consecutive frames are illustrated in the second column in Figure 6.5. Compared to our method, it's evident that ICP results initialized from odometry get stuck in local minima. The problem is more noticeable during loop closure, where a large amount of drift has accumulated, as shown in Figure 6.6.

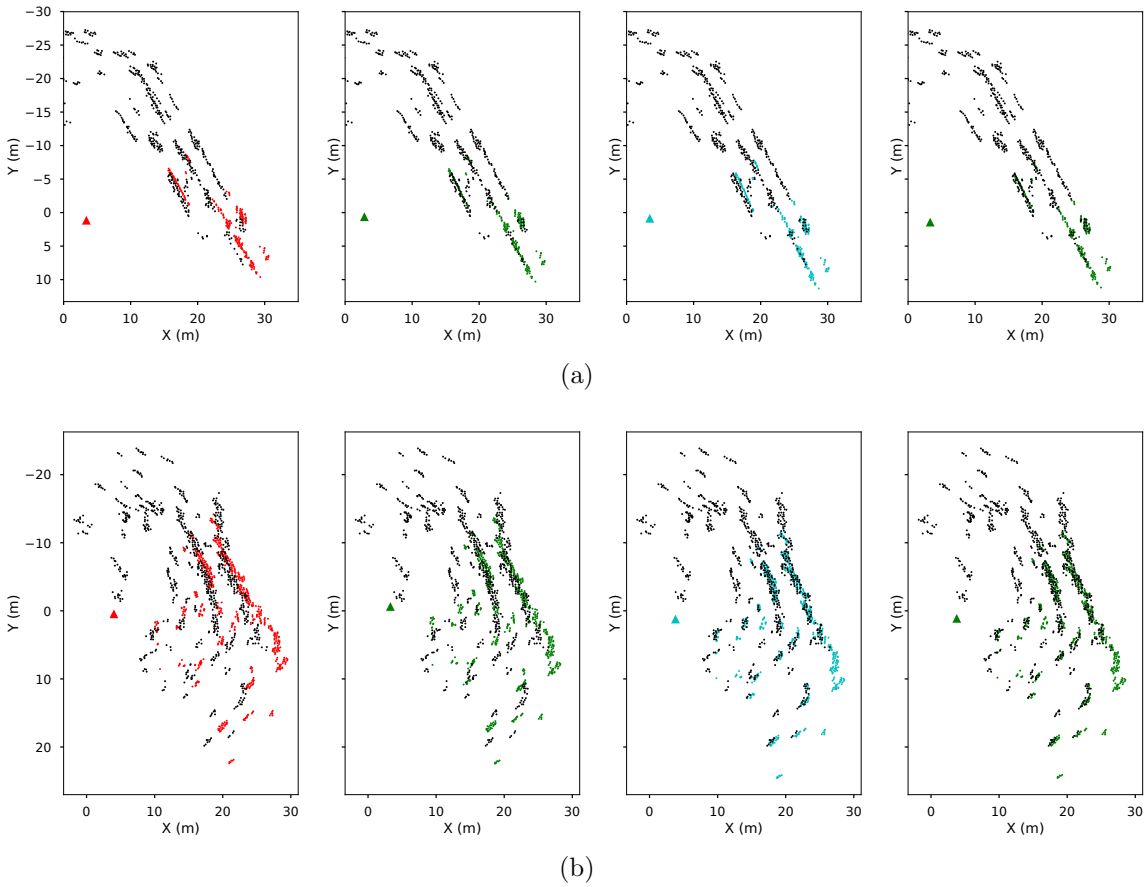


Figure 6.5: Illustration of our proposed sonar scan matching method applied to sequential keyframes. Colored points (source) represent features extracted from the current sonar frame and black points (target) represent accumulated features using the previous 3 frames. From left to right: initial transformation from odometry; ICP result based on initial guess; initialization result from our proposed sensing model; refined ICP result based on initialization in the third column.

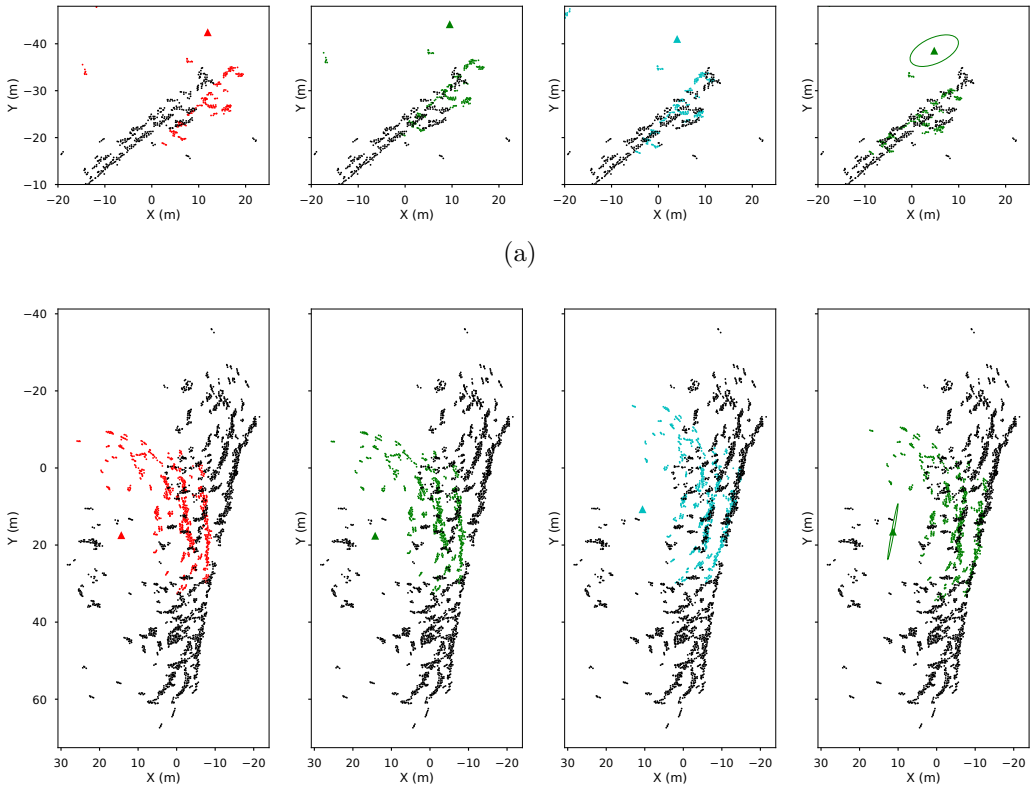


Figure 6.6: Illustration of our proposed sonar scan matching method applied to non-sequential keyframes to detect loop closure. Colored points (source) represent features extracted from the current sonar frame and black points (target) represent accumulated features. From left to right: initial transformation from odometry; ICP result based on initial guess; initialization result from our proposed sensing model; refined ICP result based on initialization in the third column.

Globally initialized ICP

Recently globally optimal solution of the ICP problem has been studied [99]; however its optimality defined by minimum matching distance isn't necessarily correct. In this work, we propose to alleviate the problem of local minima by performing a global initialization beforehand. Then ICP is used to locally refine the estimate. As shown in real experiments this tentative solution yields good results.

Let $\tilde{\mathbf{x}}_s$ be the globally initialized source pose. The proposed global initialization

follows set consensus maximization, which solves the following problem,

$$\tilde{\mathbf{x}}_s = \underset{\mathbf{x}_s}{\operatorname{argmax}} \sum_{\mathbf{p}_s \in P_s} \mathbb{I}(d(\mathbf{p}_s) \leq \epsilon), \tag{6.12}$$

where $\mathbb{I} = 1$ when the condition is true, and $\mathbb{I} = 0$ otherwise. d is defined as the distance to the nearest target point,

$$d(\mathbf{p}_s) = \min_{\mathbf{p}_t \in \mathcal{P}_t} \|\mathbf{R}_{ts}\mathbf{p}_s + \mathbf{t}_{ts} - \mathbf{p}_t\|. \tag{6.13}$$

A probabilistic derivation

We model scan matching using the same probabilistic formulation in [100]. We apply Bayes' rule to find the posterior distribution over robot's source pose

$$p(\mathbf{x}_s|\mathbf{x}_t, \mathcal{P}_t, \mathcal{P}_s, \mathbf{u}) \propto p(\mathbf{x}_s|\mathbf{x}_t, \mathbf{u}) p(\mathcal{P}_s|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t). \tag{6.14}$$

The first term can be obtained from odometry and is generally modeled as a multivariate Gaussian distribution. However we simply assume the pose is uniformly distributed around the initial guess provided by odometry, which results in

$$p(\mathbf{x}_j|\mathbf{x}_t, \mathcal{P}_t, \mathcal{P}_s, \mathbf{u}) \propto p(\mathcal{P}_s|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t). \tag{6.15}$$

The second model describes the probability of observing $\mathcal{P}_s$ at $\mathbf{x}_s$ given the existing point cloud, or map, $\mathcal{P}_t$ at $\mathbf{x}_t$. If we assume an individual point is measured independently, it can be represented as

$$p(\mathcal{P}_s|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t) = \prod_{\mathbf{p}_s \in \mathcal{P}_s} p(\mathbf{p}_s|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t). \tag{6.16}$$

Now consider the following measurement model,

$$p(\mathbf{p}_s|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t) = \begin{cases} p_1, & \text{if } d(\mathbf{p}_s) \leq \epsilon \\ p_2, & \text{otherwise.} \end{cases} . \tag{6.17}$$

It's more likely to detect features in the vicinity of existing measurements, thus $p_1 > p_2$. We can derive the same set consensus maximization problem as Equation 6.12,

$$\begin{aligned} \tilde{\mathbf{x}}_j &= \underset{\mathbf{x}_s}{\text{argmax}} \sum_{\mathbf{p}_j \in \mathcal{P}_s} \log p(\mathbf{p}_j|\mathbf{x}_s, \mathbf{x}_t, \mathcal{P}_t) \\ &= \underset{\mathbf{x}_s}{\text{argmax}} \sum_{d(\mathbf{p}_j) \leq \epsilon} \log p_1 + \sum_{d(\mathbf{p}_j) > \epsilon} \log p_2 \\ &= \underset{\mathbf{x}_s}{\text{argmax}} \sum_{d(\mathbf{p}_j) \leq \epsilon} \log p_1 + \sum_{d(\mathbf{p}_j) > \epsilon} \log p_2 - \sum_{d(\mathbf{p}_j) \leq \epsilon} \log p_2 - \sum_{d(\mathbf{p}_j) > \epsilon} \log p_2 \\ &= \underset{\mathbf{x}_s}{\text{argmax}} \sum_{d(\mathbf{p}_j) \leq \epsilon} \log \frac{p_1}{p_2} \\ &= \underset{\mathbf{x}_s}{\text{argmax}} \sum_{d(\mathbf{p}_j) \leq \epsilon} 1. \end{aligned} \tag{6.18}$$

Algorithm

Intuitively, the optimal pose corresponds to the pose that is able to observe the most points around reference cloud. More interestingly, the optimization can be achieved without specifying parameters in the sensor model. While in theory the exhausive search of source pose could be expensive, we present two methods to speed up the process. Firstly, the uncertainty of the most recent pose, which typically is the source pose, can be obtained effortlessly assuming we use incremental smoothing and mapping [86]. Therefore, it's reasonable to limit the search within a certain confidence interval around the mean estimate. Secondly, a nearest neighbor query such as using

k-d tree can be avoided by generating a look-up table in advance. Each cell in the look-up table stores whether there exists a target point within radius ϵ.

Any global and derivative-free optimization algorithms could be used to solve Equation 6.12 or 6.18. Specifically in this work we choose the SHGO algorithm [101] for fast convergence, which has been implemented in Scipy[1]. Let $\tilde{\mathbf{x}}_s$ be the optimized source pose, $\tilde{\mathbf{x}}_{ts} = \mathbf{x}_t \ominus \tilde{\mathbf{x}}_s$ be the resulting transformation. Next ICP is performed using the initialization $\tilde{\mathbf{x}}_{ts}$ and we derive a refined solution $\hat{\mathbf{x}}_{ts}$. In the next section, we will show its application of building sequential and non-sequential constraints in pose SLAM. Here we only demonstrate its performance given input poses and points as in Figure 6.5 and 6.6.

The estimated transformation from scan matching can be erroneous. We propose the following creterion to reject outliers.

1. Both source and target should contain enough points,

$$|\mathcal{P}_t| > \eta_{\text{min\_points},i}, |\mathcal{P}_s| > \eta_{\text{min\_points},j}.$$

2. The estimated transform should not deviate significantly from the initial transformation,

$$\|\hat{\mathbf{x}}_{ts} \ominus \mathbf{x}_{ts}\| < \eta_{\text{deviation}}.$$

3. The target and source points should have enough overlap upon finishing matching. The overlap is calculated as

$$\frac{1}{|\mathcal{P}_s|} \sum_{\mathbf{p}_s \in \mathcal{P}_s} \mathbb{I}(d(\mathbf{p}_s)) < \epsilon) > \eta_{\text{overlap}}.$$

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.shgo.html

6.1.3 Covariance Estimation

Estimating the uncertainty of our scan matching estimate is crucial for accurate state estimation. However existing closed-form approaches (e.g., [102], [103]) to estimating ICP uncertainty is likely to be overly optimistic, i.e., the covariance is smaller. Monte Carlo methods, which essentially run ICP with different initial parameters, are able to accurately reveal the uncertainty caused by erroneous convergence in local minima.

Suppose we repeatedly run ICP N times and at each time a different initial guess is provided. As we use SHGO optimization in the global initializaiton, we keep track of the samples during maximization and select N samples with the highest objectives. Let $\hat{\mathbf{x}}_{ts}^{(i)}$ be the estimate at the i-th run. Then the covariance matrix is given by

$$\hat{\Sigma}_{\mathbf{x}_{ts}} = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{\mathbf{x}}_{ts}^{(i)} - \bar{\hat{\mathbf{x}}}_{ts})(\hat{\mathbf{x}}_{ts}^{(i)} - \bar{\hat{\mathbf{x}}}_{ts})^{\top}, \quad \bar{\hat{\mathbf{x}}}_{ts} = \frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{x}}_{ts}^{(i)}. \tag{6.19}$$

6.1.4 Building the Factor Graph

In this section, we discuss the construction of factor nodes in the factor graph. As shown in Figure 6.7, there are two types of factor nodes computed from sequential scan matching and non-sequential scan matching, denoted as green and orange nodes respectively. The sequential factors represent the relative relationship with respect to the previous pose; error is unavoidably accumulated on the chain of poses. In contrast, non-sequential factors, also known as loop closure constraints, connect two poses that are separated in terms of time and thus are able to correct the drift of recent poses. The factor graph is given by

$$\mathbf{f}(\boldsymbol{\Theta}) = \mathbf{f}^{0}(\boldsymbol{\Theta}_0)\prod_{i}\mathbf{f}_{i}^{\mathrm{O}}(\boldsymbol{\Theta}_i)\prod_{j}\mathbf{f}_{j}^{\mathrm{SSM}}(\boldsymbol{\Theta}_j)\prod_{q}\mathbf{f}_{q}^{\mathrm{NSSM}}(\boldsymbol{\Theta}_q).$$

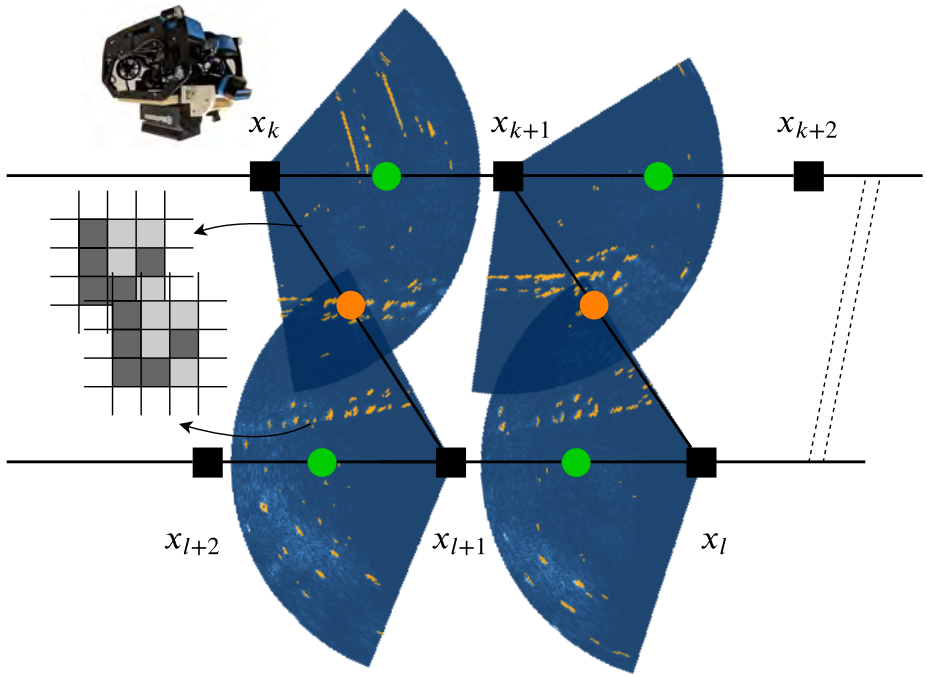


Figure 6.7: Key components of the SLAM system. The pose variable at each keyframe is denoted as black squauare. The core factor graph is comprised of pose variables at keyframes (black squares) and two types of factors: sequential scan matching factors (blue circle) and loop closure factors (orange circle). Every keyframe is associated with feature points detected in the sonar image. The detected points at every keyframe independently contribute to the final occupancy grid map.

Sequential Scan Matching

Sequential factors between two consecutive poses $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ can be computed through scan matching on two point clouds collected at these two moments. To increase the robustness of scan matching, we incorporate extracted features from the previous $N_{ssm} > 1$ frames, assuming the drift is negligable within a short period of time. Let $\mathcal{T}_{k-1,k-i}\mathcal{P}_{k-i}$ be the transformed points at keyframe $k-i$ in the coordinate

frame of pose $k-1$, then the target points are given by

$$\mathcal{P}_t := \bigcup_{i=1}^{N_{\text{ssm}}} \mathcal{T}_{k-1,k-i}\mathcal{P}_{k-i}. \tag{6.20}$$

An illustrative diagram of building sequential factors is shown in Figure 6.8. As discussed in the previous section, in the case that scan matching failed, we instead use the initial transformation from dead reckoning.

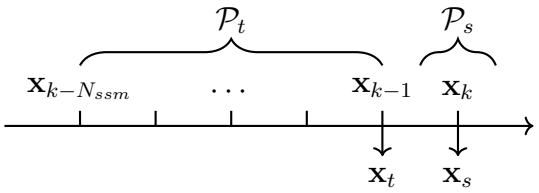


Figure 6.8: Building sequential factors from scan matching.

Non-sequential Scan Matching

Non-sequential scan matching, or loop closure detection in Figure 6.2, follows the same procedure as that in sequential matching, except that source points are limited to be meausured at an earlier time. The connection results from recognizing features that have been observed before. Non-sequential factors serve to eliminate the drift from dead reckoning or sequential scan matching.

Instead of using source points only from the current keyframe, we construct source points in a simular way to that in sequential scan matching as follows,

$$\mathcal{P}_s := \bigcup_{i=1}^{N_{\text{nssm}}} \mathcal{T}_{k,k-i}\mathcal{P}_{k-i}. \tag{6.21}$$

We limit the searching window of target frames to be far away from the current frame. Let t_n be the last keyframe in the target points and we impose the following condition

$k - t_n > N_{\text{sep}}$. Initially all keyframes prior to t_n are used for global initialization and upon finishing initialization keyframes that don't overlap with source points are discarded. The keyframe with the maximim overlapping ratio with source points is treated as the target pose. Figure 6.9 demonstrates one example set-up of non-sequential scan matching.

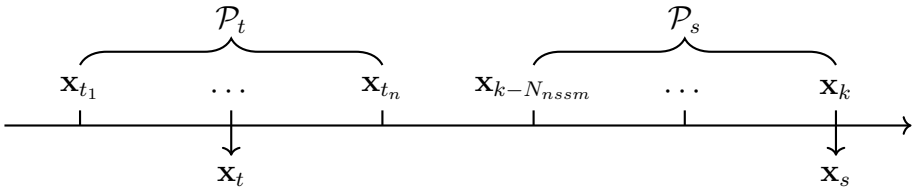


Figure 6.9: Building non-sequential factors from scan matching.

Outlier Rejection

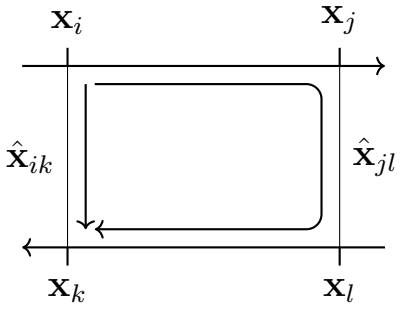


Figure 6.10: Pairwise consistency check. Two loop closure constraints, $\hat{\mathbf{x}}_{jl}$ and $\hat{\mathbf{x}}_{ik}$, are obtained. The consistency is checked by comparing the difference of transformations following two directions.

Although a few verification methods are discussed, erroneous loop closure detection still exists. The rejection of such outliers could only be achieved with the assistance of other constraints. We observe that correct loop closure contraints should be consistent (defined below) with the current pose estimate and potentially future loop closure constraints. Therefore, assuming outliers occur with low probability, a set of consistent measurements is likely to be correct.

We adopt the pairwise consistent measurement set maximization (PCM) [104] technique to reject outliers. Let $\hat{\mathbf{x}}_{jl}$ and $\hat{\mathbf{x}}_{ik}$ be a pair of loop closure measurements and we assume $l < k$. An estimate of $\hat{\mathbf{x}}_{ik}$ can be obtained using estimated poses and measurement $\hat{\mathbf{x}}_{jl}$ given by

$$\hat{\mathbf{x}}'_{ik} = \mathbf{x}_{ij} \oplus \hat{\mathbf{x}}_{jl} \oplus \mathbf{x}_{lk}. \tag{6.22}$$

The flow of relative transformations is illustrated as the long arrow in Figure 6.10. The two constraints are determined to be pairwise consistent if the following condition is satisfied,

$$\|\hat{\mathbf{x}}_{ik} \ominus \hat{\mathbf{x}}'_{ik}\|_{\Sigma} \leq \eta_{\text{pcm}}, \tag{6.23}$$

where $\|\cdot\|_{\Sigma}$ calculates the Mahalanobis distance and Σ can be obtained from ICP covariance $\hat{\Sigma}_{ik}$.

Given a set of measurements, a mutual consistency check is performed. The outcome forms an undirected graph with nodes representing pairs of measurements and edges representing pairwise consitency. An example with 4 measurements is visualized in Figure 6.11 and only 3 out of 6 pairs are pairwise consistent. The largest subset of pairwise consistent measurements, where every pair in the subset is pairwise consistent, can be identified as a maximum clique in the graph. Although finding the maximum clique is expensive and hard to approximate, fortunately the graph is typically small enough that exhausitively searching for the maximal clique is achievable in real-time experiments.

In implementation, we maintain a queue of loop closure measurements by arrival time. At each keyframe the PCM is executed to find the maximum clique. If the clique number is larger than N_{pcm}, all measurements in the clique are regarded as correct and those which haven't been added to factor graph are added. In Figure 6.11,

if $N_{\text{pcm}} = 3$, three correct measurements will be accepted until $\mathbf{x}_{13}$.

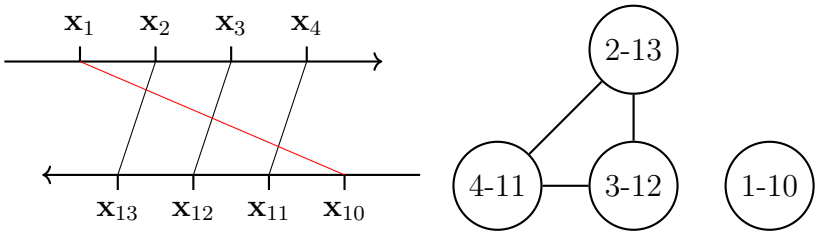


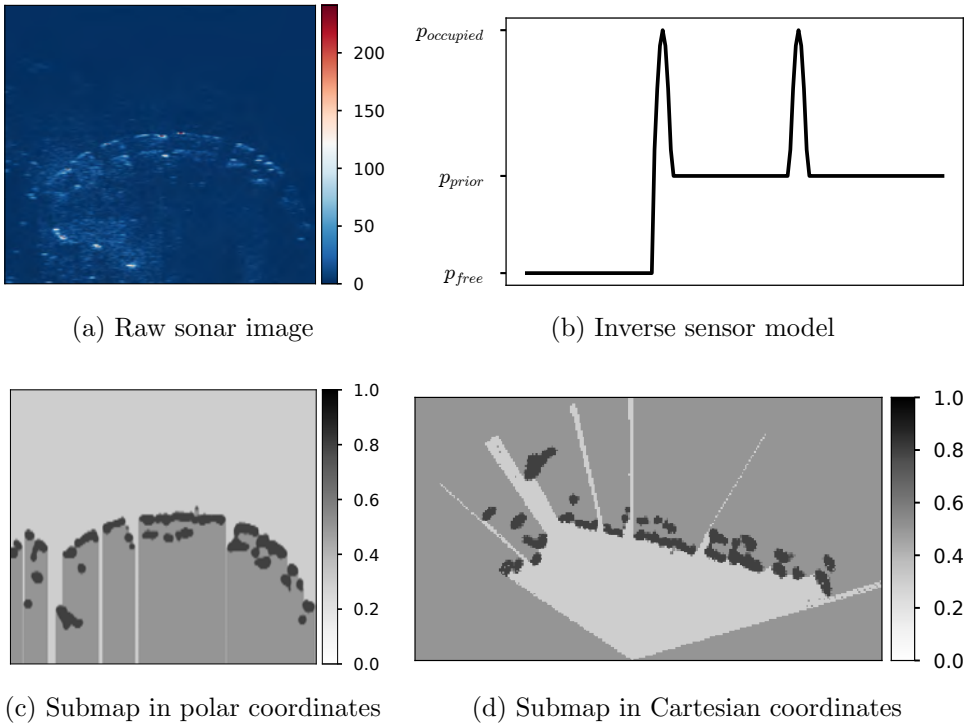
Figure 6.11: Four pairs of loop closures are detected as constraints between two keyframes. The erroneous one (red) is excluded in the maximum clique.

6.1.5 Occupancy Mapping

Our occupancy mapping framework is based on the occupancy grid mapping algorithm [26]. The entire map is discretized into independent grid cells and the occupancy probability of each cell is updated recursively through Bayes' filter. However our state estimation from SLAM exhibits severe drift; and at the moment the drift is corrected by loop closures it's desired to correct the occupancy mapping error using the updated trajectory. Therefore, we employ a submap-based occupancy mapping algorithm [105], which builds a local map anchored at each keyframe. We are able to achieve efficient map recalculation when a segment of the trajectory is changed.

Merging Submap

Suppose we are able to obtain keyframe poses in a reference frame from SLAM denoted as $\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^{K}$. Given the sonar image and extracted features, we can build a submap located at a local frame denoted as $\mathcal{S}_k = \{\mathbf{m}_i^k\}$ where $\mathbf{m}_i^k \in \mathbb{R}^2$ represents a 2D grid cell in the k-th keframe. Let $p(\mathbf{m}_i^k = 1)$ represent the probability of the cell being occupied and $l^k(\mathbf{m}_i^k) = \log \frac{p(\mathbf{m}_i^k=1)}{1-p(\mathbf{m}_i^k=1)}$ represents its log-odds notation. We use superscript k to denote cells or log-odds values in the k-th keyframe. The entire map is represented as the composition of submaps $\mathcal{M} = \{\mathcal{S}_k\}_{k=1}^{K}$.



(a) Raw sonar image

(b) Inverse sensor model

(c) Submap in polar coordinates

(d) Submap in Cartesian coordinates

Figure 6.12: Inverse sensor model and the submap from one sonar image.

In our keyframe-based SLAM, a submap only consists of one sonar frame, thus the occupancy values can be calculated from the inverse sensor model directly. Unlike laser beams, sonar is capable of detecting targets that are hidden behind the first object along sensor beams due to wide vertical aperture. This phenomenon is more obvious in our field test as there exist floating docks that don't block sonar beams entirely. Similar to the inverse sensor model for laser beams where cells before, around and behind the hit point are modeled as *free*, *occupied* and *unknown*, we also treat hit points behind the first one as *occupied*. In essence all detected targets are believed to be occupied but only the area in front of the first target is believed to be obstacle free. The inverse sensor model is shown in Figure 6.12 and Gaussian convolution is used to smooth occupied probability around hit points.

Given multiple submaps at the corresponding keyframe pose $\mathcal{X}, \mathcal{M}$, we intend

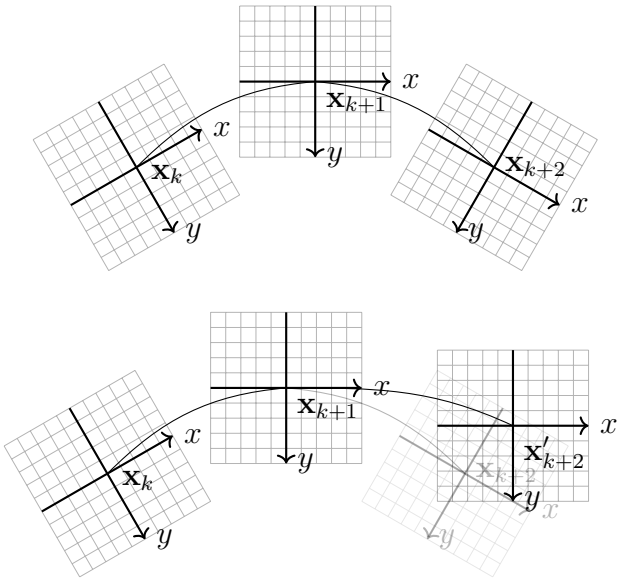


Figure 6.13: The occupancy map consists of submaps located at every keyframe. Once the pose of one keyframe (bottom right) is changed, its log-odds are corrected efficiently.

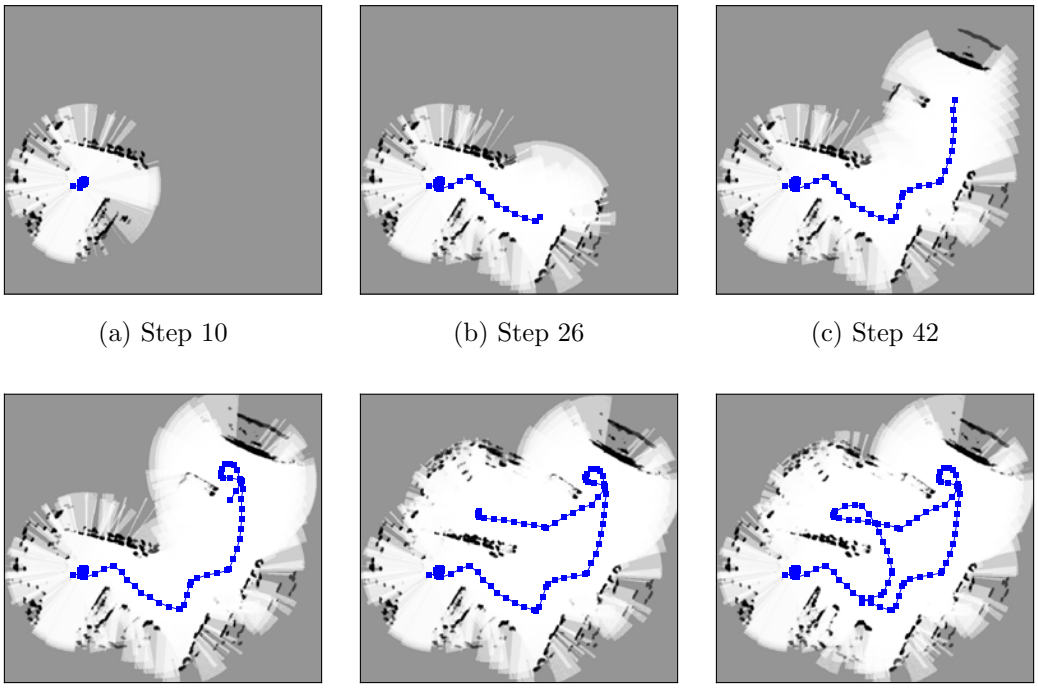
to output an occupancy map $\{m_i\}$ at cells in the global frame. The occupancy probability integrates all measurements at different poses, which can be incrementally updated through the Bayes filter as

$$l(\mathbf{m}_i) = \sum_{\mathcal{S}^k \in \mathcal{M}} l^k(\mathcal{T}_{kg}\mathbf{m}_i) \tag{6.24}$$

where $\mathcal{T}_{kg}\mathbf{m}_i$ represents the action of transforming global grid cells to a local frame. We ignore the prior in the summation as by default we assume an uninformative prior for the occupancy map $p(\mathbf{m}_i = 1) = 0.5$. Therefore, the resulting occupancy log-odds is simply the summation of lod-odds in individual submaps.

Updating A Submap

Adding non-sequential factors results in a constant change of pose history. We will update the contribution of the k-th keyframe when the change including translation



(a) Step 10 (b) Step 26 (c) Step 42

(d) Step 58 (e) Step 74 (f) Step 90

Figure 6.14: Underwater occupancy grid mapping.

and rotation exceeds a specified threshold. If computation resource permits, the maximum shift due to pose change shouldn't be larger than 1 grid cell.

Because the clamping rule [25] is not used in merging submaps, the update due to pose change can be implemented efficiently by removing the log-odds values and adding a new update,

$$l'(\mathbf{m}_i) = l(\mathbf{m}_i) - \underbrace{l^k(\mathcal{T}_{kg}\mathbf{m}_i)}_{\text{remove old update}} + \underbrace{l^k(\mathcal{T}'_{kg}\mathbf{m}_i)}_{\text{add new update}} . \tag{6.25}$$

In the equation above, the local submap represented by log-odds values doesn't need to be recalculated.

In practice, the submap in every keyframe stores the indices of its associated grid cells in the gloabal frame. During updating, we simply substract log-odds val-

ues from old indices and add new ones while updating new indices. Examples of occupancy mapping are shown in Figure 6.14.

6.2 Experiments and Results

To validate the proposed exploration algorithm in an unknown, complex underwater environment, simulated and real experiments are carried out. We first present real experiments and afterwards we will show how the simulation is realized using manually collected data. Introduction of the vehicle and sensors used in the experiments are skipped as they remain the same as in Chapter 3.

Besides nearest frontier (NF) and next-best-view (NBV) algorithms, in this chapter we also compare EM with a heuristic approach (Heuristic). It switches between two modes: if current pose uncertainty is smaller than a specified threshold, NBV is used to achieve rapid exploration; otherwise it will seek a revisitation location to reduce pose uncertainty. The metric of pose uncertainty is given by D-Opt (assuming 2D pose) $|\Sigma_{\mathbf{x}}|^{\frac{1}{3}}$.

6.2.1 Real Experiments

The real experiments were carried out in a marina at the United States Merchant Marine Academy (USMMA)[2]. An overview of the experiment setup is shown in Figure 6.15. Prior to launching the vehicle, it is manually commanded to a starting location facing upward (x direction in the figure). The goal is to explore the map within a bounding box with dimension 85 m $\times$ 60 m. Three runs were performed for each algorithm and results are shown in Figures 6.16 6.17 and 6.18.

In Figure 6.16, the estimated map represented by a time-colored point cloud, trajectory and loop closure measurements are visualized, overlaid with satellite im-

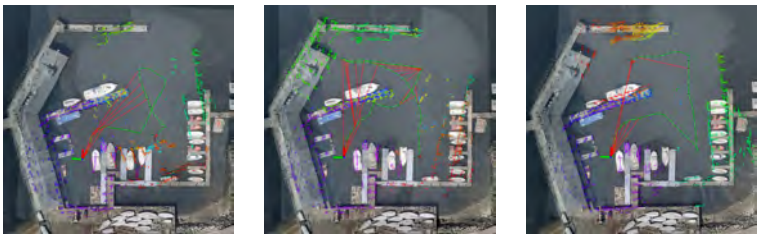
[2]https://www.usmma.edu/

Figure 6.15: Setup of real experiments at USMMA.

ages. It's worth noting that the postions of satellite images are manually adjusted and it only servers as a qualitative result to show the discrepancy between ground truth and the estimated map. Firstly, it's evident that both NF and NBV take actions that favored exploration, resulting in trajectories expanding to the top-right corner of the map quickly. As a result, loop closures are added later than EM or Heuristic and the loop closure isn't intentional. The map quality with regard to NBV is the worst among four algorithms; the final map using NF looks good but the large deviation from this final estimate during exploration is not corrected until ending the session. By contrast, the uncertainty-aware exploration algorithms, EM and Heuristic, ended up with dense loop closure constraints. Different from Heuristic, which seeks revisitation only at highly uncertain states, EM takes into account map uncertainty constantly and poses are intertwined through the entire trajectory.

Due to the lack of ground truth information and a limited number of repeated experiments, exploration performance, evaluated with respect to map coverage and

(a) Run 1 using NF (b) Run 2 using NF (c) Run 3 using NF

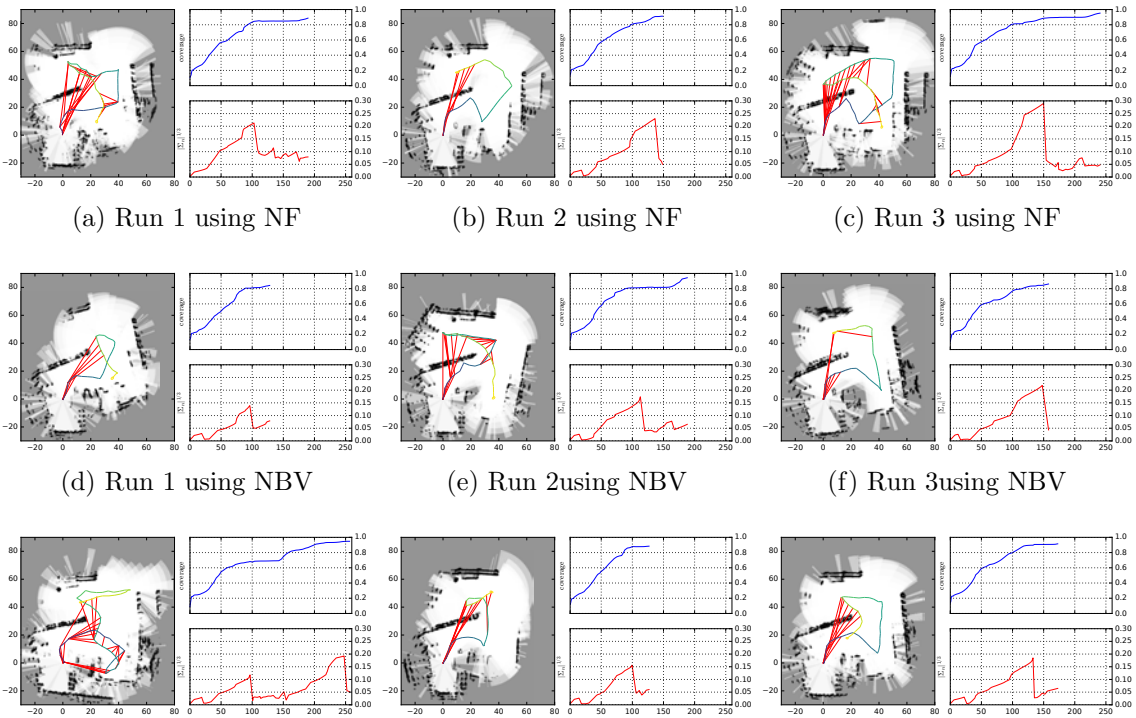


(d) Run 1 using NBV (e) Run 2 using NBV (f) Run 3 using NBV

(g) Run 1 using Heurisitic (h) Run 2 using Heurisitic (i) Run 3 using Heurisitic

(j) Run 1 using EM (k) Run 2 using EM (l) Run 3 using EM

Figure 6.16: Results overlaid with satellite imagery.



(a) Run 1 using NF (b) Run 2 using NF (c) Run 3 using NF

(d) Run 1 using NBV (e) Run 2using NBV (f) Run 3using NBV

(g) Run 1 using Heuristic (h) Run 2 using Heuristic (i) Run 3 using Heuristic

Figure 6.17: Three runs using NF, NBV and Heuristic in real environment.

pose uncertainty plotted against traveled distance, is presented to better understand different behaviors. Four intermittent steps and final steps are plotted in Figure 6.18 and to save space only final states are plotted for NF, NBV and Heuristic as in Figure 6.17. We can see from the comparison that EM keeps maintaining low uncertainty through three trials while the map coverage speed is similar to information-theoretic approaches. The revisitation behavior of Heuristic can be observed clearly in run 1, where at distance 100 the robot intentionally closed the cloop to reduce estimation uncertainty.

6.2.2 Simulated Experiments

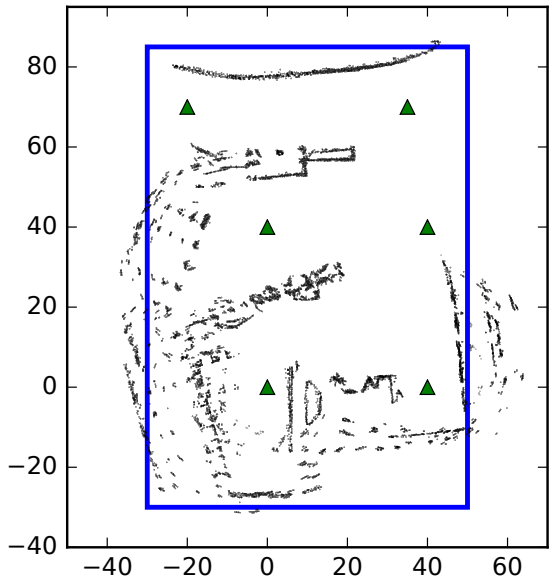


Figure 6.19: Simulation environments with six starting locations.

As there is no ground truth information in our real-world experiments, we propose to use a simulation environment to validate our proposed algorithm. We create the simulated map using manually collected data in the same environment as that in the real experiment. More specifically, the simulated map is represented by a point cloud that is obtained from the SLAM framework described above. The simulation map is shown in Figure 6.19. During simulated exploration, feature points that are within sensor field of view are sampled and Gaussian noise is added. We run 10 trials at 6 different starting locations (denoted as green triangles in Figure 6.19) for every algorithm. Mapping and localization errors are reported with ground truth information. The mapping error is calculated as follows: for every estimated feature point, we compute its distance to the nearest ground truth point.

Simulation results are shown in Figure 6.20 and detailed metrics are also presented in Tables6.1, 6.3, 6.4, 6.2. For every metric, average value and 95 % confidence

alg/distance	0	50	100	150	200	250	300	350	400
EM	0.10	0.34	0.30	0.32	0.33	0.31	0.31	0.29	0.28
NF	0.10	0.33	0.44	0.43	0.46	0.45	0.40	0.39	0.40
NBV	0.10	0.35	0.53	0.57	0.53	0.45	0.36	0.36	0.35
Heuristic	0.10	0.34	0.32	0.30	0.31	0.30	0.31	0.28	0.31

Table 6.1: Pose uncertainty in the simulated environment.

alg/distance	0	50	100	150	200	250	300	350	400
EM	0.75	1.72	1.83	1.89	1.90	1.93	1.93	1.96	1.97
NF	0.75	1.79	1.94	1.97	2.06	2.07	2.08	2.07	2.08
NBV	0.75	1.73	1.90	1.99	2.07	2.05	2.05	2.05	2.05
Heuristic	0.75	1.72	1.84	1.88	1.91	1.94	1.95	1.96	1.98

Table 6.2: Pose error in the simulated environment.

interval are visualized. As expected, NBV achieves the most efficient exploration but at the expense of the highest pose uncertainty. Pose uncertainty of Heuristic and EM is significantly lower than that in NF and NBV. It's worth mentioning that parameters in Heuristic are tuned to match the same uncertainty-awareness performance as in EM and parameters in EM are manually adjusted for this specific environment. While Heuristic and EM have similar pose uncertainty and trajectory error, Heuristic falls behind EM in terms of map coverage, which is more noticable in Table 6.3.

In Figure 6.21, one run using each algorithm is presented with four metrics with respect to traveled distance. The same phenomenon, a dense interconnection between poses, can be observed in Heuristic and EM. In contrast, greedy algorithms (NF and NBV) take the almost the same path to the top right corner in the beginning, which leads to growing pose uncertainty. Although the pose uncertainty is ultimately reduced, map accuracy is impacted as demonstrated in Figure 6.20(b) and Table 6.4.

alg/distance	50%	60%	70%	80%	90%
EM	13.79	39.53	80.92	157.15	303.92
NF	14.90	43.63	88.38	172.30	268.67
NBV	13.90	40.55	70.36	117.93	234.01
Heuristic	14.23	41.41	80.12	186.63	396.52

Table 6.3: Map coverage in the simulated environment.

alg/distance	0	50	100	150	200	250	300	350	400
EM	0.56	0.90	0.95	0.97	1.00	1.03	1.05	1.06	1.05
NF	0.60	0.97	1.03	1.08	1.13	1.15	1.12	1.13	1.13
NBV	0.58	0.87	1.01	1.08	1.12	1.12	1.12	1.12	1.12
Heuristic	0.58	0.90	0.94	0.98	0.99	1.01	1.02	1.03	1.03

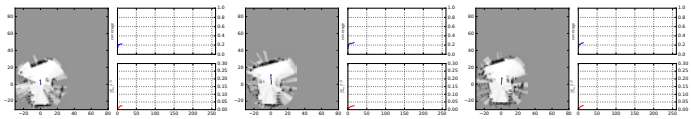
Table 6.4: Map error in simulated environment.

6.3 Conclusions

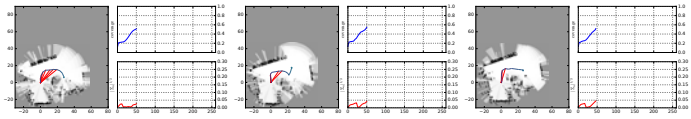
To conclude, a robust keyframe-based SLAM framework is designed and validated in real-world experiments. Factors are categorized into sequential and non-sequential (loop closure) constraints, both of which are obtained from scan matching using feature points extracted using a CFAR detector on raw sonar images. To mitigate the local minima issue in ICP, we add an initialization step prior to ICP to align two point clouds based on set consensus. We adopt pairwise consistent measurement set maximization to reject outliers in loop closure measurements. Given an estimated trajectory, a submap-based occupancy map is built and efficiently corrected upon loop closure.

We compare EM with three oether algorithms and results from both real and simulated experiments demonstrate that EM and Heuristic are capable of revisiting and closing the loop intentionally, thus having the lowest pose uncertainty and map accuracy. Compared to Heuristic, the utility function in EM constantly takes into account pose uncertainty via virtual landmarks. Periodical revisitation via unnecessary

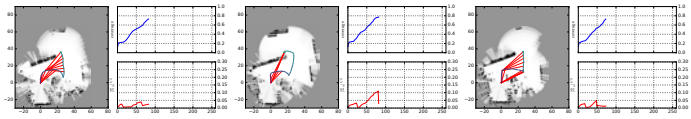
detours is avoided, which results in faster map coverage.



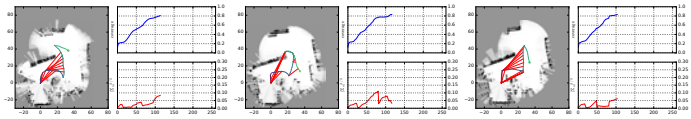
(a) Run 1: step 10 (b) Run 2: step 10 (c) Run 3: step 10



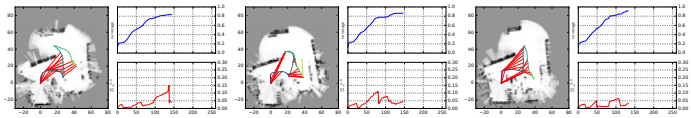
(d) Run 1: step 20 (e) Run 2: step 20 (f) Run 3: step 20



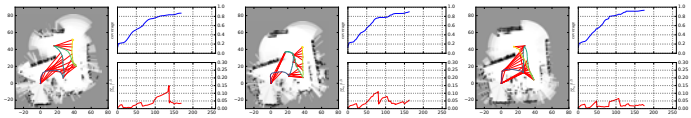
(g) Run 1: step 30 (h) Run 2: step 30 (i) Run 3: step 30



(j) Run 1: step 40 (k) Run 2: step 40 (l) Run 3: step 40

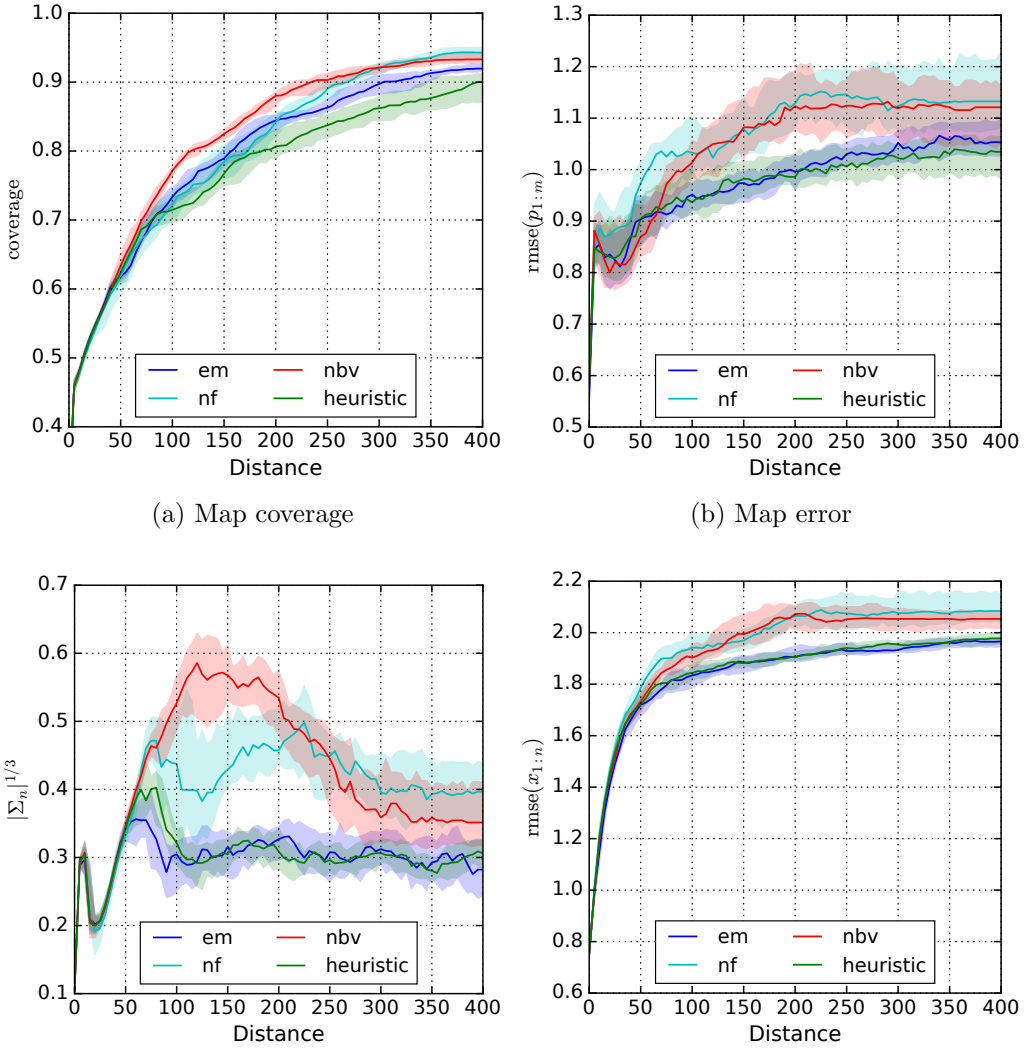


(m) Run 1: step 50 (n) Run 2: step 50 (o) Run 3: step 50



(p) Run 1: step 56 (q) Run 2: step 54 (r) Run 3: step 60

Figure 6.18: Three runs using EM in real environment.



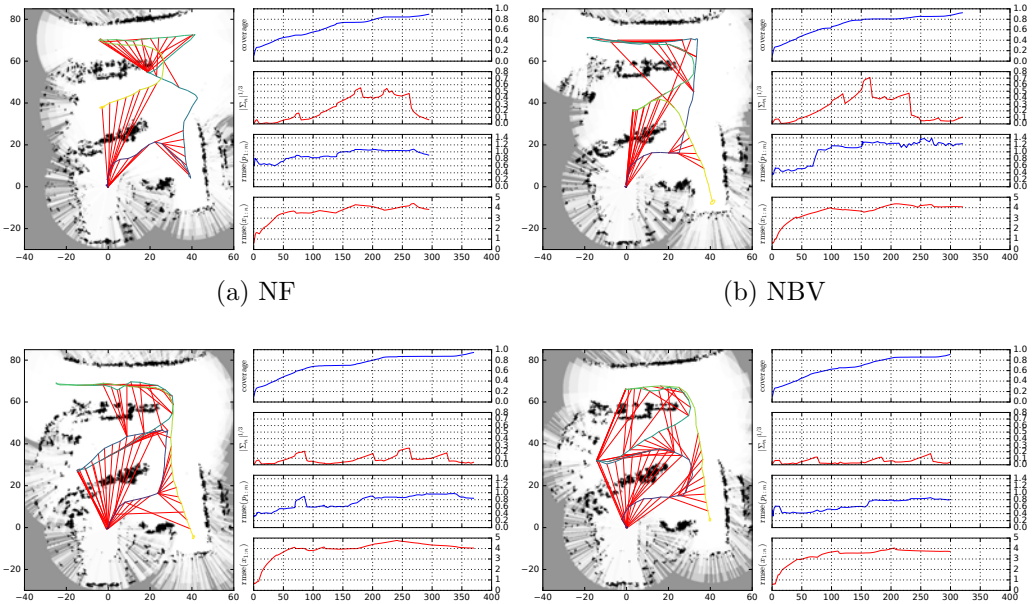
(a) Map coverage

(b) Map error

(c) Pose uncertainty

(d) Trajectory error

Figure 6.20: Exploration performance in the simulation.



(a) NF (b) NBV

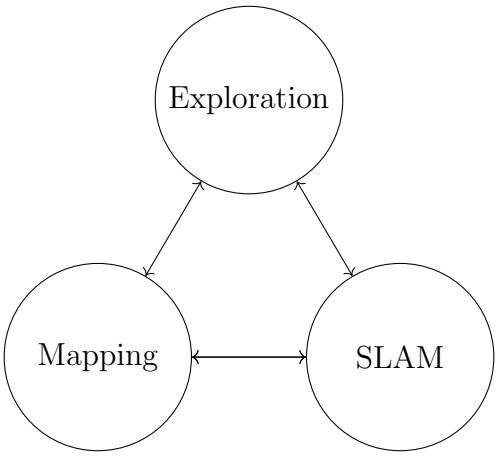
(c) Heuristic (d) EM

Figure 6.21: Simulation examples using different algorithms.

Chapter 7

Conclusions and Future Work

The objective of this research work is to achieve robust autonomy in underwater environments. To safely navigate in cluttered environments, an accurate and dense **map**, which is required to be constructed efficiently, is essential for path planning or obstacle avoidance. Robust state estimation relying on **SLAM** helps the robot localize itself within the map. **Exploration** algorithms enable the robot to autonomously inspect an unknown environment without human intervention. An uncertainty-aware exploration strategy will result in accurate mapping and state estimation. We have addressed these three major problems and proposed improvements to existing works specifically for robots navigating in underwater environments with sparse measurements and highly uncertain state estimation.



Mapping. We presented an improved formulation of Gaussian process occupancy mapping using nested BCMs and test-data octrees. Equipped with these time-saving approximations, the computational complexity of GPs has been reduced to a level

where it can be used for real-time 3D mapping, while retaining high-quality performance as a classifier. Next we offer two improvements for underwater feature-based SLAM with imaging sonar, for terrain mapping. First, data association is performed via optical flow tracking, which is more robust to noise and the absence of elevation angle. Second, a degenerate system is partly solved by adding terrain constraints connecting feature pairs, limiting their elevation values to be similar.

SLAM. An approach was presented for 3D mapping and localization for an underwater robot in cluttered, disturbance-filled shallow-water environments equipped with a single-beam scanning sonar. To overcome the limitations of using registration-based SLAM in the absence of inertial and odometry measurements, we adopt a feature-based, incremental smoothing and mapping solution, where point features are extracted from individually thresholded (via DBSCAN), clustered (via OPTICS) sonar range returns using the minimum covariance determinant. A factor graph is built using measurement and process constraints, and loop- closures are identified via iterative JCBB data association. To improve the data association robustness, we further propose a multiple hypothesis data association framework for SLAM in ambiguous environments. Multiple trajectory and map tracks are maintained using a series of association hypotheses generated from JCBB. Efficient hypothesis management is organized by limiting hypothesis generation and pruning unlikely tracks, which is enabled by providing two orderings.

Exploration. We propose the concept of virtual landmarks, which are latent variables representing the possible locations of actual landmarks in the environment. We presented a novel utility function for the autonomous exploration problem in feature-based maps, which essentially computes the covariance criteria at virtual landmarks.

The direct modeling of landmarks potentially observed in the future enables more accurate mapping and also a comparable exploration rate with respect to traditional methods. An improved version of EM-exploration that doesn't depend on feature-based SLAM is validated on an UGV and more importantly on an ROV. The application of real-time exploration is attributed to a robust SLAM framework using 2D imaging sonar.

7.1 Future Works

Figure 7.1: Dual-sonar configuration [1].

In the seciton, we present a few potential improvements in mapping, SLAM and exploration that could be done in the future. Through the entire work, we have focused on experiments using only one sonar (either mechanical scanning sonar or 2D imaging sonar) and the proposed underwater SLAM assumes a planar motion during exploration. However the assumption doesn't hold well in practice and non-zero pitch angle is expected. What's more, the unknown elevation angle is ignored in scan matching. Therefore, a scan matching method that operates in 3D space is necessary for better accuracy. Dual-sonar configuration has been also used in underwater surverying [1] as shown in Figure7.1. It can signiciantly increase the field

of view in 3D space and 3D measurements are of great benefit to mapping and SLAM.

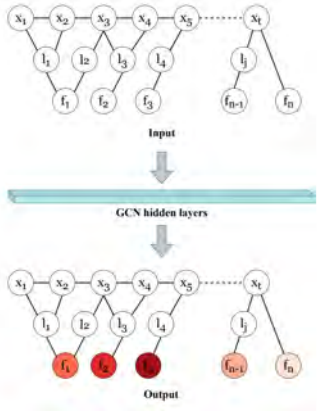
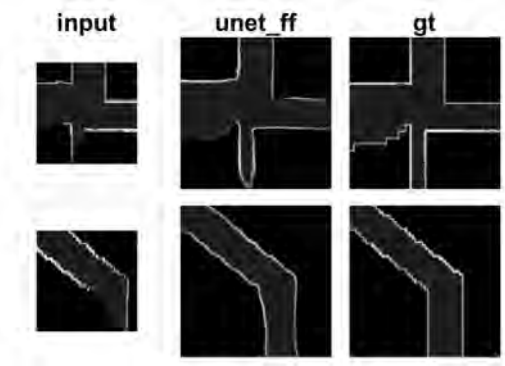


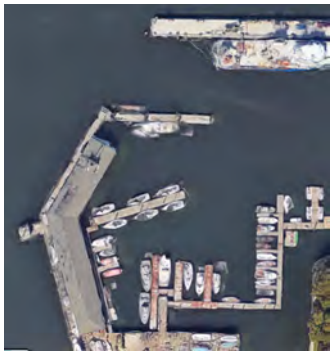
Figure 7.2: Usage of a Graph Convolutional Network (GCN) to predict exploration target [2].

Recently, deep learning has gained popularity in robotics. A novel algorithm that uses graph convolutional network (GCN) to improve the efficiency of EM-exploration has been proposed [2]. The cubic computational complexity is reduced to constant time. A further development lies in the application of GCN in reasoning about exploration target using pose-SLAM. As the key novelty of EM-exploration is the utility function that combines uncertainty and map coverage, we may regard the utility as a function of current states and derive the utility function, or reward function, from inverse reinforcement learning [106].

In EM-exploraiton, we propose the concept of virtual landmarks; however their locations are conservatively assumed to be consistent with cells in occupancy map. If we are able to predict the most likely virtual landmarks, candidate paths could be planed to effectively avoid obstacles and achieve better feature coverage. Two possible ways to predict virtual landmarks are demonstrated in Figure 7.3. Deep learning can be used to predict unobserved region given current surroundings [107]. Or a prior map could be provided from sources such as satellite images or previous inspection mission.



(a) Prediction of occupancy map[107].



(b) A prior map from satellite image.

Figure 7.3: Two ways to improve virtual landmark prediction.

Bibliography

[1] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 33, no. 7, pp. 994–1012, 2016.

[2] F. Chen, J. Wang, T. Shan, and B. Englot, "Autonomous exploration under uncertainty via graph convolutional networks," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2019.

[3] D. Ribas, P. Ridao, J. D. Tardós, and J. Neira, "Underwater SLAM in man-made structured environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 898–921, 2008.

[4] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[5] J. D. Hernández, E. Vidal, M. Moll, N. Palomeras, M. Carreras, and L. E. Kavraki, "Online motion planning for unexplored underwater environments using autonomous underwater vehicles," *Journal of Field Robotics*, vol. 36, no. 2, pp. 370–396, 2019.

[6] E. Vidal, J. D. Hernández, K. Istenič, and M. Carreras, "Online view planning for inspecting unexplored underwater structures," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1436–1443, July 2017.

[7] D. Ribas, P. Ridao, J. Neira, and J. D. Tardos, "SLAM using an imaging sonar for partially structured underwater environments," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5040–5045, Oct 2006.

[8] D. Ribas, P. Ridao, J. Domingo Tardos, and J. Neira, "Underwater SLAM in a marina environment," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1455–1460, Oct 2007.

[9] H. Johannsson, M. Kaess, B. Englot, F. Hover, and J. Leonard, "Imaging sonar-aided navigation for autonomous underwater harbor surveillance," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4396–4403, IEEE, 2010.

[10] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-Graph SLAM using forward-looking sonar," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2330–2337, July 2018.

[11] J. D. Hernández, M. Moll, E. Vidal, M. Carreras, and L. E. Kavraki, "Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1313–1320, Oct 2016.

[12] M. Carreras, J. D. Hernández, E. Vidal, N. Palomeras, and P. Ridao, "Online motion planning for underwater inspection," in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pp. 336–341, Nov 2016.

[13] E. Pairet, J. D. Hernández, M. Lahijanian, and M. Carreras, "Uncertainty-based online mapping and motion planning for marine robotics guidance," in

2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2367–2374, Oct 2018.

[14] J. Wang and B. Englot, "Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1003–1010, May 2016.

[15] J. Wang, T. Shan, and B. Englot, "Underwater terrain reconstruction from forward-looking sonar imagery," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3471–3477, May 2019.

[16] J. Wang, S. Bai, and B. Englot, "Underwater localization and 3D mapping of submerged structures with a single-beam scanning sonar," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4898–4905, May 2017.

[17] J. Wang and B. Englot, "Robust exploration with multiple hypothesis data association," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3537–3544, 2018.

[18] J. Wang and B. Englot, "Autonomous exploration with expectation-maximization," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2017.

[19] J. Wang, T. Shan, and B. Englot, "Virtual maps for autonomous exploration with pose SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, May 2019.

[20] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, p. 1–139, August 2017.

[21] "g2o." `https://github.com/RainerKuemmerle/g2o`.

[22] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, IEEE, 2011.

[23] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[24] "GTSAM." `https://bitbucket.org/gtborg/gtsam`.

[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, Apr 2013.

[26] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[27] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *2012 IEEE International Conference on Robotics and Automation*, pp. 9–15, May 2012.

[28] S. O'Callaghan, F. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using Gaussian processes," in *2009 IEEE International Conference on Robotics and Automation*, pp. 1054–1060, IEEE, 2009.

[29] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[30] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, pp. 111–127, Sep 2003.

[31] M. Veeck and W. Veeck, "Learning polyline maps from range scan data acquired with mobile robots," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, pp. 1065–1070 vol.2, Sep. 2004.

[32] M. A. Paskin and S. Thrun, "Robotic mapping with polygonal random fields," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence* (F. Bacchus and T. Jaakkola, eds.), AUAU Press, Arlington, Virginia, July 2005.

[33] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[34] K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3D Hilbert maps," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1011–1018, May 2016.

[35] K. Doherty, J. Wang, and B. Englot, "Bayesian generalized kernel inference for occupancy map prediction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3118–3124, May 2017.

[36] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, July 1997.

[37] B. J. Julian, S. Karaman, and D. Rus, "On mutual information-based control of range sensing robots for mapping applications," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5156–5163, Nov 2013.

[38] S. Bai, J. Wang, K. Doherty, and B. Englot, "Inference-enabled information-theoretic exploration of continuous action spaces," in *International Symposium on Robotics Research (ISRR)*, 2015.

[39] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with Bayesian optimization," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1816–1822, Oct 2016.

[40] S. Bai, F. Chen, and B. Englot, "Toward autonomous mapping and exploration for mobile robots through deep supervised learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2379–2384, Sep. 2017.

[41] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4791–4798, May 2015.

[42] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Y. Goldberg, P. Abbeel, N. Michael, and R. V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3D mapping," in *Robotics: Science and Systems*, 2015.

[43] M. G. Jadidi, J. V. Miró, R. Valencia, and J. Andrade-Cetto, "Exploration on continuous Gaussian process frontier maps," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6077–6082, May 2014.

[44] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650–668, 1999.

[45] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty.," in *Robotics: Science and Systems (RSS)*, 2007.

[46] R. Sim and N. Roy, "Global A-optimal robot exploration in SLAM," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 661–666, April 2005.

[47] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1462–1468, May 2016.

[48] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4568–4575, May 2017.

[49] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 540–545 vol.1, Sep. 2002.

[50] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 534–539 vol.1, Sep. 2002.

[51] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 03–21, 2007.

[52] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 33, no. 7, pp. 994–1012, 2016.

[53] M. D. Aykin and S. Negahdaripour, "Forward-look 2-D sonar image formation and 3-D reconstruction," in *2013 OCEANS-San Diego*, pp. 1–10, IEEE, 2013.

[54] T. A. Huang and M. Kaess, "Towards acoustic structure from motion for imaging sonar," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 758–765, IEEE, 2015.

[55] Y. Yang and G. Huang, "Acoustic-inertial underwater navigation.," in *2017 IEEE International Conference on Robotics and Automation*, pp. 4927–4933, 2017.

[56] T. A. Huang and M. Kaess, "Incremental data association for acoustic structure from motion," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1334–1341, IEEE, 2016.

[57] E. Westman, A. Hinduja, and M. Kaess, "Feature-based SLAM for imaging sonar with under-constrained landmarks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, IEEE, 2018.

[58] C. E. Rasmussen, "Gaussian processes for machine learning," MIT Press, 2006.

[59] A. Melkumyan and F. Ramos, "A sparse covariance function for exact Gaussian process inference in large datasets," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, (San Francisco, CA, USA), pp. 1936–1942, Morgan Kaufmann Publishers Inc., 2009.

[60] S. Kim and J. Kim, "GPmap: A unified framework for robotic mapping based on sparse Gaussian processes," in *Field and service robotics*, pp. 319–332, Springer, 2015.

[61] V. Tresp, "A bayesian committee machine," *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[62] S. Kim and J. Kim, "Recursive Bayesian updates for occupancy mapping and surface reconstruction," 2014.

[63] A. Schwaighofer and V. Tresp, "Transductive and inductive methods for approximate Gaussian process regression," in *Advances in Neural Information Processing Systems*, pp. 977–984, 2003.

[64] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, May 2011.

[65] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *British Machine Vision Conference (BMVC)*, 2013.

[66] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (DARPA)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, pp. 121–130, April 1981.

[67] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, "Gaussian process modeling of large scale terrain," in *2009 IEEE International Conference on Robotics and Automation*, pp. 1047–1053, May 2009.

[68] D. Moore and S. J. Russell, "Gaussian process random fields," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 3357–3365, Curran Associates, Inc., 2015.

[69] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Transactions on Robotics*, vol. 30, pp. 1371–1385, Dec 2014.

[70] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, May 1968.

[71] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231, AAAI Press, 1996.

[72] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, (New York, NY, USA), pp. 49–60, ACM, 1999.

[73] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, "Automatic extraction of clusters from hierarchical clustering representations," in *Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '03, (Berlin, Heidelberg), pp. 75–87, Springer-Verlag, 2003.

[74] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[75] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2001.

[76] W. E. L. Grimson and D. P. Huttenlocher, *Object recognition by computer: the role of geometric constraints*. MIT Press.

[77] Y. Li and E. B. Olson, "IPJC: The incremental posterior joint compatibility test for fast feature cloud matching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3467–3474, IEEE, 2012.

[78] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[79] D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard, "Towards lazy data association in SLAM," in *Robotics Research. The Eleventh International Symposium*, pp. 421–431, Springer, 2005.

[80] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[81] E. B. Olson, *Robust and efficient robotic mapping*. PhD Thesis, Massachusetts Institute of Technology, 2008.

[82] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a bayesian multiple hypothesis approach," *Artificial Intelligence*, vol. 66, no. 2, pp. 311–344, 1994.

[83] H. A. Fayed and A. F. Atiya, "A mixed breadth-depth first strategy for the branch and bound tree of Euclidean k-center problems," *Computational Optimization and Applications*, vol. 54, no. 3, pp. 675–703, 2013.

[84] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Robotics: Science and Systems (RSS)*, pp. 65–72, 2005.

[85] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2080–2087, May 2012.

[86] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

[87] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik, "Fast covariance recovery in incremental nonlinear least square solvers," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4636–4643, May 2015.

[88] H. Li, F. Nashashibi, and M. Yang, "Split covariance intersection filter: Theory and its application to vehicle localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1860–1871, Dec 2013.

[89] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *2017 IEEE*

International Conference on Robotics and Automation (ICRA), pp. 5266–5272, IEEE, 2017.

[90] R. Dubé, A. Cramariuc, D. Dugas, J. I. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D segment mapping using data-driven descriptors," *ArXiv*, vol. abs/1804.09557, 2018.

[91] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[92] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.

[93] H. Carrillo, Y. Latif, M. L. Rodriguez-Arevalo, J. Neira, and J. A. Castellanos, "On the monotonicity of optimality criteria during exploration in active SLAM," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1476–1483, IEEE, 2015.

[94] J.-L. Blanco, J.-A. Fernandez-Madrigal, and J. González, "A novel measure of uncertainty for mobile robot slam with Rao–Blackwellized particle filters," *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 73–89, 2008.

[95] P. P. Gandhi and S. A. Kassam, "Analysis of CFAR processors in nonhomogeneous background," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, pp. 427–445, July 1988.

[96] V. G. Hansen and J. H. Sawyers, "Detectability loss due to "greatest of" selection in a cell-averaging CFAR," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-16, pp. 115–118, Jan 1980.

[97] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, pp. 608–621, July 1983.

[98] M. Richards, *Fundamentals Of Radar Signal Processing*. McGraw-Hill Education (India) Pvt Limited, 2005.

[99] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[100] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4387–4393, IEEE, 2009.

[101] S. C. Endres, C. Sandrock, and W. W. Focke, "A simplicial homology algorithm for Lipschitz optimisation," *Journal of Global Optimization*, vol. 72, no. 2, pp. 181–217, 2018.

[102] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proceedings 2007 IEEE international conference on robotics and automation*, pp. 3167–3172, IEEE, 2007.

[103] S. Bonnabel, M. Barczyk, and F. Goulette, "On the covariance of ICP-based scan-matching techniques," in *2016 American Control Conference (ACC)*, pp. 5498–5503, IEEE, 2016.

[104] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2916–2923, IEEE, 2018.

[105] B.-J. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3d environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2175–2182, IEEE, 2018.

[106] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.,"

[107] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5453–5459, May 2019.

Vita

Jinkun Wang

Education

PH.D. Candidate in Mechanical Engineering — September 2014 - November 2019
Stevens Institute of Technology, Hoboken, NJ

B.E. in Mechanical Engineering — September 2010 - June 2014
University of Science and Technology, Hefei, China

Publications

J. Wang, T. Shan, and B. Englot, "Virtual Maps for Autonomous Exploration with Pose SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 4899–4906, November 2019.

J. Wang, T. Shan, M. Chandrasekaran, T. Osedach, and B. Englot, "Deep Learning for Detection and Tracking of Underwater Pipelines using Multibeam Imaging Sonar," *ICRA 2019 Workshop on Underwater Robotics Perception,* May 2019.

J. Wang, T. Shan, and B. Englot, "Underwater Terrain Reconstruction from Forward-Looking Sonar Imagery," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 3471–3477, May 2019.

J. Martin, **J. Wang**, and B. Englot, "Sparse Gaussian Process Temporal Difference Learning for Marine Robot Navigation," *2nd Annual Conference on Robot Learning, Proceedings of Machine Learning Research,* vol. 87, pp. 179–189, October 2018.

T. Shan, K. Doherty, **J. Wang**, and B. Englot, "Bayesian Generalized Kernel Inference for Terrain Traversability Mapping," *2nd Annual Conference on Robot Learning, Proceedings of Machine Learning Research,* vol. 87, pp. 829–838, October 2018.

J. Wang and B. Englot, "Robust Exploration with Multiple Hypothesis Data Association," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 3537–3544, October 2018.

J. Wang and B. Englot, "Autonomous Exploration with Expectation Maximization," *Proceedings of the 18th International Symposium on Robotics Research,* 16 pp., December 2017.

J. Wang, S. Bai, and B. Englot, "Underwater Localization and 3D Mapping of Submerged Structures with a Single-Beam Scanning Sonar," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 4898-4905, May 2017.

K. Doherty, **J. Wang**, and B. Englot, "Bayesian Generalized Kernel Inference for Occupancy Map Prediction," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 3118-3124, May 2017.

S. Bai, **J. Wang**, F. Chen, and B. Englot, "Information-Theoretic Exploration with Bayesian Optimization," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 1816-1822, October 2016.

J. Wang and B. Englot, "Fast, Accurate Gaussian Process Occupancy Maps via Test-Data Octrees and Nested Bayesian Fusion," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1003-1010, May 2016.

K. Doherty, **J. Wang**, and B. Englot, "Probabilistic Map Fusion for Fast, Incremental Occupancy Mapping with 3D Hilbert Maps," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1011-1018, May 2016.

S. Bai, **J. Wang**, K. Doherty, and B. Englot, "Inference-Enabled Information-Theoretic Exploration of Continuous Action Spaces," *Proceedings of the 17th International Symposium on Robotics Research,* 16 pp., September 2015.