INFERENCE WITH FACTOR GRAPHS FOR SINGLE AND MULTI-ROBOT PERCEPTION AND NAVIGATION

by

Yewei Huang

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

Yewei Huang, Candidate

ADVISORY COMMITTEE

Brendan Englot, Chairman Date

Enrique Dunn Date

Kishore Pochiraju Date

Long Wang Date

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2025

INFERENCE WITH FACTOR GRAPHS FOR SINGLE AND MULTI-ROBOT PERCEPTION AND NAVIGATION

ABSTRACT

Mobile robot perception and navigation have gained increasing attention in recent years, driven in part by the growing focus on autonomous driving. However, there are still challenges in both single and multi-robot scenarios, stemming from challenging environmental conditions, limited communication bandwidth, and the complexities of fostering efficient collaboration among neighboring robots. One valuable tool for addressing these challenges is the factor graph, a graph structure widely employed for modeling probabilistic inference problems. In this thesis, we transform common perception and navigation problems into factor graph optimization problems, offering innovative solutions that advance beyond the current state-of-the-art barriers in both single and multi-robot scenarios.

To address specific challenges in underwater environments, we introduce a Gaussian process motion planning algorithm tailored for unmanned underwater vehicles (UUVs) engaged in seafloor terrain following missions, which incorporates the influence of oceanic currents. Additionally, we present three distributed multi-robot simultaneous localization and mapping (SLAM) algorithms for range-sensing mobile robots. The first algorithm introduces a compact LiDAR descriptor, along with a two-stage global and local factor graph optimization approach. The second introduces a hierarchical scene graph for map data storage and utilizes scene graph matching for efficient inter-robot data association. The third is a distributed SLAM framework for underwater robot teams equipped with imaging sonar. This framework uses object graph matching for inter-robot data association and proposes a robust outlier de-

tection technique to enhance collaborative mapping in complex underwater environments. Furthermore, we propose an asynchronous autonomous exploration algorithm designed for multi-robot teams. This algorithm utilizes a virtual map and employs expectation-maximization (EM) to estimate the effectiveness of a robot's potential future actions.

Author: Yewei Huang

Advisor: Brendan Englot

Date: April 30, 2025

Department: Mechanical Engineering

Degree: Doctor of Philosophy

Acknowledgments

I am deeply grateful to all the individuals who have supported me throughout my journey to graduation. I would like to express my sincere appreciation to my advisor, Prof. Englot, whose guidance has been invaluable not only in my academic work but also as a role model in my life. I would like to sincerely thank my thesis committee members, Prof. Dunn, Prof. Pochiraju, and Prof. Wang, for their valuable suggestions and insights that have greatly helped improve my thesis.

I would also like to thank all the past and present members of RFAL, Dr. Shi Bai, Dr. Jinkun Wang, Dr. Tixiao Shan, Dr. Kevin Doherty, Dr. John Martin, Dr. Fanfei Chen, Dr. Jake McConnell, Dr. Erik Pearson, Paul Szenher, Ivana Collado-Gonzalez, Xi Lin, Kyle Hart, Yin Chen, Yaxuan Li, Noé Pigret and Razan Andigani for their assistance and for the many happy, unforgettable moments we've shared.

My gratitude extends to my family for their unwavering support throughout my studies. A special thanks to my husband, Yicheng; none of this would have been possible without your support and that of your family.

Finally, I would like to express my deepest respect and appreciation to all the authors who have made their code publicly available. Without their generosity, this thesis would not have been possible.

Funding Acknowledgments

This thesis was supported in part by the Defense Advanced Research Projects Agency (DARPA) Symbiotic Design for Cyber-Physical Systems (SDCPS) program, the Intelligence Advanced Research Projects Activity (IARPA) Walk-through Rendering from Images of Varying Altitude (WRIVA) program, and the Office of Naval Research (ONR) under grant N00014-21-1-2161.

Table of Contents

Abstract **iii**

Acknowledgments **v**

Funding Acknowledgments **vi**

List of Tables **x**

List of Figures **xii**

1 Introduction **1**

2 Background **5**

2.1 Gaussian Process Motion Planning (GPMP) 5

2.2 Multi-Robot Simultaneous Localization and Mapping (SLAM) 6

2.2.1 Multi-Robot SLAM with Scene Graphs 9

2.2.2 Multi-Robot SLAM for underwater environments 10

2.3 Multi-Robot Autonomous Exploration 12

3 Mission-oriented Gaussian Process Motion Planning **15**

3.1 Mission-oriented GPMP for UUVs over Current Flows 15

3.2 MGPMP Algorithm 16

3.2.1 Gaussian Prior and GP Constant Speed Prior 18

3.2.2 Obstacle Avoidance with Signed Distance Field 19

3.2.3 Seafloor Terrain Following 21

3.2.4 Robot Kinematics and the Influence of Current Flows 23

3.3 Experiments 24
3.3.1 Experimental Setup 24
3.3.2 Performance of Collision Avoidance 26
3.3.3 Performance of Seafloor Terrain Following 29
3.3.4 The Influence of Current Flows 32
3.4 Conclusions 35

4 Multi-Robot LiDAR Simultaneous Localization and Mapping 36
4.1 Distributed Multi-Robot SLAM with Two-Stage Global-Local Graph Optimization 36
4.2 DiSCo (Distributed Scan Context) - SLAM Algorithm 40
4.2.1 Incremental PCM 42
4.2.2 Two-Stage Global and Local Optimization 42
4.2.3 Message Passing Between Robots 43
4.3 SGM (Scene Graph Matching) - SLAM Algorithm 44
4.3.1 Scene Graph Construction 45
4.3.2 Scene Graph Matching 47
4.3.3 Transformation Estimation 49
4.3.4 Inter-robot Loop Closure 50
4.3.5 Communication Between Robots 51
4.4 Experiments 51
4.4.1 DiSCo-SLAM 51
4.4.2 SGM-SLAM 63
4.5 Conclusions 72

5 Multi-Robot Underwater Simultaneous Localization and Mapping 73
5.1 Local SLAM, Point-cloud Map and Object Map 74

5.1.1 Local SLAM 74
5.1.2 Object Graph Construction and Matching 76
5.1.3 Scan Registration 76
5.1.4 Group-wise Consistent Measurement Set Maximization 77
5.1.5 Communication 80
5.2 Experiments 80
5.2.1 Performance of Inter-Robot Loop Closure Detection 81
5.2.2 Performance of Inter-Robot PGO 84
5.2.3 Performance on the Real-world Dataset 86

6 Multi-Robot Exploration with Expectation-Maximization 89
6.1 Expectation-Maximization Exploration 89
6.2 Multi-Robot Expectation-Maximization Exploration Algorithm 92
6.2.1 Virtual Map 93
6.2.2 Uncertainty Propagation 94
6.2.3 Map Uncertainty Utility Computation 96
6.2.4 Complexity Analysis 97
6.3 Experiments 98
6.3.1 Experiments in 2D simulation environments 98
6.4 Conclusions 103

7 Conclusions and Future Work 104

Bibliography 106

Vita 119

List of Tables

3.1 Performance of Collision Avoidance 27

3.2 Performance of Seafloor Terrain Following 30

3.3 Runtime of MGPMP with and without currents 35

4.1 Root Mean Square Error (RMSE) w.r.t. GPS 56

4.2 RMSE w.r.t GPS for 40 trials 60

4.3 Relative pose estimation error at traj. connecting points 60

4.4 Data Sizes of Messages Sent (VLP-16) 62

4.5 Data Sizes of Messages Sent (VLP-64) 62

4.6 Processing Time for Each Algorithmic Step (ms) 63

4.7 **Success Rate** of transformation estimation for various graph sizes and amounts of overlap without semantic label error. 65

4.8 **Success Rate** of transformation estimation with various semantic label errors. 66

4.9 **Absolute Trajectory Error (ATE) [m]** in meters for the proposed method compared to state-of-art methods. 68

4.10 **Sizes of Perception Messages [KB]** for the proposed method. 71

5.1 **Runtime** for sonar scene descriptor-based inter-robot loop closure and global ICP registration in DRACo-SLAM [66] (referred to as DRACo1), and graph-based inter-robot loop closure detection and ICP registration in the proposed DRACo-SLAM2 (referred to as DRACo2) on both of our 3-robot fully simulated datasets. 84

5.2 **Mean runtime and mean number of inter-robot loop closure algorithm executions per time step** for DRACo1 and DRACo2 on both of our 3-robot fully simulated datasets. 85

5.3 **Absolute Trajectory Error (ATE)** in meters for the proposed method compared to the full Pose Graph Optimization (PGO) method on both of our 3-robot fully simulated datasets. 85

5.4 **Sizes of perception messages** for DRACo1 and DRACo2 on the USMMA real-world dataset (with simulated 3-robot comms.). 86

List of Figures

1.1 A classic factor graph representation with 4 factor nodes and 4 variables. 1

3.1 Factor graph structure of the proposed method. 17

3.2 A simple 2D example illustrating robot kinematics-based GP interpolation and the impact of water current velocity on obstacle avoidance. 21

3.3 An example showing how the obstacle avoidance cost and the seafloor following cost conflict with each other. 22

3.4 Performance of varied states of different methods on the Governors Island Datasets. 28

3.5 Performance of varied states of different methods on the Queen Elizabeth Islands Datasets. 28

3.6 3D and side view of the New York City (NYC) Lower Bay Dataset. 31

3.7 Plan view of the performance of varied states of different methods on the NYC Upper Bay Dataset. 32

3.8 The performance of varied states of different methods on the Queen Elizabeth Islands Dataset. 33

3.9 Performance with and without currents on the New York City (NYC) Upper Bay Dataset. 34

4.1 Differences between DGS and our proposed method. Green, cyan, and blue triangles indicate the poses of three different robots. 38

4.2 Overview of the architecture of DiSCo-SLAM, our proposed distributed multi-robot SLAM system, with robot α as the local robot. 40

4.3 **System Architecture.** The SGM-SLAM pipeline for each robot. 44

4.4 **The semantic SLAM result from a single robot to build a scene graph.** Keyframe poses are marked in yellow, odometry poses in green, and objects are represented by bounding boxes. 45

4.5 Segmented image with detected objects shown in different colors (left) and LiDAR scan projected onto the segmented image, with objects of interest marked by red rectangles (right). 46

4.6 A simple example illustrating edge-wise matching between two scene graphs. Edge lengths represent the weights, while node colors (green and blue) denote vertex types. 48

4.7 Our Jackal UGV instrumented with LiDAR, IMU, and a GPS used for ground truth. 52

4.8 Park dataset, at a location where there is overlap among all three robot trajectories. 54

4.9 Representative trajectory estimation results over the three-robot Park dataset, with different multi-robot SLAM configurations. 55

4.10 Optimization result on the KITTI 08 dataset. 57

4.11 Optimization result on the KITTI 00 dataset. 58

4.12 Optimization result on the Stevens campus dataset. 59

4.13 Our data-gathering platform consists of a Unitree robot dog with sensors mounted on the top. 63

4.14 An example illustrating a successful transformation estimation despite the presence of both position and semantic label errors. Objects in $\mathcal{G}_1$ are marked with red cuboids, and objects in $\mathcal{G}_2$ are marked with blue cuboids. The objects in $\mathcal{G}_2$ are transformed into the local coordinate system of $\mathcal{G}_1$ using the transformation estimation result. A table in $\mathcal{G}_1$ is misclassified as a sofa (highlighted by the black rectangle), yet the transformation estimation remains correct and serves as a sufficient initial guess. 65

4.15 SLAM trajectories from multiple robots aligned with pose estimates from COLMAP sparse reconstruction on outdoor (left) and indoor (right) datasets. 68

4.16 Top-down view (top) and side view (bottom) of the merged result from the two-robot dataset collected in an outdoor area of the SRI campus. 69

4.17 Top-down view (top) and side view (bottom) of the merged result from the three-robot dataset collected in an indoor area of the SRI campus. 70

5.1 **Overview of DRACo-SLAM2 Architecture.** Each robot's object map is clustered using DBSCAN. The local robot receives the neighboring robot's object map, aligns it to its local map using graph matching, and requests scans for ICP registration with the graph matching transformation as an initial guess. Inter-robot loop closures are then added to the pose graph for the two-step pose graph optimization (PGO). The local robot's trajectory and object vertices are shown in blue, while the neighboring robot's data is in green. Dashed lines indicate detected correspondences between objects in different maps. 73

5.2 **Graph matching-based ICP registration.** The source cloud (orange) is roughly transformed into the local robot coordinate system using the transformation estimated from object graph matching. It is then aligned with the target cloud (black) using ICP with a sliding window of size 3. The registered source cloud is shown in green. 77

5.3 **Motivation for use of GCM.** A series of three inter-robot loop closures, in the same region of the environment (from our USMMA dataset depicted in figure 5.7), are impacted by similar point cloud registration errors. Each erroneous registration result is shown in green, with ground truth shown in red. 78

5.4 **Illustration comparing PCM and the proposed GCM**. Circles represent pose estimations, arrows indicate measurements, and purple arrows highlight inter-robot loop closure measurements. 79

5.5 **Performance** of inter-robot loop closure detection and registration on both of our 3-robot fully simulated datasets. Precision (left) and the number of true positive loop closures detected (right) plotted against the overlap ratio parameter $r_{overlap}$, evaluated using various methods. 82

5.6 **Optimized trajectories** of different local robots using the proposed DRACo-SLAM2 on the fully simulated 3-robot airplane (top) and USMMA (bottom) datasets. 83

5.7 **Example DRACo-SLAM2 result with real sonar data.** Optimized trajectories and point clouds from three robots using the proposed DRACo-SLAM2 on a dataset collected at the U.S. Merchant Marine Academy, King's Point, NY, aligned with a satellite image. Inter-robot measurement constraints are shown in purple. 88

6.1 System Architecture. The pipeline of the proposed approach. 92

6.2 Problem setup. 93

6.3 EM-based uncertainty propagation with virtual observations. 95

6.4 50 trials of three robots navigating in 100m x 100m environments with 20 landmarks, each with a radius of 1m. 101

6.5 50 trials of three robots navigating in 200m x 200m environments with 20 landmarks, each with a radius of 1m. 101

6.6 Five robots navigating a 150m x 150m environment. 102

6.7 Three robots explore a 200m x 200m environment. 103

Chapter 1

Introduction

In recent years, mobile robots have gathered increased attention due to their great potential across various domains, including search-and-rescue, infrastructure inspection, household services, as well as logistical and transportation applications. For mobile robots, it's crucial to have the ability to sense the surrounding environment and make informed decisions incrementally towards a designated task. This dual capability, known as robot perception and navigation, encompasses the robot's ability to sense, understand, and interpret its surrounding environment through on-board sensors. Subsequently, it involves making movement-related decisions, both short-term and long-term, based on the information gathered from these sensors. A robust implementation of perception and navigation system is pivotal for the efficient and safe execution of robot tasks, including manipulation and human-robot interaction. These capabilities serve as prerequisites, enabling the robot to adapt its movements and actions to the unknown or partially known surrounding environments. In the context of multi-robot scenarios, additional challenges arise, such as the need for efficient task allocation and the management of limited communication bandwidth. These factors become crucial considerations in the development of perception and navigation systems for multi robot teams.

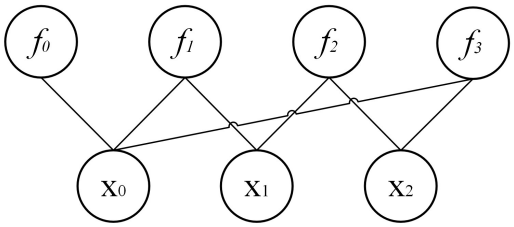


Figure 1.1: A classic factor graph representation with 4 factor nodes and 4 variables.

A factor graph is a bipartite graph consisting of variables and factors. Figure 1.1 gives a simple example of a factor graph corresponding to the factorization of a function $g(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$:

$$g(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2) = f_0(\mathbf{x}_0) f_1(\mathbf{x}_0, \mathbf{x}_1) f_2(\mathbf{x}_1, \mathbf{x}_2) f_3(\mathbf{x}_0, \mathbf{x}_2), \tag{1.0.1}$$

with each factor or variable represented by a node. Let $g(\cdot)$ be the joint probability distribution over random variables $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2\}$; equation 1.0.1 indicates the factorization of $g(\cdot)$ into a sum-product of a series of probability density functions $f_{(\cdot)}$. The graphical representation aligns well with the factorization nature inherent in some probabilistic models such as Markov process and, notably, exhibits efficiency in solving the maximum a posteriori problem.

In this thesis, we delve into the following facets of robot perception and navigation using the concept of factor graph: Motion Planning, Simultaneous Localization and Mapping (SLAM) and Autonomous Exploration. Assume a mobile robot whose trajectory is continuously sampled from a Gaussian process. The motion planning problem can be converted into an optimization problem framed as a factor graph. Here, the variable nodes correspond to the robot states, while the factor nodes, exclusively linked to the variable nodes, represent the cost functions for obstacle avoidance and specific motion planning tasks.

In real-world scenarios, sensor observations are often corrupted by noise, which can be addressed by modeling the observations as random variables. For the SLAM problem, we treat the states of the robot and landmarks as variable nodes, while sensor observations are regarded as factor nodes, and Maximum A Posteriori (MAP) state estimation can be formulated as an optimization problem on a factor graph. For individual robots, inertial and odometry sensing are helpful tools to support the

solution of this optimization problem, providing a foundation for the accurate handling of measurement constraints linking perceptual observations that are distant in time. However, handling such constraints becomes more challenging in multi-robot SLAM, where we often lack an initial guess for describing how sensor observations from different robots relate to one another. This problem can be addressed through a combination of improved factor graph optimization, and by adopting data-efficient descriptors that can be easily exchanged among robots to compare perceptual observations. Both avenues are considered in this thesis, and in our efforts to adopt data-efficient descriptors, we consider both sensor descriptors, which compactly describe a single perceptual observation, and graph-based descriptors, which describe the entirety of a robot's past observations. To the best of our knowledge, this thesis is the first work to apply graph-based descriptors to inter-robot place recognition in multi-robot SLAM.

We further consider the autonomous exploration problem by framing it as a future estimation problem constrained by a finite number of steps, tightly coupled with a SLAM factor graph. The effectiveness of prospective actions is evaluated by incorporating predictions of state and observations derived from these actions into the SLAM factor graph. The contributions of this thesis are as follows, accompanied by code that is publicly available:

- A factor graph based underwater motion planning framework that considers 3D seafloor terrain, current flows, and energy efficiency[42][1].
- A distributed multi-robot SLAM framework intended for real-time use with 3D LiDAR, and with a two-stage global-local factor graph optimization [40] [2].

[1] `https://github.com/RobustFieldAutonomyLab/Mission-Oriented-GP-Motion-Planning.git`

[2] `https://github.com/RobustFieldAutonomyLab/DiSCo-SLAM.git`

- A distributed multi-robot SLAM framework for robots equipped with LiDAR and camera sensors, which achieves a robust, general, and data-efficient SLAM method through scene graph matching. [44]
- An enhanced multi-robot SLAM system for underwater environments that achieves greater computational efficiency through object graph matching. [43]
- An asynchronous multi-robot exploration framework catering to both centralized and decentralized factor graph SLAM systems, taking into account efficient task allocation for exploration and addressing map uncertainty [41] [3].

[3] `https://github.com/RobustFieldAutonomyLab/Multi-Robot-EM-Exploration.git`

Chapter 2
Background

2.1 Gaussian Process Motion Planning (GPMP)

As an optimization-based planning method, Gaussian process motion planning (GPMP) [68], is a technique that models the set of states sampled from a smooth trajectory connecting the starting and ending goals as a Gaussian process, and formulates the motion planning problem as a nonlinear least squares problem. GPMP2 [22] formulates the Gaussian process motion planning problem over a factor graph [23], which is a graph representation of factorized functions of variables. The optimization problem is then solved using GTSAM [17], a toolkit for factor graph-based optimization problems. GPMP is well-suited for the motion planning of robots since it can produce efficient and safe paths for robots to follow. Moreover, GPMP is flexible and can be adapted to different environments. STEAP [69] employs GPMP for simultaneous localization and planning and tests with a ground robot. SCATE [56] applies a similar method to a spacecraft, demonstrating its applicability beyond terrestrial robots. In [67], GPMP is utilized to solve 2D path planning problems for an Unmanned Surface Vehicle (USV) aiming to avoid areas with high current flow velocities. Furthermore, GPMP has been used to solve multi-robot planning problems [72, 74].

In a classical Gaussian process motion planning problem, given a collection of timestamps $\boldsymbol{t} = [t_0, \cdots, t_k]^\intercal$, the robot trajectory $\boldsymbol{\theta} = [\theta_0, \cdots, \theta_k]^\intercal$ could be expressed as a joint Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Omega})$[68], with

$$\boldsymbol{\mu} = [\mu_0, \cdots, \mu_k]^\intercal, \boldsymbol{\Omega} = [\Omega_{ij}]_{i\times j | i,j\in[0,k], i,j\in\mathbb{N}}. \tag{2.1.1}$$

μ_i is the mean of the robot state at timestamp t_i, while Ω_{ij} is the covariance matrix of the robot states at timestamp t_i and timestamp t_j. At each state, θ_i, the probability of colliding with obstacles is $P(u_i|\theta_i)$. Given the collision cost function $\mathbf{h}_c(\boldsymbol{\theta})$ with covariance $\boldsymbol{\Sigma}_c$, define the negative log-likelihood function of $L_c(\theta_i)$:

$$L_c(\theta_i) = \mathbf{h}_c(\theta_i)^\intercal \boldsymbol{\Sigma}_c \mathbf{h}_c(\theta_i). \tag{2.1.2}$$

A motion planning problem considering obstacle avoidance can be represented as a maximum a posteriori estimation problem [22]:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{P(\boldsymbol{\theta}) \prod_{i=1}^{k} P(u_i|\theta_i)\} \tag{2.1.3}$$

$$= \arg\min_{\boldsymbol{\theta}} \{(\boldsymbol{\theta} - \boldsymbol{\mu})^\intercal \boldsymbol{\Omega} (\boldsymbol{\theta} - \boldsymbol{\mu}) + \sum_{i=1}^{k} L_c(\theta_i)\}, \tag{2.1.4}$$

where $P(\boldsymbol{\theta})$ is the prior distribution with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Omega})$. Gaussian process motion planning aims to find a collision-free path that minimizes the total obstacle collision cost.

2.2 Multi-Robot Simultaneous Localization and Mapping (SLAM)

Simultaneous localization and mapping (SLAM) is a fundamental capability in robot navigation, in which a mobile robot maps an unknown environment, while using relative measurements of that environment as the basis for localizing itself. Although many successful single-robot SLAM solutions have been proposed, fast and accurate scene reconstruction with a robot team remains an open problem. In multi-robot SLAM, a group of robots traverse an unknown environment and build a map cooperatively, by exchanging information. Cooperative robot teams have the potential to be

more efficient than a single robot in time-sensitive tasks, such as search- and-rescue, infrastructure inspection, household services, and logistical and transportation applications.

Let $N = \{1, 2, \cdots, n\}$ be the set of n robots. For each robot $\alpha \in N$, we denote its state at timestamp i as $\mathbf{x}_{\alpha,i}$. The robot odometry observation between present state $\mathbf{x}_{\alpha,i}$ and previous state $\mathbf{x}_{\alpha,i-1}$ is described by the equation:

$$\mathbf{z}_{\alpha,i}^{\alpha,i-1} = f(\mathbf{x}_{\alpha,i-1}, \mathbf{x}_{\alpha,i}) + \epsilon_{\alpha,i}^{\alpha,i-1}. \tag{2.2.1}$$

For landmark SLAM, assume robot α observes a landmark state $\mathbf{l}_j$ at timestamp i, we can describe the landmark observation:

$$\mathbf{z}_{j}^{\alpha,i} = g(\mathbf{x}_{\alpha,i}, \mathbf{l}_j) + \epsilon_{j}^{\alpha,i}. \tag{2.2.2}$$

If another robot β is observed at timestamp i by robot α,we refer to this as a *robot rendezvous observation*:

$$\mathbf{z}_{\beta,j}^{\alpha,i} = f(\mathbf{x}_{\alpha,i}, \mathbf{x}_{\beta,j}) + \epsilon_{\beta,j}^{\alpha,i}, \tag{2.2.3}$$

where $f(\cdot)$ denotes the state transformation between robot states, $g(\cdot)$ is the state transformation from robot state to landmark state, and $\epsilon_{\alpha,i}^{\alpha,i-1}$, $\epsilon_{j}^{\alpha,i}$ and $\epsilon_{\beta,j}^{\alpha,i}$ are zero-mean Gaussian noise variables.

At present timestamp t, $\mathcal{X} = \{\mathbf{x}_i^\alpha | \alpha \in N, i \in [0, t]\}$ represents the set containing the states of all n robots from the initial timestamp 0 to the present timestamp t. $\mathcal{L} = \{\mathbf{l}_0, \mathbf{l}_1, \ldots, \mathbf{l}_m\}$ is the set of landmarks observed until timestamp t. Additionally, let $\mathcal{Z}$ be the set containing odometry observations, landmark observations and robot rendezvous observations from all timestamps. The SLAM problem can be framed as a

maximum a posteriori estimation problem[50] in the landmark-based SLAM format:

$$\mathcal{X}^*, \mathcal{L}^* = \arg\max_{\mathcal{X},\mathcal{L}} P(\mathcal{X}, \mathcal{L}|\mathcal{Z}), \tag{2.2.4}$$

$$= -\arg\min_{\mathcal{X},\mathcal{L}} \underbrace{\log(P(\mathcal{X}, \mathcal{L}|\mathcal{Z}))}_{cost\ \ function\ \ \mathbf{F}(\mathcal{X},\mathcal{L})} . \tag{2.2.5}$$

In pose-graph SLAM where landmarks are not explicitly considered, the expression for maximum a posteriori estimation can be formulated as follows:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} P(\mathcal{X}|\mathcal{Z}), \tag{2.2.6}$$

$$= -\arg\min_{\mathcal{X}} \underbrace{\log(P(\mathcal{X}|\mathcal{Z}))}_{cost\ \ function\ \ \mathbf{F}(\mathcal{X})} . \tag{2.2.7}$$

A critical problem for multi-robot SLAM is the optimization of inter-robot constraints. Centralized methods [25, 81, 20, 53, 103], which collect all messages from local robots using a global server, can easily handle this task by optimizing all measurements in one factor graph. Distributed methods maintain several graphs across robots, making it harder to resolve ambiguities arising among them. In the widely-used distributed Gauss-Seidel (DGS) approach [12], each robot optimizes its own graph, considering other robots only when there are overlapping constraints. DDF-SAM [15] optimizes a local pose graph and constrains these local graphs using a constrained factor graph (CFG) containing shared landmarks among robots. DDF-SAM 2.0 [16] uses a decision tree in both local and neighbor graphs to avoid double-counting measurements. Tian [92] proposes Riemannian Block-Coordinate Descent (RBCD), which achieves a better convergence than the DGS method by solving a rank-restricted relaxation of the pose graph optimization problem [83].

2.2.1 Multi-Robot SLAM with Scene Graphs

Scene graphs, initially introduced by the computer vision community as a semantic image retrieval technique [48], have gained increasing attention in robotics due to their potential in enhancing human-robot interactions [84]. The introduction of scene graphs into the multi-robot SLAM problem offers additional benefits. Scene graphs are more data-efficient for communication between team members and, unlike compact sensor descriptors, provide a global and structured representation of the environment. In robotics, SLAM with scene graphs refers to a SLAM approach that processes sensor measurements as input and incrementally generates a scene graph (in addition to maps) as output. The scene graph commonly adopted in robotics is a 3D scene graph [1], which uses a graph-based structure to represent the semantic information of a 3D model of the environment. Unlike semantic maps [26, 5], a scene graph not only encapsulates the semantic details of the environment but also explicitly describes the relationships between semantic entities.

There are many single-robot or multi-robot SLAM works with scene graphs using visual sensors. However, these works focus on indoor environments due to the relatively low sensor range of RGB-D cameras. Kimera [84] and Hydra-multi [8] employ an RGB-D camera to sense the environment and incrementally improve scene graph construction accuracy using SLAM results. Radwan et al. [79] introduce visual fiducial markers to enable scene graph construction in under-construction environments. MR-COGraphs [38] proposes a learning-based, data-efficient visual feature method for inter-robot data association in multi-robot SLAM systems. The only exception is Multi S-graphs [31], which includes LiDAR in the multi-robot SLAM system. However, they use specific room-based compact descriptors for inter-robot association, which only work in indoor environments.

There are also LiDAR-based SLAM works with scene graphs in outdoor environments. For example, Greve et al. [37] and Deng et al. [18] construct scene graphs for outdoor environments using semantically labeled LiDAR scans. However, in these works, the scene graph serves only as a product, and its graph structure does not directly contribute to intra- or inter-robot loop closure detection in SLAM systems. In addition, it overlooks the crucial problem of inter-robot data association by assuming known robot initial positions (from GPS). This assumption makes these methods unsuitable for indoor environments where GPS signals are unavailable.

2.2.2 Multi-Robot SLAM for underwater environments

One key challenge that makes the underwater multi-robot SLAM problem unique, compared to other SLAM problems, is the limited communication bandwidth resulting from the marine environment. To address this constraint, Bonin-Font et al. [4] and MAM^{3}SLAM [24] propose centralized algorithms, where a server handles all inter-robot data association and optimization under the assumption of "a powerful server without communication restrictions". However, these methods are only tested on small-range datasets due to the communication bandwidth degradation as robots move farther from the server.

Zhang et al. [102] avoids the communication problem by employing a distributed local SLAM data collection approach combined with a GMRBnB-based map registration strategy, similar to the multi-session SLAM frameworks [6, 98]. While these methods are effective for bathymetric mapping tasks, they are less suitable for real-time decision-making applications, such as active planning and exploration.

Paull et al. [73] consider a scenario where robots observe one another and share their landmark observations. In their approach, both landmark and robot states are optimized in a distributed manner across team members using a graph-based repre-

sentation. Özkahraman et al. [71] explore a similar situation, intentionally designing robot rendezvous strategies to achieve more accurate localization in environments where landmarks are absent. However, achieving robot rendezvous is challenging, particularly for long-term missions, due to factors such as the vastness of marine environments, unpredictable oceanic currents, and time synchronization issues.

In this thesis, we adopt the communication assumptions described in DRACo-SLAM [66], where robots can communicate with each other (at low bandwidth, using wireless acoustic communication) when within a relatively long distance range. This assumption enables the real-time estimation of both self and neighbor robot states across a relatively large environment.

Another key consideration is the inter-robot data association or map merging strategy for multi-robot teams. Most approaches rely on extracting visual features from camera or sonar images. Bryson et al. [6] extract SIFT features from camera images and perform inter-robot structure-from-motion (SfM). Bonin-Font et al. [4] also extract SIFT features from camera images but utilize HALOC as the feature descriptor. MAM^{3}SLAM [24] extracts ORB features and associates keyframes using DBoW2. Zhang et al. [102] and Gaspar et al. [34] extract ORB features from sonar images and adopt a similar visual vocabulary strategy. While these feature-based methods are straightforward, their high-dimensional feature descriptors place significant demands on communication bandwidth.

To alleviate the communication bandwidth burden caused by high-dimensional feature descriptors, some methods use compact features to represent entire visual or sonar images. Paull et al. [73] extract mine-like objects from sonar images and exchange their positions among team members. Santos et al. [85] extract objects from sonar images and utilize scene graphs to describe and match sonar data. DRACo-SLAM [66] and Sonar-context [55] represent sonar images with compact sonar descrip-

tors for efficient data exchange and loop closure candidate detection. While compact features significantly reduce the bandwidth required for inter-robot data exchange, they can lead to errors in regions with repeating patterns (i.e., perceptual aliasing).

Some methods perform data association using features extracted from a larger submap to further mitigate perceptual aliasing. Wang et al. [97] apply the DBScan algorithm to cluster and describe the shapes of oil spills in the ocean. Deng et al. [19] utilize a deep learning-based encoder-decoder structure for point cloud transmission. Qi et al. [77] employ a terrain description feature to characterize submaps and match them using a genetic algorithm (GA). However, the GA process is computationally expensive and challenging to tune.

Other methods focus on robust loop closure outlier detection algorithms. Most approaches use a RANSAC strategy for map merging. DRACo-SLAM [66] uses pairwise consistent measurement set maximization (PCM) [65] to reduce outliers introduced by repeating patterns. However, despite its emphasis of measurement consistency, PCM is not robust to outliers characterized by high data similarity. To address this issue, Do et al. [21] propose a probabilistic approach specifically tailored for underwater scenarios to robustly reject outliers by seeking both consistency and similarity.

2.3 Multi-Robot Autonomous Exploration

Autonomous exploration and mapping describes a single robot or a group of robots navigating themselves in an unknown or partially known environment without human intervention. An accurate environment map constructed by the robots serves as the fundamental basis for all subsequent specific robot tasks [75].

Autonomous exploration and mapping have been a vigorously discussed sub-

ject for several decades. Early research has primarily centered around optimizing task distribution among team members [7, 32]. As Simultaneous Localization and Mapping (SLAM) techniques have advanced, various approaches [47, 101, 2, 57, 14] have incorporated the concept of map uncertainty into autonomous exploration. These strategies consider Gaussian noise sensor models and Gaussian noise kinematic models, aiming to strike a balance between exploration efficiency and managing uncertainties in the resulting maps. Although these techniques may differ in terms of their map representation and merging techniques, they share a similar utility function that takes into account both the information gain of the latest generated map and the required travel distance to the waypoints under consideration.

In multi-robot exploration, the communication arrangement of the robot system holds significance. Charrow et al. [9] examines the exploration of a small area using a team of robots equipped with range-only sensors using a coordinated approach. Although a centralized system maximizes the utilization of data collected by robot teams, it experiences increased computation challenges as the size of the robot team expands. Numerous algorithms, such as [7], [2], [13], [80], [86] and [10] adopt a decentralized approach. In this approach, robots exchange a significant portion of their historical data while independently making decisions based on their most recent knowledge of the environment. The asynchronous nature of the decentralized system reduces the communication and computational load in comparison to a scenario involving a centralized computer. However, it can still face limitations imposed by the communication range of the robot team. Additionally, there exist distributed approaches, discussed in [32], [61], [33], [52] and [36], in which robots communicate only when they encounter each other, adhering to constraints on communication bandwidth. Their decisions depend on the limited information available from both the robot team and the environment. Such distributed approaches are

designed for scenarios involving larger robot teams with constrained communication bandwidth, although they may involve some performance trade-offs to accommodate these conditions.

Another critical factor to consider is the criteria utilized for action selection. In earlier approaches such as [7], [32], the primary emphasis was on efficiently distributing tasks among team members. Later, the "Next Frontier" [13] introduced the concept of information potential to improve exploration efficiency. Jang et al. [47] utilize a Gaussian processes to characterize the environment, enabling obstacle avoidance and task allocation for a team of robots. In NeuralCoMapping [99], a multiplex graph neural network is introduced to strike a balance between long-term and short-term performance optimization. Tzes et al. [94] present a novel approach that integrates the exploration under uncertainty problem into a learning framework using graph neural networks (GNNs).

Given the uncertainty inherent in both robot control and sensing, some authors incorporate uncertainty into the exploration problem. Kontitsis et al. [57] incorporate Relative Entropy into the utility function when dealing with a partially known map to reduce localization uncertainty. Similarly, in [70], the authors perform joint entropy minimization to actively explore over a long-term horizon. Other researchers utilize filter-based techniques to address the uncertainty associated with future steps. In the work by Atanasov et al. [2], a square-root information filter is employed to predict the impact of future actions. Schlotfeldt et al. [86] adopt a similar filter-based approach within a decentralized system. Meanwhile, Kantaros et al. [52] present a sampling-based method aimed at reducing cumulative uncertainty stemming from dynamic hidden states. Indelman [45] presents an approach that employs belief propagation to anticipate the performance of robot teams in future actions. Chen [11] considers graph topology when making action selections.

Chapter 3

Mission-oriented Gaussian Process Motion Planning

3.1 Mission-oriented GPMP for UUVs over Current Flows

As mentioned in section 2.1, the problem of motion planning with obstacle avoidance can be formulated as a maximum a posteriori estimation problem. However, for underwater environments, some motion planning tasks require more than seeking the minimum-time collision-free path. We introduce a set of mission-related objectives with joint probability of $P(m_i|\theta_i)$ into equation (2.1.3):

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{P(\boldsymbol{\theta}) \prod_{i=1}^{k} P(u_i|\theta_i) P(m_i|\theta_i)\}. \tag{3.1.1}$$

This section addresses several considerations relevant to Unmanned Underwater Vehicles(UUV) missions: seafloor terrain-following within a specified altitude envelope, obeying the kinematics of a specific UUV, and the influence of current flows on a UUV. $P(m_i|\theta_i)$ is rewritten as:

$$P(m_i|\theta_i) = P(s_i|\theta_i) P(r_i|\theta_i). \tag{3.1.2}$$

We define $\theta_i = [x_i, \dot{x}_i]$, $x_i \in SE(3)$ and $\dot{x}_i$ as the pose and velocity at timestamp t_i. $P(s_i|\theta_i)$ indicates whether the present state obeys the specified maximum altitude relative to the seafloor. $P(r_i|\theta_i)$ is the probability corresponding to robot kinematics. The negative log-likelihood function of $P(m_i|\theta_i)$ has a similar definition as in equation

(2.1.2):

$$L_m(\theta_i) = \mathbf{h}_s(\theta_i)^\intercal \mathbf{\Sigma}_s \mathbf{h}_s(\theta_i) + \mathbf{h}_r(\theta_i)^\intercal \mathbf{\Sigma}_r \mathbf{h}_r(\theta_i). \tag{3.1.3}$$

$\mathbf{h}_s(\cdot)$, $\mathbf{h}_r(\cdot)$ and $\mathbf{\Sigma}_s$, $\mathbf{\Sigma}_r$ are the cost functions and covariances of our mission objectives. Finally, we perform factor graph optimization using the Dogleg method [17] to solve the mission-constrained motion planning problem, we term Mission-oriented GPMP (MGPMP), as a maximum a posteriori estimation problem:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{P(\boldsymbol{\theta}) \prod_{i=1}^{k} P(u_i|\theta_i) P(s_i|\theta_i) P(r_i|\theta_i)\} \tag{3.1.4}$$

$$= \arg\min_{\boldsymbol{\theta}} \{(\boldsymbol{\theta} - \boldsymbol{\mu})^\intercal \mathbf{\Omega} (\boldsymbol{\theta} - \boldsymbol{\mu}) + \sum_{i=1}^{k} \{L_c(\theta_i) + L_m(\theta_i)\}. \tag{3.1.5}$$

3.2 MGPMP Algorithm

We tackle the motion planning problem by formulating it as a nonlinear least squares problem, using a factor graph, with a starting state θ_0 and a goal state θ_k. In the time period between the starting time t_0 and ending time t_k, a set of k timestamps $\{t_0, ..., t_k\}^\intercal$ is sampled, with a time interval of Δt. At each timestamp t_i, there is a corresponding support state $\theta_i = [x_i, \dot{x}_i]$, where x_i and $\dot{x}_i$ are treated as separate vertices $\mathbf{v}_{x_i}$ and $\mathbf{v}_{\dot{x}_i}$ in the factor graph.

Figure 3.1 shows the factor graph structure of the proposed method. The neighbor pose (blue circle) and velocity (green circle) vertices at each timestamp are linked by GP prior factors, GP obstacle avoidance factors, and GP seafloor following factors. There are obstacle avoidance and seafloor following factors linked to each pose vertex. Kinematic constraint factors are linked to the velocity vertices. Similar to GPMP2 [22], prior and cost functions are represented by factors $\mathbf{f}(\cdot)$ connected to

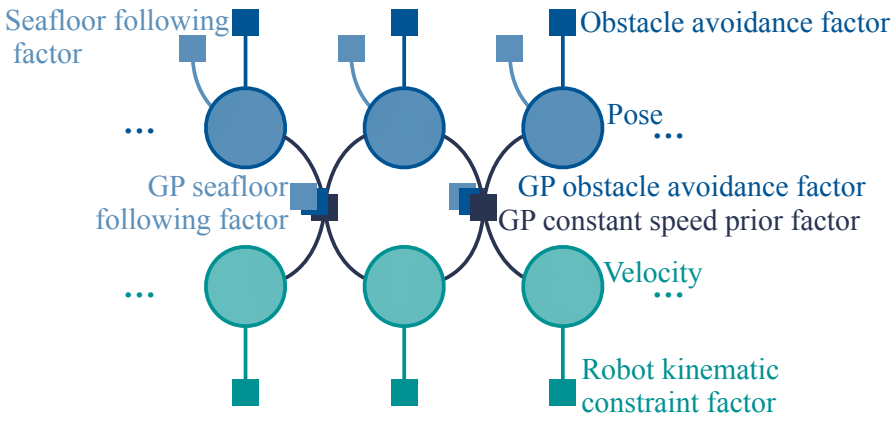


Figure 3.1: Factor graph structure of the proposed method.

vertices. We add a Gaussian prior to each pose vertex $\mathbf{v}_{x_i}$ and velocity vertex $\mathbf{v}_{\dot{x}_i}$ separately. GP constant speed prior factors considering current flows are introduced to neighboring support states to ensure smooth and efficient optimization. We describe the underwater environment and compute the collision cost at each support state using a signed distance field (SDF) [105]. Then, we add obstacle avoidance factors to pose vertices, and GP obstacle avoidance factors between adjacent pose and velocity vertices to ensure a continuous, collision-free trajectory.

In this problem, the missions mentioned in section 2.1 are also represented by factors. The seafloor terrain following mission requires the robot to maintain a certain distance from the seafloor. To achieve this, we add seafloor-following factors to the graph. These factors calculate the robot's distance-to-seafloor costs at all robot states and between neighboring states, using both Gaussian and GP seafloor-following factors. Another mission aims to minimize the effect of water currents while taking the robot's kinematics model into account. To simulate the robot's kinematics on each support state, we introduce a robot kinematic constraint factor. Additionally, when calculating the robot's inter-states for GP factors, we consider the influence of water currents on the robot.

3.2.1 Gaussian Prior and GP Constant Speed Prior

In Section 2.1, we stated that the robot state $\boldsymbol{\theta}$ can be represented as a joint distribution with mean and covariance as given in equation(2.1.1). Assuming that the support state θ_i at timestamp t_i is only correlated with the support state θ_{i-1} at previous timestamp t_{i-1}, we can represent $\boldsymbol{\Omega}$ as a symmetric matrix:

$$\boldsymbol{\Omega} = \begin{bmatrix} \Omega_{11} & \Omega_{12} & 0 & \dots & 0 \\ \Omega_{12} & \Omega_{22} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \Omega_{k-1k-1} & \Omega_{k-1k} \\ 0 & \dots & 0 & \Omega_{k-1k} & \Omega_{kk} \end{bmatrix}. \tag{3.2.1}$$

Additionally, we assume that the pose x_i and velocity $\dot{x}_i$ at any timestamp i are not correlated:

$$\theta_i \sim \mathcal{N}(\mu_i, \Omega_{ii}) \tag{3.2.2}$$

$$\mu_i = \begin{bmatrix} \mu_{x_i} \\ \mu_{\dot{x}_i} \end{bmatrix}, \boldsymbol{\Omega}_{ii} = \begin{bmatrix} \Omega_{x_i x_i} & 0 \\ 0 & \Omega_{\dot{x}_i \dot{x}_i} \end{bmatrix}. \tag{3.2.3}$$

Gaussian prior factor $\mathbf{f}_p(x_i) \sim \mathcal{N}(\mu_{x_i}, \Omega_{x_i x_i})$ and $\mathbf{f}_p(\dot{x}_i) \sim \mathcal{N}(\mu_{v_i}, \Omega_{\dot{x}_i \dot{x}_i})$ are introduced at $\mathbf{v}_{x_i}$ and $\mathbf{v}_{\dot{x}_i}$ to describe the prior distribution of support state θ_i. Given the water current speed u_{i-1} at position x_{i-1}, and assuming a linear current speed model, we approximate the neighbor states' kinematics with a simple constant speed model:

$$x_{i-1}^{\intercal} \cdot x_i = \Delta t \cdot (\dot{x}_{i-1} + u_{i-1}) \tag{3.2.4}$$

$$\dot{x}_i = \dot{x}_{i-1}. \tag{3.2.5}$$

Based on the constant speed model, we obtain the GP constant speed prior factor $\mathbf{f}_{cs}(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i) \sim \mathcal{N}(0, \Omega_{cs})$ with cost function:

$$\mathbf{h}_{cs}(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i) = \begin{bmatrix} x_{i-1}^{\intercal} \cdot x_i - \Delta t \cdot (\dot{x}_{i-1} + u_{i-1}) \\ \dot{x}_i - \dot{x}_{i-1} \end{bmatrix}. \tag{3.2.6}$$

In equation (3.2.1), for every timestamp i within the range $[0, k)$, we have $\Omega_{ii+1} = \Omega_{cs}$. The covariance Ω_{cs} is a pre-defined constant parameter across all timestamps.

3.2.2 Obstacle Avoidance with Signed Distance Field

Two kinds of obstacle avoidance factors are included in the graph: the Gaussian obstacle avoidance factor $\mathbf{f}_c(x_i)$ and the GP obstacle avoidance factor $\mathbf{f}_c(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i)$. Accordingly, $\forall \theta_i = [x_i, \dot{x}_i]$,

$$\mathbf{h}_c(\theta_i) = \mathbf{h}_c(x_i) + \mathbf{h}_c(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i). \tag{3.2.7}$$

Both factors are computed using a signed distance field (SDF) [105]. To generate an SDF model, a binary grid map b is used, where 1 represents occupied grid cells and 0 represents free space. A Euclidean distance map $d(b)$ is then generated by computing the distance to the nearest occupied cell for each grid cell in the map. We define a negative grid map $\bar{b} = J - b$, where J is a matrix of ones, and generate a corresponding Euclidean distance map $d(\bar{b})$. An SDF model can be obtained as $\mathcal{D}(b) = d(b) - d(\bar{b})$. To simplify the notation, we denote the value of the p^{th} grid cell in the SDF model as $\mathcal{D}(b(p))$.

With a pre-defined threshold d_{min}, indicating the smallest signed distance allowed, and p_i denoting the translational part of pose vertex x_i, the cost function of

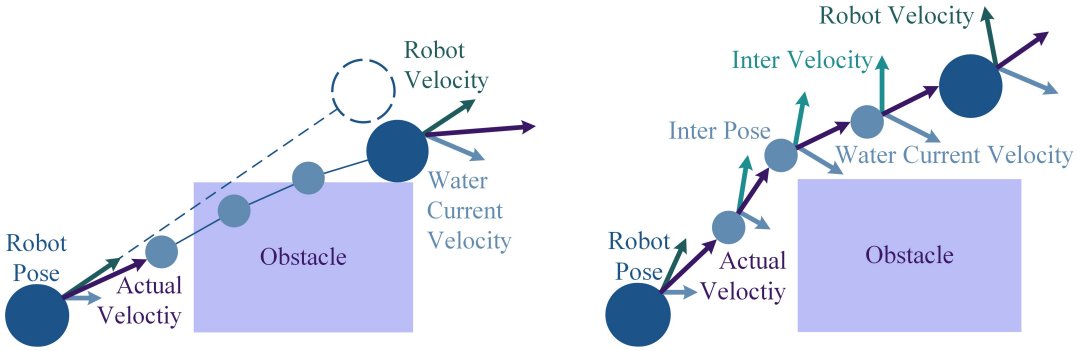
the Gaussian obstacle avoidance factor $\mathbf{f}_c(x_i)$ can then be defined:

$$\mathbf{h}_c(x_i) = \begin{cases} 0 & \mathcal{D}(b(p_i)) > d_{min} \\ d_{min} - \mathcal{D}(b(p_i)) & \mathcal{D}(b(p_i)) \leq d_{min} \end{cases}. \tag{3.2.8}$$

The GP obstacle avoidance factor, which is especially essential for robot navigation near seafloor terrain, is constructed by considering both robot kinematics and collision detection. For each pair of neighboring support states, denoted by $\{\theta_{i-1}, \theta_i\} \subset \boldsymbol{\theta}$, a group of n intermediate states are uniformly interpolated by $\mathbf{F}_{\text{inter}}(\theta_{i-1}, \theta_i)$. [22] gives a detailed description of how intermediate states are generated. As we are employing a linear speed model for both water current and inter-states interpolation, we account for the effect of the water current by taking into consideration the current speed at two neighboring states. Let $\{x_i^1, ..., x_i^n\} = \mathbf{F}_{\text{inter}}([x_{i-1}, \dot{x}_{i-1}+u_{i-1}], [x_i, \dot{x}_i+u_i])$ be poses of intermediate states between θ_{i-1} and θ_i considering the water current speed, and in turn, the cost function for GP obstacle avoidance factors can be expressed:

$$\mathbf{h}_c(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i) = \sum_{j=1}^{n} \mathbf{h}_c(x_i^j). \tag{3.2.9}$$

Figure 3.2 illustrates the significance of taking GP obstacle avoidance factors into account. Blue circles indicate the robot's position, and green arrows show the robot's velocity at different timesteps. Specifically, in figure 3.2a, when only Gaussian obstacle avoidance is considered at each support state, the robot may attempt to maintain the same speed between two adjacent states due to the GP constant speed prior. This can create a potential collision risk between two states. On the other hand,



(a) Example with only obstacle avoidance and constant speed GP prior.

(b) Example with obstacle avoidance and current velocity at each timestep.

Figure 3.2: A simple 2D example illustrating robot kinematics-based GP interpolation and the impact of water current velocity on obstacle avoidance.

figure 3.2b considers both factors and adjusts the velocity of one state to ensure a smoother, collision-free trajectory.

3.2.3 Seafloor Terrain Following

Similar to the obstacle avoidance constraints mentioned in section 3.2.2, the cost function of seafloor terrain following shown in equation (3.1.3) consists of Gaussian and GP parts:

$$\mathbf{h}_s = \mathbf{h}_s(x_i) + \mathbf{h}_s(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i). \tag{3.2.10}$$

To implement GP seafloor-following, we adopt a similar approach as described in Section 3.2.2 to interpolate n intermediate states between adjacent states subjected to currents. Consequently, the cost function for GP seafloor-following can be represented as the sum of costs for all intermediate states:

$$\mathbf{h}_s(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i) = \sum_{j=1}^{n} \mathbf{h}_s(x_i^j), \tag{3.2.11}$$

with $\{x_i^j | j \in [1, n], j \in \mathbb{N}\}$ representing the set of poses for all n intermediate states.

We represent the seafloor terrain using a 2D grid map $\mathcal{M}$ generated from bathymetric data of the seafloor terrain. Let $\mathcal{M}(p_i)$ be the seafloor depth linear-interpolated from grid map $\mathcal{M}$ at point p_i. The Gaussian seafloor-following cost function at pose x_i with position p_i is thus defined:

$$\mathbf{h}_s(x_i) = \begin{cases} 0 & \mathcal{M}(p_i) \leq d_{max} \\ \sigma(\mathcal{M}(p_i) - d_{max}) & \mathcal{M}(p_i) > d_{max} \end{cases}, \tag{3.2.12}$$

where d_{max} is the maximum allowable altitude, and $\sigma(.)$ is the Sigmoid function. The

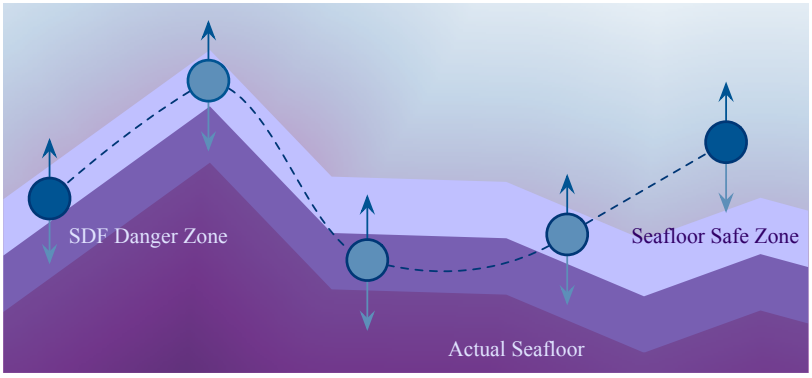


Figure 3.3: An example showing how the obstacle avoidance cost and the seafloor following cost conflict with each other.

conflict between the seafloor terrain following factor and obstacle avoidance factor in areas with complex seafloor terrain makes this problem particularly interesting. In figure 3.3, an example is shown where these two factors compete, and the conflict requires the algorithm to find an acceptable balance. The dark blue arrows represent obstacle avoidance factors, which aim to move the robot away from the seafloor, while the light blue arrows represent seafloor following factors, which pull the robot towards them. Furthermore, if a mission involves diving up and down around the starting and goal positions, the optimizer must balance between the constant speed

constraint and the seafloor following constraint. In contrast to sampling-based motion planning methods, Gaussian Process motion planning considers constraints as objective functions for optimization rather than state validity checks.

3.2.4 Robot Kinematics and the Influence of Current Flows

In this section, we consider a kinematic model for a standard, underactuated UUV, with specific parameters inspired by the Bluefin -21[1]. The model presents a 5-DoF point robot with a constraint that limits the robot's motion along the body y-axis. We impose a simple kinematic constraint on the robot at each speed vertex $\mathbf{v}_{\dot{x}_i}$, where $\mathbf{F}_{\text{getY}}(.)$ returns the y velocity component of robot body frame $\underrightarrow{\mathcal{F}_{\theta_i}}$ at state θ_i:

$$\mathbf{h}_r(\theta_i) = \mathbf{F}_{\text{getY}}(\dot{x}_i). \tag{3.2.13}$$

A notable aspect that sets our method apart from the original GPMP2 method is the incorporation of current flows, a disturbance that can greatly influence the motion of underwater robots. To incorporate current flows, we utilize a 3D grid matrix representation of water current speeds on different layers c, where each grid element denotes the water current speed at the center of the cell, in the horizontal plane defined by each layer. We focus on the current velocity in horizontal planes since it has the most significant impact on robot motion. A linear interpolation model is employed to acquire the velocity of the current at different locations, enabling effective querying and interpolation of water current speed. The velocity-related constraint factors in figure3.1 are primarily affected by the water current. Hence, we incorporate the influence of water current into our generation of all GP-related factors.

We take into account the influence of water current velocity when computing

[1] `https://gdmissionsystems.com/underwater-vehicles/bluefin-robotics`

the GP constant speed constraints and the GP inter-state interpolation, as discussed in section 3.2.1 and section 3.2.2. We first define the current speed at state θ_i by interpolating over the 3D current grid, denoting it as $u(p_i)$, where p_i represents the translation of θ_i. Next, we project $u(p_i)$ from the pre-defined global frame $\underrightarrow{\mathcal{F}_g}$ to the local frame $\underrightarrow{\mathcal{F}_{\theta_i}}$ using the transformation matrix $\mathbf{T}_{\theta_i g}$, which results in ${}_{\theta_i}u(p_i)$, indicating the water current speed at position p_i in the local robot body frame at θ_i:

$$ {}_{\theta_i}u(p_i) = \mathbf{T}_{\theta_i g} \cdot u(p_i). \tag{3.2.14} $$

The cost function of the GP constant speed prior factor in equation (3.2.6) is then rewritten as:

$$ \mathbf{h}_{cs}(x_{i-1}, \dot{x}_{i-1}, x_i, \dot{x}_i) = \begin{bmatrix} x_{i-1}^{\intercal} \cdot x_i - \Delta t \cdot (\dot{x}_{i-1} + {}_{\theta_i}u(p_i)) \\ \dot{x}_i - \dot{x}_{i-1} \end{bmatrix}. \tag{3.2.15} $$

And we generate the inter-states using the function $\mathbf{F}_{\text{inter}}$ while taking into account the effect of water current:

$$ \{x_i^1, ..., x_i^n\} = \mathbf{F}_{\text{inter}}([x_{i-1}, \dot{x}_{i-1} + {}_{\theta_{i-1}}u_{i-1})], [x_i, \dot{x}_i + {}_{\theta_i}u_i]). \tag{3.2.16} $$

3.3 Experiments

3.3.1 Experimental Setup

We conducted comparative motion planning experiments over four example environments consisting of seafloor terrain and multi-layer current grid maps. We refer to the four environment datasets as follows: (1) the Queen Elizabeth Islands dataset, (2) the Governors Island dataset, (3) the NYC lower bay dataset, and (4) the NYC upper bay

dataset. The Queen Elizabeth Islands dataset was obtained from underwater seafloor terrain models provided through DARPA's Symbiotic Design for Cyber Physical Systems (SDCPS) program. Bathymetry data provided by the National Oceanic and Atmospheric Administration (NOAA)[2] was used to resample the seafloor terrain grid map for the other three datasets. The multi-layer current grid maps were resampled from current flow data obtained from the Stevens Flood Advisory System at Davison Laboratory[3]. The current grid map used in the experiments was time-invariant, but could be modified to a time-variant current grid with minor changes. Additionally, we sized datasets (1) and (2) to create confined, challenging situations for testing obstacle avoidance capabilities.

We compared our proposed method with two state-of-the-art motion planning methods: RRT* [62] and STOMP [51]. The RRT* implementation from OMPL [90] was used, and both our proposed method and the RRT* implementation utilized a six DoF sphere robot model with a sphere safe zone of radius 1m. A kinematic constraint was applied to the y-axis of the robot frame, such that $\vec{v}_y = 0$. On the other hand, STOMP used a point-robot model, and no kinematic constraint was applied. In MGPMP, the initial and final states have zero linear and angular velocities along all axes. The covariance matrix Σ_r of kinematics constraints is set to the identity matrix of size 6x6 ($\mathbb{I}_6$). We define $d_{min} = 1$ and the covariance matrix $\Sigma_c = 0.1 \cdot \mathbb{I}_6$ for collision avoidance constraints. For seafloor terrain following constraints, we set $d_{max} = 2$ and the covariance matrix $\Sigma_s = \mathbb{I}_6$. STOMP and RRT* employ the identical range limits (d_{min} and d_{max}) as MGPMP and use the inverse covariance matrix as a weight for the objective function. All experimental comparisons were performed on a laptop computer equipped with an Intel Xeon E-2276M CPU and 31.1 GB memory

[2] `https://www.ncei.noaa.gov/maps/bathymetry`
[3] `https://hudson.dl.stevens-tech.edu/sfas`

running Ubuntu Linux 20.04. GPUs were not utilized in our experiments. In the rest of this section, we test the MGPMP algorithm under three conditions: collision avoidance (section 3.3.2), collision avoidance with seafloor terrain following (section 3.3.3), and a combination of collision avoidance, seafloor terrain following, and current flows (section 3.3.4).

3.3.2 Performance of Collision Avoidance

Benchmark Setup

We compare MGPMP, RRT* and STOMP with different number of states, using the Governors Island Dataset and Queen Elizabeth Islands dataset, which have obstacles between the start and goal states. Both methods are initialized with a straight-line trajectory connecting the starting and ending states. MGPMP and STOMP were tested with different numbers of states (10 and 30) , initialized with a straight-line trajectory. For MGPMP, we only considered SDF-based GP obstacle avoidance and kinematic constraints, and we sampled one interpolation pose between two nearby states for obstacle avoidance. The STOMP method solved only a 3D problem and used an obstacle function that only considered SDF-based obstacle avoidance. A limit of 10000 iterations was set for both methods. We also tested RRT* with two different objective functions: a single objective with only path length, denoted as "RRT* single", and a multi-objective version that included path length, kinematics constraints, and SDF-based obstacle avoidance, denoted as "RRT* multi". We set the termination time for the RRT* methods to 60 seconds to ensure enough time for a near-optimal result. Since RRT* and STOMP are not deterministic methods, we repeated each experiment 30 times.

Performance Analysis

The results presented in Table 3.1 show the mean of the 30 trials.

Table 3.1: Performance of Collision Avoidance

Dataset	Method	Config	Run Time (s)	Traj Length (m)
Governors Island	MGPMP	10	**0.09**	36.81
		30	0.41	43.03
	STOMP	10	0.83	**29.65**
		30	2.43	34.50
	RRT*	single	60	33.69
		multi	60	46.04
Queen Elizabeth Islands	MGPMP	10	**0.20**	**71.25**
		30	0.75	73.66
	STOMP	10	-	-
		30	2.37	73.31
	RRT*	single	60	72.67
		multi	60	85.77

Figure 3.4 and figure 3.5 illustrates the performance of different methods with varied parameterizations. All the methods succeed for the Governors Island dataset (figure 3.4). Our proposed method prioritizes safety over trajectory length compared to STOMP and RRT*, which favor shorter paths. However, for the Queen Elizabeth Islands Dataset (figure 3.5), STOMP with only 10 states fails to generate a collision-free trajectory because it considers obstacle avoidance at each state, but not for the trajectory connecting neighboring states. Thus, smaller state numbers increase the risk of collision for STOMP. RRT* with a single objective function offers the safest result, but at the cost of a 20% longer trajectory. Performance is also quantitatively evaluated in terms of runtime (in seconds) and total trajectory length (in meters) for

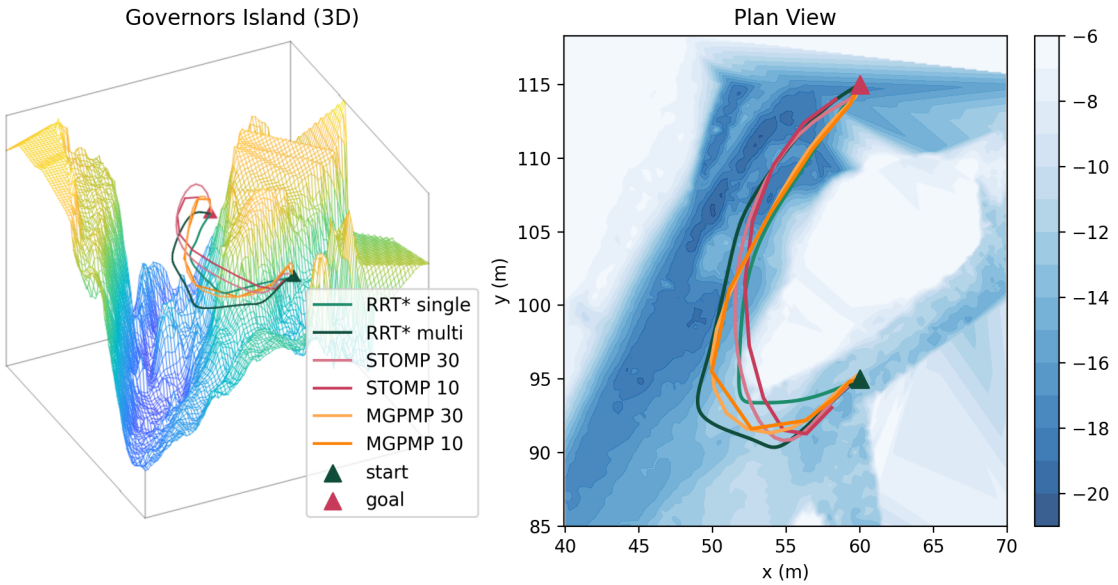


Figure 3.4: Performance of varied states of different methods on the Governors Island Datasets.

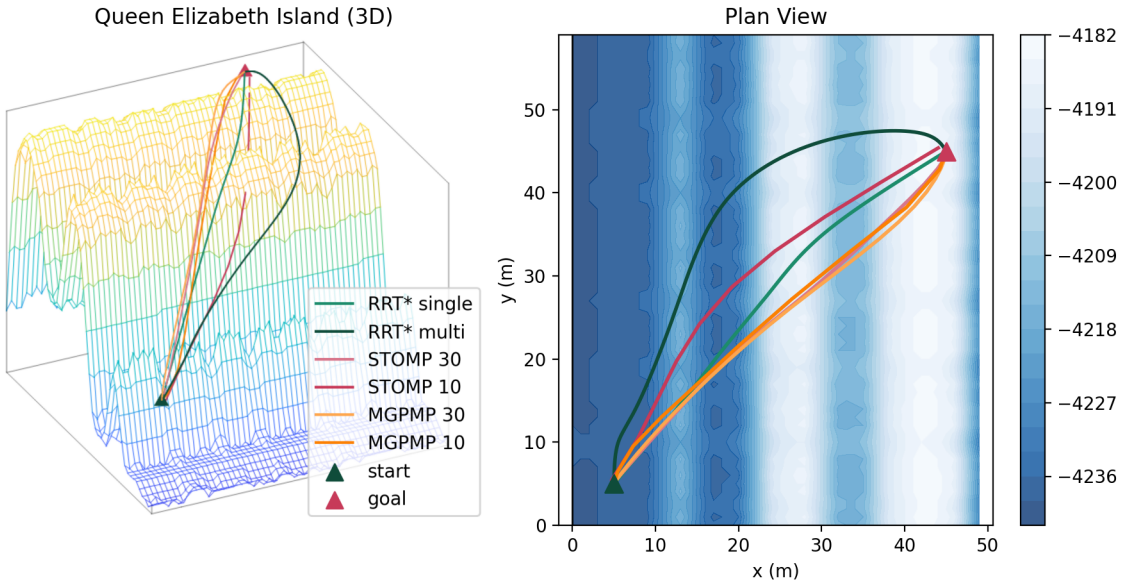


Figure 3.5: Performance of varied states of different methods on the Queen Elizabeth Islands Datasets.

each method, and the number of states used, in Table 3.1. For collision avoidance, MGPMP is the fastest method for both datasets, regardless of the number of states used.

3.3.3 Performance of Seafloor Terrain Following

Benchmark Setup

We simultaneously perform a seafloor terrain following mission and search for a collision-free trajectory using MGPMP, RRT*, and STOMP. We test these methods on the NYC Lower Bay and Upper Bay Datasets with horizontal map cell sizes of 0.1m, 1m, and 10m, as well as the Queen Elizabeth Dataset. We apply similar settings as in section 3.3.2 for all three methods, with the addition of a seafloor terrain following constraint to the cost function. For MGPMP and STOMP, we vary the number of states used for testing, selecting 30 and 100 for smaller datasets and 50 and 200 for larger datasets. The maximum time allowed for RRT* multi is 120 seconds. The quantitative results of various methods are shown in Table 3.2. Additionally, we introduce the mission violation rate (denoted as Violation Rate) and average mission violation distance (denoted as Avg. Violation Distance) to assess the performance of different methods on the seafloor terrain following mission. The mission violation rate refers to the proportion of states that fail to meet the requirements of the seafloor terrain following mission. The average violation distance represents the average violation of the max. allowable altitude, over the states that violate the mission.

Table 3.2: Performance of Seafloor Terrain Following

Method	Config	Run Time (s)	Traj Length (m)	Violation Rate (%)	Avg Violation Distance (m)
NYC Lower Bay					
MGPMP	50	1.28	1358.24	75	0.52
	200	7.21	1357.95	**43**	**0.36**
STOMP	50	**0.01**	1357.75	87	2.60
	200	0.02	**1357.67**	86	2.75
RRT*	multi	120	1652.40	74	1.99
NYC Upper Bay 0.1					
MGPMP	30	**0.30**	88.56	29 (9)	2.97 (0.82)
	100	1.59	90.06	**19 (3)**	3.04 (0.98)
STOMP	30	0.43	**81.46**	25(18)	1.62 (0.73)
	100	6.48	89.86	27 (24)	**0.77(0.57)**
RRT*	multi	120	92.73	47 (33)	2.96 (2.26)
NYC Upper Bay 1					
MGPMP	30	**0.92**	828.19	26 (13)	4.72 (1.83)
	100	3.51	839.12	**14 (6)**	3.54 (**0.60**)
STOMP	30	1.56	**806.76**	33(26)	1.75 (**0.60**)
	100	9.98	809.54	22 (20)	**1.12** (0.61)
RRT*	multi	120	875.23	47 (30)	3.51 (2.00)
NYC Upper Bay 10					
MGPMP	50	**2.12**	**8199.14**	12 (8)	**3.37 (0.31)**
	200	7.01	8691.11	**3 (2)**	8.34 (7.09)
STOMP	50	-	-	-	-
	200	-	-	-	-
RRT*	multi	120	9092.89	38 (36)	4.04 (3.68)
Queen Elizabeth Island					
MGPMP	30	**0.39**	**77.44**	**68**	11.51
	100	1.05	108.34	80	**9.93**
STOMP	30	-	-	-	-
	100	-	-	-	-
RRT*	multi	120	82.76	73	19.28

Performance Analysis

Figure 3.6 displays the performance on the NYC lower bay dataset. In the side view, the brown area represents a cross-section of the straight vertical path connecting the start and goal states and the seafloor terrain. As this environment lacks obstacles, all methods succeed with a relatively low violation rate.

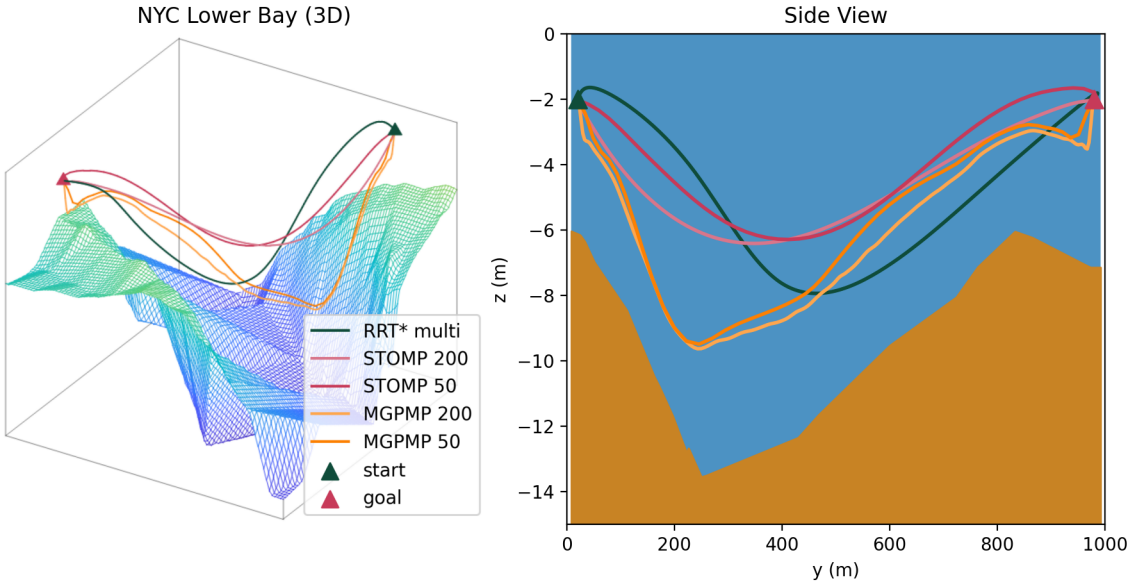


Figure 3.6: 3D and side view of the New York City (NYC) Lower Bay Dataset.

Figure 3.7 illustrates the performance on the NYC upper bay dataset with various map cell sizes, with different map scales, with "MGPMP 30" denoting results achieved using MGPMP with 30 states. In this experiment, the designated start and goal states both violate the threshold for the seafloor terrain following mission. To assess the completion of the seafloor terrain mission more precisely, two sets of violation metrics are calculated. One set includes the areas near the start and goal states (outside brackets), while the other excludes them (inside brackets). STOMP successfully discovers the global optimal trajectory with a small horizontal cell size of 0.1m. However, it fails for the map with 10m cell size. Our proposed method achieves

success across all map sizes, ensuring reliable solutions.

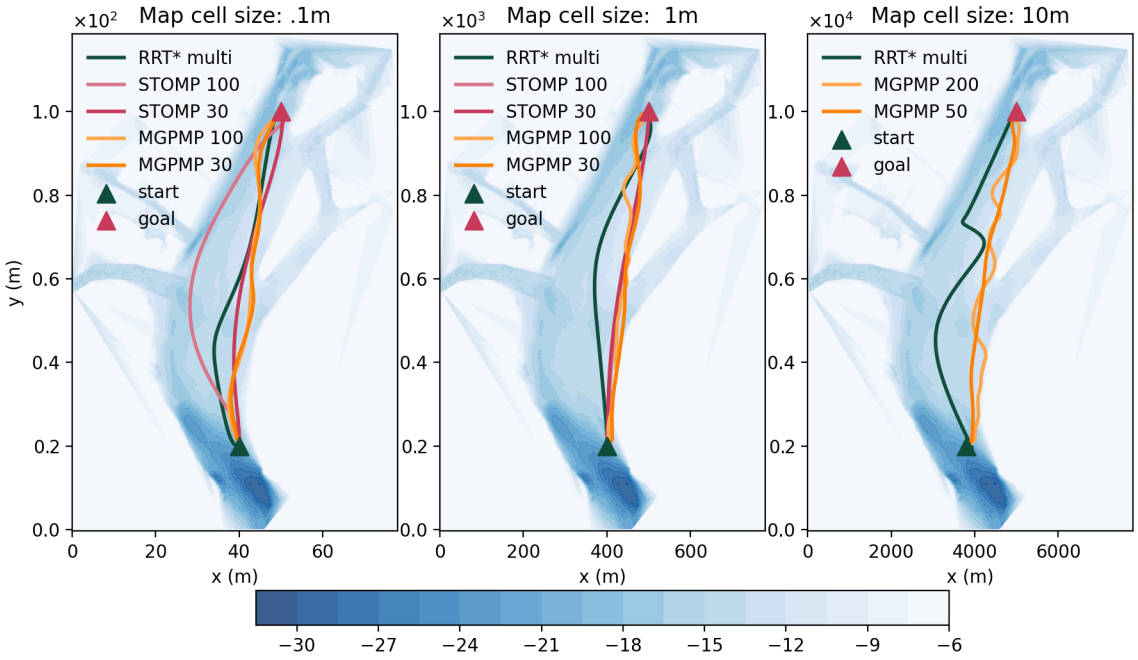


Figure 3.7: Plan view of the performance of varied states of different methods on the NYC Upper Bay Dataset.

Figure 3.8 displays the outcomes obtained on the Queen Elizabeth Islands Dataset, which is a significant challenge due to the conflicting nature of the SDF-obstacle avoidance constraint and the seafloor terrain following mission constraint in the narrow canyon sections. STOMP failed due to obstacle avoidance issues. RRT* managed to choose a safe trajectory but considerably distant from the seafloor. Our approach successfully navigates through the narrow canyon regions, achieving a valid solution while balancing both SDF obstacle avoidance and seafloor terrain following mission constraints.

3.3.4 The Influence of Current Flows

In this section, we present experimental results with and without the incorporation of current flows. Figure 3.9 shows the trajectories obtained under a suitably-tuned

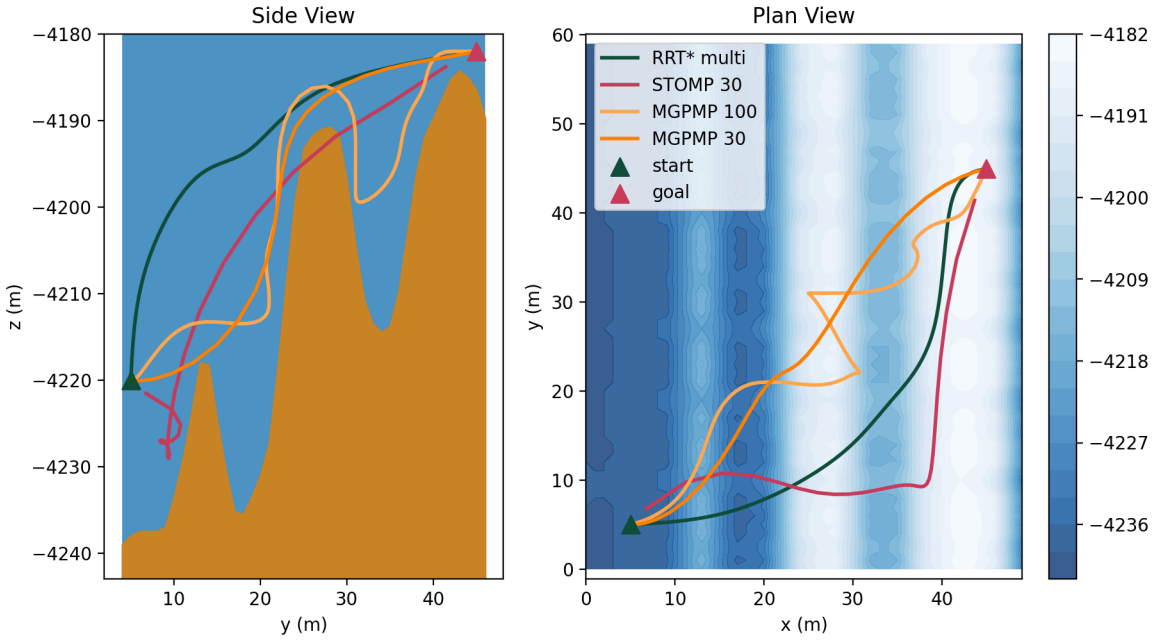


Figure 3.8: The performance of varied states of different methods on the Queen Elizabeth Islands Dataset.

50-state parameterization of MGPMP with and without considering the influence of currents. Without accounting for currents, the resulting trajectory is a straight-line path. However, when the effect of the current is taken into consideration, the trajectory exhibits a discernible leftward bend, permitting the trajectory to more safely accommodate the rightward push applied by the current flow field. To evaluate the computational impact, we measure the runtime required for different parameterizations of MGPMP, as outlined in Table 3.3. The consideration of currents introduces an additional 20% computational overhead. However, this trade-off yields trajectories that align more closely with real-world conditions, providing increased accuracy and fidelity in motion planning scenarios intended to capture real-world conditions.

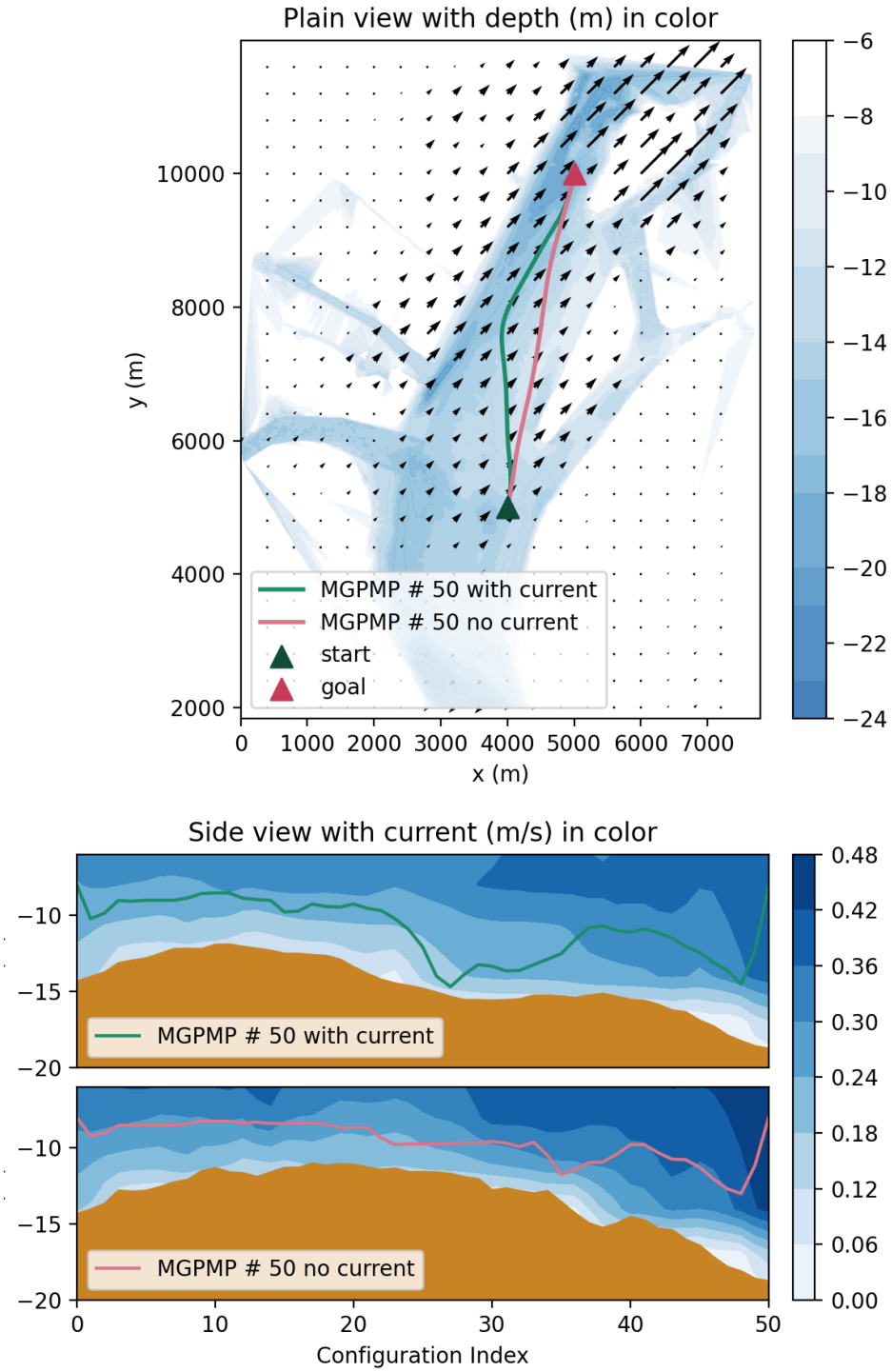


Figure 3.9: Performance with and without currents on the New York City (NYC) Upper Bay Dataset.

Table 3.3: Runtime of MGPMP with and without currents

# Config	50	200	50	200
Consider Current	✓	✓	✗	✗
Run Time (s)	0.79	10.29	0.54	8.38
Traj Length (m)	5264.49	5832.19	5104.24	5213.61

3.4 Conclusions

This chapter introduces a Mission-oriented Gaussian Process Motion Planning (MGPMP) approach for UUVs that incorporates current flows, and can be applied to maps of different scales. The proposed method includes a terrain following constraint designed for UUV missions requiring proximity to the seafloor within a specified altitude. Furthermore, the method takes into account oceanic currents and realistic UUV kinematic constraints, enhancing its relevance to real underwater missions. A comparative analysis is conducted between our method, RRT* and STOMP, considering varied parameterizations, which demonstrates the high performance and wide applicability of the proposed method.

Chapter 4

Multi-Robot LiDAR Simultaneous Localization and Mapping

4.1 Distributed Multi-Robot SLAM with Two-Stage Global-Local Graph Optimization

In Section 2.2, we formulate the multi-robot SLAM problem as a maximum a posteriori estimation issue. Within this section, we revisit equation 2.2.7, considering the intra-robot terms and inter-robot terms of the cost function:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \{\mathbf{F}_{intra}(\mathcal{X}) + \mathbf{F}_{inter}(\mathcal{X})\}. \tag{4.1.1}$$

Specifically, $\mathcal{C}_\alpha \subset \mathcal{C}$ denotes the set of intra-robot constraints among poses of robot α, while $\mathcal{C}_{\alpha\beta} \subset \mathcal{C}$ signifies the inter-robot constraints pertaining to poses of robots α and β.

$$\mathbf{F}_{intra}(\mathcal{X}) = \sum_{\alpha \in \mathcal{N}} \sum_{\langle i,j \rangle \in \mathcal{C}_\alpha} \mathbf{F}_{ij} \tag{4.1.2}$$

$$\mathbf{F}_{inter}(\mathcal{X}) = \sum_{\alpha,\beta \in \mathcal{N}, \alpha \neq \beta} \sum_{\langle i,j \rangle \in \mathcal{C}_{\alpha\beta}} \mathbf{F}_{ij} \tag{4.1.3}$$

In the distributed case, each robot optimizes its own contributions to the objective. For robot α, we have:

$$\mathbb{X}_\alpha^* = \arg\min_{\mathbb{X}_\alpha} (\overbrace{\sum_{\langle i,j \rangle \in \mathcal{C}_\alpha} \mathbf{F}_{ij}}^{\mathbf{F}_{intra}} + \overbrace{\sum_{\beta \in \mathcal{N}, \alpha \neq \beta} \sum_{\langle i,j \rangle \in \mathcal{C}_{\alpha\beta}} \mathbf{F}_{ij}}^{\mathbf{F}_{inter}}) \tag{4.1.4}$$

$$\mathbb{X}_\alpha = \mathcal{X}_\alpha \cup \left\{ \mathbf{x}_j \;\middle|\; \begin{array}{c} \mathbf{x}_j \in \mathcal{X}_\beta, \langle i,j \rangle \in \mathcal{C}_{\alpha\beta}, \\ \forall \beta \in \mathcal{N}, \beta \neq \alpha. \end{array} \right\}, \tag{4.1.5}$$

where $\mathbb{X}_\alpha$ contains all poses related to robot α. $\forall \mathbf{x} \in \mathbb{X}_\alpha$, the robot pose $\mathbf{x}$ consists of two parts, the rotation $\mathbf{R}$ and the translation $\mathbf{t}$. Since the rotation $\mathbf{R} \in \mathrm{SO}(3)$ is a non-convex component [12], equation (4.1.4) may fall into local minima instead of converging to a global minimal solution.

To address this, the widely used distributed Gauss-Seidel (DGS) method [12] (illustrated in figure 4.1(a)) rewrites $\mathbb{X}_\alpha$ as two subsets: $\mathcal{R}_\alpha$, containing the rotations of all poses, and $\mathcal{T}_\alpha$, containing the translations of all poses. Additionally, DGS entails a two-stage optimization process. For robot α, the DGS method first approximates the rotation $\mathcal{R}_\alpha$:

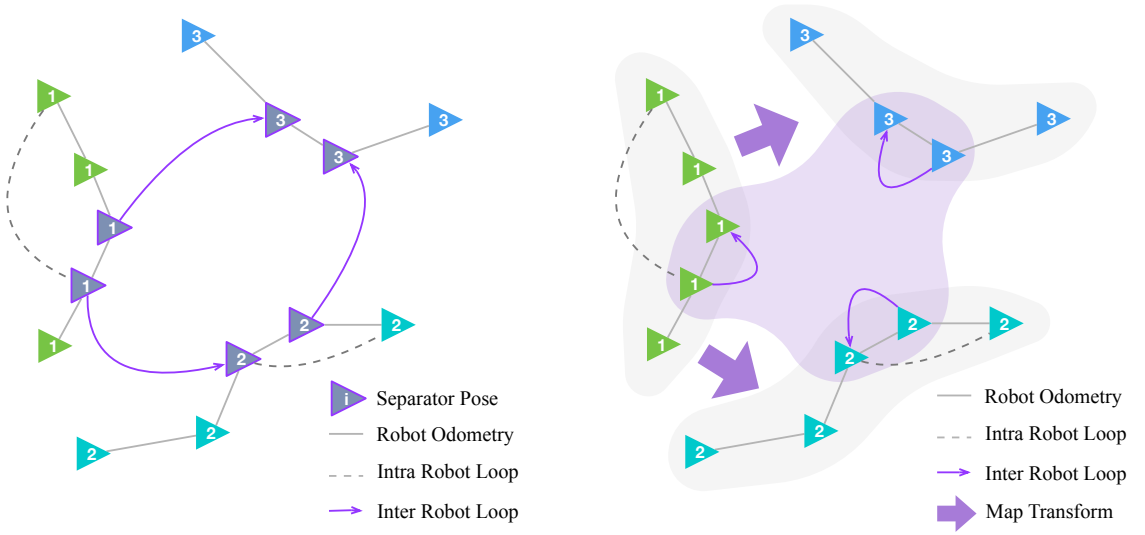
$$\mathcal{R}_\alpha^* = \arg\min_{\mathcal{R}_\alpha} \Big(\sum_{\langle i,j \rangle \in \mathcal{C}_\alpha} \mathbf{G}_{ij} + \sum_{\beta \in \mathcal{N}, \alpha \neq \beta} \sum_{\langle i,j \rangle \in \mathcal{C}_{\alpha\beta}} \mathbf{G}_{ij} \Big). \tag{4.1.6}$$

$\mathbf{G}_{ij}$ is the negative log-likelihood function only considering a robot's rotation:

$$\mathbf{G}_{ij} = [\mathbf{C}_{ij} - \hat{\mathbf{C}}_{ij}(\mathbf{R}_i, \mathbf{R}_j)]^T \omega_R [\mathbf{C}_{ij} - \hat{\mathbf{C}}_{ij}(\mathbf{R}_i, \mathbf{R}_j)]. \tag{4.1.7}$$

Similar to $\mathbf{z}_{ij}$ and $\hat{\mathbf{z}}_{ij}$, $\mathbf{C}_{ij}$ and $\hat{\mathbf{C}}_{ij}$ are the observed and expected relative rotation between $\mathbf{R}_i$ and $\mathbf{R}_j$. We rewrite Ω_{ij} as $\begin{bmatrix} \omega_R & 0 \\ 0 & \omega_t \end{bmatrix}$, where ω_R is the rotation block of Ω_{ij}. Then, the method performs a full-state graph optimization via the Gauss-Newton method, with the optimized rotation guess $\mathcal{R}_\alpha^*$, to solve equation (4.1.4). However, the full-state optimization step requires a good rotation approximation, while the first step is still solving a non-convex problem. So DGS may require a long time to

converge with a poor initial guess.



(a) DGS method: separator poses are where robots rendezvous or share common observations - they are included in each robot's optimization.

(b) Illustration of our method, where a global coordinate transformation graph (purple) and local pose graph (gray) are optimized by each robot.

Figure 4.1: Differences between DGS and our proposed method. Green, cyan, and blue triangles indicate the poses of three different robots.

Inspired by the DGS method, we propose a two-stage global-local graph optimization. The initial global optimization step solves the transformation among robots. $\forall$ pairs of separator poses $\langle \alpha_i, \beta_j \rangle \in \mathcal{C}_{\alpha\beta}$, with $\mathbf{x}_{\alpha_i} \in \mathcal{X}_\alpha$, $\mathbf{x}_{\beta_j} \in \mathcal{X}_\beta$, we define the local robot frame for robot α as $\underrightarrow{\mathcal{F}}_\alpha$. Let ${}_\alpha\mathbf{x}_{\alpha_i}$ be the pose at time i of robot α in its local coordinates, while ${}_\beta\mathbf{x}_{\alpha_i}$ is $\mathbf{x}_{\alpha_i}$ in robot β's local coordinates, $\exists \mathbf{T}_{\beta\alpha}$, transforming ${}_\alpha\mathbf{x}_{\alpha_i}$ to ${}_\beta\mathbf{x}_{\alpha_i}$ gives us:

$$ {}_\beta\mathbf{x}_{\alpha_i} = \mathbf{T}_{\beta\alpha} \cdot {}_\alpha\mathbf{x}_{\alpha_i} = {}_\beta\mathbf{x}_{\beta_j} \cdot {}_\beta\mathbf{z}_{\beta_j\alpha_i} \tag{4.1.8} $$

$$ \mathbf{T}_{\beta\alpha} = {}_\beta\mathbf{x}_{\beta_j} \cdot {}_\beta\mathbf{z}_{\beta_j\alpha_i} \cdot ({}_\alpha\mathbf{x}_{\alpha_i})^T. \tag{4.1.9} $$

Once there are inter-robot loop closures between robot β and robot α, $\mathbb{T}_{\beta\alpha}$ can be determined. $\mathbb{T}_{\beta\alpha} = \{\mathbf{T}_{\beta\alpha}^{(1)}, ..., \mathbf{T}_{\beta\alpha}^{(m)}\}$ is the set of estimations of $\mathbf{T}_{\beta\alpha}$ obtained from m inter-robot loop closures. Let us next assume the global frame $\underrightarrow{\mathcal{F}}_g$ is aligned with the local frame of the first robot $\underrightarrow{\mathcal{F}}_1$ ($_g\mathbf{x}_\alpha = \ _1\mathbf{x}_\alpha$). Let $\mathbb{T}$ be the set of transformations from any robot frame to the global frame:

$$\mathbb{T} = \{\mathbf{T}_{g\alpha} \mid \forall \alpha \in \mathcal{N}, \alpha \neq g\}. \tag{4.1.10}$$

We aim to minimize the total transformation error between local robot frames with the Levenberg-Marquardt method:

$$\mathbb{T}^* = \arg\min_{\mathbb{T}} \sum_{\alpha,\beta\in\mathcal{N}} \mathbf{e}_{\beta\alpha}^T \Omega_{\beta\alpha} \mathbf{e}_{\beta\alpha}. \tag{4.1.11}$$

$\mathbf{e}_{\beta\alpha}$ is the error of one transformation, and $\forall \mathbf{T}_{\beta\alpha}^{(i)} \in \mathbb{T}_{\beta\alpha}$:

$$\mathbf{e}_{\beta\alpha}(\mathbf{T}_{g\beta}, \mathbf{T}_{g\alpha}) = \mathbf{T}_{\beta\alpha}^{(i)} - \hat{\mathbf{T}}_{\beta\alpha}^{(i)}(\mathbf{T}_{g\beta}, \mathbf{T}_{g\alpha}). \tag{4.1.12}$$

Next, all inter-robot constraints are transformed to local robot frames, and the local graph optimization step is performed. Let us suppose there are inter-robot constraints between robot α and robot β. To perform the local optimization of robot α, we should transform the separator poses of robot β into local coordinates. Accordingly, $\forall \mathbf{x}_{\beta_j} \in \mathcal{X}_\beta \cup \mathbb{X}_\alpha$:

$$_\alpha\mathbf{x}_{\beta_j} = \hat{\mathbf{T}}_{g\alpha}^T \cdot \hat{\mathbf{T}}_{g\beta} \cdot \mathbf{x}_{\beta_j}. \tag{4.1.13}$$

Finally, consider $_\alpha\mathbb{X}_\alpha$ as the initial value for separator poses. For any two sets of separator poses, $\langle \alpha_i, \beta_j \rangle$, $\langle \alpha_k, \gamma_l \rangle$, a virtual intra-robot loop closure $_\alpha\mathbf{z}_{\beta_j\gamma_l}$ can be

computed by equation 4.2.2.

We perform pose graph optimization using the Levenberg-Marquardt method on equation 4.1.4 with a modified inter-robot term, $\mathbf{F}_{inter}$:

$$\mathbf{F}_{inter} = \sum_{\substack{\alpha,\beta,\gamma \in \mathcal{N}, \\ \alpha \neq \beta, \alpha \neq \gamma}} \sum_{\substack{\langle \alpha_i, \beta_j \rangle \in \mathcal{C}_{\alpha\beta}, \\ \langle \alpha_k, \gamma_l \rangle \in \mathcal{C}_{\alpha\gamma}}} \mathbf{e}_{vir}^T \mathbf{\Omega}_{vir} \mathbf{e}_{vir} \tag{4.1.14}$$

$$\mathbf{e}_{vir} = \mathbf{z}_{\alpha_i\beta_j} \cdot_\alpha \mathbf{z}_{\beta_j\gamma_l} \cdot (\mathbf{z}_{\alpha_k\gamma_l})^T - \hat{\mathbf{z}}_{\alpha_i\alpha_k}(\mathbf{x}_{\alpha_i}, \mathbf{x}_{\alpha_k}) \tag{4.1.15}$$

This two-stage global and local optimization ensures a high-quality initial guess for the local optimization step, which results in faster convergence. Furthermore, the introduction of local robot frames contributes to the numerical stability and consistency of the estimates and error covariances of the individual mobile robots. Thus, the SLAM systems of the local robots are only dealing with minor numerical changes in poses due to odometry drifts, which is beneficial for applications in active SLAM, path planning, and exploration.

4.2 DiSCo (Distributed Scan Context) - SLAM Algorithm

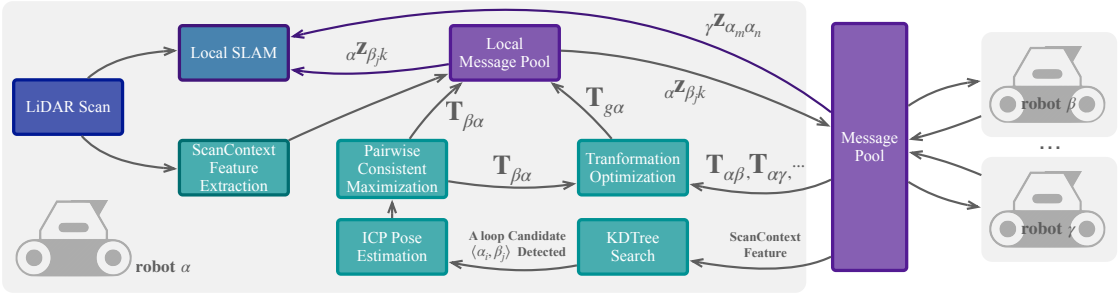


Figure 4.2: Overview of the architecture of DiSCo-SLAM, our proposed distributed multi-robot SLAM system, with robot α as the local robot.

An overview of our distributed multi-robot SLAM system is shown in figures

4.1 and 4.2. Per the architecture shown in figure 4.2, once the system receives a LiDAR scan, the local SLAM thread and feature extraction thread are activated simultaneously. We adopt LIO-SAM [89] as our local SLAM framework, and alternatively use LeGO-LOAM [88] when tightly-coupled LiDAR-inertial data is unavailable. Scan Context [54], a lightweight spatial feature descriptor for 3D LiDAR, is used to describe and match features. Then, mobile robots exchange scan context features and perform scan matching. Once a potential inter-robot loop closure candidate is detected, incremental pairwise consistent measurement set maximization (PCM) [65] is performed to remove outliers, as in [60]. A two-stage optimization is performed by each robot, first establishing a global-to-local coordinate transformation, which informs a subsequent local pose graph optimization. Finally, coordinate transformations are exchanged among robots for further optimization.

Scan Context Feature Description and Matching

Scan Context (SC) [54] describes a LiDAR scan by projecting the scan onto a 2D plane, where the z-coordinate value of each 3D point is encoded in the intensity of the corresponding 2D point. The 2D scan image is then divided into grid cells according to a specified number of sectors N_s and rings N_r. For the Velodyne VLP-16 Lidar employed in our work, we use $N_s = 60$ and $N_r = 16$. The value of each grid cell I is the maximum intensity of all points captured in the cell. Finally, a ring key feature of dimension N_r is extracted by counting the non-zero values of each ring.

A ring key KD tree is then built for loop closure candidate search. All SC features of the top N matched ring key features are further compared to identify the best loop closure candidate. The SC features are shifted along the sector axis to ensure rotation invariance. The shifting angle also serves as an initial rotation guess for the ICP scan-matching process when there is no coordinate transformation

history.

4.2.1 Incremental PCM

Unlike single-robot SLAM, for which incremental odometry measurements are frequently available to support an accurate initial guess, inter-robot loop closure candidates in multi-robot SLAM are only estimated by feature matching. Incremental pairwise consistent measurement set maximization (PCM) is introduced to avoid the acceptance of erroneous loop closures, which may result from different environmental regions with similar appearance, or objects in the environment arranged in repeating patterns.

PCM [65] checks the consistency of inter-robot constraints. A loop closure is accepted if any two inter-robot constraints $\mathbf{z}_{\beta_j\alpha_i}$ and $\mathbf{z}_{\beta_l\alpha_k}$ meet the following condition:

$$\left\|(\mathbf{z}_{\beta_l\beta_j} \cdot \mathbf{z}_{\beta_j\alpha_i} \cdot \mathbf{z}_{\alpha_i\alpha_k}) \cdot {\mathbf{z}_{\beta_l\alpha_k}}^{-1}\right\|_2^2 < \epsilon. \tag{4.2.1}$$

$\mathbf{z}_{\beta_l\beta_j}$ is the intra-robot transformation of robot β between timestamps l and j; $\mathbf{z}_{\beta_j\alpha_i}$ is the inter-robot transformation relating timestamp j of robot β and timestamp i of robot α; ϵ is a small threshold (in our experiments to follow, we choose $\epsilon = 5$). To ensure robust real-time localization, we use a lazy initialization: incremental PCM will not be performed until there is a designated number of loop closure candidates. In our case, incremental PCM is performed after more than five loop closures are detected.

4.2.2 Two-Stage Global and Local Optimization

As mentioned in section 4.1, the robots in our proposed framework perform a two-stage global and local optimization. In the global step, coordinate transformations

between robots are treated as measurements. As shown in equation 4.1.11, only transformations from local robots to the global coordinate frame are optimized. The covariance matrices of these measurements are linearly related to the timestamp, since each robot's dead reckoning error grows as time accumulates.

After global optimization, all separator poses from other robots are transformed to the local coordinate frame according to the latest coordinate transformation matrices. Then, a euclidean distance based radius search is performed to find the nearest inter-robot constraint. During the radius search, separator poses whose timestamps are too close to the present timestamp are discarded to avoid optimizing an ill-posed graph. Present separator pose ${}_{\alpha}\mathbf{x}_{\beta_j}$ and nearest separator pose ${}_{\alpha}\mathbf{x}_{k}$ are converted to a virtual intra-robot loop closure:

$$ {}_{\alpha}\mathbf{z}_{\beta_j k} = {}_{\alpha}\mathbf{x}_{\beta_j}{}^{T} \cdot {}_{\alpha}\mathbf{x}_{k}. \tag{4.2.2} $$

Finally, the virtual observations are added to the local pose graph and the local pose graph is optimized.

4.2.3 Message Passing Between Robots

When two robots rendezvous, they share their past SC features (which each robot stores in a last in, first out buffer) as well as coordinate transformations between themselves and other robots. The shared coordinate transformations are added to the global factor graph for each robot's global optimization step. Each recipient robot searches for neighbors of the shared SC features in its respective KD tree. If an SC feature match is found, the recipient robot will query its neighbor for a feature point cloud containing edge and planar features, and its corresponding pose in the neighbor robot's local coordinates. The feature point cloud is then used for scan matching.

If an inter-robot loop closure is detected, the recipient robot will send the resulting transformation to the related robot when it is feasible to do so. Our use of SC permits data-efficient communication, even when there are no communication constraints and message-passing can occur at all times. An overview of sizes/quantities of messages passed in a practical use-case is given in section 4.4.1.

4.3 SGM (Scene Graph Matching) - SLAM Algorithm

The system architecture within each robot is illustrated in figure 4.2. We perform semantic sensor fusion using both camera and LiDAR data, while simultaneously estimating the robot states through LiDAR-Inertial odometry and intra-robot loop-closure optimization. We employ Point-LIO [39] to compute LiDAR-Inertial odometry, and intra-robot loop closures are detected using a Euclidean distance-based approach similar to that of LIO-SAM [89].

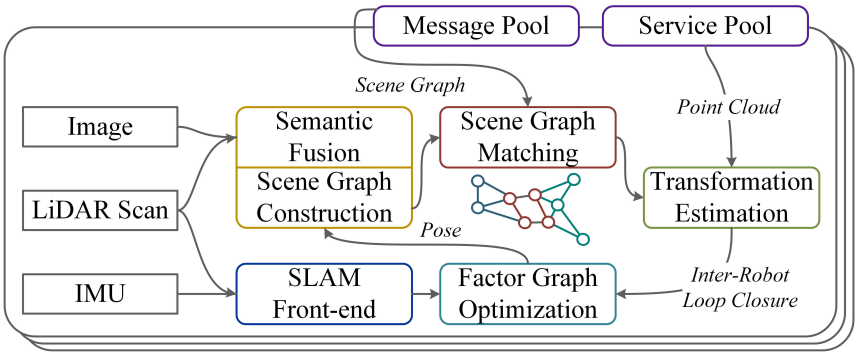


Figure 4.3: **System Architecture.** The SGM-SLAM pipeline for each robot.

Next, we construct a scene graph based on the estimated robot states and the semantically labeled point-cloud from the semantic fusion. This scene graph is then shared with robot neighbors. Once the local robot receives a scene graph from a robot neighbor, scene graph matching is performed. If a sufficient number of objects in the scene graph are matched, a relative transformation is computed to determine

the inter-robot loop closures. Finally, all the constraints, as shown in equation (4.1), are added to the SLAM factor graph. Both the local robot states and the scene graph are then subsequently updated.

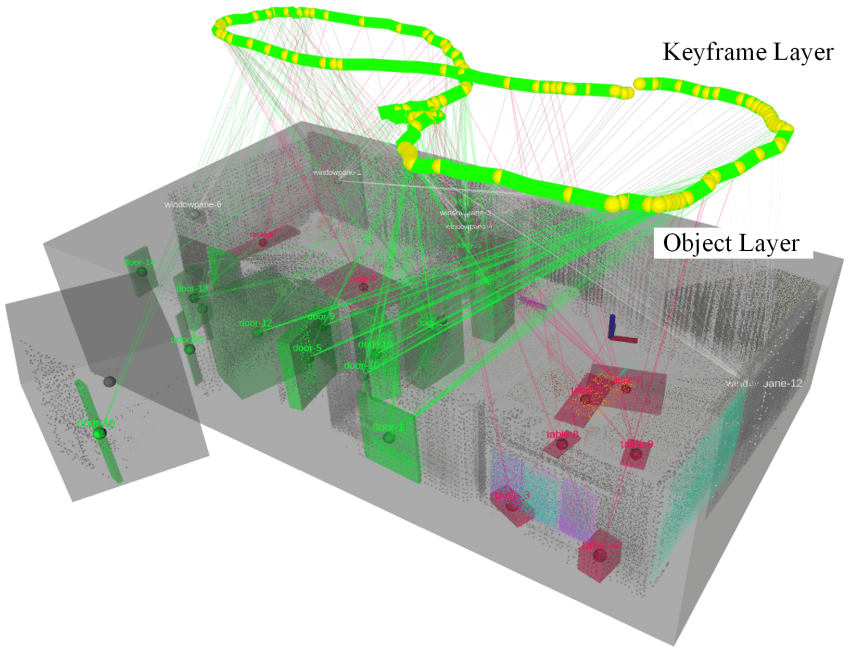


Figure 4.4: **The semantic SLAM result from a single robot to build a scene graph.** Keyframe poses are marked in yellow, odometry poses in green, and objects are represented by bounding boxes.

4.3.1 Scene Graph Construction

Figure 4.4 illustrates the required information from a single robot to build a scene graph. We build a scene graph using information from the keyframe layer and the object layer in our semantic SLAM system. The keyframe layer includes keyframes selected by the SLAM front-end as vertices, connected by odometry measurements between them, along with intra-robot and inter-robot loop closures. The object layer is a fully connected graph where object vertices are interconnected by edges, using Euclidean distances between objects as the edge weights. Keyframe vertices are linked

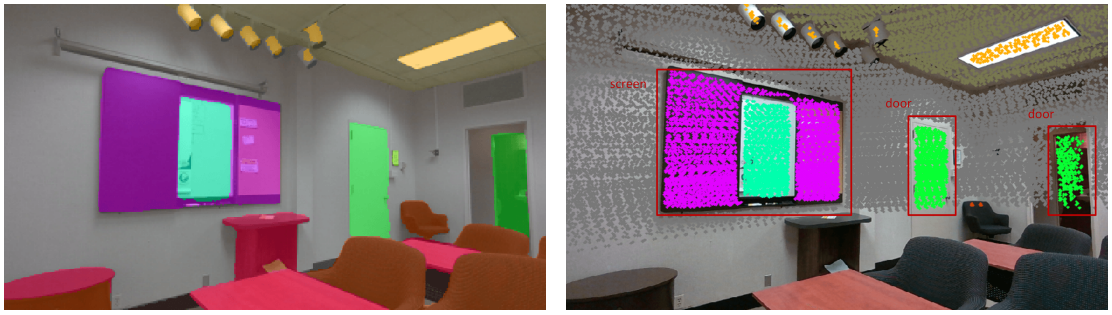


Figure 4.5: Segmented image with detected objects shown in different colors (left) and LiDAR scan projected onto the segmented image, with objects of interest marked by red rectangles (right).

to object vertices if the corresponding object is observed in the respective keyframe.

The object vertices in the object layer are detected through semantic sensor fusion. During this process, RGB images are aligned with LiDAR scans to create a semantically labeled point cloud containing objects of interest. As shown in figure 4.5, the RGB images are segmented using Oneformer [46]. The point set from the LiDAR scan is transformed into the 2D RGB image frame utilizing the LiDAR-camera extrinsics and camera intrinsics. Subsequently, the semantic label from the nearest pixel is assigned to the corresponding point. Finally, the labeled point cloud is clustered into distinct objects.

A scene graph is constructed using objects (nodes) across keyframes, with relationships (edges) weighted by Euclidean distances among objects. It can be for-

mulated as a Linear Assignment Problem (LAP):

$$A_n^* = \arg\min_{A_n} \sum_{v_j \in \mathcal{V}_l^{(t)}} \sum_{v_k \in \mathcal{V}_l^{(t-1)}} c_{jk} \cdot a_{jk}; \tag{4.3.1}$$

$$\forall v_k \in \mathcal{V}_l^{(t-1)}, \sum_{v_j \in \mathcal{V}_l^{(t)}} a_{jk} = 1;$$

$$\forall v_j \in \mathcal{V}_l^{(t)}, \sum_{v_k \in \mathcal{V}_l^{(t-1)}} a_{jk} = 1. \tag{4.3.2}$$

Given two sets of object candidates (vertices) at the current keyframe timestep t and the previous keyframe timestep $t-1$, denoted as $\mathcal{V}_l^{(t)}$ and $\mathcal{V}_l^{(t-1)}$, we define the cost matrix C such that each element $c_{jk} = \left\| p_j - p_k \right\|_2$ represents the Euclidean distance between the object centers of the two vertices. A_n is the Boolean assignment matrix indicating object vertex matches between two sets with elements $a_{jk} \in \{0, 1\}$. If $a_{jk} = 1$ and the cost $c_{jk} < c_{\max}$, then the j_{th} vertex $v_j \in \mathcal{V}_l^{(t)}$ and k_{th} vertex $v_k \in \mathcal{V}_l^{(t-1)}$ are matched. $c_{\max}$ represents the maximum allowable distance for two object candidates across keyframes to be identified as the same object. We use the Hungarian Algorithm [58] to solve this LAP.

4.3.2 Scene Graph Matching

In the scene graph matching step, we define the scene graph of the local robot l as $\mathcal{G}_l = \langle \mathcal{V}_l, \mathcal{E}_l \rangle$, and the scene graph received from a neighbor robot as $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i \rangle$. The problem of graph matching across robots can be viewed as a maximum bipartite matching problem. Due to the absence of an initial guess for inter-robot data association, we cannot directly use Euclidean distances between object centers for graph matching, as is done in the intra-robot case (section 4.3.1) for constructing the scene graph. Instead, we adopt a similar technique used in SemanticLoop [100], and

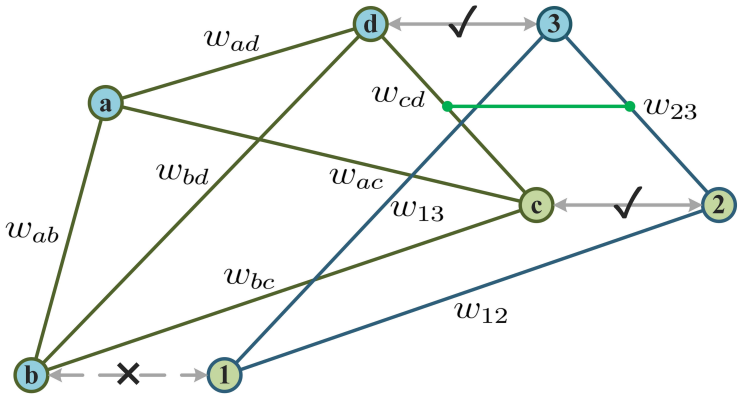


Figure 4.6: A simple example illustrating edge-wise matching between two scene graphs. Edge lengths represent the weights, while node colors (green and blue) denote vertex types.

formulate the problem as a Quadratic Assignment Problem (QAP):

$$A_e^* = \arg\max_{A_e} \sum_{v_j, v_k \in \mathcal{V}_l} \sum_{v_m, v_n \in \mathcal{V}_i} a_{jm} \cdot u_{jkmn} \cdot a_{kn}; \tag{4.3.3}$$

$$\forall v_k \in \mathcal{V}_i, \sum_{v_j \in \mathcal{V}_l} a_{jk} = 1;$$

$$\forall v_j \in \mathcal{V}_l, \sum_{v_k \in \mathcal{V}_i} a_{jk} = 1. \tag{4.3.4}$$

Here A_e is the Boolean assignment matrix where each element $a_{jk} \in \{0, 1\}$ indicates the matching status of vertices $v_j \in \mathcal{V}_l$ and $v_k \in \mathcal{V}_i$. $a_{jk} = 1$ means v_j and v_k are matched. u_{jkmn} is an element of the 4D utility tensor U:

$$u_{jkmn} = d_v(v_j, v_m) \cdot d_v(v_k, v_n) \cdot d_e(e_{jk}, e_{mn}). \tag{4.3.5}$$

The function $d_v(\cdot)$ represents the vertex-wise comparison. In our case, we compare only the labels of two objects to avoid confusion caused by partially observed objects. If two objects have the same label, we set $d_v(\cdot) = 1$; otherwise, $d_v(\cdot) = 0$. The edge-

wise comparison is defined as $d_e(e_{jk}, e_{mn}) = \exp(-\mu|w_{jk} - w_{mn}|)$, where w_{jk} is the weight of edge e_{jk} and μ is a scaling factor. Figure 4.6 showcases a simple example of edge-wise matching. In this figure, edge lengths represent the actual weights of the graph. Consequently, $d_e(w_{13}, w_{bd})$, $d_e(w_{12}, w_{bc})$ and $d_e(w_{23}, w_{cd})$ are significantly higher than the $d_e(\cdot)$ values of other matches. However, since vertex b and vertex 1 are of different types, $d_v(v_b, v_1) = 0$, resulting in only e_{23} and e_{cd} being matched.

We then vectorize the assignment matrix as $\text{vec}(A_e)$, and reshape U into a 2D matrix U_{2D}, allowing for the rewriting of equation (4.3.3):

$$A_e^* = \arg\max_{A_e} \text{vec}(A_e)^\top \cdot U_{\text{2D}} \cdot \text{vec}(A_e). \tag{4.3.6}$$

In equation (4.3.3), a_{jk} is originally constrained to be Boolean. However, to make the problem more tractable, we relax this constraint by allowing $a_{jk} \in [0, 1]$. As proved by Leordeanu et al. [64], the solution, A_e^*, is then given by the positive eigenvector of U_{2D} corresponding to its principal eigenvalue. Finally, the vertex-wise matching of the objects can once again be formulated as an LAP similar to equation (4.3.1), with cost matrix $C = -A_e^*$. Additionally, a minimum allowable eigenvector value is introduced to prevent mismatches during the assignment.

4.3.3 Transformation Estimation

The transformation estimation module performs inter-robot data association based on the graph matching result. Let the matched vertex pairs from local graph $\mathcal{G}_l$ and neighbor graph $\mathcal{G}_i$ be denoted by $\mathcal{M}_{l,i}$. Our goal is to estimate the relative robot frame transformation $\mathbf{T}_{WB_i}$ using this matched vertex pair set alone. For each matched vertex pair $\forall < v_l^m, v_i^m > \in \mathcal{M}_{l,i}$, let p_l^m and p_i^m represent the object centers of the corresponding vertices. The transformation relationship between these points

can be expressed as:

$$\begin{bmatrix} p_l^m \\ 1 \end{bmatrix} = \mathbf{T}_{WB_i} \begin{bmatrix} p_i^m \\ 1 \end{bmatrix}. \tag{4.3.7}$$

As long as we have more than 3 non-collinear objects, we can estimate $\mathbf{T}_{WB_i}$ using a rigid body transformation. To ensure a more robust solution, we assume that all objects lie on a plane, allowing us to reduce the problem to a 2D rigid body transformation estimation.

4.3.4 Inter-robot Loop Closure

The robot frame transformation estimate $\mathbf{T}_{WB_i}$ is used as the coarse initial transformation guess for inter-robot loop closure. We then refine the result by performing Iterative Closest Point (ICP) registration using the sparse object cloud associated with the matched objects.

In our SLAM system, each object is connected to keyframes where the corresponding objects are observed, allowing us to retrieve two sets of keyframes (one from each robot), $\mathcal{F}_l^m$ and $\mathcal{F}_i^m$, corresponding to the overlapping regions of the local robot and neighbor robots, respectively. These keyframes are transformed into world frame $\mathbf{W}$ based on the latest robot state estimation $\mathcal{X}_{\text{assoc}}^i$ and transformation $\mathbf{T}_{WB_i}$.

We extract the object information associated with the keyframe sets. We first construct $\mathcal{F}_i^m = \{F_i^m(j-k), \ldots, F_i^m(j+k)\}$ in a sliding window fasion, where k controls the window size. In this paper, we use $k = 5$. Several matched object vertices $v_i^m \in \mathcal{M}_{l,i}$ are observed in this subset. We then retrieve the corresponding matched object vertices v_l^m from the local scene graph and prepare our local subset, consisting of keyframes connected to the object vertices. Finally, we perform point cloud registration between these two subsets. The inter-robot loop closures are added into the SLAM factor graph to improve the local robot state estimation. To ensure

computational efficiency, each keyframe from the neighbor robot is associated with only one local keyframe.

4.3.5 Communication Between Robots

The robots continuously share their latest scene graphs with neighbor robots. Only partial scene graphs are shared, including object vertices, their corresponding features, and the IDs of keyframes linked to those objects. After receiving a scene graph from a neighbor, the robot stores and compares only the most recent scene graphs. If a sufficient number of overlapping objects is detected from scene graph matching, the robot sends a service request to the corresponding neighbor for the compressed point cloud associated with overlapping objects, optimized robot state estimation, and marginalized covariances of the keyframes connected to the matched object vertices. If a keyframe point cloud from the neighbor robot is already stored locally, only the pose and marginalized covariance are requested. To optimize communication bandwidth, the point cloud contains only geometric information. Upon receiving the request, the neighbor robot searches its dataset and sends the required data. This data-efficient message-passing strategy ensures that a lightweight scene graph is continuously shared, while the bandwidth-intensive compressed point cloud is transmitted only when required.

4.4 Experiments

4.4.1 DiSCo-SLAM

Experimental Setup

In the experiments to follow, we present results using four customized datasets configured for compatibility with 3D LiDAR-based, distributed multi-robot SLAM. We

refer to the four datasets as (1) the KITTI 08 dataset, (2) the KITTI 00 dataset, (3) the Stevens dataset, and (4) the Park dataset. We adapt our first two datasets from the KITTI Vision Benchmark raw data sequences 08 and 00 [35]. In sequence 08, 100Hz raw inertial measurement unit (IMU) data is recorded with suitable temporal consistency to be used with LIO-SAM[89]. We use LeGO-LOAM [88] to run sequence 00 with LiDAR scans only. Since both LIO-SAM and LeGO-LOAM are configured to work with a 16-channel LiDAR, we downsample the KITTI LiDAR data from 64 beams to 16. We have modified sequences 08 and 00 into a synthetic two-robot dataset and a synthetic three-robot dataset respectively, where time-stamps have been adjusted to incorporate overlap and rendezvous.

The Stevens dataset is adapted from a dataset previously gathered on our campus [88] with a Clearpath Jackal UGV equipped with a Velodyne VLP-16 LiDAR, and only LiDAR data is used for LeGO-LOAM [88]. The dataset from this earlier experiment has been modified into a two-robot dataset which includes overlap and rendezvous. Lacking RTK-GPS ground truth in this dataset, we use satellite imagery to qualitatively evaluate our experimental results.

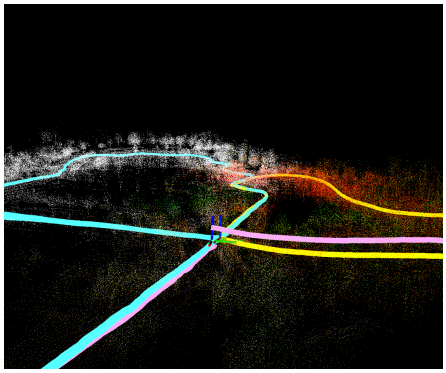
Figure 4.7: Our Jackal UGV instrumented with LiDAR, IMU, and a GPS used for ground truth.

The Park dataset was gathered specifically for this study, using a single Clearpath Jackal UGV, which is pictured in figure 4.7. Our UGV is equipped with a Velodyne VLP-16 LiDAR, a MicroStrain 3DM-GX5-25 IMU, and a Single-band RTK GNSS receiver, to both apply LIO-SAM [89] and evaluate it using RTK-GPS derived ground truth information. To generate a synthetic three-robot dataset from this single-robot mission, we rewrote the timestamps of each synthetic robot's trajectory to achieve a meaningful synchronization of three intersecting robot trajectories. Each robot executes a "figure-eight" trajectory comprised of two large loops; the individual robots will accumulate errors along these loops, but there is sufficient overlap among robots that inter-robot constraints can alleviate these errors.

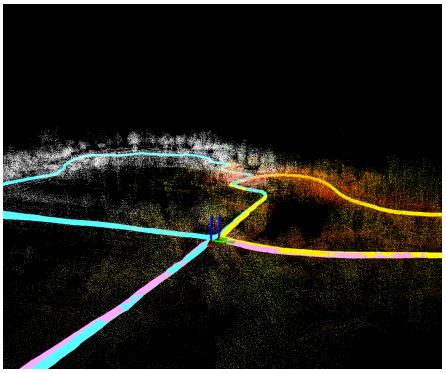
In all four datasets, KITTI (08 and 00), Stevens, and Park, each robot trajectory includes intra-robot and inter-robot loop closures, and each robot encounters at least one rendezvous with every other robot. When GPS is available, it is used as ground truth for quantitative analysis, and not used for SLAM. We project the GPS measurements onto Universal Transverse Mercator (UTM) coordinates and perform a coordinate transformation to facilitate comparisons with our SLAM results. All experimental comparisons are performed using playback of previously gathered data on a desktop computer equipped with an Intel i9-9900K CPU, 62.7 GB memory using the robot operating system (ROS) in Ubuntu Linux 18.04. All robot threads run concurrently on the same processor, and do not utilize GPUs.

Performance of Two-Stage Optimization

In this section, we perform comparisons of multiple configurations of our proposed DiSCo-SLAM two-stage optimization framework, using the three-robot Park dataset. Displayed in figure 4.9 are multi-robot SLAM results for increasing levels of optimization. These comparisons are intended to display the efficacy of the proposed



(a) No inter-robot constraints (as in figure 4.9(b)).

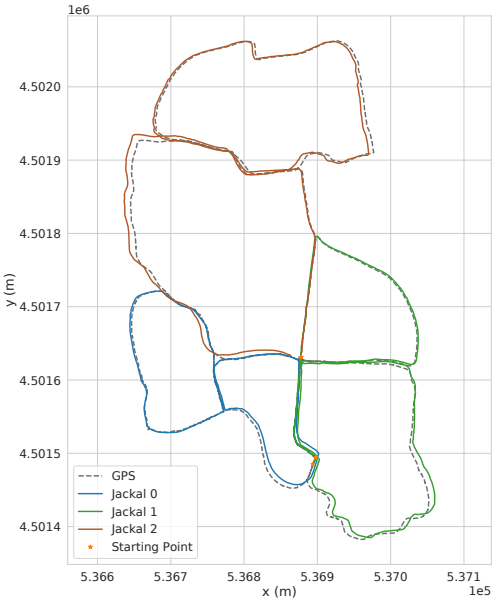


(b) With inter-robot constraints (as in figure 4.9(c)).

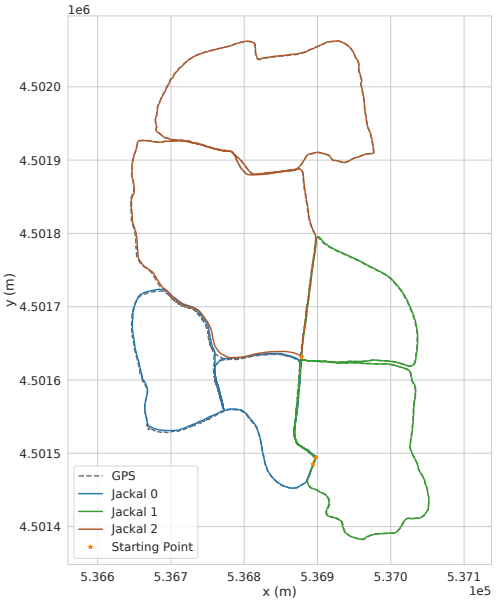
Figure 4.8: Park dataset, at a location where there is overlap among all three robot trajectories.

global and local optimization procedures in combination, versus their standalone performance. In figure 4.9(a), only the global optimization step is applied, and neither inter-robot nor intra-robot loop closure constraints are included in the local optimization step; it is produced using odometry only. Accordingly, overlap and rendezvous among robots fails to inform local pose-graph constraints, and thus results in a visible buildup of localization error in the result. The inclusion of intra-robot loop closures into the local optimization procedure, seen in figure 4.9(b), noticeably improves robot localization performance. Further improvements are apparent in figure 4.9(c), where inter-robot constraints are also added to each vehicle's local pose graph, including them in the local optimization step. A further inspection of the benefits of utilizing inter-robot constraints in local pose graphs can be seen in fugure 4.8. The "jackal 2" robot from the plots of figure 4.9 (pink) is completing a large loop with no intra-robot constraints. There, one can clearly note the higher consistency in pose estimates across overlapping robots, when leveraging these constraints.

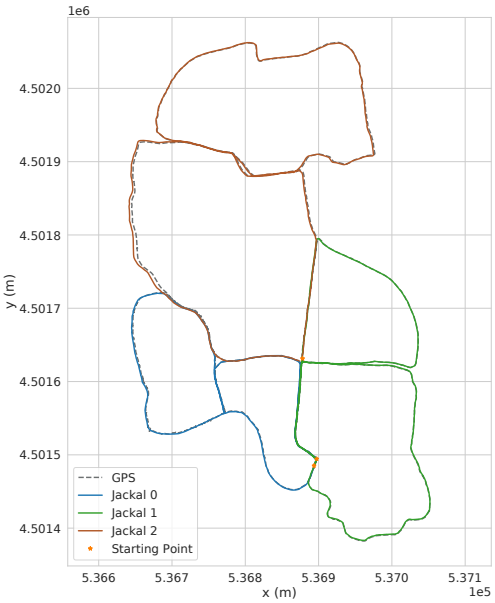
Quantitative pose estimation error metrics for a representative SLAM execution trace over the Park dataset are listed in Table 4.1. To compute these errors



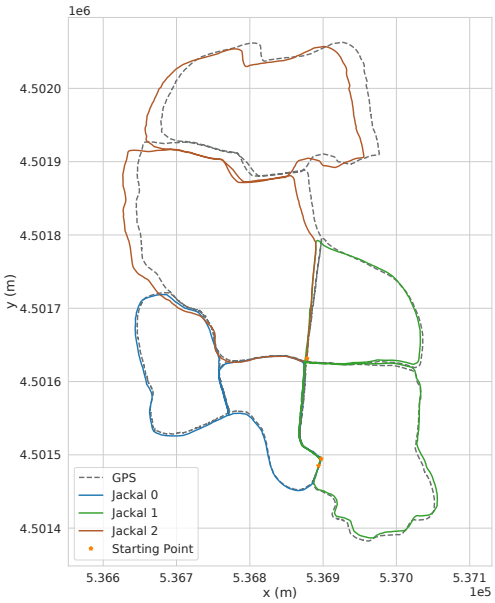
(a) DiSCo-SLAM, with global transformation optimization, and locally, only odometry.



(b) DiSCo-SLAM, w/ global and local optimization, no inter-robot constraints in local pose graphs.



(c) DiSCo-SLAM as proposed, full global and local optimization.



(d) DGS optimization with PCM.

Figure 4.9: Representative trajectory estimation results over the three-robot Park dataset, with different multi-robot SLAM configurations.

Table 4.1: Root Mean Square Error (RMSE) w.r.t. GPS

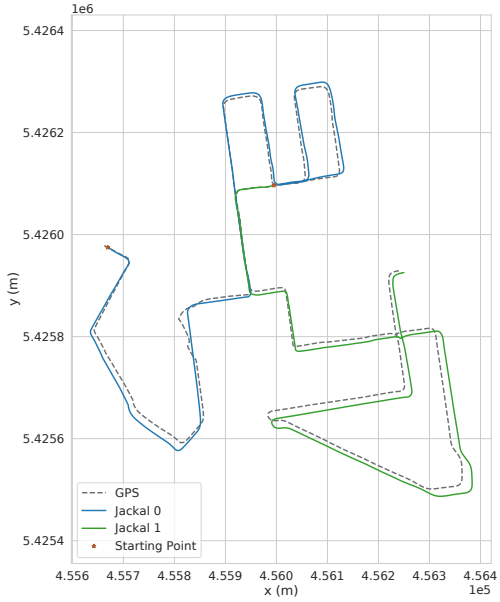
Dataset	Configure	X (m)	Y (m)	Total (m)
Park	DiSCo-Odometry	4.87	3.35	5.91
	DiSCo-Local	1.39	1.11	1.78
	Full DiSCo-SLAM	**1.31**	**0.52**	**1.43**
	DGS with PCM	11.84	5.38	13.00
KITTI 08	Full DiSCo-SLAM	**5.57**	**4.39**	**7.09**
	DGS with PCM	14.81	19.47	24.46
KITTI 00	Full DiSCo-SLAM	**3.48**	**5.61**	**6.60**
	DGS with PCM	14.54	14.78	20.73

relative to ground truth information, GPS data is collected at a rate of 5Hz, while we generate one LiDAR keyframe per second. We match each keyframe pose with the nearest GPS pose according to their timestamps. The estimated trajectories are transformed from local SLAM coordinates into UTM coordinates using the starting points of the "jackal 1" and "jackal 2" trajectories as geometric constraints (the jackal 1 and jackal 2 trajectories are denoted in the plots of figure 4.9). The result in Table 4.1 with both intra- and inter- robot constraints added to the local pose graph achieves the highest accuracy.

Comparison with Distributed Gauss-Seidel (DGS)

Due to its data-efficiency and relevance to real-time multi-robot SLAM applications, we next compare our method against DGS optimization with PCM (summarized in figure 4.1 and equation 4.1.6-4.1.7), which comprises the back-end of DOOR-SLAM [60], a framework that has supported distributed multi-robot SLAM across different platforms and sensing modalities, including LiDAR. We test both DGS with PCM and DiSCo-SLAM with the same front-end on four datasets, (1) our modified KITTI

(a) DGS optimization with PCM.

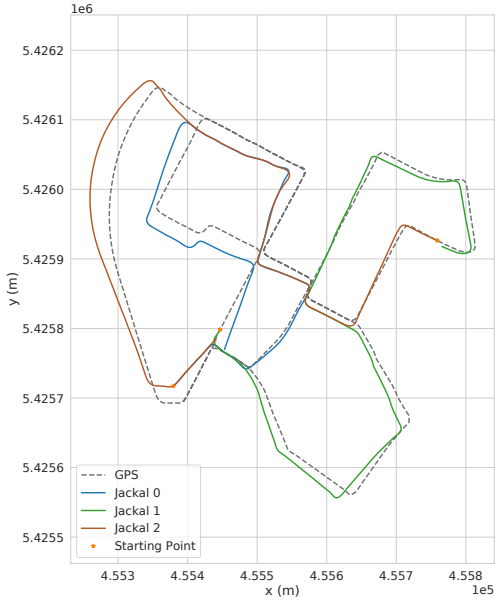


(b) DiSCo-SLAM, with full global and local optimization.

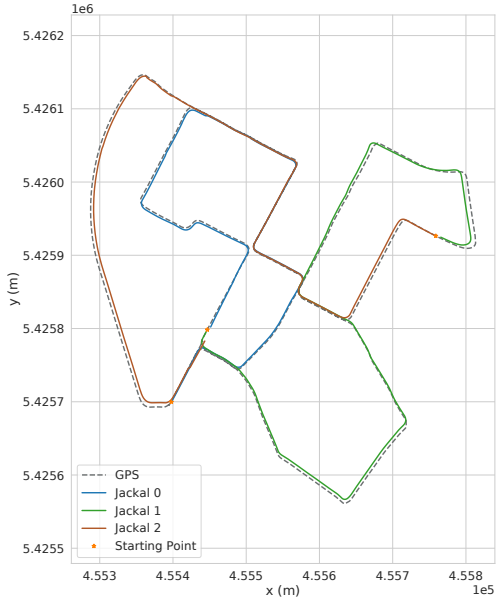
Figure 4.10: Optimization result on the KITTI 08 dataset.

08 dataset, (2) our modified KITTI 00 dataset, (3) the Stevens campus dataset and (4) the Park dataset. The RMSE with respect to GPS is given in Table 4.1.

Figure 4.10 shows representative results optimized by DGS over the KITTI 08 dataset. We transform the trajectories to align them with the GPS data according to the starting point of both robots, although the GPS data undergoes a small amount of erroneous drift in this dataset. The rotation angles are incorrectly estimated by DGS at several corners in figure 4.10a where turns occur, while there are no significant errors in the result of DiSCo-SLAM when turning corners in figure 4.10b. The KITTI 00 dataset is challenging since only LiDAR scans are used, and the LiDAR frame spacing is larger than our VLP-16 datasets. Both DiSCo-SLAM and DGS with PCM achieve low accuracy since the LiDAR frame rate is low, and thus fewer inter-robot loop closures are detected. The optimized robot trajectories of DGS with PCM (figure



(a) DGS optimization with PCM.



(b) DiSCo-SLAM, with full global and local optimization.

Figure 4.11: Optimization result on the KITTI 00 dataset.

4.11a) align well where there are inter-robot loop closures. However, their rotation estimation falls into local minima and introduces errors. figure 4.11b shows the result for DiSCo-SLAM. Since drift accumulates locally due to a lack of intra-robot loop closures, we lower the threshold for PCM and model the covariances of inter-robot loop closure measurements as Cauchy distributions. Although the resulting trajectory aligns with GPS well for most parts, errors occur along the z-axis of "jackal 1" at the ending point due to the lack of intra-robot loop closures.

For the Stevens dataset, GPS measurements along the path are not available, so we transform the trajectory estimates into UTM coordinates and project them onto satellite imagery. Figure 4.12 shows the optimized trajectories using DiSCo-SLAM (figure 4.12b), and DGS with PCM (figure 4.12a). The yellow robot trajectory, aligned with the global frame, is well-optimized in both methods. DGS's estimate of the pink

(a) DGS optimization w/ PCM.

(b) DiSCo-SLAM, with full global and local optimization.

Figure 4.12: Optimization result on the Stevens campus dataset.

robot trajectory drifts as time accumulates, while this robot's trajectory estimate in our DiSCo-SLAM method aligns with the roadways depicted in the imagery.

Table 4.2: RMSE w.r.t GPS for 40 trials

RMSE	Min (m)	Max (m)	Mean (m)	STD (m)
DiSCo-SLAM	**1.01**	**2.51**	**1.52**	**0.34**
DGS with PCM	2.87	14.37	6.23	2.72

Table 4.3: Relative pose estimation error at traj. connecting points

Dataset	**Configure**	**Roll**	**Pitch**	**Yaw**	**Total (∘)**	**X**	**Y**	**Z**	**Total (m)**
Park	DiSCo-SLAM	0.30	0.50	0.60	0.83	0.52	0.95	1.36	1.74
	DGS with PCM	**0.12**	**0.01**	**0.22**	**0.25**	**0.16**	**0.04**	**0.03**	**0.16**
	DiSCo-SLAM	0.85	**2.56**	**1.75**	**3.21**	**0.28**	**0.16**	**0.23**	**0.40**
	DGS with PCM	**0.55**	3.40	3.64	5.01	1.61	1.32	0.45	2.13
KITTI08	DiSCo-SLAM	**1.09**	**0.99**	**0.45**	**1.54**	**3.60**	**4.28**	**0.27**	**5.6**
	DGS with PCM	1.79	1.25	14.03	14.19	22.58	7.82	0.37	23.90
KITTI00	DiSCo-SLAM	**0.38**	2.99	**1.10**	**3.20**	**6.38**	**1.94**	**1.21**	**6.78**
	DGS with PCM	11.38	**1.57**	14.84	18.76	14.92	26.35	14.15	33.42
	DiSCo-SLAM	0.56	7.43	**0.62**	7.48	2.08	**6.74**	14.00	15.67
	DGS with PCM	**0.20**	**0.40**	0.75	**0.87**	**0.67**	11.35	**1.49**	**11.47**
Stevens	DiSCo-SLAM	3.22	5.07	**0.26**	6.01	**0.32**	**0.99**	**0.23**	**1.07**
	DGS with PCM	**1.61**	**3.63**	4.28	**5.84**	0.50	1.28	0.99	1.70

The Park dataset was gathered with GPS signal available throughout, so we compared our method and the DGS method using the GPS data as ground truth. Figures. 4.9d and 4.9c show results from the Park dataset using DGS optimization and our method, respectively. For this dataset, we run multiple trials to examine the robustness of our method. After each trial, the SLAM trajectory estimates are

compared against the GPS data, following the same procedure described in section 4.4.1. Table 4.2 shows the RMSE with respect to GPS ground truth data, across 40 trials of re-playing the same recorded dataset, using both methods. Although the lowest error in all the trials for DGS and our method is close, our method offers a more stable, consistent output. The DGS method's rotation optimization step often hinders convergence to a global minimum under the infrequent arrival of inter-robot constraints, as evidenced by the buildup of drift for "jackal 2" in fugure 4.9d.

Because our RTK-GPS data only covers two translational degrees of freedom, we also compare relative pose estimation error. Since all of our multi-robot datasets are obtained by dividing single-robot datasets into several parts, we use the coincidence of the ending point of one robot's trajectory and the starting point of the next as the basis for quantifying the rotational and translational errors across a representative inter-robot "rendezvous point" from each dataset, which are captured in Table 4.3 for all of our datasets. Although DiSCo-SLAM is not always superior, its worst-case performance is well below the levels occasionally reached by DGS.

Communication and Computational Efficiency

To quantify the bandwidth requirements of the proposed SLAM framework, we have examined the sizes of the messages sent between robots during execution of the datasets. The results for Velodyne VLP-16 and VLP-64 LiDAR are shown in Tables 4.4 and 4.5 respectively, which catalog the mean, minimum, and maximum size of each type of message exchanged, as well as the total quantity of each type of message exchanged, between robots during their execution of the trajectories. We assume there is no maximum communication range, so that messages can be exchanged between robots at any time. While a single laser scan from the Velodyne VLP-16 is 1.04 MB, the message size needed for our DiSCo-SLAM method for each LiDAR keyframe

is around 200 KB.

Table 4.4: Data Sizes of Messages Sent (VLP-16)

Message Info	**Mean (kB)**	**Min (kB)**	**Max (kB)**	**No. Total Msgs.**	
				Stevens	Park
SC Feature & Local Pose	4.08	4.04	4.12	3936	4222
Feature Cloud (Edge)	9.65	5.08	15.90	37	837
Feature Cloud (Planar)	71.31	54.74	85.98	37	837
Feature Cloud (Other)	70.91	50.58	83.99	37	837
Coordinate Transformation	0.70	0.70	0.70	3	711
Inter-Robot Loop Closure	0.12	0.12	0.12	3	72

Table 4.5: Data Sizes of Messages Sent (VLP-64)

Message Info	**Mean (kB)**	**Min (kB)**	**Max (kB)**	**No. Total Msgs.**	
				KITTI00	KITTI08
SC Feature & Local Pose	15.75	15.75	15.75	1134	2333
Feature Cloud (Edge)	30.60	16.76	42.62	198	12
Feature Cloud (Planar)	309.70	242.39	383.03	198	12
Feature Cloud (Other)	89.18	69.46	115.96	198	12
Coordinate Transformation	0.70	0.70	0.70	130	8
Inter-Robot Loop Closure	0.12	0.12	0.12	24	7

Table 4.6 shows the computation time of each key step of DiSCo-SLAM, using the computer described in section 4.4.A. The groupings of rows correspond to the groupings given in Tables IV and V (i.e., the algorithmic steps in one grouping yield the messages in the other). Feature description and matching are, by far, the most frequently performed steps, serving as an efficient filtering mechanism that permits costly point cloud matching to be invoked less frequently.

Table 4.6: Processing Time for Each Algorithmic Step (ms)

Subroutine	Park		KITTI08		KITTI00		Stevens	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
SC Feature Description	<1	9	<1	6	<1	6	<1	10
SC Feature Matching	61	188	34	67	2	68	24	90
Cloud Scan Matching	193	614	112	147	19	147	314	677
Incremental PCM	55	346	7	10	<1	10	5	20
Global Optimization	4	12	<1	1	<1	1	<1	<1
Local Optimization	7	36	2	14	<1	14	7	20

4.4.2 SGM-SLAM

We evaluate the performance of our proposed scene graph matching algorithm using both simulation and real-world datasets. The simulation dataset is used to test the algorithm's capability in performing sufficient graph matching and transformation estimation. The real-world datasets in this paper are collected from a heterogeneous multi-robot setup, which includes a handheld device and a Unitree Go2 EDU, both equipped with LiDAR, an inertial sensor, and a camera.

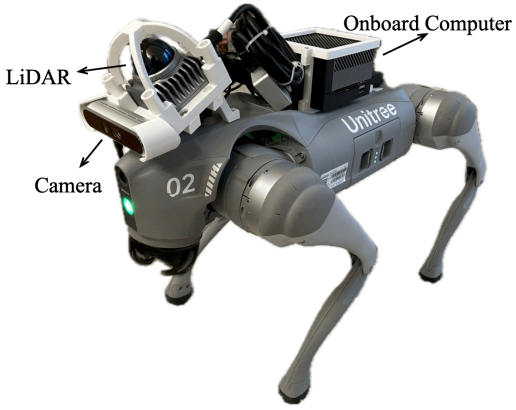


Figure 4.13: Our data-gathering platform consists of a Unitree robot dog with sensors mounted on the top.

As shown in figure 4.13, the handheld device is equipped with a LiVOX mid-360 LiDAR and an Intel RealSense Depth Camera. The onboard sensors of the Unitree robot dog are not utilized in this work; instead, the same handheld device is attached to the robot dog for data collection. The introduction of the Unitree robot dog enhances the mobility of the entire system, enabling the handheld device to gather data in dynamic environments and traverse larger areas than a stationary setup would allow.

We specifically collected this dataset because existing multi-robot SLAM datasets [104] [93] [30] use a VLP-16 LiDAR, which produces a sparse, layered point cloud that does not meet our needs for point cloud segmentation. The data sequences are collected in a suburban campus environment. Due to limitations in equipment availability, we collected sequences for each robot individually as ROS2 bag files and later synchronized them offline by playing them simultaneously. The entire system is implemented using Robot Operating System (ROS) 2 Foxy on Ubuntu 20.04. The GPU is utilized only for image segmentation.

Evaluation of Scene Graph Matching on Synthetic Data

To quantitatively evaluate the robustness of the scene graph-based transformation estimation module, we create a 60m $\times$ 60m environment containing 50 objects, each assigned a label from 6 distinct classes, along with randomly generated center poses and dimensions. This experiment simulates a hypothetical matching between robots, which can be extended to any n-robot scenario. Tab. 4.7 and 4.8 show the success rate of transformation estimation under various conditions. Each experiment is repeated 20 times, with random initial transformations applied to both graphs for consistent results. The transformation estimation is considered successful if the translation error $e_t < 2$m, and the rotation error $e_r < 20^\circ$ on each axis. In both tables, no

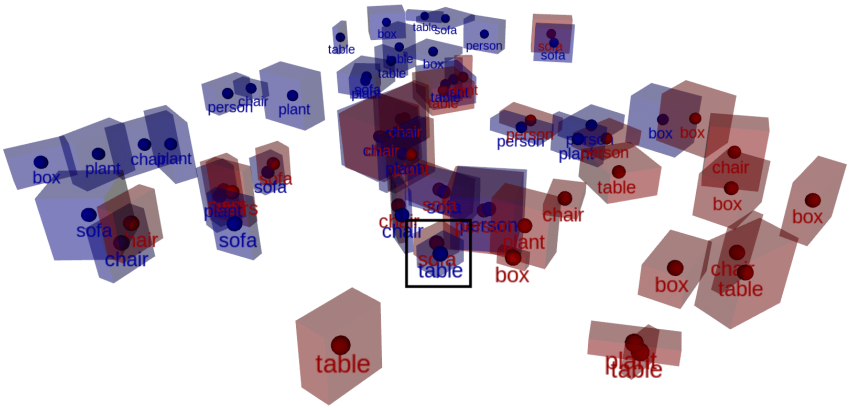


Figure 4.14: An example illustrating a successful transformation estimation despite the presence of both position and semantic label errors. Objects in $\mathcal{G}_1$ are marked with red cuboids, and objects in $\mathcal{G}_2$ are marked with blue cuboids. The objects in $\mathcal{G}_2$ are transformed into the local coordinate system of $\mathcal{G}_1$ using the transformation estimation result. A table in $\mathcal{G}_1$ is misclassified as a sofa (highlighted by the black rectangle), yet the transformation estimation remains correct and serves as a sufficient initial guess.

pose error column indicates the success rate when the true object center pose is used for graph matching and transformation estimation, whereas the pose error column corresponds to the case where the object pose includes random errors following a uniform distribution: $\pm$1m on the x- and y-axes, $\pm$0.5mm on the z-axis, $\pm 2.5^\circ$ for roll and pitch, and $\pm 10^\circ$ for yaw. While the runtime increases as the graph size grows, the proposed algorithm maintains real-time performance.

Table 4.7: **Success Rate** of transformation estimation for various graph sizes and amounts of overlap without semantic label error.

Object Count			**Success Rate (%)**		**Runtime**
$\mathcal{G}_1$	$\mathcal{G}_2$	Overlap	No Pose Error	Pose Error	**(ms)**
8	8	4	100	45	7
15	15	7	100	90	14
35	32	17	100	100	292

Tab. 4.7 presents the success rate for different graph sizes. Object count denotes the number of objects in each scene graph, while overlap refers to the number of overlapping objects between two graphs. Our algorithm succeeds in all cases when no pose error is introduced. When pose error is present, it still achieves a relatively high success rate given a sufficient number of objects in the graph. However, the success rate is significantly lower in the 4-object overlap case, which is challenging due to the limited number of correct matches for transformation estimation.

Table 4.8: **Success Rate** of transformation estimation with various semantic label errors.

Mislabel Count	**Success Rate (%)**		**Runtime**
	No Pose Error	Pose Error	**(ms)**
3	100	100	275
6	100	95	247
9	70	45	250
12	35	25	245

Tab. 4.8 presents the success rate when incorrect semantic classification is also considered. In all cases shown in Tab. 4.8, $\mathcal{G}_1$ contains 35 objects, and $\mathcal{G}_2$ contains 32 objects, with an overlap of 17 objects. Figure 4.14 illustrates a scenario where both pose and semantic errors are present, yet the proposed algorithm successfully handles the matching. The algorithm achieves a high success rate when at least 65% of the semantic labels are correct, regardless of whether pose error is introduced.

SLAM Results over Real-World Data

We compared our SGM-SLAM pipeline with a state-of-the-art LiDAR SLAM method on our self-gathered datasets. Due to equipment limitations, GPS was not available for either the indoor or outdoor dataset. Instead, we used Structure from Motion

(SfM) from COLMAP [87] as the ground truth, since it is a global method that optimizes observations across all images and performs inter-image registration globally. The SfM results ensure accurate inter-robot relative pose estimation, which is crucial for evaluating multi-robot SLAM algorithms. Swarm-SLAM is selected for comparison as it is a ROS2-based distributed method that supports both camera and LiDAR inputs. Its odometry is derived from LiDAR scans using RTAB-Map [59]. Tab. 4.9 shows the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) between our SLAM and SfM results, while figure 4.15 provides a qualitative comparison. "Swarm-SC" denotes the use of the LiDAR descriptor Scan Context [54] for inter-robot data association, while "Swarm-CP" denotes the use of the image descriptor CosPlace [3].

We collected two datasets on the SRI campus. The outdoor dataset consists of two robots covering a larger area and contains sparse objects, while the indoor dataset involves three robots covering a relatively small area with a large number of repeated objects. We use COLMAP to reconstruct only feature-rich areas, ensuring a precise reconstruction result as ground truth. The main challenge in the outdoor dataset is the low overlap ratio between trajectories, requiring robust inter-robot data association. The indoor dataset, on the other hand, is affected by dim lighting conditions. Additionally, both datasets contain repeated feature patterns from objects such as chairs, benches, and tables. To align the trajectories and evaluate error, we use EVO[1]. Since EVO is designed for single-robot usage, we combine the trajectories collected by different robots into a single file and evaluate the results as a whole trajectory. The merged result based on scene graph matching from the outdoor dataset is shown in Figs. 4.16a (side view) and 4.16b (top-down view), while the result from the indoor dataset is shown in figure 4.17.

[1]https://github.com/MichaelGrupp/evo

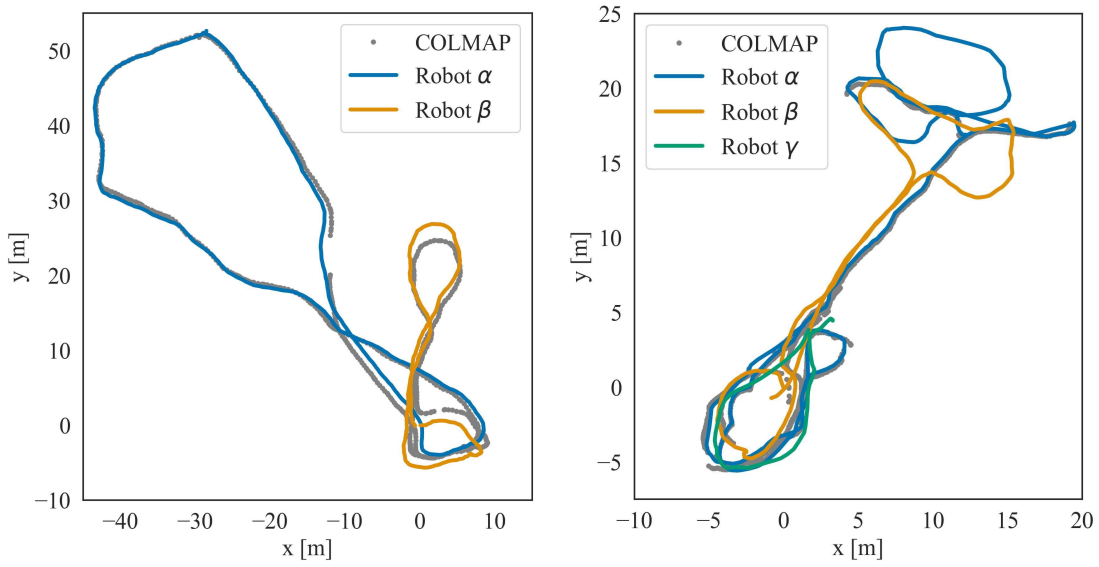


Figure 4.15: SLAM trajectories from multiple robots aligned with pose estimates from COLMAP sparse reconstruction on outdoor (left) and indoor (right) datasets.

Table 4.9: **Absolute Trajectory Error (ATE) [m]** in meters for the proposed method compared to state-of-art methods.

Method	**Swarm-SC**	**Swarm-CP**	**Ours**
Outdoor	20.29	3.92	1.32
Indoor	3.36	5.70	0.81

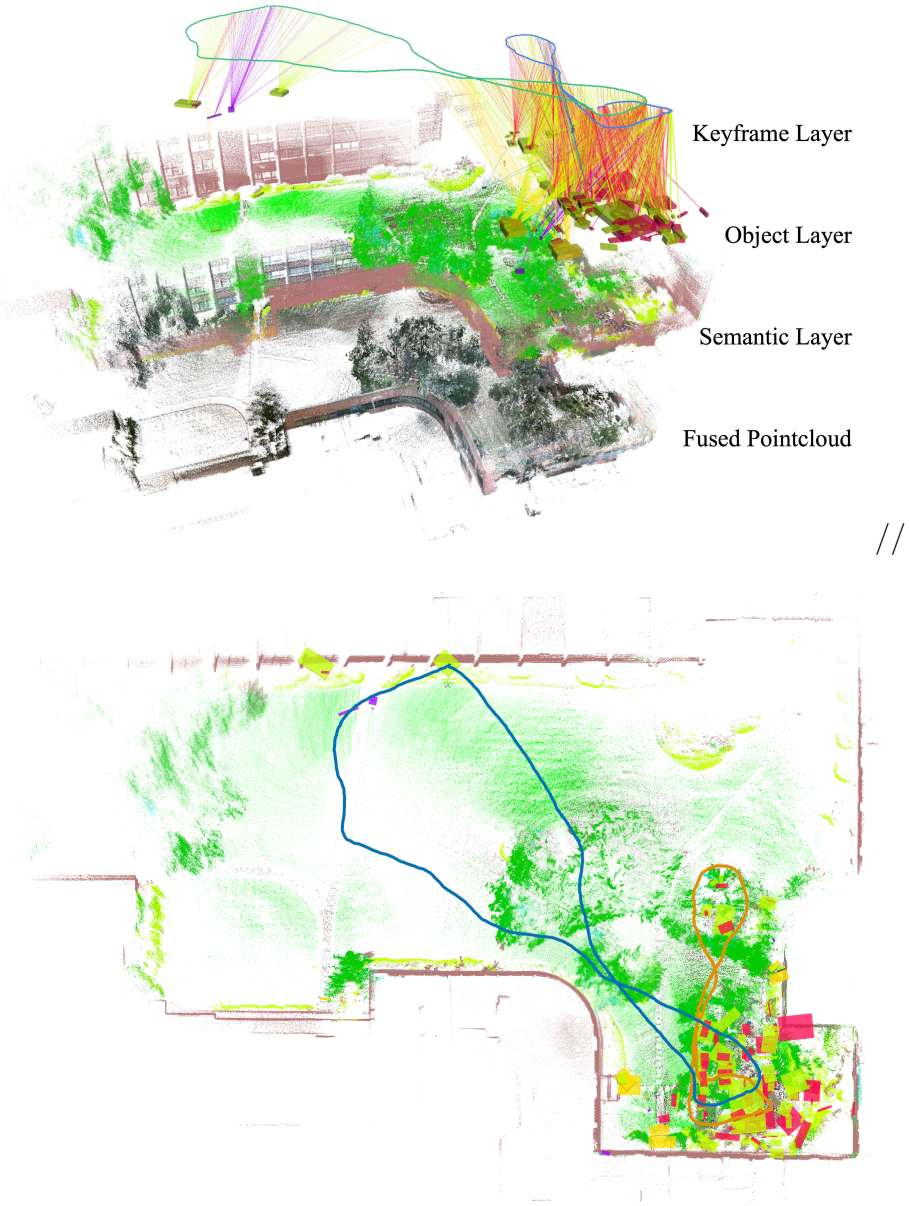


Figure 4.16: Top-down view (top) and side view (bottom) of the merged result from the two-robot dataset collected in an outdoor area of the SRI campus.

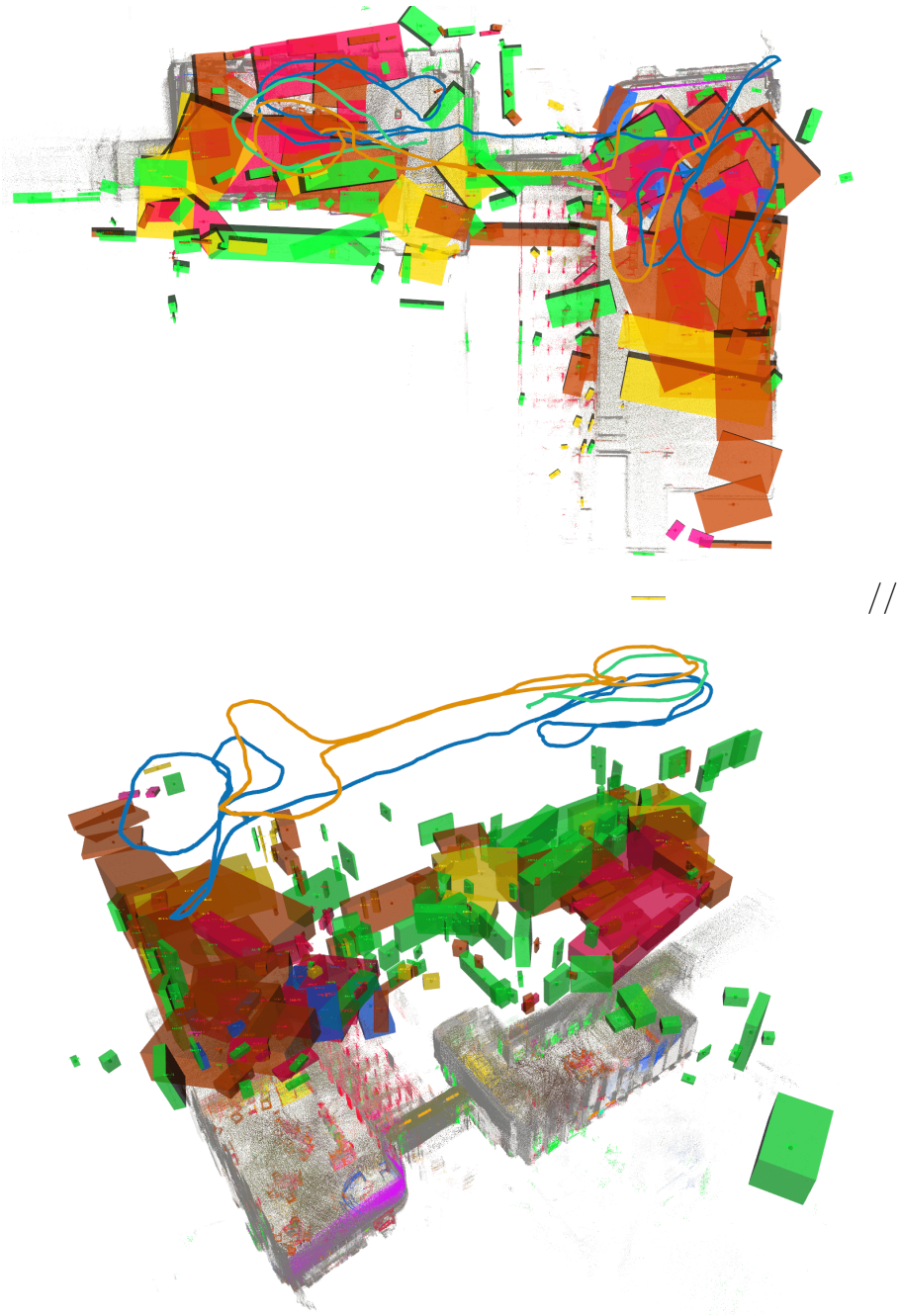


Figure 4.17: Top-down view (top) and side view (bottom) of the merged result from the three-robot dataset collected in an indoor area of the SRI campus.

Communication Analysis

We next analyze the size and quantity of perception messages exchanged in our real-world datasets, which employ simulated wireless communication. We summarize the size of perception messages for the proposed method in Tab. 4.10. The "Point-cloud Raw" baseline we compared against refers to a brute-force centralized strategy, where all raw point clouds are transmitted to a centralized server. To further reduce communication overhead, only object node information is shared, which includes the node ID, center position, dimensions, semantic information, and color. The edges are computed locally based on the object node information. Additionally, the sparse object point cloud is transmitted upon request once object-level matching is determined, optimizing communication efficiency.

Once a relative graph transformation estimation is computed, the relevant keyframes are requested from the robot neighbors. A keyframe is only sent after the requesting robot confirms it has not been sent previously. The point cloud keyframe message is encoded using PCL [78] to minimize its size.

The results in Tab. 4.10 show that the proposed method is highly data-efficient, with the geometric priors provided by graph matching effectively reducing the need to exchange most point cloud data.

Table 4.10: **Sizes of Perception Messages [KB]** for the proposed method.

Message Type	**Outdoor**			**Indoor**		
	Mean	Max	No.	Mean	Max	No.
Point-cloud Raw	508.2	516.9	3073	508.1	512.1	3079
Scene Graph	7.76	16.93	701	18.40	44.37	845
Keyframe Request	0.02	0.02	61	0.02	0.02	411
Keyframe Response	18.87	22.95	10	6.90	11.05	72

4.5 Conclusions

In this chapter, we have presented DiSCo-SLAM and SGM-SLAM, two distributed multi-robot SLAM frameworks for 3D LiDAR observations. Both feature relatively low communication bandwidth for message passing.

In DiSCo-SLAM, LiDAR scans are efficiently described using Scan Context descriptors and shared between robots. We also propose a two-stage global-local graph optimization procedure that offers robust output for relatively large scale multi-robot SLAM problems with limited occurrences of rendezvous, finding transformations relating robots that may be distant from one another. We compare our optimization strategy with the widely used distributed Gauss-Seidel method, showing the relative stability of our method.

In SGM-SLAM, we extract semantic information from LiDAR and camera data, and construct a multi-level scene graph collaboratively with the multi-robot team. The introduction of object-level graph matching improves the generality, robustness, and communication efficiency of our distributed SLAM system in both indoor and outdoor environments.

Chapter 5

Multi-Robot Underwater Simultaneous Localization and Mapping

We provide a comprehensive introduction to DRACo-SLAM2, our proposed multi-robot SLAM framework utilizing object graph matching. Figure 5.1 presents the complete pipeline of the proposed framework. We define the local robot as robot α. The inter-robot data association process begins by clustering the latest point-cloud map, $\mathcal{M}\alpha$, generated by the local SLAM algorithm into an object map, $\mathcal{V}\alpha$. The object map is then shared among the robot team.

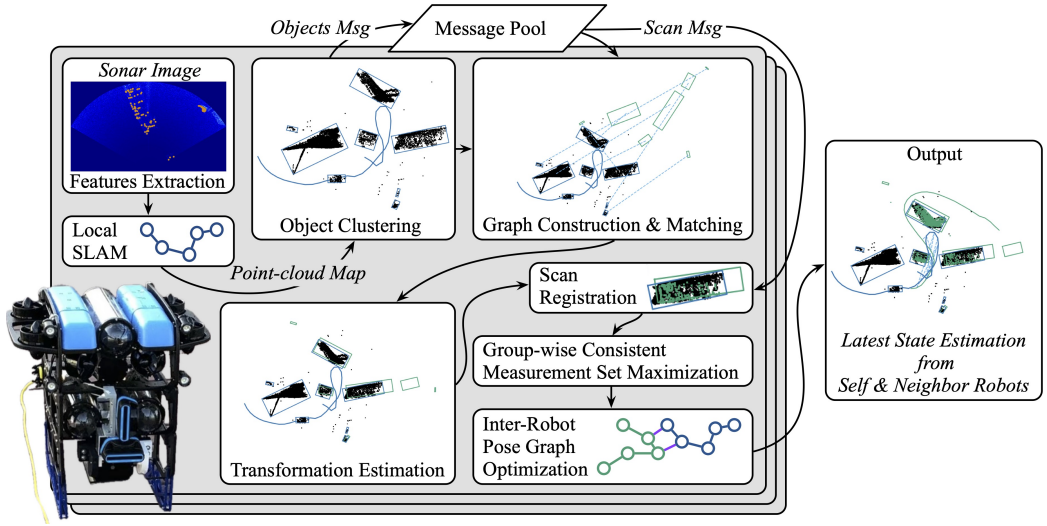


Figure 5.1: **Overview of DRACo-SLAM2 Architecture.** Each robot's object map is clustered using DBSCAN. The local robot receives the neighboring robot's object map, aligns it to its local map using graph matching, and requests scans for ICP registration with the graph matching transformation as an initial guess. Inter-robot loop closures are then added to the pose graph for the two-step pose graph optimization (PGO). The local robot's trajectory and object vertices are shown in blue, while the neighboring robot's data is in green. Dashed lines indicate detected correspondences between objects in different maps.

When the local robot receives an object map, $\mathcal{V}\beta$, from a neighboring robot β, the object maps are matched through object graph matching. A detailed description

of the object graph matching process is provided in section 5.1.2. Feature points from the scans corresponding to overlapping objects are then requested from robot β for performing scan registration. The transformation $\mathbf{T}_{\alpha}^{\beta}$, representing the coordinate transformation from $\mathcal{V}\beta$ to $\mathcal{V}\alpha$, serves as the initial estimate for scan registration, as detailed in section 5.1.3.

After registration, incremental group-wise consistent measurement set maximization (GCM) is applied to ensure robustness by mitigating errors caused by perceptual aliasing. All loop closures that pass the GCM process are subsequently added to the factor graph. A two-step inter-robot pose graph optimization is then performed to achieve a stable result, following the process presented in section 4.1. All initial guesses for both local and global PGO are obtained by applying map coordinate transformations based on object graph matching.

Inter-robot loop closures are added to the factor graph using a robust noise model to ensure accurate and reliable association. By adopting this two-step PGO approach, we enable faster convergence of the factor graph optimization. This is achieved by updating only a subset of the information at each step, rather than recalculating the entire graph, which reduces computational overhead and accelerates the optimization process.

5.1 Local SLAM, Point-cloud Map and Object Map

5.1.1 Local SLAM

We choose Bruce-SLAM [96] as our local SLAM framework. Bruce-SLAM is a sonar SLAM algorithm that uses data exclusively from forward-looking sonar (FLS) and vehicle dead-reckoning measurements. However, the flexibility of our proposed DRACo-SLAM2 system allows for seamless integration with any local sonar SLAM algorithm.

As a graph-based SLAM algorithm, Bruce SLAM formulates the SLAM optimization problem as a maximum a posteriori (MAP) estimation problem, as shown in equation 2.2.4.

Point-cloud map

To perform sequential scan matching and intra-robot data association, feature points are extracted from sonar images using SOCA-CFAR [27], a variant of the Constant False Alarm Rate (CFAR) technique [82]. Let the set of feature points detected from the sonar image captured at timestamp i by robot α be denoted as $\mathcal{F}_{\alpha_i}$. These points are transformed from the sensor frame into the local frame of robot α, with state $\mathbf{x}_{\alpha_i} \in \mathcal{X}_\alpha$, and are represented as ${}_\alpha\mathcal{F}_{\alpha_i}$. A local point-cloud map for robot α, shown as black points in figure 5.1, is thus defined as $\mathcal{M}_\alpha = \{{}_\alpha\mathcal{F}_{\alpha_i} \mid i \in [0, t]\}$, where t represents the current timestamp.

Object map

The DBSCAN clustering algorithm [28] is applied to cluster objects from the latest local point-cloud map, $\mathcal{M}_\alpha$. For each cluster, a bounding rectangle is computed based on the positions of the points in the cluster. The bounding rectangles of local objects are shown in blue in figure 5.1. Each object, denoted as $\mathbf{v}_{\alpha_i}$, is described by the center coordinates $(x_{\alpha_i}, y_{\alpha_i})$ and the dimensions (length and breadth) $(l_{\alpha_i}, b_{\alpha_i})$ of its bounding rectangle. To filter out noise misidentified as objects, only clusters with a number of points $n_p > n_{\min}$ and a dimension $\max(l_{\alpha_i}, b_{\alpha_i}) > d_{\min}$ are accepted. The set of all accepted objects forms the object map, $\mathcal{V}_\alpha$.

5.1.2 Object Graph Construction and Matching

When the object map $\mathcal{V}_\beta$ is received from the neighboring robot β, object graph $\mathcal{G}_\beta = (\mathcal{V}_\beta, \mathcal{E}_\beta)$ is constructed using $\mathcal{V}_\beta$ as vertices in the graph. $\mathcal{G}_\beta$ is a directed complete graph, and $\mathcal{E}_\beta$ is further defined as:

$$\mathcal{E}_\beta = \{\mathbf{e}_{ij} = (\mathbf{v}_i, \mathbf{v}_j, w_{ij}) \mid \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}_\beta\}, \tag{5.1.1}$$

where w_{ij} is the Euclidean distance between the centers of objects $\mathbf{v}_i$ and $\mathbf{v}_j$. A local object graph $\mathcal{G}_\alpha$ is constructed in the same manner. Thus, the object map matching problem is converted into a bipartite graph matching problem between $\mathcal{G}_\alpha$ and $\mathcal{G}_\beta$.

Inspired by SemanticLoop [100], we formulate this bipartite graph matching problem as a Quadratic Assignment Problem (QAP), as shown in equation 4.3.4.

One key difference between our method and SemanticLoop [100] is the calculation of the utility function $u(\cdot)$:

$$u(\mathbf{e}_{ij}, \mathbf{e}_{kl}) = \exp\left(-\mu|w_{ij} - w_{kl}| - |l_i - l_k| - |b_i - b_k|\right). \tag{5.1.2}$$

While SemanticLoop considers the semantic classes of objects and the Euclidean distances between their centers, our approach compares object dimensions (l_i, b_i) and corresponding edge weights, with $\mu = 4$ serving as a scaling factor. We follow the same steps outlined in section 4.3.2 to solve this QAP problem.

5.1.3 Scan Registration

When more than three matched object pairs are obtained from the graph matching process in section 5.1.2, the coordinate transformation $\mathbf{T}_\alpha^\beta$ from $\mathcal{V}\beta$ to $\mathcal{V}\alpha$ is computed using affine transformation estimation. To ensure robustness, we apply RANSAC and

accept the transformation only if it has more than four inliers. With the transformation $\mathbf{T}_{\alpha}^{\beta}$ estimated from object maps as geometric priors, we apply ICP registration in a manner similar to its application in single-robot SLAM [96]. We use a sliding window for the target cloud to improve the registration results. Figure 5.2 illustrates the source and target clouds before and after ICP registration, demonstrating that the initial guess provided by object graph matching is sufficient for successful ICP registration.

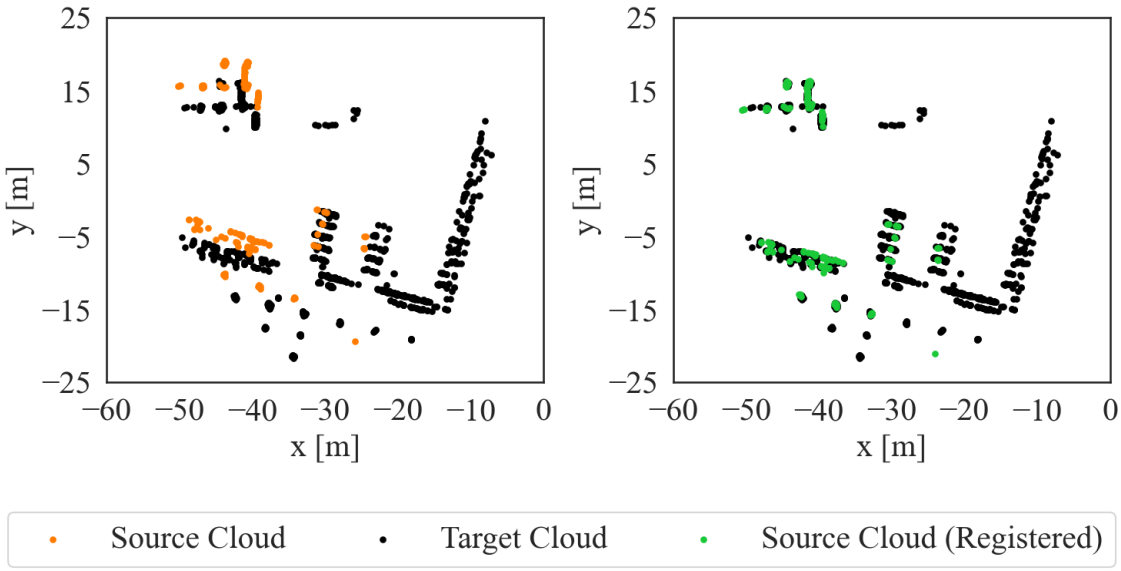


Figure 5.2: **Graph matching-based ICP registration.** The source cloud (orange) is roughly transformed into the local robot coordinate system using the transformation estimated from object graph matching. It is then aligned with the target cloud (black) using ICP with a sliding window of size 3. The registered source cloud is shown in green.

5.1.4 Group-wise Consistent Measurement Set Maximization

A challenging aspect of scan registration is the fact that separate pairs of scans located in close proximity to one another can give rise to similar registration errors, as illustrated in Figure 5.3. For sonar scan registration, the overlap percentage between

the source and target point clouds is an intuitive indicator of quality. However, as we will discuss in Section 5.2.1, even a high overlap ratio does not always guarantee minimal registration error. While this is acceptable when using a global PCM, it can lead to errors when only incremental PCM is applied. To address these challenges, we propose a modified approach that we denote Group-wise Consistent Set Measurement Maximization (GCM), inspired by PCM.

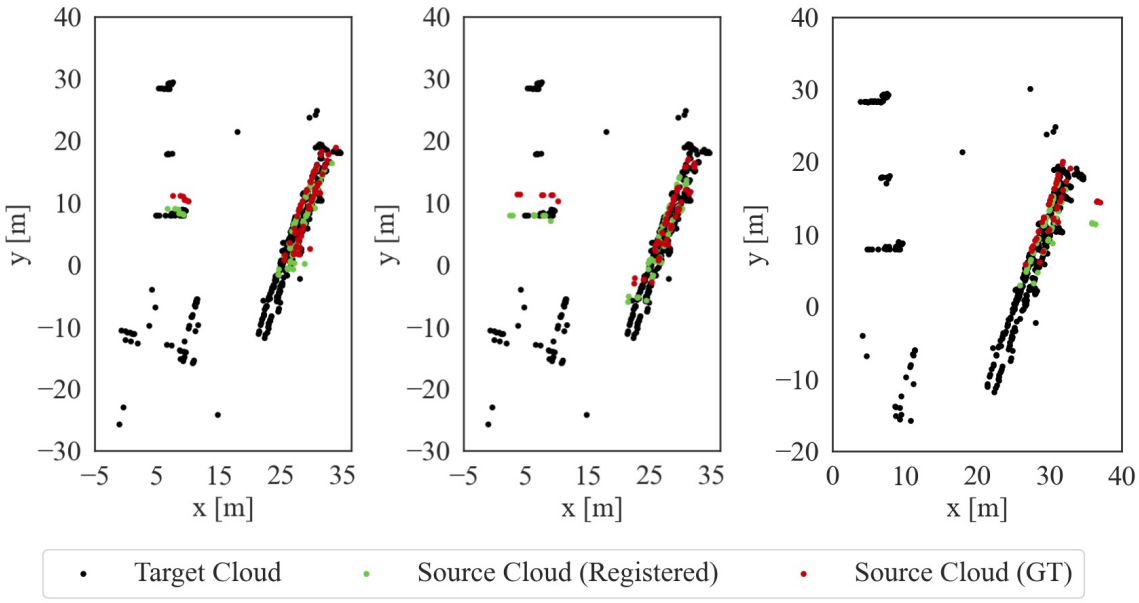


Figure 5.3: **Motivation for use of GCM.** A series of three inter-robot loop closures, in the same region of the environment (from our USMMA dataset depicted in figure 5.7), are impacted by similar point cloud registration errors. Each erroneous registration result is shown in green, with ground truth shown in red.

In PCM [65], the consistency of observations between a pair of robots is considered, as illustrated in figure 5.4a. For two inter-robot loop closure observations, $\mathbf{z}_{\alpha_k}^{\beta_i}$ and $\mathbf{z}_{\alpha_l}^{\beta_j}$, involving robot α and robot β, the method evaluates their consistency using:

$$C(\mathbf{z}_{\alpha_k}^{\beta_i}, \mathbf{z}_{\alpha_l}^{\beta_j}) = \left\| (\mathbf{z}_{\alpha_k}^{\beta_i})^{-1} \cdot \hat{\mathbf{x}}_{\beta_j}^{\beta_i} \cdot \mathbf{z}_{\alpha_l}^{\beta_j} \cdot \hat{\mathbf{x}}_{\alpha_k}^{\alpha_l} \right\|. \tag{5.1.3}$$

Where $\hat{\mathbf{x}}_{\beta_j}^{\beta_i}$ and $\hat{\mathbf{x}}_{\alpha_k}^{\alpha_l}$ are the relative state estimations marginalized from the factor

graph optimization of local SLAM.

In the proposed GCM, we consider the consistency of observations among a group of robots (figure 5.4b). For two inter robot loop closure observations, $\mathbf{z}_{\alpha_k}^{\beta_i}$ between local robot α and local robot β and $\mathbf{z}_{\alpha_l}^{\gamma_j}$ between robot α and robot γ, we perform:

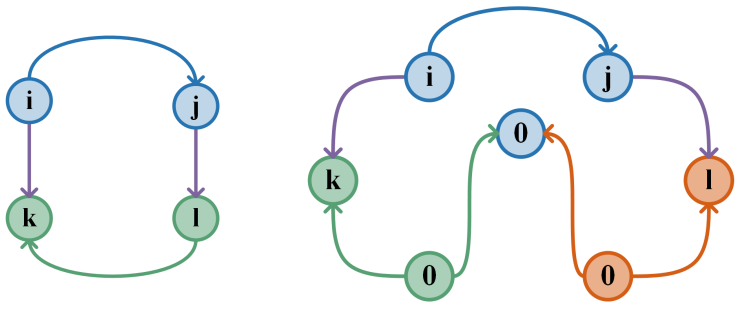
$$C(\mathbf{z}_{\alpha_k}^{\beta_i}, \mathbf{z}_{\alpha_l}^{\gamma_j}) = \left\| (\mathbf{z}_{\alpha_k}^{\beta_i})^{-1} \cdot \hat{\mathbf{x}}_{\gamma_j}^{\beta_i} \cdot \mathbf{z}_{\alpha_l}^{\gamma_j} \cdot \hat{\mathbf{x}}_{\alpha_k}^{\alpha_l} \right\|. \tag{5.1.4}$$

Robot α is considered the local robot, so $\hat{\mathbf{x}}_{\alpha_k}^{\alpha_l}$ can be obtained by marginalizing the factor graph optimization of local SLAM. We further expand $\hat{\mathbf{x}}_{\beta_k}^{\gamma_l}$:

$$\hat{\mathbf{x}}_{\gamma_j}^{\beta_i} = (\hat{\mathbf{x}}_{\beta_i}^{\alpha_0})^{-1} \cdot \hat{\mathbf{x}}_{\gamma_l}^{\alpha_0}, \tag{5.1.5}$$

$$= (\hat{\mathbf{x}}_{\beta_i}^{\beta_0})^{-1} \cdot \hat{\mathbf{x}}_{\alpha_0}^{\beta_0} \cdot (\hat{\mathbf{x}}_{\alpha_0}^{\gamma_0})^{-1} \cdot \hat{\mathbf{x}}_{\gamma_j}^{\gamma_0}. \tag{5.1.6}$$

The relative state estimations $\hat{\mathbf{x}}_{\gamma_l}^{\gamma_0}$ and $\hat{\mathbf{x}}_{\beta_k}^{\beta_0}$ can be marginalized from the local factor graph optimization of neighboring robots, respectively. The relative state estimations $\hat{x}_{\alpha_0}^{\beta_0}$ and $\hat{x}_{\alpha_0}^{\gamma_0}$ are the optimized map coordinate transformations obtained from historically accepted loop closures.



(a) PCM (b) GCM

Figure 5.4: **Illustration comparing PCM and the proposed GCM**. Circles represent pose estimations, arrows indicate measurements, and purple arrows highlight inter-robot loop closure measurements.

5.1.5 Communication

In this section, we summarize the communication overhead between the local robot and its neighboring robots. As depicted in figure 5.1, at each sonar timestep, the object map is shared with the robot team as an objects message. Additionally, pose estimations are incrementally transmitted to neighboring robots.

To optimize the use of communication bandwidth, updates to the historical state estimations for neighboring robots are only performed when significant changes result from loop closures. Once the graphs are matched, the sonar scans associated with the matched objects are requested from the neighboring robots. After the detection of inter-robot loop closures, this information is also shared with the robot neighbors.

5.2 Experiments

Experimental Setup

We evaluate DRACo-SLAM2 using both real-world and simulated data. The real-world data, shown in figure 5.7, was collected using our customized BlueROV2-Heavy (illustrated in figure 5.1) at the U.S. Merchant Marine Academy (USMMA), King's Point, NY. Details of the BlueROV2-Heavy configuration can be found in our previous work [66]. For this study, we utilize the robot's horizontally-oriented Oculus M750d sonar, along with a Rowe SeaPilot DVL and a VectorNav VN100 MEMS IMU. Two fully simulated datasets, the USMMA dataset and the airplane dataset, were generated using HoloOcean [76]. The USMMA dataset replicates key features of the environment where the real USMMA dataset was collected, including floating docks and repeating circular pier pilings. The airplane dataset is intended to simulate the site of an airplane wreck located on the seafloor (this dataset is shown in figure 5.1).

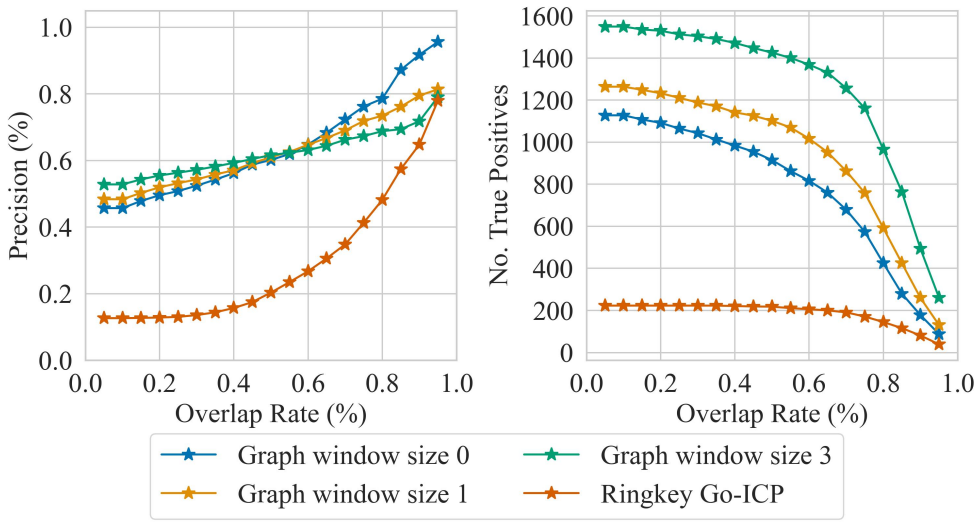
In all datasets, both real and synthetic, all robots operate at the same, fixed depth (close to the surface in the USMMA datasets, and close to the seafloor in the airplane dataset). To simulate the simultaneous operation of multiple robots, we recorded several dataset sessions and replayed them concurrently on a single computer (for both our real and synthetic datasets), with simulated communications between robots. All experiments were conducted on a single computer equipped with an Intel Xeon E-2276M CPU, 31.1 GB of memory, and running Ubuntu 20.04 with ROS Noetic.

5.2.1 Performance of Inter-Robot Loop Closure Detection

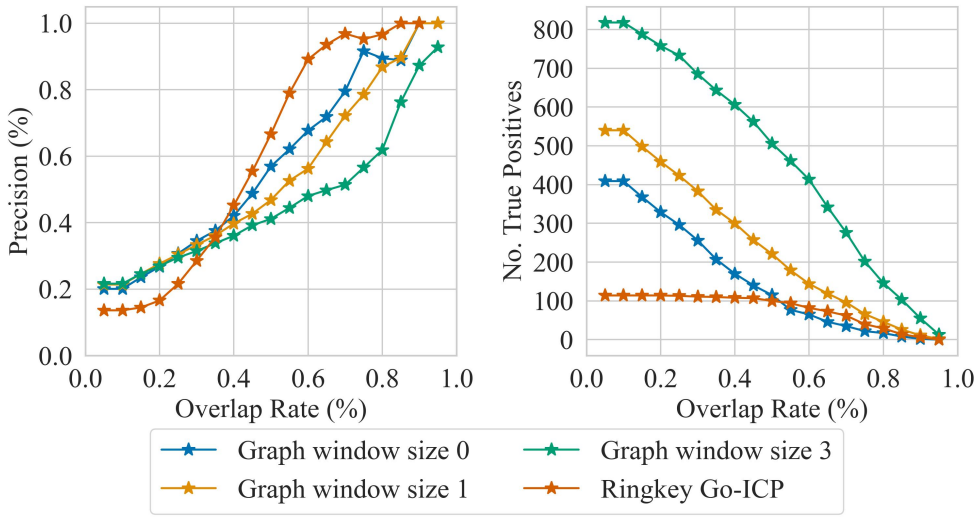
In this section, we evaluate the performance of our object-graph-matching-based inter-robot data association technique. figure 5.5 shows the precision (left) and the number of correct loop closures detected (right) as a function of the minimum acceptable overlap ratio on our two fully simulated datasets.

The overlap ratio, $r_{\text{overlap}} = \frac{n_{\text{overlap}}}{n_{\text{total}}}$, measures the proportion of points in the target cloud (n_{overlap}) that overlap with the source cloud, relative to the total number of points in the target cloud (n_{total}). A loop closure is considered a true positive if the estimated translation error is within 1.5 m and the angular error is within 15° of the ground truth. Precision is defined as the ratio of true positives to the total detected loop closures.

We tested the proposed algorithm using different sliding window sizes. A window size of 0 indicates no sliding window, while sizes 1 and 3 correspond to the inclusion of 3 and 7 nearby frames for registration, respectively. Across both datasets, the method with a sliding window of size 3 achieved the highest number of true positive loop closures. However, its precision decreased when the minimum overlap ratio threshold was relatively low, which is expected due to the sliding window strategy. To balance these factors, we set the minimum acceptable overlap ratio



(a) Result on the airplane dataset.



(b) Result on the USMMA dataset.

Figure 5.5: **Performance** of inter-robot loop closure detection and registration on both of our 3-robot fully simulated datasets. Precision (left) and the number of true positive loop closures detected (right) plotted against the overlap ratio parameter $r_{overlap}$, evaluated using various methods.

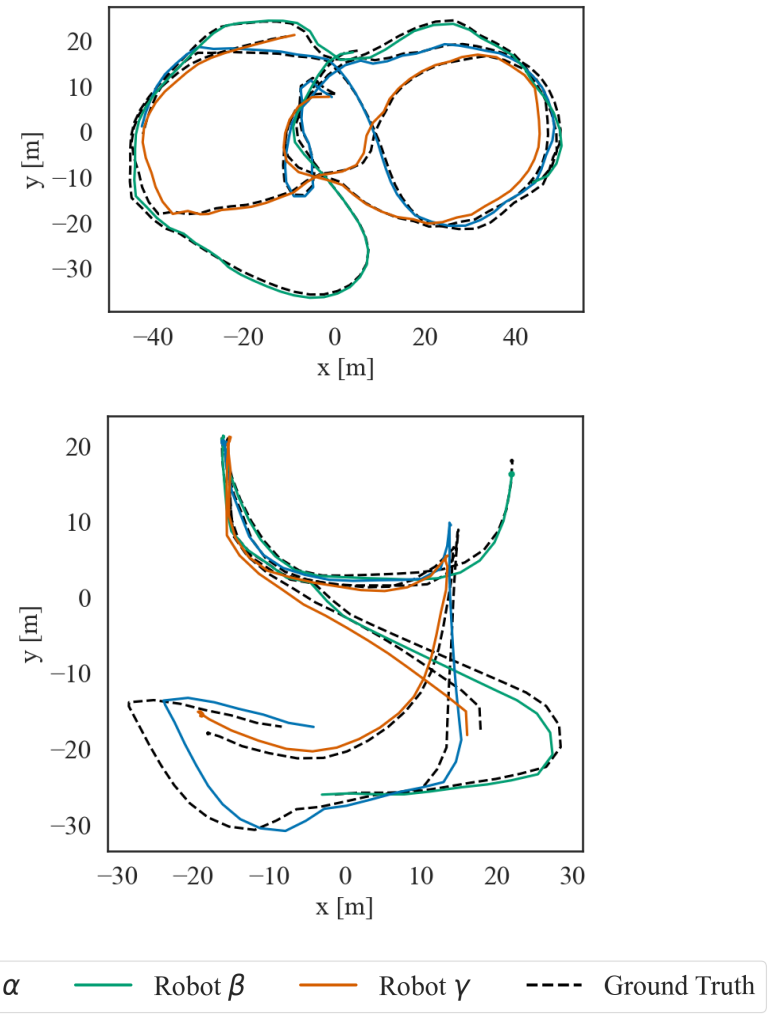


Figure 5.6: **Optimized trajectories** of different local robots using the proposed DRACo-SLAM2 on the fully simulated 3-robot airplane (top) and USMMA (bottom) datasets.

threshold to $\epsilon_{\text{overlap}} = 0.9$ for all experiments and accepted a loop closure only if $r_{\text{overlap}} > \epsilon_{\text{overlap}}$.

Table 5.1: **Runtime** for sonar scene descriptor-based inter-robot loop closure and global ICP registration in DRACo-SLAM [66] (referred to as DRACo1), and graph-based inter-robot loop closure detection and ICP registration in the proposed DRACo-SLAM2 (referred to as DRACo2) on both of our 3-robot fully simulated datasets.

Algorithm		**Runtime (ms)**	
		Mean	**Max**
DRACo1	Ring Key Matching	0.12	2.05
	Go-ICP Registration	361.83	2556.38
DRACo2	Graph Construction & Matching	5.64	32.76
	ICP Registration (Window Size 3)	12.69	358.45

We also summarize the runtime of our proposed DRACo-SLAM2 method (referred to as DRACo2) and its predecessor method, DRACo-SLAM (referred to as DRACo1) in Tab. 5.1. As shown in Tab. 5.1, the ICP registration module in DRACo2 is, on average, 20 times faster than that in DRACo1 because the time-consuming global registration step is not required in our approach. This enables us to operate at a much higher frequency and detect significantly more loop closures compared to the original method. As shown in Tab. 5.2, our total loop closure detection module achieves a runtime per timestep that is 10 times faster than DRACo1, enabling more time-efficient performance.

5.2.2 Performance of Inter-Robot PGO

We also evaluate the performance of inter-robot PGO using both PCM and GCM on our two 3-robot fully simulated datasets. A robust Cauchy noise model [63] from GTSAM [17] is employed for all methods to ensure robust performance. Figure 5.6

Table 5.2: **Mean runtime and mean number of inter-robot loop closure algorithm executions per time step** for DRACo1 and DRACo2 on both of our 3-robot fully simulated datasets.

Algorithm		Mean (s)	Mean No.
DRACo1	Ring Key Matching	< 0.01	1
	Go-ICP Registration	10.11	28
DRACo2	Graph Construction & Matching	< 0.01	1
	ICP Registration (Window Size 3)	1.55	122

shows the trajectories optimized locally from robot α on the airplane dataset (left) and the USMMA dataset (right). We align the estimated trajectory with the ground truth using EVO [1], treating the robot trajectories from different robots as a single trajectory with different timestamps. The trajectories are generally smooth, although drifts caused by local SLAM are observed in the results for the USMMA dataset.

Table 5.3: **Absolute Trajectory Error (ATE)** in meters for the proposed method compared to the full Pose Graph Optimization (PGO) method on both of our 3-robot fully simulated datasets.

Algorithm		USMMA			Airplane		
		α	β	γ	α	β	γ
DRACo	PCM	1.58	2.43	1.88	1.32	1.36	1.46
	GCM	1.43	**1.20**	**1.23**	1.33	**0.95**	**1.31**
Full PGO	PCM	1.27	1.37	1.52	**1.28**	1.91	1.63
	GCM	**1.24**	1.41	1.56	1.31	1.91	1.65

Additionally, we perform a quantitative analysis. Tab. 5.3 presents the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) for our proposed two-step PGO method (referred to as DRACo) compared to the widely used full

[1] https://github.com/MichaelGrupp/evo

PGO with PCM and the proposed GCM. The ATE is also calculated using EVO. For each robot, the estimated and ground truth trajectories are concatenated head-to-tail, starting with the trajectory of the local robot, followed by the trajectories of neighboring robots in alphabetical order. Since EVO is designed for 6DoF SLAM, we adapt it for the 3DoF condition by setting $z = 0$, $\phi = 0$, and $\theta = 0$.

As shown in Tab. 5.3, our default configuration, DRACo two-step PGO with GCM, achieves the best accuracy in most cases. This is attributed to the introduction of GCM, which efficiently reduces the influence of loop closures with similar registration errors, an issue that remains inevitable even with a robust noise model. Furthermore, the adoption of the two-step PGO prevents drifts from neighboring robots from affecting the results of the local robot. However, as a trade-off, the improved accuracy from neighboring robots has only a limited influence on the state estimation of the local robot.

5.2.3 Performance on the Real-world Dataset

Table 5.4: **Sizes of perception messages** for DRACo1 and DRACo2 on the US-MMA real-world dataset (with simulated 3-robot comms.).

Algorithm	**Message Type**	**Mean KBits**	**Max KBits**
DRACo1	Ring key Descriptor	0.13	0.13
	Point Cloud-float32	9.67	27.90
DRACo2	Object Map	1.35	2.25
	Point Cloud-float32	9.69	27.90

The proposed algorithm is next evaluated on our real-world dataset from US-MMA with simulated 3-robot communications. Figure 5.7 illustrates the optimized trajectories, with results from robots α, β, and γ marked in blue, green, and orange,

respectively. All detected inter-robot loop closures are indicated in purple for clarity. Since ground truth data is unavailable for this real-world dataset, we align the constructed point-cloud map with satellite imagery for a more intuitive and visually interpretable representation of the results.

Tab. 5.4 compares the sizes of perception messages transmitted by the full version of DRACo-SLAM (DRACo1) and DRACo-SLAM2 (DRACo2) on the USMMA real-world dataset. For DRACo1, the Ring Key Descriptor messages used for initial loop closure candidate detection are compact, with both mean and maximum sizes under 1 KBits. Object Map messages introduced by DRACo2 are larger but remain well below the 62.5 kbps bandwidth limit of HS underwater acoustic modems operating at a range of 300m. The communication bandwidth required for point-cloud transmission is similar for both methods.

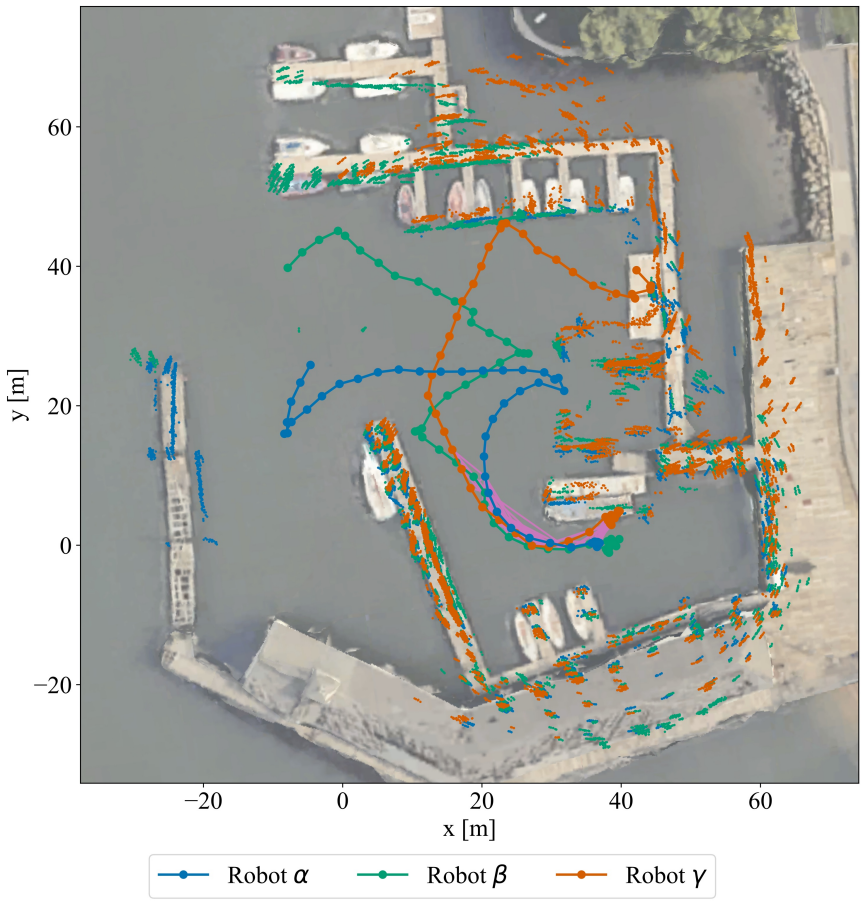


Figure 5.7: **Example DRACo-SLAM2 result with real sonar data.** Optimized trajectories and point clouds from three robots using the proposed DRACo-SLAM2 on a dataset collected at the U.S. Merchant Marine Academy, King's Point, NY, aligned with a satellite image. Inter-robot measurement constraints are shown in purple.

Chapter 6

Multi-Robot Exploration with Expectation-Maximization

6.1 Expectation-Maximization Exploration

We address an autonomous exploration problem that is tightly coupled with a landmark-based SLAM factor graph for a team of n robots. We make the assumption that the initial states of all robots are sufficiently close to each other, enabling mutual observation among group members and facilitating an efficient map initialization process. Additionally, we impose a boundary on the exploration task, where the exploration process terminates upon fully exploring the enclosed environment.

During the exploration process, when a robot α reaches its current target state $\mathbf{a}_\alpha^{\text{this}}$, it becomes necessary to select a new target state $\mathbf{a}_\alpha^{\text{next}}$ from a set of potential new states $\mathcal{A}_\alpha^{\text{next}}$. Building upon our previous research on single-robot exploration [95], we consider a frontier-based strategy that incorporates two key factors: efficient task allocation among robots [7] and the maintenance of a low-uncertainty map.

When α reaches target state $\mathbf{a}_\alpha^{\text{this}}$, $\forall \mathbf{a}_\alpha^{\text{next}\prime} \in \mathcal{A}_\alpha^{\text{next}}$, we define:

$$\mathcal{X}^{\text{new}} = \mathcal{X}^{\text{old}} \cup \mathcal{X}^{\text{predict}} \cup \mathcal{X}^{\text{next}}. \tag{6.1.1}$$

$\mathcal{X}^{\text{old}}$ contains the historical states of all n robots, $\mathcal{X}^{\text{predict}}$ denotes the set of predicted robot states for each current target state $\{\mathbf{a}_i^{\text{this}} | i \in N, i \neq \alpha\}$ and $\mathcal{X}^{\text{next}}$ represents the state sequence of robot α resulting from $\mathbf{a}_\alpha^{\text{next}\prime}$. A classification EM algorithm is used to predict the change of map uncertainty due to $\mathbf{a}_\alpha^{\text{next}\prime}$. To avoid the exponential expansion in potential virtual landmark states, we substitute the **E-step** with a classification step (**C-step**), in which we construct a virtual map using the historical

data of our robot team:

$$\mathcal{V}^* = \arg\max_{\mathcal{V}} P(\mathcal{V}|\mathcal{X}^{\text{old}}, \mathcal{Z}^{\text{old}}), \tag{6.1.2}$$

$$= V(\mathcal{X}^{\text{old}^*}, \mathcal{Z}^{\text{old}}). \tag{6.1.3}$$

Here, $\mathcal{Z}^{\text{old}}$ represents the observations associated with $\mathcal{X}^{\text{old}}$. $V(\cdot)$ is the inverse observation model used to estimate the mean and covariance of the virtual map with the given SLAM estimate $\mathcal{X}^{\text{old}^*}$. For a comprehensive explanation of the virtual map construction process, please refer to our prior work [96]. Subsequently, in the **M-step**, we perform another maximum a posteriori estimation to estimate $\mathcal{X}^{\text{new}}$ (and choose $\mathcal{X}^{\text{next}}$):

$$\mathcal{X}^{\text{new}*} = \arg\max_{\mathcal{X}^{\text{new}}} P(\mathcal{X}^{\text{new}}|\mathcal{V}^*, \mathcal{Z}^{\text{new}}), \tag{6.1.4}$$

$$\mathcal{Z}^{\text{new}} = \mathcal{Z}^{\text{old}} \cup \mathcal{Z}^{\text{predict}}. \tag{6.1.5}$$

The updated observation set, $\mathcal{Z}^{\text{new}}$, is a combination of both previously gathered observations, $\mathcal{Z}^{\text{old}}$, and anticipated future observations, $\mathcal{Z}^{\text{predict}}$. These predicted observations are determined via the measurement models outlined in equation 2.2.1, equation 2.2.2, and equation 2.2.3, based on the virtual map $\mathcal{V}^*$ and the target state assigned to each robot. Thus, we assess the overall local map uncertainty of robot α by computing the sum of the individual uncertainties for each map element in the virtual map $\mathcal{V}(\cdot)$ derived from equation 6.1.3, considering only the optimized states

of robot α, $\mathcal{X}_{\alpha}^{\text{new}*}$:

$$U_M = \phi(\Sigma_{V(\mathcal{X}_{\alpha}^{\text{new}*}, \mathcal{Z}^{\text{new}})}), \tag{6.1.6}$$

$$= \sum_{\mathbf{v_i} \in V(\mathcal{X}_{\alpha}^{\text{new}*}, \mathcal{Z}^{\text{new}})} \phi(\Sigma_{\mathbf{v}_i}). \tag{6.1.7}$$

Each element $\mathbf{v}_i$ contributes to the sum based on its covariance Σ_{v_i}, and in this chapter, we employ the A-Optimality metric as our uncertainty criterion ϕ.

To optimize the distribution of exploration tasks among the robots, we utilize a distance evaluation metric inspired by the methodology presented in [7]. For each potential new target state $\mathbf{a}_{\alpha}^{t\,\prime}$ assigned to robot α at time t, we quantify the effectiveness of task allocation using U_T:

$$U_T = \sum_{\mathbf{a}_{\beta}^{i} \in \mathcal{A}, \beta \neq \alpha} h(\|\mathbf{a}_{\alpha}^{t\,\prime} - \mathbf{a}_{\beta}^{i}\|_2), \tag{6.1.8}$$

$$h(d) = \begin{cases} 1 - \frac{d}{d_{\max}} & d < d_{\max} \\ 0 & d \geq d_{\max} \end{cases}. \tag{6.1.9}$$

Here, $\|\cdot\|_2$ represents the L2 norm, and $\mathcal{A}$ denotes the collection of all historical target states for all robots. Based on this, we can define the new target state for robot α as one of the potential target states that optimally balances the factors U_M, U_T and the Euclidean distance to target state factor, U_D, with scale factors λ_0, λ_1, λ_2:

$$\mathbf{a}_{\alpha}^{\text{next}} = \arg\min_{\mathbf{a}_{\alpha}^{t\,\prime}} (\lambda_0 U_M + \lambda_1 U_T + \lambda_2 U_D). \tag{6.1.10}$$

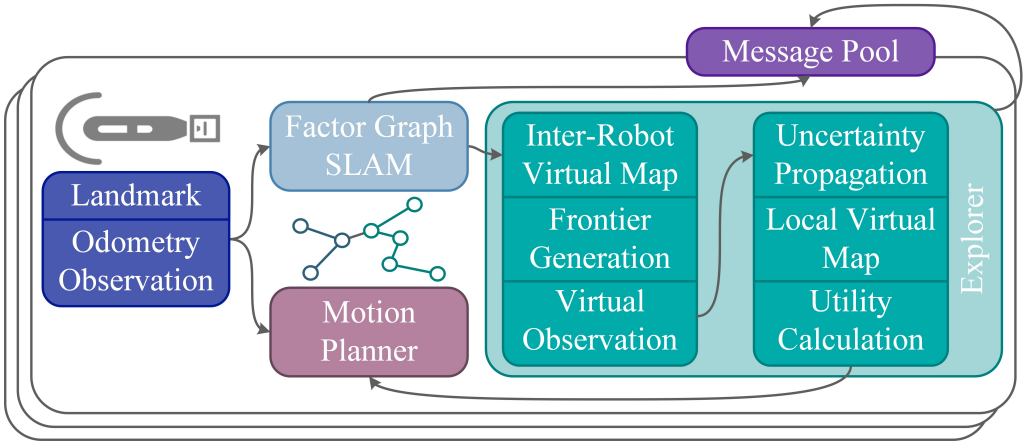


Figure 6.1: System Architecture. The pipeline of the proposed approach.

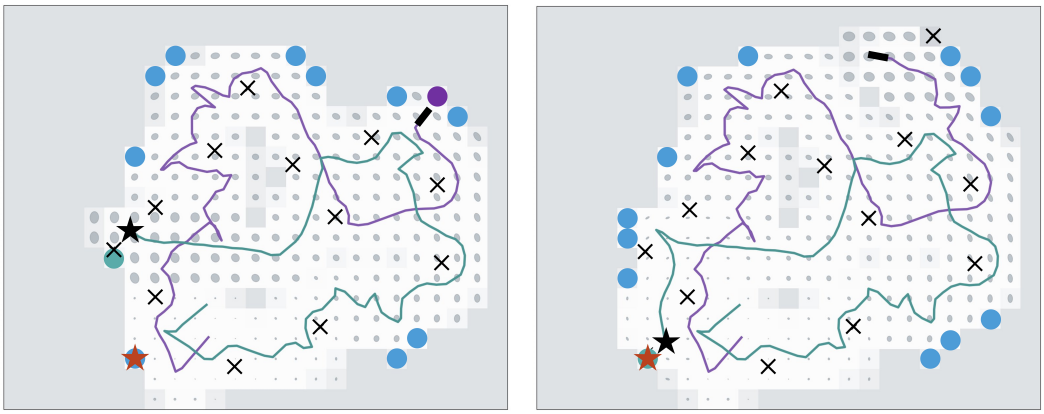
6.2 Multi-Robot Expectation-Maximization Exploration Algorithm

In this section, we discuss the details of the proposed expectation-maximization algorithm. Figure 6.1 shows the pipeline of the proposed algorithm. We consider a situation where a group of robots collaborates, and each individual robot within the group is furnished with both motion and perception sensors. A centralized or decentralized factor-graph based SLAM algorithm is assumed to operate at a specific frequency. The robot team shares their optimized SLAM trajectories, landmark positions and history of chosen target states among its members. However, individual robots maintain virtual maps locally using the latest SLAM estimates.

After a robot α successfully reaches its current target state, its virtual map is updated and a group of potential new target states $\mathcal{A}_{\alpha}^{\text{next}}$ is chosen from the virtual map. Subsequently, the virtual observation is synthesized utilizing the prevailing environmental knowledge. Next, the expectation-maximization based uncertainty propagation computes the potential impact of the target state under consideration, leading to an update in the virtual map's covariances. The new target goal is then selected according to the utility function (equation 6.1.10). Finally, the motion planner

is initiated to formulate a series of actions leading robot α to the new target state.

6.2.1 Virtual Map



(a) Choosing a revisiting frontier (blue, with red star) driven by the significant uncertainty in the map.

(b) Choosing an exploring frontier (green, with red star) due to relatively low map uncertainty.

Figure 6.2: Problem setup.

As depicted in equation 6.1.3, the virtual map $\mathcal{V}$ of a finite environment is generated from the robot states $\mathcal{X}$ and their associated observations $\mathcal{Z}$. Assuming that $\mathcal{V}$ comprises b map cells $\mathbf{v}_i$ referred to as *virtual landmarks*, the posterior can be redefined as follows:

$$P(\mathcal{V}|\mathcal{X},\mathcal{Z}) = \prod_{\mathbf{v}_i \in \mathcal{V}} P(\mathbf{v}_i|\mathcal{X},\mathcal{Z}). \tag{6.2.1}$$

The likelihood of virtual landmark $\mathbf{v}_i$ being observed is $q(\mathbf{v}_i) = \mathbb{E}[P(\mathbf{v}_i|\mathcal{X},\mathcal{Z})]$. When this virtual landmark is observed by multiple robot states, the procedure for updating $q(\mathbf{v}_i)$ is similar to updating map cell values in an occupancy grid map [91]. We assume the virtual landmark $\mathbf{v}_i$ is observed by a robot state $\mathbf{x}_{\alpha,j}$, with its estimated value denoted $\hat{\mathbf{x}}_{\alpha,j}$, and a marginal covariance $\Sigma_{\mathbf{x}_{\alpha,j}}$. We can compute the covariance:

$\Sigma_{\mathbf{v}_i} = \mathbf{H} \cdot \Sigma\mathbf{x}_{\alpha,j} \cdot \mathbf{H}^\intercal$. Here, $\mathbf{H} = \frac{\partial g(\mathbf{x}_{\alpha,j}, \mathbf{v}_i)}{\partial \mathbf{x}_{\alpha,j}} | \hat{\mathbf{x}}_{\alpha,j}$ represents the Jacobian matrix obtained by differentiating the landmark observation model $g(\mathbf{x}_{\alpha,j}, \mathbf{v}_i)$ with respect to estimated robot state $\hat{\mathbf{x}}_{\alpha,j}$. We utilize Covariance Intersection to compute the covariance of a virtual landmark that is observed by multiple robot states, as detailed in [96]. Figure 6.2 shows inter-robot virtual maps built from both local robot states and neighbors' robot states received by local robot α. In this $100m \times 80m$ virtual map created by a two-robot team, gray ellipses depict the uncertainty of visited cells. The green robot's position is denoted by a black star, and its newly selected target state is represented by a red star. The current position of the other robot on the team is marked as a black rectangle. Landmarks are expressed by black x's. Potential frontiers emerge along the boundary between the explored and unexplored areas. The three types of frontiers—exploring, revisiting, and rendezvous—are denoted by their respective colors: green, blue, and purple. The observed regions are highlighted in white; gray ellipses show covariances describing the uncertainty of the map's cells.

6.2.2 Uncertainty Propagation

As depicted in figure 6.2, most potential target states are chosen from the perimeters of the observed regions. Three types of frontiers are identified for selection: *exploration frontiers* close to the robot's latest position, *revisiting frontiers* near previously visited landmarks, and *rendezvous frontiers*, which are the current target positions of neighboring robots. Subsequently, for each potential target, a set of waypoints $\mathcal{X}^{\text{new}}$ is uniformly sampled along the shortest path connecting the robot's current state and the next potential target state. Additionally, waypoints $\mathcal{X}^{\text{predict}}$ are generated to connect the present states of all robots with their respective target states. The process of generating virtual observations $\mathcal{Z}^{\text{predict}}$ along the paths to these target states is depicted in figure 6.3. Nodes representing the current robot states are distinguished

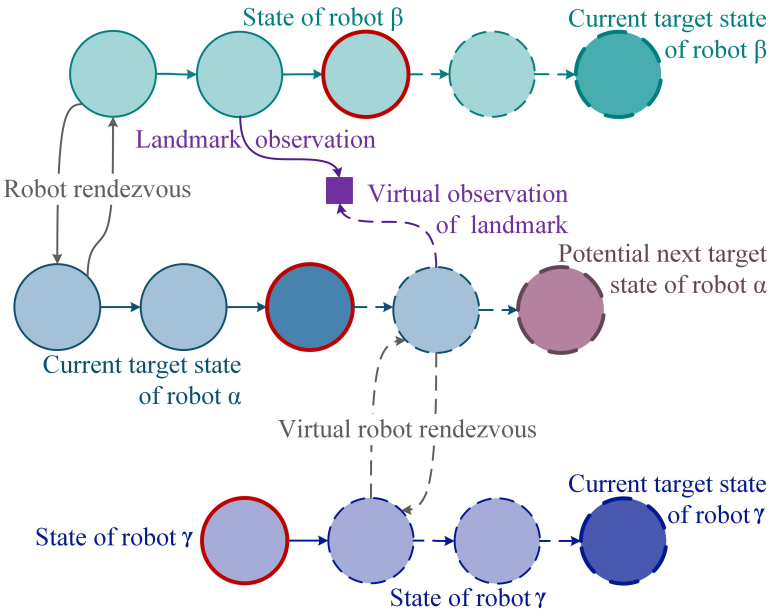


Figure 6.3: EM-based uncertainty propagation with virtual observations.

by red edges. For every potential next target state (shown in pink), a trajectory simulation is executed, leading to the generation of a sequence of virtual observations indicated by dashed arrows. Simultaneously, we model the future states and observations of other robots as they approach their individual current target states. Virtual odometry measurements are created between adjacent waypoints using equation 2.2.1. For virtual landmark observations, equation 2.2.2 is employed, generating observations between previously observed landmarks and nearby waypoints. When two robots are within each other's sensing range at the same timestep, a virtual robot observation is produced using equation 2.2.3. Following this, virtual observations are added into the SLAM graph to propagate uncertainty. The step-by-step procedure is outlined comprehensively in Algorithm 1.

Algorithm 1: Uncertainty Propagation

Global: Latest SLAM Graph $\mathcal{G}$,andInter-robot Virtual Map $\mathcal{V}^*$
Input: Potential frontier state $\mathbf{a}_\alpha^{t\,\prime}$, Current robot states $\mathcal{X}^t$, Current Target States of Neighbors $\mathcal{A}^t$
Output: Optimized robot states $\mathcal{X}^{\text{new}*}$
$\mathcal{X}^{\text{predict}} \leftarrow \emptyset$
foreach $\mathbf{a}_\gamma^t \in \mathcal{A}^t, \mathbf{x}_\gamma^t \in \mathcal{X}^t$ **do**
  $\mathcal{X}_\gamma^{t:*} \leftarrow$ `GenerateVirtualWaypoints(`$\mathbf{a}_\gamma^t, \mathbf{x}_\gamma^t$`)`
  $\mathcal{X}^{\text{predict}} \leftarrow \mathbf{X}_\gamma^t \cup \mathcal{X}^{\text{predict}}$
$\mathcal{X}^{\text{next}} \leftarrow$`GenerateVirtualWaypoints(`$\mathbf{a}_\alpha^{t\,\prime}, \mathbf{x}_\gamma^t$`)`
Calculate virtual observations
$\mathcal{Z}^{\text{predict}} \leftarrow$ `VirtualObserve(`$\mathcal{V}^*, \mathcal{X}^{predict} \cup \mathcal{X}^{next}$`)`
M-step: graph optimization
`UpdateGraph(`$\mathcal{G}, \mathcal{X}^{predict} \cup \mathcal{X}^{next}$`)`
$\mathcal{X}^{\text{new}*} \leftarrow$ `OptimizeGraph(`$\mathcal{G}$`)`
return $\mathcal{X}^{\text{new}*}$

6.2.3 Map Uncertainty Utility Computation

We compute the uncertainty of map cells by considering both the likelihood of the existing virtual map $\mathcal{V}^*$ and the optimized robot states of robot α, $\mathcal{X}_\alpha^{\text{new}*} \subset \mathcal{X}^{\text{new}*}$. Referencing figure 6.2b, it is evident that despite the purple robot's trajectory being impacted by localization uncertainty due to accumulated odometry error, this uncertainty is partially mitigated by the historical trajectory of the green robot, which exhibits relatively lower uncertainty. The construction of an inter-robot virtual map for map uncertainty estimation could potentially lead to conflicts in decision-making for a local robot. As outlined in algorithm 2, we generate a local virtual map denoted as $\mathcal{V}_\alpha^{\text{new}*}$ for robot α. This map is constructed using $\mathcal{X}_\alpha^{\text{new}*}$ and corresponding virtual observations $\mathcal{Z}_\alpha^{\text{new}}$. Then, we compute the uncertainty utility of the map by considering the overlapping observed regions common to both the current inter-robot virtual map $\mathcal{V}^*$ and the predicted local virtual map $\mathcal{V}_\alpha^{\text{new}*}$, via equation 6.1.6.

Algorithm 2: Compute U_M of potential frontier $\mathbf{a}_\alpha^{t\,\prime}$

Input: Inter-robot Virtual Map $\mathcal{V}^*$, Local Virtual Map $\mathcal{V}_\alpha^{\text{new}*}$
Output: Map uncertainty utility factor U_M
$U_M \leftarrow 0$
for $i \leftarrow 0$ **to** b **do**
  $\mathbf{v}_{i,g} \in \mathcal{V}^*$, $\mathbf{v}_{i,l} \in \mathcal{V}_\alpha^{\text{new}*}$
  # $q_{\min}$: minimum accepted probability of observed.
  if $\mathtt{q}(\mathbf{v}_{i,l}) > q_{\min}$ ***and*** $\mathtt{q}(\mathbf{v}_{i,g}) > q_{\min}$ **then**
    $U_M \leftarrow U_M + \mathtt{trace}(\Sigma_{\mathbf{v}_{i,l}})$
return U_M

6.2.4 Complexity Analysis

In this section, we analyze the time complexity of the Expectation-Maximization Explorer shown in figure 6.1. For a team of n robots, each with a maximum of N_x historical robot states, updating a robot state in the virtual map takes T_v time, resulting in a potential inter-robot virtual map construction time of $n \cdot N_x \cdot T_v$. During frontier generation, detecting exploration bounds (time T_b) containing N_c virtual landmarks and observing each landmark (time T_l) contributes to a step duration of $T_b + N_c \cdot T_l$. Frontier selection generates N_f frontiers. Crafting virtual waypoints (up to N_w) and conducting virtual robot rendezvous checks (time T_r) per frontier results in a total frontier generation time of $(N_f + n - 1) \cdot (T_l + T_r)$. Utilizing iSAM2 [50] for uncertainty propagation (time T_u per update) leads to a combined propagation time of $N_f \cdot T_u$. Constructing a local virtual map requires $N_x \cdot T_v$ time. The most time-intensive steps involve iSAM2's uncertainty propagation ($\mathcal{O}(n^{2.36})$ complexity for n states) and covariance intersection during virtual map creation ($\mathcal{O}(s^3)$ complexity with s nonzero matrix block size) [49].

6.3 Experiments

6.3.1 Experiments in 2D simulation environments

We perform simulated experiments within environments of varied size, employing randomly generated landmarks. The errors described below define 95% confidence intervals. Similar to our previous work [95], we assume each robot is furnished with a sonar with range error 0.002m, bearing error 0.5°, and max. sensing range 7.5m. Each robot also performs inertial dead reckoning; its gyro and accelerometer yield errors of .5° and 0.05m. The robot has the capability to rotate by 15° during each action, maintaining a constant speed of $1m/s$. It is restricted to moving solely in the direction of its current heading. At the beginning of each trial, we ensure that all robots are located within the sensing range of their teammates to guarantee a proper initialization of the SLAM framework. To navigate towards uncharted territory, we utilize the Artificial Potential Field (APF) method, detailed in [29], to avoid collision. The boundaries along the environment's edges are restricted from selection as frontiers to prevent the robot from exiting the mission area. We compare our proposed approach, denoted EM, with two advanced multi-robot exploration algorithms: the coordinated multi-robot exploration method by Burgard et al. [7], referred to as CE in subsequent sections, and the coperative multi-robot belief space planning technique introduced by Indelman et al. [45], denoted as BSP subsequently. Identical frontier selection criteria and virtual observation generation techniques are employed for both methods.

The CE planner [7] places emphasis on optimizing task distribution among

robots, employing a utility function that takes task allocation into account:

$$U_{CE} = \lambda_0 U_D + \lambda_1 U_T. \tag{6.3.1}$$

In this context, both U_D and U_T are computed the same way as outlined in equation 6.1.10. In our experiment, we select $\lambda_0 = 1$ and $\lambda_1 = 10$. **The BSP planner** [45] takes into account both exploration efficiency and localization uncertainty, employing a similar virtual observation strategy as ours. This leads to the formulation of the following utility function:

$$U_{BST} = \lambda_0 U_D + \lambda_1 U_M, \tag{6.3.2}$$

$$U_M = \sum_{\mathbf{x}_i \in \mathcal{X}^{\text{predict}*} \cup \mathcal{X}^{\text{next}*}} \phi_A(\sqrt{\Sigma_{\mathbf{x}_i}}). \tag{6.3.3}$$

We choose λ_0 as 5 and λ_1 as 1. Here, $\Phi_A(\cdot)$ denotes the trace of the matrix, and $\mathcal{X}^{\text{predict}*}$ and $\mathcal{X}^{\text{next}*}$ are computed through the maximum a posteriori process: (19)

$$\mathcal{X}^{\text{predict}*}, \mathcal{X}^{\text{next}*} = \underset{\mathcal{X}^{\text{predict}}, \mathcal{X}^{\text{next}}}{\arg\max} P(\mathcal{X}^{\text{predict}}, \mathcal{X}^{\text{next}} | \mathcal{Z}^{\text{predict}}, \mathcal{L}).$$

As the BSP planner exclusively takes into account the localization uncertainty of forthcoming steps, it overlooks the potential advantages arising from revisiting landmarks previously explored by the team. Since only the uncertainties at individual robot states are taken into account, U_M might become imbalanced and thus not effectively depict the overall localization uncertainty across the entire environment, especially in the presence of varying vehicle speeds.

A quantitative analysis using two distinct sizes of random generalized en-

vironments is conducted: one measuring $100m \times 100m$, and the other measuring $200m \times 200m$. Across 50 trials, each trial involves 20 landmarks with a radius of $1m$ within the environment. We introduce three robots into the environment to prevent rendezvous and landmark revisitation from becoming trivially achievable in scenarios involving a large number of robots. The landmark positions are selected randomly, ensuring a minimum distance of $10m$ between each pair of landmarks. For each trial, the robots initiate their positions from the middle left region of the environment, with their initial positions also being chosen randomly.

We test two distinct configurations for the proposed EM explorer: one with $\lambda_0 = 1, \lambda_1 = 0, \lambda_2 = 10$ referred to as EM_2, and the other with $\lambda_0 = 1, \lambda_1 = 20 \cdot (1 - r), \lambda_2 = 10$ referred to as EM_3, where r denotes the ratio of the explored area in the environment. For the $100m \times 100m$ environment, we employ a virtual map cell size of $c_v = 2m$, while for the $200m \times 200m$ environment, we choose a cell size of $c_v = 4m$ for the virtual map. We use three statistics to evaluate the performance of the proposed algorithm:

- Robot localization error: root-mean-square error (RSME) of the optimized trajectories of all robots from the SLAM framework.
- Landmark position error: RSME of the positions of all landmarks observed by the robot teams.
- Explored ratio: The proportion of the area considered as observed in the inter-robot virtual map.

Figure 6.4 and figure 6.5 depict the outcomes across different scenarios within smaller and larger environments, respectively. In figure 6.4, robots are exploring the environment by constructing a virtual map with cell size of $2m$. At left, the

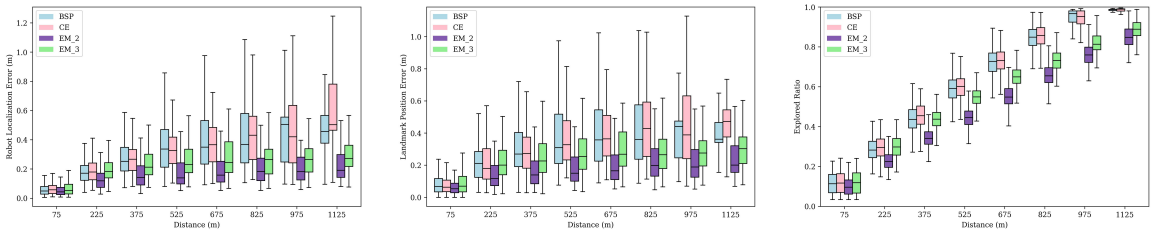


Figure 6.4: 50 trials of three robots navigating in 100m x 100m environments with 20 landmarks, each with a radius of 1m.

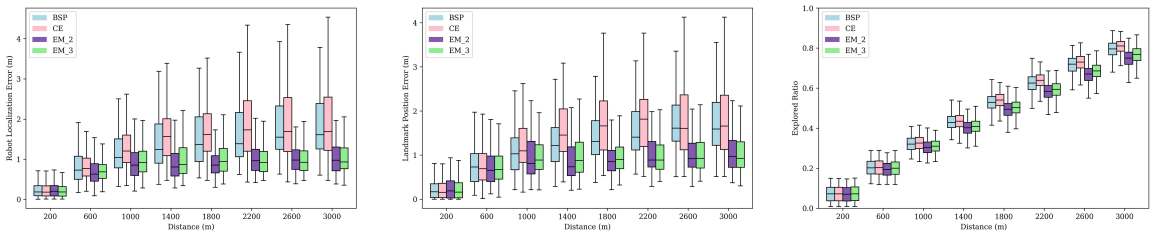


Figure 6.5: 50 trials of three robots navigating in 200m x 200m environments with 20 landmarks, each with a radius of 1m.

average robot localization error for each robot state, at center, the average landmark position error for each landmark, and at right, the explored ratio, all plotted against distance. In figure 6.5, they are exploring the environment by constructing a virtual map with cell size of $4m$. At left, the average robot localization error for each robot state, at center, the average landmark position error for each landmark, and at right, the explored ratio, all plotted against distance. EM_2 and EM_3 exhibit superior accuracy in localizing both robot and landmark states when compared to CE and BSP. This improvement stems from the fact that the proposed approach takes into account not only the forthcoming interactions among robots but also the impact of the team's historical actions. Nonetheless, it's worth noting that the proposed method demonstrates lower exploration efficiency when contrasted with both CE and BSP, especially in smaller environments. This trade-off is made in favor of achieving higher accuracy. When considering the task allocation element, the proposed approach in the EM_3 configuration demonstrates better exploration efficiency compared to the

EM_2 configuration.

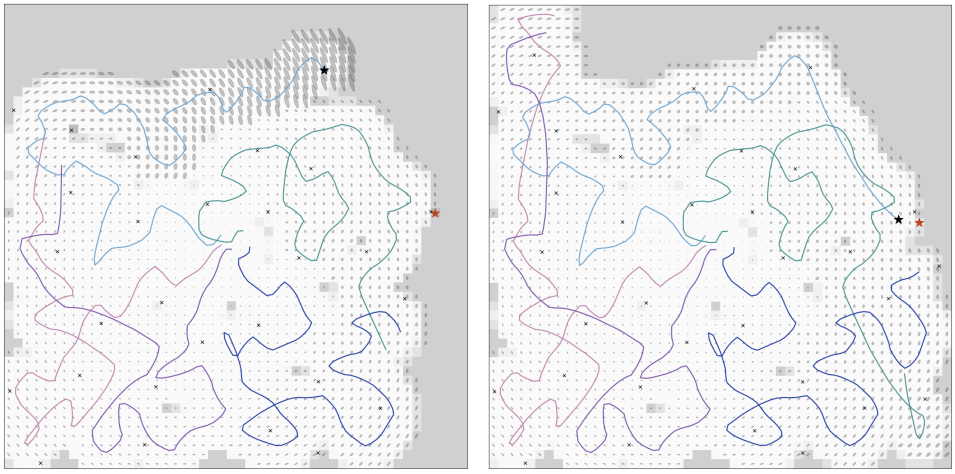


Figure 6.6: Five robots navigating a 150m x 150m environment.

Figure 6.6 and figure 6.7 present qualitative results of the proposed algorithm's EM_3 configuration. In figure 6.6, five robots navigate themselves in a $150m \times 150m$ environment with 40 landmarks. The active robot seeking its next target state is highlighted by a black star, while its chosen goal is marked with a red star. In the left map, the cyan robot revisits a previously explored landmark due to high map uncertainty (gray ellipses in each grid cell). After a loop closure in the right map, overall localization uncertainty decreases, prompting the robot to select the nearest unexplored frontier for its next move.

Three robots navigate a $200m \times 200m$ environment in figure 6.7, with only one landmark observed thus far. After some time, the accumulated localization uncertainty is relatively high (left), prompting the green robot to decide to rendezvous with its pink neighbor. This strategic move leads to a reduction in uncertainty, thanks to the inter-robot loop closure (middle). Concurrently, the light blue robot also accumulates errors and opts to rendezvous with its teammates. Once all three robots meet each other (right), the loop is closed, and uncertainty is subsequently propagated and

Figure 6.7: Three robots explore a 200m x 200m environment.

reduced.

6.4 Conclusions

This chapter introduces multi-robot exploration using expectation-maximization. We present an asynchronous exploration framework suitable for both centralized and decentralized robot teams, accounting for map uncertainty. The inclusion of rendezvous frontiers enhances the system's adaptability to environments with sparse features. The incorporation of a virtual map enables the estimation of future influence and interactions within the robot teams. The utilization of inter-robot and local maps heightens the robot's sensitivity to the accumulation of localization uncertainty in its trajectory.

Chapter 7

Conclusions and Future Work

This dissertation explores the integration of factor graph optimization into multiple aspects of perception and navigation, including motion planning, SLAM, and autonomous exploration. We introduce **MGPMP**, a Gaussian process motion planning approach tailored for UUVs conducting seafloor terrain-following missions while accounting for ocean currents. Additionally, we present distributed multi-robot SLAM algorithms designed for various robotic platforms equipped with different sensor modalities: **DiSCo-SLAM**, developed for ground vehicles with 3D LiDAR, employs a two-step global and local factor graph optimization strategy. **SGM-SLAM**, designed for legged robots equipped with 3D LiDAR and cameras, enables data-efficient object-level association through scene graph matching—even in scenarios with minimal trajectory overlap, where traditional methods based on compact sensor descriptors often fail. **DRACo-SLAM2**, intended for underwater robots with imaging sonars, achieves fast and robust inter-robot data association through graph matching and group-wise consistent measurement set maximization. Finally, we propose a multi-robot autonomous exploration algorithm based on expectation maximization, which balances exploration efficiency with SLAM localization accuracy.

For future work, we aim to further enhance the autonomy of mobile robots reliant upon factor graphs for perception and navigation, ensuring more robust performance in real-world robotic applications. Goals for future work include:

- Expanding the mission-oriented Gaussian process motion planning framework to address challenges in belief-space planning.

- Integrating our expectation maximization inspired autonomous exploration algorithm with DRACo-SLAM2 on unmanned underwater platforms, improving its scalability, and evaluating its performance in real-world environments.
- Combining multi-robot SLAM with scene graph-based algorithms and large language models (LLMs) to enable human-robot interactive motion planning and decision-making.

Bibliography

[1] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3D Scene Graph: a Structure for Unified Semantics, 3D Space, and Camera. *IEEE International Conference on Computer Vision (ICCV)*, pages 5664–5673, 2019.

[2] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas. Decentralized Active Information Acquisition: Theory and Application to Multi-robot SLAM. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4775–4782, 2015.

[3] G. Berton, C. Masone, and B. Caputo. Rethinking Visual Geo-Localization for Large-Scale Applications. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4878–4888, 2022.

[4] F. Bonin-Font and A. Burguera. Towards Multi-robot Visual Graph-SLAM for Autonomous Marine Vehicles. *Journal of Marine Science and Engineering*, 8 (6):437, 2020.

[5] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic Data Association for Semantic Slam. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729, 2017.

[6] M. Bryson, M. Johnson-Roberson, O. Pizarro, and S. Williams. Automated Registration for Multi-year Robotic Surveys of Marine Benthic Habitats. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3344–3349, 2013.

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated Multi-robot Exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[8] Y. Chang, N. Hughes, A. Ray, and L. Carlone. Hydra-multi: Collaborative Online Construction of 3D Scene Graphs with Multi-robot Teams. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10995–11002, 2023.

[9] B. Charrow, V. Kumar, and N. Michael. Approximate Representations for Multi-robot Control Policies that Maximize Mutual Information. *Autonomous Robots*, 37:383–400, 2014.

[10] S. Chen, L. Zhao, S. Huang, and Q. Hao. Multi-robot Active SLAM Based on Submap-joining for Feature-based Representation Environments. *Australasian Conference on Robotics and Automation*, 2022.

[11] Y. Chen, L. Zhao, K. M. B. Lee, C. Yoo, S. Huang, and R. Fitch. Broadcast Your Weaknesses: Cooperative Active Pose-graph SLAM for Multiple Robots. *IEEE Robotics and Automation Letters*, 5(2):2200–2207, 2020.

[12] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert. Distributed Trajectory Estimation with Privacy and Communication Constraints: a Two-stage Distributed Gauss-seidel Approach. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5261–5268, 2016.

[13] R. G. Colares and L. Chaimowicz. The Next Frontier: Combining Information Gain and Distance Cost for Decentralized Multi-robot Exploration. *ACM Symposium on Applied Computing*, pages 268–274, 2016.

[14] M. Corah, C. O'Meadhra, K. Goel, and N. Michael. Communication-efficient Planning and Mapping for Multi-robot Exploration in Large Environments. *IEEE Robotics and Automation Letters*, 4(2):1715–1721, 2019.

[15] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully Distributed SLAM Using Constrained Factor Graphs. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3025–3030, 2010.

[16] A. Cunningham, V. Indelman, and F. Dellaert. DDF-SAM 2.0: Consistent Distributed Smoothing and Mapping. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5220–5227, 2013.

[17] F. Dellaert and M. Kaess. Factor Graphs for Robot Perception. 2017. URL `http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf`.

[18] Y. Deng, J. Wang, J. Zhao, X. Tian, G. Chen, Y. Yang, and Y. Yue. OpenGraph: Open-Vocabulary Hierarchical 3D Graph Representation in Large-Scale Outdoor Environments. *IEEE Robotics and Automation Letters*, 9(10):8402–8409, 2024. doi: 10.1109/LRA.2024.3445607.

[19] Z. Deng, Y. Zhang, H. Yang, and H. Wang. Underwater Point Cloud Transmission Framework: Hybrid Encoder Implementation Based on CNN and Transformer. *Measurement Science and Technology*, 36(1):015111, 2024.

[20] I. Deutsch, M. Liu, and R. Siegwart. A Framework for Multi-robot Pose Graph SLAM. *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572, 2016.

[21] H. Do, S. Hong, and J. Kim. Robust Loop Closure Method for Multi-robot Map Fusion by Integration of Consistency and Data Similarity. *IEEE Robotics and Automation Letters*, 5(4):5701–5708, 2020.

[22] J. Dong, M. Mukadam, F. Dellaert, and B. Boots. Motion Planning As Probabilistic Inference Using Gaussian Processes and Factor Graphs. *Robotics: Science and Systems*, 12, 2016.

[23] J. Dong, M. Mukadam, B. Boots, and F. Dellaert. Sparse Gaussian Processes on Matrix Lie Groups: a Unified Framework for Optimizing Continuous-time Trajectories. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6497–6504, 2018.

[24] J. Drupt, A. I. Comport, C. Dune, and V. Hugel. MAM3SLAM: Towards

Underwater-robust Multi-agent Visual SLAM. *Ocean Engineering*, 302:117643, 2024.

[25] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena. An Online Multi-robot SLAM System for 3D LiDARs. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1004–1011, 2017.

[26] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena. SegMap: Segment-based Mapping and Localization Using Data-driven Descriptors. *The International Journal of Robotics Research*, 39 (2-3):339–355, 2020.

[27] K. El-Darymli, P. McGuire, D. Power, and C. R. Moloney. Target Detection in Synthetic Aperture Radar Imagery: a State-of-the-art Survey. *Journal of Applied Remote Sensing*, 7(1):071598, 2013. doi: 10.1117/1.JRS.7.071598.

[28] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *International Conference on Knowledge Discovery and Data Mining*, 96:226–231, 1996.

[29] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu. Improved Artificial Potential Field Method Applied for AUV Path Planning. *Mathematical Problems in Engineering*, 2020:1–21, 2020.

[30] D. Feng, Y. Qi, S. Zhong, Z. Chen, Q. Chen, H. Chen, J. Wu, and J. Ma. S3E: a Multi-Robot Multimodal Dataset for Collaborative SLAM. *IEEE Robotics and Automation Letters*, 2024.

[31] M. Fernandez-Cortizas, H. Bavle, D. Perez-Saura, J. L. Sanchez-Lopez, P. Campoy, and H. Voos. Multi S-graphs: an Efficient Distributed Semantic-relational Collaborative SLAM. *IEEE Robotics and Automation Letters*, 2024.

[32] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed Multirobot Exploration and Mapping. *Proceedings of the IEEE*, 94(7):1325–

1339, 2006.

[33] L. Freda, M. Gianni, F. Pirri, A. Gawel, R. Dubé, R. Siegwart, and C. Cadena. 3D Multi-robot Patrolling with a Two-level Coordination Strategy. *Autonomous Robots*, 43:1747–1779, 2019.

[34] A. R. Gaspar and A. Matos. Feature-Based Place Recognition Using Forward-Looking Sonar. *Journal of Marine Science and Engineering*, 11(11):2198, 2023.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision Meets Robotics: the Kitti Dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[36] M. F. Ginting, K. Otsu, J. A. Edlund, J. Gao, and A.-A. Agha-Mohammadi. CHORD: Distributed Data-sharing Via Hybrid ROS 1 and 2 for Multi-robot Exploration of Large-scale Complex Environments. *IEEE Robotics and Automation Letters*, 6(3):5064–5071, 2021.

[37] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada. Collaborative Dynamic 3d Scene Graphs for Automated Driving. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11118–11124, 2024.

[38] Q. Gu, Z. Ye, J. Yu, J. Tang, T. Yi, Y. Dong, J. Wang, J. Cui, X. Chen, and Y. Wang. MR-COGraphs: Communication-efficient Multi-Robot Open-vocabulary Mapping System Via 3D Scene Graphs. *arXiv preprint arXiv:2412.18381*, 2024.

[39] D. He, W. Xu, N. Chen, F. Kong, C. Yuan, and F. Zhang. Point-LIO: Robust High-Bandwidth Light Detection and Ranging Inertial Odometry. *Advanced Intelligent Systems*, 5(7):2200459, 2023.

[40] Y. Huang, T. Shan, F. Chen, and B. Englot. DiSCo-SLAM: Distributed Scan Context-enabled Multi-robot Lidar Slam with Two-stage Global-local Graph Optimization. *IEEE Robotics and Automation Letters*, 7(2):1150–1157, 2021.

[41] Y. Huang, X. Lin, and B. Englot. Multi-Robot Autonomous Exploration and Mapping Under Localization Uncertainty with Expectation-Maximization. *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[42] Y. Huang, X. Lin, M. Hernandez-Rocha, S. Narain, K. Pochiraju, and B. Englot. Mission-oriented Gaussian Process Motion Planning for UUVs Over Complex Seafloor Terrain and Current Flows. *IEEE Robotics and Automation Letters*, 9 (2):1780–1787, 2024.

[43] Y. Huang, J. McConnell, X. Lin, and B. Englot. DRACo-SLAM2: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams with Object Graph Matching. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[44] Y. Huang, T. Shan, A. Rajvanshi, N. Chowdhury Mithun, Y. Li, B. Englot, and H.-P. Chiu. SGM-SLAM: Scene Graph Matching for Data-Efficient Distributed SLAM. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[45] V. Indelman. Cooperative Multi-robot Belief Space Planning for Autonomous Navigation in Unknown Environments. *Autonomous Robots*, 42:353–373, 2018.

[46] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[47] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim. Multi-robot Active Sensing and Environmental Model Learning with Distributed Gaussian Process. *IEEE Robotics and Automation Letters*, 5(4):5905–5912, 2020.

[48] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image Retrieval Using Scene Graphs. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015.

[49] M. Kaess and F. Dellaert. Covariance Recovery From a Square Root Information Matrix for Data Association. *Robotics and autonomous systems*, 57(12):1198–1210, 2009.

[50] 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*, 31(2):216–235, 2012.

[51] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic Trajectory Optimization for Motion Planning. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

[52] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas. Asymptotically Optimal Planning for Non-Myopic Multi-Robot Information Gathering. *Robotics: Science and Systems*, pages 22–26, 2019.

[53] M. Karrer, P. Schmuck, and M. Chli. CVI-SLAM—collaborative Visual-inertial SLAM. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, 2018.

[54] G. Kim and A. Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3d Point Cloud Map. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809, 2018.

[55] H. Kim, G. Kang, S. Jeong, S. Ma, and Y. Cho. Robust Imaging Sonar-based Place Recognition and Localization in Underwater Environments. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1083–1089, 2023.

[56] M. King-Smith, P. Tsiotras, and F. Dellaert. Simultaneous Control and Trajectory Estimation for Collision Avoidance of Autonomous Robotic Spacecraft Systems. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 257–264, 2022.

[57] M. Kontitsis, E. A. Theodorou, and E. Todorov. Multi-robot Active Slam with

Relative Entropy Optimization. *American Control Conference (ACC)*, pages 2757–2764, 2013.

[58] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[59] M. Labbé and F. Michaud. RTAB-Map As an Open-source Lidar and Visual Simultaneous Localization and Mapping Library for Large-scale and Long-term Online Operation. *Journal of field robotics*, 36(2):416–446, 2019.

[60] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame. DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020.

[61] M. Lauri, E. Heinänen, and S. Frintrop. Multi-robot Active Information Gathering with Periodic Communication. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 851–856, 2017.

[62] S. LaValle. Rapidly-exploring Random Trees: a New Tool for Path Planning. *Research Report 9811*, 1998.

[63] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Robust Pose-graph Loop-closures with Expectation-maximization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 556–563, 2013.

[64] M. Leordeanu and M. Hebert. A Spectral Technique for Correspondence Problems Using Pairwise Constraints. *IEEE International Conference on Computer Vision (ICCV)*, 2:1482–1489 Vol. 2, 2005.

[65] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan. Pairwise Consistent Measurement Set Maximization for Robust Multi-robot Map Merging. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2916–2923, 2018.

[66] J. McConnell, Y. Huang, P. Szenher, I. Collado-Gonzalez, and B. Englot.

DRACo-SLAM: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8457–8464, 2022.

[67] J. Meng, A. Humne, R. Bucknall, B. Englot, and Y. Liu. A Fully-Autonomous Framework of Unmanned Surface Vehicles in Maritime Environments Using Gaussian Process Motion Planning. *IEEE Journal of Oceanic Engineering*, 48 (1):59–79, 2022.

[68] M. Mukadam, X. Yan, and B. Boots. Gaussian Process Motion Planning. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15, 2016.

[69] M. Mukadam, J. Dong, F. Dellaert, and B. Boots. STEAP: Simultaneous Trajectory Estimation and Planning. *Autonomous Robots*, 43:415–434, 2019.

[70] M. Ossenkopf, G. Castro, F. Pessacg, K. Geihs, and P. De Cristóforis. Long-horizon Active SLAM System for Multi-agent Coordinated Exploration. *European Conference on Mobile Robots (ECMR)*, pages 1–6, 2019.

[71] Ö. Özkahraman and P. Ögren. Collaborative Navigation-aware Coverage in Feature-poor Environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10066–10073, 2022.

[72] A. Patwardhan, R. Murai, and A. J. Davison. Distributing Collaborative Multi-Robot Planning with Gaussian Belief Propagation. *IEEE Robotics and Automation Letters*, 8(2):552–559, 2022.

[73] L. Paull, G. Huang, M. Seto, and J. J. Leonard. Communication-constrained Multi-AUV Cooperative SLAM. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516, 2015.

[74] L. Petrović, I. Marković, and M. Seder. Multi-agent Gaussian Process Mo-

tion Planning Via Probabilistic Inference. *IFAC-PapersOnLine*, 51(22):160–165, 2018.

[75] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *IEEE Transactions on Robotics*, 39(3):1686–1705, 2023. doi: 10.1109/TRO.2023.3248510.

[76] E. Potokar, K. Lay, K. Norman, D. Benham, S. Ashford, R. Peirce, T. B. Neilsen, M. Kaess, and J. G. Mangelson. HoloOcean: a Full-Featured Marine Robotics Simulator for Perception and Autonomy. *IEEE Journal of Oceanic Engineering*, 49(4):1322–1336, 2024. doi: 10.1109/JOE.2024.3410290.

[77] C. Qi, T. Ma, Y. Li, L. Lv, and Y. Ling. An Efficient Loop Closure Detection Method for Communication-constrained Bathymetric Cooperative SLAM. *Ocean Engineering*, 304:117720, 2024.

[78] Radu Bogdan Rusu and Steve Cousins. 3D Is Here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[79] A. Radwan, A. Tourani, H. Bavle, H. Voos, and J. L. Sanchez-Lopez. UAV-assisted Visual SLAM Generating Reconstructed 3D Scene Graphs in GPS-denied Environments. *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1109–1116, 2024.

[80] T. Regev and V. Indelman. Multi-robot Decentralized Belief Space Planning in Unknown Environments Via Efficient Re-evaluation of Impacted Paths. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5591–5598, 2016.

[81] L. Riazuelo, J. Civera, and J. M. Montiel. C2tam: a Cloud Framework for Cooperative Tracking and Mapping. *Robotics and Autonomous Systems*, 62(4): 401–413, 2014.

[82] M. A. Richards et al. Fundamentals of Radar Signal Processing. 1, 2005.

[83] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard. SE-Sync: a Certifiably Correct Algorithm for Synchronization Over the Special Euclidean Group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[84] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[85] M. M. Santos, G. B. Zaffari, P. O. Ribeiro, P. L. Drews-Jr, and S. S. Botelho. Underwater Place Recognition Using Forward-looking Sonar Images: a Topological Approach. *Journal of Field Robotics*, 36(2):355–369, 2019.

[86] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas. Anytime Planning for Decentralized Multirobot Active Information Gathering. *IEEE Robotics and Automation Letters*, 3(2):1025–1032, 2018.

[87] J. L. Schonberger and J.-M. Frahm. Structure-from-motion Revisited. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[88] T. Shan and B. Englot. Lego-loam: Lightweight and Ground-optimized Lidar Odometry and Mapping on Variable Terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.

[89] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. Lio-sam: Tightly-coupled Lidar Inertial Odometry Via Smoothing and Mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[90] I. A. Sucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[91] S. Thrun. Probabilistic Robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[92] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How. Distributed Certifiably Correct Pose-graph Optimization. *IEEE Transactions on Robotics*, 37(6):2137–2156, 2021.

[93] Y. Tian, Y. Chang, L. Quang, A. Schang, C. Nieto-Granda, J. P. How, and L. Carlone. Resilient and Distributed Multi-robot Visual Slam: Datasets, Experiments, and Lessons Learned. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11027–11034, 2023.

[94] M. Tzes, N. Bousias, E. Chatzipantazis, and G. J. Pappas. Graph Neural Networks for Multi-robot Active Information Acquisition. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3497–3503, 2023.

[95] J. Wang and B. Englot. Autonomous Exploration with Expectation-maximization. *Robotics Research: International Symposium ISRR*, pages 759–774, 2020.

[96] J. Wang, F. Chen, Y. Huang, J. McConnell, T. Shan, and B. Englot. Virtual Maps for Autonomous Exploration of Cluttered Underwater Environments. *IEEE Journal of Oceanic Engineering*, 47(4):916–935, 2022.

[97] Y. Wang, W. Thanyamanta, and N. Bose. Cooperation and Compressed Data Exchange Between Multiple Gliders Used to Map Oil Spills in the Ocean. *Applied Ocean Research*, 118:102999, 2022.

[98] S. B. Williams, O. Pizarro, and B. Foley. Return to Antikythera: Multi-session Slam Based Auv Mapping of a First Century Bc Wreck Site. *Field and Service Robotics: Results of the 10th International Conference*, pages 45–59, 2016.

[99] K. Ye, S. Dong, Q. Fan, H. Wang, L. Yi, F. Xia, J. Wang, and B. Chen. Multi-robot Active Mapping Via Neural Bipartite Graph Matching. *IEEE Confer-*

ence on Computer Vision and Pattern Recognition (CVPR), pages 14839–14848, 2022.

[100] J. Yu and S. Shen. SemanticLoop: Loop Closure with 3D Semantic Graph Matching. *IEEE Robotics and Automation Letters*, 8(2):568–575, 2022.

[101] J. Yu, J. Tong, Y. Xu, Z. Xu, H. Dong, T. Yang, and Y. Wang. Smmr-explore: Submap-based Multi-robot Exploration System with Multi-robot Multi-target Potential Field Exploration Method. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8779–8785, 2021.

[102] F. Zhang, D. Xu, and C. Cheng. An Underwater Distributed SLAM Approach Based on Improved GMRBnB Framework. *Journal of Marine Science and Engineering*, 11(12):2271, 2023.

[103] P. Zhang, H. Wang, B. Ding, and S. Shang. Cloud-based Framework for Scalable and Real-time Multi-robot Slam. *IEEE International Conference on Web Services (ICWS)*, pages 147–154, 2018.

[104] Y. Zhu, Y. Kong, Y. Jie, S. Xu, and H. Cheng. Graco: a Multimodal Dataset for Ground and Aerial Cooperative Localization and Mapping. *IEEE Robotics and Automation Letters*, 8(2):966–973, 2023.

[105] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. Chomp: Covariant Hamiltonian Optimization for Motion Planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

Vita

Yewei Huang

Education

Stevens Institute of Technology, Hoboken, NJ
Ph.D. in Mechanical Engineering — August 2019 — May 2025
Tongji University, Shanghai, China
M.E. in Surveying and Mapping — September 2016 — March 2019
Tongji University, Shanghai, China
B.E. in Geographic Information System — September 2012 — June 2016

Work Experience

SRI International, NJ
Summer Research Internship — May 2024 — August 2024

Publications

Y. Huang, T. Shan, F. Chen and B. Englot "DiSCo-SLAM: Distributed scan context-enabled multi-robot lidar slam with two-stage global-local graph optimization" *IEEE Robotics and Automation Letters*, 7(2):1150–1157, 2021.

Y. Huang, X. Lin, M. Hernandez-Rocha, S. Narain, K. Pochiraju and B. Englot "Mission-oriented Gaussian Process Motion Planning for UUVs over Complex Seafloor Terrain and Current Flows" *IEEE Robotics and Automation Letters*,9(2):1780-1787, 2024.

Y. Huang, X. Lin, and B. Englot "Multi-Robot Autonomous Exploration and Mapping Under Localization Uncertainty with Expectation-Maximization" *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 7236–7242, 2024.

Y. Huang, T. Shan, A. Rajvanshi, N. Chowdhury, Y. Li, B. Englot, and H.-P. Chiu "SGM-SLAM: Scene Graph Matching for Data-Efficient Distributed SLAM" *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, under review, 2025.

Y. Huang, J. McConnell, X. Lin, and B. Englot "DRACo-SLAM2: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams with Object Graph Matching" *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, under review, 2025.

F. Chen, J. Martin, **Y. Huang**, J. Wang, and B. Englot "Autonomous exploration under uncertainty via deep reinforcement learning on graphs" *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6140–6147, 2020.

F. Chen, P. Szenher, **Y. Huang**, J. Wang, T. Shan, B. Shi and B. Englot "Zero-shot reinforcement learning on graphs for autonomous exploration under uncertainty" *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 5193–5199, 2021.

J. McConnell, **Y. Huang**, P. Szenher, I. Collado-Gonzalez and B. Englot "DRACo-SLAM: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams" *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8457–8464, 2022.

J. Wang, F. Chen, **Y. Huang**, J. McConnell, T. Shan, and B. Englot "Virtual maps for autonomous exploration of cluttered underwater environments" *IEEE Journal of Oceanic Engineering*, 47(4):916–935, 2022.

X. Lin, **Y. Huang**, D. Sun, T.-Y. Lin, B. Englot, R. M. Eustice, and M. Ghaffari "A Robust Keyframe-Based Visual SLAM for RGB-D Cameras in Challenging Scenarios" *IEEE Access*, 11, 97239–97249, 2023.

X. Lin, P. Szenher, **Y. Huang**, and B. Englot "Distributional Reinforcement Learning Based Integrated Decision Making and Control for Autonomous Surface Vehicles" *IEEE Robotics and Automation Letters*, 2024.

K. Doherty, A. Papalia, **Y. Huang**, D. Rosen, B. Englot, and J. Leonard "MAC: Maximizing Algebraic Connectivity for Graph Sparsification" *arXiv e-prints*, arXiv–2403, 2024.

Y. Li, **Y. Huang**, B. Gaudel, H. Jafarnejadsani, and B. Englot "CVD-SfM: A Cross-View Deep Front-end Structure-from-Motion System for Sparse Localization in Multi-Altitude Scenes" *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, under review, 2025.