UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Filter Based Active SLAM in Static and Deformable Environments

by Mengya Xu

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of
Prof. Shoudong Huang,
Dr. Liang Zhao,
Prof. Qi Hao

University of Technology Sydney
Faculty of Engineering and Information Technology

May 2024

Declaration of Authorship

I, Mengya Xu, declare that this thesis, is submitted in fulfillment of the requirements for the award of Doctor of Philosophy, in the School of Mechanical and Mechatronic Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signed: Production Note:
Signature removed prior to publication.

Date: 04/05/2024

Abstract

UNIVERSITY OF TECHNOLOGY SYDNEY

Faculty of Engineering and Information Technology

Robotics Institute

Doctor of Philosophy

by Mengya Xu

Simultaneous localization and mapping (SLAM) has been a hot topic in mobile robots for decades. Classical SLAM systems do not control the motion of the robot, which is called passive SLAM. In contrast, active SLAM algorithms control the motion of the robot to complete certain tasks and, at the same time, maintain a good SLAM estimate. Active SLAM is a very important decision-making problem when a robot is navigating in an unknown environment. A good active SLAM algorithm can help improve the estimation accuracy of the map and robot localization, and additionally take into consideration other tasks such as coverage and time of exploration at the same time. Active SLAM in static environments has been well-studied in the last decades. It is always combined with the exploration tasks, where the exploration-exploitation dilemma is usually a challenge. Active SLAM in deformable environments is also an important research topic due to its wide applications in many different areas, such as robotic surgery. A few groups have completed some good work on SLAM in deformable environments. In most of the work, the motion of the robot is controlled by humans or predetermined. However, it would be more desirable if the robot could decide on its actions online based on different situations.

There are two typical methods to solve the SLAM back-end problem, including the optimization based method and the filter based method. The Extended Kalman Filter (EKF) based methods have been dominant in both classical and active SLAM for a long time in earlier stages of SLAM because of their high computational efficiency as compared with

the optimization based approaches. However, it suffers from the inconsistency problem because of the incorrectly calculated dimension of unobservable space due to linearization errors, leading to inaccurate SLAM estimates. Some research has enhanced the consistency of EKF SLAM in the literature. Especially, a Right Invariant Extended Kalman Filter (RIEKF) based SLAM algorithm has been proven to outperform other typical algorithms. Using all the information, the optimization based methods can achieve accurate SLAM estimates as well as consistent uncertainty estimates. However, the computational cost of optimization based SLAM is very high due to the large number of robot poses involved.

This dissertation focuses on the EKF method and its invariant format, aiming to solve the active SLAM problem in both static and deformable environments. Firstly, an improved EKF based active SLAM method is designed to explore 2D feature-based static environments. Simulation results show the great potential of using RIEKF in active SLAM. Therefore, secondly, we extend the RIEKF based active SLAM algorithm to 3D cases. The environments are 3D and have obstacles, and we improved the exploration framework accordingly. Both simulations and real experiments demonstrate that our proposed method performs better than the EKF based methods. Finally, we propose an EKF based active SLAM algorithm for 3D feature-based deformable environments. The algorithm is designed based on possible assumptions of the feature dynamic model. We have got good results in both simulation and real-world experiments, where the robot can observe the whole environment efficiently with more accurate estimates. This is the first step in developing an efficient active SLAM algorithm in deformable environments.

Acknowledgements

First and foremost, I would like to express my profound gratitude to my principal supervisor, Prof. Shoudong Huang, for his unwavering guidance, patience, and encouragement throughout my PhD journey. His expertise, dedication, and belief in my abilities have been invaluable in shaping my academic development and research. I am deeply appreciative of the time and effort he invested in mentoring me and for inspiring me to pursue excellence in robotics research. I would also like to extend my heartfelt thanks to my co-supervisor, Dr. Liang Zhao, for his practical advice and insightful feedback. His expertise and support have been crucial in overcoming challenges and refining my work. I am equally grateful to my external supervisor, Prof. Qi Hao, for his thoughtful guidance and encouragement, which greatly enriched the quality of my research and broadened my perspectives on robotics and its applications.

I am profoundly grateful for the opportunity to study at both the Southern University of Science and Technology (SUSTech) and the University of Technology Sydney (UTS). This unique collaboration has provided me with access to world-class resources, a vibrant academic environment, and the chance to work with talented researchers and colleagues from diverse backgrounds. The experiences and friendships I have gained at both institutions have made this journey deeply rewarding and memorable.

I would like to express my sincere gratitude to SUSTech for providing the financial support for my PhD studies through their scholarship. Their generosity has enabled me to focus on my research and academic pursuits. I also wish to acknowledge the administrative and technical staff at both institutions for their assistance in managing various aspects of my studies. Their efforts have ensured the smooth progress of my research and made this journey more manageable.

I extend my heartfelt thanks to all my colleagues and friends for their support and encouragement. I am especially grateful to Yang Song and Yongbo Chen for their invaluable assistance with my research papers, particularly during the early stages of my PhD study. I also wish to thank Tiancheng Li, Yang Song, Yingyu Wang, and many other colleagues who have offered tremendous help not only in my academic pursuits but also in my personal life, especially during my initial days in Sydney. Additionally, I am deeply thankful to my friends in the SUSTech Dance Society for bringing joy and good memories to an otherwise intense and often monotonous research life.

Finally, I would like to express my heartfelt gratitude to my parents, whose love, patience, and constant belief in me have been my greatest source of strength. Your constant support has provided the foundation that allowed me to pursue my dreams and overcome many challenges. I am deeply grateful for all your sacrifices, encouragement, and for always being there for me throughout my life.

To everyone who has contributed to this journey in any capacity, I offer my heartfelt thanks. This thesis is a testament to your support and inspiration.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

UTS	University of Technology Sydney
RI	Robotics Institute
SLAM	Simultaneous Localization and Mapping
EKF	Extended Kalman Filter
RIEKF	Right Invariant Extended Kalman Filter
2D	Two-Dimensional
3D	Three-Dimensional
OC-EKF	Observability constrained EKF
RRT	Rapidly exploring random tree
IT	Information theory
TOED	Theory of optimal experimental design
EM	Expectation-maximization
KLD	Kullback-Leibler divergence
FIM	Fisher information matrix
DDP	Differential dynamic programming
MPC	model predictive control
RGB-D	Red-Green-Blue-Depth

GPU	Graphics Processing Unit
MIS	Minimal invasive surgery
ORB	Oriented FAST and rotated BRIEF
NRSfM	Non-rigid Structure-from-Motion
RL	Reinforcement learning
NLS	Nonlinear least squares optimization
NLSI	Traditional nonlinear least squares optimization
NLSlb	lower bound based nonlinear least squares
Lidar	Light Detection And Ranging Sensor
CT	Computed tomography

Nomenclature

	General Notations
$\mathcal{N}$	Gaussian distribution
$n = 1, \ldots, N$	Index of time steps
$\mathbf{X}_n$, $\hat{\mathbf{X}}_n$	The n-th step state and its estimation
$\mathbf{P}_n$	The covariance matrix of the n-th step state
$\mathbf{R}_n$	Robot orientation
$\mathbf{x}_n$	Robot position
$j = 1, \ldots, M$	Index of features
$\mathbf{f}_n^j$	The coordinate of the j-th feature
$f(\cdot)$	Process model
$\mathbf{u}_n$	Robot odometry
$\boldsymbol{\omega}_n$	Angular velocity
$\mathbf{v}_n$	Linear velocity
$\mathbf{w}_n$	Noise of the process model
$\mathbf{w}_n^{\omega}$, $\mathbf{w}_n^{v}$	Noise of the robot angular velocity and linear velocity
$\boldsymbol{\Phi}_n$	Variance of noise $\mathbf{w}_n$
$\mathbf{Z}_n$	The observation at the n-th step
$h_n(\cdot)$	Observation model
$\boldsymbol{\xi}_n$	Observation noise
$\boldsymbol{\Psi}_n$	Variance of noise $\boldsymbol{\xi}_n$
O_n	The set of features observed at time n
i	Index of feature in O_n
$\boldsymbol{\Psi}_n^i$	Variance of noise of the i-th observed feature at time n

$\mathbf{e}_n$	Noise of the state
$\oplus$, $\ominus$	Retraction in differentiable geometry and its inverse
$\mathbf{F}_n$	The Jacobian of the robot motion model
$\mathbf{G}_n$	The Jacobian of the motion noise
$\mathbf{S}_n$	Innovation covariance
$\mathbf{K}_n$	Kalman gain
$\mathbf{H}_n$	The Jacobian of the observation model
$\mathbb{R}^n$	The n-dimensional Euclidean space
$\theta_n \in \mathbb{R}^1$	2D Robot orientation in Euler Angle
$\Theta_n \in \mathbb{R}^3$	3D Robot orientation in Euler Angle
$\mathbb{SO}(n)$	The special orthogonal group
$\mathcal{G}(\cdot)$	State set
$\mathfrak{g}(\cdot)$	The associated Lie algebra of $\mathcal{G}(\cdot)$
$\exp(\cdot)$	The exponential mapping
$\mathbf{I}_n$	The identity matrix
$\boldsymbol{J}$	The skew symmetric matrix
$\boldsymbol{\nabla}$	The Jacobian operator
$\exp^{\mathbb{SO}(3)}(\cdot)$	The exponential mapping
$S(\cdot)$	The skew symmetric operator
$J_r(\cdot)$	Right Jacobian
$\text{trace}(\cdot)$	The trace of a matrix
obj	The objective function
d	The distance between the robot and the goal point
w_p, w_d	The weight of $\text{trace}(\mathbf{P})$ and d
L_e	The exploration point list
$p_{explore}$	The selected exploration point
p_{good}	The feature with low uncertainty
p_{poor}	The feature with high uncertainty
$d_{explore}$, d_{good}, d_{poor}	The distance from the robot to $p_{explore}$, p_{good} and p_{poor}
k	The number of the observed features

w_k, w_n	The weight of k and n described above
const	A constant value
upperbound, *lowerbound*	The threshold for selecting the goal point
$\mathbf{q}$	Distance between the robot and the feature
$atan2(y, x)$	The angle measure (in radians) between the positive x-axis and the ray from the origin to the point (x, y)
$\mathbf{\Lambda}_n$	Information matrix
$\mathcal{LB}_n$	Lower uncertainty bound
$\mathbf{t}$, $\mathbf{r}_t$	Feature global translation and rotation
$\mathbf{d}$, $\mathbf{r}_d$	Feature local deformation and rotation
$\mathbf{w}_n^{j=1:M}$	Feature motion noise
$\mathbf{w}_{n,j}^t$	Feature global translation noise
$\mathbf{w}_{n,j}^d$	Feature local deformation noise
$\mathbf{z}_{n,j}^f$	The observation from the robot sensor to the j-th feature
$\mathbf{z}_{n,j1,j2}^c$	The observation of the constraint between feature $\mathbf{f}^{j1}$ and $\mathbf{f}^{j2}$
$\mathbf{f}^{loc}(n, j)$	The j-th feature's position relative to $\mathbf{f}_n^1$ in the local coordinate
$\boldsymbol{\xi}_n^f$	Feature measurement noise
$\boldsymbol{\xi}_n^c$	Structure measurement noise
$\mathbf{g}$	A viewpoint in the 3D space
$\mathcal{G}$	The set of candidate viewpoints
N_g	The number of viewpoints in $\mathcal{G}$
c	Cost function
$d_{i,i+1}^g$	The distance between $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$
$\mathbf{U}$	The set of candidate actions
N_u	The number of candidate actions in $\mathbf{U}$

Chapter 1

Introduction

1.1 Motivation

SLAM is the process of mapping an area while keeping track of the location of the robot within that area. SLAM has been a hot topic in mobile robots for decades because of its wide usage and great importance in various scenarios. Classical SLAM systems do not control the motion of the robot, which is called passive SLAM. In contrast, active SLAM algorithms control the motion of the robot to achieve certain tasks and, at the same time, maintain an accurate SLAM estimate.

Active SLAM is a decision-making problem where the robot's motion is planned at the same time as the SLAM process. A good active SLAM algorithm can help improve the estimation accuracy of the map and robot localization, and additionally take into consideration other tasks such as coverage and time of exploration at the same time. Active SLAM is important in many scenarios. The research on active SLAM in unknown static environments was started two decades ago and has been extensively explored [2–4]. One of the most important performance criteria for active SLAM is the quality of the SLAM estimate. In most cases, an objective function built by the Fisher information matrix or covariance matrix in the estimation process is used to select robot motion ([2, 5, 6]). Active SLAM is always combined with exploration tasks, where the robot is expected to

explore an unknown environment as soon as possible and, at the same time, keep an accurate SLAM estimation. In these problems, how to deal with the exploration-exploitation dilemma is usually a challenge.

Active SLAM in deformable environments is also an important research topic due to its wide applications in many different areas, such as robotic surgery. In recent years, active SLAM has played an increasingly important role in deformable environments like those in the human body. A few groups have completed some good work on SLAM in deformable environments. When an RGB-D sensor is used to observe the environment, a common approach is to deform the prior or build map directly based on the observations [7] [8] [9] [10] [11]. In some cases, high accuracy is required, but the sensor vision is limited; extra techniques are needed to provide additional information. For example, in some surgical cases, a computed tomography (CT) is used to provide an ideal prior model for recovering the deformation [12] [13], while [14] uses GPU and ORB-SLAM to obtain a pose estimation first. Many works like [15] [16] have investigated more challenging cases when only one monocular camera can be used. In [17], some fundamental questions about SLAM in deformable environments are discussed, such as the observability and the consistency. Existing robots have been successfully used to reconstruct the structure of organs like the stomach and intestine so that the pathological parts can be distinguished. However, the motion of these robots needs to be predetermined or controlled by humans, which limits their application scenarios and scope. In this case, it is more desirable if the robot can autonomously plan its motion online based on different situations. The environment in the human body is deformable and highly dynamic. It is necessary for the robot to take these changes into consideration so that it can successfully reach the target region by itself and describe the environment accurately. Thus, an accurate and efficient active SLAM algorithm needs to be designed for deformable environments.

The EKF algorithm is a typical estimation-theoretic based approach used to solve SLAM problems. The EKF based methods have been dominant in both classical and active SLAM for a long time in earlier stages of SLAM because of their high computational efficiency as compared with the optimization based approaches. Traditional EKF SLAM takes great advantage of the low computational cost when the number of features in the environment is limited. Still, it has been proved to get inconsistent estimates due to the wrongly

captured unobservable directions ([18–21]). In particular, the obtained covariance matrix in EKF SLAM is smaller than the actual estimate uncertainty, especially when the robot orientation error is large. Clearly, using the over confident covariance matrix obtained in EKF SLAM to guide the planning may lead to an unreliable robot trajectory in active SLAM.

Due to the inconsistency of EKF SLAM, optimization based SLAM methods become popular in the last decade, and it has also been frequently used in active SLAM. Using all information to estimate the SLAM results, the optimization based methods can obtain more accurate SLAM estimates and consistent uncertainty estimates. However, using all information on the other side leads to high computational costs. Especially when it is applied to the active SLAM problem where the SLAM algorithm needs to be performed frequently to predict the performance of each candidate action, the computational cost is really high.

As studied in ([22–24]), the SLAM problem has a nontrivial Lie group structure, and the Right Invariant EKF (RIEKF) algorithm designed on a specific Lie group structure is applied to solve the SLAM problem. It has proved that for feature based SLAM, the linearized system of RIEKF approach can automatically and correctly capture the unobservable direction of SLAM, ensuring strong consistency properties. The RIEKF based SLAM method is shown to be able to get better SLAM results with much improved consistency as compared with not only the traditional EKF based methods but also some improved ones [23] such as the observability constrained EKF (OC-EKF) [19]. When compared with the optimization based SLAM, its performance is shown to be comparable to iSAM [25] and close to full optimization based SLAM in many cases [26, 27].

1.2 Aims and objectives

This research aims to design effective active SLAM algorithms for feature based environments. We will mainly focus on the EKF based methods, especially the Right Invariant EKF algorithm. These algorithms will be designed for both static environments and deformable environments.

According to the aims, this research can be divided into the following tasks:

i. **RIEKF based active SLAM in 2D static environments.** Motivated by the recent research results on RIEKF SLAM, the first objective of this thesis is to apply RIEKF to the active SLAM algorithm, developing an efficient and accurate planning framework for the robot to explore unknown 2D environments.

ii. **RIEKF based active SLAM in 3D static environments.** In this task, we aim to extend our RIEKF based 2D active SLAM algorithm to 3D cases and improve the planning framework to solve the exploration-exploitation dilemma.

iii. **Active SLAM in 3D deformable environments.** The third objective of this thesis focuses on 3D deformable environments. We aim to design an active SLAM framework, where the robot is expected to take the environment changes into consideration so that it can successfully reach the target region by itself and map the latest environment accurately.

1.3 Contributions

We improved the exploration framework where both the predicted SLAM results for choosing control actions and the actual estimated SLAM results applying the selected control actions are computed using RIEKF algorithms. This is the first research work in this direction to consider a 2D case and apply the greedy method in the planning. The advantages over traditional EKF based active SLAM are the more accurate and consistent predicted uncertainty estimates, which result in the robustness of the active SLAM algorithm. The advantage over optimization based active SLAM is the reduced computational cost. Simulation results are presented to validate the advantages of the proposed algorithm.

As an extension of the 2D algorithm, we consider using the RIEKF method in active SLAM in 3D environments with obstacles. Similar to the 2D cases, both the predicted SLAM results for candidate control actions and the actual estimated SLAM results after applying the selected control actions are computed using RIEKF algorithms. Different from the previous work that used a weighted objective function to determine the robot's

motion in 2D, the newly designed planner is a combination of an efficient global planner and an accurate local planner in 3D. Results of the simulation and the real-world experiment demonstrate that our proposed method improves the estimation accuracy significantly and, at the same time, maintains a high exploration efficiency.

Finally, we consider the active SLAM problem in 3D deformable environments which has not been studied yet. Here, we have some reasonable assumptions about the feature dynamic model and the observation model. Based on these assumptions, an EKF based active SLAM framework is designed to estimate the SLAM result accurately and efficiently. The planner proposed is a combination of a global planner and a local planner. We compare the combined framework with using the local greedy method only and using the global planner only. Simulation results under different scenarios have shown that the proposed active SLAM algorithm provides a good balance between accuracy and efficiency as compared to the local planner and the global planner. Besides the simulation experiments, we also design an active SLAM system in a real-world phantom environment. This system uses a UR robot to handle an endoscope, which can move the camera to the expected poses that are calculated by our active SLAM algorithm. This system has been used to test the EKF based SLAM and active SLAM algorithms.

1.4 Publications

i. M. Xu, Y. Song, Y. Chen, S. Huang and Q. Hao. Invariant EKF based 2D Active SLAM with Exploration Task. In 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 5350-5356, doi: 10.1109/ICRA48506.2021.9561951.

ii. S. Huang, Y. Chen, L. Zhao, Y. Zhang and M. Xu. Some research questions for slam in deformable environments. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 7653-7660, doi: 10.1109/IROS51168.2021.9635883.

iii. M. Xu, L. Zhao, S. Huang and Q. Hao. Active SLAM in 3D Deformable Environments. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 7952-7958, doi: 10.1109/IROS47612.2022.9982224.

iv. M. Xu, S. Chen, L. Zhao, S. Huang and Q. Hao. Invariant EKF based 3D Active SLAM with Exploration Task. Submitted to IEEE Transactions on Emerging Topics in Computing (TETC) in Sep. 2024.

1.5 Thesis outline

This thesis is organized as follows:

- *Chapter 1*: This chapter discusses the background and motivation of the thesis, followed by the aims and objectives. We then highlight our contributions and publications.
- *Chapter 2*: This chapter presents the review of relevant literature on active SLAM algorithms in both static environments and deformable environments.
- *Chapter 3*: This chapter covers the background knowledge of EKF SLAM and RIEKF SLAM. The general EKF SLAM framework is first introduced, followed by the standard EKF SLAM. The 2D and 3D RIEKF algorithms are then introduced, respectively.
- *Chapter 4*: This chapter introduces our first work about RIEKF based active SLAM framework in 2D static environments. We first describe the problem we considered; then our proposed RIEKF based active SLAM method is presented in detail. In the experiments section, we introduce the experiment design and show the experiment results. The analysis of using different strategies under the considered criterion is also presented in the experiment section. Finally, we summarize the main results of this work. This work has been published in the IEEE International Conference on Robotics and Automation (ICRA) 2021 [28].
- *Chapter 5*: This chapter proposes a 3D active SLAM framework that is extended from the 2D case. The problem we consider in this work is given in the problem statement section. We then introduce each component of our proposed combined planner in detail. The experiments section presents the results and analysis of different planners

in different platforms, including simulation experiments in MATLAB and Gazebo and real-world experiments. The summary of this work is provided at the end of this chapter. This work has been submitted to IEEE Transactions on Emerging Topics in Computing (TETC) in Sep. 2024.

- *Chapter 6*: This chapter presents our work of active SLAM in deformable environments. We first describe the task of active SLAM in deformable environments, followed by the EKF based deformable SLAM model under reasonable assumptions. With the SLAM model, we then concretely describe the problem we consider and propose an efficient active SLAM framework to solve the problem. The framework is introduced in detail according to its components. We design two simulation environments to test our algorithms, and in the models and simulation settings section, we provide the details of these two environments. The simulation results section evaluates the performance of different algorithms in these two environments, respectively. We also carry out real-world experiments. The experiment settings and the main results are presented in the real-world experiments section. The summary and some discussions of this work are given in the summary section. The simulation part of this work has been published in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022 [29].

- *Chapter 7*: This chapter contains two sections. The first section gives the conclusion of this thesis, and the second section discusses potential future works.

Chapter 2

Literature Review

The task of exploring and modeling an unknown environment can be generally divided into three sub-tasks, including localization, mapping and motion control. Fig. 2.1 proposed by Makarenko et al. in [1] illustrates the relationships among the sub-tasks. Region I represents simultaneous localization and mapping (SLAM). As the name shows, SLAM is a method for autonomous mobile robots to build a map of an unknown environment while locating the robot within it [30]. Localization and mapping began to be solved simultaneously after the probabilistic approaches went mainstream in the 1990s. The poses of the robot and the location of the landmarks need to be estimated without any prior knowledge of the environment. Several families of algorithms have been designed to solve the SLAM problem in the last decades [2, 31–33]. The integration of motion control and mapping is active mapping or classic exploration, as Region II shows. It assumes that the sensor localization is known, aiming to search the optimal robot motion so that it can describe the environment accurately. There are different strategies to solve the classic exploration problem [34–37]. This problem is especially addressed in the field of computer vision for environment reconstruction [38, 39]. Region III represents active localization, which is the integration of localization and motion control. It assumes the map of the environment is known. The aim is to control the robot's motion to improve the robot pose estimation. This problem was first formulated by Fox et al. [40] and Borgi and Caglioti [41] and further investigated by many researchers [42–45]. The integration of all three components in Region IV is active SLAM. In this problem, the robot motion is planned

at the same time as the SLAM process, aiming to reduce the estimation uncertainty of the robot localization and map representation. Before the name "active SLAM" is given by Davison and Murray [46], it is also called active exploration [47], adaptive exploration [48], integrated exploration [1], and autonomous SLAM [49]. Active SLAM is illustrated as Region IV in Fig. 2.1.

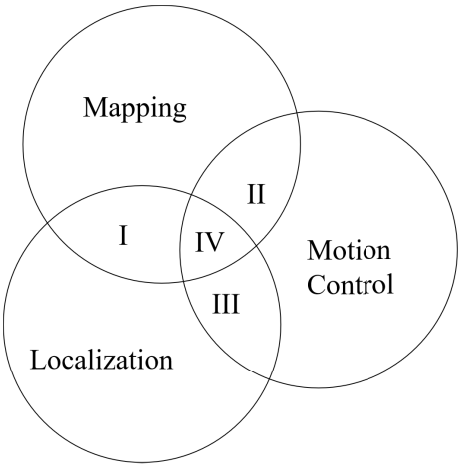


FIGURE 2.1: The relationships among the sub-tasks in modeling unknown environments [1].

2.1 Traditional methods for solving active SLAM problem

Motion control in SLAM is firstly addressed by Feder et al. in [2]. The objective of motion planning is to maximize the information obtained from observations so as to improve the estimation accuracy. It decouples the active SLAM problem into three stages: 1) Identification of the potential actions, 2) Utility computation, 3) Action selection and execution. This a traditional way to solve the active SLAM problem [1, 2, 50]. Placed et al. [4] have reviewed related work on each stage. Here, we also follow the three stages to review some of the work and focus more on work in the second stage.

2.1.1 Identification of potential actions

In traditional active SLAM approaches, the first step is to generate a set of available actions for the robot to execute. The actions can be goal points on the map, and they

can also be the robot motions. The format of the goal points strongly depends on the map representation. There are mainly four types of map representations to describe the environment, including topological maps, metric maps, metric-semantic maps, and hybrid maps. Topological maps use graphs to describe the environment information [51], but they are not frequently used in active SLAM [52, 53]. Metric maps can be divided into sparse maps and dense maps. Sparse maps use a set of features to represent the environment. This kind of representation has been widely used in optimal control [54–56]. They are the most typically used map representations in active SLAM. Occupancy grid maps are one of the most typically used dense maps. This kind of map representation is firstly proposed for perception and navigation [57, 58], and has been widely used in active SLAM [44, 59–62]. In 3D cases, the map is usually built based on octrees [63, 64] or voxel [65]. All of them have been used in active SLAM problems [66–69]. Metric-semantic maps extract the semantic information from the environment and associate it to classical metric maps [70–74], but only a few researchers consider this kind of representation in active SLAM [75, 76]. Hybrid and hierarchical maps combine some of the map representations to accomplish certain tasks like navigation [77], SLAM [78], or describing complex dynamic environments [79]. Only very few works use the hybrid maps in active SLAM [80]. In our research, we focus on the back end of the SLAM process, and the map we use is metric based.

A popular framework for active SLAM is selecting the best action from a finite set of candidate actions [81] [82]. Usually, a set of goal points is generated to guide the motion of the robot. The simplest way to choose the goal position is random-based exploration [83, 84]. In practice, frontier-based exploration [85] [36] is a more widely used approach, where a small subset of locations in the map is selected. The frontier is the region between the known and unknown areas of the map. It has been used in 2D environments with different kinds of map representations. Frontiers can be easily identified in topological maps as nodes without neighbors in certain directions [52]. In 2D occupancy grid maps, different frontier detection algorithms have been proposed, such as geometric based methods [86–88], RRTs based method [62, 89, 90] and sample based method [91]. Frontier based method is not widely used in 3D cases because of the high computational and storage cost to store and analyze the map information. Some works also use the above method in 3D cases

[92, 93], but most research make efforts to reduce the computational cost by evaluating map portions incrementally [69, 94] or only along surface [95], or by sampling particles in the known space [96]. Usually, after the frontiers are detected, a clustering step, like K-means [97] or mean-shift [89], is applied to restrict the dimension of the frontiers. However, traditional frontiers base exploration does not consider the estimation uncertainty. As studied in [49, 98, 99], revisiting previously observed areas may obtain more information. Therefore, to deal with the exploration-exploitation dilemma, the usual practice is to include the loop closure areas in the set of the goal points [100, 101], or switch between exploring frontiers and revisiting the previously observed regions [98, 102, 103]. Despite identifying global goal points, some work selects the goal points within a local area around the robot, which is widely used in reinforcement learning based methods [104, 105]. The disadvantage of this strategy is that only locally optimal results can be obtained, and when the decision making horizon is too short, wrong decisions may be made [106, 107]. Instead of selecting a specific destination, [54] introduces an attractor to influence the motion of the robot. The attractor is also selected from a series of potential points according to some specific criteria that balance the activities of exploring, improving localization and improving the map. Once selected, the attractor is in the form of an artificial feature, helps the robot to decide its action by influencing the information gain of the control actions.

2.1.2 Utility computation

The utility computation stage is the most focused stage by researchers in classical active SLAM. In this stage, the candidate actions will be evaluated according to some criterion.

In traditional exploration tasks, the utility function is usually formulated by the travel distance [36] or navigation time [92], and this kind of utility function can be used in all kinds of map representations [69, 101, 108]. Another typical method is considering the expected unknown area to be observed [83, 84, 89]. This kind of method is usually used in occupancy grid maps by counting the number of the known and unknown cells [1]. However, these approaches mainly focus on map coverage and collision avoidance, leaving the estimation uncertainty during the SLAM process unconsidered.

In classic active SLAM, one of the most important performance criteria is the quality of the SLAM estimate. Traditionally, the active SLAM problem is solved by minimizing a certain criterion in terms of the information gain, and the objective is to obtain more information [2, 5, 6]. To solve the exploration-exploitation dilemma, some works directly add a term of the information gain into the objective function [1, 52, 59], and some set thresholds to balance different factors [54, 89, 109]. However, tuning the parameters is complicated and makes a fair comparison of different metrics difficult [110].

To evaluate the information gain, the most common idea is to compare the uncertainty matrices. There are two groups of methods to quantify the uncertainty. One is using the information theory (IT), and the other is based on the theory of optimal experimental design (TOED). Information theory based methods mainly aim at occupancy grid maps, while the theory of optimal experimental design based approaches focus on Gaussian distributions over features and poses. In this section, we will first introduce these two groups of uncertainty quantifying approaches. After that, we focus on different methods of calculating the uncertainty of SLAM estimate under Gaussian distribution.

2.1.2.1 Uncertainty quantification

By using information theory, the uncertainty is quantified in the joint belief state to represent the entropy [111]. In the early stage, entropy only considers the map uncertainty, assuming no error in robot localization [36, 92, 112]. Later, Bourgault et al. [48] noticed that large errors in robot localization lead to wrong map uncertainties. Therefore, the robot localization uncertainty is taken into consideration to calculate the entropy [113]. However, the entropy of the SLAM posterior after a candidate action is executed is computationally intractable [113]. To address this challenge, many approaches rely on entropy approximations. These methods typically involve computing the utility of map and robot pose independently and then combining them heuristically [48, 102, 113, 114]. However, additional weights are usually needed to balance the two terms [81, 82]. Some work avoids this by embedding the robot pose uncertainty in a combined Shannon-Rényi utility function [59, 115]. Alternatively, the expectation-maximization (EM) algorithm [116] directly integrates the influence of the robot's uncertainty within a virtual map. In particle filter

SLAM, a similar approximation can be done by calculating the weighted mean of all possible particles [113, 117]. Instead of using Shannon's entropy directly as the utility in active SLAM, its expected reduction is commonly used to assess the utility. Concretely, the utility function is the difference between the entropy of the actual state and the expected entropy after an action is executed as the utility function, which is known as mutual information [48, 118]. Another commonly used utility function is Kullback-Leibler divergence (KLD) [119], which measures not only the change in the form of a probability density function but also how much its mean has translated [120].

For the approaches based on TOED, the uncertainty is quantified directly in the task space. Different from the IT based methods that count the number of cells in the grid map, the TOED based approaches apply to Gaussian variables. In TOED, the covariance matrix of executing a candidate action is predicted and compared. The one with a smaller covariance matrix is preferred. To compare the covariance matrices of the candidate actions, the optimality criteria is proposed and compared, including the A-optimality, T-optimality, E-optimality and D-optimality. A-optimality and T-optimality calculate the trace of the covariance matrix, where T-optimality captures the average variance, while A-optimality captures the harmonic mean variance [121]. E-optimality considers the maximum/minimum eigenvalue, capturing the radii of the covariance ellipsoid [122]. D-optimality captures the volume of the ellipsoid by calculating the determinant of the covariance matrix [123]. Feder et al. [2] first used the optimal criteria in their work, where the inverse of the estimation error covariance matrix of the next step is maximized to obtain the desired action. After that, many active SLAM methods were proposed based on TOED, especially on T-optimality [54, 124] and D-optimality [102, 104].

The covariance matrix can be large and dense, which makes the quantification computationally intensive. Therefore, many works perform quantification on the inverse of the covariance, that is, the Fisher information matrix (FIM). However, it is still computationally expensive, especially when the state space gets large. To bypass this problem, some works alternatively analyze the connectivity of the underlying pose-graph in active graph-SLAM. It was firstly noticed in [125] that a graph with minimum spanning trees is D-optimality. Following that, the relationship between the spanning trees and the optimality criteria is further investigated and extended [126–129]. These results have been

used to solve different active SLAM tasks [3, 55, 130].

2.1.2.2 Uncertainty calculation for different SLAM algorithms

As one of the most popular SLAM methods, the EKF algorithm is also used in most of the earlier active SLAM works. However, it has been realized that the EKF based SLAM may result in inconsistent estimates [18] [19] [20] [21] because of the errors introduced in the linearization process. Particularly, the covariance matrix obtained in the EKF based SLAM is too optimistic, which is smaller than the actual estimate uncertainty. The problem is more serious when the robot orientation error is large. Obviously, using the over confident covariance matrix in active SLAM may lead to a poor robot trajectory. To solve the inconsistency problem in SLAM estimates, many researches try to reduce the influence of the robot orientation errors, like adding constrains and changing the SLAM format.For example, [19] proposed the observability constrained EKF (OC-EKF) SLAM. Although the SLAM estimates are improved, the inconsistency problem is not overcome completely.

The optimization based method is another typical method used to solve the SLAM problem. As all the information is used, the optimization based methods can obtain consistent uncertainty estimates and, as a result, much more accurate SLAM estimates. However, using all information conversely brings a big problem in the computational cost. Especially when it is used to solve the active SLAM problem, the optimization process will be performed frequently to predict the performance of the candidate robot's actions, leading to high computational cost. Therefore, most works on the optimization based active SLAM focus on improving its efficiency. For example, [131] proposed a sparsification method to reduce the computational complexity of the decision making process. In [132], an efficient active SLAM approach based on submap joining, graph topology and convex optimization is proposed and is shown to be effective in reducing the computational complexity.In [133], the authors propose a strategy for maintaining a sparse and scalable state representation for large scale mapping. The matrix operations are performed by blocks, which leads to extremely fast matrix manipulation and arithmetic operations used in nonlinear least squares.

Very recently, the Right Invariant Extended Kalman Filter (RIEKF) algorithm was designed and applied in SLAM [22] [23] [24] [134] [26]. The RIEKF based SLAM algorithm has been shown to be able to produce more accurate SLAM estimates with much improved consistency as compared with the traditional EKF based SLAM, as well as some improved ones [23]. In [134], the RIEKF SLAM has been shown to have some good convergence properties. Compared with the optimization based SLAM, the performance of the RIEKF based SLAM algorithm is also pleasant. The computational cost of RIEKF SLAM is close to the traditional EKF SLAM. The estimation accuracy of RIEKF SLAM is shown to be comparable to iSAM [25], and even close to the full optimization based SLAM in many cases [26] [27].

2.1.3 Action selection and execution

In this stage, the candidate actions are evaluated according to the utility function, and the optimal action is selected and executed. When the set of actions is discrete, the problem can be solved by enumeration, where the action with the maximum/minimum utility will be selected [36, 89, 93]. When the actions are the robot motions, the selected action can be directly executed by the robot. When the actions are the goal points, an additional path planner is needed to guide the robot to move toward the goal point without collision. Depending on the representation of the environment, plenty of path planning algorithms have been well used in simulation and in reality, such as A* [135] and RRT [136] in occupancy grid map, and visibility graph-based planner in topological feature graph [52]. Recently, [137] presented a visibility graph-based planning framework for navigating in both known and unknown environments.

Active SLAM problem can also be regarded as a stochastic optimal control problem, and this can be solved by using differential dynamic programming (DDP) or model predictive control (MPC). In [2], the greedy method was proposed, which aims to minimize the estimation error in the next step. Huang et al. [5] extended the greedy method into a multi-step look-ahead method. A variant of nonlinear MPC was proposed to obtain a multi-step optimization in EKF based SLAM system within a finite time horizon.

2.2 Active SLAM in deformable environments

SLAM in deformable environments is a very challenging research area. There are some works focusing on SLAM in dynamic environments, especially in the field of service robots [138] and autonomous driving [139], where features from moving objects need to be distinguished and removed from the SLAM process. Dynamic SLAM framework has been recently investigated by many researchers, where the motion of the moving objects is considered [140–142]. However, the moving objects are assumed to be rigid, so the environments are partially dynamic. For SLAM in deformable environments, there is no static part, and all the features considered in the SLAM estimate are dynamic.

Although there have been some initial works presented by different research groups, many basic research questions have not been discussed clearly. Most of the existing researches focus on the representation of the deformation and the reconstruction work. To build the deformable map, different data structures and geometric structures are designed according to different kinds of sensor data. When an RGB-D sensor is used, the observation can be used to build the map directly. For example, the DynamicFusion algorithm [7] uses volumetric data structure to represent the non-rigid scenes. In [9], SurfelWarp is proposed using a surfel based representation of the geometry. For surgical cases, a GPU is usually used to help with real-time stereo vision. MIS-SLAM [14] is a GPU-based SLAM algorithm. It combines the pose estimation with the ORB-SLAM [143], to deal with the deformable inside body environment. When using a monocular camera only, the problem becomes more challenging. DefSLAM, proposed in [15], presents a complete framework fusing energy-minimization pose estimation and isometric Non-rigid Structure-from-Motion (NRSfM) techniques [144–147]. However, the NRSfM problem itself is still a challenging problem in computer vision. More recently, [148] proposed NR-SLAM, a novel nonrigid monocular SLAM system founded on the combination of a dynamic deformation graph with a visco-elastic deformation model.

Although progress has been made in the area of SLAM in deformable environments, many fundamental questions remain unanswered. The active SLAM problem is one of the challenging questions. Planning for mobile robots in deformable environments has also received

attention a couple of decades ago [149, 150]. However, due to the unavailability and complexity of simulators for mobile robots in deformable environments and the difficulty in map representation, no efforts have been made towards developing a deformable active SLAM framework.

2.3 Reinforcement learning based active SLAM

In recent years, reinforcement learning (RL) has been used in robot path planning to accomplish different tasks due to its outstanding performance in large sequential decision making problems. All the above problems can be solved in a way of RL. In exploration tasks, RL can be combined with the frontier based exploration to help determine the frontier [61]. There are also works using RL to generate the robot motion [151] [152] directly according to the current state. When a goal position is given, the problem becomes a navigation problem, and plenty of RL based methods have been proposed to solve it [153] [154] [155]. The objective of the task is usually achieved by adjusting the reward function. The efficiency of the RL based methods is high when we focus on a single objective or conflict-free objectives, for example, map completeness in exploration and traveling time in navigation. However, it is challenging to design an effective reward function when complex factors are considered at the same time, such as map completeness and map accuracy. In [151] and [152], map accuracy is considered in the reward function as information gain and is compared with other reward settings. However, their focus is still on map completeness and trajectory length. In [156], a two-level framework is designed to achieve different tasks. The high level planner considers map coverage to determine the goal positions, and the low level planner calculates the robot path to maximize the map accuracy. In that work, the RL model is only trained for the low level planner focusing on map accuracy, and in the high level, traditional geometric coverage planners are used. As an improvement of [156], [157] designs RL models for the high level planner and low level planner separately, but both planners exploit the local information of the environment around the robot.

2.4 The focus of this thesis

In this thesis, we consider the point feature based map representation, where the observation noises and odometry noises are under Gaussian distribution. Focusing on the EKF and RIEKF algorithms, we aim to design efficient and accurate active SLAM frameworks for both static and deformable environments.

Chapter 3

EKF and RIEKF Algorithms

In this chapter, we introduce the mathematical and background knowledge about EKF and Right Invariant EKF (RIEKF) SLAM, which is required in this thesis. We start with the general framework of EKF SLAM. The uncertainty representation of the standard EKF framework will be first recalled. After that, the RIEKF algorithm will be introduced. The Jacobians of the 2D and 3D EKF and RIEKF will be presented, respectively.

3.1 The general EKF SLAM framework

In the considered SLAM problem, the n-th step state with M point features is a Gaussian model $\mathbf{X}_n \sim \mathcal{N}(\hat{\mathbf{X}}_n, \mathbf{P}_n)$, where $\hat{\mathbf{X}}_n$ is the mean estimate of $\mathbf{X}_n$ and $\mathbf{P}_n$ is the covariance matrix. It can be written as:

$$\mathbf{X}_n = (\mathbf{R}_n, \mathbf{x}_n, \mathbf{f}_n^1, \cdots, \mathbf{f}_n^M), \tag{3.1}$$

where $\mathbf{R}_n$, $\mathbf{x}_n$ and $\mathbf{f}_n^j$ $(j = 1, \cdots, M)$ are respectively the robot orientation, robot position, and the coordinate of the j-th feature, all described in the fixed world coordinate frame.

A general motion model for a moving robot and static features can be represented by

$$\mathbf{X}_{n+1} = f(\mathbf{X}_n, \mathbf{u}_n, \mathbf{w}_n), \tag{3.2}$$

where $\mathbf{u}_n$ is the odometry, and $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{\Phi}_n)$ is the odometry noise at time n.

Suppose there are K features $\left\{\mathbf{f}_{n+1}^1, \cdots, \mathbf{f}_{n+1}^K\right\}$ observed at time step $n+1$, the observation model can be given by

$$\begin{aligned} \mathbf{Z}_{n+1} &= h_{n+1}(\mathbf{X}_{n+1}, \boldsymbol{\xi}_{n+1}) \\ &= \begin{bmatrix} h_{n+1}^1(\mathbf{R}_{n+1}^T(\mathbf{f}_{n+1}^1 - \mathbf{x}_{n+1})) \\ \vdots \\ h_{n+1}^K(\mathbf{R}_{n+1}^T(\mathbf{f}_{n+1}^K - \mathbf{x}_{n+1})) \end{bmatrix} + \boldsymbol{\xi}_{n+1}, \end{aligned} \tag{3.3}$$

where $\boldsymbol{\xi}_{n+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Psi}_{n+1})$ is the observation noise. The covariance matrix $\mathbf{\Psi}_{n+1}$ is a block diagonal matrix consisting of all $\mathbf{\Psi}_{n+1}^i$ $(i \in O_{n+1})$, where O_{n+1} is the set of features observed at time $n+1$.

Suppose $\hat{\mathbf{X}}_n$ is the mean estimate of $\mathbf{X}_n$ and $\mathbf{e}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n)$ is the Gaussian noise vector, the RIEKF framework can be given in Algorithm 1.

Algorithm 1 The general EKF framework

Input: $\hat{\mathbf{X}}_n$, $\mathbf{P}_n$, $\mathbf{u}_n$, $\mathbf{Z}_{n+1}$

Output: $\hat{\mathbf{X}}_{n+1}$, $\mathbf{P}_{n+1}$

Propagation:

$\hat{\mathbf{X}}_{n+1|n} \leftarrow f(\hat{\mathbf{X}}_n, \mathbf{u}_n, \mathbf{0})$, $\mathbf{P}_{n+1n} \leftarrow \mathbf{F}_n\mathbf{P}_n\mathbf{F}_n^T + \mathbf{G}_n\mathbf{\Phi}_n\mathbf{G}_n^T$

Update:

$\mathbf{S}_{n+1} \leftarrow \mathbf{H}_{n+1}\mathbf{P}_{n+1|n}\mathbf{H}_{n+1}^T + \mathbf{\Psi}_{n+1}$

$\mathbf{K}_{n+1} \leftarrow \mathbf{P}_{n+1|n}\mathbf{H}_{n+1}^T\mathbf{S}_{n+1}^{-1}$

$\mathbf{y}_{n+1} \leftarrow \mathbf{Z}_{n+1} - h_{n+1}(\hat{\mathbf{X}}_{n+1|n}, \mathbf{0})$

$\mathbf{X}_{n+1} \leftarrow \hat{\mathbf{X}}_{n+1|n} \oplus \mathbf{K}_{n+1}\mathbf{y}_{n+1}$

$\mathbf{P}_{n+1} \leftarrow (\mathbf{I} - \mathbf{K}_{n+1}\mathbf{H}_{n+1})\mathbf{P}_{n+1|n}$

Here, $\mathbf{S}_{n+1}$ is the innovation covariance, $\mathbf{y}_{n+1}$ is called innovation, and $\mathbf{K}_{n+1}$ is the Kalman gain. The operator $\oplus$ is called retraction in differentiable geometry [158], and the choices of $\oplus$ of EKF and RIEKF are different.

3.2 The standard EKF SLAM

The robot orientation is usually described by Euler angles, and then the state can be defined as:

$$\mathbf{X}_n = (\boldsymbol{\theta}_n, \mathbf{x}_n, \mathbf{f}_n^1, \cdots, \mathbf{f}_n^M), \tag{3.4}$$

In the standard EKF SLAM, the $\oplus$ is the standard "+", so the estimation error can be written as

$$\mathbf{e}_n = \mathbf{X}_n - \hat{\mathbf{X}}_{n+1|n},\ \mathbf{e}_{n+1|n} = \mathbf{X}_{n+1} - \hat{\mathbf{X}}_{n+1|n} \tag{3.5}$$

The Jacobians in Algorithm 1, $\mathbf{F}_n$, $\mathbf{G}_n$, and $\mathbf{H}_{n+1}$, can be calculated by $\mathbf{F}_n = \frac{\partial f}{\partial \mathbf{X}}(\hat{\mathbf{X}}_n, \mathbf{u}_n, \mathbf{0})$, $\mathbf{G}_n = \frac{\partial f}{\partial \mathbf{w}}(\hat{\mathbf{X}}_n, \mathbf{u}_n, \mathbf{0})$, $\mathbf{H}_n = \frac{\partial h_{n+1}}{\partial \mathbf{X}}(\hat{\mathbf{X}}_{n+1|n})$.

We then get:

$$\begin{aligned} \mathbf{e}_{n+1|n} &= \mathbf{F}_n \mathbf{e}_n + \mathbf{G}_n \mathbf{w}_n \\ \mathbf{Z}_{n+1} - h_{n+1}(\hat{\mathbf{X}}_{n+1|n}) &= \mathbf{H}_{n+1}\mathbf{e}_{n+1|n} + \boldsymbol{\xi}_{n+1} \end{aligned} \tag{3.6}$$

The Kalman gain $\mathbf{K}_{n+1}$ can be obtained using $\mathbf{F}_n$, $\mathbf{G}_n$, and $\mathbf{H}_{n+1}$. Given $\mathbf{y}_{n+1} = \mathbf{Z}_{n+1} - h_{n+1}(\hat{\mathbf{X}}_{n+1|n}, \mathbf{0})$, the estimate of the error $\mathbf{X}_{n+1} - \hat{\mathbf{X}}_{n+1|n}$ can be calculated by $\mathbf{e}_{n+1} = \mathbf{K}_{n+1}\mathbf{y}_{n+1}$. Therefore, the state can be updated accordingly:

$$\mathbf{X}_{n+1} = \hat{\mathbf{X}}_{n+1|n} + \mathbf{K}_{n+1}\mathbf{y}_{n+1} \tag{3.7}$$

3.3 RIEKF based SLAM

SLAM problem has a nontrivial Lie group structure. Different from the EKF method, RIEKF performs the linearization on Lie groups. The linearized system of RIEKF approach can automatically and correctly capture the unobservable direction of SLAM, ensuring strong consistency properties [23].

3.3.1 2D RIEKF

The 2D RIEKF SLAM algorithm was first proposed in [23]. In the considered 2D SLAM problem, the notation $\mathbf{R}_n$ in eq. (3.1) is the rotation matrix related to the orientation θ:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

And we denote $\mathbb{SO}(2)$ as the set of all 2D rotation matrices.

The set of all such possible states is denoted as

$$\mathcal{G}(M) = \left\{ \left(\mathbf{R}, \mathbf{x}, \mathbf{f}^1, \cdots, \mathbf{f}^M\right) | \mathbf{R} \in \mathbb{SO}(2), \mathbf{x} \text{ and } \mathbf{f}^i \in \mathbb{R}^2 \right\}. \tag{3.8}$$

It is a Lie group with the group action

$$\mathbf{X}_1 \oplus \mathbf{X}_2 = \left(\mathbf{R}_1\mathbf{R}_2, \mathbf{R}_1\mathbf{x}_2 + \mathbf{x}_1, \cdots, \mathbf{R}_1\mathbf{f}_2^M + \mathbf{f}_1^M\right), \tag{3.9}$$

for all $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{G}(M)$.

Denote $\mathfrak{g}(M)$ as the associated Lie algebra of $\mathcal{G}(M)$, which is isomorphic to $\mathbb{R}^{2M+3}$. And the exponential mapping $\exp(\cdot)$ from $\mathfrak{g}(M)$ to $\mathcal{G}(M)$ is defined by

$$\exp(\mathbf{e}) = \left(\mathbf{R}(\delta\theta), \mathbf{B}(\delta\theta)\boldsymbol{\delta}\mathbf{x}, \mathbf{B}(\delta\theta)\boldsymbol{\delta}\mathbf{f}^1, \cdots, \mathbf{B}(\delta\theta)\boldsymbol{\delta}\mathbf{f}^M\right), \tag{3.10}$$

where

$$\begin{aligned} \mathbf{e}^T &= (\delta\theta, \boldsymbol{\delta}\mathbf{x}^T, \boldsymbol{\delta}\mathbf{f}^{1T}, \cdots, \boldsymbol{\delta}\mathbf{f}^{MT}), \\ \mathbf{B}(\delta\theta) &= \begin{bmatrix} \frac{\sin(\delta\theta)}{\delta\theta} & -\frac{1-\cos(\delta\theta)}{\delta\theta} \\ \frac{1-\cos(\delta\theta)}{\delta\theta} & \frac{\sin(\delta\theta)}{\delta\theta} \end{bmatrix}. \end{aligned} \tag{3.11}$$

Then we can define the stochastic model on the proposed Lie group by

$$\mathbf{X} = \exp(\mathbf{e}) \oplus \hat{\mathbf{X}}, \tag{3.12}$$

where $\hat{\mathbf{X}}$ represents the mean, and $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$ is the error with covariance matrix $\mathbf{P}$. Then $\mathbf{X}$ is said to be log-Gaussian, denoted by $\mathbf{X} \sim \mathcal{N}_{log}(\hat{\mathbf{X}}, \mathbf{P})$.

Considering the specific 2D SLAM problem, the process model is

$$\mathbf{X}_{n+1} = f(\mathbf{X}_n, \mathbf{u}_n, \mathbf{w}_n) = \begin{bmatrix} \mathbf{R}(\theta_n + \omega_n + \mathrm{w}_n^{\omega}) \\ \mathbf{R}(\theta_n)(\mathbf{v}_n + \mathbf{w}_n^{v})^T \\ \mathbf{f}_n^{\mathrm{l}} \\ \vdots \\ \mathbf{f}_n^{M} \end{bmatrix}, \tag{3.13}$$

where $\mathbf{u}_n = \begin{bmatrix} \omega_n \\ \mathbf{v}_n \end{bmatrix}$ is the control input, and $\mathbf{w}_n = \begin{bmatrix} \mathrm{w}_n^{\omega} \\ \mathbf{w}_n^{v} \end{bmatrix}$ is the noise.

Due to different linearization process (compared to the traditional EKF), the Jacobians in Algorithm 1, $\mathbf{F}_n$ is the identity matrix $\mathbf{I}_{2M+3}$, $\mathbf{G}_n$, and $\mathbf{H}_{n+1}$, are given by

$$\mathbf{G}_n = \begin{bmatrix} 1 & \mathbf{0}_{1,2} \\ -\boldsymbol{J}\hat{\mathbf{x}}_n & \mathbf{R}(\hat{\theta}_n) \\ -\boldsymbol{J}\hat{\mathbf{f}}_n^1 & \mathbf{0}_{2,2} \\ \vdots & \\ -\boldsymbol{J}\hat{\mathbf{f}}_n^N & \mathbf{0}_{2,2} \end{bmatrix}, \ \mathbf{H}_{n+1} = \begin{bmatrix} \boldsymbol{\nabla}\mathbf{h}_{n+1}^1 \mathbf{H}_{n+1}^1 \\ \boldsymbol{\nabla}\mathbf{h}_{n+1}^2 \mathbf{H}_{n+1}^2 \\ \vdots \\ \boldsymbol{\nabla}\mathbf{h}_{n+1}^K \mathbf{H}_{n+1}^K \end{bmatrix}, \tag{3.14}$$

where $\boldsymbol{J}$ represents the skew symmetric matrix:

$$\boldsymbol{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

$\boldsymbol{\nabla}\mathbf{h}_{n+1}^j$ is the Jacobian of $\mathbf{h}_{n+1}^j$ computed at $\mathbf{R}(\hat{\theta}_{n|n-1})^T(\hat{\mathbf{p}}_{n|n-1}^{l_j} - \hat{\mathbf{x}}_{n|n-1})$, and

$$\mathbf{H}_{n+1}^j = \begin{bmatrix} \mathbf{0}_{2,1} & -\mathbf{R}(\hat{\theta}_{n+1|n})^T & \mathbf{0}_{2,2j-2} & \mathbf{R}(\hat{\theta}_{n+1|n})^T & \mathbf{0}_{2,2K-2j} \end{bmatrix}.$$

3.3.2 3D RIEKF

The 3D RIEKF SLAM algorithm was first proposed in [134]. In 3D cases, the set of all such possible states is denoted as

$$\mathcal{G}(M) = \left\{ \left(\mathbf{R}, \mathbf{x}, \mathbf{f}^1, \cdots, \mathbf{f}^M\right) | \mathbf{R} \in \mathbb{SO}(3), \mathbf{x} \text{ and } \mathbf{f}^i \in \mathbb{R}^3 \right\}. \tag{3.15}$$

The Lie group action is the same as that in the 2D case:

$$\mathbf{X}_1 \oplus \mathbf{X}_2 = \left(\mathbf{R}_1\mathbf{R}_2, \mathbf{R}_1\mathbf{x}_2 + \mathbf{x}_1, \cdots, \mathbf{R}_1\mathbf{f}_2^N + \mathbf{f}_1^N\right), \tag{3.16}$$

for all $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{G}(M)$.

The process model can be written as:

$$\begin{aligned} \mathbf{X}_{n+1} &= f(\mathbf{X}_n, \mathbf{u}_n, \mathbf{w}_n), \\ &= (\mathbf{R}_n \exp^{\mathbb{SO}(3)}(\boldsymbol{\omega}_n)\mathbf{R}_n^u, \mathbf{x}_n + \mathbf{R}_n(\mathbf{v}_n + \mathbf{w}_n^v), \mathbf{f}_n^1, \cdots, \mathbf{f}_n^M), \end{aligned} \tag{3.17}$$

where $\mathbf{u}_n = \begin{bmatrix} \mathbf{R}_n^u \\ \mathbf{v}_n \end{bmatrix}$ is the odometry, and $\mathbf{w}_n = \begin{bmatrix} \boldsymbol{\omega}_n \\ \mathbf{w}_n^v \end{bmatrix}$ is the odometry noise at time n, and $\exp^{\mathbb{SO}(3)}(\cdot)$ is the exponential map of $\mathbb{SO}(3)$.

Denote $\mathfrak{g}(M)$ as the associated Lie algebra of $\mathcal{G}(M)$, which is isomorphic to $\mathbb{R}^{3M+6}$. The error state $\mathbf{e}$ in Lie algebra can be constructed as:

$$\mathbf{e}^T = ((\mathbf{e}^R)^T, (\mathbf{e}^x)^T, (\mathbf{e}^1)^T, \cdots, (\mathbf{e}^M)^T). \tag{3.18}$$

The exponential map on this Lie group can be defined by:

$$\begin{aligned} \exp(\mathbf{e}) =& (\exp^{\mathbb{SO}(3)}(\mathbf{e}^R), J_r(-\mathbf{e}^R)\mathbf{e}^x, \\ & J_r(-\mathbf{e}^R)\mathbf{e}^1, \cdots, J_r(-\mathbf{e}^R)\mathbf{e}^M), \end{aligned} \tag{3.19}$$

where $J_r(\mathbf{a}) = \sum_{k=0}^{\infty} \frac{(S(a))^k}{(k+1)!}$, and $S(\cdot)$ is the skew symmetric operator that transforms a 3D vector into a skew symmetric matrix. Then the error state $\mathbf{e}_n$, the estimated state $\hat{\mathbf{X}}_n$,

and the true state $\mathbf{X}_n$ satisfy

$$\mathbf{X}_n = \exp(\mathbf{e}_n) \oplus \hat{\mathbf{X}}_n. \tag{3.20}$$

The Jacobians in Algorithm 1, $\mathbf{F}_n$ is the identity matrix $\mathbf{I}_{3M+6}$, $\mathbf{G}_n$, and $\mathbf{H}_{n+1}$, are given by

$$\mathbf{G}_n = \begin{bmatrix} \hat{\mathbf{R}}_n & \mathbf{0}_{3,3} \\ S(\hat{\mathbf{x}}_n + \hat{\mathbf{R}}_n \mathbf{u})\hat{\mathbf{R}}_n & \hat{\mathbf{R}}_n \\ S(\hat{\mathbf{f}}_n^1)\hat{\mathbf{R}}_n & \mathbf{0}_{3,3} \\ \vdots & \\ S(\hat{\mathbf{f}}_n^M)\hat{\mathbf{R}}_n & \mathbf{0}_{3,3} \end{bmatrix}, \ \mathbf{H}_{n+1} = \begin{bmatrix} \boldsymbol{\nabla}\mathbf{h}_{n+1}^1 \mathbf{H}_{n+1}^1 \\ \boldsymbol{\nabla}\mathbf{h}_{n+1}^2 \mathbf{H}_{n+1}^2 \\ \vdots \\ \boldsymbol{\nabla}\mathbf{h}_{n+1}^K \mathbf{H}_{n+1}^K \end{bmatrix}, \tag{3.21}$$

where $\boldsymbol{\nabla}\mathbf{h}_{n+1}^j$ is the Jacobian of $\mathbf{h}_{n+1}^j$ computed at $\hat{\mathbf{R}}_{n|n-1}^T(\hat{\mathbf{f}}_{n|n-1}^j - \hat{\mathbf{x}}_{n|n-1})$, and

$$\mathbf{H}_{n+1}^j = \begin{bmatrix} \mathbf{0}_{3,3} & -\hat{\mathbf{R}}_{n+1|n}^T & \mathbf{0}_{3,3j-3} & \mathbf{R}_{n+1|n}^T & \mathbf{0}_{3,3K-3j} \end{bmatrix}.$$

3.4 Summary

We introduced the EKF and RIEKF SLAM algorithms mathematically in this chapter. We first presented the general EKF SLAM framework; then we gave the detailed equations of EKF and RIEKF SLAM, respectively. Both 2D and 3D cases are considered. These equations are the foundation of our active SLAM methods which will be introduced in the following chapters.

Chapter 4

Invariant EKF based 2D Active SLAM with Exploration Task

In this chapter, we introduce our first framework in RIEKF based 2D active SLAM problem. We propose to use the RIEKF SLAM algorithm in active SLAM, where both the predicted SLAM results for choosing control actions and the actual estimated SLAM results applying the selected control actions are computed using RIEKF algorithms. The advantages over traditional EKF based active SLAM are the more accurate and consistent predicted uncertainty estimates, which result in the robustness of the active SLAM algorithm. The advantage over optimization based active SLAM is the reduced computational cost. Simulation results are presented to validate the advantages of the proposed algorithm.

4.1 Problem statement

For the active SLAM problem considered in this chapter, given a complex indoor environment, we assume only the size of the environment is known. The robot starts from a fixed location in the environment. The objective is to plan the robot trajectory for a given time horizon so that it can explore the environment as much as possible and estimate the observed features and the robot poses accurately.

4.2 Method

4.2.1 Proposed active SLAM method

In this work, the RIEKF SLAM algorithm is used to predict the uncertainty of the SLAM estimate after taking a potential control action, and it is also used to perform the SLAM estimation after a control action is taken.

We use a one-step look-ahead strategy to maximize the information that will be obtained in the next step. In the RIEKF based method, the information gained in terms of the SLAM estimate can be described by the resulting covariance matrix after the control action is taken.

Concretely, given $\hat{\mathbf{X}}_n$ and $\mathbf{P}_n$, we would like to select the control vector $\mathbf{u}_n$ such that a certain metric (e.g. the trace) of the covariance matrix of the next step is optimized. That is, we want $\text{trace}(\mathbf{P}_{n+1})$ to be as small as possible:

$$obj = \min \text{trace}(\mathbf{P}_{n+1}), \tag{4.1}$$

where $\mathbf{P}_{n+1}$ is obtained by Algorithm 1.

Note that the feature observation $\mathbf{Z}_{n+1}$ is not available when planning is performed, so we assume that no new feature will be observed. Whether an old feature can be observed at time $n+1$ is determined according to the predicted state estimate $\hat{\mathbf{X}}_{n+1|n}$ and the sensor range.

In the proposed method, in order to take into account the exploration task in the planning, we consider not only the SLAM uncertainty, but also the distance, d, between the predicted robot position, $\hat{\mathbf{x}}_{n+1|n}$, and the goal point.

Thus, the objective function to be minimized is a weighted sum of $\text{trace}(\mathbf{P}_{n+1})$ and d

$$obj = w_p \text{trace}(\mathbf{P}_{n+1}) + w_d d, \tag{4.2}$$

where the relative weights w_p and w_d are dependent and can be adjusted continuously or as an abrupt mode switch.

4.2.2 Goal point selection

In each step of the planning, there is one goal point selected to be the next destination which is used in the objective function (4.2).

We first generate a list of exploration points based on the size of the environment called L_e. For example, the exploration points can be uniformly distributed in the environment. The number of exploration points depends on the sensor range of the robot.

The goal point is selected according to the current state, similar to [54]. There are three states in total, called *explore*, *improve localization*, and *improve map*.

The three states are described as follows.

- **Explore.** When the uncertainty is below a threshold *lowerbound*, the state is transformed to *explore*. The goal point is set to be the closest exploration point selected from the exploration point list L_e. We write the selected point as $p_{explore}$. When the robot has approached the destination, $p_{explore}$ is labeled to be explored and deleted from the exploration point list. Once all the exploration points are explored, the *explore* state is no longer available, and the *lowerbound* becomes invalid. The task of the robot is to improve localization or improve the map according to the *upperbound*.
- **Improve localization.** When the uncertainty is exceeding the threshold *upperbound*, the state is changed to *improve localization*, and we want the robot to re-visit a good feature, p_{good}, whose uncertainty is low enough. To keep the re-visiting efficiency, p_{good} should not be too far away from the robot. Thus, the goal point is set to be the feature with the lowest uncertainty within a predetermined distance from the robot.
- **Improve map.** Otherwise, the state is changed to *improve map* and a poor feature, p_{poor}, is selected to be the goal point, of which the uncertainty is the highest within a given distance from the robot.

With the goal point selected as above, we can obtain the distance d in (4.2) as:

$$d = \begin{cases} d_{explore}, & \text{trace}(\mathbf{P}) < lowerbound \text{ and } L_e \neq \emptyset \\ d_{poor}, & lowerbound \leq \text{trace}(\mathbf{P}) < upperbound \\ & \text{or } (\text{trace}(\mathbf{P}) < upperbound \text{ and } L_e = \emptyset) \\ d_{good}, & \text{trace}(\mathbf{P}) \geq upperbound, \end{cases} \tag{4.3}$$

where $d_{explore}$, d_{poor} and d_{good} are the distances from the estimated robot pose to $p_{explore}$, p_{poor} and p_{good} respectively. And $\mathbf{P}$ is the covariance matrix.

The thresholds are constantly adjusted according to the number of the observed features and the current steps:

$$\begin{aligned} upperbound &= w_k k + w_n n, \\ lowerbound &= w_k k + w_n n - const, \end{aligned} \tag{4.4}$$

where w_k is the weight of the number of the observed features k, w_n is the weight of the current total step n, and $const$ is a constant. As the values of the objective function are different in different algorithms, the values of w_k and w_n are adjusted according to the environment and the algorithms.

4.3 Experiments

4.3.1 Simulation settings

In this section, we evaluate the proposed RIEKF based active SLAM by comparing it with EKF based active SLAM and two optimization based active SLAM.

The process model for the active SLAM is shown in Section 3.3.1. And the specific observation map h^i_{n+1} for the i-th observed feature $\mathbf{f}^i$ at the $(n+1)$-th step is

$$h^i_{n+1}(\mathbf{q}) = \begin{bmatrix} \sqrt{q_1^2 + q_2^2} \\ \text{atan2}(q_2, q_1) \end{bmatrix}, \tag{4.5}$$

where $\mathbf{q} = [q_1, q_2]^T = \mathbf{R}(\theta_{n+1})^T(\mathbf{f}^i_{n+1} - \mathbf{x}_{n+1})$.

In our simulation, there are 50 features generated randomly in the range of 100 m $\times$ 100 m, and the total step of the robot is set to be 500. The initial robot pose is $[0, 0, 0]$, and the sensor range is 20 m. The covariance matrix of control noise $\mathbf{\Phi}_n$ is $\text{diag}[(0.02\text{rad})^2,\ (0.03\text{m})^2, (0.03\text{m})^2]$. And the covariance matrix of observation noise $\mathbf{\Psi}^i_n$ is set as $\text{diag}[(0.04\text{m})^2,\ (0.04\text{rad})^2]$.

The compared EKF based active SLAM is similar to that in [54]. It uses the same approach as RIEKF based active SLAM but replaces RIEKF with EKF.

In the first compared optimization based method, to calculate the next control, the information matrix of the next step can be directly obtained and maximized. Thus, its objective function is set to be:

$$obj = w_p \log(\det(\mathbf{\Lambda}_{n+1})) + w_d d, \tag{4.6}$$

where $\mathbf{\Lambda}_{n+1}$ is the information matrix.

The main computational requirement is the evaluation of the covariance update. In the EKF based method, the complexity is $\sim O(M^2)$, while in the traditional optimization based method, the complexity is $\sim O(n^2M^2)$, where n is the current time step, and M is the number of the feature. The dimension of the information matrix is very large due to the increased robot poses involved. To reduce the computational complexity, in the second compared optimization based method, we use the lower bound based optimization method proposed in [132] and [55]. Instead of processing the large information matrix, a lower uncertainty bound in terms of the log determinant of a weighted Laplacian matrix is calculated. We write the predicted lower uncertainty bound as $\mathcal{LB}_{n+1}$, then the objective

function becomes:

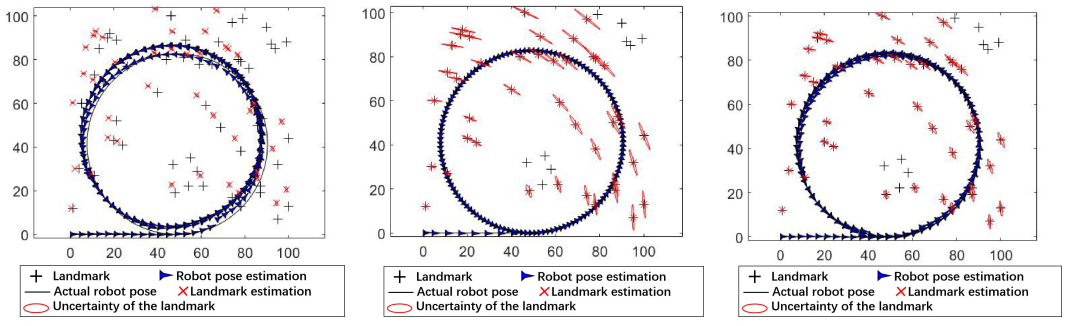
$$obj = w_p \log(\det(\mathcal{LB}_{n+1})) + w_d d. \tag{4.7}$$

The detailed formula about $\mathcal{LB}_{n+1}$ can be found in [132] and [55].

We compare the performance of these algorithms in terms of coverage, accuracy and processing time. The coverage is compared by counting the number of unexplored features. The accuracy is measured by calculating the maximal/average error between the estimated values and the ground truth, including the robot pose error and the feature position error. We record the processing time of both the decision making part and the SLAM part to compare the speed of determining the next control and the speed of estimating the current state, respectively. For each algorithm, we perform the simulation several times and present a representative result.

4.3.2 Results of using a predetermined path

We first present the different SLAM results using a predetermined path based on the EKF, optimization and RIEKF. The predetermined path is set to be a circle of which the radius is 45m.



(A) EKF (B) NLS (C) RIEKF

FIGURE 4.1: Result of using the predetermined path based on different SLAM methods in the environment with 50 randomly distributed features.

As Fig. 4.1A shows, the result of EKF SLAM is clearly inconsistent, as the actual feature positions of most of the features are out of the 99% confidence ellipses. The nonlinear least squares optimization (NLS) based SLAM and RIEKF based SLAM can both obtain good

quality estimates as shown in Fig. 4.1B and Fig. 4.1C, respectively. However, because the path is predetermined, 9 features remain undetected.

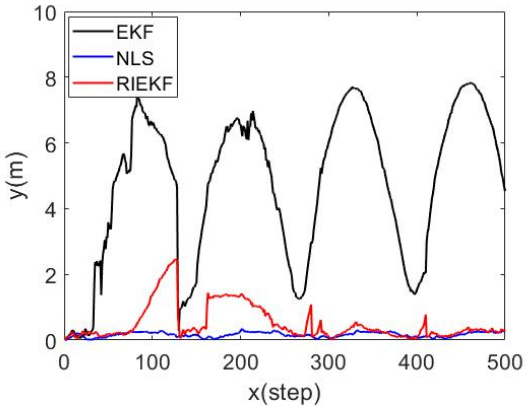


FIGURE 4.2: The pose error of using the predetermined path based on EKF, the optimization algorithm and the RIEKF.

Fig. 4.2 shows the robot pose error of using the predetermined path. Because of the inconsistency issue, the average errors of the EKF based method are almost ten times larger than those of the RIEKF based method. The RIEKF based method achieves much better results, and the pose error is similar to that of the optimization based method.

Table 4.1 shows the pose and feature estimation error of using the different approaches. The estimation error of optimization based algorithm is the smallest among the three algorithms.

TABLE 4.1: Estimation error comparison

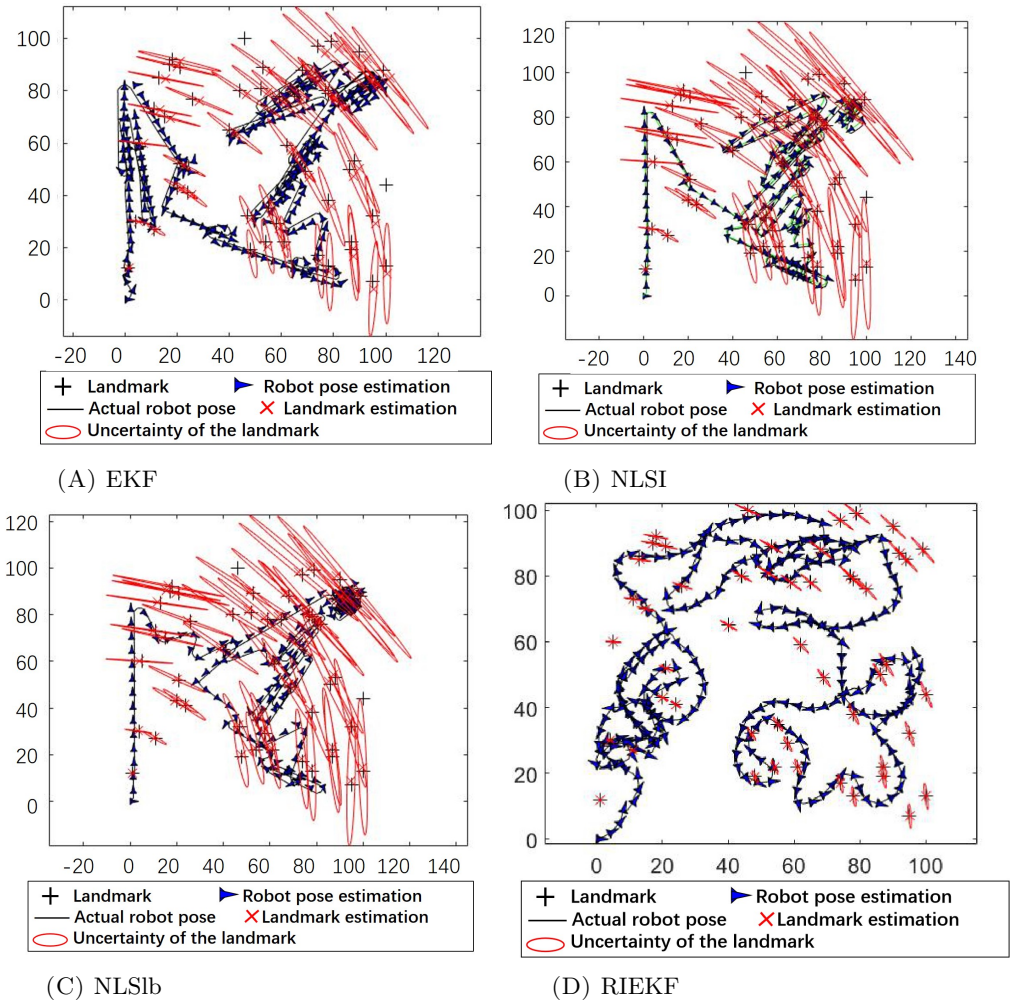
	Predetermined path		
	EKF	***NLS***	***RIEKF***
Maximum error robot (m)	7.8303	**0.3450**	2.4603
Average error robot (m)	4.7247	**0.1631**	0.5094
Maximum error feature (m)	8.9188	**0.3182**	0.3548
Average error feature (m)	5.9575	**0.1777**	0.2032

'Maximum error robot' and 'Average error robot' are, respectively, the maximal and average errors of the robot pose. 'Maximum error feature' and 'Average error feature' represent the maximal and average errors of the landmarks, respectively.

4.3.3 Comparison of the different active SLAM methods

4.3.3.1 Coverage

In this part, we show the coverage performance of the different methods under the same planning environment. The ground truth of the robot trajectory and the features and the results based on different methods (including the estimated poses and the estimated features, and the covariance ellipse of the features) are shown in Fig. 4.3A, Fig. 4.3B, Fig. 4.3C, and Fig. 4.3D.



(A) EKF (B) NLSI

(C) NLSlb (D) RIEKF

FIGURE 4.3: Result of using different active SLAM methods in the environment with 50 randomly distributed landmarks/features.

The performance of the coverage task is much better when the planning method is used, compared with the results given by the predetermined path. Fig. 4.3A shows the result of

the EKF based active SLAM algorithm. The accuracy is improved a lot by using the active SLAM algorithm. In general cases, there are 3 to 8 features remaining undetected, which is also better than using the predetermined path. Fig. 4.3B and Fig. 4.3C show the results of using the traditional optimization method and the lower bound based optimization method, respectively. There are 3 features left unseen in both of them. As shown in Fig. 4.3D, when applying the RIEKF based active SLAM algorithm, all features can be detected. Usually, all features can be detected within 290 steps.

4.3.3.2 Accuracy

In this part, we compare the accuracy performance of the obtained active SLAM results using the different methods. The results of the pose error based on different methods are shown in Fig. 4.4.

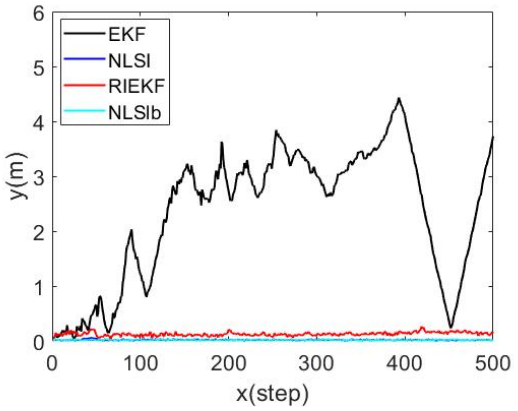


FIGURE 4.4: The robot pose error of using different active SLAM methods. The results of the two optimization based methods (NLSI and NLSlb) are very similar.

Two optimization algorithms achieve the best accuracy. The estimation error of the RIEKF based method is much smaller than the EKF based method in most cases. The average robot pose error and maximum robot pose error shown in Table 4.2 suggest that the RIEKF based method can obtain much smaller errors than EKF.

Besides the robot pose error, Table 4.2 also shows the maximum feature estimation error and the average feature estimation error in the last step. Similar to Section 4.3.2, the optimization algorithms have some advantages. Compared with the EKF based algorithm, the RIEKF based one can get smaller maximum error and average error.

By comparing Table 4.1 and Table 4.2, we can see the improvement in the accuracy by using planning algorithms.

TABLE 4.2: Estimation error comparison

	Active SLAM			
	EKF	***NLSI***	***NLSlb***	***RIEKF***
Maximum error robot	4.4331	0.0592	0.0459	0.2634
Average error robot	2.3257	0.0186	0.0161	0.1254
Maximum error feature	4.5722	1.8556	1.1315	0.1629
Average error feature	2.8808	1.1821	0.7541	0.1175

4.3.3.3 Processing time

For efficiency, we first compare the processing time in the SLAM part. The result is as expected. The optimization based method takes much longer time than the EKF and RIEKF based method. EKF based method and RIEKF based method take almost the same time for the SLAM part. Then, our focus is on the processing time in the decision making part.

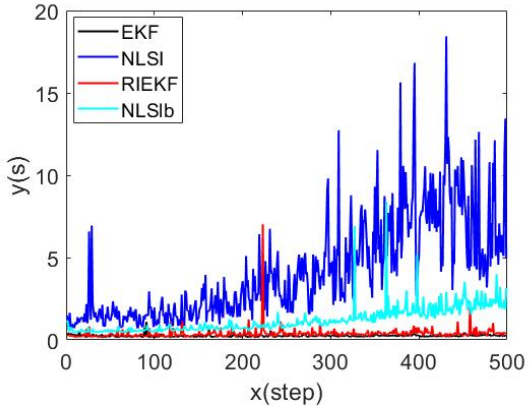


FIGURE 4.5: The comparison of the decision making time.

We can see in Fig. 4.5 that the EKF based method costs the least time, and the traditional optimization method NLSI costs much more time than others. The average value shows that the RIEKF based method costs about two times longer than the EKF based method. The lower bound based optimization method NLSlb is remarkable in reducing the computation time. Its speed in the first 300 steps is almost the same as the RIEKF

based method. However, with the increase in the number of steps, its computation time increases gradually.

4.4 Summary

In this chapter, we proposed an RIEKF based active SLAM algorithm in 2D cases. Because of the improved consistency, the proposed algorithm shows superior performance in accuracy as compared with EKF based active SLAM. It is also demonstrated that the proposed algorithm has acceptable performance in accuracy and much lower computational cost as compared with optimization based active SLAM algorithms. In the next chapter, we will consider 3D cases.

Chapter 5

Invariant EKF based 3D Active SLAM with Exploration Task

In this chapter, we introduce the RIEKF based active 3D SLAM algorithm that is extended from the 2D case. A new framework that combines a local planner and a global planner is proposed to solve the exploration problem. We first introduce the planning method that minimizes a certain criterion to get the action in each step, which we call the local planner here. Secondly, a global planner for efficient exploration is presented. Finally, we propose a combined planner that combines the local planning method and the global planning method.

5.1 Problem statement

For the active SLAM problem considered in this chapter, given a complex indoor environment, we assume the size of the environment is known. The robot starts from a fixed location in the environment. The objective is to plan the robot trajectory for a given time horizon so that it can explore the environment as much as possible and estimate the observed features and the robot poses accurately.

5.2 Method

To solve this problem, we propose an RIEKF based active SLAM framework as shown in Fig. 5.1. In our proposed method, the RIEKF SLAM algorithm is used in the SLAM module and the local planner module. In the SLAM process, it is used to perform the SLAM estimation after a control action is taken, as presented in Section 3. In the local planner, RIEKF is used to predict the uncertainty of the SLAM estimate after taking a potential control action, which will be introduced in Section 5.2.2.

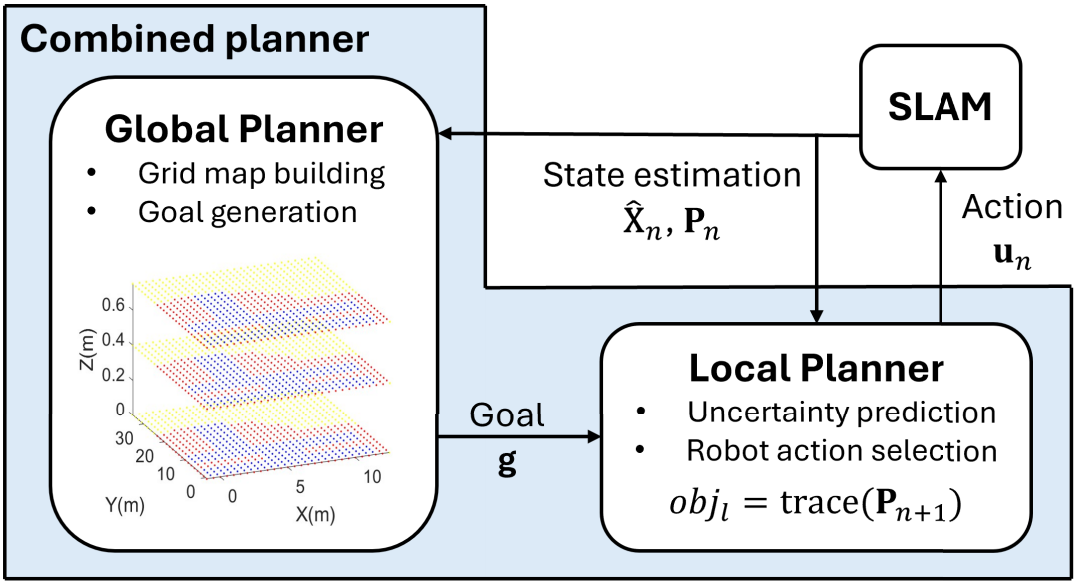


FIGURE 5.1: Our active SLAM framework.

In the following part of this section, we first introduce the global planner for efficient exploration. Secondly, we introduce the planning method that minimizes a certain criterion to get the action in each step, which we call the local planner here. Finally, we show how the combined planner works by combining the global planner and the local planner.

5.2.1 Global planner

The global planner takes the estimated robot pose and the observation information as input and outputs the goal points to guide the robot to explore the environments. In this part, a coarse occupancy grid map and a visibility graph are built. In each episode of the planning process, a goal point is generated according to the grid map, and a rough path is

planned from the robot to the goal point using the visibility graph method. An episode is ended once the robot reaches the current goal point. The visibility graph is also updated at the end of each episode.

5.2.1.1 Map building

In our work, we assume only the size of the environment is known, but the actual robot pose is unknown. The estimated robot pose from the SLAM process and observation are used to build the grid map. The grid map here is independent of the feature map estimated in the SLAM module. The resolution of the occupancy grid is determined by the robot sensor range and the environment complexity. As the global planner is a rough planner aiming to guide the robot to explore the environment as soon as possible, the resolution of the grid can be lowered to improve the planning efficiency.

There are three states of the cells in the grid, *occupied*, $free$, and *unknown*. When determining the state of a cell, we consider the center of each cell. We use L_e to represent the set of the centers of all cells, as shown in the global planner module in Fig. 5.1. Here, we call these centers exploration points. The states of the exploration points roughly indicate the environment information, and they are only updated at the end of each episode.

The introduction of the three states is as follows.

- **Occupied.** An exploration point is labeled as *occupied* if the point is within the robot sensor range but currently unobserved. The state *occupied* means the point is blocked by some obstacles and currently unreachable by the robot.
- **Free.** Once an exploration point is observed by the robot, it is marked as $free$. An exploration point with state $free$ is thought to be reachable by the robot.
- **Unknown.** All exploration points that are never covered by the robot sensor range are thought to be *unknown*. If the state of an exploration point is *unknown*, the area around the point is thought to be unexplored by the robot.

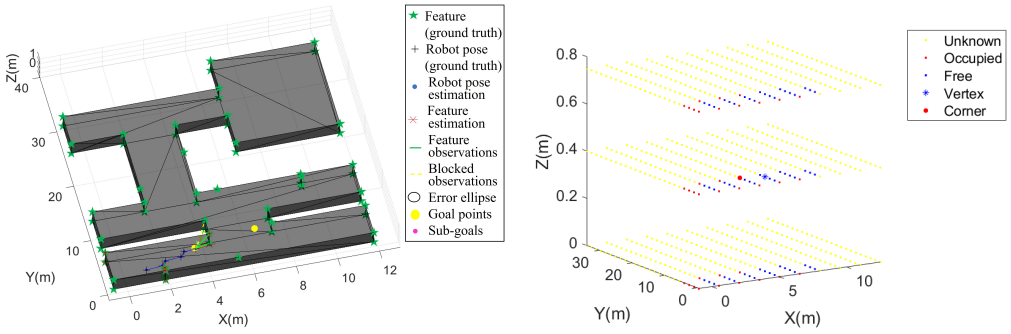
5.2.1.2 Goal selection

The goal point is selected from L_e according to the current state of the exploration points and the pose of the robot. We first determine the frontiers of the explored area of the environment. Similar to the definition in [36], a frontier represents the boundary between the known and unknown areas. Here, an exploration point is defined to be a frontier if its state is $free$ and there is at least one *unknown* point adjacent to it. More specifically, given a $free$ exploration point p, whose coordinate is $[x, y, z]$, if one of the six points, whose coordinates are $[x \pm 1, y, z]$, $[x, y \pm 1, z]$, or $[x, y, z \pm 1]$, is labeled to be *unknown*, the point p is determined to be a frontier.

We determine all frontiers in this way and save them in a frontier set $L_f \in L_e$. These frontiers will be evaluated by calculating an objective function, and the one with the highest score will be selected as the goal point. Here, we consider the exploration efficiency, where the robot is expected to observe more unknown areas. Therefore, the point that is surrounded by more *unknown* points will get a higher score. When counting the number of the *unknown* points surrounding a frontier p_f, we just consider the local area surrounding p_f. In our work, the local area is set to be an $a \times b \times c$ box centered at p_f, and only the points within the box will be counted. The values of a, b and c are determined according to the robot sensor range. We use the notation $r(p_f)$ to represent the number of *unknown* points surrounding p_f, then the objective function can be formulated by:

$$obj_g = r(p_f). \tag{5.1}$$

Fig. 5.2 gives an example. Fig. 5.2A shows an indoor environment where the robot starts from $[1, 1.5, 0.5]$ and has reached the first goal point. The coordinate of the first goal point is $[3.5, 3.8, 0.5]$, and is shown by a yellow dot. The corresponding exploration points and their states have been updated and are shown in Fig. 5.2B, where the blue points are $free$, the red points are *occupied*, and the yellow points are *unknown*. With the information of the point state, the goal point can be determined by maximizing obj_g in eq. (5.1) resulting the yellow dot at $[6.5, 5.3, 0.5]$ in Fig. 5.2A.



(A) Indoor environment (B) Exploration points

FIGURE 5.2: An example of the goal selection and visibility graph building process in the global planner.

5.2.1.3 Visibility graph for robot navigation

The visibility graph in our method aims to guide the robot to avoid obstacles so that it can reach the goal point without being blocked. As this is a rough planner, a set of sparse sub-goals is enough to guide the robot.

To make the robot successfully avoid the obstacles on the way to the goal point, we need to indicate the corners of the obstacles first.

When considering the 3D environment, we can identify the corners in different layers. As shown in Fig. 5.2B, the environment is divided into 3 layers. We take the second layer as an example. An *occupied* point p is labeled as a corner if two points that are adjacent to p and along different axes are $free$. Specifically, given the coordinate of p is $[x, y]$, then there are four points adjacent to p, and their coordinates are $[x, y \pm 1]$ and $[x \pm 1, y]$. We take $[x, y+1]$ and $[x-1, y]$ as an example. They are along different axes. If both of them are $free$, we consider p as a corner. In the figure, the larger red point is a corner of the second layer.

When planning the robot path, we need a $free$ point to guide the robot to bypass the corner. We still use the example of p that is described above. We have assumed that p is a corner, and $[x, y+1]$ and $[x-1, y]$ are $free$. Then, if $[x-1, y+1]$ is free, we assign the point $[x-1, y+1]$ to be a visibility vertex of the visibility graph. As Fig. 5.2B shows, the blue asterisk (*) is a visibility vertex. After indicating the vertices, the visibility graph can be built by connecting the vertices with non-blocking visibility edges. Two vertices

can be connected if the edge between them does not go through any *occupied* or *unknown* regions.

Given the robot position and the goal, we would like to search the graph for the shortest path between them. This can be solved by the visibility graph method [159]. The visibility vertices involved in the path are set to be the sub-goals for the robot to follow.

5.2.2 Local planner

The input of the local planner module is the goal point generated from the global planner, the estimated robot pose and feature positions and the covariance matrix from the SLAM module. The local planner uses the greedy method, where the information that will be obtained in the next step is maximized to obtain the control action of the robot. The information gained in terms of the SLAM estimate can be described by the resulting covariance matrix after the control action is taken and the information from the new observations is used.

Specifically, given $\hat{\mathbf{X}}_n$ and $\mathbf{P}_n$, the trace of the covariance matrix of the next step, $\text{trace}(\mathbf{P}_{n+1})$, is minimized to obtain the optimal control vector $\mathbf{u}_n$:

$$obj = \min \text{trace}(\mathbf{P}_{n+1}), \tag{5.2}$$

where $\mathbf{P}_{n+1}$ is obtained by Algorithm. 1.

Note that the feature observation $\mathbf{Z}_{n+1}$ is not available when the planning is performed, so we assume that no new feature will be observed. Whether an old feature can be observed at time $n+1$ is determined according to the predicted state estimate $\hat{\mathbf{X}}_{n+1|n}$ and the sensor range.

Note that the covariance matrix here is also calculated using RIEKF, so the final robot action is determined by the RIEKF result.

5.2.3 Combined planner

The combined planner is a combination of the local planner and the global planner, aiming to improve the SLAM estimation accuracy while exploring the environment.

In each episode, we use the global planner to generate a goal point to guide the robot to observe more unknown regions. A series of sub-goals from the robot to the goal point is also generated by the global planner to avoid collision. Then, we use the local planner to follow the sub-goals in order and, at the same time, improve the accuracy by minimizing the SLAM estimation uncertainty of the next step calculated using RIEKF.

In particular, the robot action $\mathbf{u}_n$ is selected from a set of candidate actions $\mathbf{U} = \{\mathbf{u}_n^i \mid i = 1, 2, ..., N_u\}$ in each step, where N_u is the number of the candidate actions. The robot pose after taking each candidate action is predicted, and the distance D between the predicted robot pose and the current sub-goal is calculated. The action set is sorted according to the distances from small to large, and then the first n_u candidate actions are selected to be the new action set. This process ensures that all actions in the new action set are likely to guide the robot to approach the goal point. The final action will be selected from the new action set using the local planner, which minimizes the trace of the covariance matrix that is based on RIEKF, as shown in eq. (5.2).

5.3 Experiments

Our algorithm is tested both in simulation and in the real world. In each of the experiments, the robot starts from a fixed pose, aiming to explore the whole environment as soon as possible and, at the same time, obtain more accurate SLAM estimates.

We do simulations in both MATLAB and Gazebo. In the simulation part, four algorithms are compared:

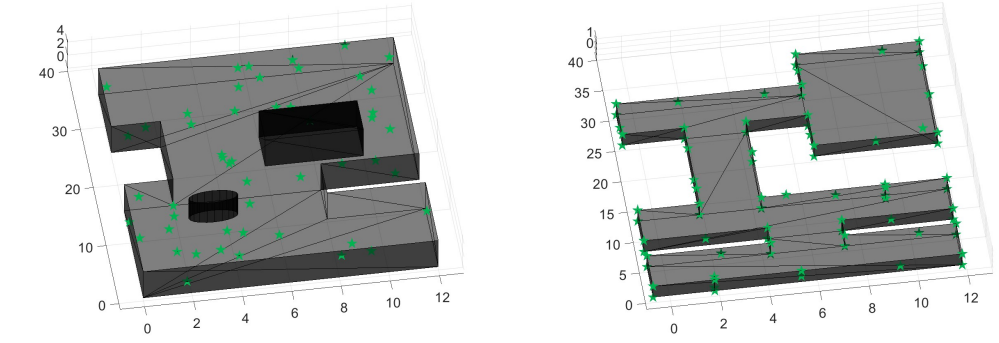
i. **RIEKF based combined planner.** This is our proposed method, as Fig. 5.1 shows, where RIEKF is used in the local planner and SLAM modules.

ii. **RIEKF based global planner.** In this algorithm, we still use the global planner that we propose to generate the goal points. After the goal point is generated, a short distance prior strategy is used for the robot to reach the goal. Therefore, RIEKF is not used in the planning process, but only used in the SLAM process.

iii. **RIEKF based circle planner.** In this algorithm, we still use the global planner that we propose to generate the goal points. After the goal point is generated, a short distance prior strategy is used for the robot to reach the goal. After reaching each goal, the robot rotates on the spot for 360 degrees. RIEKF is not used in the planning process, but only used in the SLAM process.

iv. **EKF based combined planner.** The framework is the same as Fig. 5.1, except that we use EKF instead of RIEKF in both the local planner and the SLAM modules.

After testing and evaluating in the simulation platforms, our proposed combined planner is also applied to the real-world robot for further verification.



(A) Environment 1 (B) Environment 2

FIGURE 5.3: Simulation environment in MATLAB. The gray structures are walls and obstacles. The green stars are sparsely distributed features.

5.3.1 Simulation in MATLAB

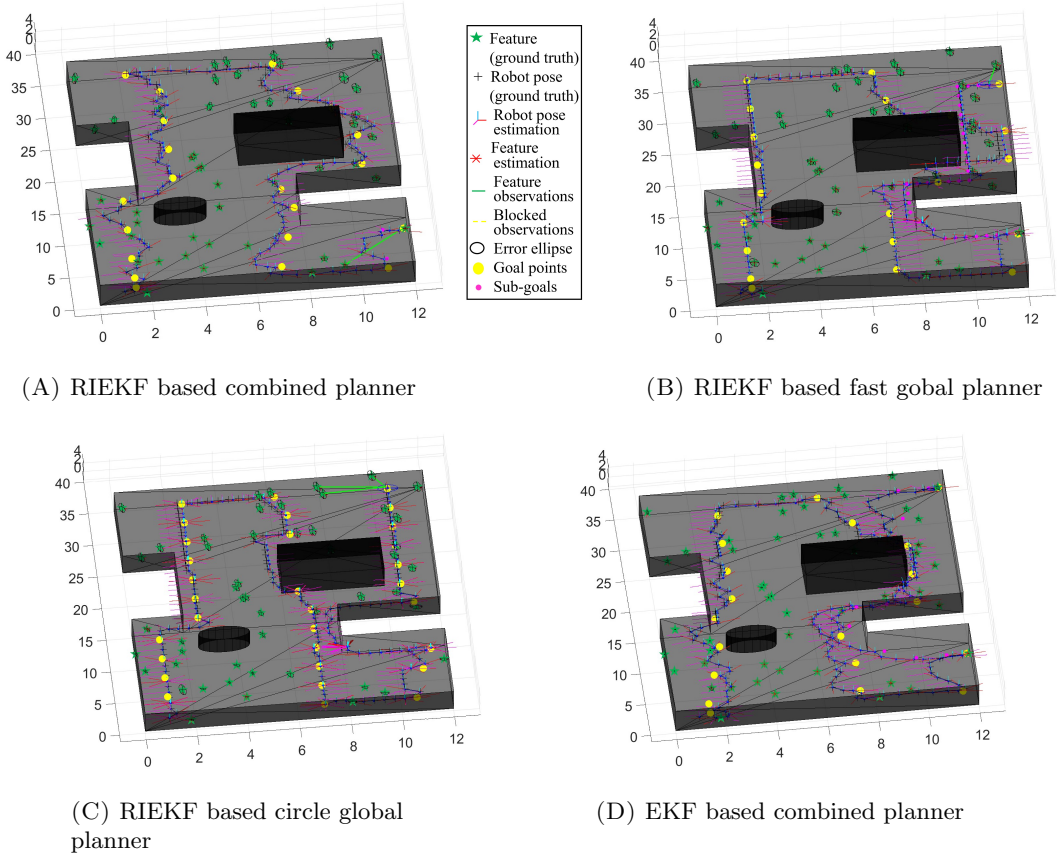
We designed two complex indoor environments in MATLAB, as shown in Fig. 5.3. Environment 1 is a room-like environment with obstacles. Environment 2 is a corridor-like environment with passages of varying widths. The features are sparsely distributed in the environments, and the simulated robot can move in the 3D space.

In our experiments, we set two groups of noise levels for both environments. Table 5.1 shows the covariance of odometry noise, $\mathbf{\Phi}_n$, and covariance of observation noise, $\mathbf{\Psi}_{n+1}$. In each environment, each algorithm is run 10 times starting at the same pose $[0, 0, 0, 1, 1.5, 0.5]$, and the total time step is set to be 300.

TABLE 5.1: Covariances of noises

Noise level	$\mathbf{\Phi}_n$	$\mathbf{\Psi}_n$
i	$0.005^2 \times \mathbf{I}_6$[1]	$0.005^2 \times \mathbf{I}_3$
ii	$0.01^2 \times \mathbf{I}_6$	$0.01^2 \times \mathbf{I}_3$

[1] $\mathbf{I}_m$ is the $m \times m$ identity matrix.



(A) RIEKF based combined planner

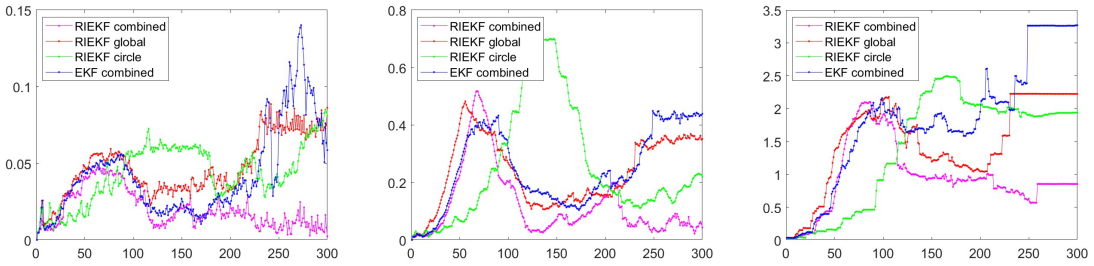
(B) RIEKF based fast gobal planner

(C) RIEKF based circle global planner

(D) EKF based combined planner

FIGURE 5.4: Results in MATLAB Environment 1 under noise level ii. The yellow dots are the goal points. The black/blue lines are the actual/estimated robot paths, and the coordinate systems on the blue line suggest the estimated robot pose at each step. The robot paths and features are in the 3D space, and the robot poses are 6 DOF.

Fig. 5.4 shows the resulting robot trajectories and the estimates of robot pose and feature positions by using different algorithms in Environment 1 in a representative run of noise



(A) RMSE for robot orientation

(B) RMSE for robot position

(C) RMSE for feature position

FIGURE 5.5: Accuracy (RMSE) of different algorithms in MATLAB Environment 1 under noise level ii.

level ii as described in Table 5.1. Instead of searching for the shortest path to the goal point, the RIEKF and EKF based combined planners consider minimizing the estimation uncertainty at each step.

We use the root mean squared error (RMSE) to evaluate the accuracy of each algorithm. Fig. 5.5 shows the average RMSE of 10 runs at each step under noise level ii. We can see that our proposed RIEKF based combined planner can get more accurate robot pose and feature position estimates during the whole time horizon.

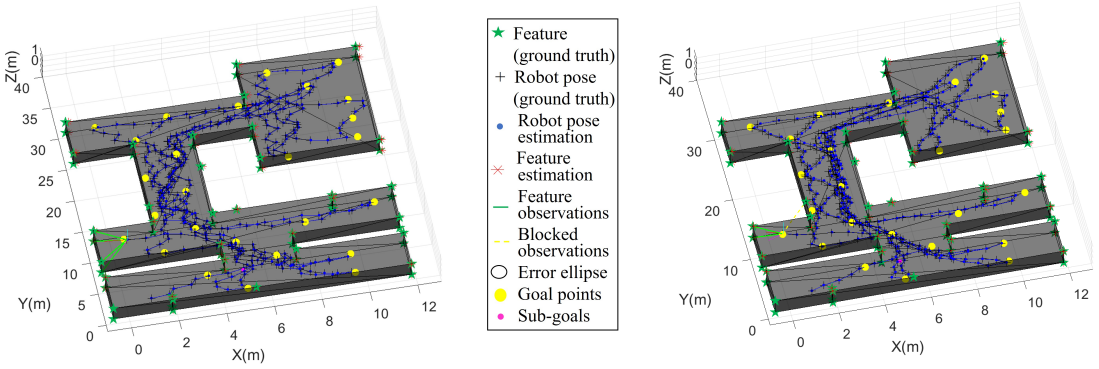
TABLE 5.2: Comparison in MATLAB Environment 1

	Noise level	***RIEKF Combined***	***RIEKF Global***	***RIEKF Circle***	***EKF Combined***
Num features	i	**49.1**	47.1	45	49.1
	ii	**49.9**	43.8	42	46.6
Steps	i	**207**	212	299	207
	ii	208	**205**	299	227
RMSE (Pose)	i	**0.1253**	0.2419	0.2599	0.2510
	ii	**0.3574**	0.5046	0.5046	0.9115
RMSE (Feature)	i	**0.1253**	1.4862	1.4058	0.2510
	ii	**0.3574**	2.4530	1.9899	0.9115
Plan time	i	0.2368	0.1771	**0.1299**	0.2340
	ii	0.3248	0.3886	**0.1478**	0.3098

[1] The total number of features in the environment is 50.

We record the detailed data in Table 5.2 to compare the performance of different algorithms. Besides the accuracy (RMSE), the table also shows the number of features observed in the whole time horizon (Num features), the time steps used to cover the entire

environment (Steps) and the planning time (Plan time). The number of features (Num feature) can be used to evaluate the exploration completeness, and Steps can be used to evaluate the exploration efficiency. The planning time is the time for the robot to decide its motion according to the current situation at each step. It includes the time for map building, goal selection, reference path planning and motion planning. For each noise level, the corresponding values in the table are the average values of 10 runs, so the values could be non-integers.



(A) RIEKF based combined planner

(B) RIEKF based gobal planner

(C) RIEKF based circle global planner

(D) EKF based combined planner

FIGURE 5.6: Results in MATLAB Environment 2 under noise level ii. The yellow dots are the goal points. The black/blue lines are the actual/estimated robot paths. The robot paths and features are in the 3D space, and the robot poses are 6 DOF.

The data supports that the RIEKF based combined planner can always get the most accurate estimates of the robot poses and feature positions. What's more, by using the RIEKF based combined planner, the robot can cover the entire environment and finish the exploration task in fewer time steps. All features can be observed during the exploration

process in almost all runs. The planning time of the circle global planner is the smallest because no decision needs to be made when the robot rotates at the goal point. For the RIEKF based combined planner, RIEKF based global planner and EKF based combined planner, the planning time doesn't differ too much, but because of the use of RIEKF in the planning process, the accuracy of RIEKF based combined planner improved a lot.

Similarly, we compare different algorithms in Environment 2 under two noise levels. Fig. 5.6 shows the results of using different algorithms in a representative run of noise level ii. Obviously, the EKF based planner cannot complete the exploration task because of the large errors caused by inconsistency. The results of the three RIEKF based planners are hard to evaluate according to the figures.

TABLE 5.3: Comparison in MATLAB Environment 2

	Noise level	***RIEKF Combined***	***RIEKF Global***	***RIEKF Circle***	***EKF Combined***
Num features [1]	i	**58**	58	58	56
	ii	**58**	57	58	42
Steps	i	399	**355**	506	399
	ii	399	**356**	546	-[2]
RMSE (Pose)	i	**0.0017**	0.0021	0.0017	0.1317
	ii	**0.0275**	0.0560	0.0794	2.3583
RMSE (Feature)	i	**0.0007**	0.0008	0.0015	0.2824
	ii	**0.0331**	0.0382	0.0543	1.6312
Plan time	i	0.0539	0.00375	**0.0361**	0.0686
	ii	0.0666	0.0680	**0.0256**	0.0448

[1] The total number of features in this environment is 60.
[2] The EKF based planner cannot complete the exploration task within the given time steps (1000 steps).

We also record the average data in Table 5.3. The data supports that the EKF based method can only succeed in the exploration task when the noise level is really small. Otherwise, it will fail because of the large estimation errors. It cannot select a proper goal point to guide the robot to explore the environment. For three RIEKF based methods, the RIEKF based circle planner has advantage over others in terms of observed features, but it takes much more time steps to complete the exploration task. The RIEKF based global planner takes relatively fewer time steps to complete the exploration task, but its estimation errors are larger than the combined planner. The RIEKF based combined

planner has great advantages in estimation accuracy. The planning time of the RIEKF based circle planner is the smallest, because no decision needs to be made when the robot rotates at the goal point. For the RIEKF based combined planner, RIEKF based global planner and EKF based combined planner, the planning time doesn't differ too much.

In general, the use of RIEKF in the SLAM estimation process can help obtain more accurate estimates than using EKF, and further helps the planning process. The use of RIEKF in the planning process can help select an efficient and accurate trajectory for the robot to complete the exploration task.

5.3.2 Simulation in Gazebo

In the Gazebo simulator, TurtleBot equipped with lidar (Rplidar) and RGB-D (Kinect) camera is used. Images of AprilTag are set on the wall and will be detected and recognized as features by the robot during the exploration process. Note that the features and observations are all 3D in the experiments, and the system is based on 3D algorithms. The simulation environment is as shown in Fig. 5.7.

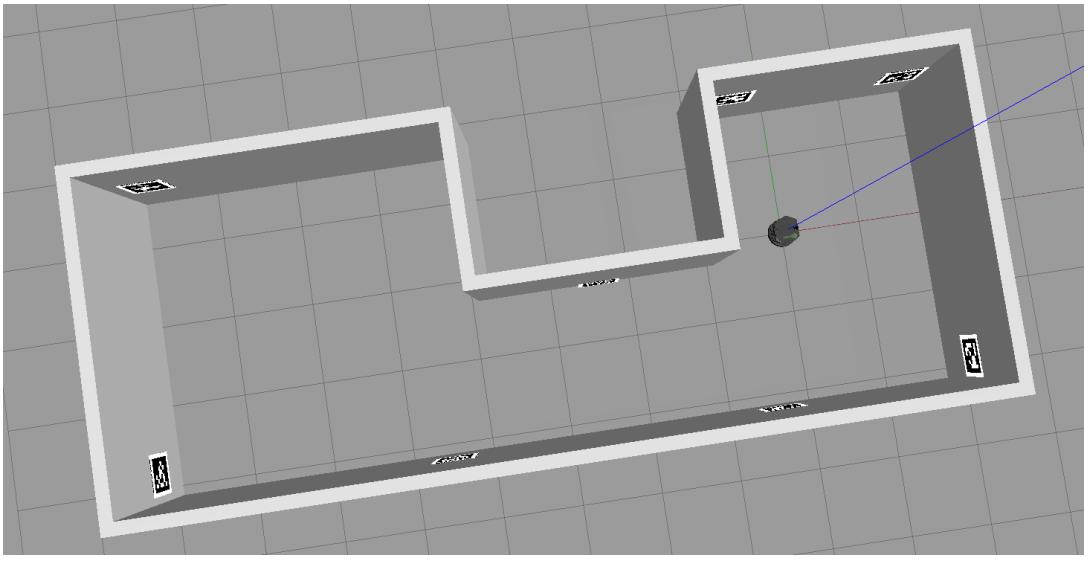
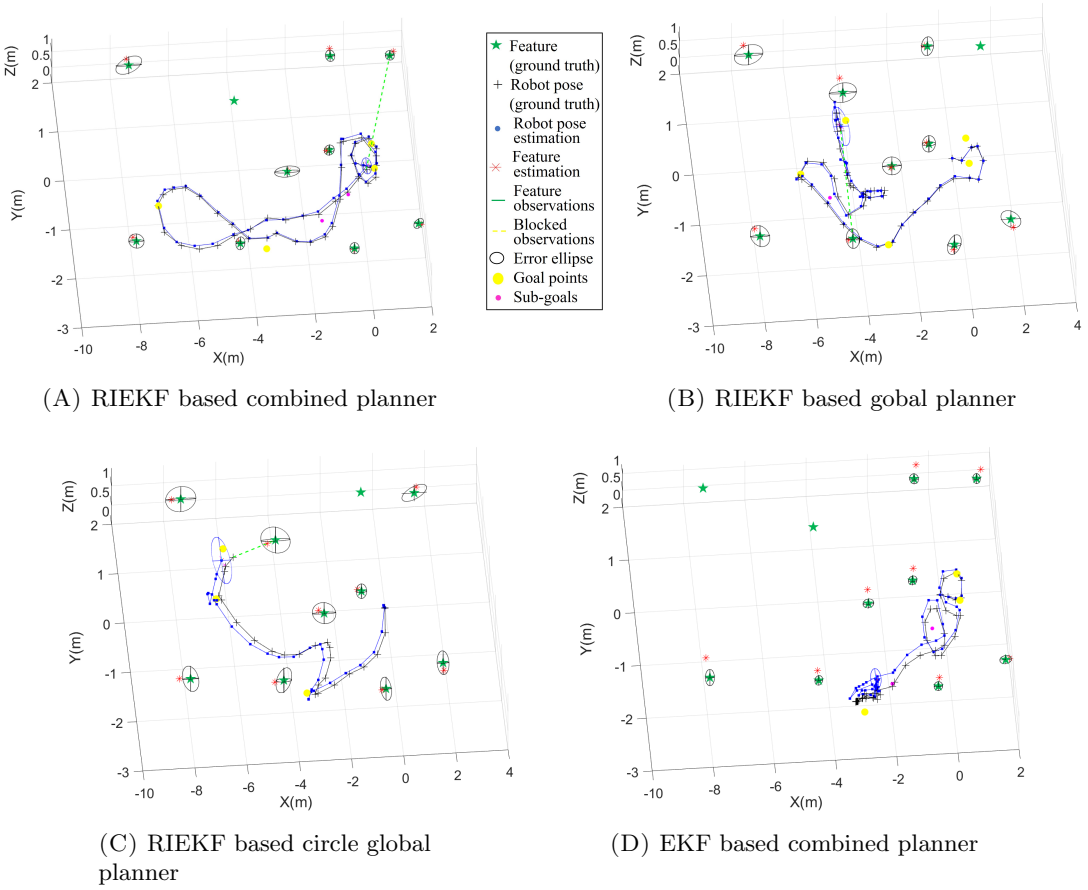


FIGURE 5.7: Simulation environment in Gazebo.

Similarly, we test three RIEKF based planners and an EKF based combined planner in this environment. Each of them is tested three times under the same noise level, and the



(A) RIEKF based combined planner

(B) RIEKF based gobal planner

(C) RIEKF based circle global planner

(D) EKF based combined planner

FIGURE 5.8: Results in the Gazebo environment.

average results are calculated. The results of the four planners in a representative run are shown in Fig. 5.8.

All of the RIEKF based planners can guide the robot in exploring the environment completely. The EKF based combined planner failed to complete the exploration task within the given time horizon in two of the three tests. Because the EKF based planner cannot build an accurate map for the environment, the selected goal point cannot guide the robot in exploring the whole environment.

The detailed data are recorded in Table 5.4. The RIEKF based combined planner can get the most accurate SLAM estimates and leave fewer features undetected. The RIEKF based circle planner takes the fewest steps in exploring the whole environment, because the environment is relatively small, but its estimation accuracy is the worst. The EKF

TABLE 5.4: Comparison in Gazebo

	RIEKF Combined	***RIEKF Global***	***RIEKF Circle***	***EKF Combined***
Num features [1]	**9.3**	8.6	9.0	7.3
Steps	50	53	**44**	- [2]
RMSE (Pose)	**0.0921**	0.1247	0.1672	0.2163
RMSE (Feature)	**0.1112**	0.1528	0.1926	0.2076
Plan time	0.0506	0.0273	**0.0168**	0.0558

[1] The total number of features in this environment is 10.
[2] The EKF based planner cannot complete the exploration task within the given time steps (500 steps).

based combined planner only successfully explores the environment once, but the errors are very large compared with the RIEKF based ones.

5.3.3 Real-world experiment

To test the performance of the proposed framework in real-time operating systems, we implement these three RIEKF based planners and an EKF based planner in an autonomous robot, LIMO, shown in Fig. 5.9. This robot can be viewed as a size-scaled road vehicle equipped with a 2D Lidar and an RGB camera. For the real-world experiments, due to equipment limitations, the robot can only move in a 2D plane. However, the features and observations are all in 3D, and the system is still based on 3D algorithms.

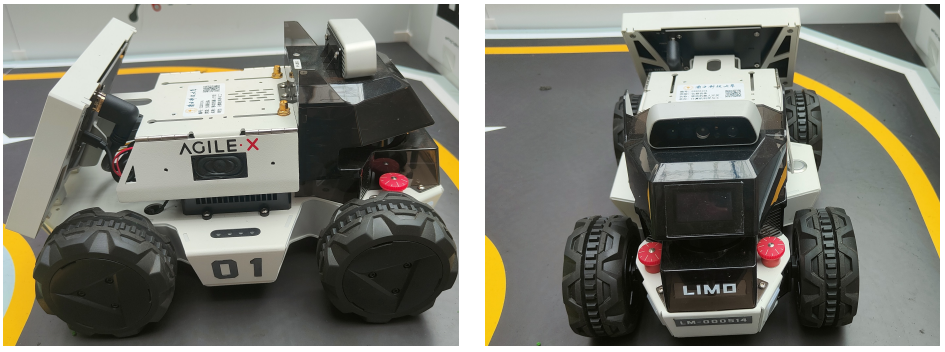


FIGURE 5.9: Autonomous robot, LIMO.

We design the testing environments in a squared area, as shown in Fig. 5.10. Obstacles are built within this area, and images of AprilTag are posted randomly on the walls and

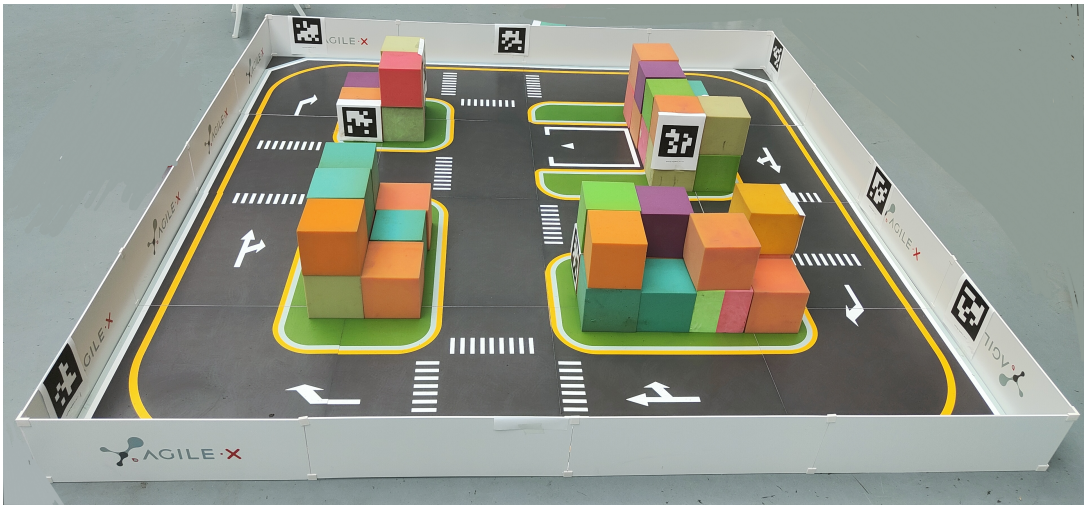
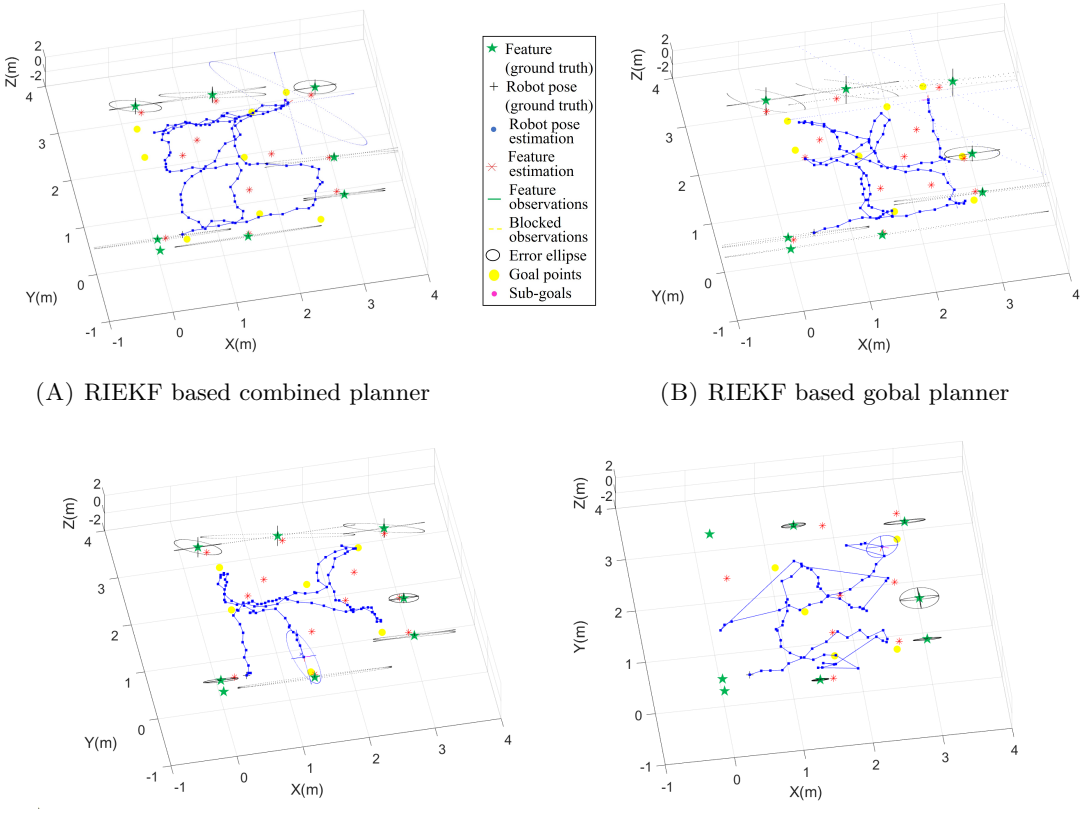


FIGURE 5.10: Real-world environment.

obstacles. We test the same four planners in this environment three times under the same noise level. The results of a representative run are shown in Fig. 5.11.



(A) RIEKF based combined planner

(B) RIEKF based gobal planner

(C) RIEKF based circle planner

(D) EKF based combined planner

FIGURE 5.11: Results in the real-world environment.

Similarly, for each algorithm, we record the average data in the three tests to compare their performance. For the performance in accuracy, it is hard to obtain the ground truth of the robot pose, so we just compare the RMSE for feature position. The results of the four planners are shown in Table 5.5. For the EKF based combined planner, it cannot complete the exploration task because of large estimation errors caused by inconsistency. Then, we compare the performance of three RIEKF based methods. As we can see, the number of observed features of the three planners doesn't differ much, but in general, the RIEKF based combined planner costs the fewest time steps to complete the task. The RMSE (feature) of the combined planner is also smaller than the other two. The main gap in planning time lies in reference path planning, because recursion is used to find the reference points, and the time varies depending on the robot position, goal position and the current map. The combined planner can make decisions with less average time, and more importantly, it is more stable, which means the robot can always explore the environment in an efficient path.

TABLE 5.5: Comparison in real-world environment

	RIEKF Combined	***RIEKF Global***	***RIEKF Circle***	***EKF Combined***
Num features [1]	**12**	12	12	11
Steps	**128**	135	158	- [2]
RMSE (Pose)	**0.1078**	0.1394	0.1189	0.4526
RMSE (Feature)	**0.0772**	0.5311	0.3221	0.5138

[1] The total number of features in this environment is 16.
[2] The EKF based planner cannot complete the exploration task within the given time steps (500 steps).

5.4 Summary

In this chapter, we propose a combined planner based on RIEKF for efficient and accurate exploration of unknown 3D static environments. Our proposed planner contains an efficient global planner that generates goal frontiers for the robot to explore the environment as soon as possible and an accurate local planner that determines the robot's motion considering SLAM estimation uncertainty. Our method is tested in both simulation and real-world environments. The combined framework is shown to have a good balance in exploration

efficiency and estimation accuracy. Compared with using the global planner only, the combined planner can guide the robot to explore the environments in fewer time steps and, at the same time, describe the environment more accurately. Comparison with EKF based framework demonstrates large advantages of using RIEKF in active SLAM. In the next chapter, we will consider active SLAM in deformable environments.

Chapter 6

Active SLAM in 3D Deformable Environments

This chapter considers the active SLAM problem for 3D deformable environments where the trajectory of the robot is planned to optimize the SLAM results. We first introduce the EKF SLAM framework in deformable environments. Secondly, a planning strategy combining an efficient global planner with an accurate local planner is proposed to solve the problem. We test our algorithms in two simulation environments and present the results. Finally, we design a real-world system to test our deformable SLAM and active SLAM algorithms.

6.1 Problem statement

We consider the active SLAM problem in 3D deformable environments. In our proposed method, the RIEKF SLAM algorithm is used to predict the uncertainty of the SLAM estimate after taking a potential control action in the local planning part, and it is also used to perform the SLAM estimation after a control action is taken in the SLAM process. The environment we consider is deformable and highly dynamic. Active SLAM in dynamic environments has also been investigated by many researchers, but it's different from deformable environments. The environments they considered are usually partially

dynamic, and the moving objects need to be distinguished and removed from the SLAM process. However, in deformable environments, there are no static parts, and the dynamic part needs to be taken into consideration. The robot considers the environment changes, aiming to reach the target region by itself successfully and map the latest environment accurately.

6.2 EKF SLAM in deformable environments

It is well known that the SLAM problem in deformable environments is not solvable unless some assumptions on the possible deformation and/or robot trajectory are made [17]. In our considered problem, the robot odometry model, observation model and feature dynamic models are formulated according to the available information considering the 3D case. In this section, we will introduce the EKF based 3D SLAM in deformable environments.

In the considered 3D feature based EKF SLAM problem, the state with M features at n-th step is

$$\mathbf{X}_n = (\mathbf{R}(\Theta_n), \mathbf{x}_n, \mathbf{f}_n^1, \cdots, \mathbf{f}_n^M), \tag{6.1}$$

where $\mathbf{R}(\Theta_n) \in \mathbb{SO}(3)$, $\mathbf{x}_n \in \mathbb{R}^3$ and $\mathbf{f}_n^j \in \mathbb{R}^3$ $(j = 1, \cdots, M)$ are respectively the robot orientation, robot position, and the coordinate of the j-th feature, all described in the fixed world coordinate frame. Note that the feature positions are also changing over time.

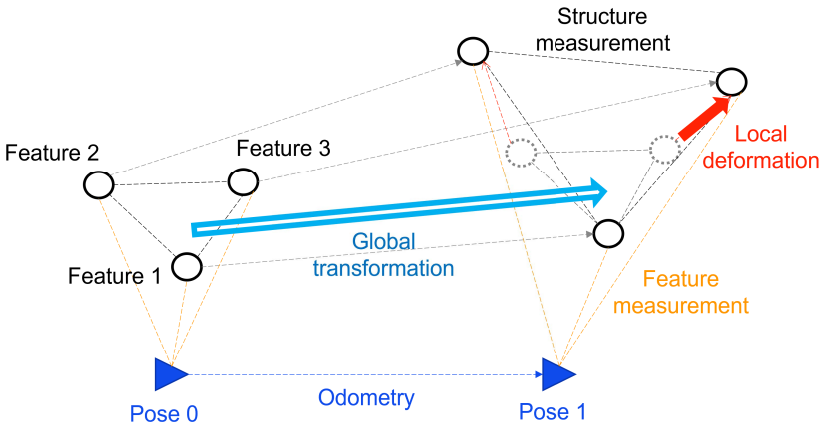


FIGURE 6.1: SLAM models in deformable environments.

The SLAM models in deformable environments are shown in Fig. 6.1. The robot motion model is the same as in SLAM in static environments. In general, for the SLAM problem in deformable environments, some knowledge of both the global rigid motion and the local deformation of the features is available. Therefore, we can assume that the movement of the features includes the global transformation $\mathbf{t}$ and the local deformation $\mathbf{d}$, and $\mathbf{r}_t$ and $\mathbf{r}_d$ are the corresponding rotation matrices.

Thus, the process model of the SLAM problem is

$$\mathbf{X}_{n+1} = \boldsymbol{f}(\mathbf{X}_n, \mathbf{u}_n, \mathbf{w}_n) = \begin{bmatrix} \mathbf{R}(\Theta_n + \boldsymbol{\omega}_n + \mathbf{w}_n^{\omega}) \\ \mathbf{x}_n + \mathbf{R}(\Theta_n)\mathbf{v}_n + \mathbf{w}_n^{v} \\ \mathbf{f}_n^1 + \mathbf{r}_t\mathbf{t}_n + \mathbf{r}_d\mathbf{d}_n^1 + \mathbf{w}_n^1 \\ \vdots \\ \mathbf{f}_n^M + \mathbf{r}_t\mathbf{t}_n + \mathbf{r}_d\mathbf{d}_n^M + \mathbf{w}_n^M \end{bmatrix}, \tag{6.2}$$

where $\mathbf{X}_n \sim \mathcal{N}(\hat{\mathbf{X}}_n, \mathbf{P}_n)$, $\mathbf{u}_n = [\boldsymbol{\omega}_n, \mathbf{v}_n]$ is the control input, and $\mathbf{w}_n = \left[\mathbf{w}_n^{\omega}, \mathbf{w}_n^{v}, \mathbf{w}_n^{j=1:M}\right] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Phi}_n)$ is the noise.

Note that the feature movement noise $\mathbf{w}_n^j$ contains two parts, the global transformation noise $\mathbf{w}_{n,j}^t$ and the local deformation noise $\mathbf{w}_{n,j}^d$. That is, $\mathbf{w}_n^j = \mathbf{w}_{n,j}^t + \mathbf{w}_{n,j}^d$.

The observation model also contains two parts, the measurement of the features and the measurement of the structure.

$$\mathbf{Z}_n = [\mathbf{z}_n^f; \mathbf{z}_n^c] = h_n(\mathbf{X}_n, \boldsymbol{\xi}_n). \tag{6.3}$$

Feature measurement $\mathbf{z}_{n,j}^f$ is the observations from the robot sensor to the j-th feature, just as in static environment.

Structure measurement $\mathbf{z}_{n,j_1,j_2}^c$ are the observations of the constraints between each pair of features, $\mathbf{f}^{j_1=1:M}$ and $\mathbf{f}^{j_2=1:M}$. In our experiment, the structure measurement between feature $\mathbf{f}^{j_1}$ and $\mathbf{f}^{j_2}$ is set to be the relative position between them:

$$\mathbf{z}_{n,j1,j2}^c = \mathbf{f}^{loc}(n, j_2) - \mathbf{f}^{loc}(n, j_1) + \boldsymbol{\xi}_n^c, \tag{6.4}$$

where $\mathbf{f}^{loc}(n,j)$ is the feature's position relative to $\mathbf{f}_n^1$ in the local coordinate, given by

$$\mathbf{f}^{loc}(n,j) = \mathbf{f}_n^j - \mathbf{f}_n^1. \tag{6.5}$$

The observation noise $\boldsymbol{\xi}_n = [\boldsymbol{\xi}_n^f, \boldsymbol{\xi}_n^c] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_n)$, where $\boldsymbol{\xi}_n^f$ and $\boldsymbol{\xi}_n^c$ correspond to the feature measurement noise and the structure measurement noise, respectively.

Similarly, the EKF based SLAM algorithm in deformable environments is also as Algorithm 1 shows.

6.3 Active SLAM in deformable environments

In this section, we first introduce the traditional planning method that minimizes a certain criterion to get the action in each step, which we call local planner here. Secondly, the global planner that is similar to [160] but for deformable environments is presented. Finally, we propose a combined planner that combines the local planning method and the global planning method.

6.3.1 The active SLAM problem

For the active SLAM problem considered in this thesis, given a deformable object of which the global transformation model and the local deformation model are known, we assume there are a known number of features distributed on the surface of the object, and some prior knowledge of the feature distribution is known. The robot starts from a fixed location in the environment. The objective is to plan the robot trajectory for a given time horizon so that it can observe the object completely and estimate the observed features and the robot poses accurately.

6.3.2 Local planner

The local planner is the greedy method, where the information that will be obtained in the next step is maximized to obtain the control action of the robot. The information gained

in terms of the SLAM estimate can be described by the resulting covariance matrix after the control action is taken and the information from the new observations is used.

Concretely, given $\hat{\mathbf{X}}_n$ and $\mathbf{P}_n$, we would like to select the control vector $\mathbf{u}_n$ such that a certain metric (e.g. the trace) of the covariance matrix in the next step is optimized. That is, the trace($\mathbf{P}_{n+1}$) is expected to be as small as possible:

$$obj = \min \text{trace}(\mathbf{P}_{n+1}), \tag{6.6}$$

where $\mathbf{P}_{n+1}$ is obtained by Algorithm 1.

Note that the feature observation $\mathbf{Z}_{n+1}$ is not available when the planning is performed, so we assume that no new feature will be observed. Whether an old feature can be observed at time $n+1$ is determined according to the predicted state estimate $\hat{\mathbf{X}}_{n+1|n}$ and the sensor range.

6.3.3 Global planner

In global planning, the objective is to generate a set of viewpoints so that the robot can observe each feature at least once by visiting the viewpoints. Here, a viewpoint $\mathbf{g}$ is a 3D position. The reward of each viewpoint, r, is defined as the number of unobserved features that can be observed at $\mathbf{g}$. In our proposed method, the global planner is a coarse but efficient planner. To simplify the calculation process, we do not consider the orientation of the robot, so features that are within the distance of the sensor range are considered to be able to be observed. Note that the reward of each candidate viewpoint is updated according to the previously selected viewpoints. As the same feature can be observed from multiple viewpoints, once it is observed by the selected viewpoint, it needs to be removed from others' field-of-view, and thus the reward needs to be updated accordingly.

Algorithm 2 presents the process of viewpoint set generation, which is similar to the sampling process shown in [160]. The algorithm first generates a set of viewpoint candidates $\mathcal{G}_{cand}$ distributed uniformly in the 3D space around the object. Secondly, the rewards of all viewpoint candidates in $\mathcal{G}_{cand}$ are computed based on the initial robot pose and the

initially estimated feature positions. More concretely, the reward of each candidate viewpoint is the number of features that can be observed in the viewpoint minus the number of features that have been observed by the robot in the initial pose.

Algorithm 2 Viewpoint set generation algorithm

Require: traversable space $\mathcal{S}$

Ensure: viewpoint set $\mathcal{G}_{final}$

1: Generate a set of viewpoint candidates $\mathcal{G}_{cand}$ in $\mathcal{S}$
2: For each candidate $\mathbf{g}_i$, calculate the number of the unobserved features that can be observed at $\mathbf{g}_i$, and set it to be its reward r_i
3: $c_{best} = \infty$
4: **for** $i = 1 : K$ **do**
5: $\quad \mathcal{G} = \emptyset$
6: $\quad \mathcal{G}'_{cand} = \mathcal{G}_{cand}$
7: $\quad R = \sum_{i=1}^{length(\mathcal{G}'_{cand})} r_i$
8: $\quad$ **while** $R \neq 0$ **do**
9: $\quad\quad$ Probabilistically pick viewpoint g' from $\mathcal{G}'_{cand}$
10: $\quad\quad$ Remove g' from $\mathcal{G}'_{cand}$
11: $\quad\quad \mathcal{G} \leftarrow \mathcal{G} \cup g'$
12: $\quad\quad$ Update r_i for all viewpoints in $\mathcal{G}'_{cand}$
13: $\quad\quad R = \sum_{i=1}^{length(\mathcal{G}'_{cand})} r_i$
14: $\quad$ **end while**
15: $\quad$ Compute cost c using equation (6.7)
16: $\quad$ **if** $c < c_{best}$ **then**
17: $\quad\quad \mathcal{G}_{final} = \mathcal{G}$, $c_{best} = c$
18: $\quad$ **end if**
19: **end for**
20: **return** $\mathcal{G}_{final}$

Next, a process is iterated K times to determine the final viewpoint set $\mathcal{G}_{final}$ that contains the viewpoints to be visited in order. Here, K is the number of sample sets to be compared, which can be determined according to the experiment requirement. In each iteration, a

subset of viewpoints $\mathcal{G}$ is generated from $\mathcal{G}_{cand}$ according to their rewards. Concretely, the viewpoints are selected with probabilities proportional to their rewards and put in $\mathcal{G}$. After a viewpoint is selected, the rewards of the remaining viewpoints are reduced accordingly. Each iteration process finishes when the total reward of the remaining candidate viewpoints is zero.

After the sampling process, we obtain K sets of viewpoints. The one with the minimum cost function will be selected to be the final viewpoint set $\mathcal{G}_{final}$. The cost function is defined as

$$c = \sum_{i=1}^{I} d^{g}_{i,i+1}, \tag{6.7}$$

where $d^{g}_{i,i+1}$ is the Euclidean distance between two adjacent viewpoints $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$ in $\mathcal{G}$.

After the viewpoint set is determined, the viewpoint $\mathbf{g}_i$ is set to be the current goal point $\mathbf{g}_{cur}$ one by one from $i = 1 : N_g$ in the order of the set, where N_g is the number of viewpoints in the set. In each step, the robot selects the action that minimizes the distance d between the position of the robot $\mathbf{x}_{n+1|n}$ and the position of the current goal point $\mathbf{g}_{cur}$:

$$obj = \min d(\mathbf{x}_{n+1|n}, \mathbf{g}_{cur}), \tag{6.8}$$

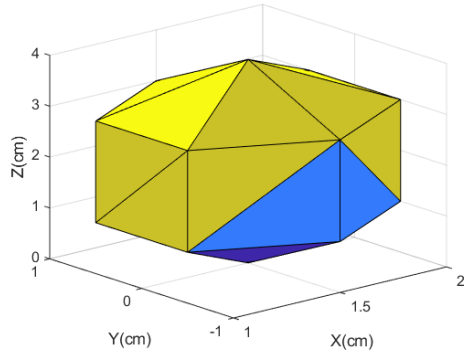
Once all the viewpoints are visited, the next viewpoint will be the first viewpoint of the set. That is, visiting these viewpoints in sequence and repeating in a loop.

6.3.4 Combined planner

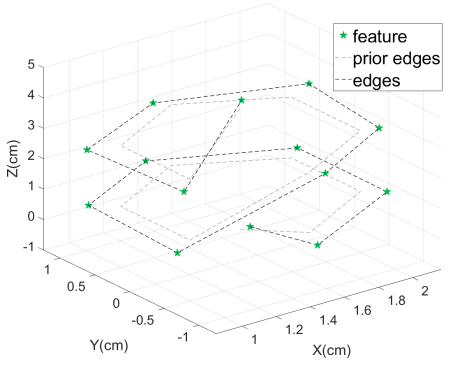
The combined planner is a combination of the local planner and the global planner, aiming to balance the performance in accuracy, coverage and processing time. We use the global planner to generate the viewpoint set to ensure that all features can be observed as soon as possible. At the same time, the local planner, which aims to minimize the uncertainty, will be used to improve the accuracy.

In particular, the robot action $\mathbf{u}_n$ is selected from a set of candidate actions $\mathbf{U} = \{\mathbf{u}^i_n | i = 1, 2, ..., N_u\}$ in each step, where N_u is the number of the candidate actions. The robot pose after taking each candidate action is predicted, and the distance d between the predicted

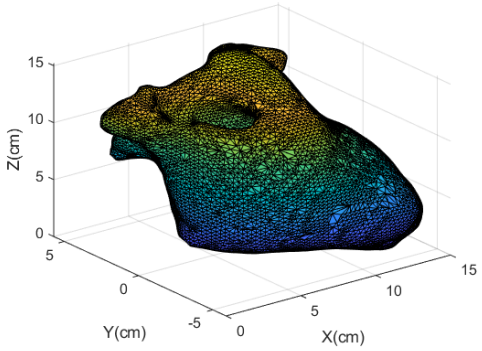
robot pose and the current goal point is calculated. The action set is sorted according to the distances from small to large, and then the first n_u candidate actions are selected to be the new action set. This process ensures that all actions in the new action set are likely to guide the robot to approach the goal point. The final action will be selected from the new action set using the local planner, which minimizes the trace of the covariance matrix, as shown in eq. (6.6). Here, n_u is the number of actions in the new action set, where a smaller value of n_u means a larger probability of the robot to approach the goal point. If $n_u = 1$, it becomes the global planner. On the contrary, if $n_u = N_u$, it is the same as the local planner.



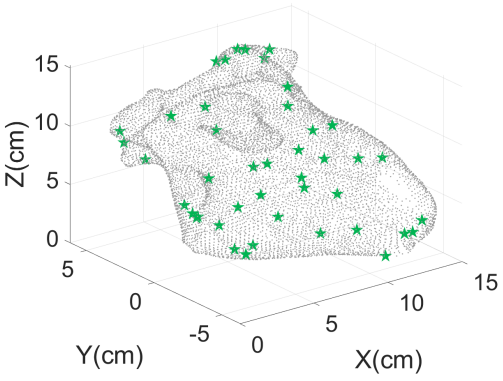
(A) Mesh model of the polygon environment



(B) Feature model of the polygon environment

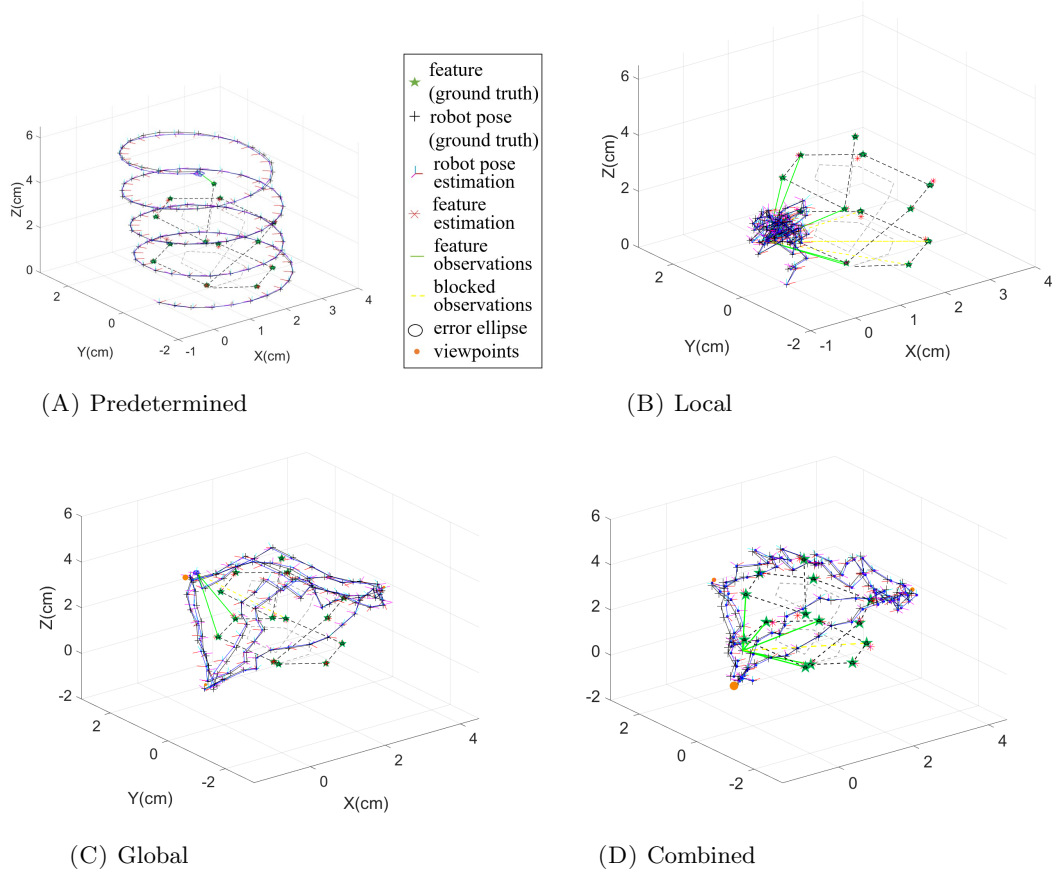


(C) Mesh model of the heart environment



(D) Feature model of the heart environment

FIGURE 6.2: Environment models.



(A) Predetermined

(B) Local

(C) Global

(D) Combined

FIGURE 6.3: Result of using different active SLAM methods in the polygon environment.

6.4 Models and simulation settings

The process model and feature movement models for the active SLAM are described in Section 6.2. As stated above, the movement of the features includes the global transformation $\mathbf{t}$ and the local deformation $\mathbf{d}$. Here, we consider the linear case only, that is, $\mathbf{r}_t = \mathbf{I}_3$ and $\mathbf{r}_d = \mathbf{I}_3$.

For the feature measurement model, the j-th feature observed at the n-th step is given by

$$\mathbf{z}_{n,j}^f = \mathbf{R}(\Theta_n)(\mathbf{x}_n - \mathbf{f}_n^j) + \boldsymbol{\xi}_n^f. \tag{6.9}$$

Two environments with different target deformable objects are used to validate the algorithms in simulation, where the target object is what we want to estimate. One is a

polygon environment that contains a created simple object model, as shown in Fig. 6.2A and 6.2B. The other is a heart environment that contains a heart model segmented from a CT scan, downloaded from OpenHELP [161], as shown in Fig. 6.2C and 6.2D. Our objective is to observe the target object and obtain an accurate SLAM estimate.

For the polygon model, we set 14 point features to form a prismatic impenetrable object, as shown in Fig. 6.2B. Globally, the object moves back and forth along the x-axis regularly by small degrees; that is, all features move forward 0.1 cm and then come back in the next step. Locally, the deformation is a regular expansion and shrinkage. In our simulation, we set the first feature to be the anchor point. All the other features move away from the anchor point 0.2 cm in each time step for 3 steps and then move back 3 steps in the same distance. The total step of the robot motion is set to be 100. The initial robot pose is $[0, 0, 0, 0, 0, 0]$, and the sensor range is 3 cm. The covariance matrix of control noise $[\mathbf{w}_n^{\omega}, \mathbf{w}_n^{v}]$ is $\text{diag}[(0.02\text{rad})^2, (0.02\text{rad})^2, (0.02\text{rad})^2, (0.03\text{cm})^2, (0.03\text{cm})^2, (0.03\text{cm})^2]$. The covariance matrix of feature movement noise $\mathbf{w}_n^j = \mathbf{w}_{n,j}^t + \mathbf{w}_{n,j}^d$ is $\text{diag}[(0.03\text{cm})^2 + (0.05\text{cm})^2, (0.03\text{cm})^2 + (0.05\text{cm})^2, (0.03\text{cm})^2 + (0.05\text{cm})^2]$. And the covariance matrix of observation noise $\boldsymbol{\xi}_n^j$ also contains two parts. For the feature measurement noise $\boldsymbol{\xi}_n^f$, the corresponding covariance matrix is $\text{diag}[(0.02\text{cm})^2, (0.02\text{cm})^2, (0.02\text{cm})^2]$. For the structure measurement noise $\boldsymbol{\xi}_n^c$, the corresponding covariance matrix is $\text{diag}[(0.5\text{cm})^2, (0.5\text{cm})^2, (0.5\text{cm})^2]$. Note that the deformable objects are impenetrable, so in the planning algorithms, we restrict the robot to keep a safe distance of 0.3 cm from the object's surface.

The heart model was segmented from a CT scan of a healthy, young male undergoing shock room diagnostics. There are thousands of vertex that form the mesh model, as shown in Fig. 6.2C. In the experiment, we randomly picked fifty of them to be the features that are used in the decision making process and the SLAM estimation process, as the green stars show in Fig. 6.2D. The models and parameters used in the heart model are the same as those in the polygon model, except that the sensor range is set to be 10 cm, and the safe distance between the robot and the object surface is 3 cm. Note that the global transformation and local deformation are applied to all vertex of the model, but we just consider the estimate of the selected features. The total step of the robot is set to be 200.

6.5 Simulation results

The coverage is compared by counting the total number of the observed feature times in the whole time horizon, and the time steps that all features have been observed at least once. The accuracy is measured by calculating the maximum/average error between the estimated values and the ground truth, including the robot pose error (position error and orientation error) and the feature position error. We record the processing time of the decision making part to compare the computational cost of determining the next control. For each algorithm, we perform the simulation 5 times and label them as ***No.*** 1-5. The corresponding figures shown in this section are the representative ones.

6.5.1 Polygon environment

We first present the SLAM results in the polygon environment. The ground truth of the robot trajectory and the features and the results based on different methods (including the estimated poses and the estimated features, and the covariance ellipse of the features) are shown in Fig. 6.3.

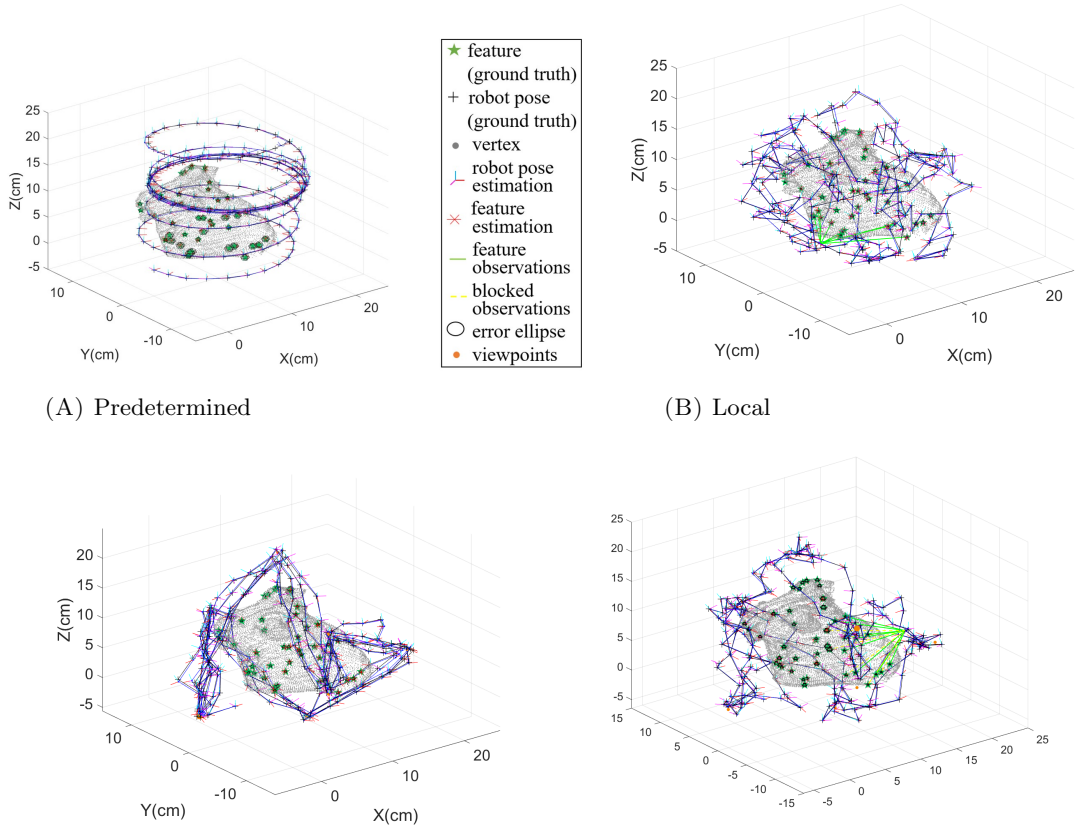
TABLE 6.1: Coverage comparison for the polygon environment

	No.	***Predetermined***	***Local***	***Global***	***Combined***
Observed feature times [1]	1	354	681	289	583
	2	393	687	257	515
	3	359	750	266	505
	4	338	655	324	586
	5	292	702	253	518
Steps [2]	1	35	– [3]	19	24
	2	24	–	26	24
	3	24	–	22	29
	4	25	–	22	24
	5	24	–	19	22

[1] The total number of feature times that are observed in the whole time horizon. If 3 features are observed 2 times each, then the feature times is 6.
[2] The time steps that all features have been observed at least once.
[3] '–' means the robot does not observe all the features within the time horizon.

6.5.1.1 Coverage

In this part, we show the coverage performance of the different methods. The performance of the coverage task is evaluated by counting the total number of the observed feature times in the whole time horizon and the time steps needed to observe all the features at least once, as Table 6.1 shows. Four methods are performed, including using a predetermined path, the global planner, the local planner and using the proposed combined planner.



(A) Predetermined (B) Local

(C) Global (D) Combined

FIGURE 6.4: Results of using different active SLAM methods in the heart environment.

When the predetermined path is used, although all features can be observed at least once eventually, it takes at most 35 steps. What's more, as the path is predetermined without considering the movement of the object, it has the risk of collision. Although this can be solved by setting the path radius larger, the premise is that there is enough free space around the object, which is impractical in many cases. However, this can be easily solved by using active SLAM algorithms by restricting the motion of the robot.

The number of observed features by using the local planner is the largest. However, it cannot observe all features within the time horizon. This is because the local planner only considers the information gain to improve the accuracy, causing the robot to continuously revisit the previously observed features to reduce the estimation error, as shown in Fig. 6.3B.

The global planner costs the least steps (i.e., 19 steps) to observe all features at least once in this experiment. However, the total number of the observed feature times in the whole process is the least. The reason is that when we select the viewpoints, we consider the largest number of features that can be observed at that viewpoint. In the planning process, we only consider to minimize the distance between the robot and the goal point. It is possible that the total number of the observed feature times is small, because the robot may not observe all features that are expected to be observed at the selected viewpoints, especially when the environment is deforming. For the same reason, it is possible that the steps needed for observing all features are large.

For the proposed combined planner, its performance is as expected. More features can be observed during the whole process compared with the global planner and using a predetermined path. The number of steps used to observe all features is much smaller when compared with the predetermined path and the local planner.

6.5.1.2 Accuracy

In this part, we compare the accuracy performance of the obtained active SLAM results using different methods. The results of the pose error based on different methods in a single run are shown in Fig. 6.5A.

Here, the robot position error and robot orientation error at the n-th step are calculated by:

$$e_\theta = \sqrt{(\Theta_n - \hat{\Theta}_n)^{\mathrm{T}}(\Theta_n - \hat{\Theta}_n)},\ e_x = \sqrt{(\mathbf{x}_n - \hat{\mathbf{x}}_n)^{\mathrm{T}}(\mathbf{x}_n - \hat{\mathbf{x}}_n)} \tag{6.10}$$

where $\Theta_n - \hat{\Theta}_n$ is the error between the orientation ground truth and the estimated orientation, and $\mathbf{x}_n - \hat{\mathbf{x}}_n$ is the error between the position ground truth and the estimated position.

The combined planner and the local planner perform almost the same in terms of the pose error. The global planning algorithm, using the predetermined path, obtains relatively larger errors. The average robot pose error and maximum robot pose error of all runs are shown in Table 6.2. It suggests that the combined planner can obtain much smaller errors than the global planner or using a predetermined path. Compared with the local planner, the combined planner is roughly the same.

TABLE 6.2: Estimation error comparison for the polygon environment

	Active SLAM			
	Predetermined	***Local***	***Global***	***Combined***
Max error position (cm)	0.7433	0.240	0.2750	0.2258
Ave error position (cm)	0.1878	0.0810	0.1219	0.0925
Max error orientation (rad)	0.1153	0.0983	0.1956	0.1283
Ave error orientation (rad)	0.0562	0.0427	0.0985	0.0596
Max error feature (cm)	0.3316	0.3477	0.3434	0.2622
Ave error feature (cm)	0.1799	0.1199	0.1421	0.1109

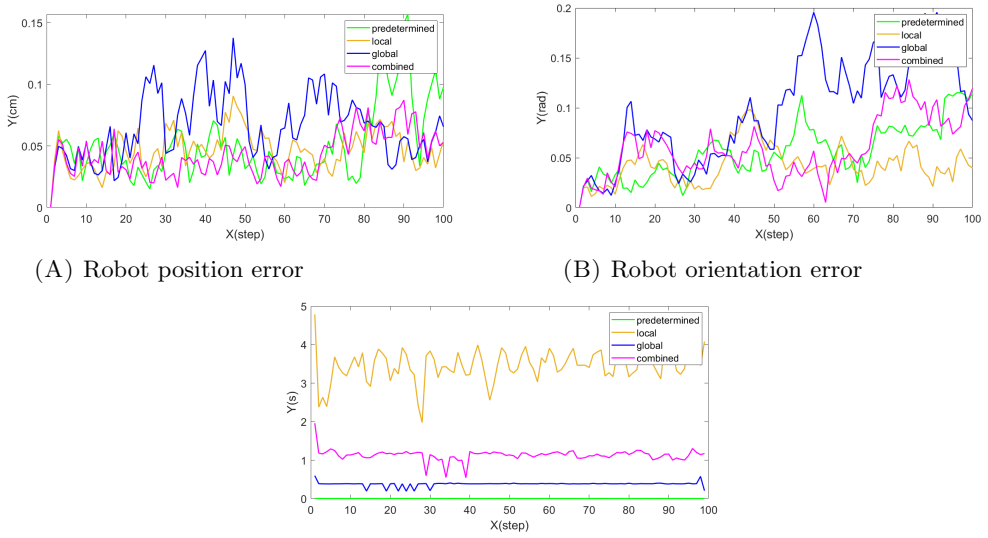
Besides the robot pose error, Table 6.2 also shows the maximum and average feature estimation errors in the last step. The error of the j-th feature can be obtained by calculating the Euclidean distance between the feature position ground truth and the estimated feature position.

$$e_p = \sqrt{(\mathbf{f}^j - \hat{\mathbf{f}}^j)^{\mathrm{T}}(\mathbf{f}^j - \hat{\mathbf{f}}^j)}, \tag{6.11}$$

Obviously, the combined planner has advantages over the local planner. Compared with the global planner or using a predetermined path, the combined one can get much smaller maximum errors and average errors.

6.5.1.3 Processing time

For efficiency, we compare the processing time in the decision making part, that is, the time used to calculate the next action of the robot. Fig. 6.5C shows the results of a single run. We can see that using a predetermined path costs the least time, since there is no decision to make. The combined method costs about 2.5 times longer than the global planning method. The local planner costs much longer time than the others. What's more,



(A) Robot position error

(B) Robot orientation error

(C) Decision making time

FIGURE 6.5: Comparison of using different active SLAM methods in the polygon environment.

the decision making time cost by the local planner is related to the number of features, because we need to predict the visibility of all features to calculate the covariance matrix.

6.5.2 Heart environment

Further tests were run for the heart environment. The same four strategies are compared, and the results are shown in Fig. 6.4.

TABLE 6.3: Coverage comparison for the heart environment

	No.	***Predetermined***	***Local***	***Global***	***Combined***
Observed feature times	1	1358	2042	864	1619
	2	1181	2098	1233	1597
	3	1579	2134	1089	1515
	4	1216	2019	1097	1594
	5	844	2108	922	1664
Steps	1	66	31	72	25
	2	71	62	63	35
	3	70	51	48	35
	4	65	75	47	35
	5	65	74	30	30

6.5.2.1 Coverage

The coverage performance in terms of the observed feature times and the steps to observe all features are shown in Table 6.3. We can see the great advantage of using the combined planner over using a predetermined path and the global method. The combined planner uses much fewer steps to observe all features, and during the whole time horizon, the observed feature times are much more than those two. Although the local planner has more observed feature times in the whole time horizon, it costs more steps to observe all features at least once.

TABLE 6.4: Estimation error comparison for the heart environment

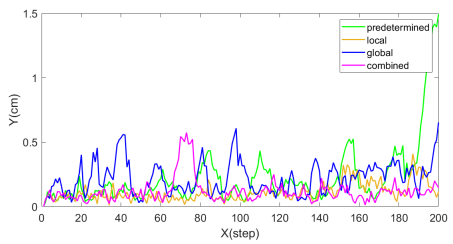
	Predetermined	***Local***	***Global***	***Combined***
Max error position (cm)	11.0275	0.4094	0.6529	0.5740
Ave error position (cm)	0.6714	0.0998	0.1582	0.1203
Max error orientation (rad)	0.0825	0.0432	0.0747	0.0694
Ave error orientation (rad)	0.0215	0.0178	0.0233	0.0206
Max error feature (cm)	0.4554	0.3384	0.2892	0.1975
Ave error feature (cm)	0.1670	0.1025	0.1186	0.1047

6.5.2.2 Accuracy

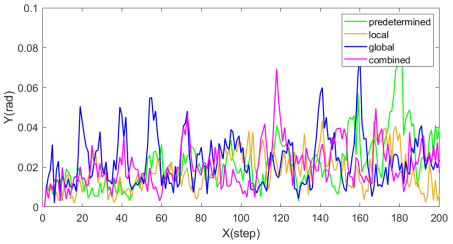
The robot pose errors based on different methods in a single run are shown in Fig. 6.6A. It is obvious that the combined method gets relatively smaller errors than using a predetermined path and the global method. The detailed values about the maximum and average errors of 5 runs are shown in Table 6.4. It also suggests that the combined planner can get much more accurate results compared with those two methods. The errors obtained by the local planner and the combined planner are close, and much smaller than those obtained by the other two methods.

6.5.2.3 Processing time

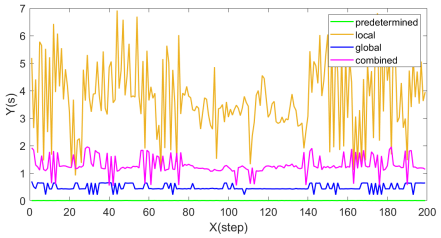
The performance in processing time is as expected. The result of a single run is shown in Fig. 6.6C. The combined planner costs about 2 times longer than the global planner.



(A) Robot position error



(B) Robot orientation error



(C) Decision making time

FIGURE 6.6: Comparison of using different active SLAM methods in the heart environment.

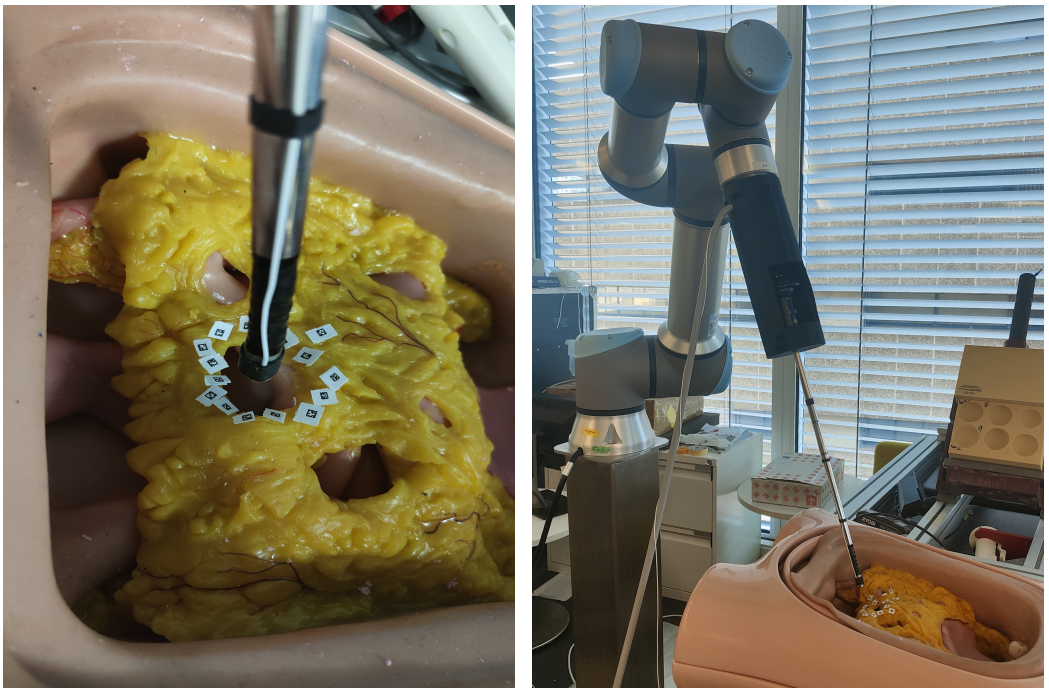
6.6 Real-world experiments

In this section, we design an active SLAM system in a real-world phantom environment. This system uses a UR robot to handle an endoscope, which can move the camera to the expected poses that are calculated by our active SLAM algorithm. We first keep the environment static and test the SLAM and active SLAM algorithms. Both EKF based methods and RIEKF based methods are tested. After that, we make the environment deformable and analyze the motion model of the features. We design an EKF based active SLAM method and test it in the deformable environment.

6.6.1 Experiment settings

In the experiment, we use Olympus ENDOEYE FLEX 3D to detect the AprilTags inside a phantom. As we focus on the active SLAM algorithms rather than the feature recognition, we skip the feature extraction step. Instead, we put some AprilTags on the manikin surface to be the features. We use the first 12 tags in the tag family "TAG36H11". These AprilTags can be detected by the endoscope cameras. Here, we use the images from the left camera to recognize the tags and calculate the poses of these tags. We use a UR16e robot to

control the endoscope to move to the desired poses. A 3D printed handler is used to link the UR robot and the endoscope. The phantom environment and the robot system are shown in Fig. 6.7A and Fig. 6.7B.



(A) Environment inside the phantom

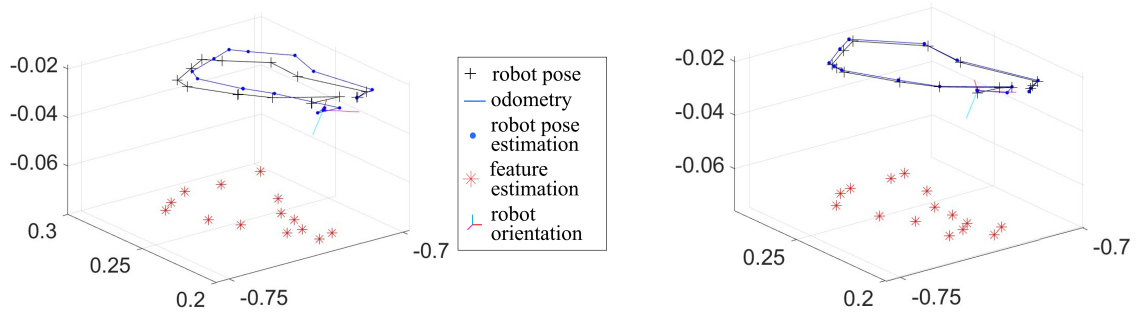
(B) Endoscope handled by the UR16e robot

FIGURE 6.7: Real-world phantom experiment settings.

In this system, we first keep the environment static, testing the EKF and RIEKF SLAM algorithms with a predetermined path, and the EKF and RIEKF based active SLAM algorithms where the camera pose at each step is determined by the greedy method. Then we make the environment deformable. With the motion model of the features, we test the EKF based SLAM and active SLAM algorithms in this system.

6.6.2 SLAM and active SLAM in the static environment

For the SLAM algorithms, the camera pose at each step is predetermined. A serials of control command are given to the UR robot, so that it can move the endoscope camera to the specific poses. The SLAM results of using EKF and RIEKF are shown in Fig. 6.8A and Fig. 6.8B, respectively. Obviously, the RIEKF based SLAM algorithm can get much more accurate results than the EKF based one.



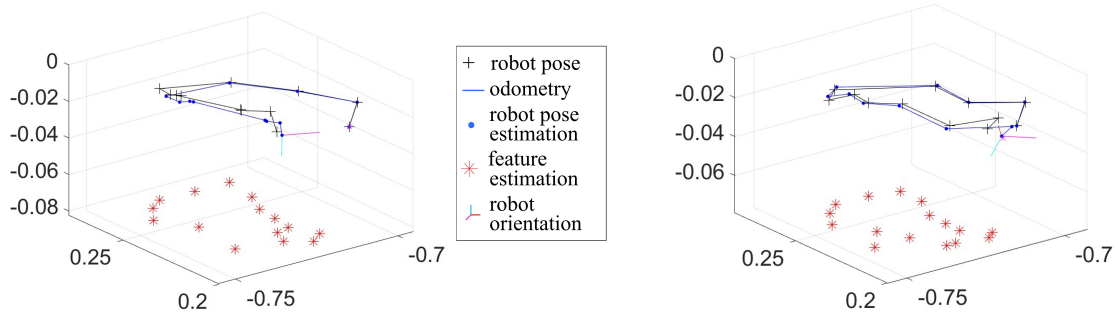
(A) EKF result algorithm

(B) RIEKF result algorithm

FIGURE 6.8: SLAM results of using EKF and RIEKF algorithms in the static phantom environment.

The active SLAM algorithm is to use a greedy method to improve the predetermined path. Concretely, given a predetermined pose at step i, noted as p_i, we aim to find a better pose around p_i, so that a more accurate SLAM estimate can be obtained. At each step, given a predetermined pose p_i, we sample a serial of available poses centered at p_i. We predict the features that can be detected at each candidate pose, and further predict the corresponding SLAM result. The one with the smallest uncertainty, that is, the smallest trace of the covariance matrix, will be selected to be the next camera pose. Once the next camera pose is determined, the UR robot can control the endoscope to the desired pose. The pose of the end effector can be calculated from the camera pose by frame transformation. The translation matrix between the end effector and the camera can be obtained by doing hand-eye calibration. We use MoveIt to control the end effector to move to a specific pose. MoveIt is a toolbox that incorporates the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. Here, we send the calculated goal pose to it; then it can control the end effector to move to that pose.

The results are shown in Fig. 6.9. No clear difference is observed between EKF and RIEKF based active SLAM. This might be due to the small environment used.



(A) EKF based active algorithm

(B) RIEKF based active algorithm

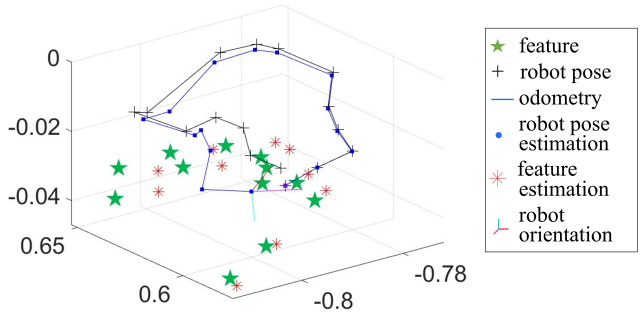
FIGURE 6.9: Active SLAM results of using EKF and RIEKF algorithms in the static phantom environment.

6.6.3 SLAM and active SLAM in the deformable environment

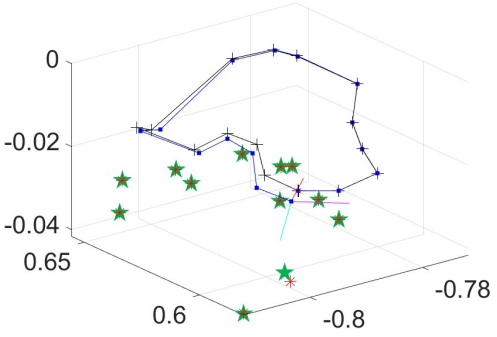
To make the phantom deformable, we use a pump to inflate and deflate the model regularly. We use an RGB camera to record the motion of the features several times and analyze their motion model. The motion of each tag is described under the frame of $Tag0$ at time 0, which we denote as f_0. $Tag0$ is the tag whose tag id is 0. As the motion of the environment is regular, we divided the motion period into several phases. For each phase, we analyze the position of each feature under f_0 and store the data accordingly. In the experiment, as long as anyone tag is observed and f_0 is known, we can derive the current phase of the feature motion, and further infer the current positions of each feature. We can also predict the next positions of the features using the feature motion model, so that we can better determine the next robot motion.

Similarly, we test the EKF based deformable SLAM algorithm in the deformable environment, and compare it with the traditional EKF based SLAM, where the features are assumed to be all stationary. A predetermined path of the robot is given to test the algorithms. For the active SLAM algorithm, we also use the greedy method to improve the predetermined path, as we do in the static phantom environment. The results are as Fig. 6.10 shows.

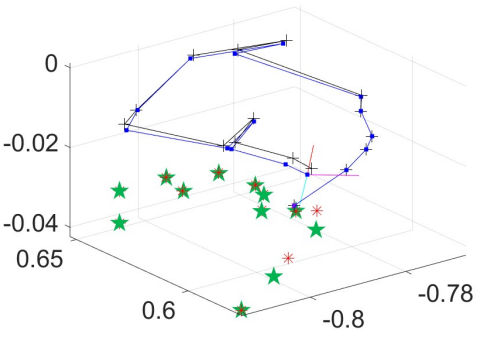
We compare the number of features that are observed within the given time horizon. As Table 6.5 suggests, by using the predetermined path, the robot leaves two features unobserved. By considering the feature motion model in the decision making process, the



(A) Traditional EKF based SLAM



(B) EKF based deformable SLAM



(C) EKF based active deformable SLAM

FIGURE 6.10: Results of using different algorithms in the deformable phantom environment.

active SLAM algorithm successfully observes all features within the given time horizon. We also calculate the average robot pose errors of using each algorithm. We can see in Table 6.5 that the deformable SLAM algorithm where the feature motion model is considered can get smaller estimation errors than the traditional EKF SLAM. By using the active SLAM algorithm, we can get the most accurate SLAM estimates.

TABLE 6.5: Comparison of feature number and robot pose error in real-world deformable environment

	EKF	***Deformable EKF***	***Active***
Number of observed features	10	10	12
Average pose error (cm)	0.2888	0.0925	0.0582

6.7 Summary

In this chapter, we proposed an active SLAM algorithm for 3D deformable environments. Because of the combination of the efficient global planner and the accurate local planner, the proposed algorithm shows better performance in accuracy as compared with the global planner. It is also demonstrated that the proposed algorithm has satisfactory performance in accuracy and much lower computational cost as compared with the local planner.

We also developed a real-world active SLAM system. We successfully test the EKF and RIEKF based SLAM and active SLAM algorithms in a static manikin environment. The RIEKF based algorithms are shown to be able to get more accurate SLAM results than the EKF based ones in the real-world static environment. After that, we make the phantom deformable and get an approximate motion model of the features. With the motion model, we test the EKF based deformable SLAM algorithm in the deformable environment and get better SLAM results than the EKF SLAM algorithm. We also test the active SLAM algorithm and get some good results.

Currently, we haven't applied RIEKF in deformable environments. As the use of RIEKF is to solve the inconsistency problem in EKF SLAM, firstly, we need to prove that the inconsistency problem also exists in the EKF based deformable SLAM. However, no one has proven the inconsistency of EKF based deformable SLAM yet. If the inconsistency does exist, we also need to prove that the RIEKF algorithm can solve the inconsistency problem. If RIEKF is theoretically proven to be able to work better in deformable environments than standard EKF, we will carry out experiments to test the SLAM results, and then design active SLAM algorithms accordingly.

Chapter 7

Conclusion and Future Work

This chapter briefly reviews our contributions, identifies limitations and explores future research plans.

7.1 Contributions

This thesis has developed three active SLAM algorithms as follows.

i. We applied RIEKF to the active SLAM algorithm, developing an efficient and accurate planning framework for the robot to explore unknown 2D environments.

ii. We extend our RIEKF based 2D active SLAM algorithm to 3D cases, and improve the planning framework to solve the exploration-exploitation dilemma.

iii. Taking the environment changes into consideration, we developed an EKF based active SLAM framework for 3D deformable environments.

In Chapter 4, we proposed to use RIEKF SLAM algorithm in 2D active SLAM where both the predicted SLAM results for choosing control actions and the actual estimated SLAM results applying the selected control actions are computed using RIEKF algorithms. This is the first research work in this direction that considers the 2D cases and applies the greedy methods in the planning. The advantages over traditional EKF based active SLAM

are the more accurate and consistent predicted uncertainty estimates, which result in the robustness of the active SLAM algorithm. The advantage over optimization based active SLAM is the reduced computational cost. Simulation results are presented to validate the advantages of the proposed algorithm.

In Chapter 5, we consider using RIEKF method in active SLAM in 3D environments with exploration tasks. Similar to the 2D cases, both the predicted SLAM results for candidate control actions and the actual estimated SLAM results after applying the selected control actions are computed using RIEKF algorithms. Different from the previous work that used a weighted objective function to determine the robot's motion in 2D, the newly designed planner is a combination of an efficient global planner and an accurate local planner in 3D. Results of the simulation and the real-world experiment demonstrate that our proposed method improves the estimation accuracy significantly and, at the same time, maintains a high exploration efficiency.

In Chapter 6, we consider the active SLAM problem in 3D deformable environments. An EKF based active SLAM framework is designed to estimate the SLAM result accurately and efficiently. The planner proposed in this work is a combination of a global planner and a local planner. We compare the combined framework with using the local greedy method only and using the global planner only. Simulation results under different scenarios have shown that the proposed active SLAM algorithm provides a good balance between accuracy and efficiency as compared to the local planner and the global planner. Besides the simulation experiments, we also design an active SLAM system in a real-world manikin environment. This system uses a UR robot to handle an endoscope, which can move the camera to the expected poses that are calculated by our active SLAM algorithm. This system has been used to test the EKF based SLAM and active SLAM algorithms, and the RIEKF based SLAM and active SLAM algorithms. We have got good and stable SLAM and active SLAM results when the environment is static.

Throughout this thesis, the experimental evaluations of our proposed active SLAM algorithms show good performance in the environments we considered.

7.2 Limitations and future work

The research topics covered in this thesis are long-standing robotic problems, and our solutions are far from being completed. Here, we present some limitations of this thesis and provide a few possible future research directions.

Firstly, in our active SLAM algorithms, we mainly consider the greedy method when determining the robot pose of the next step. Although greedy methods are computationally efficient, their global optimality is not guaranteed. Some other planning strategies, such as multiple steps look ahead, will be investigated and evaluated. Future research work will also include other active SLAM problems where not only an exploration task is needed.

Secondly, the environments used for testing our algorithms are relatively simple. Our work is based on a point feature map, which assumes the features are known to us. However, in reality, how to extract and recognize features from various environments is an important and challenging task. In the future, we will investigate more practical non-feature-based active SLAM problems.

Finally, we will consider to investigate the use of RIEKF in deformable SLAM. As the use of RIEKF is to solve the inconsistency problem in EKF SLAM, firstly, we need to prove that the inconsistency problem also exists in the EKF based deformable SLAM. If it exists, the next task is to find whether there is a specific Lie group structure to set up RIEKF SLAM in deformable environments, and whether the RIEKF algorithm can solve the inconsistency problem. Finally, experiments will be carried out to test the results, and the active SLAM algorithm will be designed accordingly.

Bibliography

[1] Alexei Makarenko, Stefan Williams, Frédéric Bourgault, and Hugh Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 534–539 vol.1, 2002. doi: 10.1109/IRDS.2002.1041445.

[2] Hans Jacob S. Feder, John J. Leonard, and Christopher M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18: 650 – 668, 1999.

[3] Yongbo Chen, Liang Zhao, Ki Myung Brian Lee, Chanyeol Yoo, Shoudong Huang, and Robert Fitch. Broadcast your weaknesses: Cooperative active pose-graph slam for multiple robots. *IEEE Robotics and Automation Letters*, 5(2):2200–2207, 2020. doi: 10.1109/LRA.2020.2970665.

[4] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José 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.

[5] Shoudong Huang, N.M. Kwok, G. Dissanayake, Q.P. Ha, and Gu Fang. Multi-step look-ahead trajectory planning in slam: Possibility and necessity. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1091–1096, 2005. doi: 10.1109/ROBOT.2005.1570261.

[6] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning under uncertainty in the continuous domain: A generalized belief space approach. In *2014 IEEE*

International Conference on Robotics and Automation (ICRA), pages 6763–6770, 2014. doi: 10.1109/ICRA.2014.6907858.

[7] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. doi: 10.1109/CVPR.2015.7298631.

[8] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and System (RSS)*, 06 2018. doi: 10.15607/RSS.2018.XIV.029.

[9] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4), November 2008. ISSN 0730-0301. doi: 10.1145/1409625.1409627.

[10] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 362–379, Cham, 2016. Springer International Publishing.

[11] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Fanello, Adarsh Kowdle, Sergio Orts, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics*, 35, 07 2016. doi: 10.1145/2897824.2925969.

[12] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgbd sensor. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–501, 2015. doi: 10.1109/CVPR.2015.7298647.

[13] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Dynamic reconstruction of deformable soft-tissue with stereo scope in minimal invasive surgery. *IEEE Robotics and Automation Letters*, 3(1):155–162, 2018. doi: 10.1109/LRA.2017.2735487.

[14] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing. *IEEE Robotics and Automation Letters*, 3(4):4068–4075, 2018. doi: 10.1109/LRA.2018.2856519.

[15] Jose Lamarca, Shaifali Parashar, Adrien Bartoli, and J. M. M. Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *IEEE Transactions on Robotics*, PP:1–13, 09 2020. doi: 10.1109/TRO.2020.3020739.

[16] Antonio Agudo. Total estimation from rgb video: On-line camera self-calibration, non-rigid shape and motion. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8140–8147, 2021. doi: 10.1109/ICPR48806.2021.9412923.

[17] Shoudong Huang, Yongbo Chen, Liang Zhao, Yanhao Zhang, and Mengya Xu. Some research questions for slam in deformable environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7653–7660, 2021. doi: 10.1109/IROS51168.2021.9635883.

[18] Shoudong Huang and Gamini Dissanayake. Convergence and consistency analysis for extended kalman filter based slam. *IEEE Transactions on Robotics*, 23(5):1036–1049, 2007. doi: 10.1109/TRO.2007.903811.

[19] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 473–479, 2008. doi: 10.1109/ROBOT.2008.4543252.

[20] Jose Castellanos, Ruben Martinez-Cantin, Juan Tardos, and José Neira. Robocentric map joining: Improving the consistency of ekf-slam. *Robotics and Autonomous Systems*, 55:21–29, 01 2007. doi: 10.1016/j.robot.2006.06.005.

[21] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. Observability-based rules for designing consistent ekf slam estimators. *Int. J. Rob. Res.*, 29(5):502–528, April 2010. ISSN 0278-3649. doi: 10.1177/0278364909353640.

[22] Silvere Bonnabel. Symmetries in observer design: review of some recent results and applications to ekf-based slam, 2011. URL `https://arxiv.org/abs/1105.2254`.

[23] Axel Barrau and Silvere Bonnabel. An ekf-slam algorithm with consistency properties, 2016. URL https://arxiv.org/abs/1510.06263.

[24] Axel Barrau and Silvère Bonnabel. The invariant extended kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, 62(4):1797–1812, 2017. doi: 10.1109/TAC.2016.2594085.

[25] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. Isam2: Incremental smoothing and mapping using the bayes tree. *Int. J. Rob. Res.*, 31(2):216–235, February 2012. ISSN 0278-3649. doi: 10.1177/0278364911430419.

[26] Martin Brossard, Axel Barrau, and Silvère Bonnabel. Exploiting symmetries to design ekfs with consistency properties for navigation and slam. *IEEE Sensors Journal*, 19(4):1572–1579, 2019. doi: 10.1109/JSEN.2018.2882714.

[27] Yanhao Zhang, Teng Zhang, and Shoudong Huang. Comparison of ekf based slam and optimization based slam algorithms. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1308–1313, 2018. doi: 10.1109/ICIEA.2018.8397911.

[28] Mengya Xu, Yang Song, Yongbo Chen, Shoudong Huang, and Qi Hao. Invariant ekf based 2d active slam with exploration task. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5350–5356, 2021. doi: 10.1109/ICRA48506.2021.9561951.

[29] Mengya Xu, Liang Zhao, Shoudong Huang, and Qi Hao. Active slam in 3d deformable environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7952–7958, 2022. doi: 10.1109/IROS47612.2022.9982224.

[30] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[31] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Auton. Robots*, 5(3–4):253–271, jul 1998. ISSN 0929-5593. doi: 10.1023/A:1008806205438.

[32] Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001. doi: 10.1109/70.938381.

[33] David Charles Lee. *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Mobile Robot: An Experimental, Quantitative Evaluation.* Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

[34] Christopher I. Connolly. The determination of next best views. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 432–435, 1985. doi: 10.1109/ROBOT.1985.1087372.

[35] Jasna Maver and Ruzena Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):417–433, 1993. doi: 10.1109/34.211463.

[36] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997. doi: 10.1109/CIRA.1997.613851.

[37] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (10):1016–1030, 1999. doi: 10.1109/34.799908.

[38] Dimitrios Gallos and Frank Ferrie. Active vision in the era of convolutional neural networks. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 81–88, 2019. doi: 10.1109/CRV.2019.00019.

[39] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, 30(11):1343–1377, 2011. doi: 10.1177/0278364911410755.

[40] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3):195–207, 1998. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(98)00049-9. Autonomous Mobile Robots.

[41] Giuseppe Borghi and Vincenzo Caglioti. Minimum uncertainty explorations in the self-localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 14(6):902–911, 1998. doi: 10.1109/70.736774.

[42] Patric Jensfelt and Steen Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760, 2001. doi: 10.1109/70.964673.

[43] Christian Mostegel, Andreas Wendel, and Horst Bischof. Active monocular localization: Towards autonomous monocular exploration for multirotor mavs. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3848–3855, 2014. doi: 10.1109/ICRA.2014.6907417.

[44] Sai Krishna Gottipati, Keehong Seo, Dhaivat Bhatt, Vincent Mai, Krishna Murthy, and Liam Paull. Deep active localization. *IEEE Robotics and Automation Letters*, 4(4):4394–4401, 2019. doi: 10.1109/LRA.2019.2932575.

[45] Jared Strader, Kyohei Otsu, and Ali-akbar Agha-mohammadi. Perception-aware autonomous mast motion planning for planetary exploration rovers. *Journal of Field Robotics*, 37(5):812–829, December 2019. ISSN 1556-4967. doi: 10.1002/rob.21925.

[46] Andrew Davison and David Murray. Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24:865–880, 08 2002. doi: 10.1109/TPAMI.2002.1017615.

[47] Sebastian Thrun and Knut Möller. Active exploration in dynamic environments. In *Neural Information Processing Systems*, 1991.

[48] Frédéric Bourgault, Alexei Makarenko, Stefan Williams, Ben Grocholsky, and Hugh Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545 vol.1, 2002. doi: 10.1109/IRDS.2002.1041446.

[49] Paul Newman, Michael Bosse, and John Leonard. Autonomous feature-based exploration. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 1234–1240 vol.1, 2003. doi: 10.1109/ROBOT.2003.1241761.

[50] Hans Jacob S. Feder, John J. Leonard, and Christopher M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18 (7):650–668, 1999. doi: 10.1177/02783649922066484.

[51] Richard Bormann, Florian Jordan, Wenzhe Li, Joshua Hampp, and Martin Hägele. Room segmentation: Survey, implementation, and analysis. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1019–1026, 2016. doi: 10.1109/ICRA.2016.7487234.

[52] Beipeng Mu, Matthew Giamou, Liam Paull, Ali-akbar Agha-mohammadi, John Leonard, and Jonathan How. Information-based active slam via topological feature graphs. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5583–5590, 2016. doi: 10.1109/CDC.2016.7799127.

[53] Leonardo Fermín-Leon, José Neira, and José A. Castellanos. Tigre: Topological graph based robotic exploration. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2017. doi: 10.1109/ECMR.2017.8098718.

[54] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active slam using model predictive control and attractor based exploration. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5031, 2006. doi: 10.1109/IROS.2006.282530.

[55] Yongbo Chen, Shoudong Huang, and Robert Fitch. Active slam for mobile robots with area coverage and obstacle avoidance. *IEEE/ASME Transactions on Mechatronics*, 25(3):1182–1192, 2020. doi: 10.1109/TMECH.2019.2963439.

[56] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J. Pappas. Decentralized active information acquisition: Theory and application to multi-robot slam. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4775–4782, 2015. doi: 10.1109/ICRA.2015.7139863.

[57] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. doi: 10.1109/2.30720.

[58] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Mag.*, 9(2): 61–74, jun 1988. ISSN 0738-4602. doi: 10.1609/aimag.v9i2.676.

[59] Henry Carrillo, Philip Dames, Vijay Kumar, and Jose Castellanos. Autonomous robotic exploration using a utility function based on rényi's general theory of entropy. *Autonomous Robots*, 42, 02 2018. doi: 10.1007/s10514-017-9662-9.

[60] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Kumar Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *ArXiv*, abs/2004.05155, 2020.

[61] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019. doi: 10.1109/LRA.2019.2891991.

[62] Julio A. Placed and José A. Castellanos. Fast uncertainty quantification for active graph slam. *ArXiv*, abs/2110.01289, 2021.

[63] Armin Hornung, Kai M. Wurm, Maren Bennewitz, C. Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34:189 – 206, 2013.

[64] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul H. J. Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2): 1144–1151, 2018. doi: 10.1109/LRA.2018.2792537.

[65] Manasi Muglikar, Zichao Zhang, and Davide Scaramuzza. Voxel map for visual slam. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4181–4187, 2020. doi: 10.1109/ICRA40945.2020.9197357.

[66] Narcís Palomeras, Marc Carreras, and Juan Andrade-Cetto. Active slam for autonomous underwater exploration. *Remote Sensing*, 11(23), 2019. ISSN 2072-4292. doi: 10.3390/rs11232827.

[67] Magnus Selin, Mattias Tiger, Daniel Duberg, Fredrik Heintz, and Patric Jensfelt. Efficient autonomous exploration planning of large-scale 3-d environments. *IEEE Robotics and Automation Letters*, 4(2):1699–1706, 2019. doi: 10.1109/LRA.2019.2897343.

[68] Di Deng, Zhefan Xu, Wenbo Zhao, and Kenji Shimada. Frontier-based automatic-differentiable information gain measure for robotic exploration of unknown 3d environments. *ArXiv*, abs/2011.05288, 2020.

[69] Ana Batinovic, Tamara Petrovic, Antun Ivanovic, Frano Petric, and Stjepan Bogdan. A multi-resolution frontier-based planner for autonomous 3d exploration. *IEEE Robotics and Automation Letters*, 6(3):4528–4535, 2021. doi: 10.1109/LRA.2021.3068923.

[70] Sean L. Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J. Pappas. Probabilistic data association for semantic slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729, 2017. doi: 10.1109/ICRA.2017.7989203.

[71] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2019. doi: 10.1109/LRA.2018.2866205.

[72] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019. doi: 10.1109/LRA.2019.2923960.

[73] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, 2020. doi: 10.1109/ICRA40945.2020.9196885.

[74] John Mccormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41, 2018. doi: 10.1109/3DV.2018.00015.

[75] Robert Eidenberger and Josef Scharinger. Active perception and scene modeling by planning with probabilistic 6d object poses. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1036–1043, 2010. doi: 10.1109/IROS.2010.5651927.

[76] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J. Pappas, and Kostas Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, 30(5):1078–1090, 2014. doi: 10.1109/TRO.2014.2320795.

[77] Sebastian Thrun and Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 944–950. AAAI Press, 1996. ISBN 026251091X.

[78] Nicola Tomatis, Illah Nourbakhsh, and Roland Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3–14, 2003. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(03)00006-X. Best Papers of the Eurobot '01 Workshop.

[79] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40 (12-14):1510–1546, 2021. doi: 10.1177/02783649211056674.

[80] Clara Gomez, Alejandra C. Hernandez, and Ramon Barber. Topological frontier-based exploration and map-building using semantic information. *Sensors*, 19(20), 2019. ISSN 1424-8220. doi: 10.3390/s19204595.

[81] Jose Luis Blanco, J.-A Fernández-Madrigal, and Javier González-Jiménez. A novel measure of uncertainty for mobile robot slam with rao blackwellized particle filters. *I. J. Robotic Res.*, 27:73–89, 01 2008. doi: 10.1177/0278364907082610.

[82] Luca Carlone, Jingjing Du, Miguel Kaouk, Basilio Bona, and Marina Indri. Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent and Robotic Systems*, 75, 08 2013. doi: 10.1007/s10846-013-9981-9.

[83] Héctor H. González-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21 (10-11):829–848, 2002. doi: 10.1177/027836402021010834.

[84] Benjamín Tovar, Lourdes Muñoz-Gómez, Rafael Murrieta-Cid, Moisés Alencastre-Miranda, Raúl Monroy, and Seth Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4): 314–331, 2006. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2005.11.006.

[85] Matan Keidar and Gal A. Kaminka. Efficient frontier detection for robot exploration. *Int. J. Rob. Res.*, 33(2):215–236, feb 2014. ISSN 0278-3649. doi: 10.1177/0278364913494911.

[86] Phillip Quin, Dac Dang Khoa Nguyen, Thanh Long Vu, Alen Alempijevic, and Gavin Paul. Approaches for efficiently detecting frontier cells in robotics exploration. *Frontiers in Robotics and AI*, 8, 2021. ISSN 2296-9144. doi: 10.3389/frobt.2021.616470.

[87] Matan Keidar and Gal A. Kaminka. Robot exploration with fast frontier detection: theory and experiments. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 113–120, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0981738117.

[88] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, 2010.

[89] Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1396–1402, 2017. doi: 10.1109/IROS.2017.8202319.

[90] Cheng-Yan Wu and Huei-Yung Lin. Autonomous mobile robot exploration in unknown indoor environments based on rapidly-exploring random tree. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 1345–1350, 2019. doi: 10.1109/ICIT.2019.8754938.

[91] Wenchuan Qiao, Zheng Fang, and Bailu Si. Sample-based frontier detection for autonomous robot exploration. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1165–1170, 2018. doi: 10.1109/ROBIO.2018.8665066.

[92] Anna Dai, Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, and Stefan Leutenegger. Fast frontier-based information-driven autonomous exploration with an mav. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9570–9576, 2020. doi: 10.1109/ICRA40945.2020.9196707.

[93] Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3d. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 351–356, 2011. doi: 10.1109/SSRR.2011.6106778.

[94] Cheng Zhu, Rong Ding, Mengxiang Lin, and Yuanyuan Wu. A 3d frontier-based exploration tool for mavs. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 348–352, 2015. doi: 10.1109/ICTAI.2015.60.

[95] Namal Senarathne and Danwei Wang. Towards autonomous 3d exploration using surface frontiers. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 34–41, 2016. doi: 10.1109/SSRR.2016.7784274.

[96] Shaojie Shen, Nathan Michael, and Vijay R. Kumar. Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle. *The International Journal of Robotics Research*, 31:1431 – 1444, 2012.

[97] Liang Lu, Carlos Redondo, and Pascual Campoy. Optimal frontier-based autonomous exploration in unconstructed environment using rgb-d sensor. *Sensors*, 20(22), 2020. ISSN 1424-8220.

[98] Cyrill Stachniss, D. Hahnel, and Wolfram Burgard. Exploration with active loop-closing for fastslam. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1505–1510 vol.2, 2004. doi: 10.1109/IROS.2004.1389609.

[99] Robert Grabowski, Pradeep Khosla, and H. Choset. Autonomous exploration via regions of interest. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1691–1696 vol.2, 2003. doi: 10.1109/IROS.2003.1248887.

[100] Rafael Valencia, Jaime Valls Miró, Gamini Dissanayake, and Juan Andrade-Cetto. Active pose slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1885–1891, 2012. doi: 10.1109/IROS.2012.6385637.

[101] Narcís Palomeras, Natalia Hurtós, Eduard Vidal, and Marc Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2):1619–1625, 2019. doi: 10.1109/LRA.2019.2896759.

[102] Sudharshan Suresh, Paloma Sodhi, Joshua G. Mangelson, David Wettergreen, and Michael Kaess. Active slam using 3d submap saliency for underwater volumetric exploration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3132–3138, 2020. doi: 10.1109/ICRA40945.2020.9196939.

[103] Ayoung Kim and Ryan Eustice. Active visual slam for robotic area coverage: Theory and experiment. *The International Journal of Robotics Research*, 34:457–475, 04 2014. doi: 10.1177/0278364914547893.

[104] Julio A. Placed and José A. Castellanos. A deep reinforcement learning approach for active slam. *Applied Sciences*, 2020.

[105] Delong Zhu, Tingguang Li, Danny Ho, Chaoqun Wang, and Max Q.-H. Meng. Deep reinforcement learning supervised autonomous exploration in office environments.

In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7548–7555, 2018. doi: 10.1109/ICRA.2018.8463213.

[106] Hans Jacob S. Feder, John J. Leonard, and Christopher M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18: 650 – 668, 1999.

[107] Jur P. van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31:1263 – 1278, 2012.

[108] Clara Gómez, Alejandra Carolina Hernández, and Ramón Barber. Topological frontier-based exploration and map-building using semantic information. *Sensors (Basel, Switzerland)*, 19, 2019.

[109] Alan C. Schultz, William Adams, and Brian Yamauchi. Integrating exploration, localization, navigation and planning with a common representation. *Auton. Robots*, 6(3):293–308, June 1999. ISSN 0929-5593. doi: 10.1023/A:1008936413435.

[110] Miguel Juliá, Arturo Gil, and Óscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33:427–444, 2012.

[111] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[112] Rafael Gonçalves Colares and Luiz Chaimowicz. The next frontier: combining information gain and distance cost for decentralized multi-robot exploration. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016.

[113] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, 2005.

[114] Joan Vallvé and Juan Andrade-Cetto. Dense entropy decrease estimation for mobile robot exploration. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6083–6089, 2014. doi: 10.1109/ICRA.2014.6907755.

[115] Marija Popović, Teresa Vidal-Calleja, Jen Jen Chung, Juan Nieto, and Roland Siegwart. Informative path planning for active field mapping under localization uncertainty. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10751–10757, 2020. doi: 10.1109/ICRA40945.2020.9197034.

[116] Jinkun Wang and Brendan Englot. Autonomous exploration with expectation-maximization. In *International Symposium of Robotics Research*, 2017.

[117] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: a factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, 2002.

[118] Cyrill Stachniss and Wolfram Burgard. Mapping and exploration with mobile robots using coverage maps. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 467–472 vol.1, 2003. doi: 10.1109/IROS.2003.1250673.

[119] Solomon Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851.

[120] Lyudmila S. Mihaylova, Tine Lefebvre, Herman Bruyninckx, Klaas Gadeyne, and Joris De Schutter. A comparison of decision making criteria and optimization methods for active robotic sensing. In *Numerical Methods and Application*, 2002.

[121] Herman Chernoff. Locally optimal designs for estimating parameters. *Annals of Mathematical Statistics*, 24:586–602, 1953.

[122] Sylvain Ehrenfeld. On the Efficiency of Experimental Designs. *The Annals of Mathematical Statistics*, 26(2):247 – 255, 1955. doi: 10.1214/aoms/1177728541.

[123] Abraham Wald. On the efficient design of statistical investigations. *Annals of Mathematical Statistics*, 14:134–140, 1943.

[124] Robert Sim and Nabadip Roy. Global a-optimal robot exploration in slam. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 661–666, 2005. doi: 10.1109/ROBOT.2005.1570193.

[125] Ching-Shui Cheng. Maximizing the total number of spanning trees in a graph: Two related problems in graph theory and optimum design theory. *Journal of Combinatorial Theory, Series B*, 31(2):240–248, 1981. ISSN 0095-8956. doi: https://doi.org/10.1016/S0095-8956(81)80028-7.

[126] Kasra Khosoussi, Shoudong Huang, and Gamini Dissanayake. Novel insights into the impact of graph structure on slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2707–2714, 2014. doi: 10.1109/IROS.2014.6942932.

[127] Kasra Khosoussi, Matthew Giamou, Gaurav S. Sukhatme, Shoudong Huang, Gamini Dissanayake, and Jonathan P. How. Reliable graphs for slam. *The International Journal of Robotics Research*, 38:260 – 298, 2019.

[128] Yongbo Chen, Shoudong Huang, Liang Zhao, and Gamini Dissanayake. Cramér–rao bounds and optimal design metrics for pose-graph slam. *IEEE Transactions on Robotics*, 37(2):627–641, 2021. doi: 10.1109/TRO.2020.3001718.

[129] Julio A. Placed and José A. Castellanos. Fast autonomous robotic exploration using the underlying graph structure. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6672–6679, 2021. doi: 10.1109/IROS51168.2021.9636148.

[130] Julio A. Placed and José A. Castellanos. Enough is enough: Towards autonomous uncertainty-driven stopping criteria. *ArXiv*, abs/2204.10631, 2022.

[131] Vadim Indelman. No correlations involved: Decision making under uncertainty in a conservative sparse information space. *IEEE Robotics and Automation Letters*, 1 (1):407–414, 2016. doi: 10.1109/LRA.2016.2518224.

[132] Yongbo Chen, Shoudong Huang, Robert Fitch, and Jianqiao Yu. Efficient active slam based on submap joining, graph topology and convex optimization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5159–5166, 2018. doi: 10.1109/ICRA.2018.8460864.

[133] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. Slam++ -a highly efficient and temporally scalable incremental slam framework. *The International Journal of Robotics Research*, 36:027836491769111, 02 2017. doi: 10.1177/0278364917691110.

[134] Teng Zhang, Kanzhi Wu, Jingwei Song, Shoudong Huang, and Gamini Dissanayake. Convergence and consistency analysis for a 3-d invariant-ekf slam. *IEEE Robotics and Automation Letters*, 2(2):733–740, 2017. doi: 10.1109/LRA.2017.2651376.

[135] James Doran and Donald Michie. Experiments with the graph traverser program. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 294:235 – 259, 1966.

[136] Steven LaValle and James Kuffner. Randomized kinodynamic planning. volume 20, pages 473–479, 01 1999. doi: 10.1109/ROBOT.1999.770022.

[137] Fan Yang, Chao Cao, Hongbiao Zhu, Jean Oh, and Ji Zhang. Far planner: Fast, attemptable route planner using dynamic visibility update. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9–16, 2022. doi: 10.1109/IROS47612.2022.9981574.

[138] Su-Yong An, Lae-Kyoung Lee, and Se-Young Oh. Ceiling vision-based active slam framework for dynamic and wide-open environments. In *Autonomous Robots*, page 40:291–324, 2015.

[139] Nikolas Brasch, Aljaz Bozic, Joe Lallemand, and Federico Tombari. Semantic monocular slam for highly dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 393–400, 2018. doi: 10.1109/IROS.2018.8593828.

[140] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. Dynamic slam: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129, 2020. doi: 10.1109/ICRA40945.2020.9196895.

[141] Berta Bescos, Carlos Campos, Juan D. Tardós, and José Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *IEEE Robotics and Automation Letters*, 6 (3):5191–5198, 2021. doi: 10.1109/LRA.2021.3068640.

[142] Jesse Morris, Yiduo Wang, and Viorela Ila. The importance of coordinate frames in dynamic slam, 2024.

[143] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015. ISSN 1941-0468. doi: 10.1109/tro.2015.2463671.

[144] Pan Ji, Hongdong Li, Yuchao Dai, and Ian Reid. "maximizing rigidity" revisited: a convex programming approach for generic 3d shape reconstruction from multiple perspective views. 07 2017. doi: 10.48550/arXiv.1707.05009.

[145] Ajad Chhatkuli, Daniel Pizarro, Toby Collins, and Adrien Bartoli. Inextensible non-rigid structure-from-motion by second-order cone programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2428–2441, 2018. doi: 10.1109/TPAMI.2017.2762669.

[146] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Local deformable 3d reconstruction with cartan's connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(12):3011–3026, 2020. doi: 10.1109/TPAMI.2019.2920821.

[147] Shaifali Parashar, Mathieu Salzmann, and Pascal Fua. Local non-rigid structure-from-motion from diffeomorphic mappings. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2056–2064, 2020. doi: 10.1109/CVPR42600.2020.00213.

[148] Juan J. Gómez Rodríguez, José M.M. Montiel, and Juan D. Tardós. Nr-slam: Non-rigid monocular slam. *IEEE Transactions on Robotics*, 40:4252–4264, 2024. doi: 10.1109/TRO.2024.3422004.

[149] Elliot Anshelevich, Scott Owens, Florent Lamiraux, and Lydia E. Kavraki. Deformable volumes in path planning applications. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 3:2290–2295 vol.3, 2000.

[150] Samuel Rodríguez, Jyh-Ming Lien, and Nancy M. Amato. Planning motion in completely deformable environments. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2466–2471, 2006.

[151] Nicolò Botteghi, Beril Kallfelz Sirmacek, R. Schulte, Mannes Poel, and Christoph Brune. Reinforcement learning helps slam: Learning to build maps. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2020:329–335, 2020. doi: 10.5194/isprs-archives-XLIII-B4-2020-329-2020.

[152] Mikhail Frank, Juxi Leitner, Marijn Stollenga, Alexander Förster, and Jürgen Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in neurorobotics*, 7:25, 01 2014. doi: 10.3389/fnbot.2013.00025.

[153] Lei Tai and Ming Liu. A robot exploration strategy based on q-learning network. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 57–62, 2016. doi: 10.1109/RCAR.2016.7784001.

[154] Jingwei Zhang, Lei Tai, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Neural slam: Learning to explore with external memory, 2020.

[155] Kaichena Zhang, Farzad Niroui, Maurizio Ficocelli, and Goldie Nejat. Robot navigation of environments with unknown rough terrain using deep reinforcement learning. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7, 2018. doi: 10.1109/SSRR.2018.8468643.

[156] Thomas Kollar and Nicholas Roy. Trajectory optimization using reinforcement learning for map exploration. *I. J. Robotic Res.*, 27:175–196, 02 2008. doi: 10.1177/0278364907087426.

[157] Bastian van Manen. Learning to explore and map with ekf slam and rl, August 2021. URL `http://essay.utwente.nl/88118/`.

[158] Pierre-Antoine Absil, Christopher G. Baker, and Kyle A. Gallivan. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7: 303–330, 2007.

[159] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, oct 1979. ISSN 0001-0782. doi: 10.1145/359156.359164.

[160] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.018.

[161] Hannes Kenngott, J Wünscher, Martin Wagner, A Preukschas, anna-laura Wekerle, Peter Neher, Stefan Suwelack, Stefanie Speidel, Felix Nickel, Dare Oladokun, Lena Maier-Hein, Rüdiger Dillmann, Hans-Peter Meinzer, and Beat Müller. Openhelp (heidelberg laparoscopy phantom): development of an open-source surgical evaluation and training tool. *Surgical endoscopy*, 29(11):3338—3347, November 2015. ISSN 0930-2794. doi: 10.1007/s00464-015-4094-0.