Multi-hypothesis Incremental Smoothing and Mapping for Ambiguity-aware Passive and Active SLAM

Ming Hsiao

CMU-RI-TR-20-14

The Robotics Insitute
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Keywords: Simultaneous localization and mapping (SLAM); ambiguity-aware SLAM; multi-hypothesis SLAM; robust optimizer; state estimation; RGB-D-inertial odometry; direct tracking; dense 3D reconstruction; planar SLAM; active SLAM; path planning

In memory of my grandpa

Abstract

Simultaneous localization and mapping (SLAM) is the problem of estimating the state of a moving agent with sensor(s) on it while simultaneously reconstructing a map of its surrounding environment, which has been a popular research field due to its wide applications, such as inspection and reconstruction, autonomous transportation and delivery, virtual/augmented/mixed reality (VR/AR/MR), search and rescue, and all kinds of service robotics that involve moving platforms. As many state-of-the-art SLAM algorithms can already achieve high accuracy in both state estimation and mapping, improving the *robustness* of SLAM systems has become a research focus in both academia and the industry in recent years.

The most common challenge to robustness is insufficient information. When no information is observed by the adopted sensors, e.g.: the view of a camera is fully blocked, and the signal of the *Global Positioning System (GPS)* is denied in indoor or underwater environments, it is impossible to estimate the state or the map correctly. As a result, most real world SLAM systems adopt multiple sensors in order to achieve better robustness. However, occasionally the observations from all the sensors might still be insufficient to determine a unique solution. For example, when the visual odometry (VO) estimation does not agree with the pose estimation of the GPS, it can be hard to tell which one is correct, or maybe both of them are wrong. In the case that more than one interpretation could be plausible for the same observations, which is known as the *ambiguity* problem, it is theoretically possible to keep track of all the highly likely interpretations until more information is observed later to disambiguate the ambiguities. However, most of the state-of-the-art SLAM systems only estimate a single solution (and possibly unimodal uncertainties) without considering the impact from ambiguous measurements in the entire pipeline.

Therefore, in this thesis, a novel multi-hypothesis back-end optimizer called MH-iSAM2 is introduced to take ambiguities into account and output multi-hypothesis solutions when the ambiguities are temporarily unsolvable. Our novel optimizer allows nonlinear incremental updates in all hypotheses while avoiding redundant computations across different hypotheses, which results in better efficiency than computing each hypothesis individually. Then, an *ambiguity-aware planar-inertial SLAM (API-SLAM)* system is developed based on MH-iSAM2 to reconstruct dense 3D models of indoor environments in real-time, which provides an example of applying MH-iSAM2 in a *multi-hypothesis SLAM (MH-SLAM)* framework for better robustness. Finally, an *ambiguity-aware active SLAM* framework is proposed to make use of the multi-hypothesis state and map estimates from the MH-SLAM system in decision making and path planning, which demonstrates a complete and interactive usage of the multi-hypothesis estimations in a real-world robotic system. The experimental results show that MH-iSAM2 can be applied properly to improve the robustness of a SLAM or active SLAM system, especially for handling the ambiguity problem in real-world tasks.

Acknowledgments

First and foremost, I want to express my deepest appreciation to my advisor Michael Kaess for his tremendous guidance, support, and trust throughout my PhD journey at Carnegie Mellon University (CMU). Michael has been my role model of a great researcher and mentor. He provided me sufficient opportunities and freedom to explore research directions that interest me most, helped me overcome various challenges to complete this dissertation, and led me by example with his wisdom, dedication, and kindness. I could not imagine having a better advisor than Michael.

I would like to thank my thesis defense committee members sincerely for all the thoughtful feedback and inspiring discussions. Sebastian Scherer has been reminding me the importance of having a realistic research plan and a solid evaluation procedure since the very beginning, which had helped me make this work more complete and convincing. George Kantor has brought me many ideas of extending the theoretical part of this work to solve various robotic problems, which motivated me to explore more possibilities in my research directions. Christian Debrunner has been strongly supporting me and my research projects for years. Our enthusiastic discussions in the monthly meetings had helped enhance the quality of this work thoroughly from its theoretical foundation to its real world applications. My qualifier committee members, Simon Lucey and Chung-Yao Chuang, had also provided valuable feedback in the early stage of this work. All of them have helped me pursue higher goals in my research process, which gave me the chance to not only improve this dissertation but also sharpen my research skills comprehensively.

It has been an honor and privilege to work with my colleagues and peers in the Robot Perception Lab (RPL) at CMU. They have been the most motivated, creative, and hard-working people that I have ever worked with. I cannot overemphasize how important Eric Westman has been to this work and to myself. The first project Eric and I worked on together had become an inspiration and a crucial foundation of this research. And throughout all these years, Eric has been not only a wonderful labmate that I really enjoy collaborating and brainstorming with, but also a true friend who have helped me in many aspects wholeheartedly. As the senior members in the lab, Joshua Mangelson, Guofeng Zhang, Rafael Valencia, and Lipu Zhou had shared their knowledge, experiences, and insights openly with me, which broadened my horizon beyond my imagination. All of the other RPL members had made positive impacts to me and my research, and I will never forget the great time we had together. Tiffany, Garrett, Puneet, Jerry, Bing, Paloma, Monty, Suddhu, Jack, Zimo, Allie, Akshay, Wei, Chenfeng, Joshua, Eric, Allison, Akash, Prakruti, Daoyuan, Anand, and all others who have been with RPL, thank you all from the bottom of my heart.

It would not have been possible to complete this dissertation without the funding support from Lockheed Martin Corporation, National Science Foundation, Naval Research, and Taiwan Ministry of Education Study Abroad Scholarship.

The CMU community has been a strong support to me. I want to thank Chieh-Chih Wang for inspiring and helping me since we met at National Taiwan University (NTU). Without him, I might not even start my dream of studying PhD in Robotics at CMU. I also want to thank Tejas, Sibi, Shaurya, Rai, and all other RI PhDs in the same year for being such great officemates, classmates, “free food tellers”, “deadline companions”, and most importantly, comrades that have encouraged each others countless times through this long challenging journey. Chuck, Suzanne, Nora and many other CMU staffs had helped me a lot to make my research and school life much easier. As an international student, I am grateful to have a group of fellows from the same hometown that helped and supported each other in the new environment: Yao, Hush, Vincent, Shao-Po, Yen-Chi, “Boss” Hua, Chia-Yin, De-an, Shihyun, “JG” Chen-Hsuan, Wei-Chiu, Roger, Yiu-Chang, Shih-En, Zach, Robin, Yu-Te, Po-Wei, Guan-Horng, Wei-Hsin, Samuel, Man-Ning, and many others that I have met in CMU.

In addition to studying and doing research, playing volleyball and potluck with friends on weekends had been my favorite activities. I enjoyed every moment with “Sir” Shoou-i, “Coach” Yu-San, “Gary” Chien-Hua, “Miang Chang” Chang, Lin Wei, Watson, Li Lin & Ya-Ting, Kevin, Sz-Rung, Tzu-Ruey, Austin, Willian, Tzu-Chieh, I-Chen, Chih-Yang, Ching-Yi, Jessie, Bowen & Xin, and all others who had joined the activities. Thank you for balancing my life and recharging my energy all the time. The caring and encouragement from my old friends, no matter in Taiwan or abroad, had also been a great support and solace to me throughout the journey.

I am extremely grateful to my family for the unfailing love, support, and respect in my life. My parents have been the strongest support for me to focus on studying and digging into my favorite fields, respecting every of my decisions since I was a kid. My aunt and cousins in Canada always welcome my visits warmly, making me feel like having another home. Last but definitely not least, I can never thank my wife Ling-Wan enough. She has loved, accompanied, and supported me through all the ups and downs, blessing me with a colorful PhD life and beyond.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is SLAM?

Simultaneous localization and mapping (SLAM) is a well-known problem in robotics research, which consists of two fundamental parts: *localization* and *mapping*. For any mobile robot that desires to conduct a task, knowing its *location* (or *pose* to be more specific, which consists of both *position* and *orientation*) with respect to a *map* of the environment is important; how to estimate the location of the mobile robot with available information (e.g.: sensor measurements, prior knowledge, control commands, etc) is the localization problem. A common approach to localize a robot in a known map is to first match the distinctive sensor measurements (also known as *features*) with the distinctive structures or textures in the map (also known as *landmarks*), and then compute the location of the robot directly. On the other hand, one of the best ways to describe an environment is to draw a map of it; how to generate a map automatically from sensor measurements is the mapping problem. Since most sensors have limited ranges, perform better when they are closer to the target, or can be occluded by obstacles, it is desirable to move the sensors around to better map a larger environment, assuming that the poses of the sensors are known throughout the mapping process.

Even though localization and mapping are relatively easy to solve separately, in many real world scenarios they have to be solved at the same time. For example, an up-to-date map of the environment for localization might not be available because the robot is exploring a new area, or the environment has been changed since it was mapped. In these cases, it is desirable for the robot to also generate the map or update an existing map for better localization results. As for mapping, the assumption of known sensor poses is hard to satisfy in reality without conducting any localization process in advance. Therefore, SLAM is defined to describe this chicken-and-egg problem considering localization and mapping at the same time, which is a much more challenging problem and is still an ongoing research field in the robotics community even after more than 30 years of study [8].

A typical framework to solve the SLAM problem is to divide the system into two parts: *front-end* and *back-end*. The front-end usually consists of data preprocessing, feature extraction, and data association, while the back-end is typically a nonlinear solver that aims at optimizing the state and map together based on the information passed from the front-end. This common SLAM framework can be extended to solve *active SLAM* problems [128][10][12] by incorporating control, planning, and decision making modules while conducting localization and mapping, which is expected to achieve even better accuracy, coverage, robustness, and efficiency compared to the passive approach.

1.2 Ambiguity Problems in SLAM

As various state-of-the-art SLAM solutions have been proposed to achieve high accuracy in state estimation, mapping, or both [20][82][14], new research directions in SLAM other than improving accuracy alone have been of great interest to the SLAM community in recent years [8]. In this thesis, we focus on handling the *ambiguity* problems explicitly in order to improve the *robustness* of existing SLAM solutions.

Ambiguity is defined as the situation when more than one interpretation is plausible for the same observations. A basic concept of ambiguity in robot perception is shown in Fig. 1.1, where the robot cannot tell which of the two possible hypotheses is the correct one. The source of ambiguity depends a lot on the sensor and the adopted algorithm in the front-end. For example, when conducting SLAM with visual sensors and using feature-based algorithms, moving objects, lighting changes, textureless scenes, repeated patterns, and motion blurs that result from aggressive motions can all cause ambiguities in the measurements. However, since most of the existing front-end algorithms only model the observations with a single mode instead of all the possible modes, some wrong modes might be taken and result in outliers in the measurements. And since most of the existing back-end solvers (e.g.: g2o[61], Ceres [2], iSAM [49], and iSAM2 [50]) can only handle single mode measurements and output only one solution for each state (see Fig. 1.2-a), outlier measurements can lead to errors in the estimations or even failures of the entire SLAM system.

1.3 Multi-hypothesis Solver

To model and solve the ambiguity problems in SLAM, we argue that it would be desirable for front-end algorithms to detect and model the temporarily unsolvable ambiguities explicitly, and pass these information to a back-end solver that can explicitly account for the ambiguities and output all the highly probable solutions (see Fig. 1.2-b). This proposed framework allows the later modules in the robotic system (e.g.: control or planning) to be aware of the temporarily unsolvable ambiguities and therefore is expected to greatly enhance the robustness of the entire robotic system.

To realize the proposed framework, a back-end optimizer that can keep track of the ambiguities in the measurements and compute the corresponding solutions to each combination of the modes of all the ambiguous measurements is crucial. And since SLAM is an online problem that is desired to be solved in real-time, the capability of *incrementally updating* the current solutions based on the previous solutions is highly desired for better efficiency. One popular solution to enable incremental updates in SLAM is called *incremental smoothing and mapping (iSAM)*, which was first developed in [49]. Later, a new iSAM algorithm called iSAM2 [50] is

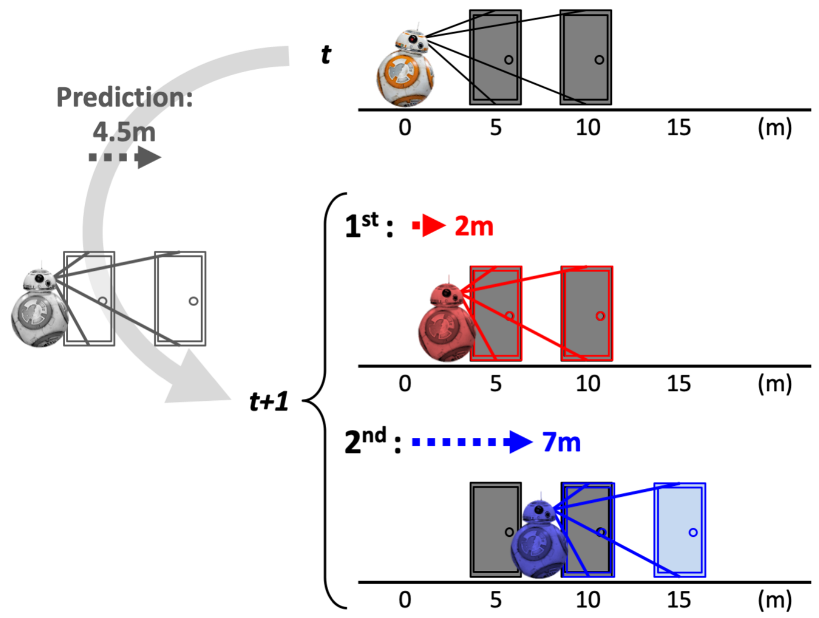


Figure 1.1: An example of ambiguity problem in SLAM. A robot (image source: https://en.wikipedia.org/wiki/BB-8) is moving down a long corridor, trying to localize itself while mapping the environment. Assuming that at time t, the robot observes two identical doors in front of it (top right). As a result, the robot generates a map that contains the information of these two doors (gray). Then, the robot moves forward for a certain distance (with a rough prediction of 4.5m), and observes two identical doors again at time $t + 1$. Based on the prediction, there are two similarly likely hypotheses that both satisfy the observations. The first hypothesis (middle right) is that the robot actually only moves forward for 2 meters and sees the same two doors. In this case, no new information should be added into the map. In the second hypothesis (bottom right), the robot moves forward for 7 meters and sees a new door and an old one. In this case, the map should be updated with the new door (light blue). From this example, we can have a general idea on how ambiguity can result in multiple hypotheses in both localization and mapping.

developed to work better with rotation and other nonlinear terms that are heavily involved in a SLAM problem, which is so far the only state-of-the-art SLAM optimizer that allows nonlinear incremental updates.

In this thesis, we propose a multi-hypothesis back-end optimizer called *multi-hypothesis iSAM using Bayes tree and Hypo-tree (MH-iSAM2)* based on the original iSAM2 algorithm [50], which tracks all highly possible ambiguous hypotheses, updates the multiple possible solutions of each state incrementally in a nonlinear fashion, and solves the ambiguities by pruning the less likely hypotheses when sufficient information is received. Through simulations, we evaluate the robustness, accuracy, and efficiency of the proposed MH-iSAM2 algorithm, and demonstrate its unique properties as the first multi-hypothesis nonlinear incremental solver for SLAM problems.

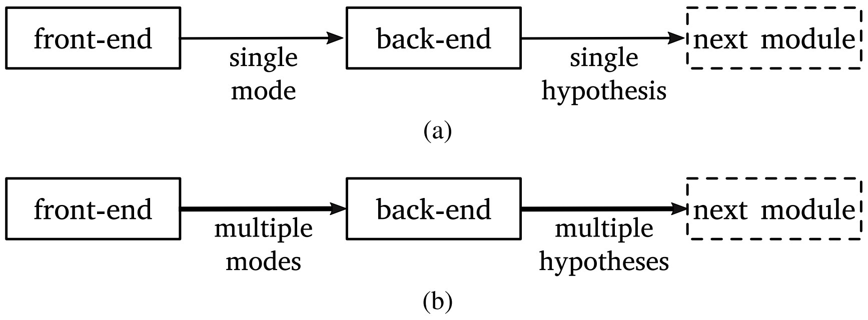


Figure 1.2: Two types of SLAM frameworks: (a) In most modern SLAM systems, the back-end optimizer assumes that both of its inputs and outputs have only one single possibility. (b) However, we argue that a back-end optimizer that can handle multiple possibilities should achieve better robustness for not only SLAM but also the entire robotic system.

1.4 Ambiguity-aware Passive SLAM

Based on MH-iSAM2, a real world multi-hypothesis SLAM (MH-SLAM) algorithm is developed to show an example of integrating MH-iSAM2 with an existing SLAM solution to improve robustness against ambiguities. Comparing to outdoor environments that are mostly well mapped due to the necessity for automobile navigation and public accessibility, indoor environments are where people spend most of their time but are not properly mapped in general due to their compartmentalized design and to privacy issues. Therefore, we focus on developing an out-of-the-box ambiguity-aware MH-SLAM solution that can reconstruct 3D models of indoor environments.

Among various existing 3D SLAM solutions, some reconstruct *sparse* maps that usually consist of feature points (e.g.: ORB-SLAM [82]), which are useful for localization or state estimation, but not suitable for inspection, semantic understanding, path planning, visualization, or other interactive functions. On the contrary, *dense* map representations such as pointcloud (e.g.: pointcloud-based object map [107]), surfel (e.g.: ElasticFusion [134]), mesh (e.g.: mesh reconstruction step [45] in OpenMVS [92]), occupancy grid (e.g.: normal distributions transform occupancy maps [109]), or signed distance field (SDF) (e.g.: KinectFusion [85]) all preserve much richer information of the environment and therefore allow much more applications. However, more computation and memory are usually required to construct these dense maps than sparse maps, and GPU acceleration is usually required for real-time operation on these methods. As a result, it is desired to incorporate an efficient mapping algorithm into the proposed ambiguity-aware passive SLAM system that can reconstruct dense 3D models in real-time on only a CPU, which will allow wider applications on lightweight platforms and mobile devices.

To achieve the goals, we build the ambiguity-aware MH-SLAM system based on DPI-SLAM (see Ch. 5), which utilizes an RGB-D camera and an inertial measurement unit (IMU) to estimate the motion states and reconstruct dense 3D models of indoor environments with pointcloud submaps and planes in real-time on a CPU. Several front-end components of DPI-SLAM are modified for ambiguity detection, and the original iSAM2 back-end is replaced by a hierarchical optimization approach consisting of iSAM2 and MH-iSAM2. The experimental results on real world datasets show the significant improvements on robustness because of handling ambiguities explicitly using the proposed multi-hypothesis framework.

1.5 Ambiguity-aware Active SLAM

Most existing SLAM algorithms assume that the motion of the sensors is determined by a separate source, and the measurements are also received accordingly. As a result, those SLAM algorithms only passively map the environment using the given data. However, in many real-world applications, it is allowed or even desired to actively decide the motion of the sensors online during the SLAM process based on the received information. For example, when a robot is exploring a new indoor environment and constructing a map at the same time, it can keep going toward unknown areas based on the most up-to-date information from the growing map, or decide to go back to a visited place to correct the accumulated drift in the map through loop closing. As a result, this approach, which is known as *active SLAM* in the robotics community, is crucial for a fully-autonomous mobile robotic system.

A more formal definition of active SLAM is as follows: it is the problem of actively exploring an environment with moving sensors while simultaneously estimating the state of the sensors and reconstructing a map that satisfies certain requirements (e.g. bounded uncertainty). Most existing active SLAM algorithms [128][10][12] consider unimodal state and map estimations for decision making and planning as they are built upon state estimation [43][18][125] and mapping methods [79][36] that only estimate single solutions (and possibly unimodal uncertainties). However, when ambiguities occur, the single estimation can be polluted by wrong information as discussed in Sec. 1.2, which can result in bad decisions or even failures in the active SLAM. Even though several nonparametric or multi-hypothesis solutions have been proposed to handle ambiguities explicitly and output multimodal solutions when the ambiguities are temporarily unsolvable [78][28][80][42][38], to the best of our knowledge, no existing active SLAM algorithm makes use of both multimodal state and map estimations to achieve better robustness. Therefore, we introduce the *ambiguity-aware active SLAM* framework that takes the multi-hypothesis state and map estimates from a MH-SLAM system (see Sec. 1.4) as inputs and makes single decisions online based on them, which is expected to improve the overall robustness of the entire robotic system.

Similar to other active SLAM solutions, ambiguity-aware active SLAM consists of two main modules: *exploration* and *active loop closing*. In addition, a *path planning* module is implemented to generate the actual motion command for each of them. All these three modules are designed to take multi-hypothesis state and map estimates into account explicitly, and therefore are computationally efficient and can run in real-time on a CPU with reasonable constraints. We evaluate our ambiguity-aware active SLAM approach intensively in simulations and apply it in an *assistive mapping system* that can guide a human user to explore and map indoor environments with hand-held sensors through instructions on the monitor. The experiments show that the proposed ambiguity-aware active SLAM framework can achieve much better robustness to ambiguities compared to a similar active SLAM framework that only handles single hypothesis, and the developed system also shows a good example of applying MH-iSAM2 to solve active SLAM problems.

1.6 Contributions

The key contributions of this thesis are summarized as follows:

- Developing MH-iSAM2, a robust back-end optimizer that models the ambiguous input measurements as multi-hypothesis factors, tracks the hypotheses of each state explicitly, and prunes the unlikely hypotheses based on all available information.
- Developing an ambiguity-aware front-end based on the fast RGB-D-inertial odometry (FRIO) algorithm, which detects and models ambiguities by checking the joint estimation results using the RGB-D and inertial data.
- Developing a keyframe-based MH-SLAM solution that utilizes submaps and planes for ambiguity-aware robust indoor dense 3D reconstruction, which can operate in real-time on a CPU.
- Introducing an ambiguity-aware active SLAM framework that makes use of the multi-hypothesis state and map estimations from a MH-SLAM system for better robustness.
- Applying the proposed ambiguity-aware active SLAM framework in an assistive mapping system that guides a human user to explore and map an indoor environment online, which also runs in real-time on a CPU.

1.7 Outline of the Thesis

In **Chapter** 2 we first review the literature on ambiguity-aware solutions to passive and active SLAM problems, and then go through other works that are relevant to the approach proposed in this thesis, including state estimation using visual and inertial sensors, SLAM with high-level features, and various offline and online dense 3D SLAM approaches.

In **Chapter** 3 we introduce the theoretical background of MH-iSAM2 and its implementation. We also test MH-iSAM2 on simulated datasets to show its robustness, accuracy, and efficiency.

In **Chapter** 4 we introduce the state estimation component used in our ambiguity-aware passive and active SLAM systems: a fast RGB-D-inertial odometry (FRIO) algorithm. Its performance is shown through real-world experiments.

In **Chapter** 5, we introduce the mapping component implemented for our ambiguity-aware passive and active SLAM systems: a dense 3D reconstruction framework that utilizes submaps and planes for efficient large-scale mapping. Intermediate results from integrating FRIO and iSAM2 into this mapping framework demonstrate the accuracy and efficiency of our approach.

In **Chapter** 6, we introduce how to integrate MH-iSAM2 (see Ch. 3), FRIO (see Ch. 4), and the submap and plane-based mapping components (see Ch. 5) together to realize an ambiguity-aware MH-SLAM system. Real-world experiments are conducted to support the proposed solution.

In **Chapter** 7, we introduce the ambiguity-aware active SLAM framework that is based on a MH-SLAM system (see Ch. 6), and utilize both simulation and real world experimental results to evaluate the proposed solution.

In **Chapter** 8, we conclude this thesis and point out future research directions to keep improving the robustness of SLAM solutions.

Chapter 2

Related Works

2.1 Ambiguity-aware SLAM

As more SLAM systems being developed, the robustness of SLAM solutions has been one of the main research focuses in this field [8]. As for SLAM solutions that specifically aim at handling or even solving ambiguities, some of them focus on solving the ambiguities immediately and preserve unimodal measurements in the front-end, which are usually integrated with a conventional back-end optimizer to solve the entire SLAM problem. Others take multimodal measurements as inputs and output either one or multiple possible solutions from the back-end optimizers. We will compare all these methods against each other as well as our proposed approach in this section.

2.1.1 Front-end Methods

The most commonly used front-end solution to handle ambiguity problems is *random sample consensus (RANSAC)* [24], which can estimate the model from a set of data without being influenced by the outliers. However, it requires more iterations to find the solution when the outlier ratio is larger, so it cannot work efficiently on measurements with multimodal ambiguities. Moreover, since RANSAC is usually applied in finding one solution only, if multiple modes all seem likely, it might select a wrong mode instead of the correct mode.

More recently, *joint compatibility branch and bound (JCBB)* [83] and various of its extensions [89][70] have been proposed to deal with the discrete ambiguities in SLAM problems through measuring the joint compatibility of a subset of association hypotheses and rejecting spurious matches (see Fig. 2.1). There are other font-end methods that make use of prior information to remove outliers for SLAM in dynamic environments [123]. However, none of these methods can guarantee perfect outlier rejection, especially when the observed information is insufficient. As a result, a possible solution is to also handle the ambiguities in the back-end as proposed in Sec. 1.3.

2.1.2 Back-end Methods

Considering the fact that not all of the ambiguities can be resolved in the front-ends of SLAM systems, various back-end approaches have been proposed. As a specific type of ambiguity that is

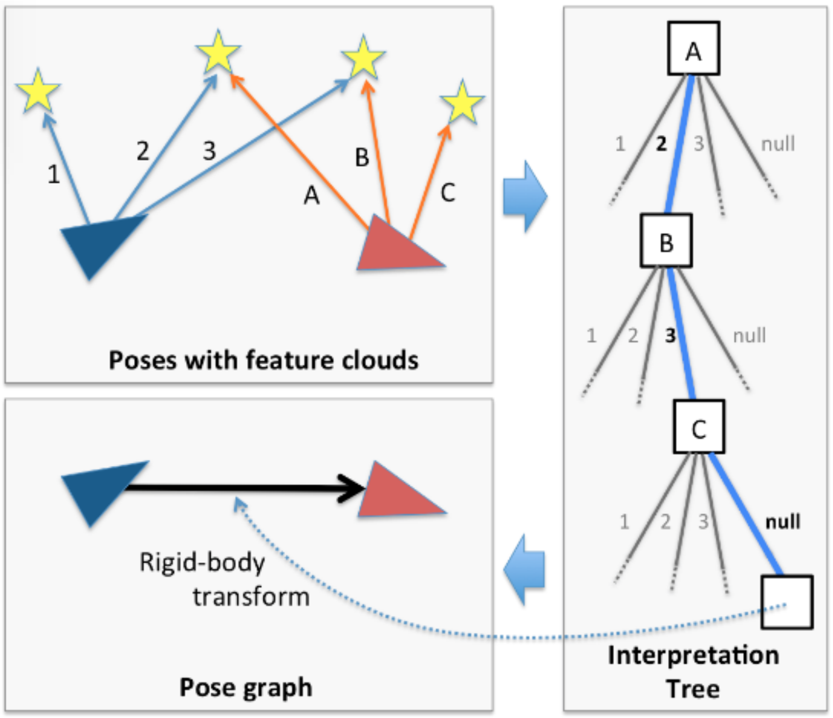


Figure 2.1: A figure from [89] describes the framework of JCBB, which uses an interpretation tree to explore the combinations of data association.

commonly encountered in SLAM, loop closing ambiguity and its solutions have been intensively discussed [62][4][127][66]. For other types of ambiguity sources or even more general solutions, there are also various existing methods. FastSLAM [78] can handle unknown data association though sampling particles over all possibilities, and [80][28] apply nonparametric methods to deal with ambiguities in data association or loop closing. These approaches in theory model all possibilities, but can only approximate the best solution(s) instead of solving for the exact values. On the other hand, [42] tracks multiple possible exact solutions and prunes the unlikely hypotheses incrementally. However, it requires additional batch steps for relinearization, and can be improved for SLAM applications, such as tracking the hypothesis of each variable more efficiently (see Sec. 3.4.1).

There are other back-end solutions that change the topology of the underlying graph [119][120], vary the distribution parameters [101] through self-tuning, scale the covariance of each measurement based on its chi-squared error [1], or replace the conventional sum-mixture of Gaussians with max-mixture [90] during optimization with ambiguity. Adding a step before conventional optimizers to choose the best hypothesis [102] is also another approach to solve the ambiguities. However, none of them can model more than one mode in the final output. As a result, when unsolvable ambiguities occur, they cannot track all the possible solutions at once.

In this thesis, the proposed back-end optimizer MH-iSAM2 can output multiple possible solutions for each variable according to the ambiguous inputs. It is developed based on iSAM2 [51] instead of other open source solvers [61][2][49] because iSAM2 allows online nonlinear incremental updates for efficiency. Most of the advantages of iSAM2 are still preserved in MH-iSAM2, which is further discussed in Ch. 3.

2.2 Ambiguity-aware Active SLAM

Built upon SLAM systems, active SLAM algorithms usually aim at exploring as much space (within a desired area is there is one) as possible while maintaining low uncertainty of the estimated poses and map. While *partially-observable Markov decision process (POMDP)* [114][47] offers a theoretical foundation to solve active SLAM problems considering all possible scenarios with insufficient information including ambiguous observations and more, finding an exact solution is computationally intractable. As a result, most existing active SLAM solutions do not handle ambiguity problems at all. Instead, they focus on solving exploration efficiently, such as *frontier*-based exploration [135][124] and *next best view (NBV)* planners [7][94] using *rapidly-exploring random tree (RRT)* [63][64], or finding suitable places for revisiting [116][72] that enable loop closing to correct the accumulated drift in the states and maps.

To take ambiguous information into account for decision making or path planning in real-world tasks, robust algorithms based on *belief space planning (BSP)* [46][3][98][96][97] and other methods [108] have been developed. However, none of them considers the coexistence of multi-hypothesis state and map estimations. More recently, a robust exploration algorithm [129] has been developed based on *multiple hypothesis JCBB* [83], which handles the multiple hypotheses of the feature-based map resulting from uncertain data associations and decides the future path based on the weighted combination of all the hypotheses. Another systematic solution [100] focused on robust homing considers only two hypotheses: is the up-to-date 2D map consistent or not. However, to the best of our knowledge, there is no existing work that conducts active SLAM based on multi-hypothesis dense 3D maps. As a result, we develop the ambiguity-aware active SLAM framework (see Ch. 7) that utilizes the multi-hypothesis state and map estimations from a MH-SLAM system. This new ambiguity-aware active SLAM framework is expected to plan reasonable motion even when unsolvable ambiguities occur during the SLAM process, and is regarded as a new step towards full autonomy especially in improving the robustness of the entire robotic system.

2.3 Visual Odometry

An algorithm that estimates the motion of a visual sensor based on its own data is called *visual odometry (VO)*, or *tracking* in the computer vision literature. Since visual sensors are widely used in SLAM systems due to their capability of offering rich information of the surrounding environment, various VO algorithms have been developed to achieve fast, robust, and accurate odometry estimation for SLAM or other related tasks, which are introduced in this section.

2.3.1 Indirect Methods

Obtaining accurate pose estimates of each sensor frame is essential to 3D SLAM algorithms. Feature point based methods in either VO or SLAM field adopt an *indirect* framework to estimate the poses or states of the camera [115][59][16][82], which consists of three common steps: feature point extraction, data association, and optimization. In the feature point extraction step, descriptors such as SIFT [71], SURF [5], FAST [105], BRIEF [9], or ORB [106] are applied to extract feature points from each input image. Then, points across images are associated with each other based on the similarity of their descriptors. Finally, all these measurements are used to optimize the poses of the sensor or even the corresponding landmark points under SLAM frameworks. In addition to the great success on image data, indirect methods are also applied to other types of sensor data such as RGB-D sensor [41] or sonar [132] due to its generality. However, all existing indirect algorithms can fail if insufficient features are extracted (e.g.: a blank wall without any feature), or wrong data association occurs (e.g.: repeated patterns that results in similar features).

2.3.2 Direct Methods

On the other hand, *direct* methods that optimize directly on pixel intensities or point distances without feature extraction might be able to recover accurate pose estimation when indirect methods fail. For example, in Fig. 2.2 indirect methods will fail because there is only one feature point in the scene, which is insufficient for pose estimation. However, direct methods can still estimate the pose correctly, such as semi-dense VO [21] and dense RGB-D odometry [117, 55] that minimize the photometric error of the reprojected pixels, or *iterative closest point (ICP)* [6] that minimize the geometric error between point pairs from two dense pointclouds.

To achieve high accuracy and better robustness, jointly optimizing the geometric and photometric costs is a well-known approach [133]. However, due to the expensive calculations over all the points and pixels, parallel computing using GPU is usually required for real-time applications. As a result, we develop a fast dense RGB-D odometry algorithm in Ch. 4 that jointly optimizes the geometric and photometric costs but can still run faster than real-time on a CPU.

2.4 Visual-inertial Fusion for State Estimation

V*isual-inertial odometry (VIO)* algorithms are widely used to estimate the states of mobile robots, which are usually extensions of existing VO methods that utilize the complementary nature of visual and inertial sensors for better state estimation. To be more specific, the time-varying biases from the raw inertial data can be corrected by the visual measurements, and the high frequency inertial measurements can offer good initialization or improve data association for VO algorithms. Based on the optimization framework, VIO algorithms can be classified into two categories as described in Sec. 2.4.1 and Sec. 2.4.2. In addition, we will introduce the *IMU preintegration* method [26] in Sec. 2.4.3 in detail since it will be applied in this thesis.

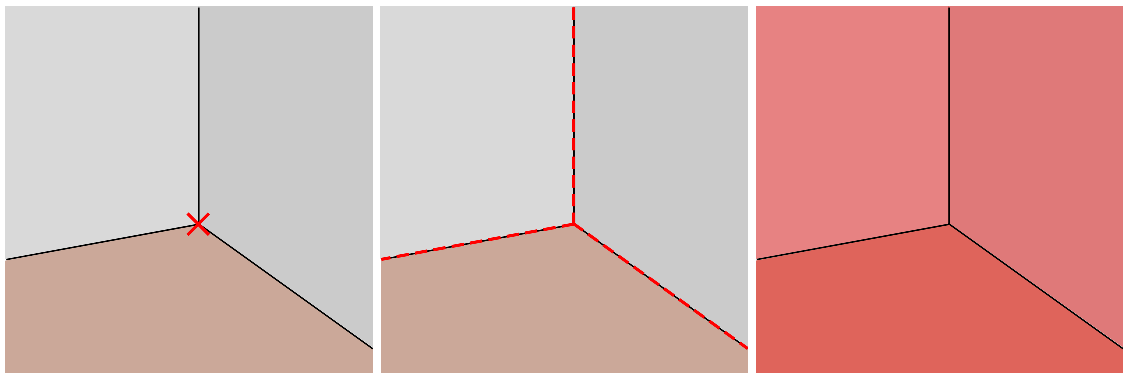

(a) Feature point-based (indirect) (b) Semi-dense (direct) (c) ICP (direct)

Figure 2.2: An example situation that direct methods can survive while indirect methods fail. To fully constrain a 3D pose during odometry estimation: (a) feature point-base methods (indirect) require three or more feature points, but only one (red cross) is observed; (b) semi-dense methods (direct) require high gradient pixels not lying on one single straight line, which is satisfied (three red dotted lines); (c) ICP (direct) requires three or more planes with their normal vectors not lying on the same plane, which is satisfied (three red planes) as well. It is worth noting that indirect methods can be extended to use other type of features such as lines or planes (see Sec. 2.5), which could achieve similar performance to the direct semi-dense method in (b) or (c) respectively in this example. There are also *semi-direct* methods [27][32]) that conduct direct operations around extracted features only in order to achieve robustness, accuracy, and efficiency all at once.

2.4.1 Loosely-coupled Methods

Various studies have focused on fusing inertial measurements with VO methods in recent years. A simple way to fuse them is called *loosely-coupled* [60][130][74], which optimizes odometry using the 6-DoF output from the VO methods instead of the raw visual measurements together with the inertial measurements. Even though loosely-coupled approaches might sacrifice some accuracy, they usually allow faster update to the current state and cheaper computation to correct the biases globally.

2.4.2 Tightly-coupled Methods

In contrast, *tightly-coupled* methods [52, 69, 81] use raw visual measurements and the inertial measurements together in each iteration of the nonlinear optimization, which can achieve more accurate results but is usually too costly for global optimization to correct the biases.

An alternative solution introduced in visual-inertial direct SLAM [13] tightly couples the inertial measurements and the direct semi-dense VO constraints for each pose estimation only, and marginalizes the IMU and VO measurements into pose-to-pose constraints in a pose graph. Even though this method can estimate the relative motion between frames more accurately, it does not allow the later observations to update the IMU biases for global optimization.

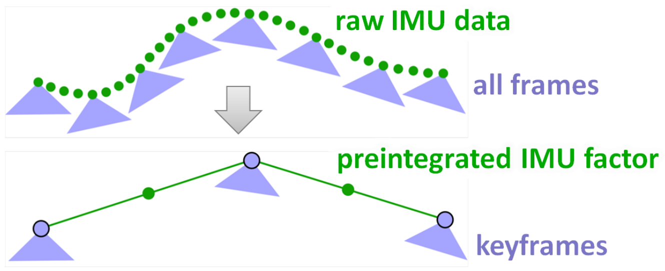


Figure 2.3: The concept of IMU preintegration. Raw IMU data within a predefined time interval (usually between two consecutive keyframes) are preintegrated into a factor, which allows linear updates of the estimated biases in a global optimization framework.

2.4.3 IMU Preintegration

Recently, IMU preintegration [26] has been developed to offer an efficient solution to either loosely or tightly couple the IMU measurements with visual constraints. Comparing to other methods, the key advantage of IMU preintegration is that the inertial measurements including angular velocities and accelerations are locally integrated in a specific form, which allows the biases to be updated by later observations. The resulting *preintegrated IMU factors* can be added into a factor graph (see Fig. 2.3) and jointly optimized with other measurements to solve the SLAM problems.

IMU preintegration follows the common modeling of robot states with inertial measurements. Each *IMU state* $\mathbf{x}_t$ at time t contains the pose, velocity, and bias terms for both gyroscope and accelerometer, which can be represented as a 15-vector

$$\mathbf{x}_t = \begin{bmatrix} \boldsymbol{\xi}_t^\top & \mathbf{v}_t^\top & \mathbf{b}_\omega^\top & \mathbf{b}_a^\top \end{bmatrix}^\top, \tag{2.1}$$

where $\boldsymbol{\xi}_t$ represents the 6-DoF pose of the IMU, which can also be represented as a rotation matrix $\mathtt{R}_t$ and a translation vector $\mathbf{p}_t$, and $\mathbf{v}_t$ is the 3-DoF velocity. $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are the 3-DoF bias terms for the gyroscope and accelerometer respectively, which are assumed to be static over each preintegration interval (and can change between intervals). Assuming that the raw angular velocity $\boldsymbol{\omega}_t$ and acceleration $\mathbf{a}_t$ arrives every Δt seconds, we can define the *preintegrated rotation*, *velocity*, and *translation* of the preintegrated IMU factor between the two consecutive reference frames at time t and $t' = t + m\Delta t$ respectively as

$$\Delta \mathtt{R}_{t'}^t = \prod_{k=t}^{t'} \operatorname{Exp}\left(\left(\boldsymbol{\omega}_k - \mathbf{b}_\omega\right)\Delta t\right), \tag{2.2}$$

$$\Delta \mathbf{v}_{t'}^t = \sum_{k=t}^{t'} \Delta \mathtt{R}_k^t \left(\mathbf{a}_k - \mathbf{b}_a\right)\Delta t, \tag{2.3}$$

$$\Delta \mathbf{p}_{t'}^t = \sum_{k=t}^{t'} \left[\Delta \mathbf{v}_k^t \Delta t + \frac{1}{2}\Delta \mathtt{R}_k^t \left(\mathbf{a}_k - \mathbf{b}_a\right)\Delta t^2\right], \tag{2.4}$$

and the error functions of their corresponding factors in $\mathcal{G}$ are

$$e_{\Delta \mathtt{R}^t_{t'}} = \mathrm{Log}\left\{\left[\Delta \mathtt{R}^t_{t'} \mathrm{Exp}\left(J_{\Delta \mathtt{R}^t_{t'}}\begin{bmatrix}\delta \mathbf{b}_\omega \\ \delta \mathbf{b}_a\end{bmatrix}\right)\right]^\top \mathtt{R}_t^\top \mathtt{R}_{t'}\right\}, \tag{2.5}$$

$$e_{\Delta \mathbf{v}^t_{t'}} = \mathtt{R}_t^\top(\mathbf{v}_{t'} - \mathbf{v}_t - \mathbf{g}m\Delta t) - \Delta \mathbf{v}^t_{t'} - J_{\Delta \mathbf{v}^t_{t'}}\begin{bmatrix}\delta \mathbf{b}_\omega \\ \delta \mathbf{b}_a\end{bmatrix}, \tag{2.6}$$

$$\begin{aligned} e_{\Delta \mathbf{p}^t_{t'}} = \mathtt{R}_t^\top &\left(\mathbf{p}_{t'} - \mathbf{p}_t - \mathbf{v}_t m\Delta t_m - \frac{1}{2}\mathbf{g}m^2\Delta t^2\right) \\ &- \Delta \mathbf{p}^t_{t'} - J_{\Delta \mathbf{p}^t_{t'}}\begin{bmatrix}\delta \mathbf{b}_\omega \\ \delta \mathbf{b}_a\end{bmatrix}, \end{aligned} \tag{2.7}$$

where $J_{\Delta \mathtt{R}^t_{t'}}$, $J_{\Delta \mathbf{v}^t_{t'}}$, and $J_{\Delta \mathbf{p}^t_{t'}}$ are the Jacobians of $\Delta \mathtt{R}^t_{t'}$, $\Delta \mathbf{v}^t_{t'}$, and $\Delta \mathbf{p}^t_{t'}$ with respect to the bias vector $\begin{bmatrix}\mathbf{b}_\omega^\top & \mathbf{b}_a^\top\end{bmatrix}^\top$ respectively, which allows updating the bias terms linearly. The $\mathrm{Exp}(.)$ and $\mathrm{Log}(.)$ functions are the exponential and log maps that transform the rotation in 3D between $SO(3)$ and its minimal representation in $\mathbb{R}^3$. $\mathbf{g}$ is the constant gravity vector, which should be defined in advance.

Similar to other applications of IMU preintegration, we combine the preintegrated IMU factors with other sensor measurements in a SLAM optimization framework, which corrects the drifting IMU biases and results in consistent state estimations. Moreover, we develop an ambiguity detection function for both our passive and active SLAM approaches based on the IMU preintegration framework. More details of our approach can be found in Sec. 4.7 and Ch. 6. As for more details about IMU preintegration, please see [26].

2.5 SLAM with High-level Features

Other than point features, SLAM algorithms that use high-level features, such as lines, planes, or even semantic objects, have been developed for various reasons and applications. In general, a high-level feature preserves more information than a single point, which can improve the robustness or accuracy of SLAM. However, more assumptions have to be made when using these advanced features, so the resulting algorithms are usually less general. For example, SLAM with lines usually works well in artificial environments but not in nature because the former usually contains lots of straight lines while the latter does not. As a result, using suitable types of features based on the target environments is crucial for SLAM algorithms. In this section, we will introduce different types of features used in existing SLAM algorithms, discuss their pros and cons, and explain our choice of feature for indoor dense 3D reconstruction in this thesis.

2.5.1 Geometric Features

SLAM using line features has been studied for more than a decade. Various representations of lines have been applied, such as two end points [113][104], orthonormal representation [138], or Plücker coordinates [68]. However, since lines do not model surfaces explicitly, none of these methods can recover dense models of the environments.

On the contrary, as another commonly observed feature in general indoor environments, planar surfaces have been used for mapping tasks in earlier works [95][126][67][111], and are further used for state estimation or even SLAM in recent years. Point-plane SLAM [122] combines three point/plane primitives to fully constrain camera poses and generate a plane-based 3D model. CPA-SLAM [73] requires GPU acceleration for tracking towards both keyframes and plane models in real-time. Although it applies a soft labeling technique to reduce the effect of incorrect plane segmentation on the pose estimate, the plane model can still be wrong in the global optimization.

Comparing to line features, planes preserve much more surface information, which enables more applications for the output map, such as visualization of the dense models or path planning. As a result, we adopt plane features in the SLAM system in this work. We extract planes from smoothed depth maps that are noticeably less noisy than raw data, which results in more accurate plane segmentation (see Sec. 5.2). Then, we model each plane in a quaternion-based minimal representation as described in [48], and jointly optimize all the planar measurements with other measurements in a SLAM framework (see Sec. 5.7), which is expected to greatly enhance the robustness and accuracy of the SLAM system. Moreover, the implicit free, occupied, and unknown space information in the modeled planes will be extracted and used in our proposed active SLAM algorithm (see Ch. 7).

2.5.2 Structural Constraints

Structural constraints, such as orthogonality and parallelism between lines or planes, are expected to further correct the drift or distortion in the reconstructions of man-made environments. [88] and [87] demonstrate the advantages of applying these structural constraints on 2D and 3D mapping respectively. A similar but more limited concept called *Manhattan world assumption* is also applied in [137][25][99] to achieve better mapping results through estimating the global orientation of each frame. In this thesis, we will also exploit the structural constraints of the planes to further improve reconstruction results (see Sec. 5.6.2), especially when there is no loop closure constraint to correct the accumulated error in the SLAM system.

2.5.3 Objects And Semantics

As modern machine learning methods greatly enhance the robustness and accuracy of semantic segmentation and object recognition, more studies on SLAM with objects or semantics have been done in recent years. SemanticFusion [76] fuses the semantic labels of pixels from different images into a global 3D pointcloud, which improves the accuracy of the semantic labeling but not tracking or mapping. To use the information of objects in both tracking and mapping, SLAM++ [110] and Fusion++ [75] both register each input frame to the models of each detected objects in the previous frames, and optimize the pose estimates of each frame and object in a global factor graph. Since semantic understanding is not our focus in this thesis, we will only detect floor planes for the proposed active SLAM algorithm (see Ch. 7).

2.6 Dense 3D reconstructions

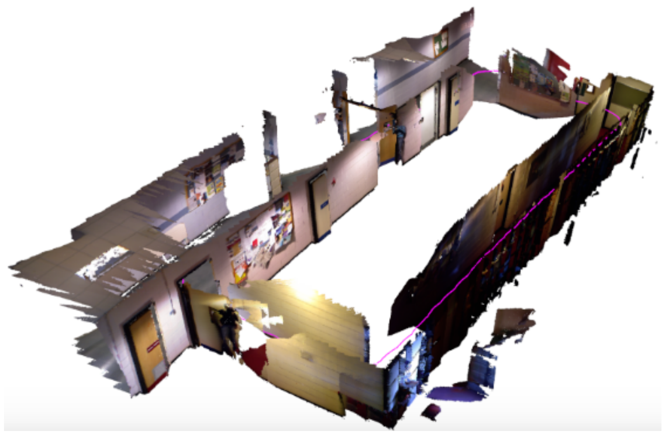
2.6.1 Offline Methods

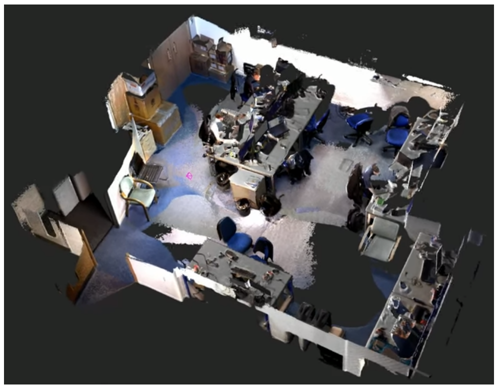
Dense 3D reconstruction has been a popular research topic for decades. Many offline methods have been developed [92][57][112], which usually require hours of computation to generate dense 3D models from numerous images. Existing industrial solutions such as Matterport [44] and Faro [23] scanners can generate very accurate 3D reconstructions through registration of static scans. However, these offline methods are not suitable for real-time applications, such as exploration and AR. As a result, we will focus on real-time solutions that can reconstruct the most up-to-date models based on the accumulated input data at any given time in this thesis.

2.6.2 Online Methods

Comparing to the offline methods, online methods usually aim at real-time performance and therefore require cheaper computation. DTAM [84] computes dense depth maps from a single moving camera in real-time using GPU, and REMODE [103] further estimates the depth in a probabilistic manner. Many other camera-based solutions [77][30][33] have been proposed in the last decade, which demonstrate the capability and applications of real-time dense 3D reconstruction.

Dense 3D models may also be generated using RGB-D sensors by fusing depth or even color data from multiple frames, a process that must be accelerated by a GPU in order to achieve real-time performance [85] [53] [133] [134][14] (see Fig. 2.4). In our mapping approach, we aim at using RGB-D data to reconstruct dense 3D models in real-time using CPU only. As a result, a computationally cheaper method for dense mapping is developed in Ch. 5.


(a) Kintinuous [133]


(b) ElasticFusion [134]

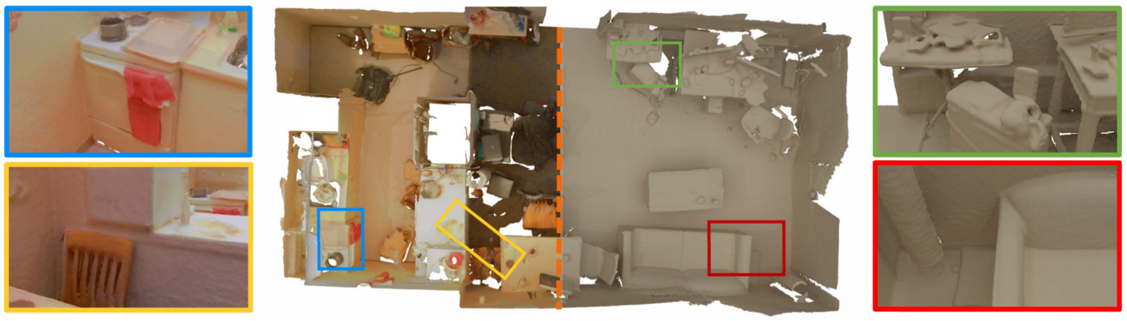

(c) BundleFusion [14]

Figure 2.4: Example outputs of existing real-time indoor dense 3D reconstruction algorithms.

Chapter 3

Multi-hypothesis Optimizer for Robust SLAM

3.1 SLAM Using Factor Graph

From the probabilistic point of view, SLAM can be modeled as a *maximum a posteriori (MAP)* estimation problem $\hat{\Theta} = \arg\max_{\Theta} P(\Theta|Z)$ that aims at solving all variabes $\theta_p \in \Theta$ given the set of all measurements $z_k \in Z$. Based on the *Bayes' rule* $P(\Theta|Z) = \frac{P(Z|\Theta)P(\Theta)}{P(Z)} \propto P(Z|\Theta)P(\Theta)$ and adding the assumption of no prior for any variable, the MAP problem becomes a *maximum likelihood estimation (MLE)* problem:

$$\hat{\Theta} = \arg\max_{\Theta} P(Z|\Theta) = \arg\max_{\Theta} \prod_{k} P(z_k|\Theta_k), \tag{3.1}$$

where $\Theta_k \subseteq \Theta$ is the subset of variables that directly affect z_k in the conditional probability $P(z_k|\Theta_k)$. $\hat{\Theta}$ is the set of solutions of all variables that maximize $P(Z|\Theta)$. Typical solutions to this MLE problem can be found in Appendix A.1.

In recent robotics studies, a *factor graph* [17] (see Fig. 3.1) is commonly used to represent the MLE problem of SLAM, which is a bipartite graph that consists of two types of nodes: factors F and variables Θ. Each factor $f_k \in F$ represents the conditional probability $P(z_k|\Theta_k)$ of an input measurement, and each unknown variable θ_p can be solved by a nonlinear optimizer [61][2][49][50] if every measurement z_k is sampled from a unimodal distribution (usually Gaussian distribution), which is a common assumption for most of the SLAM problems.

3.2 From Probability to Hypotheses

The assumption of unimodal distribution no longer holds for ambiguous measurements because one single mode cannot model all the possible cases. A simple extension is to use a *Gaussian mixture model (GMM)* to represent the multimodality of an ambiguous measurement z_r with

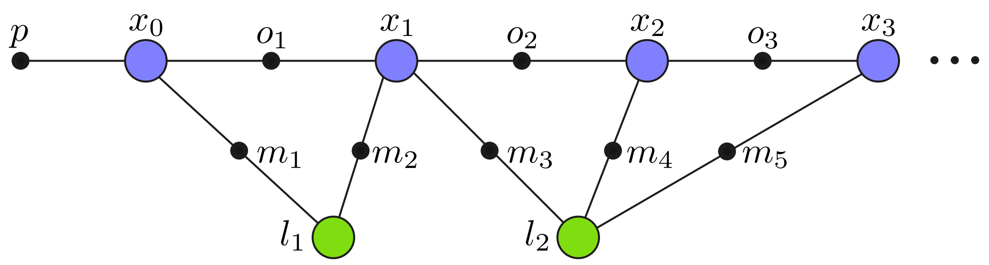


Figure 3.1: An example of a factor graph, which consists of variable nodes $\Theta = \{x, l\}$ (colored dots) and factor nodes $F = \{p, o, m\}$ (black dots).

multiple modes $z_{r(i)}$, which can be written as

$$P_{\mathrm{M}}(z_r|\Theta_r) = \sum_{i=0}^{m_r} w_{r(i)} \mathcal{N}(\mu_{r(i)}, \Sigma_{r(i)}) = \sum_{i=0}^{m_r} w_{r(i)} P\left(z_{r(i)}|\Theta_r\right), \tag{3.2}$$

where m_r is the number of modes in z_r, and $w_{r(i)}$ is a weighting for each $\mathcal{N}\left(\mu_{r(i)}, \Sigma_{r(i)}\right)$ which satisfies $\sum_{i=0}^{m_r} w_{r(i)} = 1$. The subscript "$_{(i)}$" indicates each mode i of z_r. Assuming that every z_r is independent of all others, we can rewrite the MLE problem in Eq. A.1 as

$$\hat{\Theta} = \arg\max_{\Theta} \left[\prod_s P\left(z_s|\Theta_s\right)\right]\left[\prod_r P_{\mathrm{M}}\left(z_r|\Theta_r\right)\right] \tag{3.3}$$

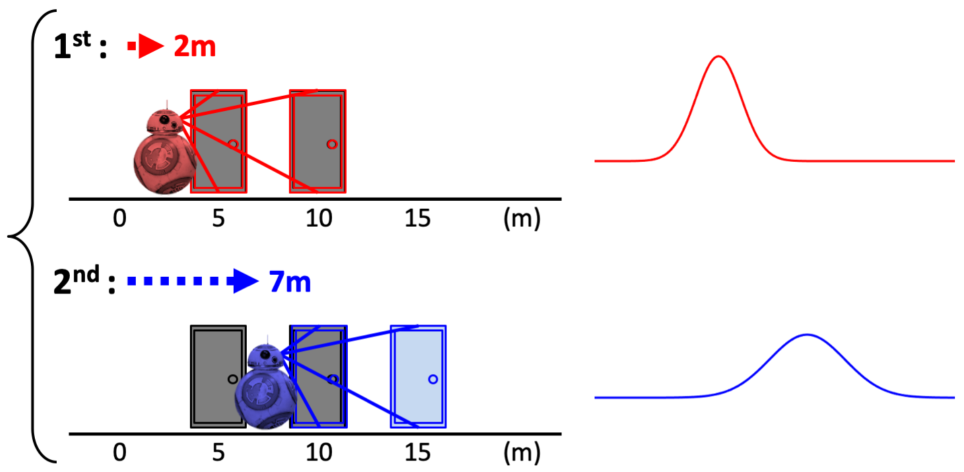
to represent SLAM problems with ambiguity, where each index s corresponds to a single-mode measurement z_s while index r corresponds to a multi-mode measurement z_r. Even though we can still solve this MLE problem and get a single estimation $\hat{\Theta}$ that corresponds to one of the highest peaks of the resulting GMM, the information of all other peaks that result from other combinations of modes is lost. To preserve all the combinations of modes, we relax the MLE problem in Eq. 3.3 to a *multi-hypothesis MLE problem (MH-MLE)*:

$$\hat{\mathbf{\Theta}}_{\mathrm{M}} = \left\{\hat{\Theta}_{[\mathbf{i}]} | \mathbf{i} \in \mathbb{N}^t\right\}, \tag{3.4}$$

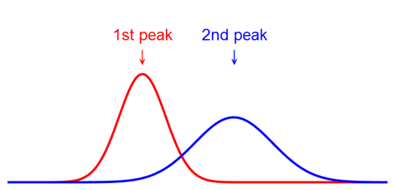
$$\hat{\Theta}_{[\mathbf{i}]} = \arg\max_{\Theta} \left[\prod_s P(z_s|\Theta_s)\right]\left[\prod_{r=1}^{t} w_{r(i_r)} P\left(z_{r(i_r)}|\Theta_r\right)\right], \tag{3.5}$$

where t is the total number of multi-mode measurements z_r. Index $\mathbf{i}$ is a t dimensional vector whose r-th element i_r indicates the choice of mode of z_r, and the entire $\mathbf{i}$ vector represents one *overall hypothesis* $h_j^{\{t\}}$ (j is a scalar index that 1-to-1 associates with $\mathbf{i}$), which is one of the combinations of all modes (we use "mode" for inputs and "hypothesis" for outputs in this thesis). Notice that $w_{r(i_r)}$ can be dropped without affecting the solution $\hat{\Theta}_{[\mathbf{i}]}$ at all, which turns Eq. 3.5 into a MLE problem again. Therefore, the MH-MLE problem is actually a combination of multiple MLE problems, each corresponds to one $h_j^{\{t\}}$ and one solution $\hat{\Theta}_{[\mathbf{i}]}$. The entire set $\hat{\mathbf{\Theta}}_{\mathrm{M}}$ covers all the combinations of all modes of all z_r.

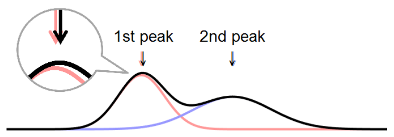
Comparing to other approaches discussed in Sec. 2.1.2, solving the ambiguity problem through the proposed MH-MLE approach is better in several aspects. Let us take the same



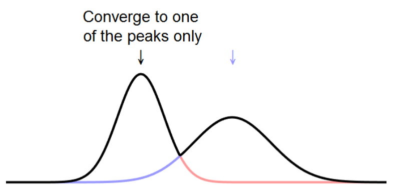
(a) Two hypotheses (left) and their corresponding distributions (right)



(b) MH-MLE (ours)



(c) Sum-mixture



(d) Max-mixture

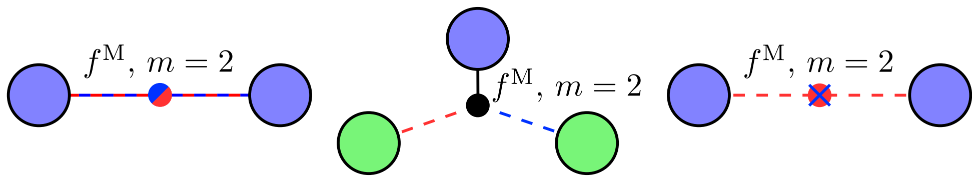
Figure 3.2: Comparison of different approaches using the same example in Fig. 1.1. As each hypothesis is represented as a unimodal distribution with only one peak in (a), we can recover the two peaks (red and blue arrows) separately following our MH-MLE approach in (b). In (c), we can recover two peaks (black arrows), but both of them are off from their original solutions (light blue and red arrows), and the correspondences of the peaks and their original input distributions are no longer tracked. In (d), we can solve for only one original peak (black arrow) at convergence. See Sec. 2.1.2 for more information about sum-mixture and max-mixture methods.

example in Fig. 1.1, where each hypothesis at $t+1$ and their corresponding unimodal distributions are shown in Fig. 3.2-a. Our MH-MLE approach allows us to solve for the exact peaks of each individual distribution (see Fig. 3.2-b). However, we cannot recover the exact peaks using those nonparametric methods due to their sum-mixture formulation (see Fig. 3.2-c). The solutions based on max-mixture of distributions can only recover one exact peak instead of all of them (see Fig. 3.2-d).

3.3 Multi-hypothesis Factor Graph

3.3.1 Basic Structure

An MH-MLE problem can be represented in a *multi-hypothesis factor graph (MHFG)*, which is an extension of the original factor graph [17] and can be converted into a multi-hypothesis Bayes


f^{M}, $m = 2$
f^{M}, $m = 2$
f^{M}, $m = 2$

(a) Multi-measurement (b) Multi-association (c) Boolean

Figure 3.3: The three types of MMFs f^{M}. Red and blue show the two modes ($m = 2$). Purple and green nodes are poses and landmarks respectively.

tree (MHBT) and solved efficiently (see Sec. 3.5). An MHFG consists of *single-mode factors (SMF), multi-mode factors (MMF)*, and *multi-hypothesis variables (MHV)*. An SMF corresponds to one $P(z_s|\Theta_s)$ in Eq. 3.5, which is the same as a factor in the original factor graph. An MMF models each mode of an ambiguous measurement as a individual Gaussian distribution as described in Eq. 3.5. Three types of MMFs are defined in Sec. 3.3.2). An MHV $\theta_p \in \hat{\Theta}_{\mathrm{M}}$ can represent its multiple values from each hypothesis in an efficient way (see Sec. 3.3.3 and 3.4.1).

3.3.2 Multi-mode Factors (MMF)

We define three types of multi-mode factors (MMF) f^{M}, each with m modes ($m > 1$), to model most kinds of discrete ambiguities that cannot be solved by front-ends. Type #1 is the *multi-measurement factor*, which consists of m different measurements that are all connected among the same MHVs (see Fig. 3.3-a). For example, two different visual odometry (VO) estimates can be loosely-coupled in one pose graph for better accuracy. However, when the two estimates are very different, it is very likely that one of them is an outlier and should not be considered when computing the optimal result. In this case, we can model them as a type #1 MMF with two modes.

Type #2 is the *multi-association factor*, which contains only one measurement but is connected among m MHVs that are the same type and at least one other MHV (see Fig. 3.3-b). For example, when a newly observed feature point is very similar to more than one landmark both geometrically and in appearance, the front-end again cannot tell which is the accurate association without additional information.

Type #3 is the *Boolean factor*, which represents whether a factor should exist or not ($m = 2$). One common application is to model each loop closure candidate in a loop closing ambiguity problem [62][4][127] (see Fig. 3.3-c).

3.3.3 Multi-hypothesis Variables (MHV)

A multi-hypothesis variable (MHV) θ_p contains multiple estimates for one variable, each corresponds to one hypothesis of θ_p. The hypotheses of θ_p are determined by all the MMFs that affect it, which is hard to track since it depends on the topological structure of the MHFG. Therefore, we introduce the Hypo-tree data structure in Sec. 3.4 to simplify the hypothesis tracking process.

Another challenge is that there can be causal relationships among MMFs, e.g.: the previous

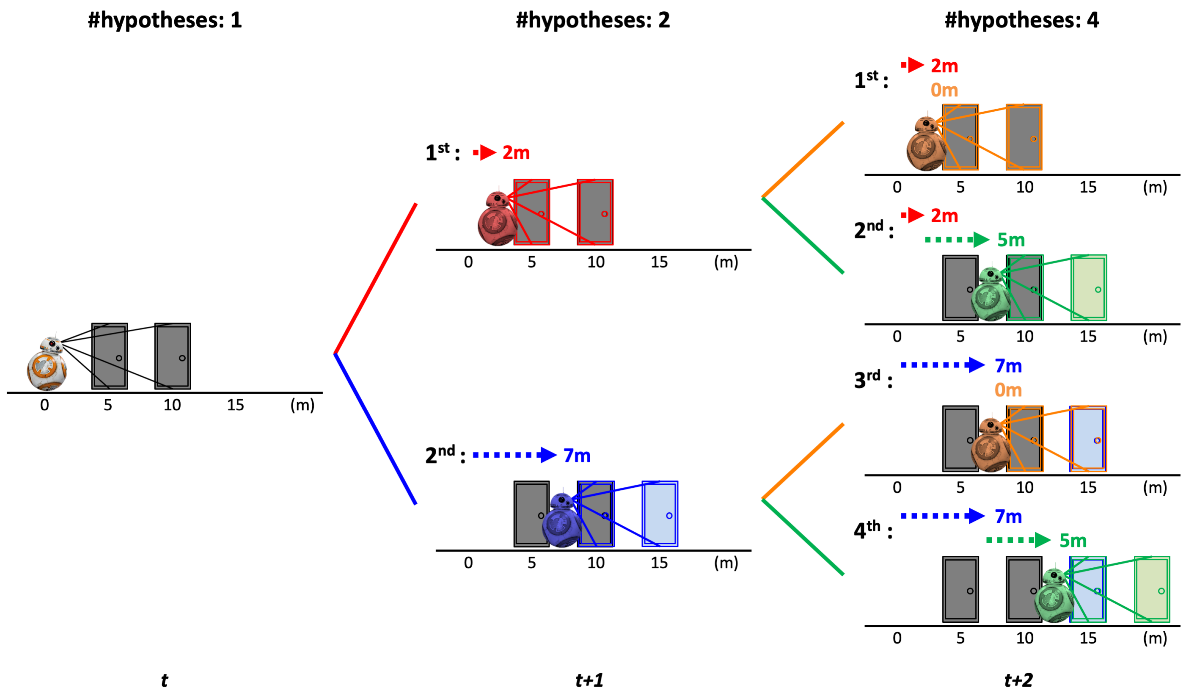


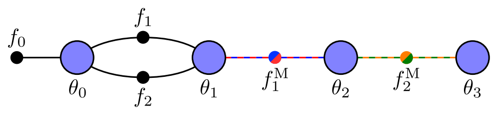
Figure 3.4: An extension of the example in Fig. 1.1 and Fig. 3.2 that shows how the growing of hypotheses affects both localization and mapping (the color of each of the newly added landmark door represent the mode that first observes it). At time $t+2$, a similar ambiguous observation with two modes is made, which results in the four overall hypotheses. Each overall hypothesis can be derived from one of the two local hypotheses at the previous time $t+1$ with a choice of mode in the newly observed ambiguity. As a result, each hypothesis has its own trajectory and map, which can be very expensive to track as the total number of hypotheses grows exponentially.

choice of closing a loop or not can affect a later data association, so the hypotheses of the affected θ_p are even harder to track. However, we can still assume that all of the MMFs are independent of each other as defined in MH-MLE without losing generality. Even though each θ_p might contain redundant hypotheses from impossible combinations of modes, it at least preserves all the possible hypotheses, and those redundancies can be removed later through hypotheses pruning (see Sec. 3.6).

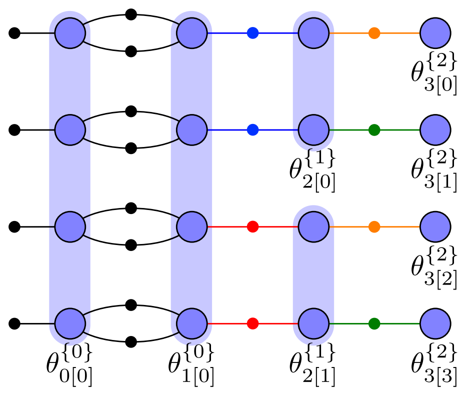
3.4 Hypotheses Tracking in Hypo-tree

3.4.1 Overall and Local Hypotheses

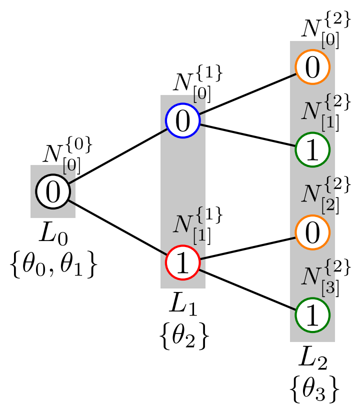
Because of the independence assumption (see Sec. 3.2), when multiple MMFs exist in one MHFG, the number n_t of the *overall hypotheses* $h^{\{t\}} = \left\{ h_j^{\{t\}} | 0 \leq j < n_t \right\}$ of the entire system is $n_t = \prod_{r=1}^{t} m_r$, where m_r is the number of modes of each MMF f_r^{M}, and t is the total number of MMFs. Even though n_t grows exponentially with t (see Fig. 3.4) and has to be pruned (see Sec.



(a)



(b)



(c)

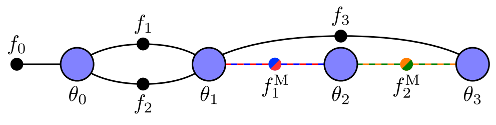
Figure 3.5: An example of hypotheses growing. (a) A MHFG that contains two MMFs (each with 2 modes: blue/red and orange/green) can be regarded as 4 individual factor graphs in (b). However, some of the MHVs (θ_0, θ_1, and θ_2) share the same values across different factor graphs (purple shadows), which implies that fewer hypotheses are required to model these MHVs. To be more specific, θ_0 and θ_1 are not affected by any MMF, and θ_2 is only affected by f_1^{M}. (c) We can associate a MHV θ_p with a Hypo-layer L_r in Hypo–tree to track its hypotheses (written as $\{\cdot\}$).

3.6) to maintain a tractable size, the number of *local hypotheses* $h^{\{r\}} = \left\{ h_j^{\{r\}} | 0 \leq j < n_r \right\}$ of each MHV θ_p can be less than n_t because a MHV might not be affected by all MMFs (see Fig. 3.5). As a result, we track $h^{\{r\}}$ of each θ_p instead of $h^{\{t\}}$ to improve efficiency.

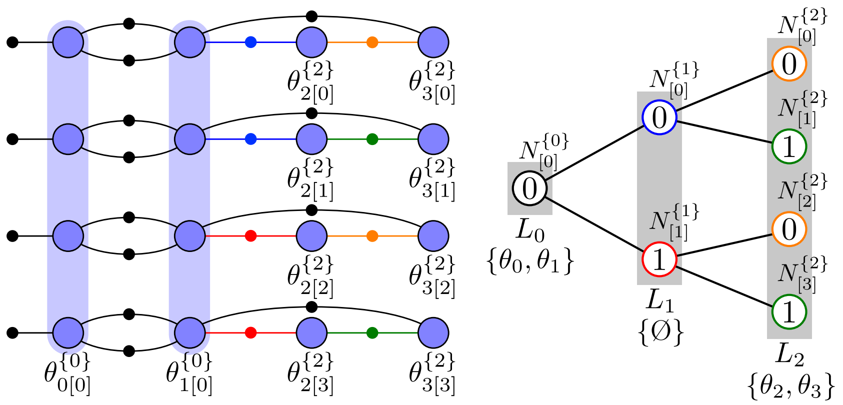
However, $h^{\{r\}}$ of each θ_p can change as more measurements (SMFs or MMFs) are added into the system. For example, as shown in Fig. 3.6-a, if a loop closure is added into the MHFG in Fig. 3.5-a, some of the θ_p begin to be affected by the MMFs that originally do not affect θ_p (see Fig. 3.6-b). Therefore, those θ_p have to expand their $h^{\{r\}}$ accordingly.

3.4.2 Construction of Hypo-tree

To handle both the growing of $h^{\{t\}}$ and the expansion of $h^{\{r\}}$ efficiently, we propose the Hypo-tree data structure (see Fig. 3.5-c and 3.6-c). It consists of several *Hypo-layers* L_r, each results from one MMF f_r^{M} following the temporal ordering $r = 0, ..., t$ and contains several *Hypo-nodes* $N_{[j]}^{\{r\}}$ ($j = 0, ..., n_r$) that represent local hypotheses $h_{[j]}^{\{r\}}$. Starting from L_0 that contains only one Hypo-node $N_{[0]}^{\{0\}}$, whenever a new MMF f_{t+1}^{M} is observed, a new Hypo-layer L_{t+1} will be created, and m_{t+1} new Hypo-nodes $N_{[j']}^{\{t+1\}}$ will be generated in L_{t+1} as the children of each $N_{[j]}^{\{t\}}$ in L_t. Therefore, the total number of Hypo-nodes in L_{t+1} is $n_{t+1} = n_t \cdot m_{t+1}$. Due to this incremental construction procedure, the structure of previous layers $L_0, \ldots, L_t$ never change.



(a)



(b) (c)

Figure 3.6: An example of hypotheses expansion. (a) Factors that link the current variable with an earlier one can result in hypotheses expansion of MHV, which can be observed by comparing (b) and Fig. 3.5-b. (c) In this case, we only have to update the association between Hypo-layers and MHVs without changing the structure of Hypo-tree.

3.4.3 Association and Correspondence

Every new MHV θ_p is associated with the latest Hypo-layer L_r (denoted as $\theta_p^{\{r\}}$) once added into the system. Also, each value of $\theta_p^{\{r\}}$ (denoted as $\theta_{p[j]}^{\{r\}}$) is associated with each Hypo-node $N_{[j]}^{\{r\}}$ (or local hypothesis $h_{[j]}^{\{r\}}$) in L_r, for $0 \leq j < n_r$. When the local hypotheses of θ_p have to be expanded (e.g. the example in Sec. 3.4.1), we only have to update the association of $\theta_p^{\{r\}}$ to a different Hypo-layer $L_{r'}$, which can be denoted as $\theta_p^{\{r\}} \rightarrow \theta_p^{\{r'\}}$, and expand the number of values in $\theta_p^{\{r'\}}$ accordingly.

Based on the association, searching for the corresponding values of local hypotheses between variables is simple. The $h_{[j^*]}^{\{r^*\}}$ of a MHV $\theta_{p^*}^{\{r^*\}}$ that is the ancestor of $h_{[j]}^{\{r\}}$ of another $\theta_p^{\{r\}}$ can be found by traversing from $N_{[j]}^{\{r\}}$ towards the root till reaching L_{r^*} ($r^* < r$). Or, the set of local hypotheses $\left\{h_{[j']}^{\{r'\}} | 0 \leq j' < n_{r'}\right\}$ of $\theta_{p'}^{\{r'\}}$ that are the descendants of $h_{[j]}^{\{r\}}$ of $\theta_p^{\{r\}}$ can be found through traversing from $N_{[j]}^{\{r\}}$ towards the leaves till reaching $L_{r'}$ ($r' > r$). Same method is applied to search for the hypotheses correspondences among other components in the inference process (see Sec. 3.5).

Comparing to the *levels* in a *hypothesis tree* described in [127], the Hypo-layers in a Hypo-tree track the hypotheses of each MHVs in addition. Moreover, in our approach the associations between MHVs and Hypo-layers can be changed when the hypotheses expansion occurs. Finally, it is worth noting that even though the Hypo-tree data structure is originated from the nature of growing of hypotheses as shown in Fig. 3.4, it is not only a representation of the growing of hypotheses but also serve as an efficient reference for the overall/local hypotheses of the computational components in the proposed multi-hypothesis optimization algorithm (see Sec. 3.5 and Sec. 3.6).

3.5 Inference in Multi-hypothesis Bayes Tree

3.5.1 Multi-hypothesis Bayes Tree (MHBT)

Multi-hypothesis Bayes tree (MHBT) is an extension of the original Bayes tree [51] (see Appendix A.2 for more details) that conducts efficient inference for a MHFG. A MHBT stores *multi-hypothesis conditional densities (MHCD)* $\gamma_q^{\{r\}}$ in each of its cliques C_q, and applies a multi-hypothesis inference process throughout all the cliques (see Fig. 3.7) to solve for all the MHVs.

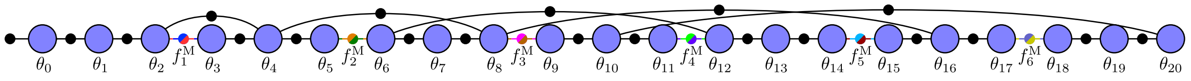
A MHBT can be constructed from a MHFG based on an ordering of all MHVs. In each clique C_q, the relevant SMFs, MMFs, and all of the *multi-hypothesis marginal densities (MHMD)* $\omega_{q^*}^{\{r^*\}}$ that are passed from C_q's child cliques C_{q^*} (if any) are combined into a *multi-hypothesis joint density (MHJD)* $\phi_q^{\{r\}}$. Then, $\phi_q^{\{r\}}$ is factorized into a MHCD $\gamma_q^{\{r\}}$ and a MHMD $\omega_q^{\{r\}}$. Finally, $\omega_q^{\{r\}}$ are passed to C_q's parent clique $C_{q'}$. Repeating this process from leaves to root completes one iteration of inference, and several iterations are needed before convergence for a nonlinear SLAM problem. Notice that each $\phi_q^{\{r\}}$, $\gamma_q^{\{r\}}$, and $\omega_q^{\{r\}}$ is associated with a Hypo-layer L_r for the search of hypotheses correspondences among them (see Sec. 3.4.3).

3.5.2 Variable Ordering

All MHVs are ordered based on the same CCOLAMD algorithm [15] applied in [51] to construct a MHBT from a MHFG. Since all hypotheses have to share the same ordering, all the edges that represent a mode in each multi-association factor are regarded as connected, and all Boolean factors are regarded as true (connected) in the ordering process.

3.5.3 Linearization

Each nonlinear factor f_s (a SMF) or f_r^{M} (a MMF) is linearized with respect to a linearization point of all the relevant MHVs Θ_s or Θ_r (see Sec. 3.2) if required (*fluid relinearization* [51] is applied). Consequently, each local hypothesis $h_{[j]}^{\{r'\}}$ of a *multi-hypothesis linearized factor (MHLF)* $l_s^{\{r'\}}$ (from f_s) or $l_r^{\mathrm{M}\{r'\}}$ (from f_r^{M}) is calculated by finding the correspondences among the local hypotheses of Θ_s or Θ_r, and also the corresponding modes of f_r^{M} for $l_r^{\mathrm{M}\{r'\}}$ only (see Fig. 3.7-b). Notice the associated Hypo-layer $L_{r'}$ of $l_r^{\mathrm{M}\{r'\}}$ might not be the same as L_r that results from f_r^{M} since some of its relevant MHVs θ_p might be affected by later MMFs (thus $r' \geq r$). Based on the Gaussian assumption in Eq. 3.5, each MHLF is a set of Jacobian matrices $\mathbf{A}^{\{r'\}} = \left\{A_{[j]}^{\{r'\}} | 0 \leq j < n_{r'}\right\}$, and each $A_{[j]}^{\{r'\}}$ contains the right-hand-side (RHS) vector as an additional


(a) A MHFG with 20 odometry factors (14 SMFs and 6 type #1 MMFs) and 5 loop closures

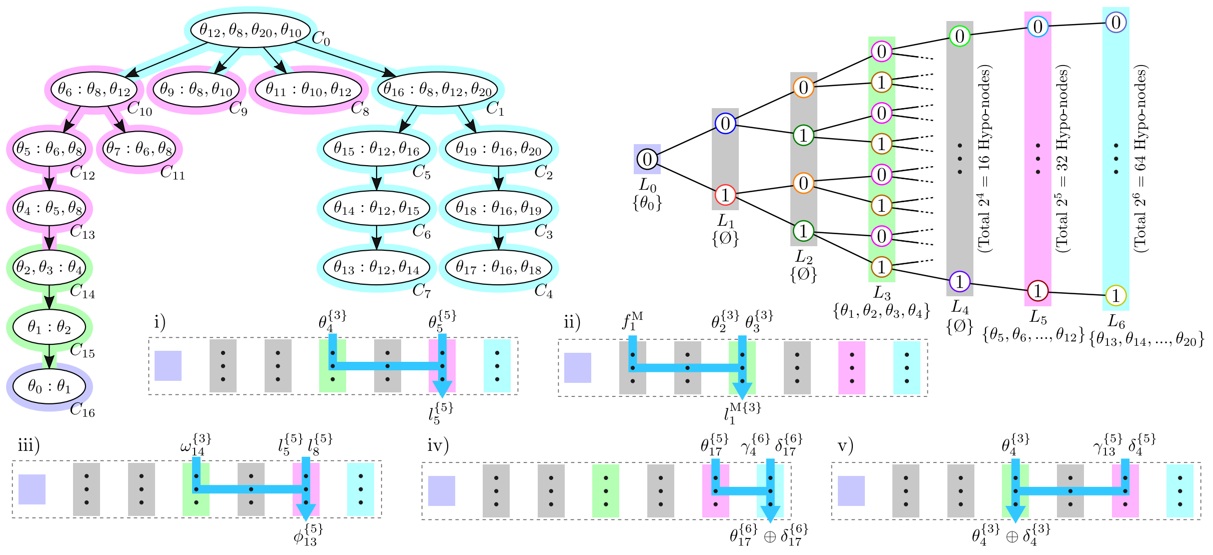

(b) The associations between the MHBT (left) and the Hypo-tree (right), and examples of finding the correspondences of local hypotheses

Figure 3.7: An example of inference process in a MHBT with the help of Hypo-tree. (a) A SLAM problem with ambiguity is represented as a MHFG. (b) The corresponding MHBT and Hypo-tree are constructed from the MHFG and associated with each other (the MHCD $\gamma_q^{\{r\}}$ in each clique C_q is associated with the Hypo-layer L_r that is colored the same as the shadow of C_q). Hypo-tree is used to not only find the correspondences among the modes of MMFs f_r^{M} and the hypotheses of MHVs $\theta_p^{\{r\}}$, MHJDs $\phi_q^{\{r\}}$, MHCDs $\gamma_q^{\{r\}}$, and MHMDs $\omega_q^{\{r\}}$, but also determine the output hypotheses of $\theta_p^{\{r\}}$, $\phi_q^{\{r\}}$, and $\gamma_q^{\{r\}}$. For example: i) Linearization of the SMF between θ_4 and θ_5. ii) Linearization of the MMF f_1^{M}. iii) Forming the MHJDs $\phi_{13}^{\{r\}}$ of clique C_{13}. iv) Retraction (denoted as $\oplus$) of θ_{17} in C_4, assuming the loop closing factor between θ_{10} and θ_{20} is newly added in this iteration (expanding the hypotheses of θ_{17}). v) Retraction of θ_4 in C_{13} (merging the hypotheses of δ_4).

column in practice. Then, their Hessian matrices $\Lambda_{[j]}^{\{r'\}} = \left(A_{[j]}^{\{r'\}}\right)^{\top}\left(A_{[j]}^{\{r'\}}\right)$ that represents the local densities are generated for the next step.

3.5.4 Clique-based Elimination

Based on the ordering of MHVs in Sec. 3.5.2 and the clique formation algorithm in [51], each MHLF is grouped into one of the cliques. Then, the MHJD $\phi_q^{\{r'\}}$ of each clique C_q can be generated as a set of Hessian matrices $\mathbf{\Phi}_q^{\{r'\}} = \left\{\Phi_{q[j]}^{\{r'\}} | 0 \leq j < n_{r'}\right\}$ by summing up the corresponding $\Lambda_{[j]}^{\{r\}}$ and input MHMDs $\omega_{q^*}^{\{r^*\}}$ from its children C_{q^*}, which is also a set of Hessian matrices . Because of the incremental update strategy (see Sec. 3.5.6), the number of hypotheses $n_{r'}$ of $\phi_q^{\{r'\}}$ in clique C_q is no less than n_{r^*} of any of its input MHMDs $\omega_{q^*}^{\{r^*\}}$ ($r' \geq r^* \Rightarrow n_{r'} > n_{r^*}$). As a result, one matrix of an input $\omega_{q^*}^{\{r^*\}}$ can be reused by more than one $\Phi_{q[j]}^{\{r'\}}$ of $\phi_q^{\{r'\}}$, which is more efficient than conducting the same process in individual Bayes trees for each $h_{[j]}^{\{t\}}$.

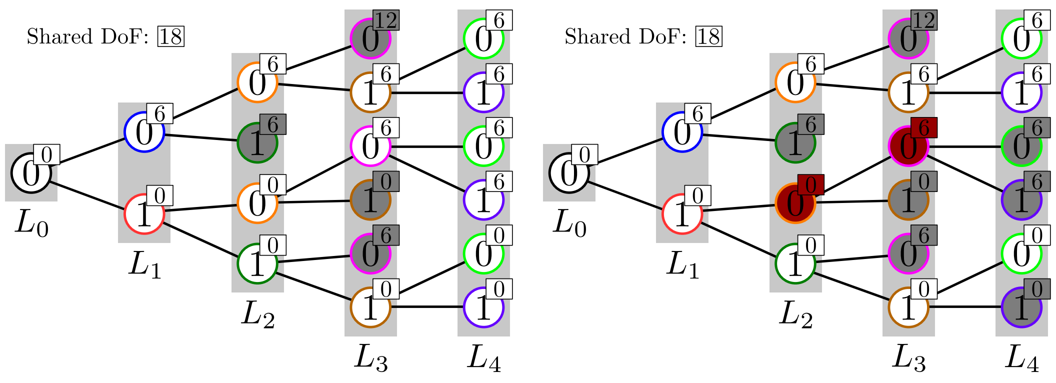
Finally, we apply *partial Cholesky factorization* on each $\Phi_{q[j]}^{\{r'\}}$ of $\mathbf{\Phi}_q^{\{r'\}}$ based on the *frontals* Θ_q^{F} and *separators* Θ_q^{S} in C_q (as defined in [51]) and *eliminate* the frontals Θ_q^{F} from the rest of the inference process:

$$\Phi_{q[j]}^{\{r'\}} = Q_{q[j]}^{\top}\left(\left(\Gamma_{q[j]}^{\{r'\}}\right)^{\top}\left(\Gamma_{q[j]}^{\{r'\}}\right) + \begin{bmatrix} 0 & 0 \\ 0 & \Omega_{q[j]}^{\{r'\}} \end{bmatrix}\right) Q_{q[j]}, \tag{3.6}$$

where $Q_{q[j]}$ is an orthonormal matrix that makes $\Gamma_{q[j]}^{\{r'\}}$ upper-triangular. The set of factorized matrices $\Gamma_q^{\{r'\}} = \left\{\Gamma_{q[j]}^{\{r'\}} | 0 \leq j < n_{r'}\right\}$ that represents the MHCD $\gamma_q^{\{r'\}}$ is stored in C_q, while $\Omega_q^{\{r'\}} = \left\{\Omega_{q[j]}^{\{r'\}} | 0 \leq j < n_{r'}\right\}$ is the set of Hessian matrices of the MHMD $\omega_q^{\{r'\}}$ that is passed to the parent clique of C_q. In practice, we cache $\omega_q^{\{r'\}}$ as in [51] to save computations in incremental updates (see Sec. 3.5.6).

3.5.5 Backsubstitution and Retraction

Following the backsubstitution algorithm in [51], we can solve the *multi-hypothesis linear updates (MHLU)* $\delta_p = \left\{\delta_{p[j]} | 0 \leq j < n_{r''}\right\}$ of each frontal $\theta_p^{\mathrm{F}} \in \Theta_0^{\mathrm{F}}$ from the root to all the leaves. However, since δ_p is calculated from $\gamma_q^{\{r\}}$ and each $\delta_{p'}$ of its corresponding separator $\theta_{p'} \in \Theta_q^{\mathrm{S}}$, the number $n_{r''}$ of hypotheses of δ_p is determined by the largest number of hypotheses among $\gamma_q^{\{r\}}$ and all $\delta_{p'}$, which might be greater than n_r of $\theta_p^{\{r\}}$ and even contain redundant duplicated values. Thus, we first try to merge numerically similar values in δ_p based on their hypotheses correspondences. Then, if the number $n_{r'''}$ of hypotheses of δ_p is still larger than that of $\theta_p^{\{r\}}$ after merging, we expand the hypotheses of $\theta_p^{\{r\}}$ to match with it ($\theta_p^{\{r\}} \rightarrow \theta_p^{\{r'''\}}$, see Fig. 3.7-b) for retraction.



(a) Before pruning in L_4

(b) After pruning in L_4

Figure 3.8: An example of hypotheses pruning and DoF recording (small boxes) in Hypo-tree, where f_1^{M} and f_3^{M} (associated to L_1 and L_3) are type #3 MMFs. (a) Before pruning in L_4. (b) After pruning in L_4 (dark gray nodes) with backward pruning (dark red nodes). The part of DoF that results from type #3 MMFs (which can be different for each $h_{[j]}^{\{t\}}$) is stored in each $h_{[j]}^{\{t\}}$ while the rest part of DoF that results from all other factors is shared by all $h_{[j]}^{\{t\}}$. For example, the DoF of $h_{[1]}^{\{4\}}$ (see the Hypo-node at the top-right corner) is $18+6=24$.

3.5.6 Incremental Update

Because of the causality property of the growth of hypotheses, constructing MHBT incrementally can achieve better efficiency. As new factors being added into the corresponding MHFG, the top part of the MHBT is rebuilt without changing the subtrees that are not directly linked with the new factors. As a result, the hypotheses of the MHVs in those subtrees are not changed except for the hypotheses expansion of MHVs during backsubstitution.

3.6 Hypotheses Pruning

3.6.1 Pruning Criteria

The unwanted and unlikely hypotheses (see Sec. 3.3.3 and 3.4.1) are pruned to maintain the efficiency right after the elimination step (see Sec. 3.5.4). First, every overall hypothesis $h_{[j]}^{\{t\}}$ in the the last Hypo-layer L_t with its corresponding *squared system error* $e_{[j]}^2$ larger than its 95% chi-square threshold $\chi_{[j]}^2$ is pruned. If the number n_{R_1} of remaining hypotheses is greater than a threshold n_{desire}, we further prune those $h_{[j]}^{\{t\}}$ with the fewest degrees of freedom (DoF) $d_{[j]}^{\{t\}}$ (the same DoF of the chi-square function). Then, if the number n_{R_2} of remaining hypotheses is greater than another threshold n_{limit} (e.g. all the $d_{[j]}^{\{t\}}$ are the same), we prune those with lower chi-square probabilities one-by-one until their final remaining number n_{R_3} is smaller than n_{limit} ($n_{\text{limit}} \geq n_{\text{desire}}$ is a relaxed bound that allows tracking more hypotheses when too many of them all seem likely).

In practice we use the error $e'_{[j]}$ of the linearized system in each iteration to approximate the actual system error $e_{[j]}$, which is calculated directly in the elimination step (see Sec. 3.5.4) as the bottom right element of the matrix $\Gamma_{0[j]}^{\{t\}}$ of the MHCD $\gamma_0^{\{t\}}$ of the root clique C_0. Each DoF $d_{[j]}^{\{t\}}$ is recorded in two parts as shown in Fig. 3.8.

3.6.2 Pruning in Both Trees

Once an overall hypothesis $h_{[j]}^{\{t\}}$ is pruned, we flag the corresponding Hypo-node $N_{[j]}^{\{t\}}$ as pruned, and no children will be created from $N_{[j]}^{\{t\}}$ (see Fig. 3.8-a). Also, after finishing flagging in the last Hypo-layer L_t, we check every unflagged Hypo-node $N_{[j]}^{\{r\}}$ from L_{t-1} to L_1 and flag it if all its children are flagged (see Fig. 3.8-b). Then, the associated values of those flagged $N_{[j]}^{\{r\}}$ in each MHV $\theta_p^{\{r\}}$ and MHCD $\gamma_q^{\{r\}}$ of the MHBT are removed immediately. Notice that we only remove the associated $\Omega_{q[j]}^{\{r\}}$ of a cached MHMD $\omega_q^{\{r\}}$ when it is used in the incremental update step.

3.7 Simulations

3.7.1 Settings

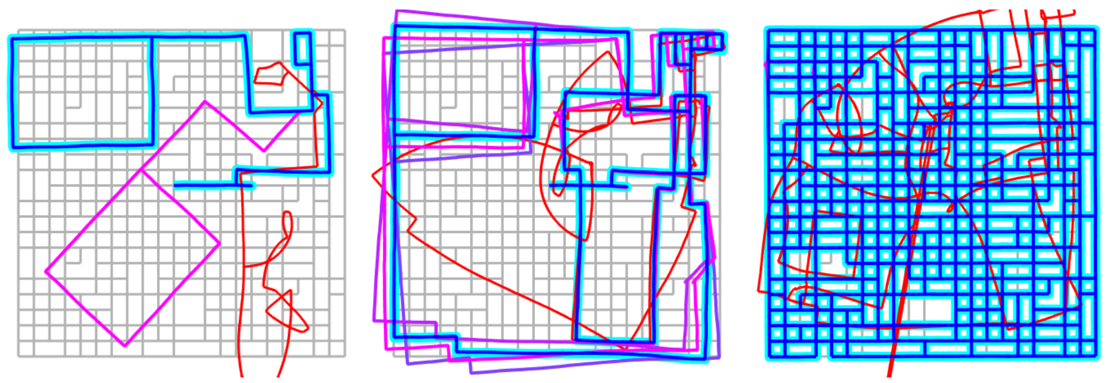
At this stage, we evaluate the accuracy and efficiency of MH-iSAM2 through simulations. All computation is executed on a desktop computer with an Intel Core i7-4790 processor. Whenever a new observation is added into the system, we run one iteration of update in the MHBT and calculate the most up-to-date estimates of all MHVs.

3.7.2 Results

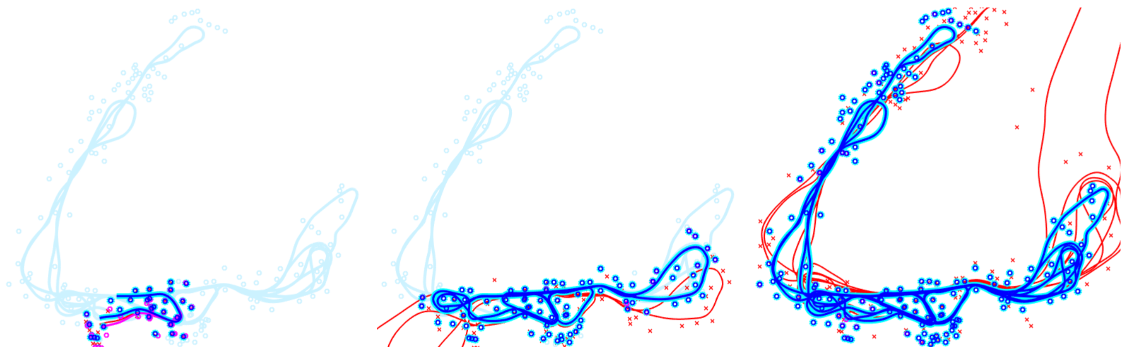
The three different types of ambiguities (see Sec. 3.3.2) are simulated based on the *city10000* or *Victoria Park* dataset, and some example outputs are shown in Fig. 3.9. From the timing analysis in Fig. 3.10, we can tell that the speed of MH-iSAM2 is constant to the overall complexity of ambiguity in general. Moreover, MH-iSAM2 is efficient enough to track up to $n_{\text{limit}} = 30$ hypotheses within less than $30\times$ of time of original iSAM2 in all of the simulations. However, in each iteration the speed can vary due to the extra computation for hypothesis handling. Also, since the number of modes m of each type #2 or type #3 MMF affects the topology and density of the MHFG, it can affect the efficiency as well.

As for the accuracy analysis of the hypothesis with the smallest root-mean-square error (RMSE) in Fig. 3.10, we can tell that MH-iSAM2 can keep track of the "correct hypothesis" with reasonable cost of time. Although some wrong loop closures might be added into all remaining hypotheses (see Fig. 3.9-c and 3.10-c), their pollution of the outputs are hardly visible since they all pass the chi-square threshold.

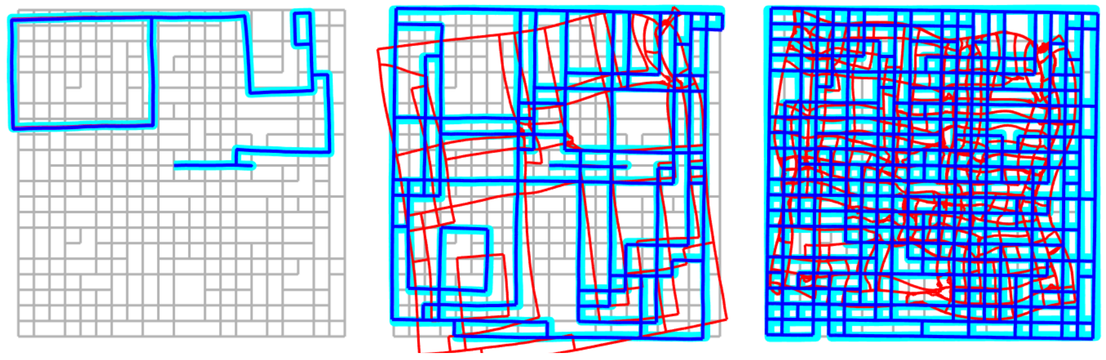
Even though each test case in the analysis above only contains one type of MMF, and each tested MMF only contains 2 modes, there is no constraint on the number of modes of each MMF or combinations of their types in the current MH-iSAM2 framework (see Fig. 3.11). In real



(a) Type #1: ambiguity in odometry with total 2^{173} complexity



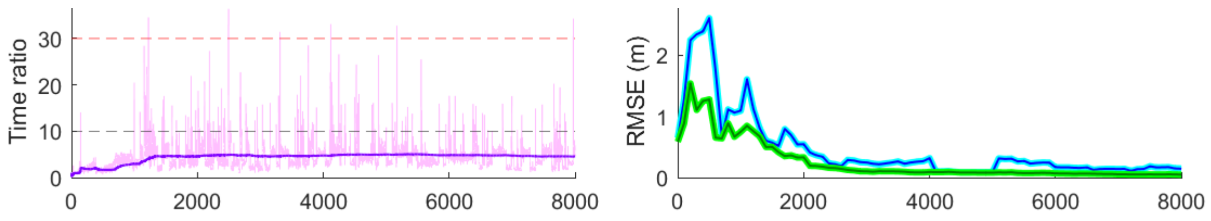
(b) Type #2: ambiguity in data association with total 2^{412} complexity



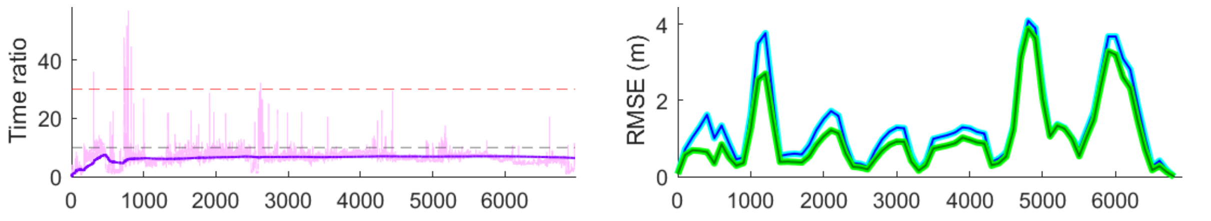
(c) Type #3: ambiguity in loop closing with total 2^{8000} complexity

🟦 Most possible solutions from MH-iSAM2 (w/ ambiguity)	🟥 iSAM2 (w/ ambiguity)	🟦 iSAM2 (w/o ambiguity)
	▨ Groundtruth	🟦 Substitution for groundtruth (above setting, entire dataset)

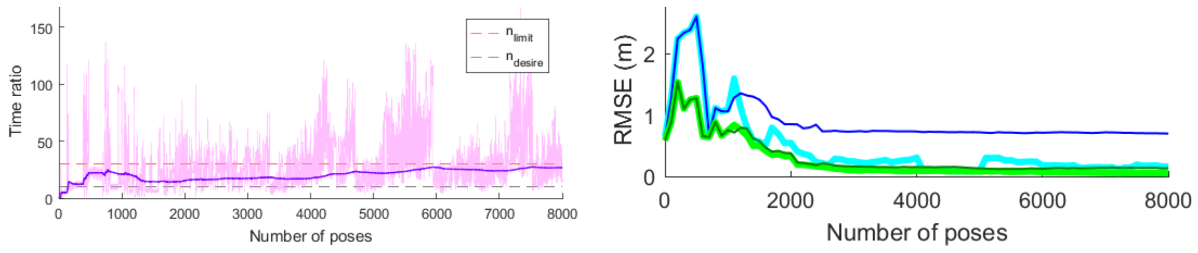
Figure 3.9: Examples of MH-iSAM2 tracking and solving various types of ambiguities (see Sec. 3.3.2) in SLAM problems online incrementally as more observations are added into the system (left to right). Multiple most possible solutions can be solved from (a) a pose graph with ambiguous odometry measurements, (b) a SLAM problem with ambiguous data association of 2D feature points, and (c) a pose graph with ambiguous loop closures.



(a) Type #1: Multi-measurement factor (based on city10000 dataset)



(b) Type #2: Multi-association factor (based on Victoria Park dataset)



(c) Type #3: Boolean factor (based on city10000 dataset)

🟪 Time ratio $\frac{\text{MH-iSAM2}}{\text{iSAM2}}$ at each iteration	iSAM2:	🟦 Raw RMSE	🟩 ATE RMSE
🟪 Accumulated time ratio	The most accurate hypothesis of MH-iSAM2:	🟦 Raw RMSE	🟩 ATE RMSE

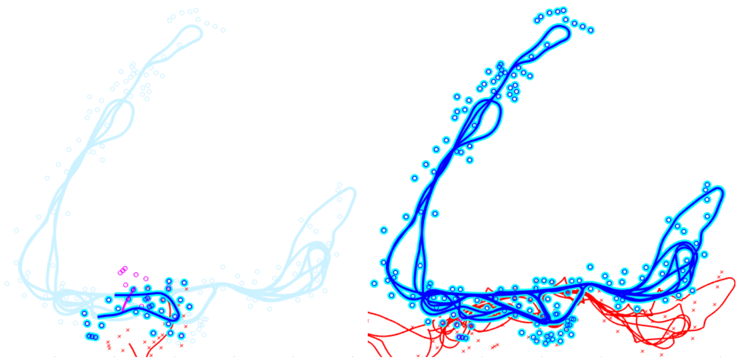
Figure 3.10: Speed (left column) and accuracy (right column) analysis of each type of MMF. Notice that in (b) we take the complete iSAM2 result as groundtruth since the real groundtruth is unavailable.

world applications, various types of ambiguities can occur at the same time. We will show how MH-iSAM2 handles them properly in Ch. 6.

Finally, the detailed percentage of time spent in each component in MH-iSAM2 is shown in Fig. 3.12. The percentage varies from case to case due to the different types of factors (both SMFs and MMFs), different number of hypotheses tracked throughout the process, and different sizes of MHFG.

3.7.3 Discussion

So far, only the basics of MH-iSAM2 is developed and evaluated. There are several improvements and extensions that can be done in the near future. First, the type #3 MMF can be combined with type #1 or type #2 MMF to represent an additional possibility that all the existing modes are invalid. Second, since the merging of δ_p only consider the numerical changes in each


(a) Type #1 and type #2

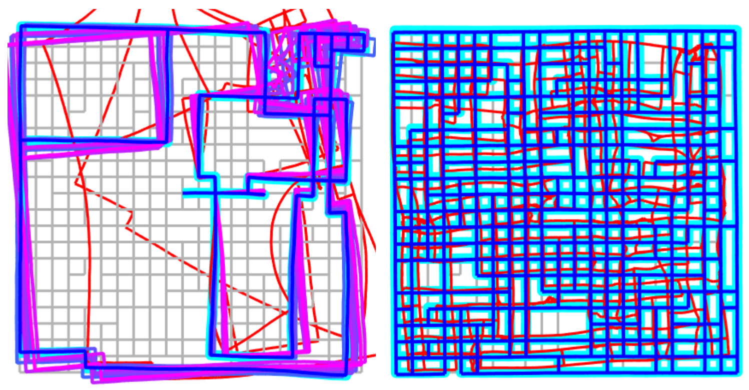

(b) Type #1 and type #3

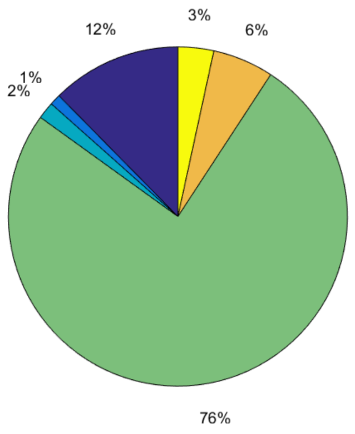
Figure 3.11: The example results of MH-iSAM2 with two types of ambiguities.

iteration instead of the entire process until convergence, some of the θ_p might be expanded accidentally and end up containing more hypotheses than needed (e.g.: θ_1 in Fig. 3.7). Third, $n_{\max}$ and n_{limit} should be adjusted online based on the current complexity of ambiguity to avoid losing track of the correct hypotheses. Lastly, even though current MH-iSAM2 framework deals with discrete ambiguity only, modeling the degeneracy and continuous ambiguity in the same framework seems possible yet requires more studies.

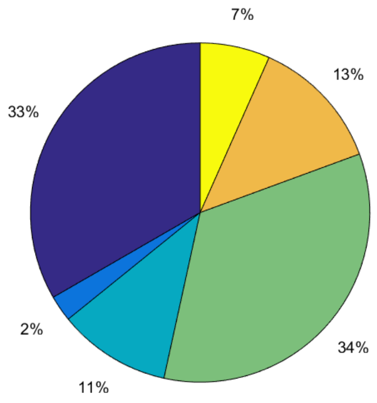
Designed as an online incremental SLAM solver, both Hypo-tree and MHBT in MH-iSAM2 are constructed based on temporal ordering intuitively. However, if we would like to solve the same SLAM problem with ambiguities offline in batch, a different ordering of Hypo-layers and MHVs might give us better efficiency. For example, if some MMFs are easier to be disambiguated than others, we might be able to choose an ordering to solve them first so that the growing number of hypotheses can be reduced early. Again, this could be a new research direction that requires more studies.

3.8 Conclusion

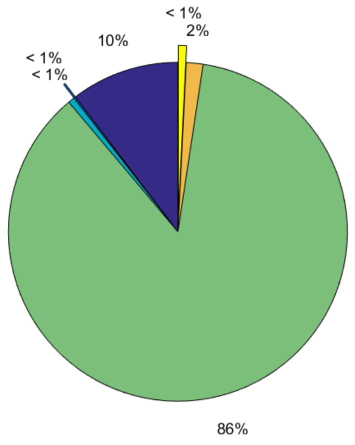
In this chapter, we present the novel online incremental nonlinear optimizer MH-iSAM2 to handle the ambiguities in SLAM. Based on the Hypo-tree, MHBT, and the hypothesis pruning al-



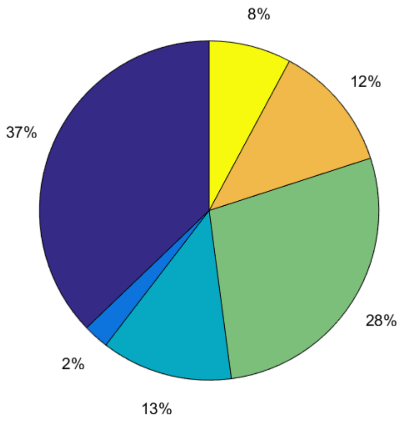
(a) From dataset in Fig. 3.9-a



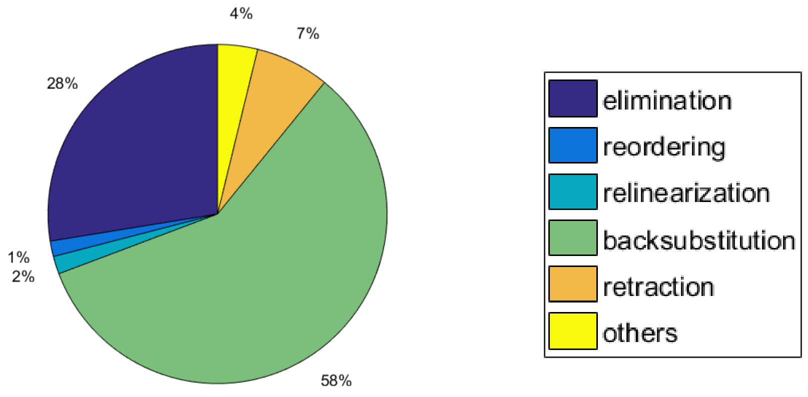
(b) From dataset in Fig. 3.9-b



(c) From dataset in Fig. 3.9-c



(d) From dataset in Fig. 3.11-a



(e) From dataset in Fig. 3.11-b

Figure 3.12: Percentage of time spent in each component of MH-iSAM2 for the 5 different datasets in Fig. 3.9 and 3.11. Notice that in (b) and (d), elimination dominates the time cost more because of using type #2 MMFs that potentially make the MHFG denser. On the other hand, backsubstitution dominates the time cost in (a), (c), and (e) since their numbers of hypotheses and sizes of MHGFs are larger.

gorithm, MH-iSAM2 can take multi-mode measurements as inputs and output multi–hypothesis results efficiently, therefore greatly enhance the robustness of SLAM systems.

Based on MH-iSAM2, we will focus on solving real-world ambiguity problems in SLAM in Ch. 6, and further extend the developed system to improve the robustness of active SLAM in Ch. 7.

Chapter 4

Fast RGB-D-inertial Odometry for Robust State Estimation

4.1 Choice of Sensors

In modern SLAM systems, various types of on-board sensors and their combinations have been adopted, such as cameras, LiDARs, sonars, inertial sensors, and wheel odometers (if the system includes a mobile platform). More recently, RGB-D sensors and event cameras have also been studied for state estimation and 3D mapping. Each of these sensors has its own advantages and disadvantages, and one might even be complementary to the other. For example, cameras are popular and low-cost, but cannot measure depth directly and therefore require more computation to reconstruct high quality dense maps from multiple images (e.g.: searching for match point along each epipolar line [103]). On the other hand, LiDAR measures depth directly, but its spatial resolution is usually low, and it is also more expensive than other sensors.

To achieve good balance among cost, complexity, and information diversity, we combine two different types of sensors in our proposed ambiguity-aware passive and active SLAM system. The first one is an RGB-D sensor, which consists of an RGB camera and an infrared camera with an infrared pattern projector to actively measure the depth of the scene (e.g.: Microsoft Kinect [136]). Even though the maximum range of depth measurement is short (around 8 meters), and the infrared projector and camera do not work well in sun light, an RGB-D sensor still preserves the advantages of both camera and LiDAR, and is suitable for indoor 3D SLAM. The second sensor is an inertial measurement unit (IMU), which utilize microelectromechanical systems (MEMS) or optics to measure angular velocity and acceleration at a high frequency (up to 500~1000 Hz). The complementary nature of these two sensors is obvious: while an RGB-D sensor can acquire the structure and texture information of the environment for mapping, it also relies on them for state estimation. On the other hand, IMU cannot contribute any information to the map directly, but it greatly helps state estimation especially in textureless environments due to directly measuring its own motion. As a result, the proposed sensor combination is expected to achieve high accuracy and robustness. However, since ambiguity problems can still occur in some critical situations, the robust back-end (see Sec. 1.3) is still desired, and an approach to detect and model ambiguities will be discussed in Ch. 6. In this section, we will focus on developing an efficient RGB-D-inertial odometry algorithm that can operate in real-time on a single CPU.

4.2 Direct RGB-D Odometry Framework

We combine geometric and photometric methods as described in [133][134] to realize a *fast RGB-D odometry (FRO)* algorithm. Geometric and photometric methods are known to be complementary to each other and can achieve better accuracy and robustness together if they are jointly optimized:

$$E_{\text{total}} = E_{\text{geo}} + \lambda E_{\text{pho}}, \tag{4.1}$$

where E_{geo} and E_{pho} are the error terms from the geometric and photometric method respectively, and E_{total} is the overall energy term that should be minimized in the optimization. The weighting λ is used to adjust the relative importance of the two error terms.

A two-stage framework is also applied in FRO to improve the robustness of pose estimation. In the first step, a novel *iterative projected plane (IPP)* method (as the geometric component) and a pyramid dense RGB-D odometry method [117, 55] using Laplacian images (as the photometric component) are jointly optimized to estimate a rough odometry. Then, in the second step, the same IPP will be combined with a *semi-dense RGB-D odometry (SRO)* method, as the geometric and photometric components, respectively, to estimate a precise odometry. These three methods are described in detail in Sec. 4.3, Sec. 4.4, and Sec. 4.5 respectively.

4.3 Iterative Projected Plane

The basic idea of IPP algorithm is to only use the planar regions for 3D registration, which allows a much faster pose estimation if sufficient planar regions are observed in both the new frame F_{n} and the reference frame R_{r}, which is usually true in general indoor environments. First, planes are fitted to small regions from all over the depth image of each frame. Then, camera projection is applied to find associations between the planes in the two frames. By minimizing the distances between the associated plane pairs, the relative transformation between the two frames can be updated iteratively until convergence. However, it is hard to extract accurate and robust small planar regions efficiently from the noisy raw depth data. Therefore we preprocess the depth image to reduce noise in the potentially planar regions before extracting the planes.

In general indoor scenes, if a region is smooth in intensity (color) it is likely to be smooth in depth as well. As a result, we iteratively smooth the depth image in areas with low intensity gradient with a three-pixel kernel. Note that we do not smooth the entire depth image so that the non-planar regions with strong geometric features (e.g. edges and corners) will not be averaged out and mistaken as planar regions.

After the depth smoothing, we extract planes from small regions in the partially-smoothed depth image. First, we uniformly divide the depth image into small grid cells. In each grid cell, we uniformly sample several points and calculate the local normal directions of those points. If more than a threshold of normal vectors are close to parallel, we use these points and normal vectors to generate a plane model, find inlier points in the same grid cell, and refine the plane model using all the inliers (see Fig. 4.1-a). A plane $\pi_{\text{n}}^{[i]}$ in F_{n} is corresponded to a plane $\pi_{\text{r}}^{[i]}$ in R_{r} if the projection of the center point $\mathbf{c}_{\text{n}}^{[i]} = \begin{bmatrix} x_{\text{n}}^{[i]} & y_{\text{n}}^{[i]} & z_{\text{n}}^{[i]} \end{bmatrix}^\top$ of $\pi_{\text{n}}^{[i]}$ onto R_{r} is within the grid cell that contains $\pi_{\text{r}}^{[i]}$ given the current estimation of the relative transformation between these two frames. With m plane correspondences found in any iteration, the relative transformation between the two frames can be found by minimizing the geometric error:

$$E_{\text{geo}} = \sum_{i=1}^{m} \left\| \left(R(\boldsymbol{\xi}) \mathbf{c}_{\text{n}}^{[i]} + \mathbf{t}(\boldsymbol{\xi}) \right)^\top \mathbf{n}_{\text{r}}^{[i]} + d_{\text{r}}^{[i]} \right\|^2 \tag{4.2}$$

$$\approx \sum_{i=1}^{m} \left\| A^{[i]} \boldsymbol{\xi} - e^{[i]} \right\|^2, \tag{4.3}$$

where $A^{[i]} = (\mathbf{n}_{\text{r}}^{[\text{i}]})^\top \begin{bmatrix} -[\mathbf{c}_{\text{n}}^{[\text{i}]}]_\times \mid \mathbf{I}_{3\times3} \end{bmatrix}$ and $e^{[i]} = -\mathbf{n}_{\text{r}}^{[i]\top} \mathbf{c}_{\text{n}}^{[i]} - d_{\text{r}}^{[i]}$. $\boldsymbol{\xi} = \begin{bmatrix} \beta & \gamma & \alpha & t_x & t_y & t_z \end{bmatrix}^\top$ is the minimal representation of the desired 6 DoF relative transformation with three rotations (β, γ, α) and three translations (t_x, t_y, t_z). $R(\boldsymbol{\xi})$ and $\mathbf{t}(\boldsymbol{\xi})$ are the corresponding rotation matrix and translation vector of $\boldsymbol{\xi}$. $\mathbf{c}_{\text{n}}^{[i]}$ is approximated as the intersection of $\pi_{\text{n}}^{[i]}$ and the projection ray that goes through the center pixel of the grid cell that contains $\pi_{\text{n}}^{[i]}$. $\mathbf{n}_{\text{r}}^{[i]} = \begin{bmatrix} a_{\text{r}}^{[i]} & b_{\text{r}}^{[i]} & c_{\text{r}}^{[i]} \end{bmatrix}^\top$ is the unit normal vector and $d_{\text{r}}^{[i]}$ is the distance parameter of the plane $\pi_{\text{r}}^{[i]}$,

$$a_{\text{r}}^{[i]} x + b_{\text{r}}^{[i]} y + c_{\text{r}}^{[i]} z + d_{\text{r}}^{[i]} = 0, \tag{4.4}$$

that corresponds with $\pi_{\text{n}}^{[i]}$. Note that Eq. 4.3 is an approximation of Eq. 4.2 when the transformation is small, which is a valid assumption for most 30 fps hand-held SLAM problems. IPP can run at about 100 fps on a single thread of a CPU due to the projective plane association and the relatively small number of planes compared to the number of points in the raw image.

Since IPP relies on planar surfaces only, it cannot find accurate pose estimations alone if sufficient planes are not observed and matched in the scene (e.g. cluttered scene without enough planar surfaces). As a result, we integrate other photometric methods with IPP to solve this problem.

4.4 Pyramid Dense RGB-D Odometry

Our pyramid dense RGB-D odometry method mostly follows [117, 55] except that we utilize the Laplacian of the downsampled images. We use Laplacian (see Fig. 4.1-b) instead of grayscale images to alleviate the effect of illumination variation. In each iteration of optimization, our pyramid RGB-D odometry method minimizes the photometric error:

$$E_{\text{pho}} = \sum_{i=1}^{n} \left\| J_{\text{r}}^{[i]} \boldsymbol{\xi} - r^{[i]} \right\|^2, \tag{4.5}$$

where $J_{\text{r}}^{[i]}$ is the Jacobian of the i-th valid pixel in the downsampled Laplacian image of R_{r} with respect to a 6 DoF perturbation (3 rotations and 3 translations), $\boldsymbol{\xi}$ is again the desired 6 DoF

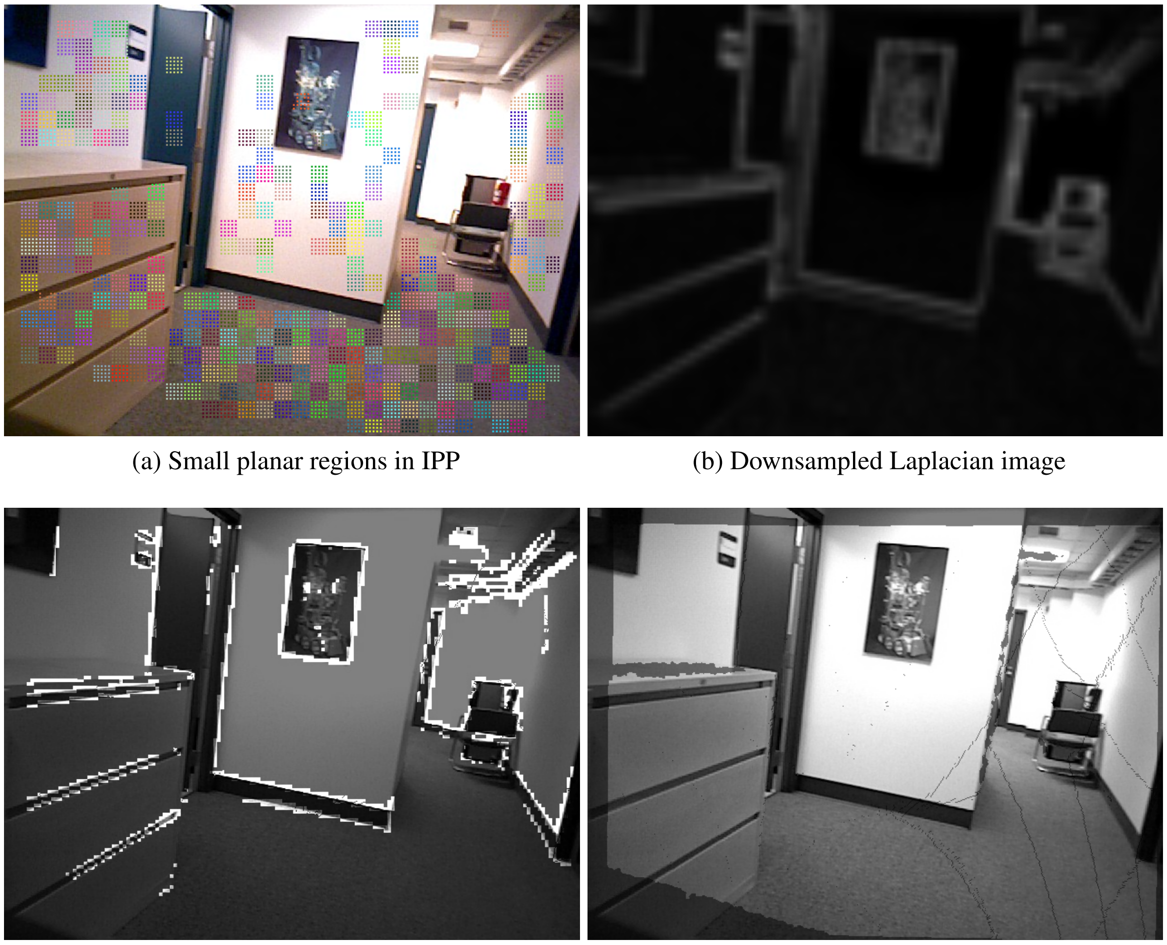

(a) Small planar regions in IPP
(b) Downsampled Laplacian image
(c) Selected pixels in SRO
(d) Final alignment

Figure 4.1: A rough odometry transformation is estimated by combining IPP (a) and pyramid odometry with Laplace images (b). Using this pose as an initialization point, we can combine IPP (a) and SRO (c) to estimate a more precise transformation (d).

relative transformation that is gradually refined in each iteration, and $r^{[i]}$ is the residual between each valid pixel pair in the downsampled Laplacian image of R_{r} and the reprojection (based on $\boldsymbol{\xi}$) of the downsampled Laplacian image of F_{n}. n is the number of valid pixel pairs that are used in each iteration. Note that we start at the 5th pyramid level (coarsest) and stop at the 3rd for computational efficiency, only estimating rotation at the 5th level. In addition to efficiency, this coarse-to-fine scheme helps handle larger motion and avoids the optimization getting stuck in wrong local minimum.

4.5 Semi-dense RGB-D Odometry

Inspired by [21], our SRO algorithm only uses few pixels with high intensity gradients in the residual minimization process instead of the entire image [117, 55] for better efficiency. This simplification is based on the fact that many of the calculations on the regions with low intensity gradients do not contribute much to optimizing odometry and therefore can be discarded without losing much robustness.

Ideally we can directly select those pixels with large intensity gradients only and ignore all others. However, if a region with large intensity gradient corresponds to a discontinuous structure in the real world, the depth measurements of these pixels are often missing, and their reprojection cannot be calculated. As a result, we first choose some *interest pixels* with large intensity gradients, and then select all the pixels with valid depth data within a small patch around each interest pixel (see Fig. 4.1-c) for calculation. Again, with the semi-dense pixels selected from the patches in both R_{r} and F_{n}, the relative transformation between the two frames can be found by minimizing the photometric error in Eq. 4.5. $J_{\mathrm{r}}^{[i]}$ is now the Jacobian of the i-th selected pixel in the grayscale image of R_{r} with respect to the same 6 DoF perturbation, and $r^{[i]} = p_{\mathrm{r}}^{[i]} - p_{\mathrm{n}}^{[i']}$ is the residual between each selected pixel $p_{\mathrm{r}}^{[i]}$ in R_{r} and corresponding selected pixel $p_{\mathrm{n}}^{[i']}$ in F_{n}, with correspondences found projectively. n is again the number of valid pixel pairs that are used in each iteration.

4.6 Reference Frame Sharing

In the actual implementation of the algorithm, a group of neighboring frames share the same reference frame for better efficiency. The shared reference frames are selected based on the motion of the sensor, where every two consecutive reference frames satisfy a threshold on relative transformation. Moreover, because we warp input pixels toward the reference frame in each iteration of the optimization, the Jacobians of the image pyramid and the semi-dense patches in SRO are calculated only when a new reference frame is defined, which saves significant computation time compared to calculating the Jacobians in each frame. Another advantage of reference frame sharing is that there might be less drift accumulation compared to a frame-to-frame formulation. The overall dense RGB-D odometry method can operate faster than real-time (more than 50 fps) on a single CPU, which leaves plenty of time for other components in the proposed integrated SLAM system, such as fusion and dense mapping (see Ch. 5), multi-hypothesis state estimation (see Ch. 6), or even active SLAM (see Ch. 7).

4.7 RGB-D-Inertial Fusion Based on IMU Preintegration

To combine inertial measurements with the FRO for better state estimation, we can predict the relative pose of each frame to the most recent reference frame using the preintegrated rotation $\Delta\mathtt{R}_{t+n\Delta t}^{t}$ and translation $\Delta\mathbf{p}_{t+n\Delta t}^{t}$ over the n number of IMU measurements between the two frames as

$$\tilde{\mathtt{R}}_{t+n\Delta t} = \mathtt{R}_t \Delta\mathtt{R}_{t+n\Delta t}^{t}, \tag{4.6}$$

$$\tilde{\mathbf{p}}_{t+n\Delta t} = \mathbf{p}_t + \mathtt{R}_t \Delta\mathbf{p}_{t+n\Delta t}^{t} + \mathbf{v}_t n\Delta t - \frac{1}{2}\mathbf{g} n\Delta t^2, \tag{4.7}$$

which can be computed easily from the preintegrated IMU factor (see Sec. 2.4.3). Using the preintegrated IMU measurements to predict the initial pose of each frame results in better odometry estimation, especially in the cases of fast rotation and lack of texture.

Jointly optimizing RGB-D odometry with IMU measurements at every reference frame in a loosely-coupled manner (see Sec. 2.4.1) further results in more accurate estimation of IMU states in the global optimization, which also allows the correction of the biases of the IMU measurements efficiently. The inertial constraints between each two consecutive reference frames is defined as in Eq. 2.5-2.7, and the RGB-D odometry constraint is defined as a 6-DoF pose-to-pose factor. Notice that since the poses of the IMU states represent the poses of the IMU instead of RGB-D camera, each original RGB-D odometry estimation ${}^{c}\mathtt{T}_{\mathrm{rgbd}}$ has to be transformed into the IMU coordinates before taken as a odometry factor between IMU states. The transformation

$$\mathtt{T}_{\mathrm{rgbd}} = \mathtt{T}_{\mathrm{c}} \cdot {}^{c}\mathtt{T}_{\mathrm{rgbd}} \cdot \mathtt{T}_{\mathrm{c}}^{-1} \tag{4.8}$$

is based on the relative camera pose $\mathtt{T}_{\mathrm{c}}$ in the IMU coordinates, which can be calibrated offline in advance. This loosely-coupled *fast RGB-D-inertial odometry (FRIO)* algorithm is designed to be integrated with a SLAM and dense mapping system (see Ch. 5), and is expected to achieve reliable pose estimation most of the time. As for its failure cases, we propose to apply MH-iSAM2 (see Ch. 3) to handle it in the back-end (see Ch. 6).

4.8 Preliminary Experiment

To test our FRIO algorithm, we generate simple dense 3D maps of indoor environment based on it. The adopted sensors are an ASUS Xtion Pro Live with 30 fps and 640×480 resolution in both color and depth images, and a Microstrain 3DM-GX4-25 with 1000 Hz raw rotational velocity and acceleration measurements (see Fig. 4.2). Other than some drift in rotation, FRIO can estimate the poses correctly in general (see Fig. 4.3). Since there is no public RGB-D-inertial dataset with groundtruth, and the purpose of the FRIO algorithm is for dense 3D SLAM, the detailed experiments of FRO and FRIO are conducted by integrating them into a dense planar SLAM system (see Ch. 5) with their results shown in Sec. 5.7.

4.9 Conclusion

In this chapter, we introduce the fast RGB-D-inertial odometry (FRIO) algorithm that can estimate the states of the sensors from a sequence of RGB-D and inertial data. Utilizing the state-of-the-art techniques for direct odometry estimation and visual-inertial fusion as well as our improvement in efficiency, FRIO can run faster than real-time on a CPU while maintaining good robustness and accuracy in general static indoor environments. As for challenging cases that can fail FRIO, such as aggressive motions, textureless scenes, or dynamic objects, we will apply the robust back-end optimizer (see Ch. 3) with ambiguity detection and modeling methods in the front-end of the SLAM system (see Ch. 6) to handle them in order to achieve better robustness.

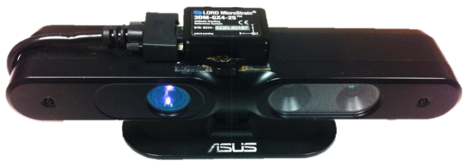
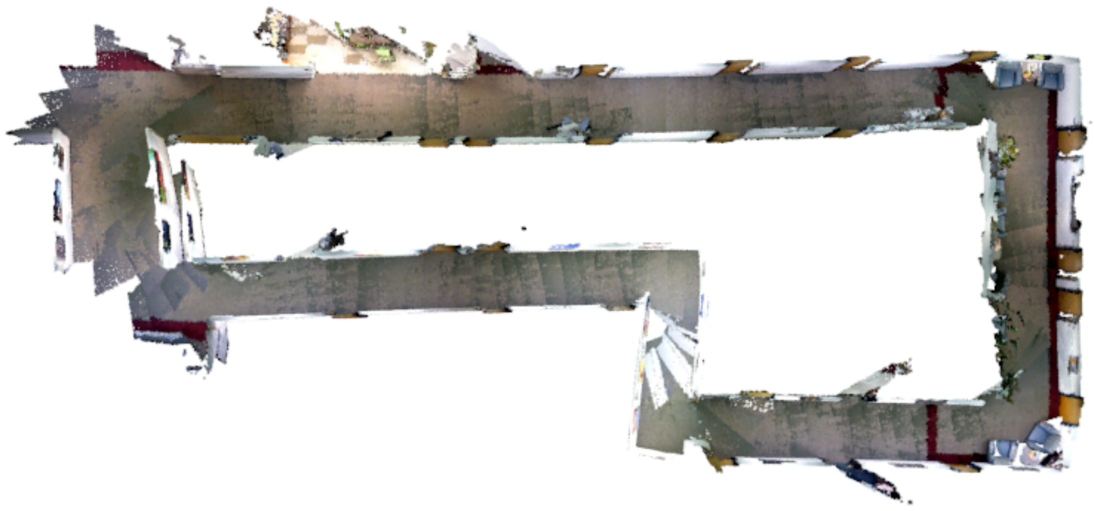


Figure 4.2: The IMU (Microstrain 3DM-GX4-25) is rigidly attached on the top of the RGB-D sensor (ASUS Xtion Pro Live).


(a) Top view

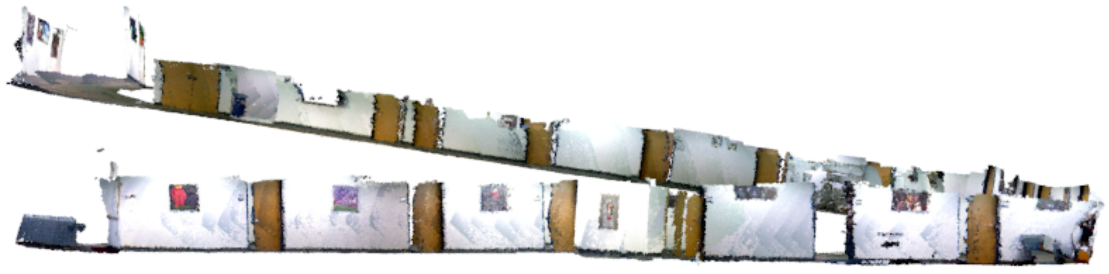

(b) Side view

Figure 4.3: An example result of dense 3D mapping based on the proposed FRIO algorithm.

Chapter 5

Dense 3D Mapping Using Submaps and Planes

5.1 Introduction

In order to reconstruct dense 3D maps online efficiently even under ambiguities, we study several possible mapping approaches adopted in existing SLAM systems that are also suitable for the proposed ambiguity-aware multi-hypothesis SLAM framework. The first idea is hierarchical mapping, where one simple example is to generate local submaps first, and use these submaps to represent a complete global map by placing them at the correct locations. We will discuss how to make this process as efficient as possible in order to run in real-time on a CPU with all other components in a SLAM system. Another idea is to use planes as landmarks in a SLAM system so that the general geometric structure of indoor environments can be preserved in a much cheaper yet still dense representation, while the global structure of the reconstructed maps can be more accurate due to the additional constraints offered by the planar measurements. We will discuss these ideas in this chapter with other relevant materials such as structural constraints and loop closings, and will show their advantages by integrating them in real world SLAM systems, including the conventional single-hypothesis solutions (see Sec. 5.7) and the proposed multi-hypothesis approach (see Sec. 6.5).

5.2 Local Fusion for Submap Reconstruction

Fusing multiple overlapping depth images with known poses into a dense map is a straightforward way for 3D mapping, which also greatly reduces the noise in the output map. In this work, we fuse neighboring frames locally into the latest keyframe to generate a dense submap. Placing all dense submaps in the global coordinates with their corresponding keyframe poses can represent the entire model, which can be updated globally by updating the keyframe poses. Notice that keyframes are selected based on the same sensor-motion criteria used to select reference frames (see Sec. 4.6) but with larger thresholds.

The depth submap M_i of a keyframe K_i is initialized simply with the depth image of K_i. Again by utilizing the same sensor-motion criteria used to select reference frames (see Sec. 4.6),

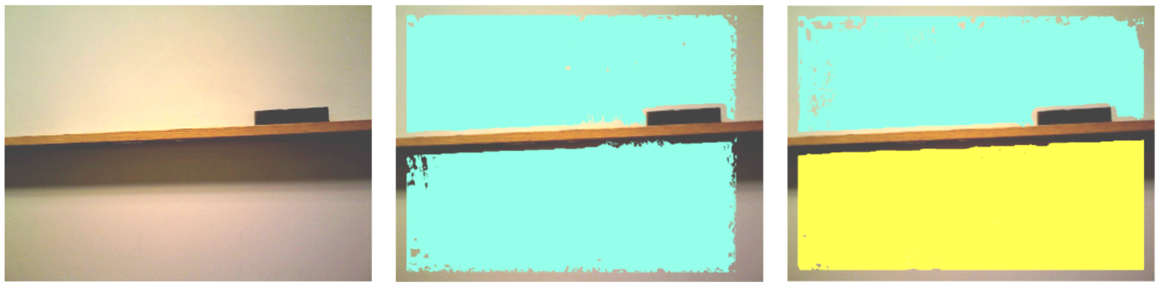


Figure 5.1: (a) Color image from a keyframe shows a whiteboard offset from a wall by approximately 1cm. (b) The segmentation algorithm cannot discriminate between the wall and the whiteboard when performed on the raw depth map. (c) The fused depth map allows the algorithm to correctly segment the wall and whiteboard as separate planes.

but this time with smaller thresholds, fusion frames are regularly selected to fuse depth data into the current local map M_i. The depth measurements from these fusion frames are projected into the depth image of K_i based on the odometry estimation and then fused into M_i using a running average method for each pixel. To avoid fusing incorrectly associated depth measurements, only measurements that are within a small threshold of the keyframe's corresponding depth measurement are fused.

With reasonably accurate pose estimates, the local fusion process produces a significantly smoothed model after fusing as few as 3-4 frames. The frequency of fusion frame selection may be adjusted to allow for real-time performance of the overall system. Fusion frames will continue to be selected and fused with M_i until a new keyframe K_{i+1} is selected, at which point planes will be segmented from M_i and a new submap M_{i+1} will be initialized.

Combining the submaps with their known poses allows fast global reconstruction of large-scale environments, and any new information can update the global map easily by moving the submaps locally. Moreover, the fused local maps enable more precise plane segmentations than are possible using just a single frame (see Fig. 5.1), as we use the clustering algorithm described in [35] for plane segmentation.

5.3 Plane Fitting and Uncertainty Estimation

Following segmentation, a plane model is fitted to each point cluster using the linear model described in [22]: $\delta^{[i]} = au^{[i]} + bv^{[i]} + c$, where $\delta^{[i]}$ is the disparity (proportional to inverse of depth measurement), $u^{[i]}$ and $v^{[i]}$ are the pixel coordinates, and a, b, and c are the unknown parameters that depend on both the (known) camera intrinsics as well as the parameters of the underlying plane $\boldsymbol{\pi}$. Our noise model assumes additive Gaussian noise on disparity, which leads to the standard least squares model

$$\left\| \begin{pmatrix} \delta^{[1]} \\ \delta^{[2]} \\ \vdots \\ \delta^{[n]} \end{pmatrix} - \begin{pmatrix} u^{[1]} & v^{[1]} & 1 \\ u^{[2]} & v^{[2]} & 1 \\ \vdots & \vdots & \vdots \\ u^{[n]} & v^{[n]} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right\|^2 = \|\mathbf{y} - X\boldsymbol{\beta}\|^2 . \tag{5.1}$$

The optimal parameters are solved for as $\boldsymbol{\beta}^* = \left(X^\top X\right)^{-1} X^\top \mathbf{y}$ which may be used to compute the optimal plane parameters $\boldsymbol{\pi}^* = T\left(\boldsymbol{\beta}^*\right)$. The explicit form of T is omitted for brevity. The covariance of the parameters $\boldsymbol{\beta}^*$ is $\Sigma_\beta = \left(X^\top X\right)^{-1}$. Σ_β is transformed to the space of the plane parameters $\boldsymbol{\pi}$ using the Jacobian of T computed numerically: $\Sigma_{\boldsymbol{\pi}} = J_T \Sigma_\beta J_T^\top$. However, since global optimization utilizes the minimal parametrization to update planes, the covariance matrix must match the dimensionality of the minimal parametrization (three). Therefore, we compute the 3×3 covariance matrix $\Sigma'_{\boldsymbol{\pi}} = J_l \Sigma_{\boldsymbol{\pi}} J_l^\top$ using the Jacobian of the log map, which is described in [48]. This is the final covariance matrix that is used in the global optimization.

5.4 Data Association

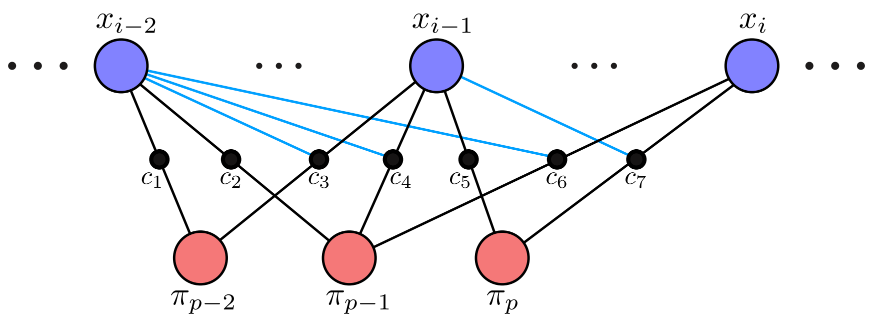
We implement a novel projective data association algorithm for matching planes between keyframes. Once planes are extracted from keyframe K_{i-1}, all of the landmarks seen in the previous 10 keyframes are considered candidates for data association and are projected into the frame of K_{i-1} using the globally optimized pose estimates. An exhaustive search across measurement-landmark pairs is used to find the best correspondences. Three criteria must be met in order to match a new plane measurement with a previously existing landmark. The first two criteria are common in the literature: the normals must be within a small threshold of each other as well as the distances of the planes from the origin (we use $10°$ and 0.2m). The last criterion computes the residual of the landmark's plane model using the points from the plane measurement, normalized for the number of points. That is, for landmark Π_p and measurement c_q, we compute the cost

$$C_{pq} = \frac{\left\|\mathbf{y}_q - X_q \boldsymbol{\beta}_p\right\|^2}{n} \tag{5.2}$$

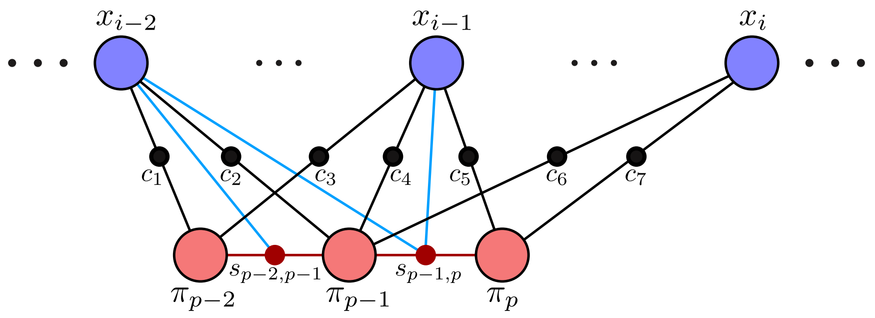
where $\mathbf{y}_q$ and X_q are the corresponding data from measurement c_q as defined in Eq. 5.1, n is the number of points observed in measurement c_q, and $\boldsymbol{\beta}_p$ is the vector of regression parameters corresponding to landmark Π_p. We use a threshold of 10 for C_{pq}, which was empirically determined to minimize the number of false positive correspondences while allowing for some uncertainty in the odometry motion estimate. Plane measurements from K_{i-1} that are not matched with previously observed landmarks are added to the graph as new landmarks.

5.5 Optimization of 3D Planes

An infinite plane landmark is represented as a unit length homogeneous vector $\boldsymbol{\pi} = \begin{bmatrix}\mathbf{n}^\top & d\end{bmatrix}^\top \in \mathbb{P}^3$ in projective space in our global optimization, where $\mathbf{n}$ is the normal vector of the plane and



(a) Bases of planar constraints



(b) Bases of structural constraints

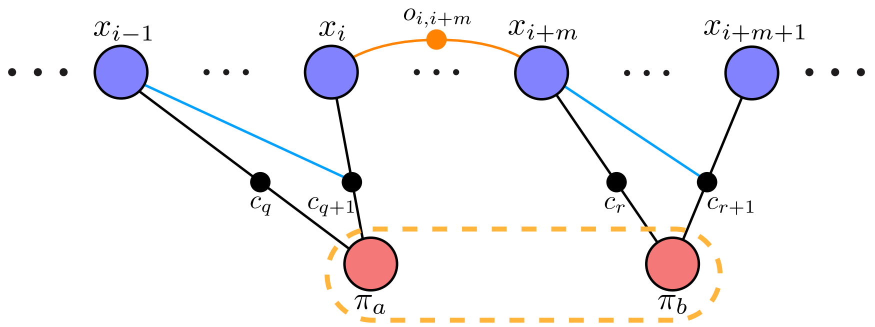
Figure 5.2: Both planar and structural constraints have links (blue lines) to their bases. (a) All the planar factors that link to the same landmark plane are also linked to the same base, which is the IMU state of the keyframe that first observes that landmark plane. (b) Each structural constraint is linked to two bases, which are the IMU states of the keyframes that first observe the two landmark planes that the structural constraint is linking to.

d is its distance from the origin. By enforcing $\|\boldsymbol{\pi}\| = 1$ we parametrize planes on S^3. We deal with over-parametrization by using the same minimal representation $\boldsymbol{\omega} \in \mathbb{R}^3$ as for quaternions, exploiting the exponential map

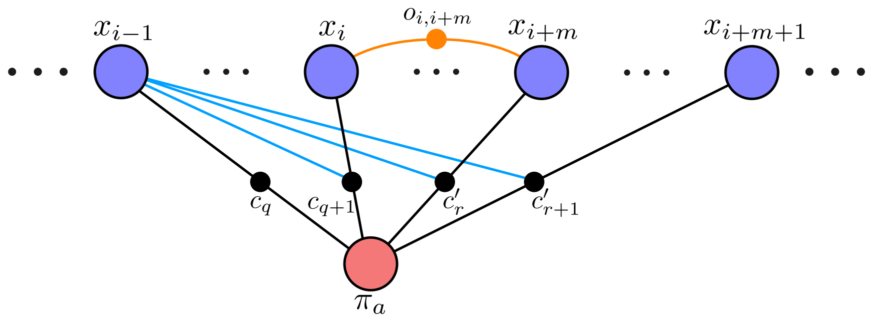
$$\exp(\boldsymbol{\omega}) = \begin{pmatrix} \frac{1}{2} sinc\left(\frac{1}{2}\|\boldsymbol{\omega}\|\right)\boldsymbol{\omega} \\ \cos\left(\frac{1}{2}\|\boldsymbol{\omega}\|\right) \end{pmatrix} \in S^3. \tag{5.3}$$

for updating the plane during optimization, as discussed in [48].

By optimizing the submap poses together with the landmark planes, both of them can be further refined, and the corresponding dense 3D model can be updated accordingly. Based on this parameterization of 3D planes, a relative formulation is further applied for each landmark plane π_p by setting its base as the pose x_i that corresponds to the keyframe K_i that first observes π_p. Every plane observation factor c_q that links a pose node to π_p will be additionally linked to x_i (the corresponding pose of K_i) through a ternary factor (see Fig. 5.2-a). This allows faster convergence especially in loop closures since the planes anchored to a pose will be automatically moved along with the pose when there is a global update.



(a) Before merging two landmark planes



(b) After two landmark planes are merged

Figure 5.3: Loop closing with landmark planes merging in the global factor graph $\mathcal{G}$. (a) A pose-to-pose loop closing constraint is added, and two landmark planes π_a and π_b are detected to be representing the same plane. (b) The two landmark planes are merged, and the new factors c'_r and c'_{r+1} are added into $\mathcal{G}$ to replace c_r and c_{r+1}. When all of the factors of the landmark plane π_b are removed, π_b will be automatically removed from $\mathcal{G}$ in the applied GTSAM implementation of iSAM2.

5.6 Global Drift Correction

5.6.1 Loop Closure

No matter how accurate the pose estimation from the VIO algorithm is, it can still drift over time. Loop closure is the common solution to correct the global drift of a SLAM system, which usually includes place recognition and relative pose estimation in the pipeline. In this thesis, we apply a bag-of-words approach [29] to detect loops and the following algorithm to close the loops.

For every keyframe K_j that is detected to be a loop closure candidate with a previous keyframe K_i, we apply the RANSAC-based perspective-*n*-point (P*n*P) algorithm in OpenCV [91] on the SURF [5] feature points extracted from K_i and K_j to estimate the relative transformation first, then apply our fast dense RGB-D odometry method to refine it. The refined output is added into the global factor graph $\mathcal{G}$ as a constraint between the poses of K_i and K_j. Optimizing $\mathcal{G}$ with the keyframe-to-keyframe loop closing constraint usually requires updating a larger part of the underlying Bayes tree in iSAM2, which takes several iterations to converge.

Finally, we can check the similarity of landmark planes that are observed in these two keyframes using the same association method, and merge those that are actually representing the same plane to further constrain the solution and avoid the duplication of landmarks. The merging is implemented in iSAM2 [51] by relinking the factors of each new landmark plane to the corresponding old one while also updating their base poses (see Fig. 5.3).

5.6.2 Structural Constraint

Other than loop closures, structural constraints between planes can also help correct global rotational drift. Orthogonality and parallelism are the two most common structural constraints found between two planar surfaces in indoor environments, which can be added into our planar-inertial SLAM system to further correct the drift in rotation.

In our system, two landmark planes π_a and π_b that are observed within a short interval are the candidate pairs for structural constraints. For each pair π_a and π_b, we first compute $h_{ab} = \left|(\mathtt{R}_a\mathbf{n}_a)^\top \mathtt{R}_b\mathbf{n}_b\right|$, which is the absolute value of the dot product of their normal vectors in the global coordinates, where $\mathtt{R}_a$, $\mathtt{R}_b$ are the rotation matrices of their base poses respectively. Then, the two planes are regarded as parallel if h_{ab} is greater than a parallel threshold $h_{\parallel}$, orthogonal if h_{ab} is less than an orthogonal threshold $h_{\perp}$, or no specific relationship if none of the above.

Each structural constraint factor is linked to not only the two corresponding landmark planes but also their bases as a quaternary factor (see Fig. 5.2-b) because the relative formulation (see Sec. 5.5) results in faster convergence in the optimization. For any pair of π_a and π_b, the error function of their orthogonal constraint factor is

$$e_{\perp} = \frac{1}{\sigma_{\perp}^2}\left[(\mathtt{R}_a\mathbf{n}_a)^\top \mathtt{R}_b\mathbf{n}_b\right]^2, \tag{5.4}$$

and the error function of the parallel constraint factor is

$$e_{\parallel} = \frac{1}{\sigma_{\parallel}^2}\left\|[\mathtt{R}_a\mathbf{n}_a]_\times \mathtt{R}_b\mathbf{n}_b\right\|^2. \tag{5.5}$$

The variances $\sigma_{\perp}$ and $\sigma_{\parallel}$ for the orthogonal and parallel factors are set to be small (3×10^{-5}) for strong constraints.

5.7 Experiments

5.7.1 Implementations and Settings

We develop two SLAM systems to evaluate the performance of FRO (see Sec. 4.2), FRIO (see Sec. 4.7), and the map representation of submaps and planes all together. The first system is called keyframe-based dense planar *SLAM (KDP-SLAM)*, which uses FRO without an IMU sensor for state estimation (see Fig. 5.4). The second one is called dense planar-inertial SLAM (DPI-SLAM), which uses FRIO that fuses the data from an RGB-D sensor and an IMU for better state estimation (see Fig. 5.5). We implement both SLAM systems in a multi-thread fashion on a desktop computer with an Intel Core i7-4790 processor, and GPU being used only for visualization, not computation. Our own datasets are collected using the same sensor setup

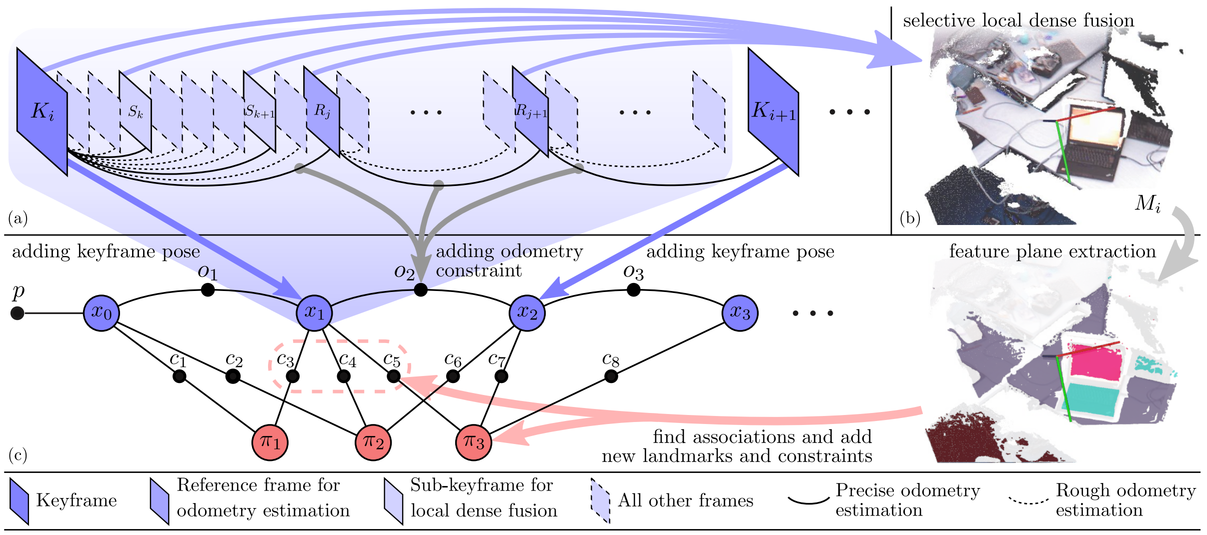


Figure 5.4: The KDP-SLAM system consists of three concurrent threads: (a) FRO and frame labeling process, (b) selective local depth fusion algorithm, and (c) global planar mapping with loop closing. Note that for the set of all keyframes $\mathcal{K}$, all reference frames $\mathcal{R}$, and all fusion frames $\mathcal{U}$, $\mathcal{K} \subset \mathcal{R} \subset \mathcal{U}$ holds. Also, the possible loop closing constraints are not shown here but in Fig. 5.3 for readability.

described in Sec. 4.8. *Lightweight communications and marshalling (LCM)* [40] is adopted to transmit the RGB-D and inertial data to our system online, or log and replay them to simulate the real-time process.

5.7.2 KDP-SLAM Results

We compare our KDP-SLAM with other dense RGB-D SLAM and planar SLAM methods on the synthetic ICL-NUIM datasets [34] and the real-world TUM RGB-D datasets [118] quantitatively. In addition, we provide the 3D reconstructions generated by KDP-SLAM and Kintinuous [133] (one of the state-of-the-art large-scale dense SLAM algorithms) for qualitative evaluation since ground truth trajectories and maps are not available for our own sequences. Another state-of-the-art dense SLAM system ElasticFusion fails catastrophically on our large-scale datasets since it is not designed for such environments.

Table 5.1 shows the absolute trajectory (ATE) [118] root-mean-square error (RMSE) of the resulting trajectories of the *living room sequences with noise* in the ICL-NUIM synthetic dataset. See Fig. 5.6-a for sample 3D reconstruction using KDP-SLAM. The trajectory error of our method is comparable to the state-of-the-art, with KDP-SLAM outperforming each of the alternative methods on at least one of the sequences. Note that KDP-SLAM outperforms the only other CPU-only algorithm in most datasets, and occasionally outperforms the GPU-accelerated systems.

Although KDP-SLAM can reconstruct the general structure from the TUM datasets (see Fig. 5.6-b), its quantitative results (ATE RMSE) are about 5 times worse than the results of other state-of-the-art methods. This is entirely expected, as KDP-SLAM utilizes a much cheaper

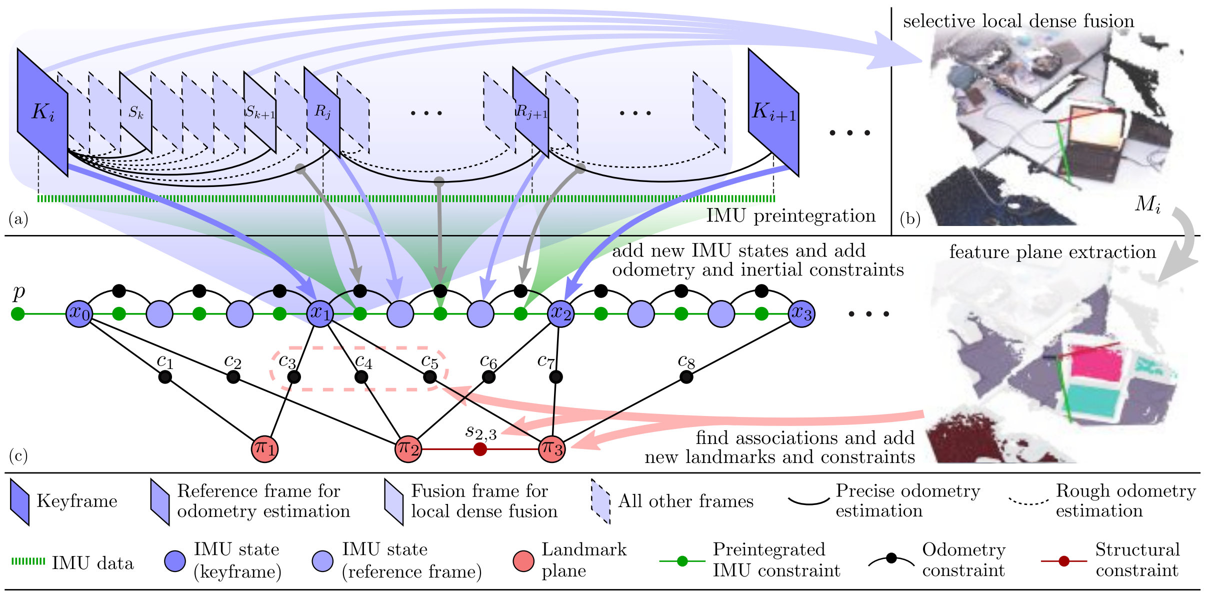

selective local dense fusion
K_i
S_k
S_{k+1}
R_j
R_{j+1}
K_{i+1}
M_i
(a)
IMU preintegration
(b)
add new IMU states and add odometry and inertial constraints
feature plane extraction
p
x_0
x_1
x_2
x_3
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
$s_{2,3}$
π_1
π_2
π_3
(c)
find associations and add new landmarks and constraints
Keyframe
Reference frame for odometry estimation
Fusion frame for local dense fusion
All other frames
Precise odometry estimation
Rough odometry estimation
IMU data
IMU state (keyframe)
IMU state (reference frame)
Landmark plane
Preintegrated IMU constraint
Odometry constraint
Structural constraint

Figure 5.5: The system structure of DPI-SLAM, which is similar to KDP-SLAM but modified to allow fusing IMU measurements: (a) FRIO and frame labeling process. (b) Selective local depth fusion algorithm. (c) Optimization of IMU states and planar landmarks in the global factor graph $\mathcal{G}$ with structural constraints and loop closing (see Fig. 5.3).

Table 5.1: Comparison of ATE RMSE (unit: m) on the synthetic ICL-NUIM datasets. The *italicized* methods require GPU for computation. The errors that are smaller than ours are underlined.

System	lr kt0n	lr kt1n	lr kt2n	lr kt3n
DVO SLAM [54]	0.104	0.029	0.191	0.152
RGB-D SLAM [19]	0.026	0.008	0.018	0.433
Kintinuous [133]	0.072	0.005	0.010	0.355
ElasticFusion [134]	0.009	0.009	0.014	0.106
Dense planar SLAM [111]	0.246	0.016	-	-
CPA-SLAM [73]	0.007	0.006	0.089	0.009
KDP-SLAM (Ours)	0.009	0.019	0.029	0.153

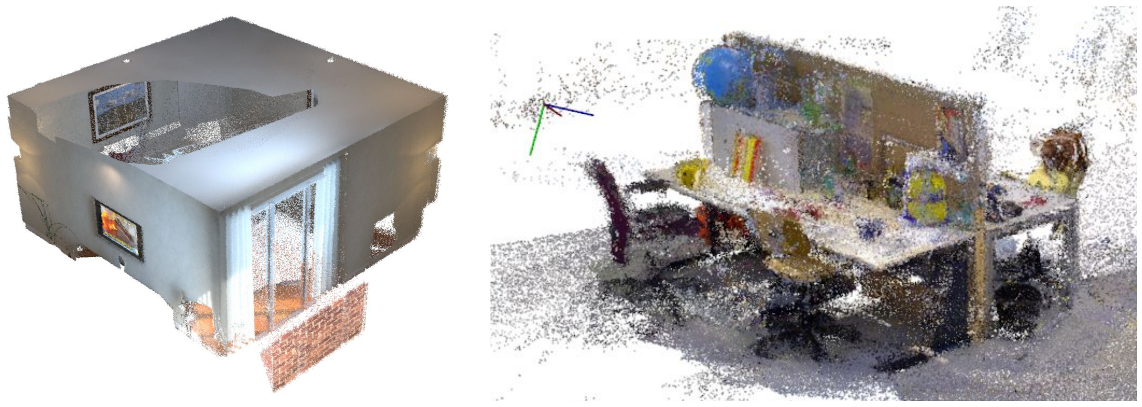


Figure 5.6: KDP-SLAM reconstruction of (a) ICL-NUIM “lr kt2n” sequence and (b) TUM “freiburg3_long_office_household” sequence (10x downsampled pointcloud).

and less accurate odometry method than those algorithms, which makes tracking difficult in the presence of strong rotation, image blur, rolling shutter effects, lighting changes or misalignment between color and depth images (all of which are present in the TUM sequences). Furthermore, many of the sequences from the TUM dataset capture highly cluttered environments with few distinguishable planes, whereas KDP-SLAM is specifically designed for highly-planar environments.

Fig. 5.7 shows a sample dataset gathered using the hand-held RGB-D sensor. The sequence traverses two corridors on different floors and the connecting staircases before finishing with a large loop closure. As we can observe from the results, Kintinuous distorts the planes even with loop closing, while KDP-SLAM maintains the planar structure and significantly reduces the drift in the map. More results with highlighted planar structures are shown in Fig. 5.8 and 5.9 (10x downsampled pointcloud).

5.7.3 DPI-SLAM Results

Since no public RGB-D-inertial SLAM dataset with ground truth was available at the time of this work, we compare the 3D reconstructions generated by different settings of DPI-SLAM with KDP-SLAM for qualitative evaluation. We also provide a quantitative evaluation by comparing the output model from DPI-SLAM with a ground truth model. The dense 3D ground truth model is obtained with a FARO Focus3D survey LiDAR scanner [23] from a sequence of stationary 360 degree scans.

The reconstruction results from the various settings of the system are shown in Figs. 5.10 and 5.11. Notice that even though this dataset is collected at the same place as the dataset in Fig. 5.8 is, we move about 3 times faster during the data collection process in order to demonstrate the improved robustness and accuracy of FRIO in DPI-SLAM comparing to FRO in KDP-SLAM. As a result, even though KDP-SLAM can reduce drift along each corridor, the drift at each corner and along the long corridor are still visible in the output map (see Fig. 5.10-a). Naively adding preintegrated IMU factors between keyframes does affect the result in some way (see

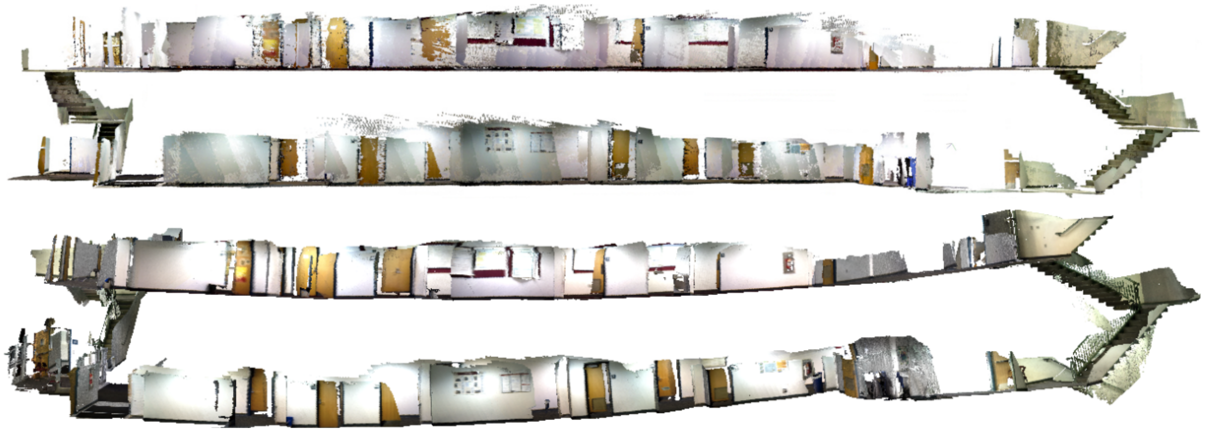


Figure 5.7: Two floor dataset reconstructions by KDP-SLAM (top) and Kintinuous (bottom). Note that KDP-SLAM has reduced drift and maintained the planar structure after loop closure compared to Kintinuous.

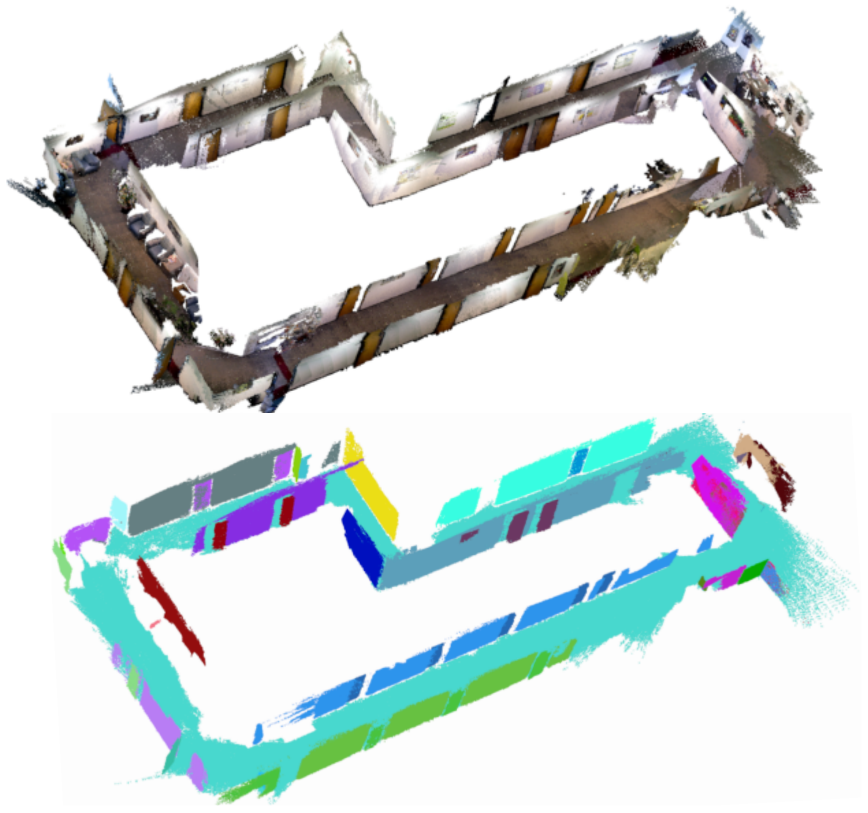


Figure 5.8: Our KDP-SLAM system can reconstruct large indoor environments with loops. Top: An example of dense pointcloud map. Bottom: Dense map with false-colored planes.

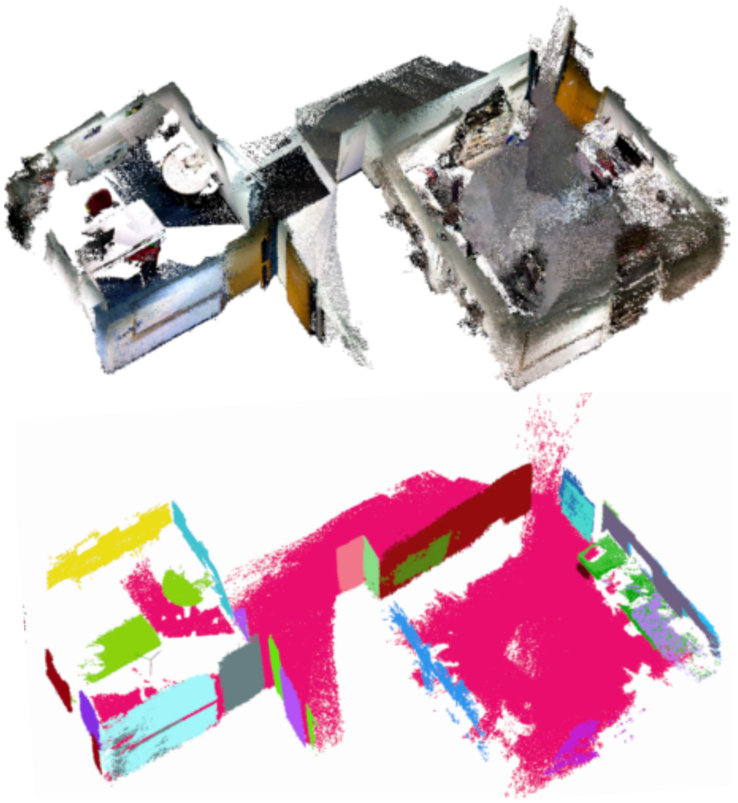


Figure 5.9: Reconstruction of two rooms using our KDP-SLAM system (top) and its corresponding false-colored planar map (bottom).

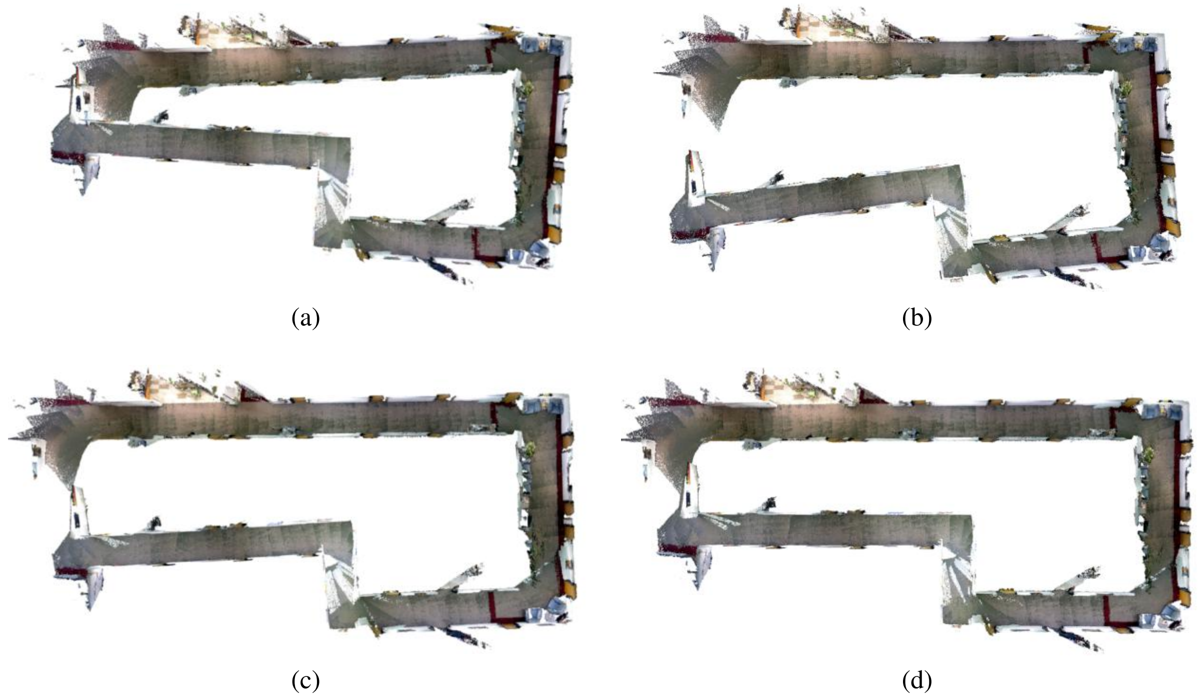


Figure 5.10: The real-time dense 3D reconstructions (from top view) *without loop closure* in four different settings: (a) KDP-SLAM without using IMU data, (b) planar-inertial SLAM with IMU preintegrated over each two keyframes, (c) DPI-SLAM: planar-inertial SLAM with IMU preintegrated over each two reference frames, and (d) DPI-SLAM with structural constraints.

Fig. 5.10-b), but its drift is still quite large in both rotation (e.g. the upper-right corner) and translation (e.g. the upper long corridor) because the IMU biases are not corrected frequently enough, and also the assumption of constant bias for IMU preintegration might not hold within longer preintegration intervals. The proposed method of preintgerating IMU measurements over each two consecutive reference frames results in a much better reconstruction (see Fig. 5.10-c), where the drift in rotation at the corners and translation along the corridor are both reduced significantly. Adding structural constraints between landmark planes further corrects the drift (see Fig. 5.10-d) and allows better loop closing results, which is also true when the loop is detected and closed (see Fig. 5.11). Finally, we can use DPI-SLAM with structural constraints to reconstruct the 3D dense model of the entire floor from the full dataset (see Fig. 5.13). More reconstruction results of different indoor environments are shown in Figs. 5.14 and 5.15.

After registering our output model in Fig. 5.11-b with the survey LiDAR model using ICP, we calculate the point-to-plane root-mean-square error (RMSE) and mean absolute error (MAE) between the two models, which are 0.069m and 0.049m respectively. Given that the entire model is about $30\text{m} \times 13\text{m} \times 3\text{m}$, the average error ratio is less than 0.7%. Since we cannot record the RGB-D and IMU data sequence at the same time when the LiDAR scans are collected, there can be uncontrollable changes in the public environment (see Fig. 5.12). Therefore, the actual errors between the two methods should be even smaller.

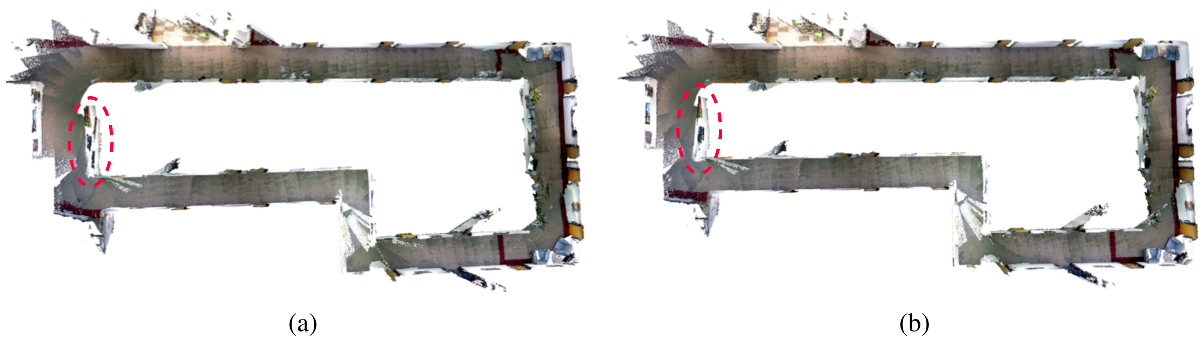


Figure 5.11: The dense 3D reconstructions *with loop closure* (circled in red) in two different system settings. (a) DPI-SLAM without structural constrains (corresponds to Fig. 5.10-c), which takes about 30 iterations to close the loop. (b) DPI-SLAM with structural constrains (corresponds to Fig. 5.10-d), which takes only 10 iterations to close the loop.

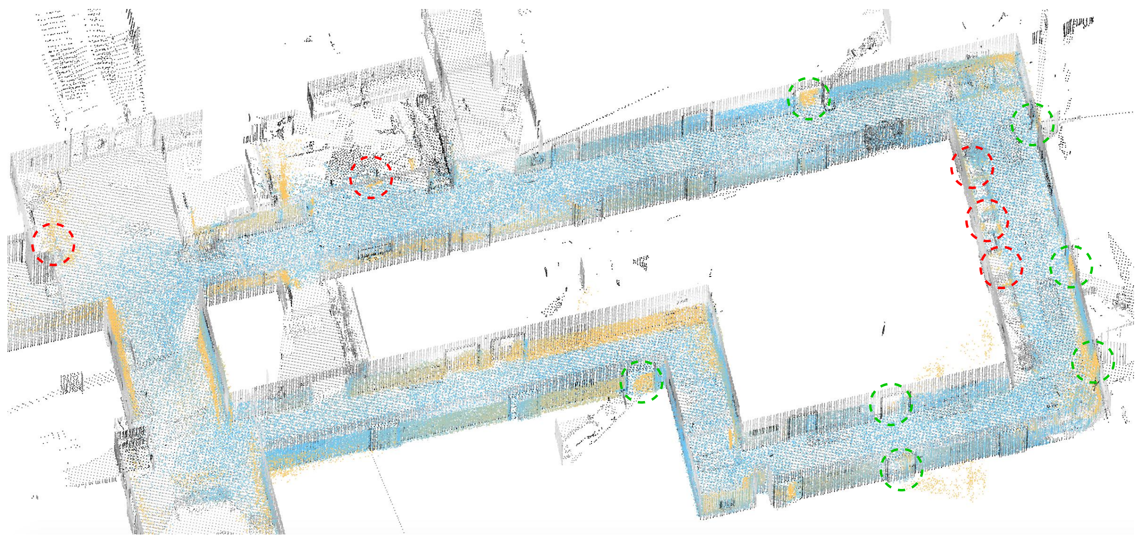
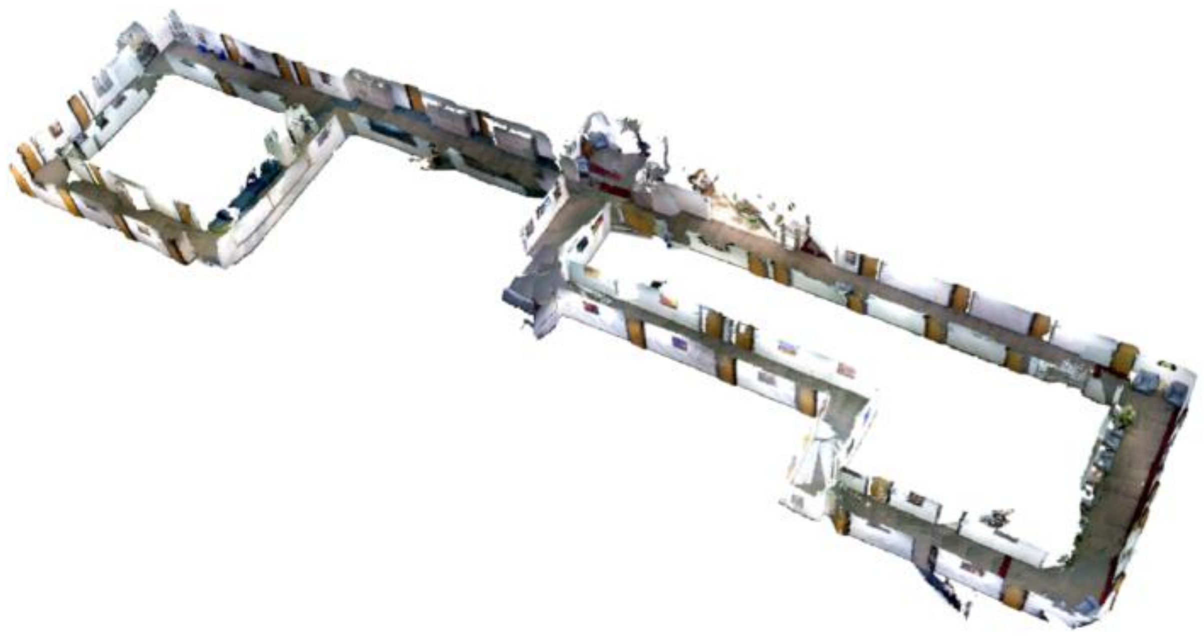
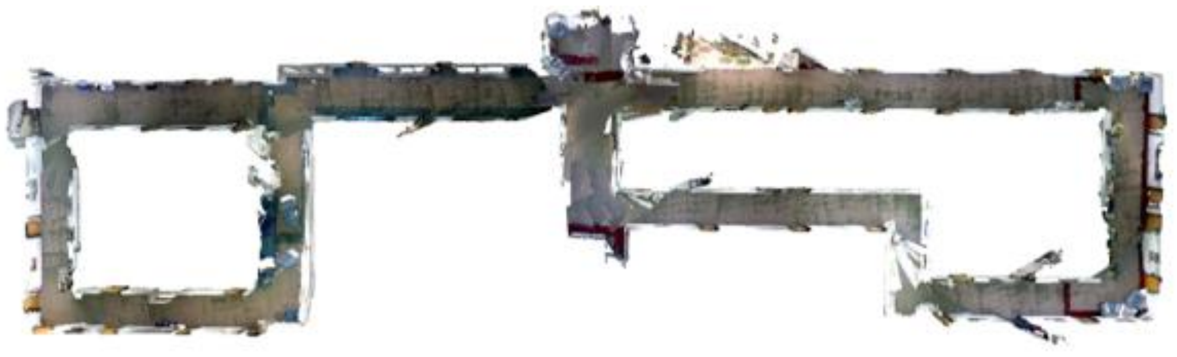


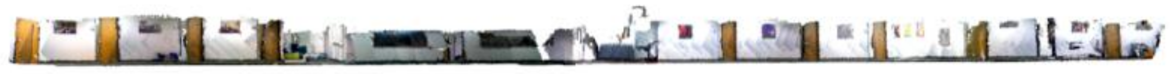
Figure 5.12: The registration of our output model (colored) onto the survey LiDAR ground truth model (black), both downsampled approximately 100 times to save calculation. The points with small deviation from ground truth are shown in blue, while larger RMSE is indicated by yellow. Notice that between recording these two datasets, some doors (green circles) and chairs (red circles) had been moved.



(a)

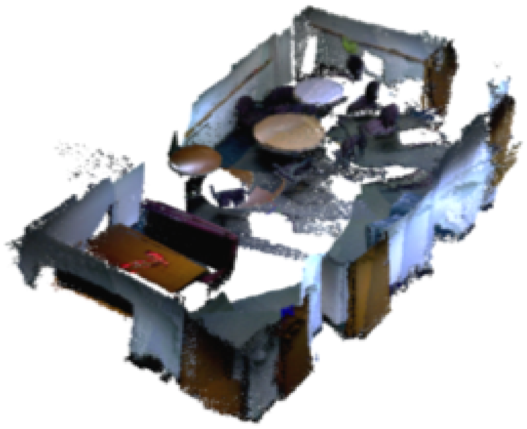


(b)



(c)

Figure 5.13: The reconstruction of a large indoor environment using DPI-SLAM system with structural constraints. The drift in the output 3D dense model (a) is significantly reduced, which can also be observed from the top (b) and side (c) views.


(a)

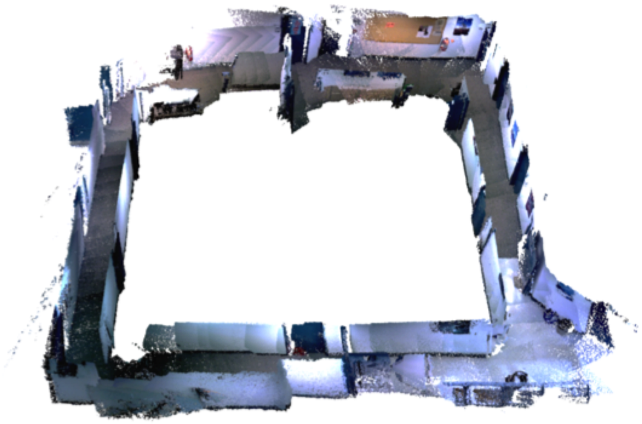
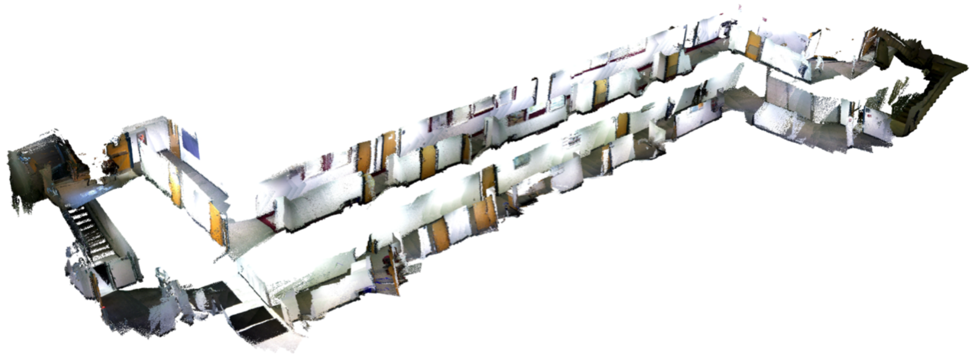

(b)

Figure 5.14: Dense 3D reconstruction of various indoor environments using our DPI-SLAM system. (a) An open space with three round tables. (b) Corridors in a loop with lighting changes.


(a)

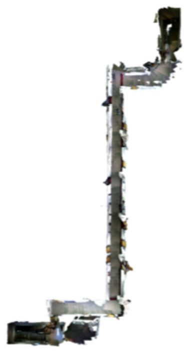

(b)

Figure 5.15: The reconstruction result of a large two-floor dataset with a loop using DPI-SLAM, where the dense model of the corridors and the stairs are clearly shown in (a). The top view (b) shows that the two long corridors on the different floors are well aligned with each other.

5.7.4 Discussion

From the experiments, we can conclude that adding inertial measurements improves planar SLAM, and further combining structural constraints can achieve the best reconstruction results. The efficiency of the dense map representation using submaps and planes is also demonstrated in these two real-time SLAM systems.

As for closing a loop until convergence, DPI-SLAM sometimes cannot run in real-time because having more IMU states in the global factor graph requires more iterations to converge (e.g. the loop closure process in Fig. 5.15). Fortunately, with structural constraints, the drift in the trajectory can be much smaller, and therefore the loop closure process can be faster (see Fig. 5.11). Also, if a converged result is not required immediately, DPI-SLAM can distribute the iteration steps to the later update steps of the global factor graph at each reference frame so that the system will not slow down during loop closing (e.g. the right loop in Fig. 5.13 gradually converges as the mapping process of the left part continues). Lastly, adding structural constraints into the pure planar SLAM system might also improve its results. However, without IMU measurements, the drift in rotation can be too large for the system to decide if there should be structural constraints or not. In this case, if a relaxed threshold is chosen, wrong structural constraints might be added into the system and cause more errors.

5.8 Conclusion

In this chapter, we discuss the approaches that are efficient for online dense 3D mapping, including submap formulation, SLAM with planes, and other related methods. We also integrate these mapping techniques with the FRO and FRIO algorithms proposed in Ch. 4 to build two SLAM systems, KDP-SLAM and DPI-SLAM, and evaluate their performances with real world datasets. Since the results are promising, we will develop novel ambiguity-aware SLAM systems based on DPI-SLAM, which will incorporate MH-iSAM2 (see Ch. 3) as the back-end optimizer to conduct multi-hypothesis state and map estimations under ambiguities.

Chapter 6

Ambiguity-aware Passive SLAM for Robust Dense 3D Reconstruction

6.1 Introduction

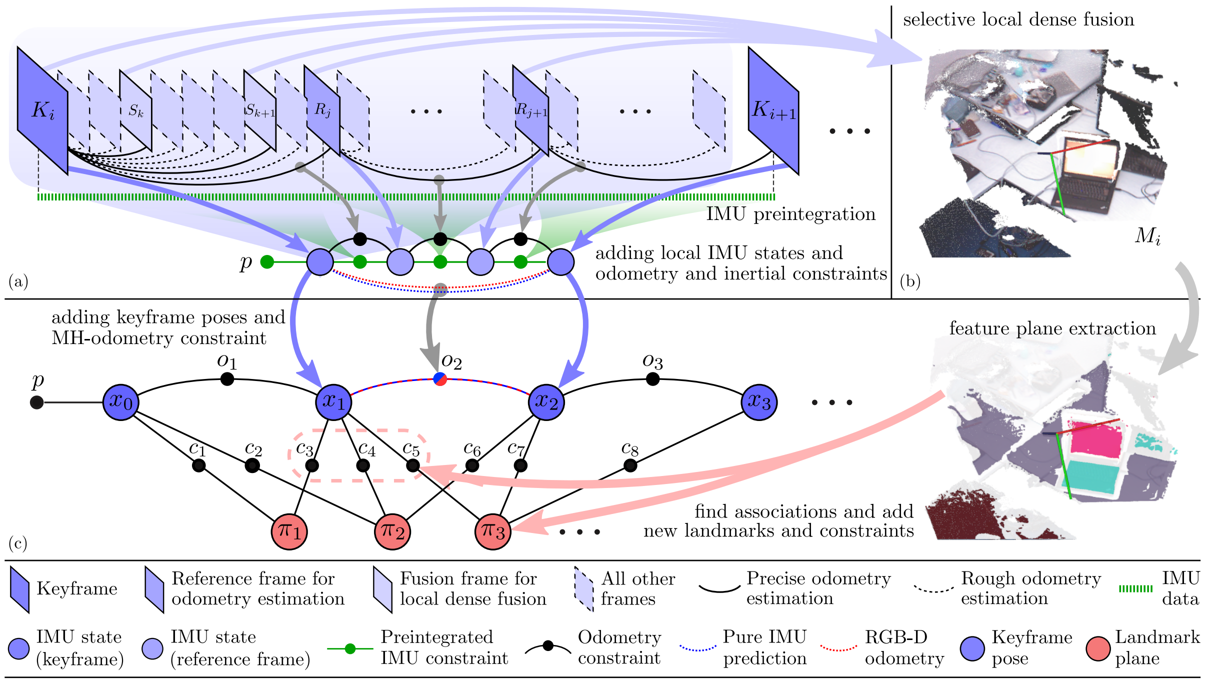
In this chapter, we demonstrate how to apply MH-iSAM2 (see Ch. 3) to solve a real-world passive SLAM problem in a multi-hypothesis fashion to achieve better robustness to ambiguities (defined in Sec. 1.2). To be more specific, we aim at modifying the existing DPI-SLAM system (see Ch. 5) into an ambiguity-aware planar-inertial SLAM (API-SLAM) system. The developed API-SLAM system is expected to be able to detect and model ambiguities in the input measurements from the RGB-D and inertial sensors (see Sec. 4.1), and output multi-hypothesis pose estimates as well as multi-hypothesis dense 3D maps online in real-time. We evaluate the robustness of the proposed API-SLAM system through real world experiments, and show how robustness can be improved comparing to the original DPI-SLAM system.

6.2 System Structure

The system structure of the proposed API-SLAM is as shown in Fig. 6.1. Similar to the system structure of DPI-SLAM, there are three main blocks: odometry estimation, local depth fusion, and global optimization with keyframe poses and planes. However, functions that detect and model ambiguities are incorporated into the odometry estimation, which adopts a fixed window factor graph optimization to estimate keyframe-to-keyframe odometry as well as generating the corresponding SMFs or MMFs (see Sec. 3.3) to model the estimated odometry (see Fig. 6.1-a). Notice that the prior of the first IMU state in each local interval is set to either the estimate from the last interval, or a default value if ambiguities are detected in the previous interval (so that the estimates might not be correct anymore).

Local depth fusion is almost the same as the original algorithm developed in Sec. 5.2 except that it will be terminated whenever an ambiguity is detected (see Fig. 6.1-b).

The global optimization is conducted using MH-iSAM2 (see Ch. 3), where all odometry factors between keyframes, all valid planar constraints, and all loop closure candidates are jointly



selective local dense fusion
K_i
S_k
S_{k+1}
R_j
R_{j+1}
K_{i+1}
IMU preintegration
p
adding local IMU states and odometry and inertial constraints
(a)
(b)
M_i
adding keyframe poses and MH-odometry constraint
feature plane extraction
o_1 o_2 o_3
x_0 x_1 x_2 x_3
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
π_1 π_2 π_3
find associations and add new landmarks and constraints
(c)
Keyframe
Reference frame for odometry estimation
Fusion frame for local dense fusion
All other frames
Precise odometry estimation
Rough odometry estimation
IMU data
IMU state (keyframe)
IMU state (reference frame)
Preintegrated IMU constraint
Odometry constraint
Pure IMU prediction
RGB-D odometry
Keyframe pose
Landmark plane

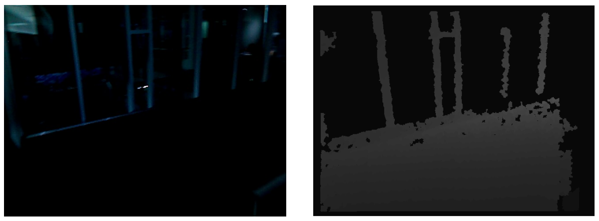
Figure 6.1: Modified from DPI-SLAM (see Fig. 5.5), the proposed API-SLAM system consists of the same three main modules: (a) RGB-D inertial odometry estimation, (b) local depth fusion, and (c) global optimization. However, additional functions based on local fixed window optimization are added into the odometry estimation pipeline to detect and model ambiguities, and the global optimization is conducted using MH-iSAM2 that can take different types of MMFs as input and solve for multi-hypothesis solutions if the ambiguities are temporarily unsolvable.

optimized to estimate the final states of the keyframe poses and planes, which may contain multiple hypotheses if the ambiguities are temporarily unsolvable (see Fig. 6.1-c).

6.3 Detecting and Modeling Ambiguities

In our approach, two types of ambiguities are handled explicitly: ambiguous odometry and ambiguous loop closures. For ambiguous odometry, any joint odometry estimation of the fast dense RGB-D odometry (see Sec. 4.2) and IMU preintegration [26] is regarded as ambiguous if any of the estimated IMU biases is larger than a threshold. In this case, we assume that either of them (RGB-D or IMU only) can still estimate the states correctly, and model their individual estimates as two individual modes in a type #1 MMF (see Sec. 3.3). The corresponding real world scenario can be considered as follows: either the RGB-D fail to track the motion due to the lack of texture, aggressive motion, or dynamic scenes (see Fig. 6.2), or the IMU preintegration is wrong due to bad prior of velocity or biases that are propagated from the previous local interval (see Sec. 6.2).

As for generating the individual modes for the type #1 MMF, the odometry estimate based on RGB-D data only is already computed, and the IMU-only odometry prediction can be computed by integrating all the preintegrated rotations (see Eq. 4.6) and translations (see Eq. 4.7) within


(a) Lack of texture (too dark)

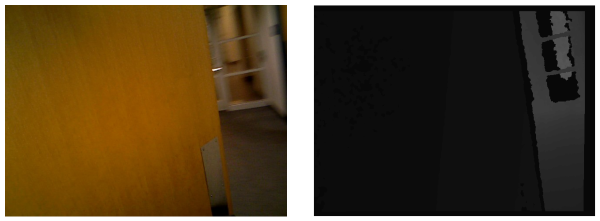

(b) Dynamic scene (opening a door)

Figure 6.2: Examples of ambiguities for fast dense RGB-D odometry (see Ch. 4).

the interval.

In the API-SLAM system, loop closures are detected and registered as described in Sec. 5.6. However, since we cannot be perfectly sure that any loop closure is not a false positive, we model every loop closure as a type #3 MMF (see Sec. 3.3) for generality. The MH-iSAM2 back-end solver will decide which of them to take into account based on the pruning strategy (see Sec. 3.6).

6.4 Multi-hypothesis Dense 3D Mapping

Following the same idea in Sec. 5.2, RGB-D frames within a local interval between two consecutive keyframes are merged into a local submap that is anchored to the corresponding keyframe pose. And since now every keyframe pose can contain multiple hypotheses, we can generate a multi-hypothesis dense 3D map by simply placing all the submaps according to each hypothesis of their keyframe poses in a global frame. In practice, the same submap only has to be processed once and stored as a single copy. As these procedures do not scale with the number of hypotheses, they are more efficient than processing and storing multiple maps for individual hypothesis (as described in the previous example in Fig. 3.4).

Besides submaps, we also use planes as landmarks (see Sec. 5.3, 5.4, and 5.5) in the global MH-iSAM2 optimization. Notice that for simplicity, only the planar measurements that agree in all existing hypotheses will be added into the MHFG as SMFs. The additional information provided by the planar SMFs can potentially help the pruning algorithm (see Sec. 3.6) to tell the unlikely hypotheses apart from the likely ones.

6.5 Experimental Results

6.5.1 Settings

The API-SLAM system is implemented in C++ and executed on a laptop with an Intel Core i7-8850H processor. Our implementation, which has not been optimized, runs roughly at 30 fps. However, if occasionally one frame takes too long to process, the very next frame will be skipped to maintain the overall real-time operation.

Since there was no suitable public RGB-D-inertial SLAM dataset for the evaluation of ambiguity-awareness and robustness, we collected our own dataset with real-world challenges that can result in ambiguities. Then, we compare the performances of API-SLAM against the baseline single-hypothesis DPI-SLAM system (see Sec. 5.7.3). Notice that throughout this experiment, we use the same sets of thresholds in the adopted appearance-based loop closure detection algorithm [29] in both API-SLAM and DPI-SLAM, which is different from the results of KDP-SLAM and DPI-SLAM shown in Ch. 5 that are based on fine-tuned thresholds. This setup allows a fair comparison of robustness between the two systems.

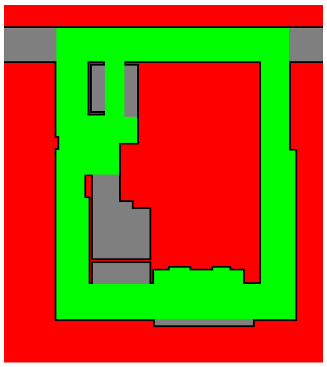
6.5.2 Real World Datasets

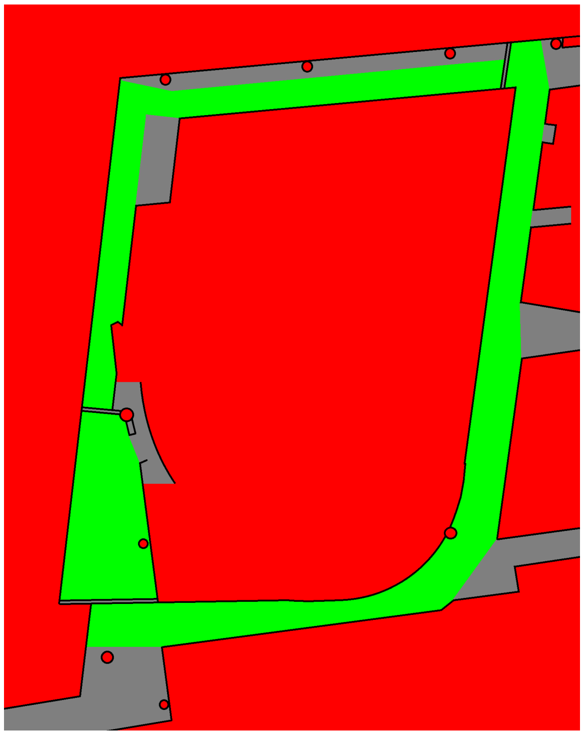
A qualitative evaluation on robustness of the proposed API-SLAM system can be done by visually comparing its output 3D models with the 3D models reconstructed by DPI-SLAM. However, it is still desired to have a properly designed quantitative evaluation to verify the improvement of API-SLAM over DPI-SLAM on ambiguity-awareness and robustness.

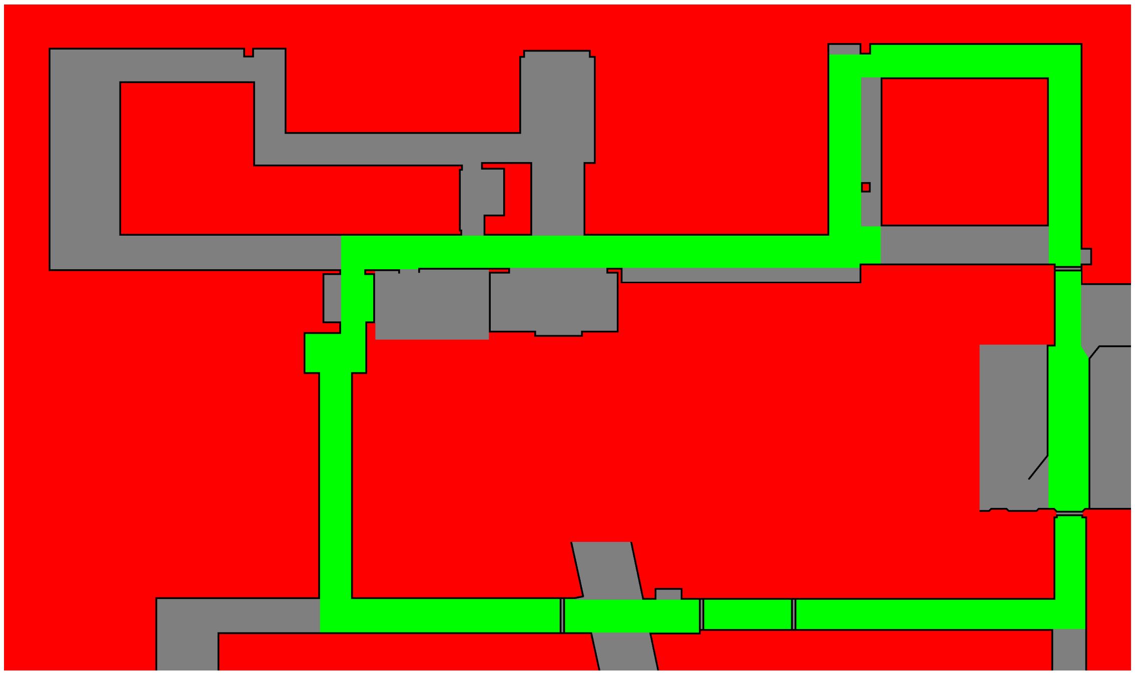
However, one big challenge is that the survey LiDAR scans or something similar that can serve as ground truth 3D models of various indoor environments are hard to obtain. Therefore, we come up with a new evaluation framework that uses the 2D floor plans of the buildings as reference. We first label a floor plan with *free space*, *unseen space*, and *uncertain space* (see Fig. 6.3), and project the output 3D model onto the 2D ground plane using the known gravity direction. Then, we uniformly sample 2D points from the covered area of the projected model and the free and uncertain space in the grid map respectively. Finally, we conduct a two-way registration and evaluation process (see Fig. 6.4 and Fig. 6.5) to generate an overall error that can represent the robustness of the tested methods.

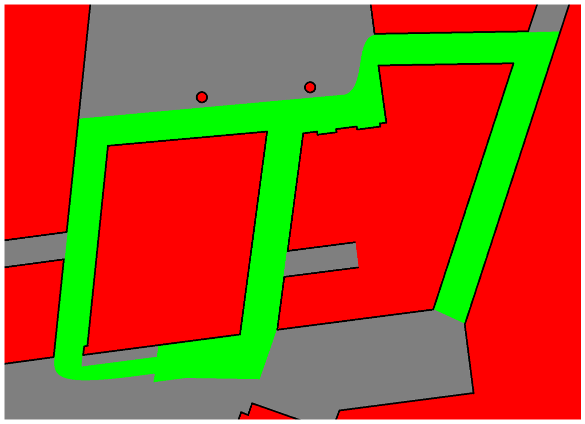
The two-way registration and evaluation process is described as below: First, we use 2D point-to-point ICP to find a 3 DoF rigid body transformation that registers the resampled 2D model to the union of free and uncertain space (green and gray area) of the corresponding labeled floor plan, and compute a *forward registration error* between each closest point pairs. Then, we register only the free space (green area) in the floor plan to the model using the same ICP algorithm, and compute a *backward registration error* similarly. As each of the forward or backward error only describe how similar the model and the floor plan are in one direction, averaging the two of them into an *overall registration error* can better describe the similarity between the two.

Since the error reflects how bad the output model distorts due to bad measurements and drifts, it can be a good representation of the robustness of the method in general. We can also think of this evaluation method as using a number to summarize the visual similarity between the reconstructed model and the ground truth floor plan regarding the overall geometric structure. Moreover, as long as the assumption of known gravity direction stands, the evaluation on the projected 2D plane can represent the robustness of the the output 3D models properly and efficiently. In our experiments, both *mean absolute error (MAE)* and *root mean square error (RMSE)* are computed for a better sense of how the errors between each pair of points are distributed. We also show the histograms of each of these two types of errors in Fig. 6.4 and Fig. 6.5 along with the false-colored registration results. Some of the examples of the online reconstructed 3D models of API-SLAM are visualized in Appendix B.1.


(a) From NSH 2F


(b) From GHC 4F


(c) From NSH 4F


(d) From GHC 6F (south)

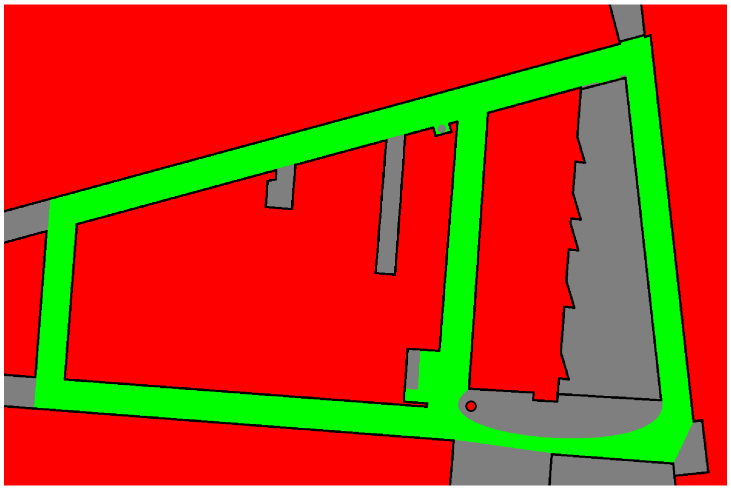
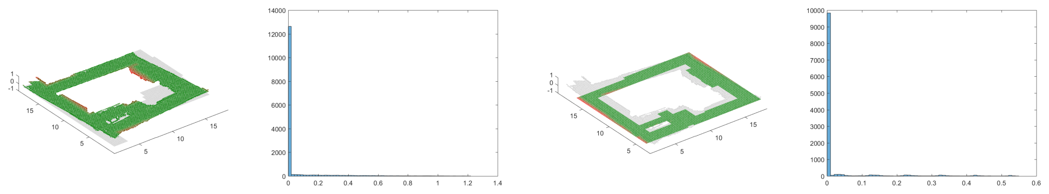
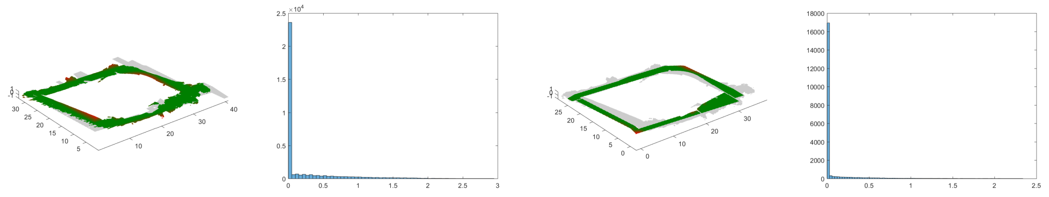

(e) From GHC 6F (north)

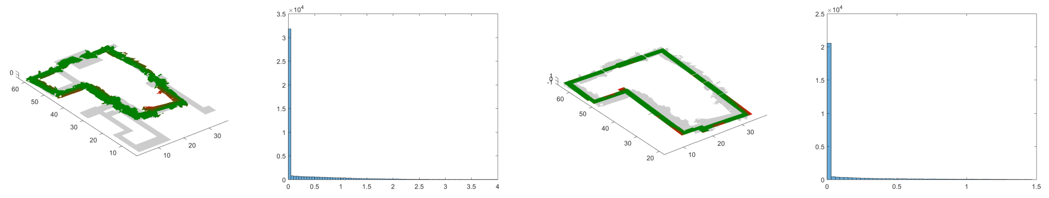
Figure 6.3: The labeled floor plans of the campus buildings (NSH/GHC) of Carnegie Mellon University (CMU) are taken as ground truth in our quantitative evaluation for robustness. Each sampled point in the floor plan is labeled as free (green), unseen (red), or uncertain (gray).



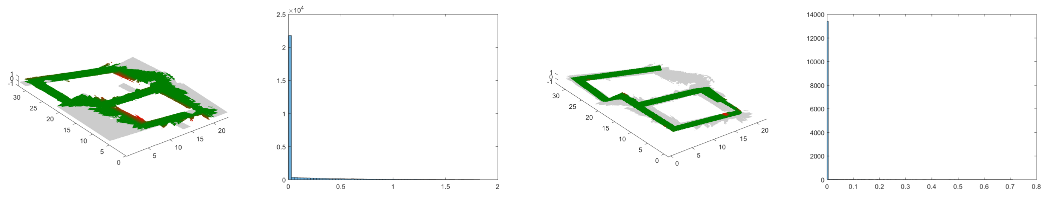
(a) From NSH 2F



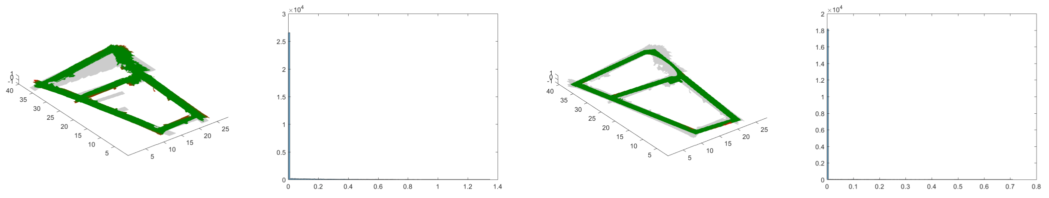
(b) From GHC 4F (south)



(c) From NSH 4F

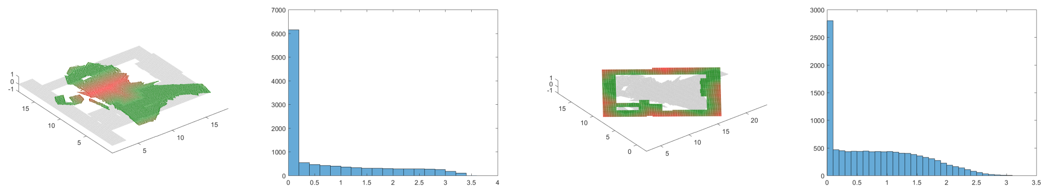


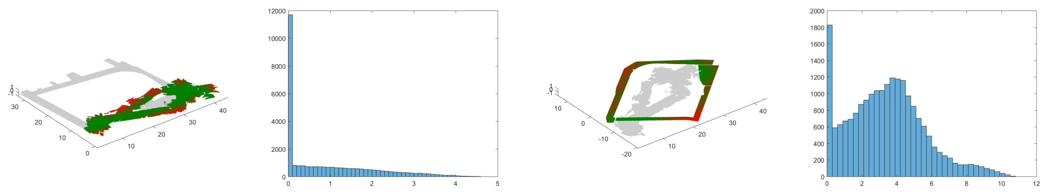
(d) From GHC 6F (south)

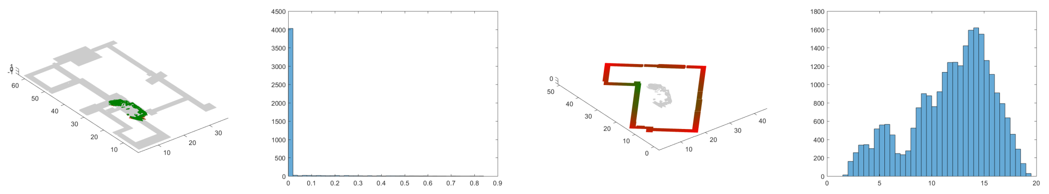


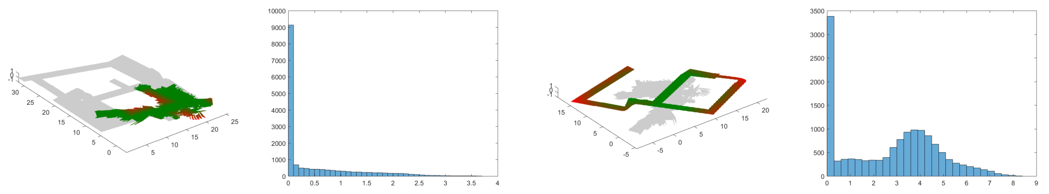
(e) From GHC 6F (north)

Figure 6.4: Examples of the two-way ICP registration for robustness evaluation of the proposed API-SLAM system. The left two columns show the results of forward registration: models (colored) to the target floor plan areas (gray), and their corresponding histograms of absolute distance errors (unit: m). The right two columns show the results of backward registration: free space in the floor plans (colored) to the models (gray), and their corresponding histograms of absolute distance errors as well. Some of the online reconstructed 3D models of these datasets can be found in Appendix B.1.


(a) From NSH 2F


(b) From GHC 4F (south)


(c) From NSH 4F


(d) From GHC 6F (south)

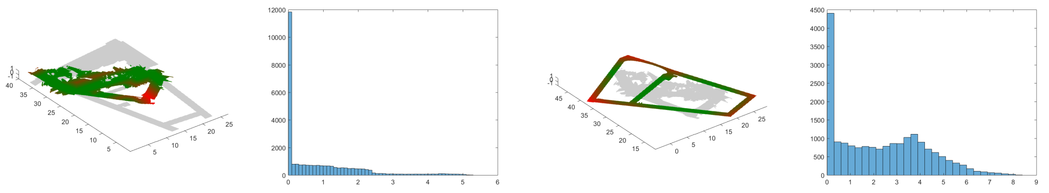

(e) From GHC 6F (north)

Figure 6.5: Examples of the two-way ICP registration for robustness evaluation of the baseline DPI-SLAM system. The registration results and their histograms are shown in the same format as in Fig. 6.4.

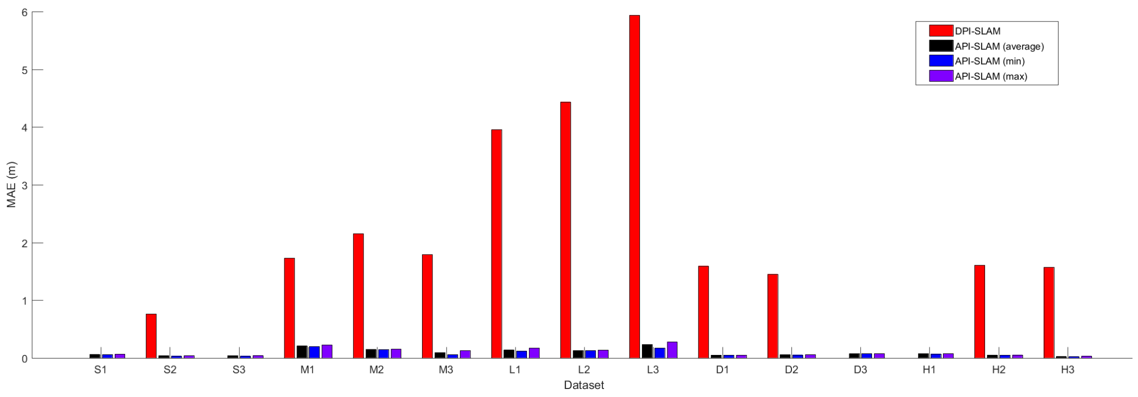
To achieve a thorough evaluation, we select 5 different environments with various sizes and topologies (see Fig. 6.3 again) and collect 3 sequences of RGB-D and inertial data in each of them. Notice that in Fig. 6.3-b and Fig. 6.3-c, the trajectories go through 2 and 5 originally closed doors respectively, and the human user opens each of these doors while pointing the RGB-D sensor at it, which is an intentionally created source of ambiguity in odometry estimation. Other ambiguity sources (e.g.: walking passersby in the scene, lack of features, strong rotation, false positive loop closures, etc) occur more randomly during the data collection. The results in Fig. 6.6 show that API-SLAM can still reconstruct reasonable structures of the target environments even with all these ambiguities. On the other hand, DPI-SLAM fails to deal with the ambiguities and therefore cannot generate any meaningful result. Therefore, we can conclude that API-SLAM outperforms DPI-SLAM significantly in robustness when multiple ambiguities occur.

6.5.3 Discussion

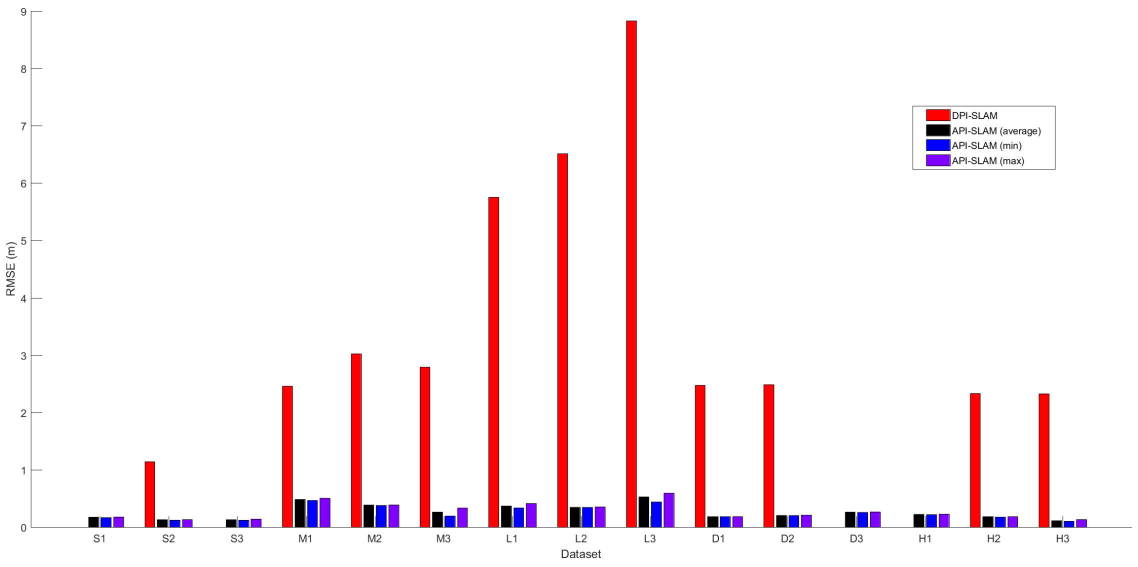
We find that in most cases, the online reconstruction results of DPI-SLAM start to distort when one wrong odometry estimation takes place, and soon become severely distorted due to wrong loop closure registrations. Even though it is possible to filter out some of the wrong loop closure candidates if adding a more strict geometric consistency check, it still cannot guarantee perfect outlier rejection in practice. As a result, we keep the DPI-SLAM algorithm unchanged as in Ch. 5, and focus our experiment more on comparing the overall robustness of the two systems.

Even though the proposed API-SLAM achieves much better robustness than the baseline DPI-SLAM under ambiguities, it can still be improved and extended in several ways. First of all, adding more sensors under the same ambiguity-aware framework using MH-iSAM2 but with different ambiguity detection and modeling approaches might further improve the robustness of the system. Adding prior knowledge such as a motion model or structural constraints as ambiguous input and let the MH-iSAM2 back-end decide whether they are valid is also an interesting idea that is worth trying. Moreover, if a bad trajectory is taken, it could be possible that no sufficient loops are closed to disambiguate the ambiguities, especially the ambiguous odometry. In the current system setup, the hypotheses with the MMF modes resulting from pure IMU propagation are more likely to be preserved after pruning because their larger uncertainties would result in smaller system errors in general (after jointly optimizing with planar constraints). It is debatable whether a better approach to deal with this situation exists, yet it is clear that being able to add loop closures into the system whenever needed is desirable. Therefore, how to actively revisit the mapped area and close loops under the multi-hypothesis framework will be one focus of study in Ch. 7.

Finally, projecting models into 2D inevitably loses most of the details within and only preserves the high-level topology and geometry. Therefore, the evaluation method based on it is only good to evaluate robustness, not accuracy. However, this evaluation method is efficient and easy to extend. For example, simply modifying Fig. 6.3-c to Fig. 6.7 will allow us to test a different dataset (see Fig. 6.8).



(a) MAE



(b) RMSE

Figure 6.6: Bar charts of the overall registration error for robustness evaluation. Smaller errors within the same batch represent better robustness. However, because some of the results from DPI-SLAM are severely distorted, the absolute values of the errors do not have much actual meanings in those cases. In addition, some of the bars are missing due to the numerical problem in optimization, which usually implies large distortion in the output model resulting from outliers in the optimization.

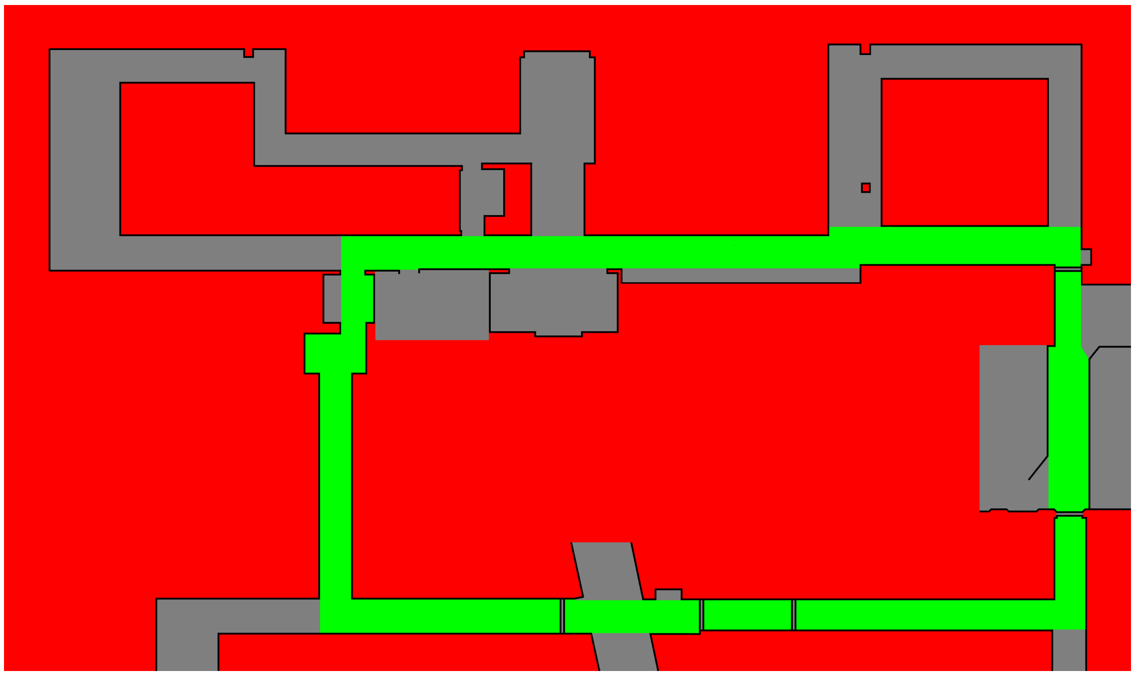
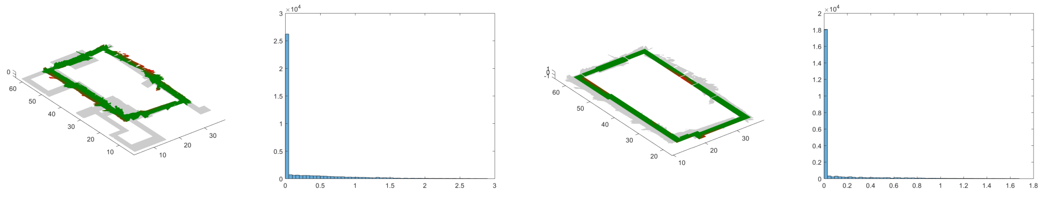
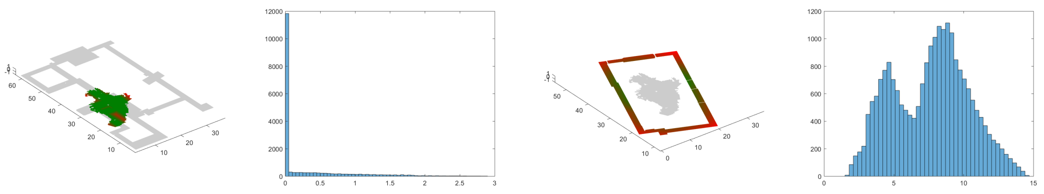


Figure 6.7: The labeled floor plans modified from the one in Fig. 6.3-c. Notice that because of the change in the trajectory of the dataset, some free and uncertain areas are labeled differently on the upper-right part of the floor plan.



(a) Result of one hypothesis from API-SLAM



(b) Result from DPI-SLAM

Figure 6.8: Example of evaluating a new dataset with a little modification on the originally labeled floor plan. The registration results and their histograms are shown in the same format as in Fig. 6.4.

6.6 Conclusion

In this chapter, we developed the ambiguity-aware planar inertial SLAM (API-SLAM) system based on the previously developed DPI-SLAM in Ch. 5, which demonstrates a real world SLAM application of the proposed MH-iSAM2 back-end solver. The ambiguity in odometry estimation is detected when the measurements from the two input sensors (RGB-D camera and IMU) do not agree, and the ambiguity in loop closure is regarded as part of its nature that always exists. Both types of ambiguities are modeled explicitly using the two types of MMFs in MH-iSAM2 respectively, and planar constraints are added to potentially help disambiguate the ambiguities in addition to correcting the drift. The experimental results show that API-SLAM is able to reconstruct reasonable 3D models in multiple hypotheses even under ambiguities, which significantly outperforms the baseline DPI-SLAM algorithm that can only handle a single hypothesis. To further study how to make use of the multi-hypothesis state and map estimates results in a robotic task, we will work on extending API-SLAM into an active SLAM system in Ch. 7.

Chapter 7

Ambiguity-aware Active SLAM Based on Multi-hypothesis State and Map Estimations

7.1 Introduction

In this chapter, we study how multi-hypothesis state and map estimates can be used in active SLAM to improve robustness to ambiguities (defined in Sec. 1.2). The fundamental goal of the proposed *ambiguity-aware active SLAM framework* is similar to existing active SLAM solutions: exploring the unknown area of the surrounding environment while reducing the error of the reconstructed map. Therefore, the ambiguity-aware active SLAM system consists of the same two main modules as other active SLAM systems: exploration and active loop closing. However, the needs of considering multiple probable solutions at the same time and actively reducing the ambiguity among these solutions would be the main challenges of developing the two modules.

The goal of exploration is to move the robot in certain ways so that new information of the environment can be observed to update the states and the map. Even though optimizing the information gain and motion cost based on a global map theoretically optimizes exploration efficiency [7][94], it might not be the best choice for our multi-hypothesis framework since maintaining all the global maps for each hypothesis can be very inefficient in both memory and computation. As a result, we develop a multi-hypothesis exploration algorithm based on *ambiguity-free submaps* (an example of submap can be found in Sec. 5.2) to achieve better efficiency.

On the other hand, active loop closing is typically responsible for finding mapped places to revisit for loop closures that correct the accumulated drift and uncertainty [116][72][11][131]. However, because the exponential growth of the number of hypotheses can slow down the entire system drastically, we aim at bounding the number of hypotheses based on the constraints of the system while keeping track of the correct hypothesis (assuming only one is correct) under our multi-hypothesis framework. Therefore, the goal of our active loop closing algorithm is to find certain places to revisit so that the detected loop closures can provide sufficient information to distinguish and prune enough wrong hypotheses to reduce computational cost, or even preserve

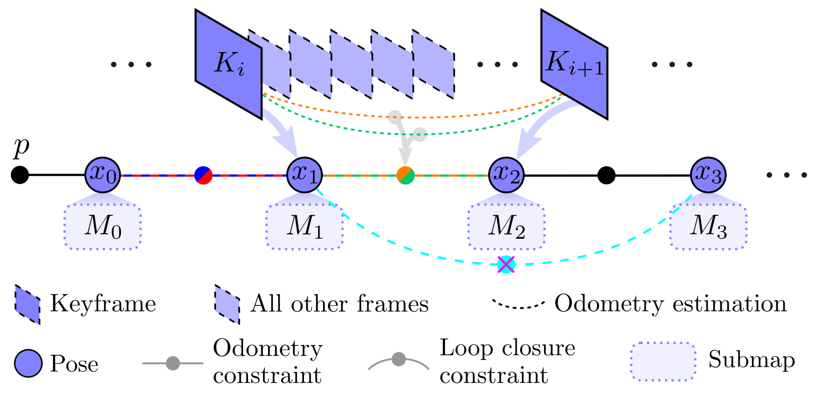


Figure 7.1: The system structure of a desired MH-SLAM algorithm.

the correct hypothesis only.

In addition to the two main modules, a simple path planning algorithm is implemented for exploration and active loop closing in each hypothesis, which can be replaced by a state-of-the-art planner as needed. We will introduce how a predefined passive *multi-hypothesis SLAM (MH-SLAM)* system (see Sec. 7.2) and these three active SLAM modules interact with each other in Sec. 7.3, and discuss the details of each of them in Sec. 7.4, 7.5, and 7.6 respectively.

7.2 Multi-hypothesis SLAM Preliminaries

The proposed ambiguity-aware active SLAM framework is designed based on a passive multi-hypothesis SLAM (MH-SLAM) system (see Fig. 7.1) that satisfies several properties. While the API-SLAM system introduced in Ch. 6 is already a good example of MH-SLAM, we would still like to list the two main properties as follows for generalization.

7.2.1 Using MH-iSAM2

The MH-SLAM in the ambiguity-aware active SLAM framework should adopt MH-iSAM2 (see Ch. 3) as its back-end solver. Ambiguous measurements are modeled as MMFs f_r^{M} with corresponding types and arbitrary number of modes (see Sec. 3.3.2), and the overall and local hypotheses (h and $h^{\{r\}}$, see Sec. 3.4) of all MHVs can be optimized accordingly and associated with each other through the Hypo-tree. In addition, the unlikely hypotheses are pruned during each update so that all the remaining hypotheses are regarded as highly probable.

7.2.2 Ambiguity-free Submaps

We adopt a keyframe-based framework similar to KDP-SLAM (see Sec. 5.7) in our MH-SLAM, which selects a new *keyframe* K_i whenever the overlapping scene between the current frame and the previous keyframe is less than a threshold in any hypothesis. We further assume that all ambiguities between any two keyframes can be summarized into a type #1 MMF (different measurements among the same set of variables, which is for ambiguous odometry estimation

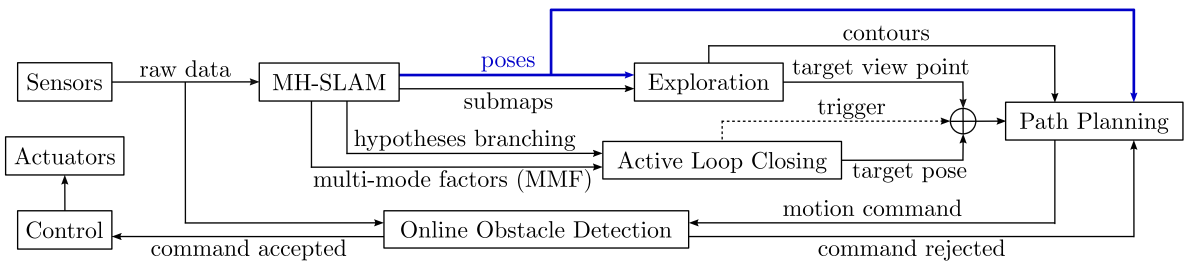


Figure 7.2: Block diagram of the ambiguity-aware active SLAM framework. Note that the poses $\mathtt{T}_i$ (blue arrows) are the only multi-hypothesis information passed between modules.

here) or type #3 MMF (whether the measurement is valid or not, which is for ambiguous loop closure here). As a result, only the multi-hypothesis poses $\mathtt{T}_i$ of all K_i are optimized globally in MH-iSAM2 for simplicity and efficiency. Moreover, we can generate an ambiguity-free local dense submap M_i at each K_i, by fusing the raw range or depth information from the adopted sensor(s) between consecutive keyframes into a less noisy, outlier-free 3D map representation (e.g.: *local depth fusion* in [39]). As each K_i is anchored to $\mathtt{T}_i$, the occupancy information at any global location in any h can be preserved efficiently in all nearby $\mathtt{T}_{i,[h]}$ and their M_i without the need of generating individual global maps for each h.

Finally, we assume that the environment we want to map lies roughly on a horizontal plane (e.g.: one floor of a building), and the sensors also move roughly on a horizontal plane (e.g.: mounted on a drone or hand-held). These assumptions simplify the active SLAM problem into 2D (although the states and maps can still be 3D), which help us to focus on making use of the multi-hypothesis estimates in the ambiguity-aware active SLAM efficiently.

7.3 Ambiguity-aware Active SLAM Framework

Based on the MH-SLAM discussed in Sec. 7.2, our ambiguity-aware active SLAM framework is developed with three other main modules: exploration, active loop closing, and path planning (see Fig. 7.2). In every iteration, the exploration module takes the submaps M_i and multi-hypothesis poses $\mathtt{T}_i$ as inputs from the MH-SLAM module, and generates a *target view point* P^* that aims at exploring unknown area (see Sec. 7.4). Meanwhile, the active loop closing module will find a *target submap* M^* to revisit based on the hypotheses branching and modes in each MMF (see Sec. 7.5). Either P^* or M^* will be passed to the path planning module as a target pose depending on whether the active loop closing is triggered or not, and a *motion command* will be generated accordingly (see Sec. 7.6). The motion command has to be verified as valid (no conflict with the current measurements) by an online obstacle detection module before being conducted.

This framework is efficient since both P^* and M^* are single outputs anchored to the multi-hypothesis poses of their corresponding keyframes, and their global poses can be updated inherently when the keyframe poses are updated in each hypothesis (see Sec. 7.2). Moreover, as exploration and path planning are the only two modules that deal with the multi-hypothesis poses $\mathtt{T}_i$ directly, they are both designed to operate on M_i so that expensive computations such

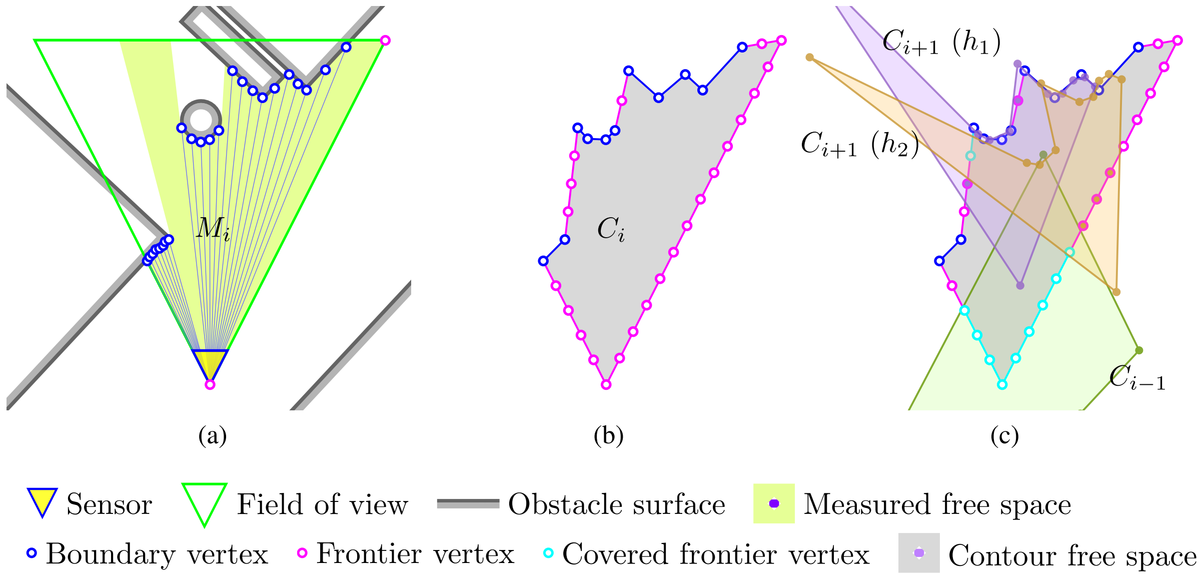


Figure 7.3: The extraction and multi-hypothesis update of a local contour C_i. (a) Boundary vertices $\mathbf{v}^{\mathrm{B}}_{\{i\},p}$ are extracted from the submap M_i, and two frontier vertices $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ are added at the sensor origin and the maximum range on the right edge. (b) More $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ are interpolated with equal spacing, and some $\mathbf{v}^{\mathrm{B}}_{\{i\},p}$ are removed to smooth the edge. (c) The frontier vertices can be updated as covered ($\mathbf{v}^{\mathrm{F}}_{\{i\},p} \rightarrow \mathbf{v}^{\mathrm{F,c}}_{\{i\},p}$) based on the neighboring submaps and their relative pose estimates in all hypotheses.

as maintaining multi-hypothesis global maps can be avoided.

7.4 Multi-hypothesis Exploration

Based on the assumptions in Sec. 7.2, there is only one single hypothesis within each ambiguity-free submap, yet the pose of the submap might contain multiple hypotheses. We take advantage of this assumption to develop the multi-hypothesis exploration algorithm by conducting as many computations as possible within each submap, and only deal with the complex multi-hypothesis estimates when needed.

7.4.1 Local Contours Extraction from Submaps

From each submap M_i, we extract a *local contour* C_i to represent the local free space and encode the *frontier* [135] and obstacle boundary information. To compute C_i, we first extract boundary vertices $\mathbf{v}^{\mathrm{B}}_{\{i\},p}$ directly from the fused depth/range measurements in M_i on a given height assuming that the gravity direction $\mathbf{g}_i$ in the local coordinates of M_i is known (see Fig. 7.3-a). Then, a frontier vertex $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ is added at the sensor origin, while one or two other $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ might be added at the maximum sensor range along the edges of the sensor's *field of view (FoV)* if no $\mathbf{v}^{\mathrm{B}}_{\{i\},p}$ is

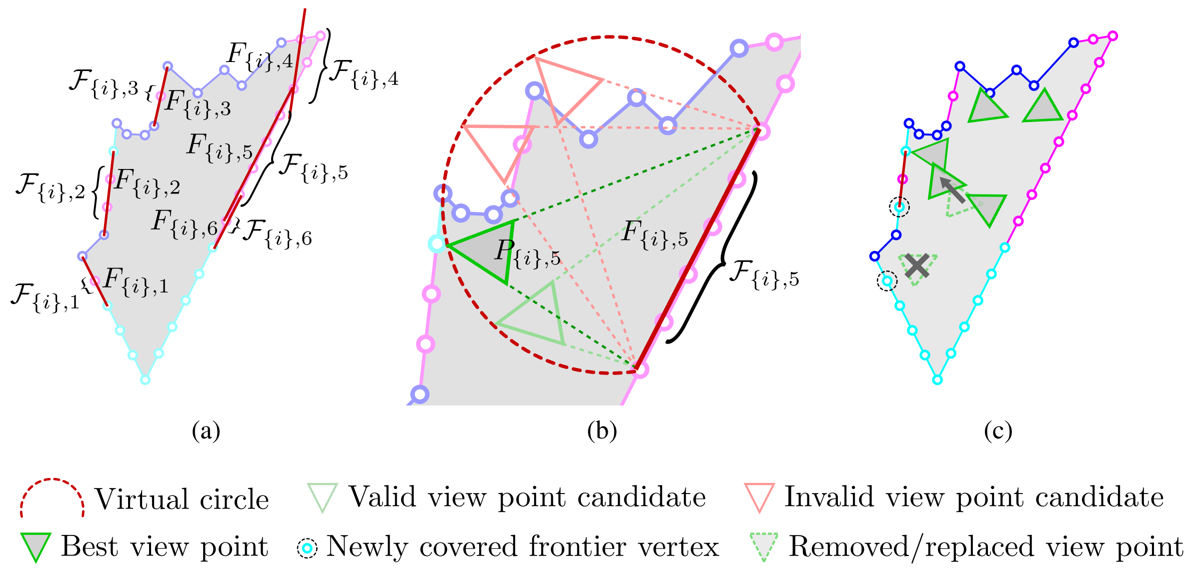


Figure 7.4: The extraction and update of view points $P_{\{i\},k}$. (a) Frontier segments $F_{\{i\},k}$ are defined based on each sequence of uncovered frontier vertices $\mathcal{F}_{\{i\},k}$. (b) To cover each $F_{\{i\},k}$, candidate view points can be sampled on the virtual circle (only on one side) since all the *inscribed angles* that subtend the same arc have the same angle, which can be regarded as the FoV of the sensor. (c) Each selected best view point $P_{\{i\},k}$ can be updated or removed based on the updates of $\mathcal{F}_{\{i\},k}$.

extracted near them. After sorting all these vertices in the clockwise angular ordering, we interpolate more frontier vertices $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ between each pair of consecutive vertices that are farther apart than a threshold (see Fig. 7.3-b). Finally, C_i is refined by removing some $\mathbf{v}^{\mathrm{B}}_{\{i\},p}$ to smooth its edges.

Every C_i has a *pseudo timestamp* t_i, which is initialized as the timestamp of its corresponding K_i. Any two contours C_i and C_j can be added as *neighbors* of each other if t_i and t_j are close enough, and at least the distance $d_{ij,[h]}$ between their poses ($\mathtt{T}_{i,[h]}$ and $\mathtt{T}_{j,[h]}$) in one of the hypotheses h is within a threshold. Whenever a loop between K_j and K_i (with $j > i$) is closed in all hypotheses, t_i will be updated as t_j so that every newly created nearby contour and C_i can become neighbors of each other. The same update is also applied to all the neighbors of C_i for the same reason.

Whenever the number of neighbors of any C_i increases, we check though each $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ if it is inside at least one other neighboring contour in each hypothesis. If so, that $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ will be updated as *covered*, which is denoted as $\mathbf{v}^{\mathrm{F}}_{\{i\},p} \rightarrow \mathbf{v}^{\mathrm{F,c}}_{\{i\},p}$ (see Fig. 7.3-c). Since the covered frontier vertices $\mathbf{v}^{\mathrm{F,c}}_{\{i\},p}$ no longer represent the horizon of the explored region in any hypothesis, we will only use the uncovered $\mathbf{v}^{\mathrm{F}}_{\{i\},p}$ to decide future view points for exploration (see Sec. 7.4.2).

7.4.2 View Points Selection

In each C_i, a set of valid view points $\mathcal{P}_i = \{P_{\{i\},k},\ k \in \mathbb{N}\}$ is found that can observe all the uncovered $\mathbf{v}^{\mathrm{F}}_{\{i\}p}$ and the unknown space beyond them. Since C_i summarizes the local effects from all hypotheses (see Sec. 7.4.1), each view point $P_{\{i\},k}$ can be computed only once as a pose $\mathtt{T}_{\{i\}k}$ in the local coordinates of M_i, and transformed to different global or relative poses in each hypothesis using $\mathtt{T}_{i,[h]}$ (see Sec. 7.2).

Starting from the $\mathbf{v}^{\mathrm{F}}_{\{i\},0}$ at the sensor origin and following the clockwise order, we define one *frontier segment* $F_{\{i\},k}$ for each sequence of consecutive *uncovered frontier vertices* $\mathcal{F}_{\{i\},k}$, which is a line segment connecting the two frontier vertices at the two ends of $\mathcal{F}_{\{i\},k}$ with certain margin extended on both sides (see Fig. 7.4-a). Any frontier segment $F_{\{i\},k}$ with its length $l_{\{i\},k}$ exceeding a threshold is divided into several shorter ones until all of the $F_{\{i\},k}$ satisfy the length threshold. Then, we adopt the *inscribed angle theorem* to sample view point candidates on the *virtual circle* defined by the frontier segment $F_{\{i\},k}$ (see Fig. 7.4-b). A view point candidate is valid only if it is inside C_i and its view is not blocked by any edge of C_i. Finally, the best view point $P_{\{i\},k}$ for $F_{\{i\},k}$ is naively selected as the valid candidate with its viewing angle most perpendicular to $F_{\{i\},k}$. If all candidates are invalid, $F_{\{i\},k}$ will be divided into two shorter frontier segments, and the same algorithm will be repeated onto each of them until every $\mathcal{F}_{\{i\},k}$ finds a $P_{\{i\},k}$.

When any $\mathbf{v}^{\mathrm{F}}_{\{i\}p}$ is updated to $\mathbf{v}^{\mathrm{F,c}}_{\{i\},p}$ (see Sec. 7.4.1), it will be removed from $\mathcal{F}_{\{i\},k}$, and $F_{\{i\},k}$ will also be updated if at least one of the two ends of $\mathcal{F}_{\{i\},k}$ is affected. Then, a new view point $P'_{\{i\},k}$ will be computed according to the new $F'_{\{i\},k}$ and substituted for $P_{\{i\},k}$ (see Fig. 7.4-c). If all $\mathbf{v}^{\mathrm{F}}_{\{i\}p}$ in $\mathcal{F}_{\{i\},k}$ are removed, $P_{\{i\},k}$ will also be removed from $\mathcal{P}_i$.

Very occasionally in the view point finding or updating steps, if no single valid view point can be found for a frontier segment that can no longer be divided (only contains one frontier vertex), we simply label this frontier vertex as covered and ignore it in all future computations or updates on this view points. Experiments show that skipping one frontier vertex in this corner case does not affect the coverage or performance of the overall algorithm much since neighboring contours are very likely to cover the same frontier with valid view points.

7.4.3 Exploration Tree

To choose a target view point $P^*_{\{i\}}$ from all $\mathcal{P}_i$ in all C_i to visit, we use an *exploration tree* data structure to sort $\mathcal{P}_i$ implicitly, and choose $P^*_{\{i\}}$ from the selected $\mathcal{P}_i$. Every node N_i in the tree represents one $\mathcal{P}_i$ except for the root node N_{R}, and every new node will be added as the first child of the latest node that has its contour updated (see Sec. 7.4.1) in the same iteration (see Fig. 7.5-a). If any $\mathcal{P}_j = \{\emptyset\}$ due to the view point removal process (see Sec. 7.4.2), its corresponding node N_j will be removed from the tree (see Fig. 7.5-b).

After updating the tree, the first view point $P_{\{i\},1}$ in $\mathcal{P}_i$ of the first child N_i of N_{R} is naively selected as the target view point $P^*_{\{i\}}$ (see Fig. 7.5), and its pose $\mathtt{T}^*_{\{i\}}$ in the local coordinates of the submap M_i is defined as the *local target pose* and passed to the path planning module (see Sec. 7.6). If N_{R} is the only remaining node in the tree, which means that no view point has to be visited, we can conclude the exploration and terminate the entire active SLAM algorithm.

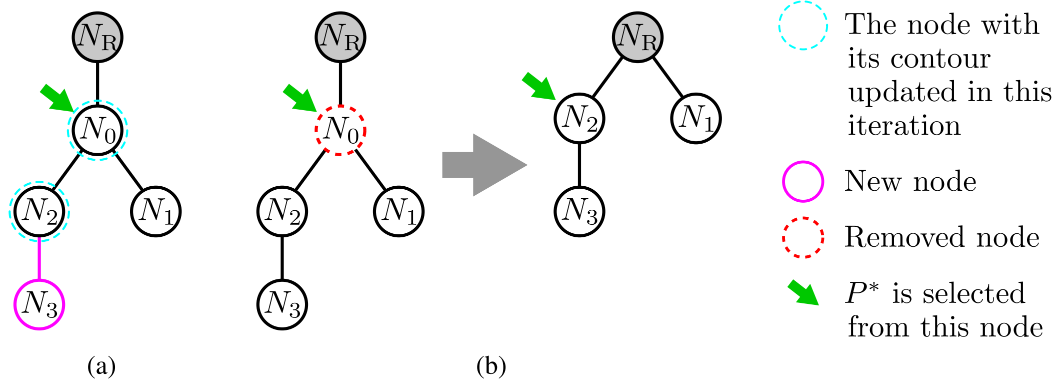

N_R
N_0
N_1
N_2
N_3
(a)
(b)
The node with its contour updated in this iteration
New node
Removed node
P^* is selected from this node

Figure 7.5: The construction and update of the exploration tree. (a) A new node N_3 that represents the new set of view points $\mathcal{P}_3$ is added. (b) If C_3 covers all the remaining $\mathbf{v}^{\mathrm{F}}_{\{0\}p}$ in C_0, $\mathcal{P}_0$ will be empty and N_0 will be removed.

7.5 Active Loop Closing for Correct Pruning

7.5.1 Triggering

Assume that n_{limit} is the upper bound of the number of hypotheses our entire system can handle (see Sec. 6.1) and n is the number of equally likely hypotheses (see Sec. 7.2) in each iteration. Whenever $n > n_{\text{limit}}$, the correct hypothesis might be pruned accidentally due to the lack of information. So, the active loop closing must be triggered at right time so that loop closures can be detected and registered to provide sufficient information for *correct pruning* (keep tracking the correct hypothesis) while satisfying $n \le n_{\text{limit}}$ at any time.

Since new ambiguities might occur on the way to revisit M^* and further increases n before any true positive loop is detected, the ideal approach is to predict future ambiguities and loop closures, and trigger active loop closing accordingly beforehand. Even though there are existing algorithms that help predicting future loop closures [56], it is still an open question on how to predict future ambiguities. As a result, we simply choose a threshold $n_{\text{trigger}} < n_{\text{limit}}$ and trigger active loop closing when $n > n_{\text{trigger}}$ as a baseline approach. And occasionally when n goes beyond n_{limit}, we choose to prune some of the hypotheses before obtaining sufficient information in this work. In the simulation in Sec. 7.7.2, we test various combinations of n_{trigger} and n_{limit} to offer a good reference for choosing n_{trigger} based on n_{limit} for real-world applications. More discussions on this issue can be found in Sec. 7.7.4.

7.5.2 Target Submap Selection

When the active loop closing is triggered, a target submap M^* will be selected from the existing submaps for revisiting. Then, at least one loop that can result in correct pruning is expected to be detected and closed when the pose of M^* is successfully revisited, or even earlier on the way to M^*.

To select a proper M^*, we first check for each type #1 MMF f^{M}_r that models one ambiguous

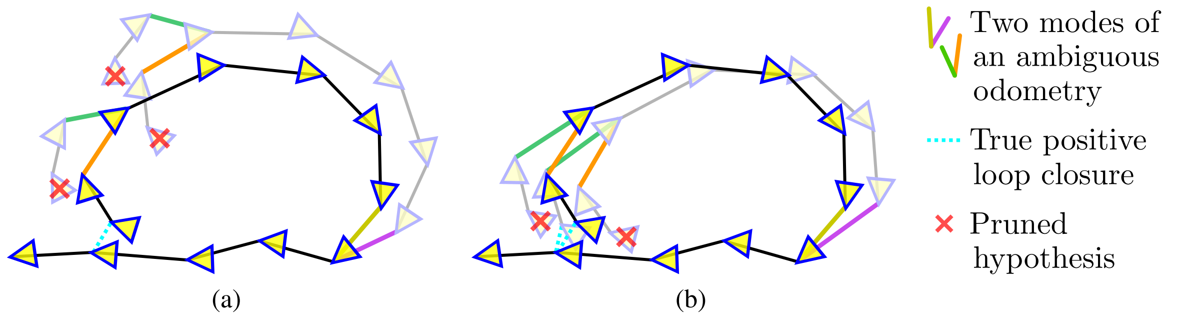

Two modes of an ambiguous odometry
True positive loop closure
Pruned hypothesis
(a)
(b)

Figure 7.6: Two examples of how a loop closure disambiguates the ambiguities modeled in the MMFs. (a) Usually, multiple undisambiguated MMFs $f_r^{\mathrm{M,u}}$ can be disambiguated by a single loop closure. (b) In special cases, some combinations of the modes of different $f_r^{\mathrm{M,u}}$ in different hypotheses can all seem to be correct even after the loop is closed.

odometry estimate (see Sec. 7.2) if more than one of its modes are selected in all existing hypotheses. If so, we can tell that the ambiguity modeled in f_r^{M} is not disambiguated yet, and flag f_r^{M} as *undisambiguated* (denoted as $f_r^{\mathrm{M,u}}$). The loop closure measurement from the current submap M_i to any submap M_j prior to an undisambiguated type #1 MMF $f_r^{\mathrm{M,u}}$ is very likely to provide the information that distinguishes the correct mode from the wrong ones in $f_r^{\mathrm{M,u}}$. Then, correct pruning can be conducted accordingly, which will preserve only the correct mode in all remaining hypotheses. If there are multiple $f_r^{\mathrm{M,u}}$ between M_i and M_j, it is possible to disambiguate all of them together with one single loop closure connecting M_i and M_j (see Fig. 7.6-a). However, this can fail in some special cases (see Fig. 7.6-b).

Based on the above discussions, we randomly select one target submap M_j^* from all the submaps prior to the earliest $f_r^{\mathrm{M,u}}$. And if no loop is detected before M_j^* is reached, we will select another $M_{j'}^*$ within the same interval, and repeat this process until $n \leq n_{\text{trigger}}$. This approach avoids the need of accessing or comparing the multi-hypothesis $\mathtt{T}_j$ of all possible M_j, which saves a lot of time especially when n or the map is large. Please see Sec. 7.7.2 for more discussions.

7.6 Path Planning with Multi-hypothesis Poses

In each hypothesis h, we first compute the *global target pose* $\mathtt{T}_{[h]}^* = \mathtt{T}_{i,[h]}\mathtt{T}_{\{i\}}^*$ in the case of exploration towards $P_{\{i\}}^*$ (see Sec. 7.4.3) or $\mathtt{T}_{[h]}^* = \mathtt{T}_{j,[h]}$ in the case of active loop closing towards M_j^* (see Sec. 7.5.2). Then, we check if there is a *valid straight path* (collision-free against any mapped obstacles) from the current pose $\mathtt{T}_{N,[h]}$ to the target pose $\mathtt{T}_{[h]}^*$ given all the existing submaps arranged based on their estimated poses in this hypothesis. If such *direct path* exists, the output motion command will be moving directly along this straight line with a certain distance. If not, we further check if there is a valid straight path from $\mathtt{T}_{N,[h]}$ to any of the previous poses between $\mathtt{T}_{N,[h]}$ and $\mathtt{T}_{i,[h]}$. If such *indirect path* exist, the output motion command will be again moving directly along this straight line with a certain distance, and a direct path is expected to be found after traveling through one or several indirect paths if h is the correct hypothesis.

If no paths are found or $\mathtt{T}_{[h]}^*$ is reached in h, we switch to the next hypothesis $h+1$ and repeat the same process until a new P^* or M^* is passed into the path planning module, which happens when the current $P^*_{\{i\}}$ is covered and removed, or the switch between exploration and active loop closing occurs. When the growing or pruning of hypotheses happens on the way to the same target, the new hypothesis h' will be selected as the first child of h (if it exists), or the first child of the next valid sibling of h (if h is pruned by the *backward pruning*. See Sec. 3.6). Finally, if no motion command can be found in any hypothesis, or all the commands are rejected by the obstacle detection module (see Sec. 7.3), random small motions will be conducted repeatedly until a valid path is found again to avoid the robot being stuck in corner cases. If more than a certain number of these random small motions is conducted in a row, it would be regarded as a failure case that cannot complete the active SLAM process.

It is worth noting that we can conduct planning in each hypothesis because we track the combination of modes from all MMFs in the Hypo-tree in MH-iSAM2 (see Sec. 3.4). The same algorithm does not work with nonparametric back-end solvers because they cannot recover the correspondences across hypotheses/peaks in different states (see Fig. 3.2).

7.7 Experimental Results

7.7.1 Implementations and Settings

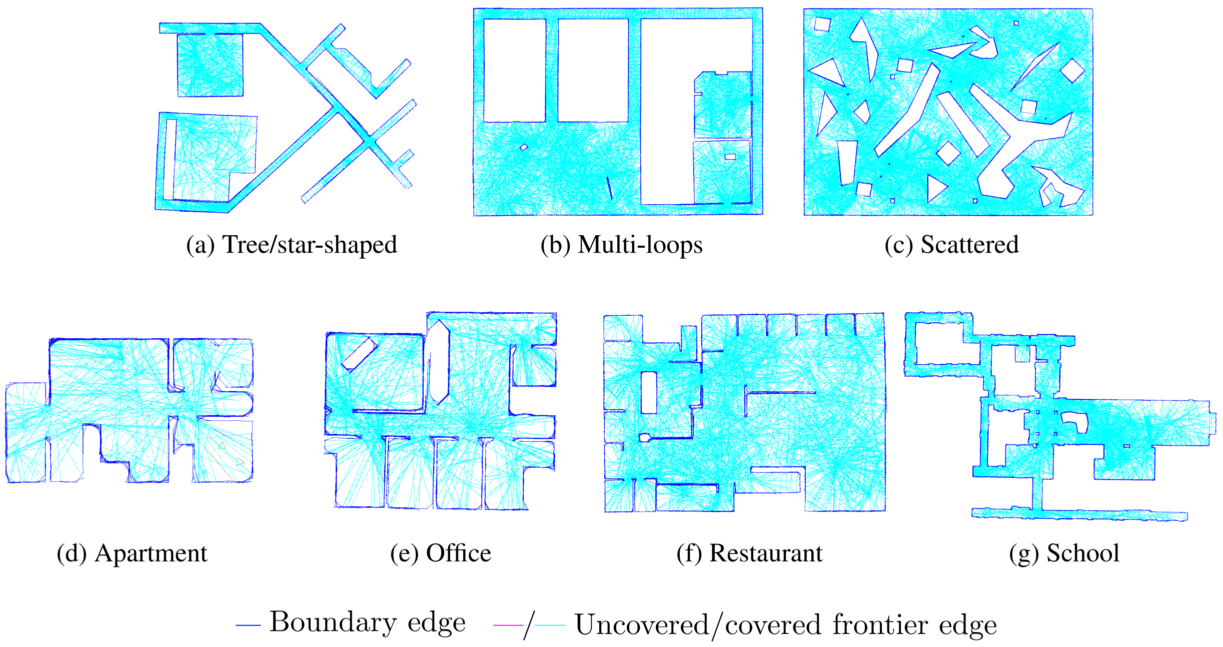
Both the simulation and the system for real-world application are implemented in C++ and executed on a laptop with an Intel Core i7-8850H processor. Exploration and active loop closing are computed in two parallel threads, and both of them are one keyframe behind the MH-SLAM process to wait for the latest submap being generated.

7.7.2 Simulation

In the simulation, Gaussian noise is added to all depth measurements, odometry, loop closures, and robot motions. We adopt two different ways to generate ambiguous odometry measurements. In most of our experiments, they are set to occur randomly with probabilities p_{a}, which simulates the types of ambiguities resulting from aggressive motion or dynamic scenes. And close to the end of this section, they are set to occur whenever the robot is close to one of the several randomly selected places, which simulates the other types of ambiguities resulting from lack of texture or repeated patterns in the environment. False positive loop closures are set to occur randomly with probabilities p_{f} in all of the simulation cases. The values of all the wrong modes in these simulated ambiguities are randomly generated within reasonable ranges.

We first show that ambiguity-aware active SLAM can achieve full coverage in various simulated indoor environments in Fig. 7.7. Then, we evaluate ambiguity-aware active SLAM with various test cases in the same multi-loop environment (see Fig. 7.7-b) and generate statistical results from 30 runs of each case. The results are shown in Fig. 7.8, 7.9, 7.10, and 7.12, and in each case we run the algorithm 30 times. Examples of the process of simulation are visualized in Appendix B.2.

Fig. 7.8 shows the fundamental properties of our ambiguity-aware active SLAM system with $n_{\mathrm{limit}} = \infty$ (no upper bound of the number of hypotheses) and various n_{trigger}. Even though



(a) Tree/star-shaped (b) Multi-loops (c) Scattered

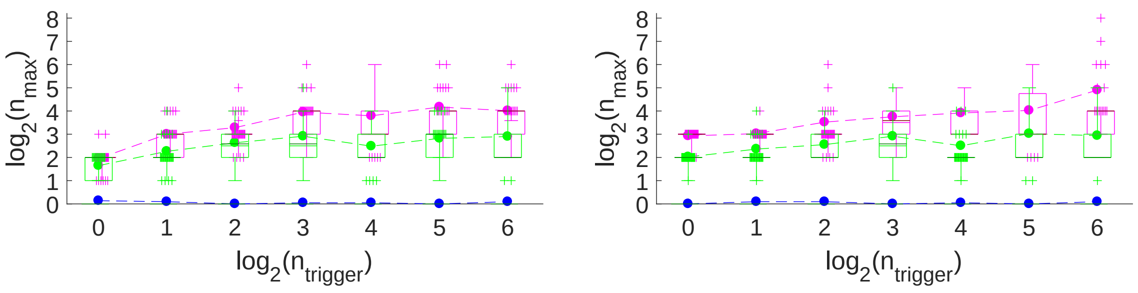
(d) Apartment (e) Office (f) Restaurant (g) School

Figure 7.7: Plotting all C_i in the global coordinates shows that our algorithm can achieve full coverage in various indoor environments in simulation with default settings: $n_{\text{trigger}} = 16$, $n_{\text{limit}} = \infty$, $p_{\text{a}} = 1\%$ and $p_{\text{f}} = 2\%$. The edges formed by $\mathbf{v}^{\text{B}}_{\{i\},p}$ and $\mathbf{v}^{\text{F,c}}_{\{i\},p}$ are shown in blue and cyan respectively.

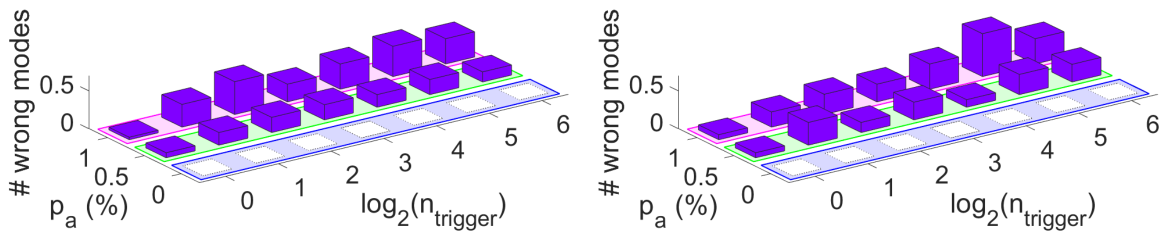
$n_{\text{limit}} = \infty$, the maximum number of hypothesis ever tracked (denoted as n_{max}) does not grow unbounded (see Fig. 7.8-a) since true positive loop closures are very likely to be detected shortly after the active loop closing is triggered. The correct hypothesis can still be pruned occasionally (see Fig. 7.8-b) since some wrong hypotheses that take wrong modes (especially in the type #1 MMFs of ambiguous odometry) might seem more likely than the correct one temporarily due to the accumulated drift. Therefore, some reconstructed maps are polluted (see Fig. 7.8-c). However, our ambiguity-aware active SLAM system can still explore the entire environment with these slightly distorted maps. Finally, Fig. 7.8-d shows that larger p_{a} results in longer path length, but neither n_{trigger} nor p_{f} have a strong effect on it.

Fig. 7.9 shows the first advanced evaluation on robustness based on various combinations of n_{trigger} and n_{limit}. Comparing to the results in Fig. 7.8-b and Fig. 7.8-c, bounding n_{limit} does increase the number of wrong modes taken and the failure rate as expected. In some cases when the map is extremely distorted, the ambiguity-aware active SLAM can even fail to complete the task (shown in orange in Fig. 7.9-b). And since larger n_{limit} and smaller n_{trigger} results in less failure, setting a smaller n_{trigger} for the system that has a specific n_{limit} is better for the overall robustness in general.

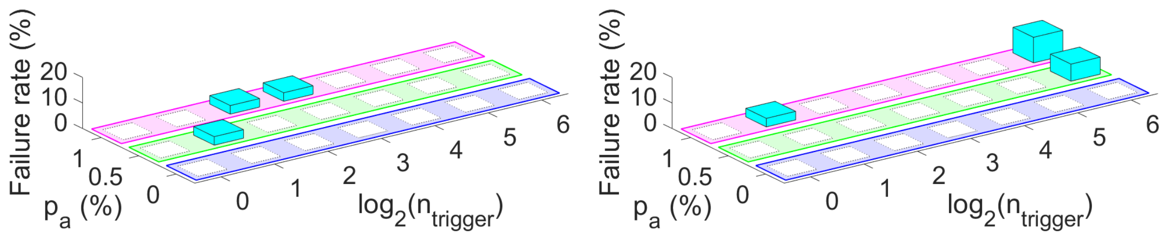
Fig. 7.10 further evaluates how the magnitude differences between each pair of correct mode and wrong mode can affect the performance of the proposed algorithm. We can see that even though MMFs with larger magnitude differences between modes are easier to be disambiguated (less wrong modes are taken in case “LL” than other cases in Fig. 7.10-a), they can result in



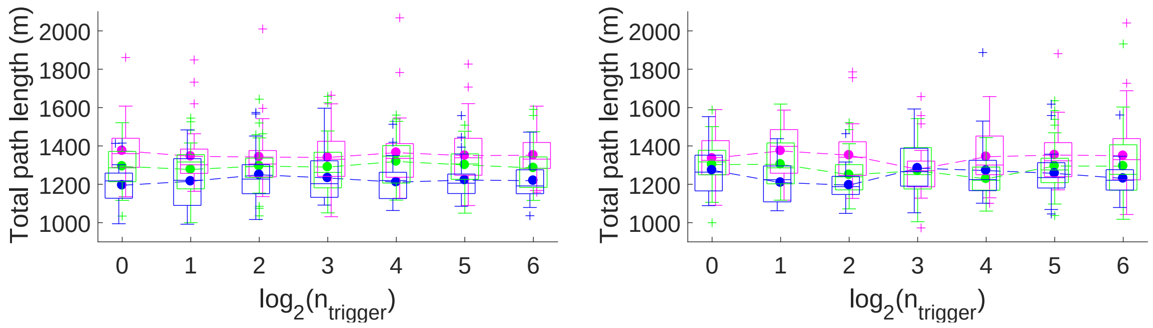
(a) Box plot and means (dotted curves) of n_{max}



(b) Bar graph of the average number of wrong modes taken



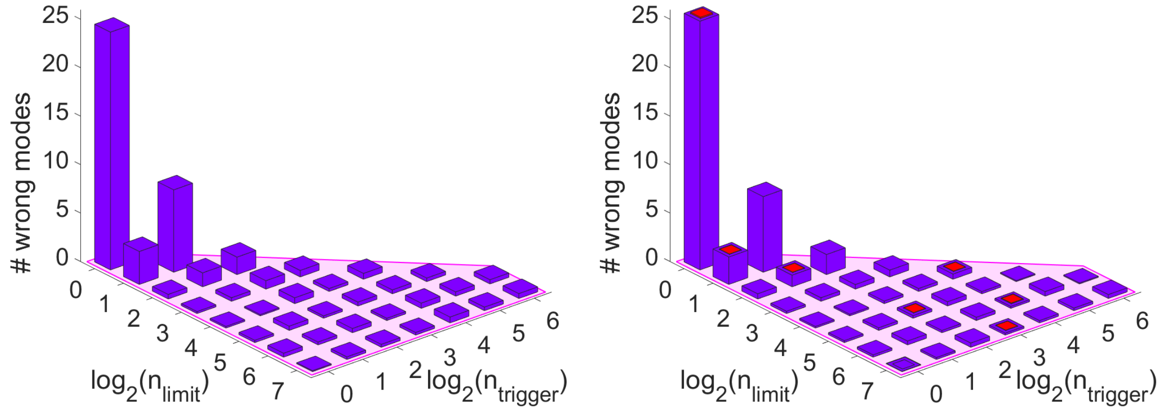
(c) Bar graph of failure rate



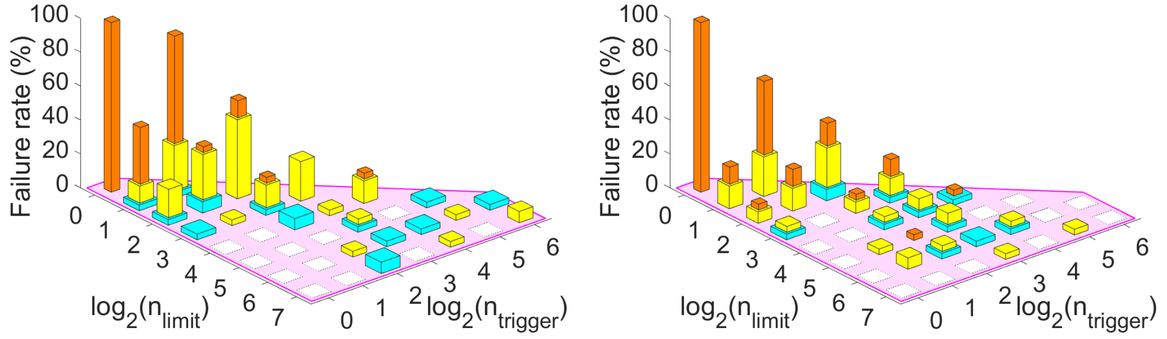
(d) Box plot and means (dotted curves) of the total path lengths

— $p_{\mathrm{a}}=0\%$ — $p_{\mathrm{a}}=0.5\%$ — $p_{\mathrm{a}}=1\%$ 🟪 Wrong odometry 🟥 False positive loop
🟦 Small distortion 🟨 Large distortion 🟧 Fail to achieve full coverage

Figure 7.8: Simulation results with $n_{\mathrm{limit}} = \infty$ and various n_{trigger}. Three different p_{a} (0%, 0.5% and 1%) are tested with $p_{\mathrm{f}}=0\%$ (left column) and $p_{\mathrm{f}}=2\%$ (right column). In (c), different levels of distortions are categorized based on thresholds on the *absolute trajectory error (ATE)* [118], and we can conclude that the active SLAM process can be completed successfully with occasional small distortions in the output maps as long as $n_{\mathrm{limit}} = \infty$.

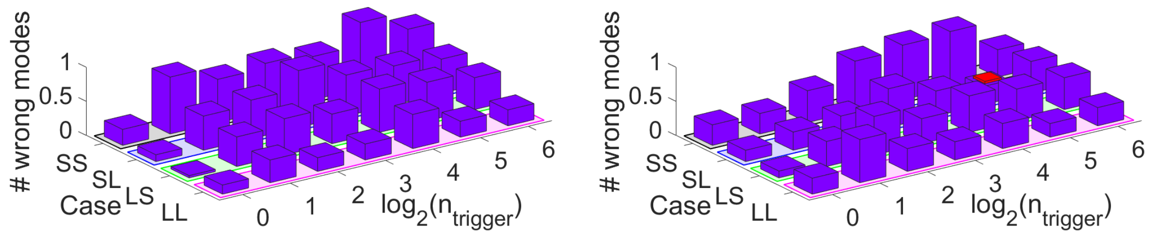


(a) Bar graph of the average number of wrong modes taken

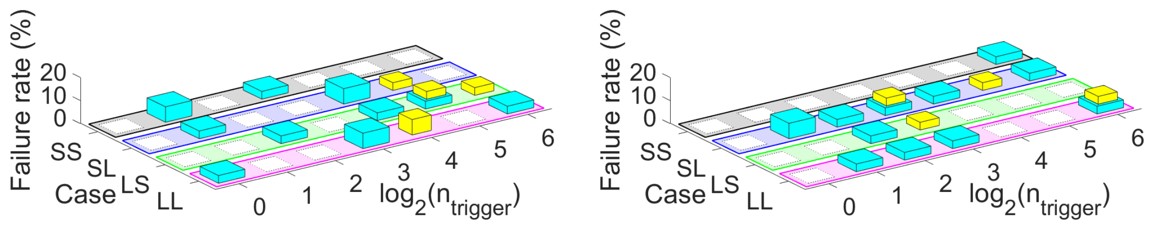


(b) Bar graph of failure rate

Figure 7.9: Simulation results with various n_{trigger} and n_{limit} given $p_{\text{a}} = 1\%$. The legends are the same as in Fig. 7.8. $p_{\text{f}} = 0\%$ and $p_{\text{f}} = 2\%$ are again shown in the left and right columns respectively. The leftmost bars in each graph represent the results of active SLAM with single-hypothesis estimation, where many wrong modes are taken and all the runs fail drastically.



(a) Bar graph of the average number of wrong modes taken



(b) Bar graph of failure rate

Rotation		Translation	
S	L	S	L
$5° \sim 20°$	$20° \sim 40°$	$0.1\text{m} \sim 0.6\text{m}$	$0.6\text{m} \sim 1.2\text{m}$

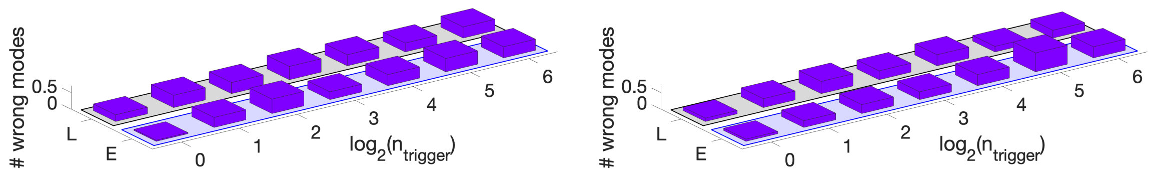
(c) Range of “S” and “L” in rotation and translation

Figure 7.10: Simulation results with different rotation and translation magnitudes of the wrong modes in ambiguous odometry, where “S” and “L” represent small or large magnitudes respectively (e.g.: “SL” means small magnitude in rotation and large magnitude in translation). Again, $p_{\text{a}} = 1\%$ is adopted in this experiment, and the left and right columns show the results with $p_{\text{f}} = 0\%$ and $p_{\text{f}} = 2\%$ respectively. The bars are colored in the same way as defined in Fig. 7.8. We can tell from these figures that smaller magnitudes in the wrong modes make them harder to be distinguished from the correct ones. However, they also result in less distortions in the output map.

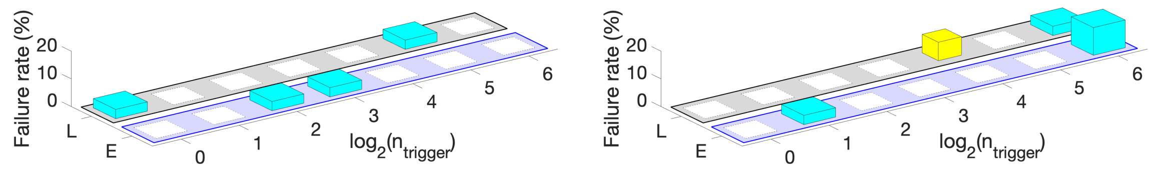
larger errors in the final map instead if the wrong modes are taken, especially in the case of large magnitude differences in translation (see case “SL” and “LL” in Fig. 7.10-b)

As for the selection of target submaps M^* for revisiting, even though there is no proof that our current approach (see Sec. 7.5.2) is optimal, some preliminary experimental results in Fig. 7.11 support our assumption: earlier target submaps for revisiting result in better robustness than later ones when the upper bound n_{limit} exists.

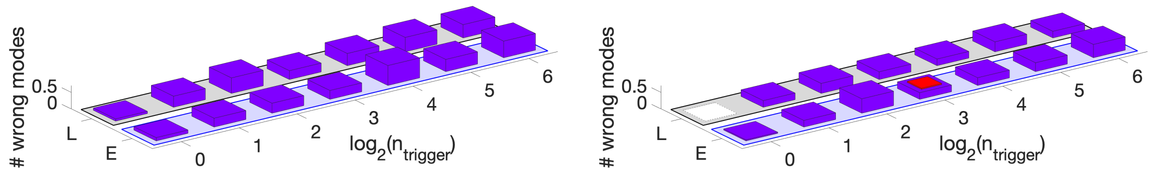
Finally, in Fig. 7.12 we change the way of generating ambiguities in odometry estimation to simulate ambiguities that result from the environments instead. In this simulation, whenever an ambiguity occurs, the robot is very likely to observe similar types of ambiguities when it is still moving around the same area, and has to travel a longer distance (leave the area where the ambiguity occurs) before being able to detect a valid loop closure to disambiguate the ambiguities. As a result, more wrong modes are taken as shown in Fig. 7.12-a, which results in higher failure rates in general as shown in Fig. 7.12-b. Moreover, we can see that for the cases with n_{trigger}



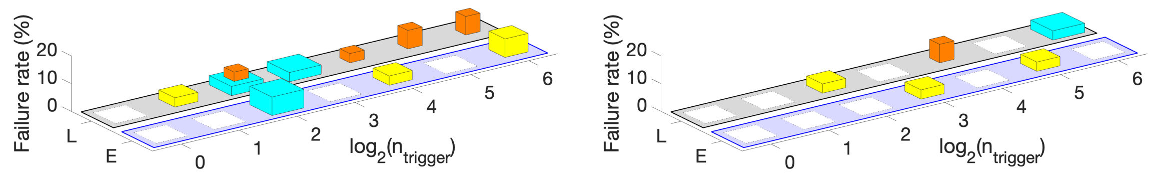
(a) Bar graph of the average number of wrong modes taken (no upper bound n_{limit})



(b) Bar graph of failure rate (no upper bound n_{limit})

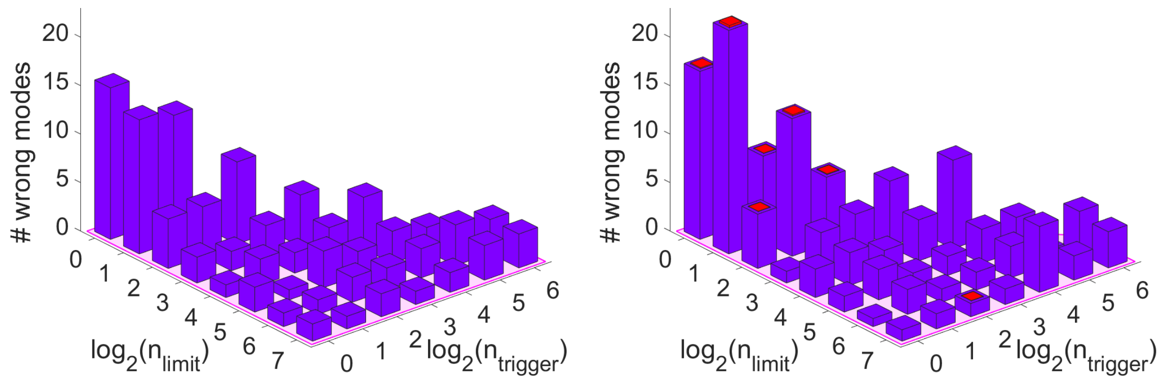


(c) Bar graph of the average number of wrong modes taken ($n_{\text{limit}} = 128$)

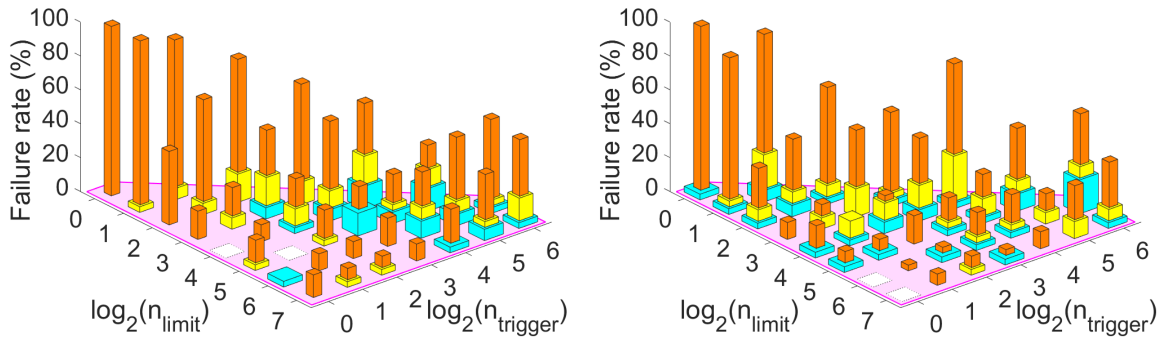


(d) Bar graph of failure rate ($n_{\text{limit}} = 128$)

Figure 7.11: Simulation results comparing two different target submap selection strategies under two settings: $n_{\text{limit}} = \infty$ in (a) and (b), and $n_{\text{limit}} = 128$ in (c) and (d). "L" and "E" represent choosing the target submap right before the latest and earliest undisambiguated MMF of ambiguous odometry estimate respectively. And again, $p_{\text{a}} = 1\%$ is adopted in both settings experiment, and the left and right columns show the results with $p_{\text{f}} = 0\%$ and $p_{\text{f}} = 2\%$ respectively. The bars are colored in the same way as defined in Fig. 7.8. We can tell from (a) and (b) that the choices of target submap for revisiting does not affect robustness much when there is no upper bound n_{limit}. However, an earlier submap can result in better robustness than a later one for revisiting when n_{limit} exists.



(a) Bar graph of the average number of wrong modes taken



(b) Bar graph of failure rate

Figure 7.12: Simulation results with ambiguous odometry measurements generated based on locations in the simulated environment. The bars are colored in the same way as defined in Fig. 7.8. We can find that the active SLAM process fails more than the results in Fig. 7.9.

and n_{limit} close to each other, their failure rates increase more than other cases comparing to the results in Fig. 7.9-b, which implies that to deal with environment-oriented ambiguities, a larger buffer for the number of growing hypotheses from n_{trigger} to n_{limit} is desired to maintain correct pruning.

7.7.3 Real-world Experiment

We apply the ambiguity-aware active SLAM framework in an *assistive mapping system* that guides a human user to explore and map an indoor environment with hand-held sensors though instructions (locations of P^* and M^*) on an augmented reality (AR) viewer and a top-down viewer (see Fig. 7.16-a). The passive ambiguity-aware planar-inertial SLAM (API-SLAM) system discussed in Ch. 6 is adopted here, which is slightly different from the simulation because additional planar constraints (see Sec. 5.3, 5.4, and 5.5) that satisfy all hypotheses are jointly optimized with the keyframe poses globally, which reduces rotational drift and potentially helps correct pruning. Moreover, since a human user can take care of path planning and online obstacle detection intuitively, these two functions are disabled for this application. Finally, since we cannot or might not want to map certain areas beyond the frontiers, e.g.: area behind a glass window, private spaces (offices or labs), or the bridge to another building, we add a function that allows the user to manually delete the current P^{*}, which will update the exploration tree accordingly and select a new P^{*}.

Since we do not have the ground truth of correct hypothesis in the real-world task, we cannot evaluate the robustness of the assistive mapping system as in Fig. 7.8-a or Fig. 7.12-a. Instead, we can use the same evaluation method in Sec. 6.5.2 here as long as the floor plans are available. The reconstructed multi-hypothesis 3D models, corresponding labeled floor plans, and the two-way registration results are shown in Fig. 7.13, Fig. 7.14, and Fig. 7.15 respectively.

The results in Fig. 7.15 show that our assistive mapping system successfully helps the user to explore a large indoor environment and reconstruct its dense 3D model even with multiple ambiguous odometry estimates and false positive loop closures. Comparing the results with that from the passive SLAM evaluation in Fig. 7.15-c and Fig. 7.15-d, we can tell that the magnitudes of errors of the assistive mapping system is close to that of the successful cases of API-SLAM, which implies that the active reconstruction results of the assistive mapping system are robust.

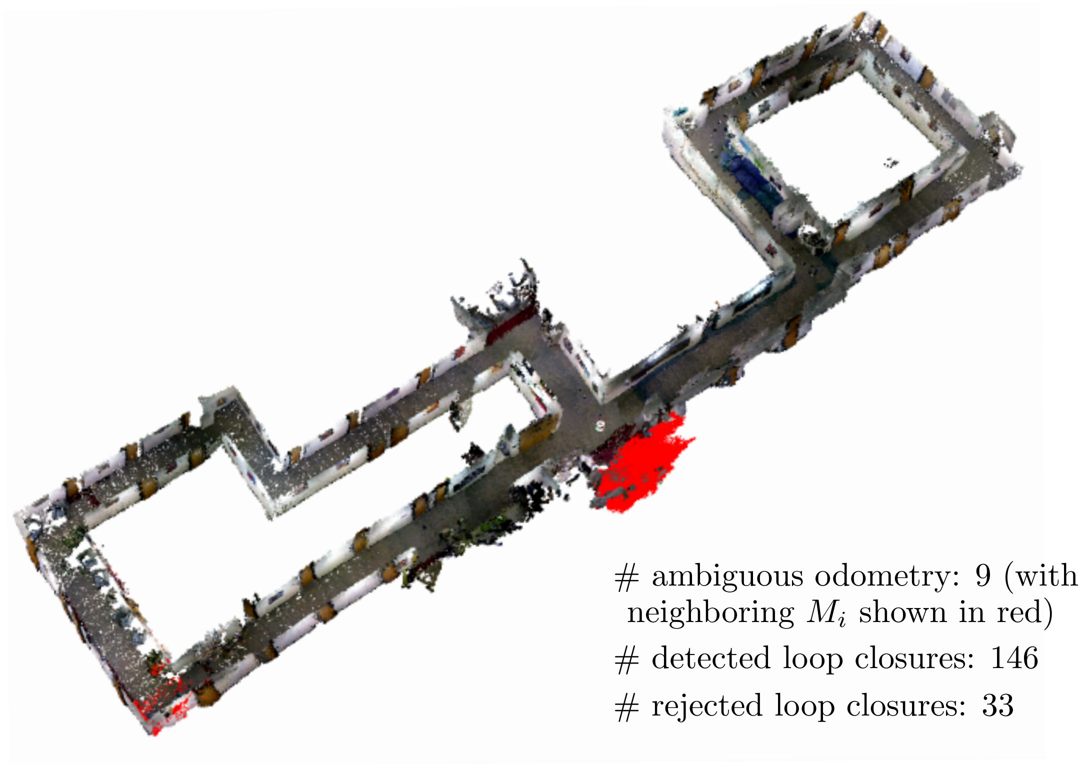
In addition to the robustness evaluation above, we can apply the same method in Fig. 5.12 to evaluate the accuracy of the output dense 3D models if the corresponding ground truth survey LiDAR models exist. Since we have the survey LiDAR model for the result in 7.13-a, we can calculate its point-to-plane mean absolute error (MAE) and root-mean-square error (RMSE) with respect to the ground truth survey LiDAR model as shown in Fig. 7.16.

Finally, note that the active SLAM process is conducted in real-time with the human user walking and rotating in normal pace and speed, and the ambiguities in the experiments are resulting from fast motion and the appearance-based loop closure detector (similar scenes at different locations). We also modify the assistive mapping system to work with only one single hypothesis, and apparently it fails easily due to taking outlier measurements into the optimization. The online active SLAM process of both single and multi-hypothesis approaches are shown in Appendix B.3.

7.7.4 Discussion

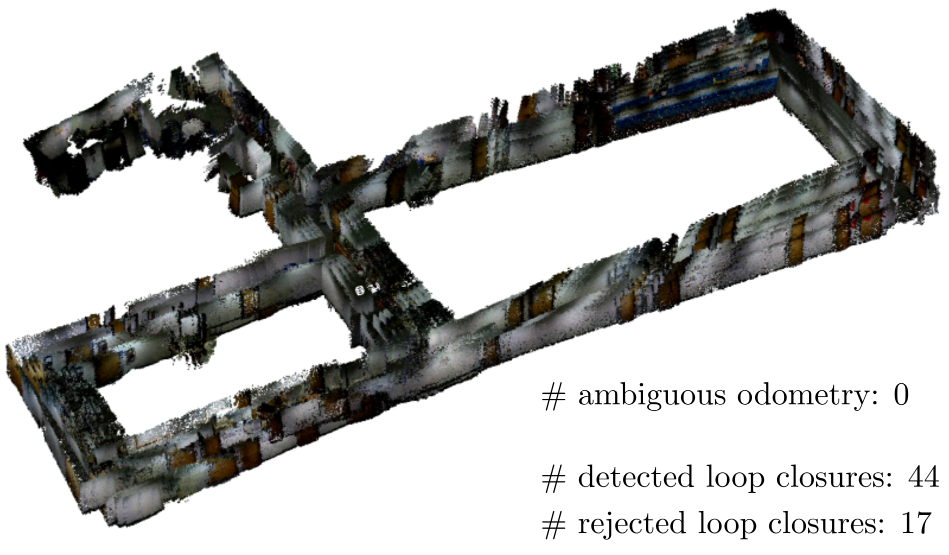
Based on the simulation results, we can tell that when the number of ambiguities is large but the computation is limited, we have to sacrifice speed for robustness. Else, the ambiguity-aware active SLAM would still fail to complete the active SLAM task. As a result, when designing a real world robotic system, we should try to solve the ambiguity problems during the state and map estimation process or even earlier, e.g. adopting proper sensor combination based on the motions, environments, and tasks to reduce ambiguous measurements, and only pass the unsolvable ambiguities to the exploration and active loop closing modules.

In the real-world application, we find that active loop closing is triggered more often with a small n_{trigger}, and following the revisiting instructions to walk back and forth frequently can result in bad user experience. So, we set $n_{\text{trigger}} = 2$ instead of 1 in this task to make the process slightly more comfortable. However, more studies on how to trigger active loop closing based on both n and the uncertainty of the current state estimates (a conventional approach) is still of interest. Besides, we also want to try if considering visual or structural saliency [86] in target submap selection helps detect more true positive loop closures. Finally, ambiguity-aware active SLAM is only robust to ambiguities but not to many other problems. Therefore, integrating more



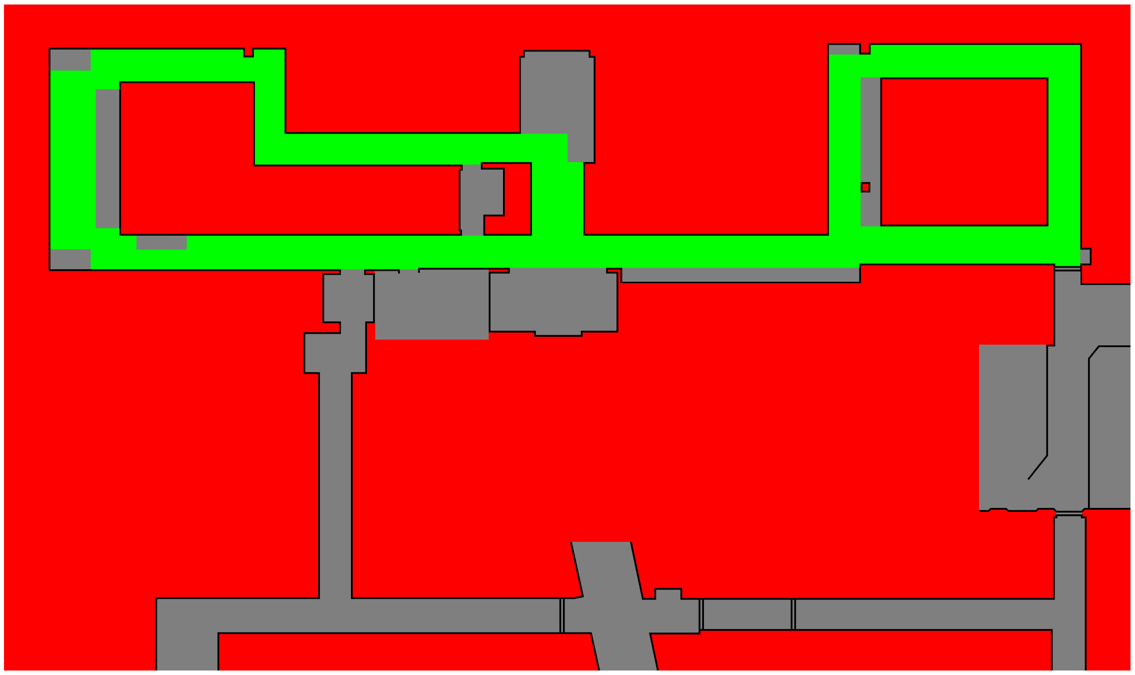
# ambiguous odometry: 9 (with neighboring M_i shown in red)
# detected loop closures: 146
# rejected loop closures: 33

(a) From NSH 4F



(b) From NSH floor A (basement)

Figure 7.13: The multi-hypothesis 3D models reconstructed by the assistive mapping system.


(a) From NSH 4F

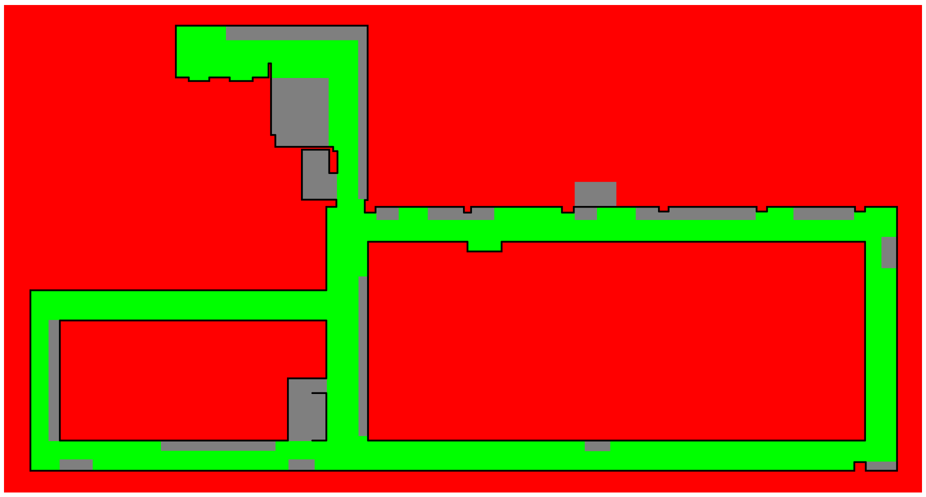
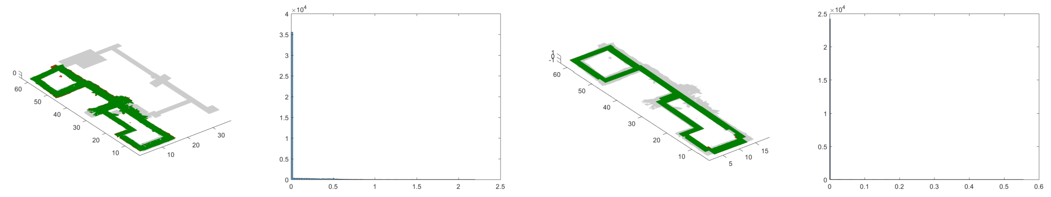
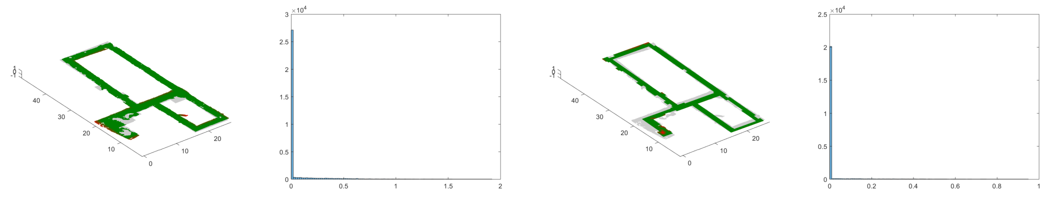
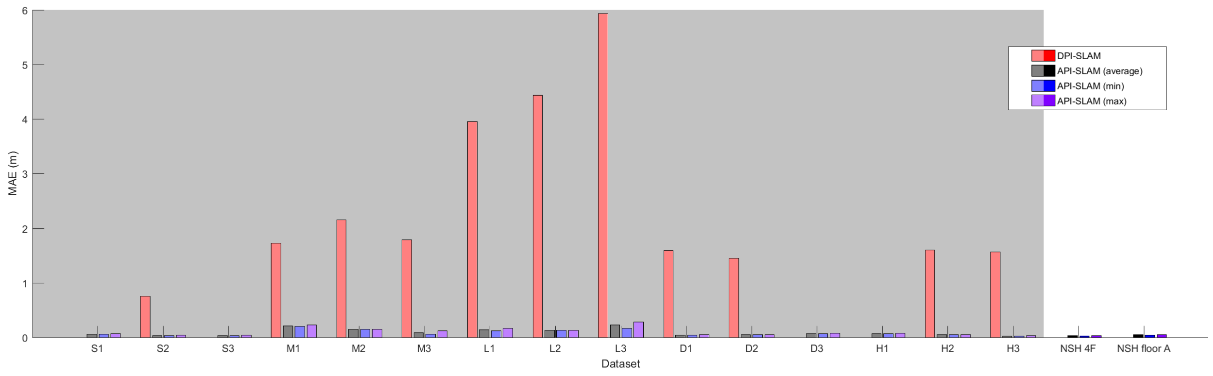

(b) From NSH floor A (basement)

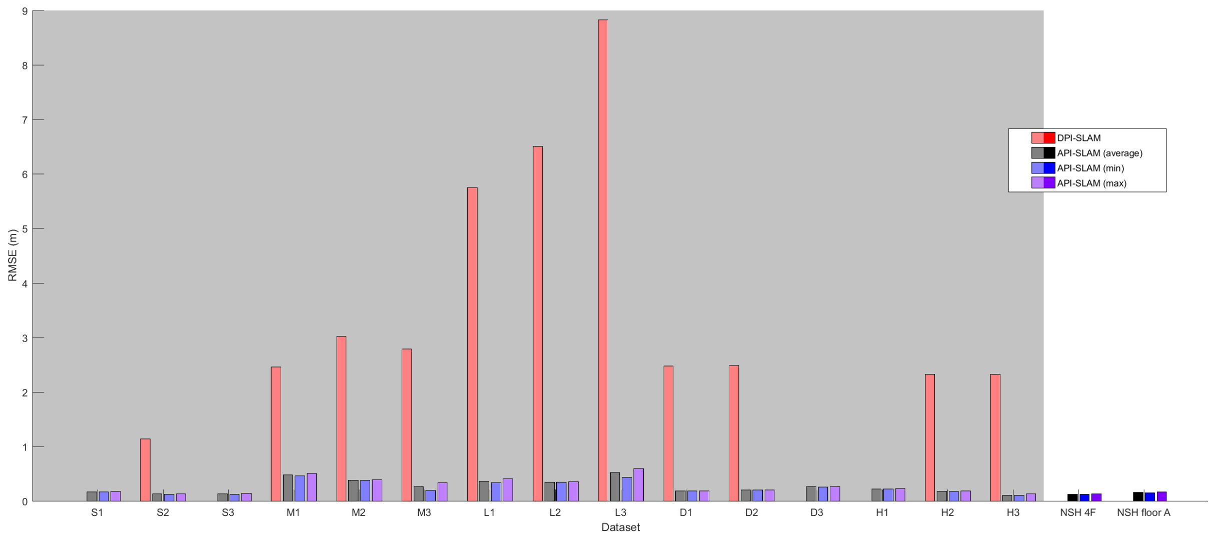
Figure 7.14: The labeled floor plans of the campus buildings that corresponds to the output 3D models in Fig. 7.13.


(a) From NSH 4F


(b) From NSH floor A (basement)



(c) MAE



(d) RMSE

Figure 7.15: The robustness evaluation of the assistive mapping system. In (a) and (b), the registration results are shown in the same format as in Fig. 6.4. In (c) and (d), the MAE and RMSE of the overall registration error are shown in two bar charts respectively. The evaluation of passive API-SLAM in Fig. 6.6 are shown alongside (shadowed areas) for comparison.

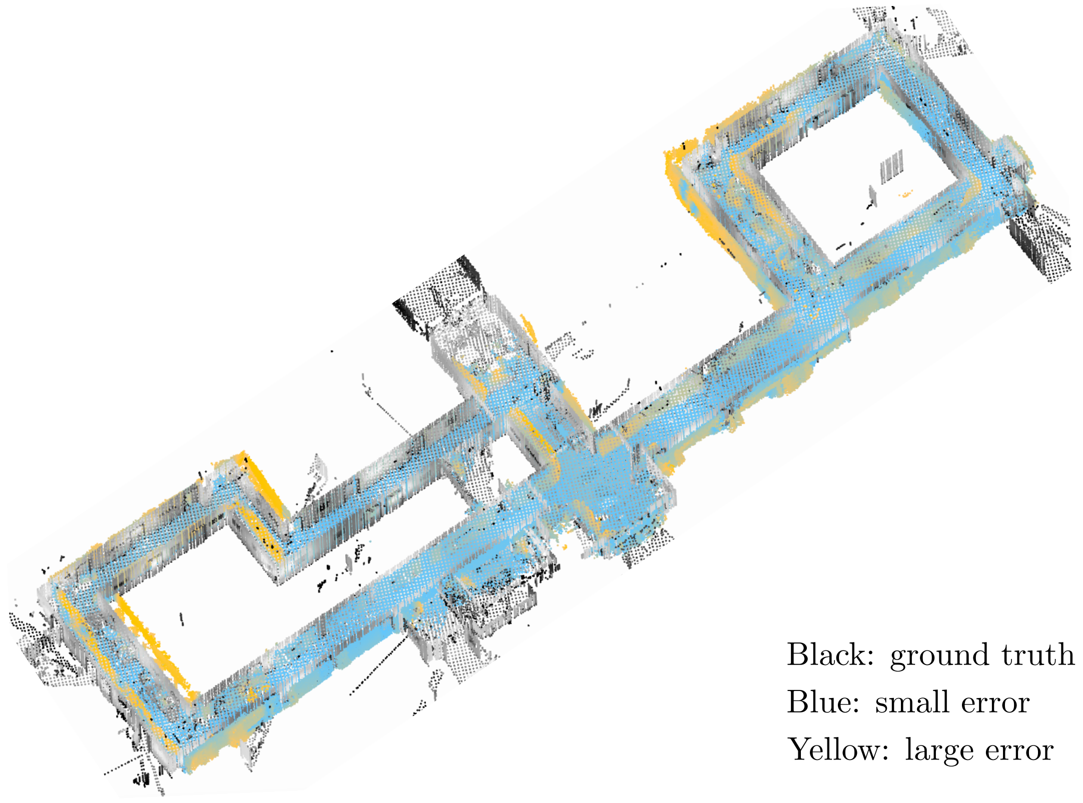


Figure 7.16: The accuracy evaluation of the most likely 3D model from the results shown in Fig. 7.13-a (with largest chi-square confidence among all remaining hypotheses) with respect to a ground truth survey LiDAR model. The MAE and RMSE are 0.187m and 0.269m respectively. Given that the entire model is about 58m$\times$15m$\times$3m, the average error ratio is less than 2%.

functions to handle various real world challenges (e.g.: relocalization [58] for tracking failures in all sensors) is still desired for better robustness of the entire system.

Lastly, extending current implementation to conduct active SLAM in full 3D is possible. However, it might require the usage of polyhedrons, which might not be as efficient as the current contour-based approach in 2D. We are looking forward to studying more on this direction for other applications such as underwater exploration [93][121].

7.8 Conclusion

In this chapter, we proposed the first ambiguity-aware active SLAM framework that makes use of multi-hypothesis state and map estimates from a MH-SLAM system to handle ambiguities and improve robustness. Its exploration module selects possible view points based on local submaps, which avoids the complexity of computing the view points in each of the hypotheses explicitly. And under certain reasonable conditions, the active loop closing algorithm can bound the growing number of hypotheses by potentially providing loop closure information in time, which allows multi-hypothesis solutions to remain tractable. In addition, a simple path planning method is adopted to compute motion commands that move towards the target locations. The experimental results show that explicitly considering multiple highly possible hypotheses greatly improves the robustness of active SLAM, and it is possible to achieve robustness and efficiency at the same time with a carefully designed system and fine-tuned parameters under certain scenarios.

In the future, we will work on the directions discussed in Sec. 7.7.4, and try different combinations of sensors and algorithms under the ambiguity-aware active SLAM framework. The ultimate goal is to integrate the ambiguity-aware active SLAM framework into a fully autonomous robotic platform that conducts daily tasks robustly.

Chapter 8

Conclusion

8.1 Summary

In this dissertation, the ambiguity problems in SLAM have been studied, and a series of solutions have been developed and tested with simulations or real world experiments.

The core idea of dealing with ambiguity problems is that the temporarily unsolvable ambiguities should be handled explicitly, and the SLAM results can contain multiple hypotheses accordingly whenever necessary. First of all, The MH-iSAM2 algorithm (see Ch. 3) is introduced to conduct nonlinear incremental optimization with multiple hypotheses efficiently, which serves as a back-end solver that allows handling and solving ambiguities explicitly. Then, several efficient SLAM components, including RGB-D-inertial odometry estimation (see Ch. 4), local depth fusion (see Ch. 5), and SLAM with 3D planes (see Ch. 5), are developed and tested under a conventional single-hypothesis SLAM framework before being adopted in the real world multi-hypothesis API-SLAM system (see Ch. 6) that conducts real-time multi-hypothesis dense 3D reconstruction on a CPU only. Finally, an ambiguity-aware active SLAM framework (see Ch. 7) is developed to make use of the multi-hypothesis state and map estimations from the API-SLAM system for exploration and active loop closing, which also helps reducing the growing number of hypotheses actively. All the works together demonstrates an integrated solution to the ambiguity problems of SLAM from all three aspects: front-end, back-end, and active perception.

8.2 Significance and Future Work

To the best of our knowledge, this is the first study that considers multi-hypothesis solutions explicitly throughout the entire SLAM system including tracking, optimization, mapping, exploration, active loop closing, and path planning. Even though the implemented system currently only works with the sensor combination of RGB-D camera and IMU, and the exploration and path planning is limited in 2D, the great potential of this research direction towards robust full autonomy is clear.

8.2.1 Multi-hypothesis Back-end Solver

The MH-iSAM2 algorithm proposed in Ch. 3 is a novel nonlinear incremental optimization solver for SLAM problems that takes multi-mode measurements that represent ambiguities as inputs, and outputs multi-hypothesis solutions accordingly. The Hypo-tree and the Bayes tree are integrated to avoid duplicated computations, which allows efficient updates of the most recent multi-hypothesis variables. While a pruning algorithm is developed to reduce the number of hypotheses when needed, future research on the selection of hypotheses given prior knowledge or active feedback is of great interest. How to make use of this back-end solver properly for various real world applications would also require a wide range of study and testing in the future.

8.2.2 Ambiguity-aware Passive SLAM

The API-SLAM proposed in Ch. 6 is the first multi-hypothesis dense 3D SLAM system that can handle ambiguities explicitly to the best of our knowledge. Detecting and modeling ambiguities from sensor measurements properly greatly improves the robustness of the SLAM system, and adopting submaps and planar constraints are the keys to an efficient dense 3D mapping process that allows the API-SLAM system to operate in real-time on a CPU only. In the near future, adding more sensors to further improve the system robustness under the multi-hypothesis framework would be a good extension. Analyzing, detecting, and modeling more types of ambiguities is also an interesting research direction.

8.2.3 Ambiguity-aware Active SLAM

A novel ambiguity-aware active SLAM framework is introduced in Ch. 7, which considers all the multiple state and map estimations from a MH-SLAM system in exploration, active loop closing, and path planning. The multi-hypothesis exploration algorithm based on ambiguity-free submaps is efficient enough to allow the entire system operating in real-time, and the active loop closing algorithm helps pruning the unlikely hypotheses and maintains tractable computation. Integrating the ambiguity-aware active SLAM framework into an actual robotic system for real world applications is the ultimate goal, and future studies on ambiguity-aware *interactive SLAM (the robot can move the objects in the environment actively)* or even in an *multi-agent system (MAS)* are possible extensions of this thesis research.

The final takeaway: This dissertation tackles the ambiguity problem both theoretically and systematically, and takes an important step towards “the best possible robustness”. However, robustness is still a crucial problem and not fully solved by the SLAM community. In the near future, we are looking forward to seeing more research on solving various types of robustness issues in SLAM.

Appendix A

Mathematical Foundations in SLAM

A.1 From MLE to Least Squares

As described in Sec. 3.1, a SLAM problem can be regarded as a maximum likelihood estimation (MLE) problem:

$$\hat{\Theta} = \arg\max_{\Theta} P\left(Z|\Theta\right) = \arg\max_{\Theta} \prod_k P\left(z_k|\Theta_k\right) \tag{A.1}$$

Assuming Gaussian noise for all measurements $P\left(z_k|\Theta_k\right)$, Eq. A.1 can be written as:

$$\arg\max_{\Theta} \prod_k \mathcal{N}\left(\boldsymbol{\mu}_k, \Sigma_k\right) = \arg\max_{\Theta} \prod_k \frac{1}{\sqrt{(2\pi)^p\left|\Sigma_k\right|}} \exp\left(-\frac{1}{2}\|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k}\right), \tag{A.2}$$

where $\boldsymbol{\mu}_k$ and Σ_k are the one or multi-dimensional mean and covariance of the Gaussian noise of the measurement z_k, and $h_k(\cdot)$ is the prediction function that generates predicted measurement value $\tilde{\mathbf{z}}_k$ from the set of relevant variables Θ_k. As we can see, the peak of each Gaussian is at $\tilde{\mathbf{z}}_k = \mathbf{z}_k$, which means that the prediction $h_k\left(\Theta_k\right) = \tilde{\mathbf{z}}_k$ agrees with the actual measurement value $\mathbf{z}_k$. Since our goal is to solve for the set of all variables $\hat{\Theta}$ that maximize the multiplication of the Gaussians, ignoring the constant scale terms and applying a log operator (which is monotonic) do not affect the solution:

$$\begin{aligned}\arg\max_{\Theta} \prod_k \exp\left(-\frac{1}{2}\|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k}\right) &= \arg\max_{\Theta} \log\left(\prod_k \exp\left(-\frac{1}{2}\|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k}\right)\right) \\ &= \arg\max_{\Theta} \sum_k \left(-\frac{1}{2}\|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k}\right) = \arg\min_{\Theta} \sum_k \|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k}\end{aligned} \tag{A.3}$$

In Eq. A.3, we can notice that the multiplication becomes summation after applying the log operation, and the $\arg\max$ becomes $\arg\min$ because of removing the negative sign in front of each *Mahalanobis distance* term $\|h_k(\Theta_k) - \mathbf{z}_k\|^2_{\Sigma_k} = \left(h_k(\Theta_k) - \mathbf{z}_k\right)^\top \Sigma_k^{-1} \left(h_k(\Theta_k) - \mathbf{z}_k\right)$. The resulting form is a nonlinear least squares problem since the prediction functions $h_k(\cdot)$ are mostly nonlinear in SLAM problems, which can be solved using the *iterative nonlinear optimization* technique by solving the linearized function at each iteration to update the estimations until convergence. The linearized function can be written as:

$$\begin{aligned}\arg\min_{\Theta}\sum_{k}\|(A_k\boldsymbol{\theta}_k-\mathbf{b}_k)-\mathbf{z}_k\|^2_{\Sigma_k}&=\arg\min_{\Theta}\sum_{k}\left\|\Sigma_k^{-\frac{1}{2}}A_k\boldsymbol{\theta}_k-\Sigma_k^{-\frac{1}{2}}(\mathbf{b}_k-\mathbf{z}_k)\right\|^2\\&=\arg\min_{\Theta}\sum_{k}\|A'_k\boldsymbol{\theta}_k-\mathbf{b}'_k\|^2,\end{aligned}\tag{A.4}$$

where the covariance can be combined with the Jacobian matrix A and the constant term $\mathbf{b}_k-\mathbf{z}_k$, which can result in a standard form of *linear least squares* $\arg\min_{\Theta}\|A\boldsymbol{\theta}-\mathbf{b}\|^2$ if we concatenate all the terms into one linear equation based on a given variable ordering.

A straightforward way to solve this least squares problem is based on the fact that the first derivative of the function at its optimal must equal to zero:

$$\arg\min_{\Theta}\|A\boldsymbol{\theta}-\mathbf{b}\|^2\Rightarrow A^\top A\boldsymbol{\theta}-A^\top\mathbf{b}=0\Rightarrow\boldsymbol{\theta}=\left(A^\top A\right)^{-1}A^\top\mathbf{b},\tag{A.5}$$

where $\left(A^\top A\right)^{-1}A^\top$ is known as the *pseudoinverse* of A. Even though this method is theoretically correct, it can cause several problems in practice. First of all, the dimensions of A depend on the number of measurement factors and variables, which can be very large as the SLAM problem grows over time. In this case, the required inverse calculation $\left(A^\top A\right)^{-1}$ can be very time consuming since even the computational complexity of the most efficient matrix inversion algorithm is worse than quadratic ($O(n^{2.373})$ for *optimal Coppersmith–Winograd algorithm* [65]). Also, computing the inverse of a matrix on a computer can run into numerical problems if insufficient digits are applied (e.g. type *double* works better than type *float* in C/C++ programs). As a result, a more elegant way to solve this least squares problem is to apply *matrix factorization* or *decomposition*:

$$\begin{aligned}\arg\min_{\Theta}\|A\boldsymbol{\theta}-\mathbf{b}\|^2=\arg\min_{\Theta}\|QA\boldsymbol{\theta}-Q\mathbf{b}\|^2&=\arg\min_{\Theta}\left\|\begin{bmatrix}R\\0\end{bmatrix}\boldsymbol{\theta}-\begin{bmatrix}\mathbf{d}\\\mathbf{e}\end{bmatrix}\right\|^2\\&=\arg\min_{\Theta}\|R\boldsymbol{\theta}-\mathbf{d}\|^2+\|\mathbf{e}\|^2,\end{aligned}\tag{A.6}$$

where Q is a orthonormal matrix ($Q^\top Q=I$) that transforms A into a upper triangular matrix R and zero rows in the bottom. Notice that multiplying Q to A and $\mathbf{b}$ does not affect the solution since $\|QA\boldsymbol{\theta}-Q\mathbf{b}\|^2=(QA\boldsymbol{\theta}-Q\mathbf{b})^\top(QA\boldsymbol{\theta}-Q\mathbf{b})=(A\boldsymbol{\theta}-\mathbf{b})^\top Q^\top Q\,(A\boldsymbol{\theta}-\mathbf{b})=\|A\boldsymbol{\theta}-\mathbf{b}\|^2$. We can further separate the resulting linear system into two parts as shown in Eq. A.6, where an exact solution to $\boldsymbol{\theta}$ can be found from $\|R\boldsymbol{\theta}-\mathbf{d}\|^2=0\Rightarrow R\boldsymbol{\theta}=\mathbf{d}$ through backsubstitution, and $\|\mathbf{e}\|^2$ is the corresponding error of this solution.

In practice, Q does not have to be constructed specifically. Instead, efficient methods such as *Givens rotation* [31] or *Householder reflection* [37] both modify on the A and $\mathbf{b}$ directly to generate R, $\mathbf{d}$, and $\mathbf{e}$. Advanced algorithms such as iSAM [49] or iSAM2 [50] even enable incremental updates of the R, $\mathbf{d}$, and $\mathbf{e}$ for real-time SLAM tasks.

A.2 From Factor Graph to Bayes Tree

A factor graph (see Sec. 3.1) describes a SLAM problem from the probabilistic point of view: solving posterior $P(\Theta|Z)$ is equivalent to solving the likelihood $P(Z|\Theta)$ assuming no prior $p(\Theta)$. The likelihood $P(Z|\Theta) = \prod_k P(z_k|\Theta_k)$ is the joint probability of all measurement factors $P(z_k|\Theta_k)$, which is high-dimensional and hard to compute directly. To explore the sparsity of SLAM problems and better understand its inference process, the Bayes tree is developed in [50] to convert the joint probability of the entire factor graph into conditional densities and store them in individual cliques in a tree structure.

The construction process of a Bayes tree consists of two major steps. In the first step, a factor graph (see Fig. A.1-a) can be converted into a *Bayes net* (see Fig. A.1-b) through linearization and *variable elimination*, which is equivalent to applying matrix factorization onto the linearized jacobian matrix A in each iteration of solving the MLE problem (see Eq. A.4 and Eq. A.6). In the example in Fig. A.1, the elimination ordering is given as l_1, l_2, x_1, x_2, x_3. As a result, each variable only depends on the variables that are eliminated later, which is represented using the arrows in the Bayes net in Fig. A.1-b (e.g.: l_1 depends on x_1 and x_2, so there are two arrows pointing from x_1 and x_2 to l_1 respectively). Same dependency structure can be observed in the factorized upper-triangular matrix in Fig. A.1-b, where the corresponding row of one variable only contains *non-zero elements* on the corresponding columns of (1) the variables that it depends on and (2) itself (e.g.: again l_1 depends on x_1 and x_2, so there are three non-zero elements in the first row: on the column of x_1, x_2, and l_1 itself).

From the probabilistic point of view, the same variable elimination process in Fig. A.1 can be explained as below. The desired posterior (see Sec. 3.1) can be written as:

$$P(\Theta|Z) \propto P(Z|\Theta) = \prod_k P(z_k|\Theta_k), \tag{A.7}$$

where

$$\prod_k P(z_k|\Theta_k) = P(z_1|x_1)P(z_2|x_1,l_1)P(z_3|x_1,x_2)P(z_4|x_2,l_1)P(z_5|x_2,x_3)P(z_6|x_3,l_2) \tag{A.8}$$

is the multiplication of the probability of all measurement factors. Following the same variable ordering l_1, l_2, x_1, x_2, x_3 in Fig. A.1 to group the factors, we can rewrite Eq. A.8 as:

$$\begin{aligned}\prod_k P(z_k|\Theta_k) = {} & [P(z_2|x_1,l_1)\,P(z_4|x_2,l_1)]_{\{l_1\}}\,[P(z_6|x_3,l_2)]_{\{l_2\}} \\ & [P(z_1|x_1)\,P(z_3|x_1,x_2)]_{\{x_1\}}\,[P(z_5|x_2,x_3)]_{\{x_2\}}\,[\cdot]_{\{x_3\}}\,,\end{aligned} \tag{A.9}$$

where$_{\{\cdot\}}$ denotes first variable that affects the group of factors in $[\cdot]$ based on the variable ordering. Notice that there is no factor in that last group of variable x_3.

Now, we can eliminate each variable following the given ordering. Based on the fact that each measurement only depends on the variables that directly connect to it in the factor graph,

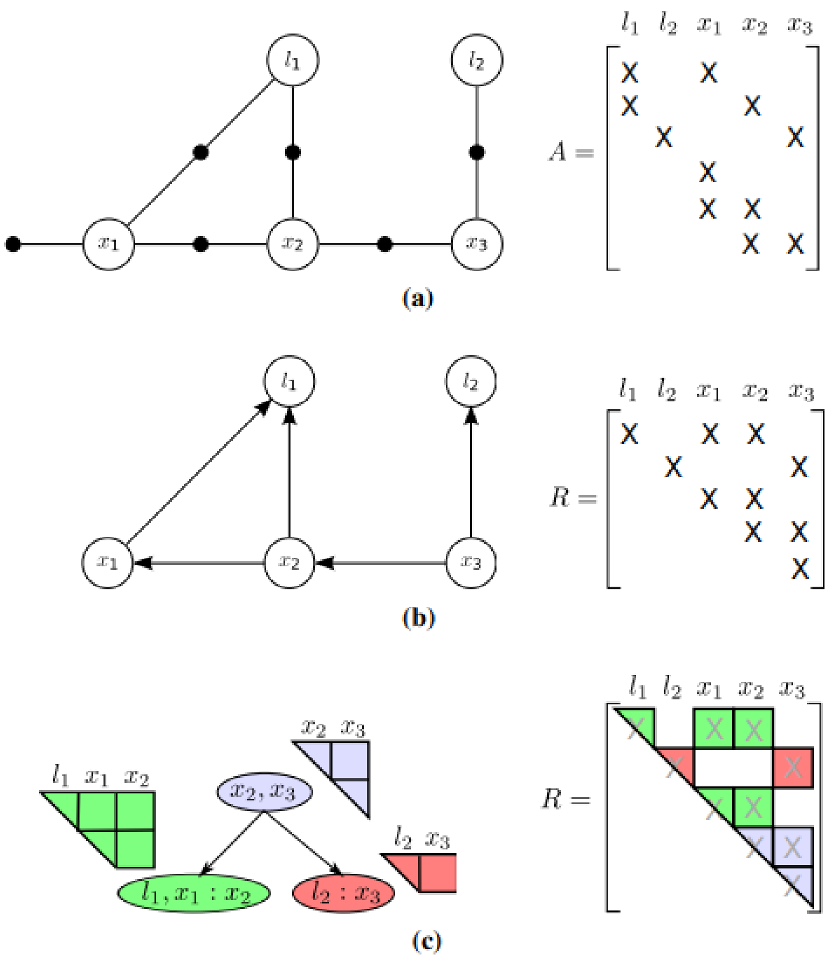


Figure A.1: An example of constructing a Bayes tree from a factor graph (figure from [50]). (a) A factor graph that consists of nonlinear factors can be linearized and represented as a Jacobian matrix in each iteration of nonlinear optimization. Notice that the same sparse structure can be observed in both the graph and the matrix. (b) Applying variable elimination on a factor graph with a given variable ordering (l_1, l_2, x_1, x_2, x_3) results in a Bayes net that preserves sparsity. The equivalent computation on the Jacobian matrix is matrix factorization/decomposition, which generates a upper triangular matrix that also preserves sparsity. (c) A Bayes tree can be constructed with the cliques found in the Bayes net, where each clique corresponds to a dense block in the factorized matrix.

we can combine all the probabilities in the first group and apply Bayes' rule to create a *local joint density*:

$$[P(z_2|x_1,l_1)\,P(z_4|x_2,l_1)]_{\{l_1\}} = [P(z_2,z_4|x_1,x_2,l_1)]_{\{l_1\}} \propto [P(x_1,x_2,l_1,z_2,z_4)]_{\{l_1\}} \tag{A.10}$$

Then, we define frontals as the variable labels in $_{\{\cdot\}}$ and separators as the rest of the variables in the joint density, which allows us to create a *conditional density* and a *marginal density* using the *chain rule*:

$$\begin{aligned}[P(x_1,x_2,l_1,z_2,z_4)]_{\{l_1\}} &= \left[P\left(l_1^F,x_1^S,x_2^S,z_2,z_4\right)\right]_{\{l_1\}} \\ &= \left[P\left(l_1^F|x_1^S,x_2^S,z_2,z_4\right)P\left(x_1^S,x_2^S,z_2,z_4\right)\right]_{\{l_1\}},\end{aligned} \tag{A.11}$$

where $\cdot^F$ and $\cdot^S$ denote the frontals and separators respectively. Notice that since l_1 is entirely eliminated from the marginal density $P\left(x_1^S,x_2^S,z_2,z_4\right)$, we can pass this marginal density to the next group that is labeled one of the separators (it is x_1 under this variable ordering). Following the same logic, we can apply this process to each group following the variable ordering to complete variable elimination:

$$\begin{aligned}P(\Theta|Z) \propto P(Z|\Theta) = [P(z_2|x_1,l_1)\,P(z_4|x_2,l_1)]_{\{l_1\}}\,[P(z_6|x_3,l_2)]_{\{l_2\}} \\ [P(z_1|x_1)\,P(z_3|x_1,x_2)]_{\{x_1\}}\,[P(z_5|x_2,x_3)]_{\{x_2\}}\,[\cdot]_{\{x_3\}}\end{aligned} \tag{A.12}$$

$$\propto \left[P\left(l_1^F|x_1^S,x_2^S,z_2,z_4\right)P\left(x_1^S,x_2^S,z_2,z_4\right)\right]_{\{l_1\}}\cdots[P(z_1|x_1)\,P(z_3|x_1,x_2)]_{\{x_1\}}\cdots \tag{A.13}$$

$$= \left[P\left(l_1^F|x_1^S,x_2^S,z_2,z_4\right)\right]_{\{l_1\}}\cdots\left[P\left(x_1^S,x_2^S,z_2,z_4\right)P(z_1|x_1)\,P(z_3|x_1,x_2)\right]_{\{x_1\}}\cdots \tag{A.14}$$

$$\propto \cdots\left[P\left(l_2^F|x_3^S,z_6\right)P\left(x_3^S,z_6\right)\right]_{\{l_2\}}\cdots[P(z_5|x_2,x_3)]_{\{x_2\}}\,[\cdot]_{\{x_3\}} \tag{A.15}$$

$$= \cdots\left[P\left(l_2^F|x_3^S,z_6\right)\right]_{\{l_2\}}\cdots[P(z_5|x_2,x_3)]_{\{x_2\}}\left[P\left(x_3^S,z_6\right)\right]_{\{x_3\}} \tag{A.16}$$

$$\propto \cdots\left[P\left(x_1^F|x_2^S,z_1,z_2,z_3,z_4\right)P\left(x_2^S,z_1,z_2,z_3,z_4\right)\right]_{\{x_1\}}[P(z_5|x_2,x_3)]_{\{x_2\}}\cdots \tag{A.17}$$

$$= \cdots\left[P\left(x_1^F|x_2^S,z_1,z_2,z_3,z_4\right)\right]_{\{x_1\}}\left[P\left(x_2^S,z_1,z_2,z_3,z_4\right)P(z_5|x_2,x_3)\right]_{\{x_2\}}\cdots \tag{A.18}$$

$$\propto \cdots\left[P\left(x_2^F|x_3^S,z_1,z_2,z_3,z_4,z_5\right)P\left(x_3^S,z_1,z_2,z_3,z_4,z_5\right)\right]_{\{x_2\}}\left[P\left(x_3^S,z_6\right)\right]_{\{x_3\}} \tag{A.19}$$

$$= \cdots \left[P\big(x_2^F|x_3^S, z_1, z_2, z_3, z_4, z_5\big)\right]_{\{x_2\}} \left[P\big(x_3^S, z_1, z_2, z_3, z_4, z_5\big)\, P\big(x_3^S, z_6\big)\right]_{\{x_3\}} \tag{A.20}$$

$$\begin{aligned} &\propto \left[P\big(l_1^F|x_1^S, x_2^S, z_2, z_4\big)\right]_{\{l_1\}} \left[P\big(l_2^F|x_3^S, z_6\big)\right]_{\{l_2\}} \left[P\big(x_1^F|x_2^S, z_1, z_2, z_3, z_4\big)\right]_{\{x_1\}} \\ &\quad \left[P\big(x_2^F|x_3^S, z_1, z_2, z_3, z_4, z_5\big)\right]_{\{x_2\}} \left[P\big(x_3^F, z_1, z_2, z_3, z_4, z_5, z_6\big)\right]_{\{x_3\}} \end{aligned} \tag{A.21}$$

From the equations above, we can learn that constructing the local joint density of a group requires merging the marginal information passed from previous groups. As a result, the conditional density in each group must depend on the densities in latter groups, which is identical to the structure of the Bayes net in Fig. A.1-b.

The second step of constructing a Bayes tree is to find cliques in the Bayes net and explore the tree structure among them. A clique is defined as a largest densely connected group of variables as shown in Fig. A.1-c. After finding all the cliques in the Bayes net, we can directly find the conditional relationships among them based on the arrows that cross different cliques, and construct a Bayes tree based on these conditions. In the corresponding factorized matrix, each clique also represents a dense block of elements, and the sparse structure of the matrix is identical to the Bayes net that can be represented as a tree.

From the probabilistic point of view, constructing the cliques and the Bayes tree can be written as grouping and combining the probabilities in Eq. A.21 following the backward ordering:

$$P(\Theta|Z) \propto P\big(l_1^F|x_1^S, x_2^S, \cdots\big)\, P\big(l_2^F|x_3^S, \cdots\big)\, P\big(x_1^F|x_2^S, \cdots\big)\, P\big(x_2^F|x_3^S, \cdots\big)\, P\big(x_3^F, \cdots\big)$$

$$= \left[P\big(l_2^F|x_3^S, \cdots\big)\right]_{C_2} \left[P\big(l_1^F|x_1^S, x_2^S, \cdots\big) P\big(x_1^F|x_2^S, \cdots\big)\right]_{C_1} \left[P\big(x_2^F|x_3^S, \cdots\big) P\big(x_3^F, \cdots\big)\right]_{C_0}$$

$$= \left[P\big(l_2^F|x_3^S, \cdots\big)\right]_{C_2} \left[P\big(l_1^F, x_1^F|x_2^S, \cdots\big)\right]_{C_1} \left[P\big(x_2^F, x_3^F, \cdots\big)\right]_{C_0} \tag{A.22}$$

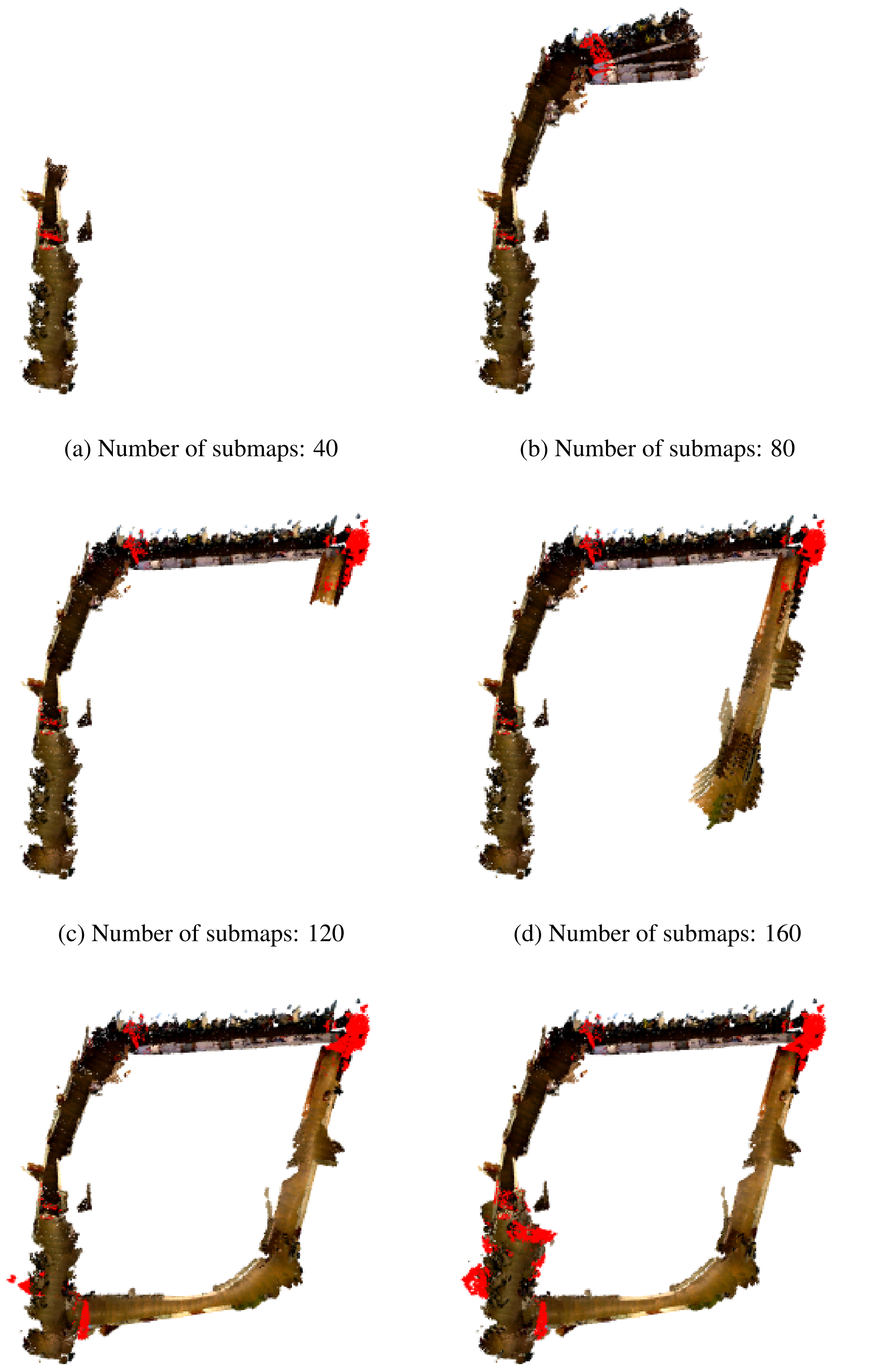
Notice that in practice we can first work on the structure of the factor graph to generate the structure of Bayes net and the cliques without calculating the actual probabilities, then directly compute the conditional and marginal densities of each clique following the process from Eq. A.8 to Eq. A.22. In iSAM2 [50], since Gaussian noise assumption is applied to all measurements, the actual computation of the densities is implemented as matrix factorization on the local Jacobian of all the relevant factors in each clique (see Appendix A.1). However, since constructing a Bayes tree from a factor graph does not assume any distribution for any of the probabilities in the above derivations, this algorithm can be generalized to the construction of a multi-hypothesis Bayes tree (MHBT) from a multi-hypothesis factor graph (MHFG) in the proposed robust back-end optimizer MH-iSAM2 (see Sec. 3.5).

Appendix B

Supplemental Results

B.1 3D Reconstructions of API-SLAM

Fig. B.1, Fig. B.3, Fig. B.2, and Fig. B.4 show three examples of the mapping process of API-SLAM, where all the remaining hypotheses of the map are shown together in each figure. We can observe that when ambiguities of odometry estimation occur in Fig. B.1-b, Fig. B.2-d, Fig. B.3-d (corresponding color and depth inputs are shown in Fig. 6.2-a) and Fig. B.4-e (corresponding color and depth inputs are shown in Fig. 6.2-b), multiple hypotheses of the map with obviously different geometry are kept tracked by API-SLAM at the same time. Then, some of these hypotheses are pruned either due to the limitation of maximum number of hypotheses (Fig. B.1-c and Fig. B.2-f) or additional information provided such as loop closures (Fig. B.3-g, Fig. B.2-h and Fig. B.4-h). We can visually observe that at least one of the remaining hypotheses in each of these examples preserves the correct topology/geometry of the environment in the reconstructed 3D map.



(a) Number of submaps: 40

(b) Number of submaps: 80

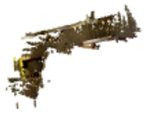
(c) Number of submaps: 120

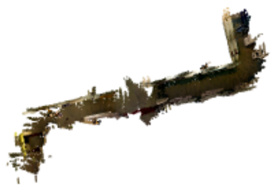
(d) Number of submaps: 160

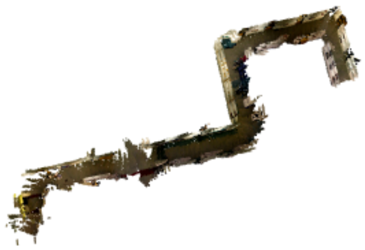
(e) Number of submaps: 200

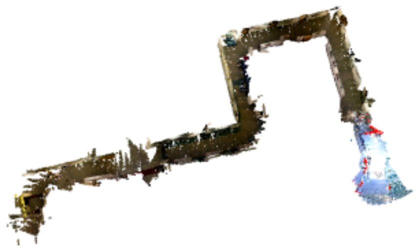
(f) Number of submaps: 218 (final result)

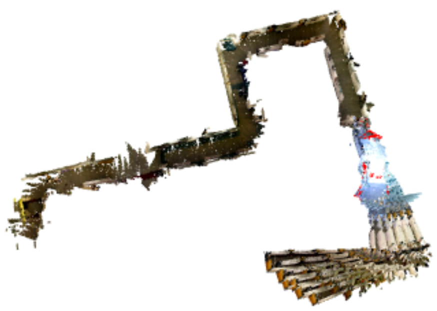
Figure B.1: An example of the mapping process of the API-SLAM system (corresponds to the dataset and result in Fig. 6.4-b). Some submaps are colored in red to show the locations where ambiguous odometry estimates occur.

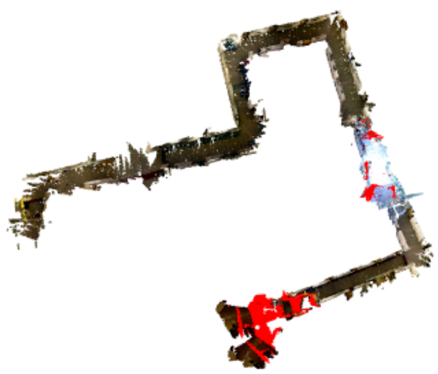

(a) Number of submaps: 40

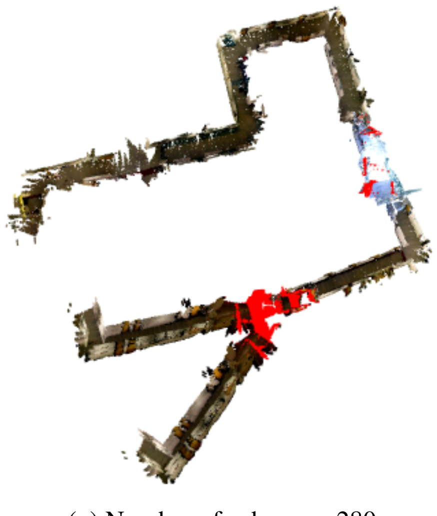

(b) Number of submaps: 80


(c) Number of submaps: 120


(d) Number of submaps: 160


(e) Number of submaps: 200


(f) Number of submaps: 240


(g) Number of submaps: 280

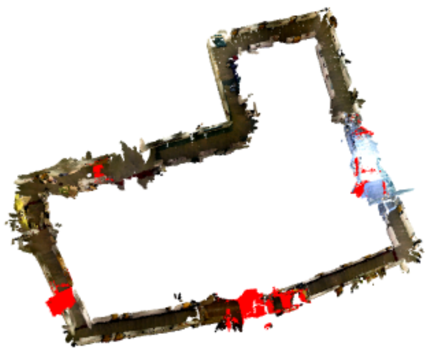

(h) Number of submaps: 332 (final result)

Figure B.2: An example of the mapping process of the API-SLAM system (corresponds to the dataset and result in Fig. 6.4-c). Some submaps are colored in red to show the locations where ambiguous odometry estimates occur.

(a) Number of submaps: 30

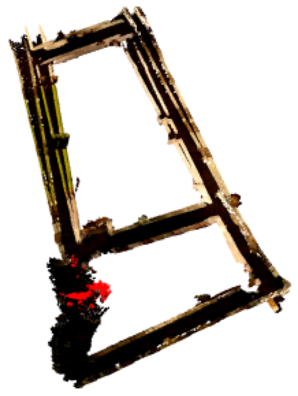
(b) Number of submaps: 60

(c) Number of submaps: 90

(d) Number of submaps: 120

(e) Number of submaps: 150

(f) Number of submaps: 180


(g) Number of submaps: 210

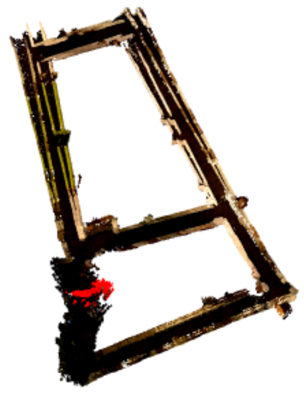
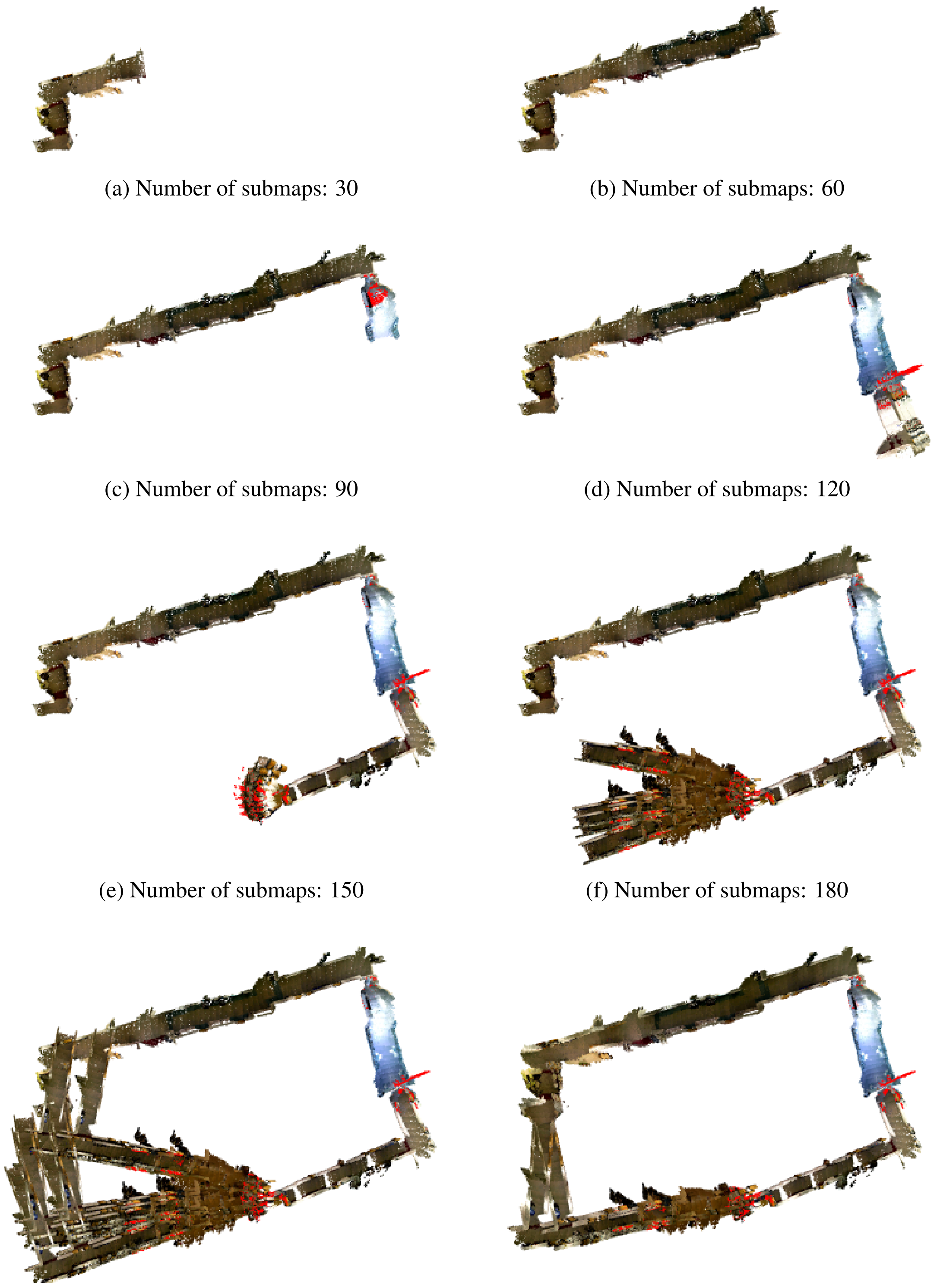

(h) Number of submaps: 258 (final result)

Figure B.3: An example of the mapping process of the API-SLAM system (corresponds to the dataset and result in Fig. 6.4-e).



(a) Number of submaps: 30

(b) Number of submaps: 60

(c) Number of submaps: 90

(d) Number of submaps: 120

(e) Number of submaps: 150

(f) Number of submaps: 180

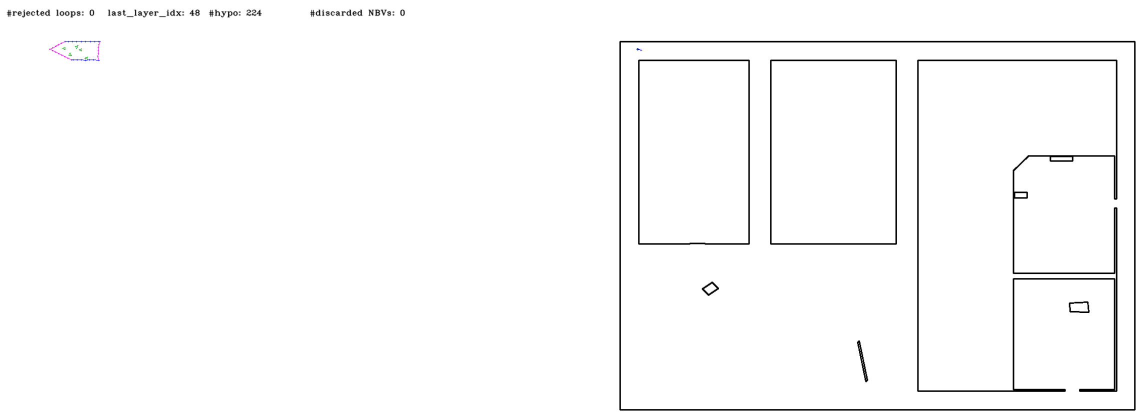
(g) Number of submaps: 210

(h) Number of submaps: 244 (final result)

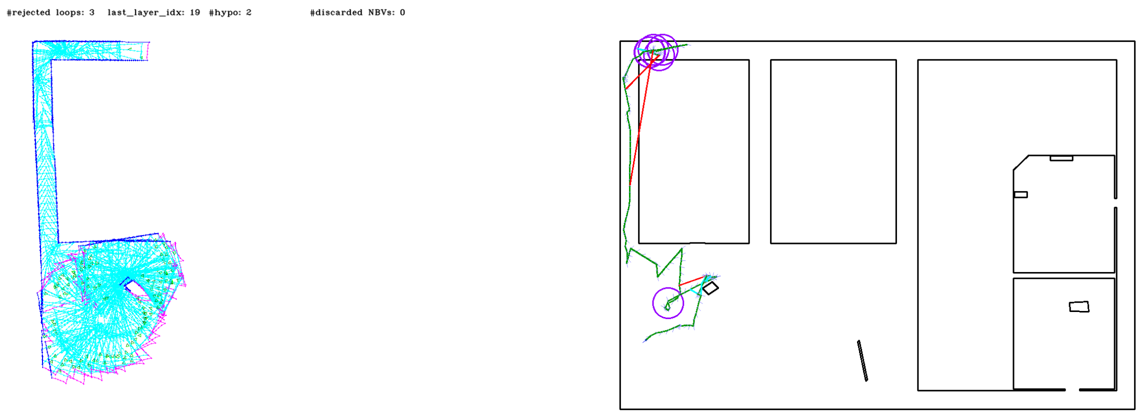
Figure B.4: An example of the mapping process of the API-SLAM system (corresponds to the dataset and result in Fig. 6.8-a).

B.2 Simulation of Ambiguity-aware Active SLAM

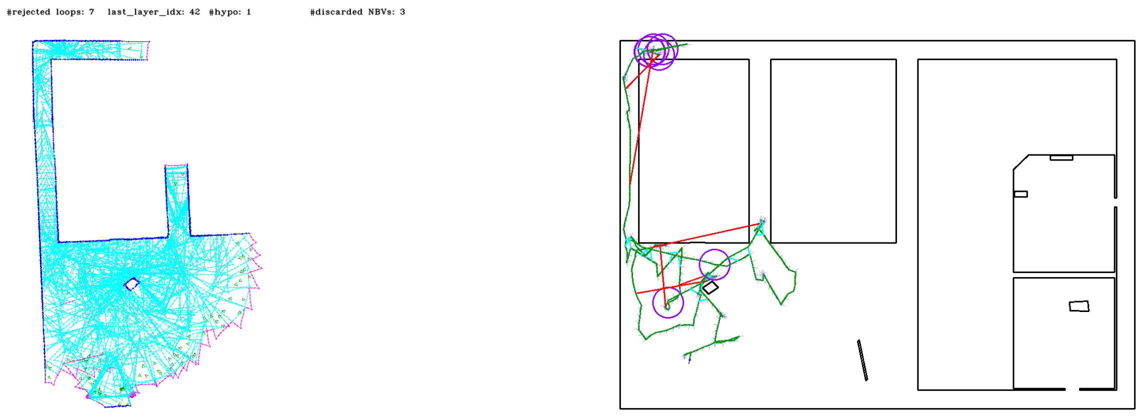
Fig. B.5 and Fig. B.6 show two examples of the simulation steps during the evaluation of the ambiguity-aware active SLAM approach (see Sec. 7.7). We take the snapshots of both the online reconstructed maps and the ground truth information each time when the number of submaps grows, and shows the uniformly downsampled set of images here. In both examples, we set $n_{\mathrm{trigger}} = 4$ and $n_{\mathrm{limit}} = \infty$.


#rejected loops: 0 last_layer_idx: 48 #hypo: 224 #discarded NBVs: 0

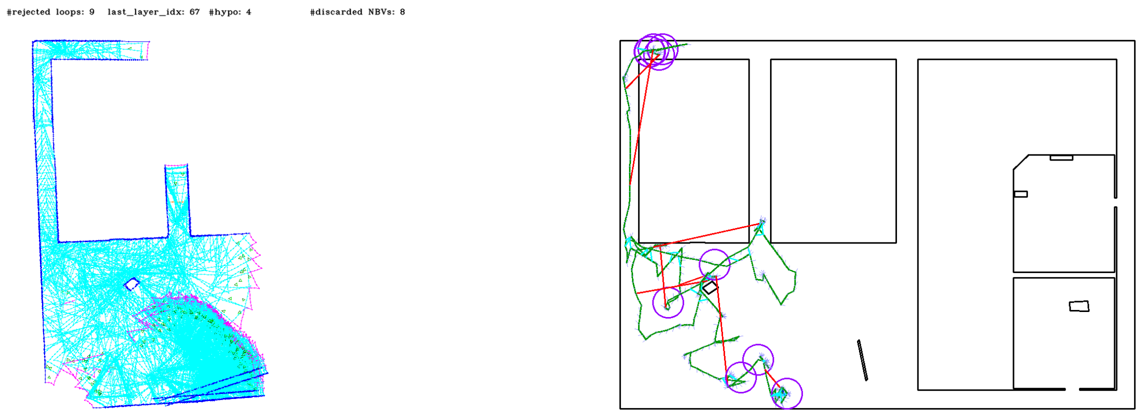
(a) Number of submaps: 1


#rejected loops: 3 last_layer_idx: 19 #hypo: 2 #discarded NBVs: 0

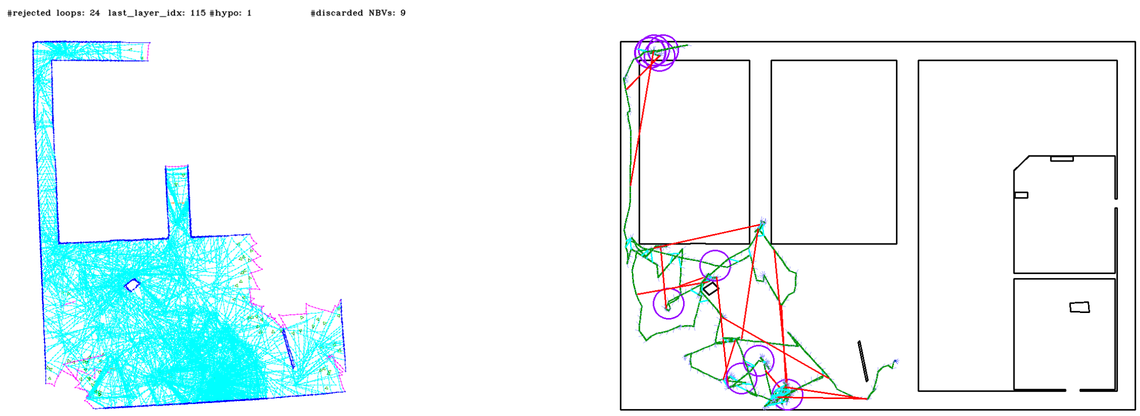
(b) Number of submaps: 201


#rejected loops: 7 last_layer_idx: 42 #hypo: 1 #discarded NBVs: 3

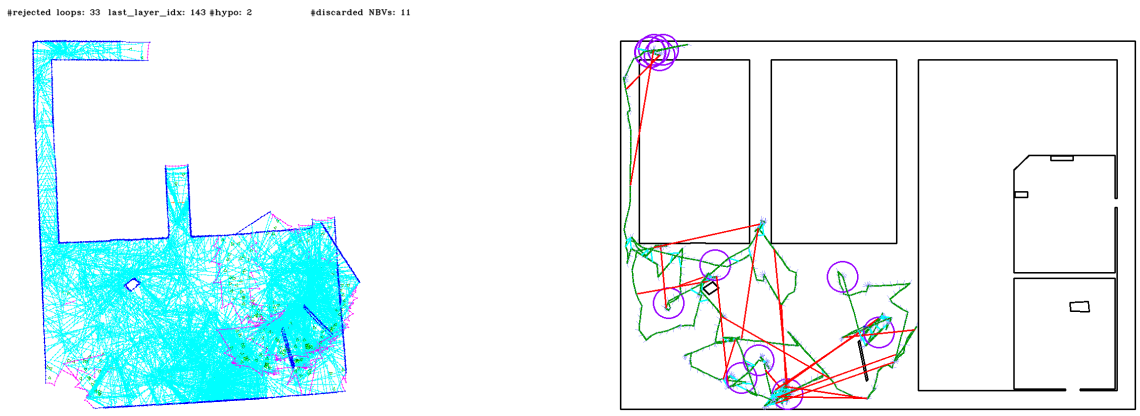
(c) Number of submaps: 401



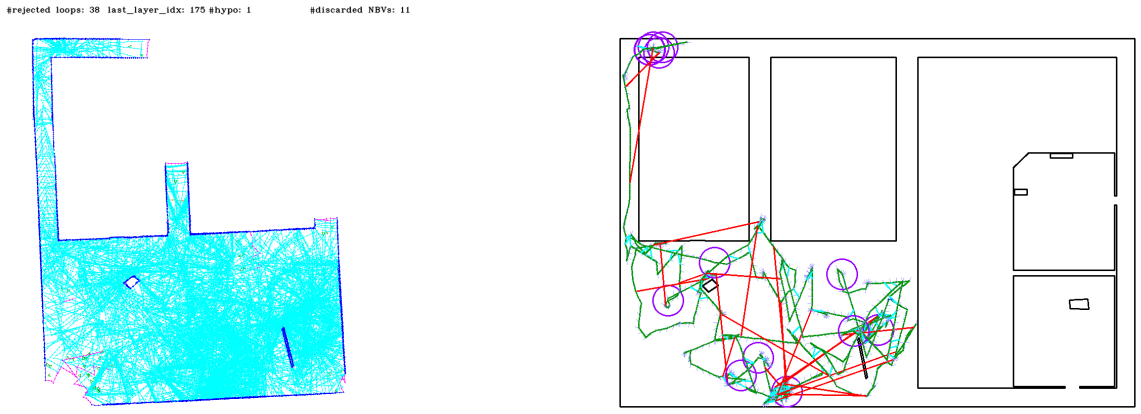
(d) Number of submaps: 601



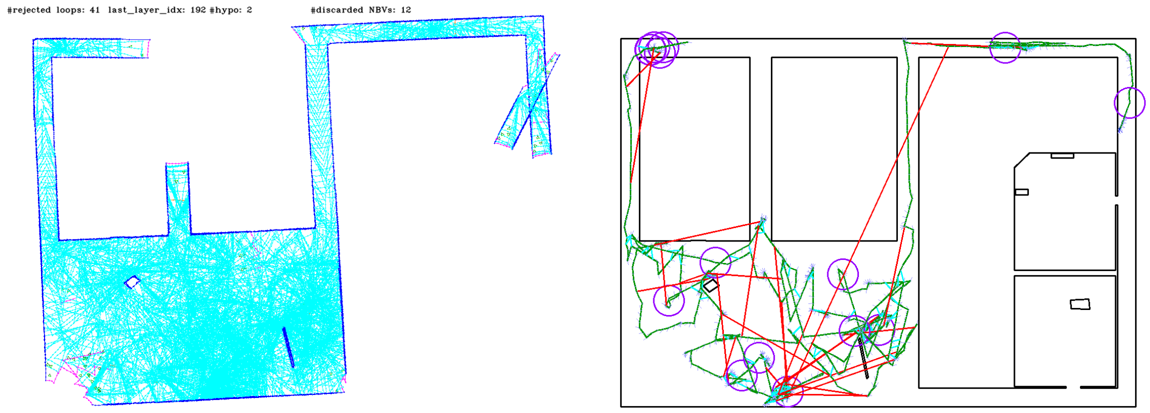
(e) Number of submaps: 801



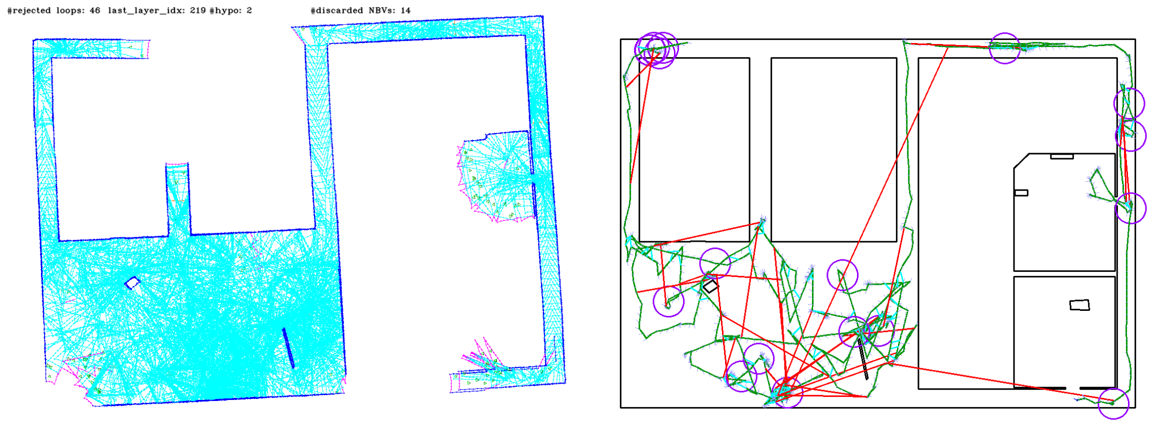
(f) Number of submaps: 1001



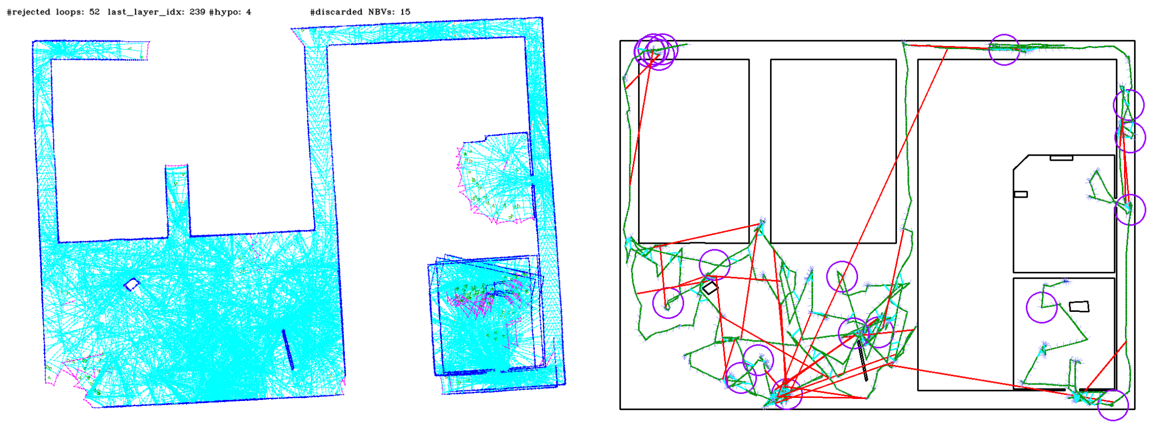
(g) Number of submaps: 1201



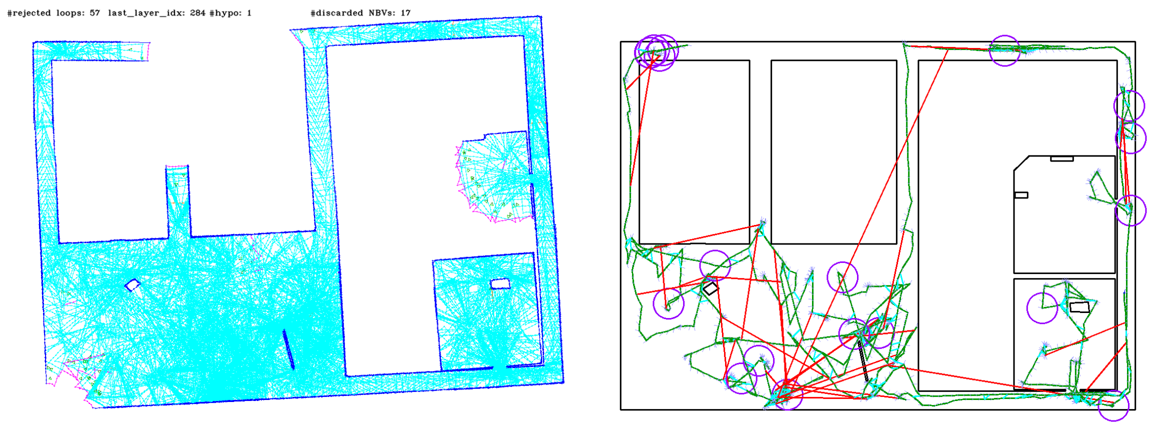
(h) Number of submaps: 1401



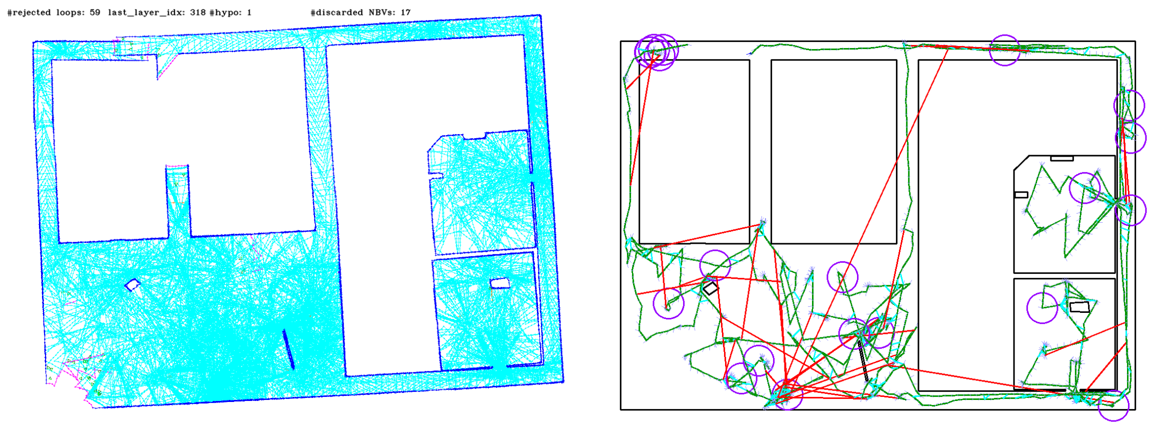
(i) Number of submaps: 1601



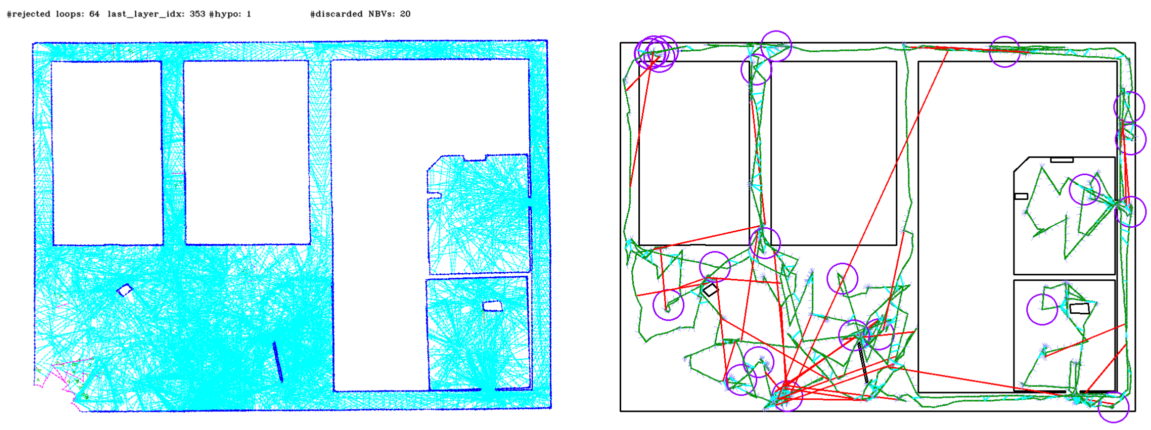
(j) Number of submaps: 1801



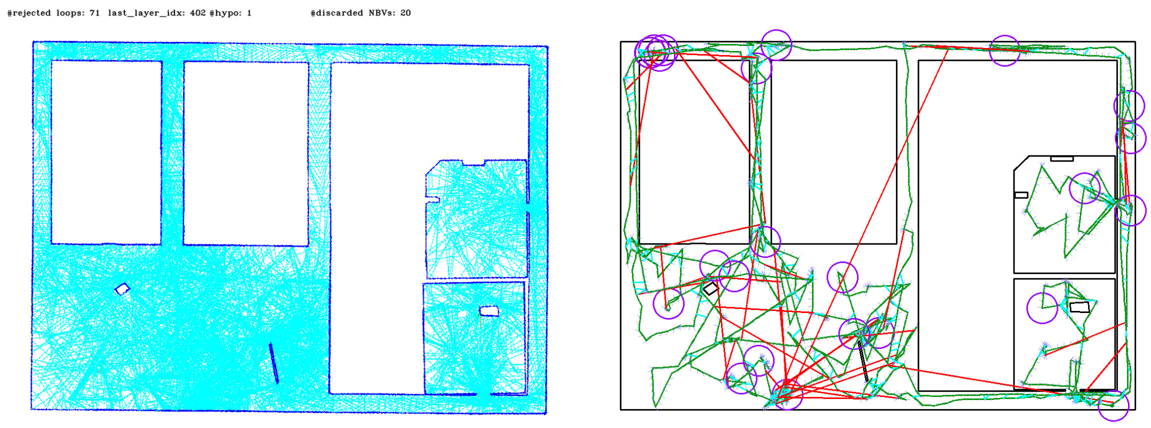
(k) Number of submaps: 2001



(l) Number of submaps: 2201



(m) Number of submaps: 2401

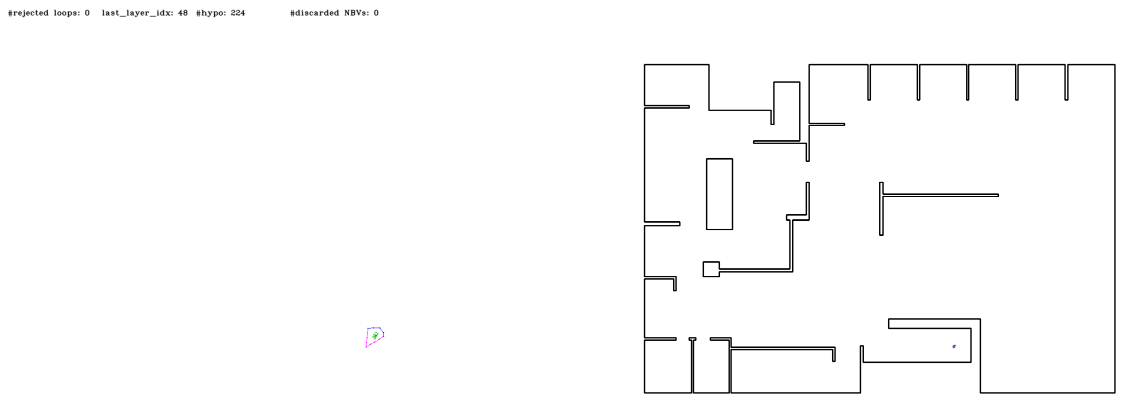


(n) Number of submaps: 2598 (final result)

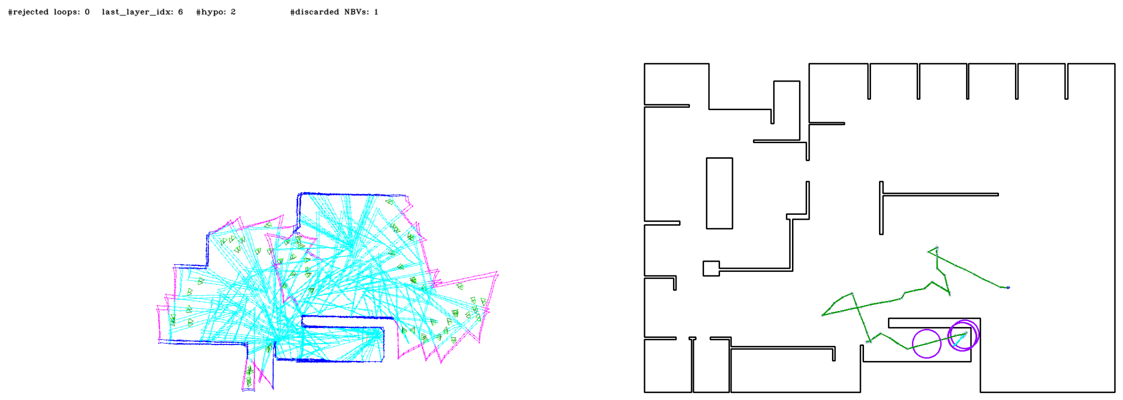
Left column: ▽ Robot pose — Boundary edge —/— Uncovered/covered frontier edge

Right column: — Trajectory ○ Ambiguous odometry —/— True/false positive loop

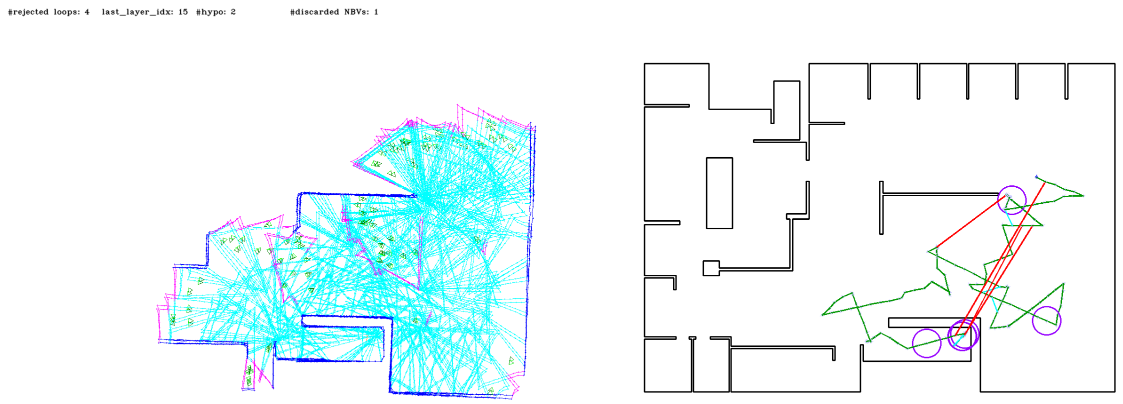
Figure B.5: An example of the simulation process of the ambiguity-aware active SLAM framework in the simulated environment shown in Fig. 7.7-b. The left column shows the reconstructed maps in all hypotheses, while the right column shows the ground truth map and trajectory.


#rejected loops: 0   last_layer_idx: 48   #hypo: 224   #discarded NBVs: 0

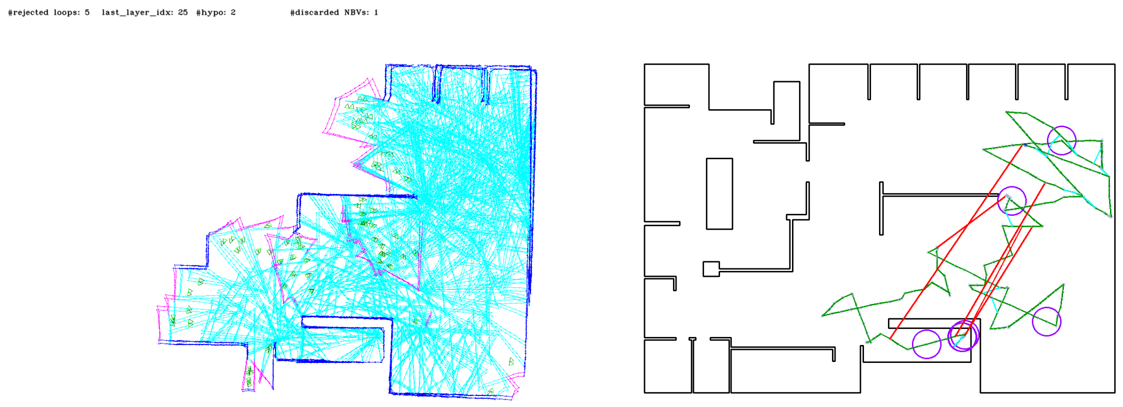
(a) Number of submaps: 1


#rejected loops: 0   last_layer_idx: 6   #hypo: 2   #discarded NBVs: 1

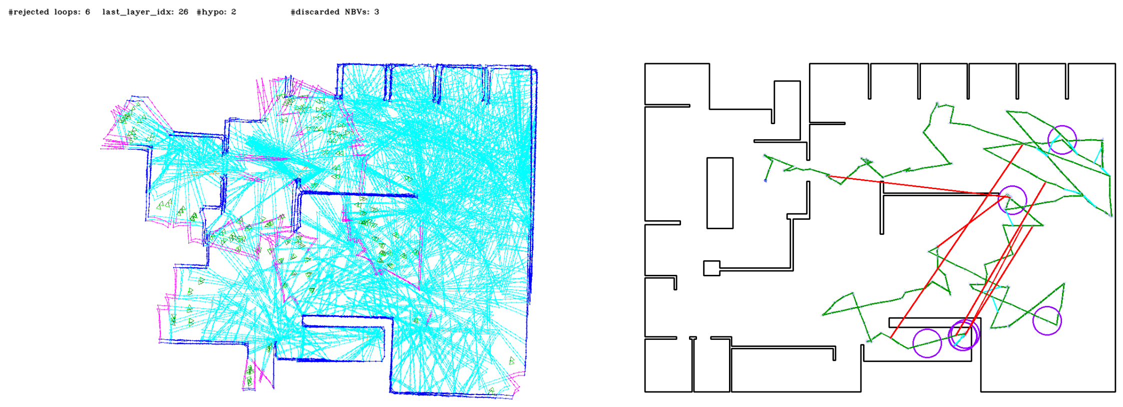
(b) Number of submaps: 101


#rejected loops: 4   last_layer_idx: 15   #hypo: 2   #discarded NBVs: 1

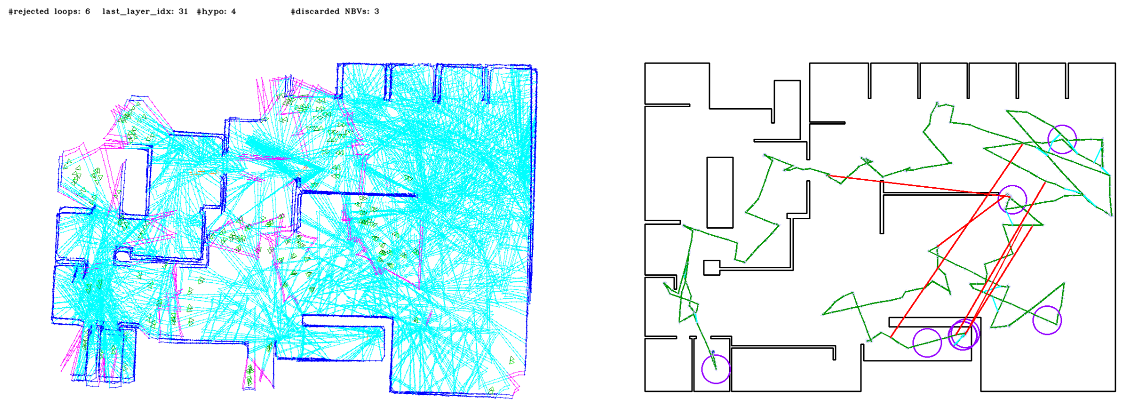
(c) Number of submaps: 201


#rejected loops: 5 last_layer_idx: 25 #hypo: 2 #discarded NBVs: 1

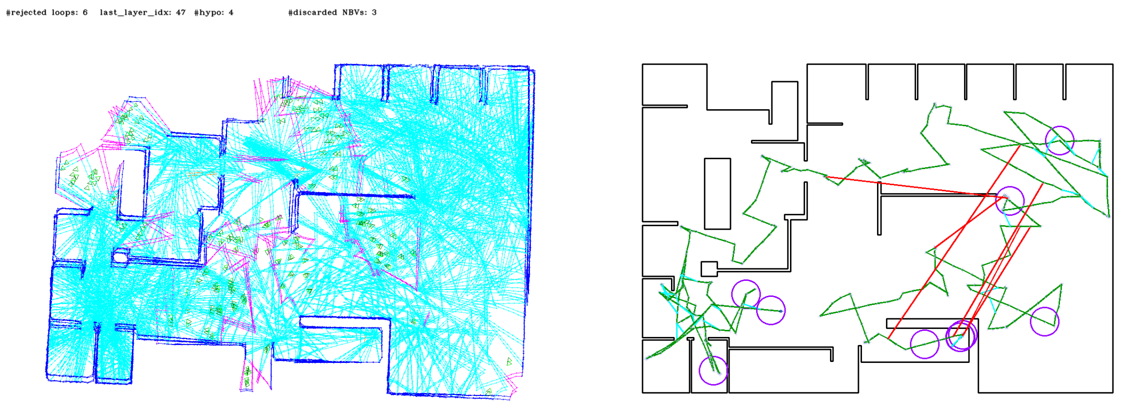
(d) Number of submaps: 301


#rejected loops: 6 last_layer_idx: 26 #hypo: 2 #discarded NBVs: 3

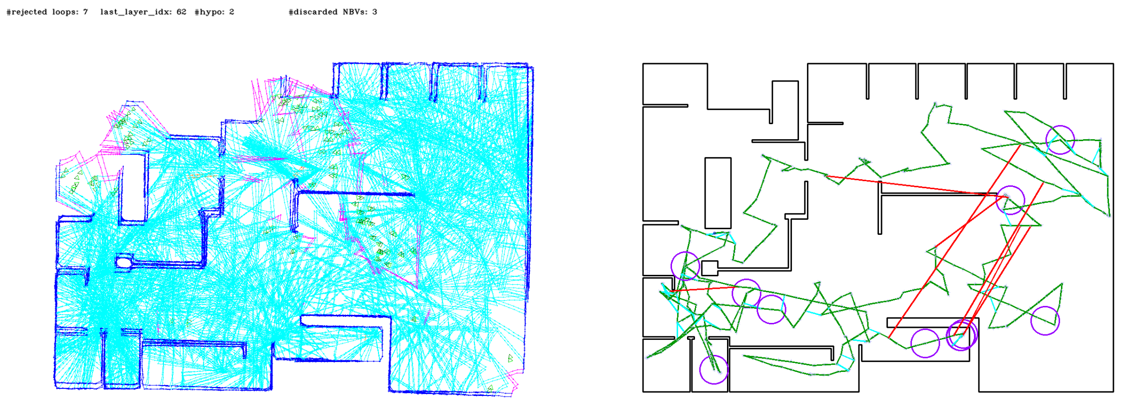
(e) Number of submaps: 401


#rejected loops: 6 last_layer_idx: 31 #hypo: 4 #discarded NBVs: 3

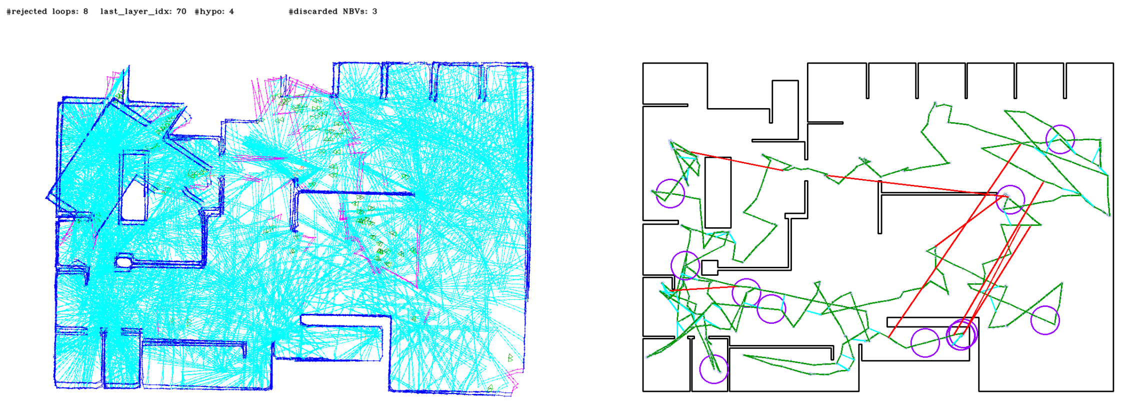
(f) Number of submaps: 501



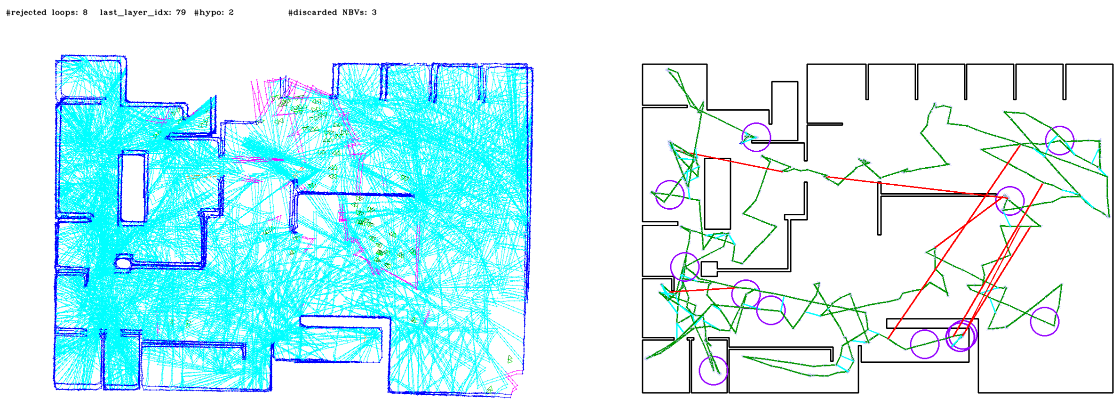
(g) Number of submaps: 601



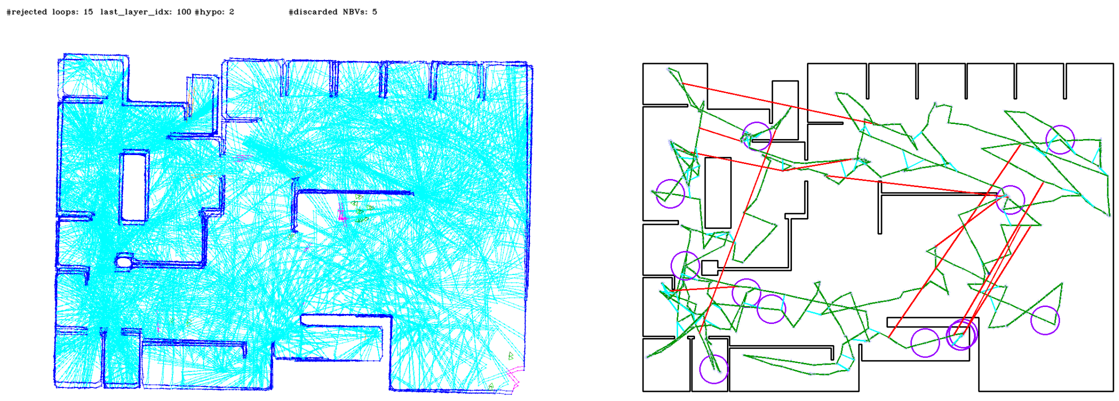
(h) Number of submaps: 701



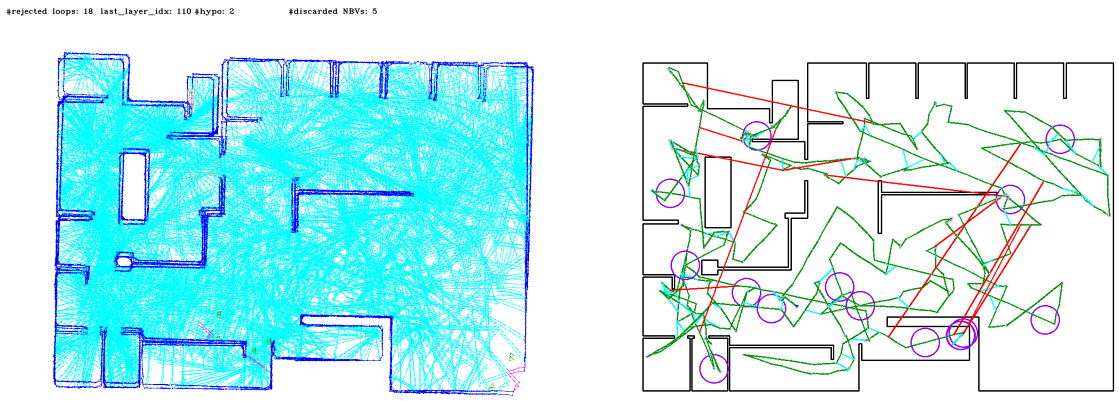
(i) Number of submaps: 801



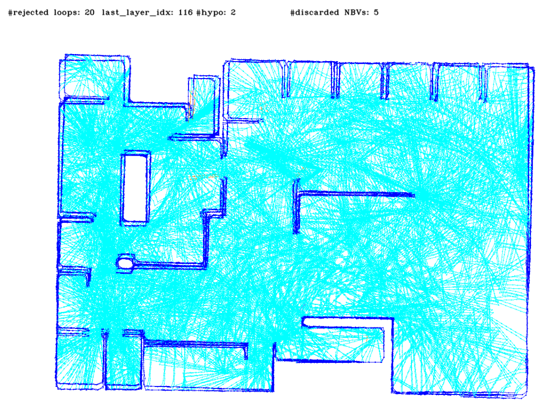
(j) Number of submaps: 901

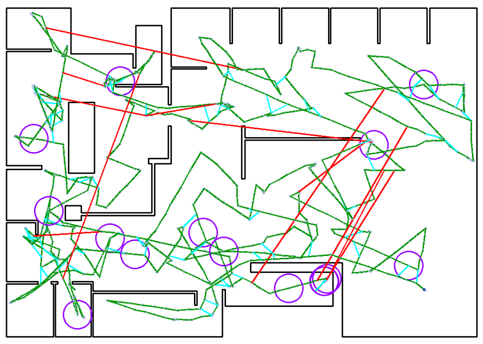


(k) Number of submaps: 1001



(l) Number of submaps: 1101





(m) Number of submaps: 1133 (final result)

Left column: ▽ Robot pose — Boundary edge —/— Uncovered/covered frontier edge

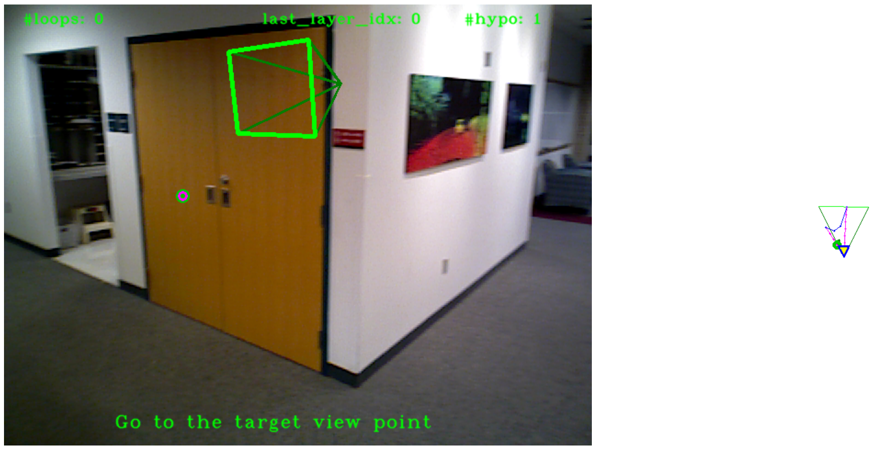
Right column: — Trajectory ○ Ambiguous odometry —/— True/false positive loop

Figure B.6: An example of the simulation process of the ambiguity-aware active SLAM framework in another simulated environment shown in Fig. 7.7-f. Again, the left column shows the reconstructed maps in all hypotheses, while the right column shows the ground truth map and trajectory. Notice that different from the example result in Fig. B.5-n, in this example the final output map still contains two similar hypotheses that cannot be disambiguated.

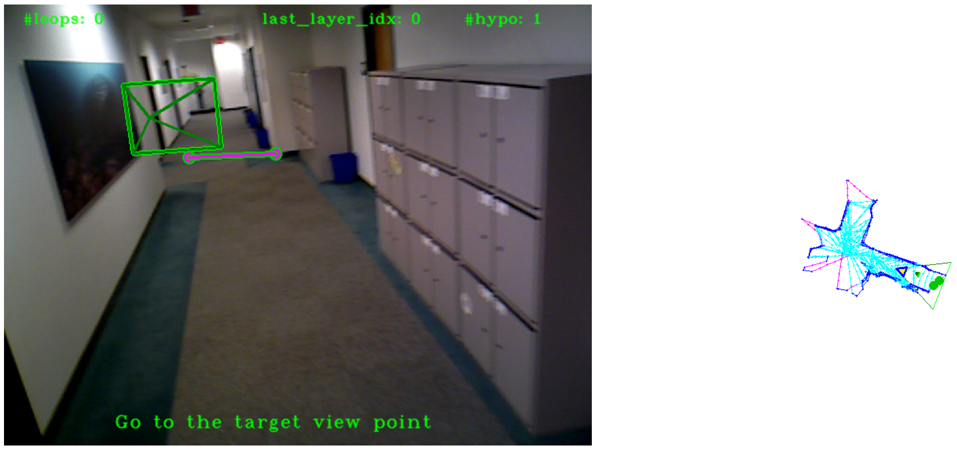
B.3 Real World Results of Assistive Mapping System

Fig. B.7 shows one example of the active mapping process using the assistive mapping system (see Sec. 7.7), where its final output 3D model is visualized and evaluated in Fig. 7.16. Similar to Fig. B.5, we take the snapshots of both the online AR viewer and the top-down viewer each time when the number of submaps grows, and show the uniformly downsampled set of images here. In this example, we set $n_{\text{trigger}} = 2$ and $n_{\text{limit}} = 16$, which is able to run in real-time on a CPU.

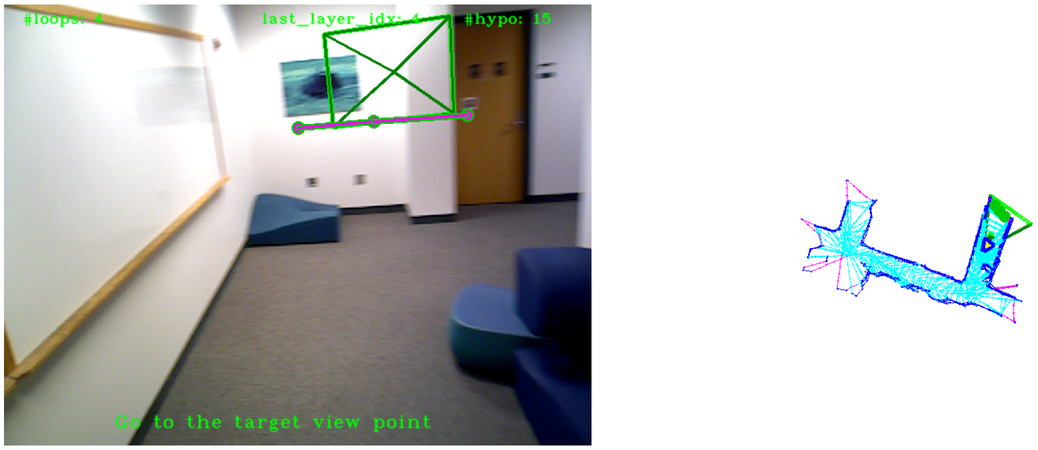
We also test the assistive mapping system with one single hypothesis ($n_{\text{limit}} = n_{\text{trigger}} = 1$), which runs into numerical issue and crashes soon after a false positive loop closure measurement being added into the optimization in the example shown in Fig. B.8.



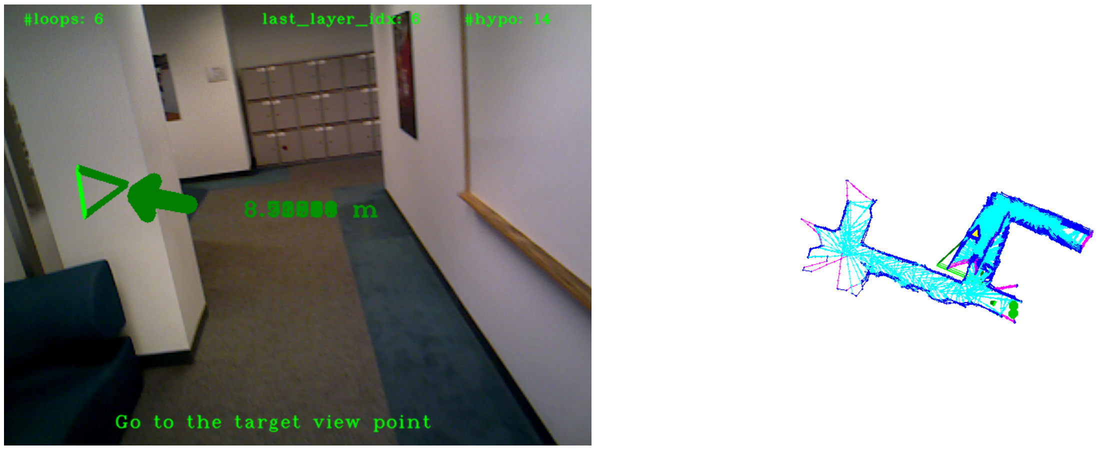
(a) Number of submaps: 1



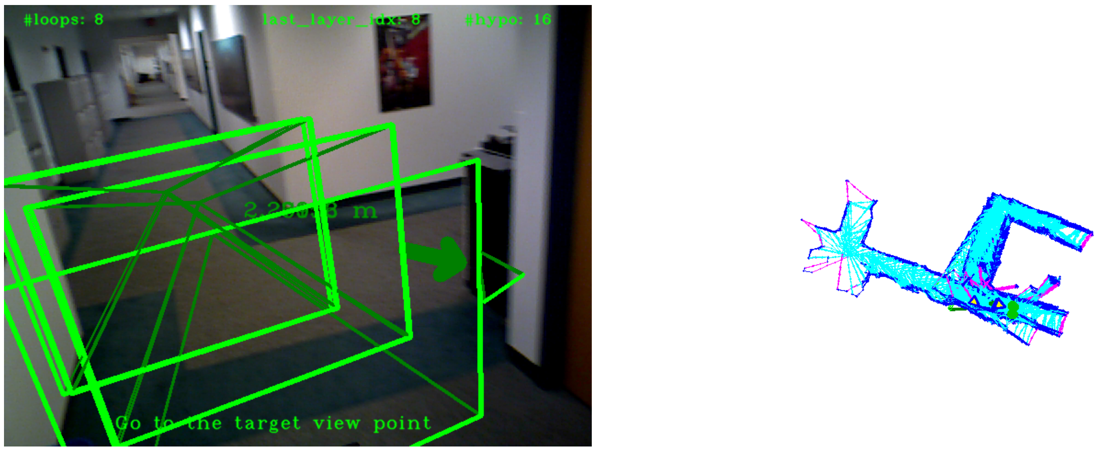
(b) Number of submaps: 41



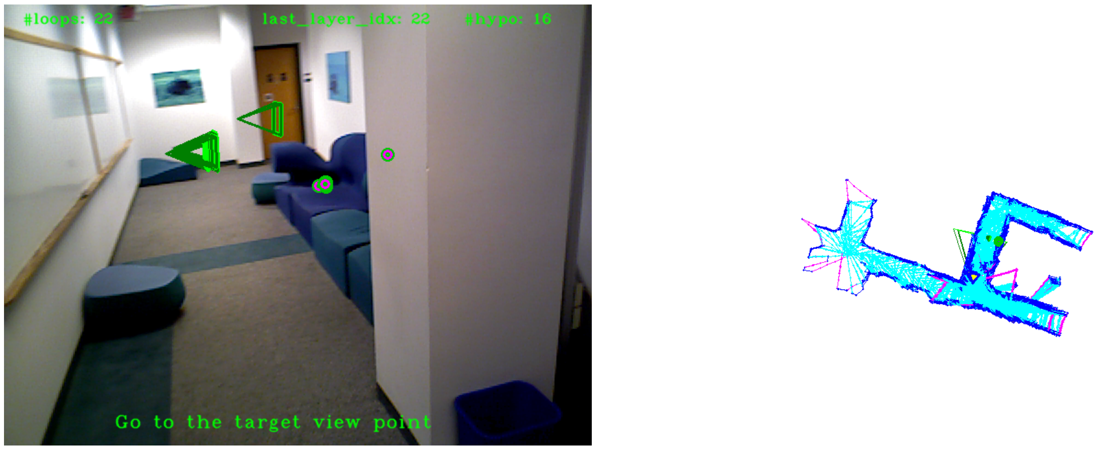
(c) Number of submaps: 81



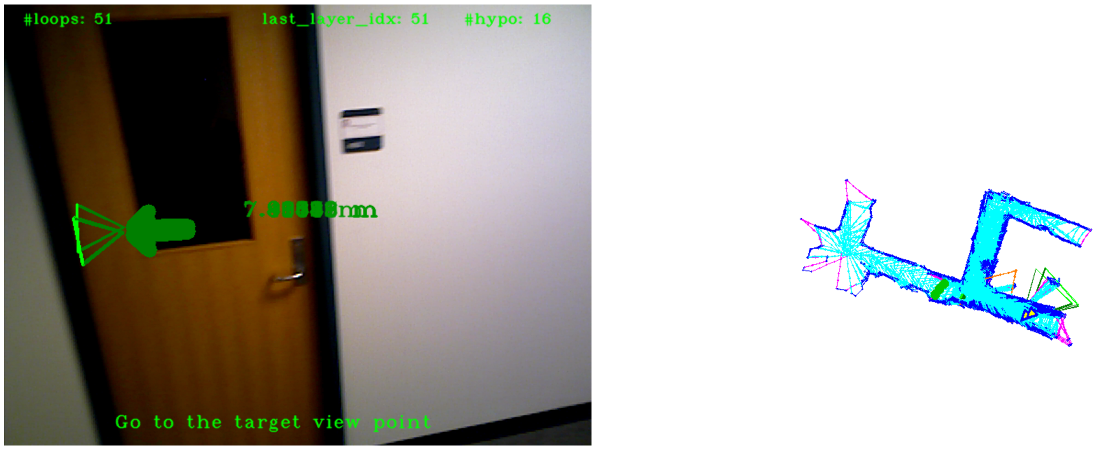
(d) Number of submaps: 121



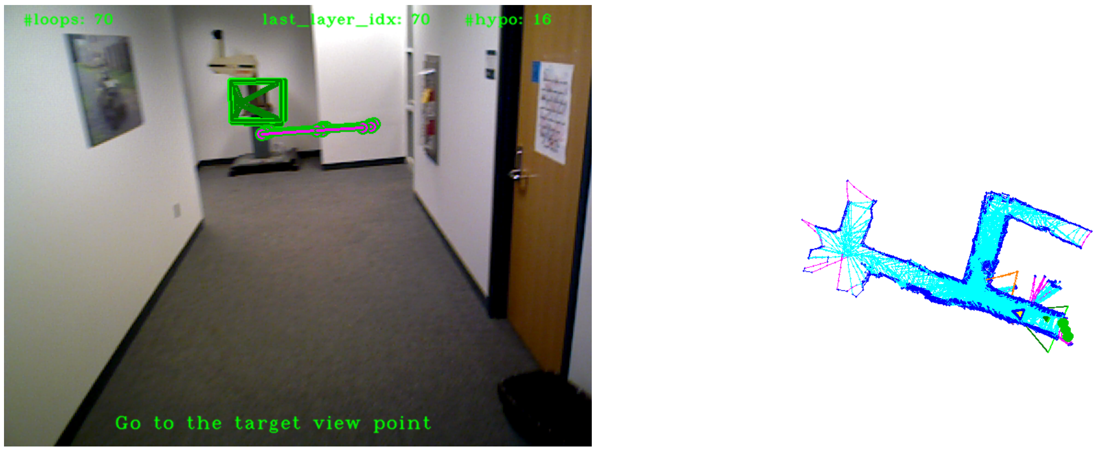
(e) Number of submaps: 161



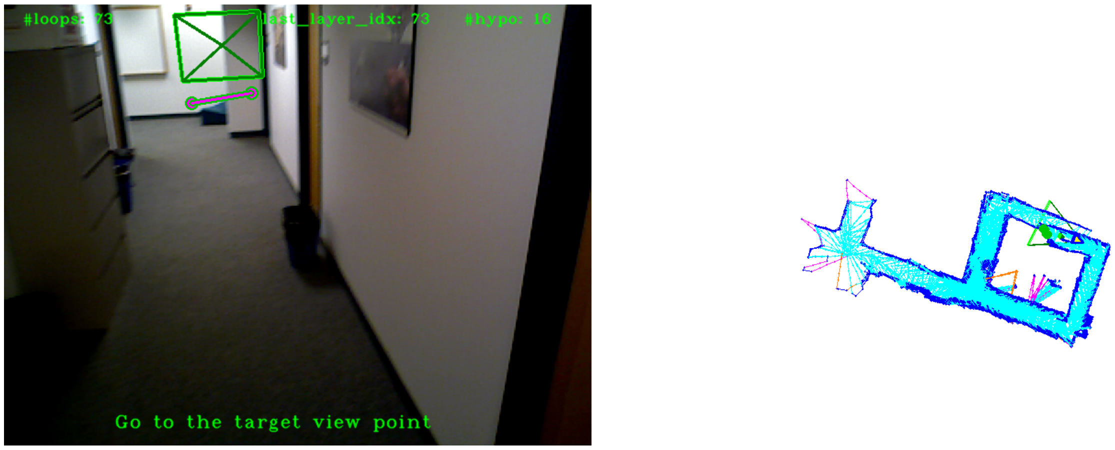
(f) Number of submaps: 201



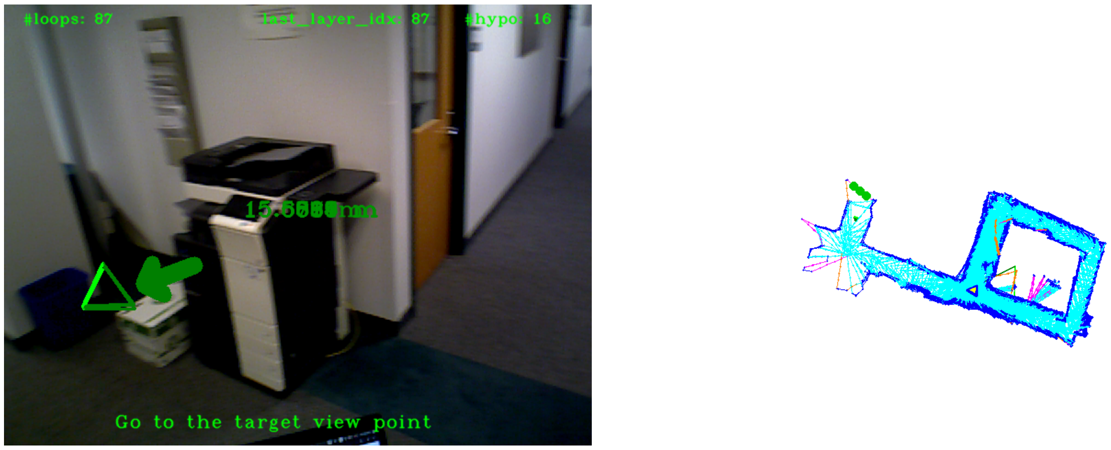
(g) Number of submaps: 241



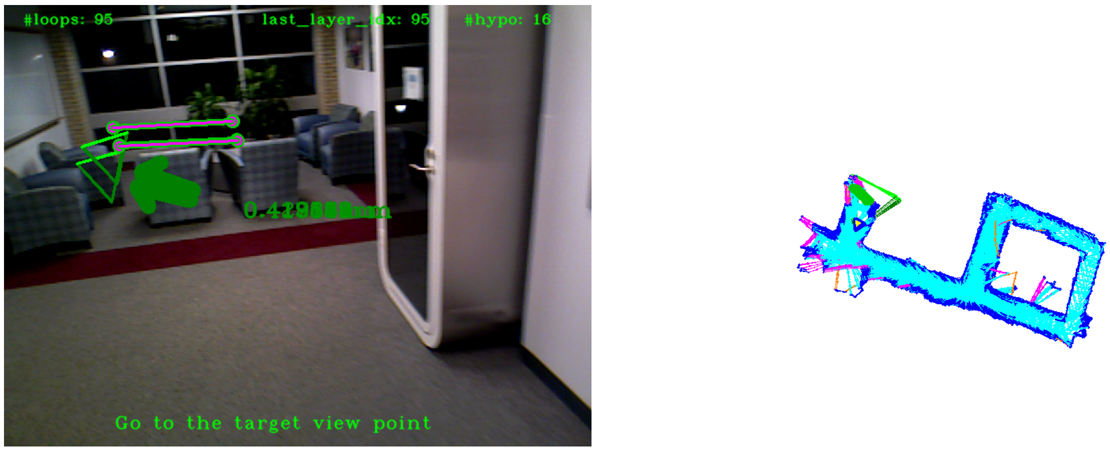
(h) Number of submaps: 281



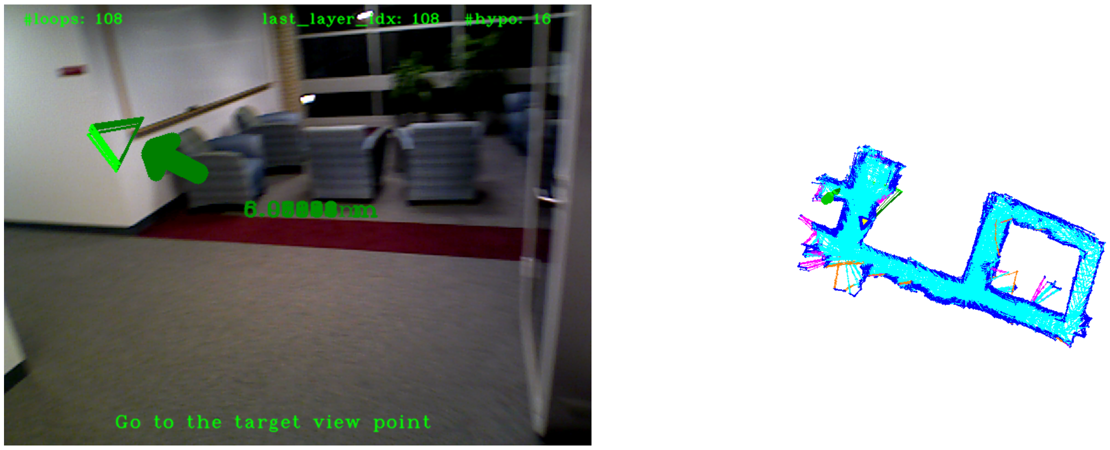
(i) Number of submaps: 321



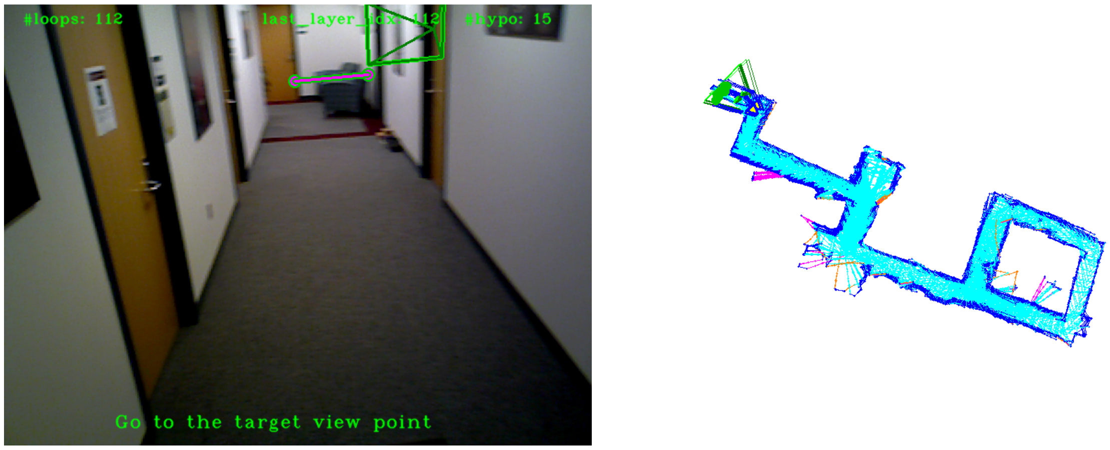
(j) Number of submaps: 361



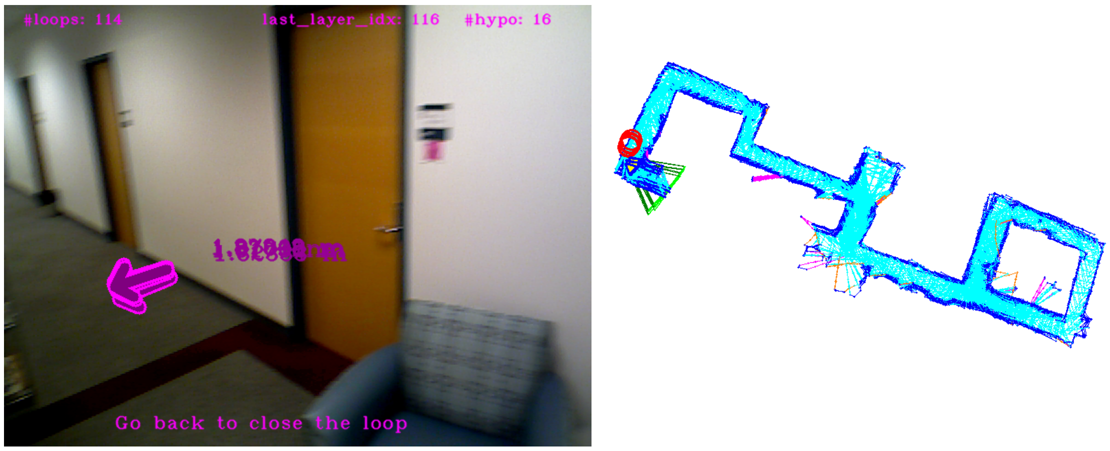
(k) Number of submaps: 401



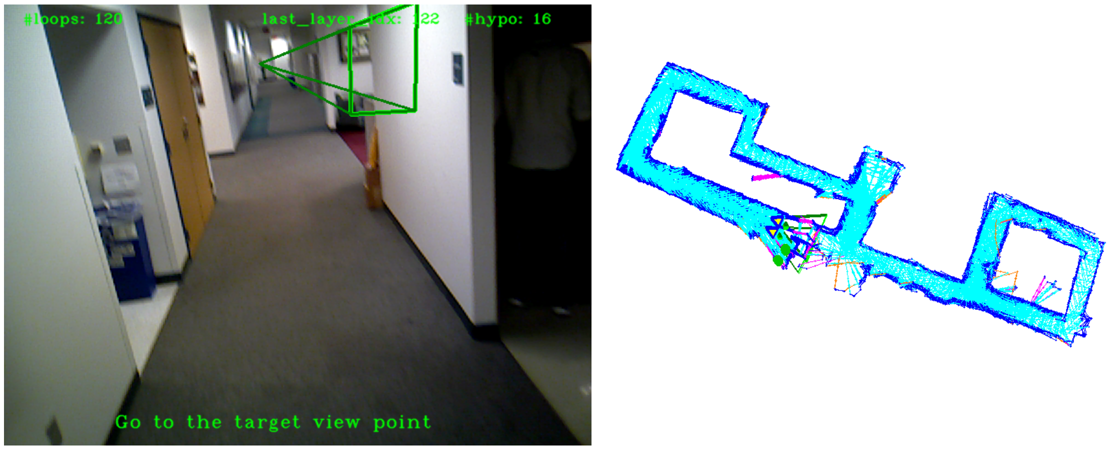
(l) Number of submaps: 441



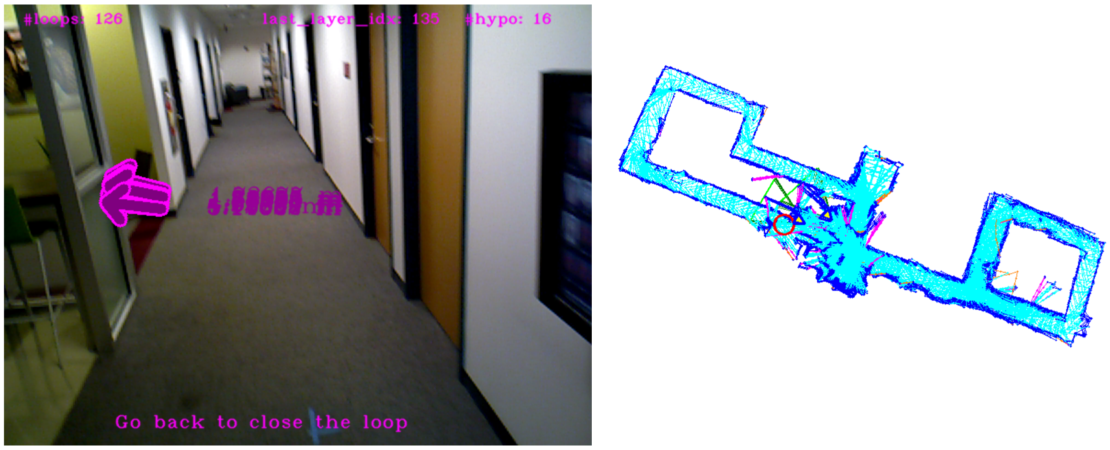
(m) Number of submaps: 481



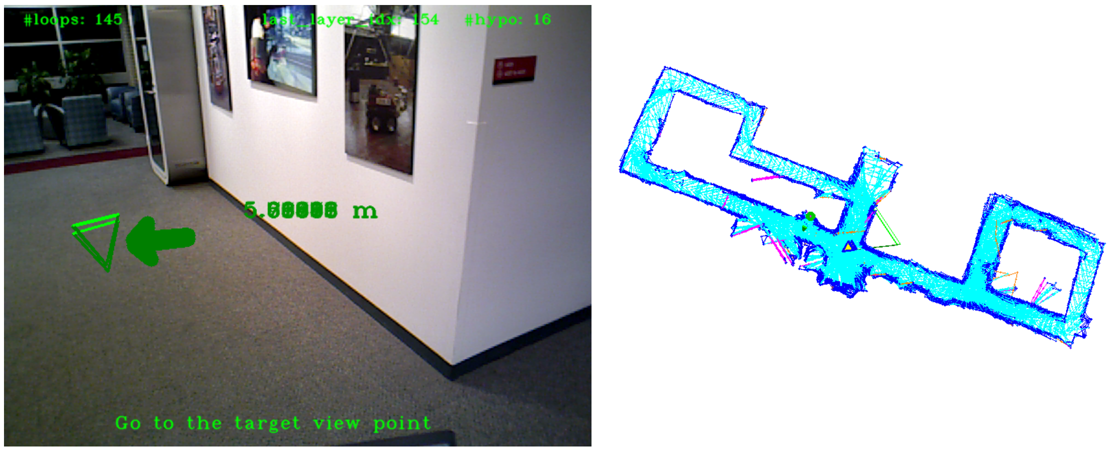
(n) Number of submaps: 521



(o) Number of submaps: 561



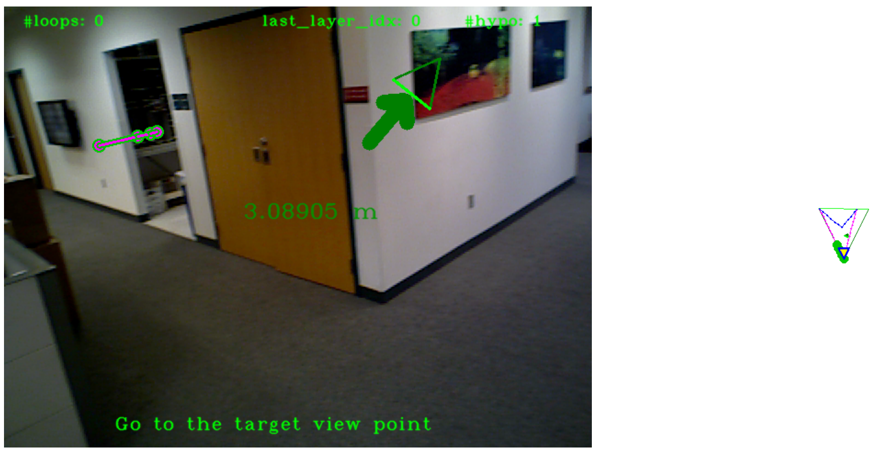
(p) Number of submaps: 601



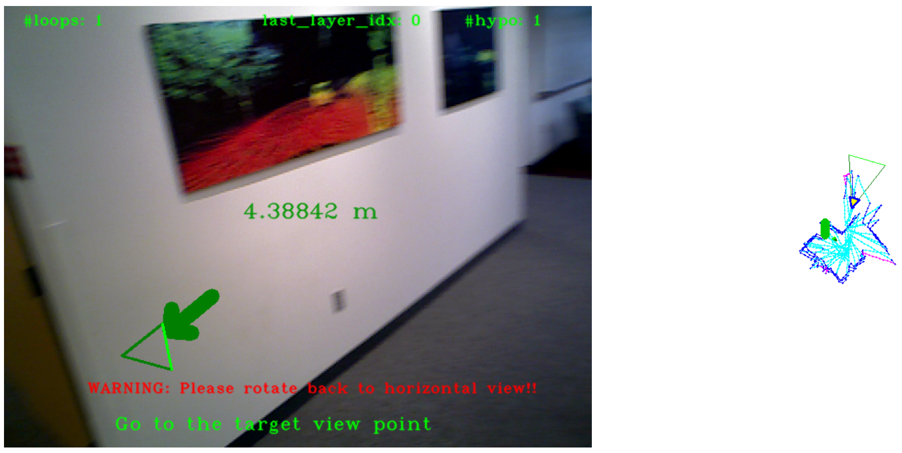
(q) Number of submaps: 637 (final result)

▽ Current pose and view ●●● Target frontier ▽ Target view point ○ Revisiting target

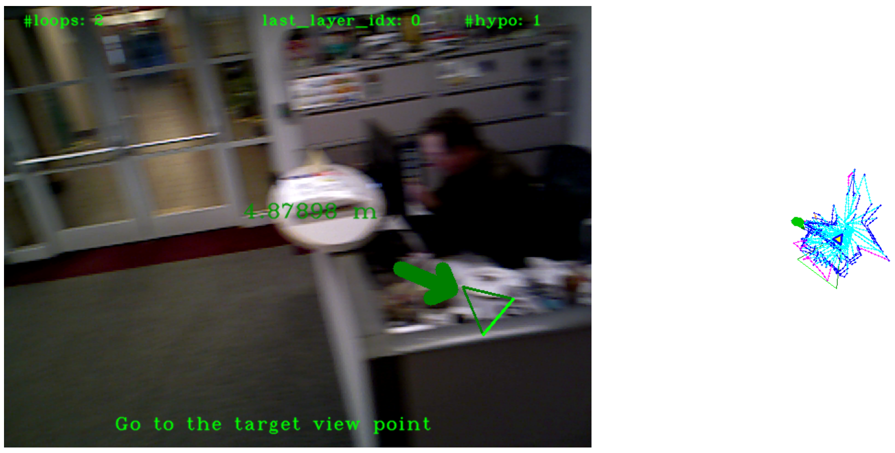
Figure B.7: An example of the mapping process using the assistive mapping system. The left column is the AR viewer, and the right column is the corresponding top-down viewer that shows the reconstructed maps in all hypotheses. Instructions (target view points P^*, its corresponding target frontiers, and the poses of the loop closing target submaps M^*) are visualized in both of them online in real-time.



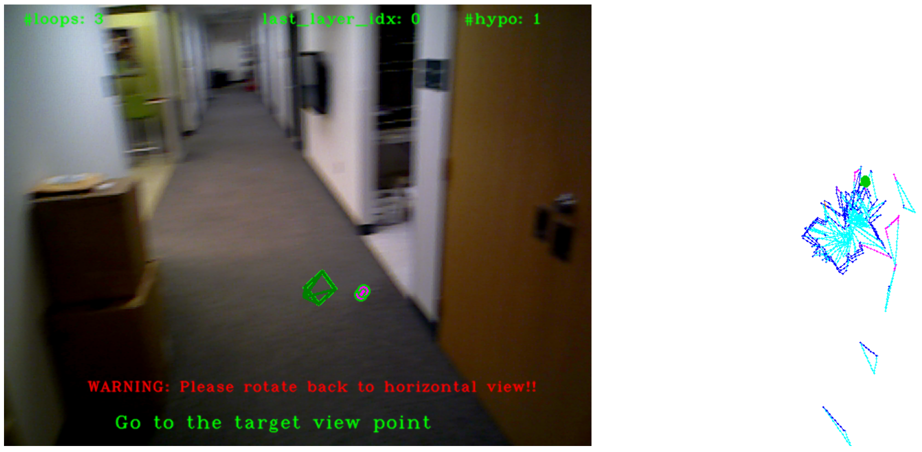
(a) Number of submaps: 1


#loops: 1
last_layer_idx: 0
#hypo: 1
4.38842 m
WARNING: Please rotate back to horizontal view!!
Go to the target view point

(b) Number of submaps: 21



(c) Number of submaps: 41


#loops: 3 last_layer_idx: 0 #hypo: 1

WARNING: Please rotate back to horizontal view!!

Go to the target view point

(d) Number of submaps: 51 (crash here)

▽ Current pose and view ●●● Target frontier ▽ Target view point ○ Revisiting target

Figure B.8: An example of the mapping process using the assistive mapping system with only a single hypothesis. The left column is the AR viewer, and the right column is the corresponding top-down viewer that shows the reconstructed maps in all hypotheses.

Bibliography

[1] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. "Robust map optimization using dynamic covariance scaling". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 62–69. DOI: 10.1109/ICRA.2013.6630557.

[2] S. Agarwal, K. Mierle, and Others. *Ceres Solver*. http://ceres-solver.org.

[3] A. Agha-mohammadi, S. Agarwal, A. Mahadevan, S. Chakravorty, D. Tomkins, J. Denny, and N. M. Amato. "Robust online belief space planning in changing environments: Application to physical mobile robots". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 149–156. DOI: 10.1109/ICRA.2014.6906602.

[4] H. Baltzakis and P. Trahanias. "Using multi-hypothesis mapping to close loops in complex cyclic environments". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2006, pp. 824–829. DOI: 10.1109/ROBOT.2006.1641811.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[6] P. J. Besl and H. D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. "Receding Horizon "Next-Best-View" Planner for 3D Exploration". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2016, pp. 1462–1468. DOI: 10.1109/ICRA.2016.7487281.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Trans. Robotics* 32.6 (2016), pp. 1309–1332. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: Binary Robust Independent Elementary Features". In: *Eur. Conf. on Computer Vision (ECCV)*. ECCV'10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 778–792. ISBN: 3-642-15560-X, 978-3-

642-15560-4. URL: http://dl.acm.org/citation.cfm?id=1888089.1888148.

[10] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri. "Active SLAM and Exploration with Particle Filters Using Kullback-Leibler Divergence". In: *Journal of Intelligent and Robotic Systems* 75 (2014), pp. 291–311.

[11] S. M. Chaves, A. Kim, and R. M. Eustice. "Opportunistic sampling-based planning for active visual SLAM". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2014, pp. 3073–3080. DOI: 10.1109/IROS.2014.6942987.

[12] Y. Chen, S. Huang, R. Fitch, L. Zhao, H. Yu, and D. Yang. "On-line 3D active pose-graph SLAM based on key poses using graph topology and sub-maps". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 169–175. DOI: 10.1109/ICRA.2019.8793632.

[13] A. Concha, G. Loianno, V. Kumar, and J. Civera. "Visual-inertial direct SLAM". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2016, pp. 1331–1338. DOI: 10.1109/ICRA.2016.7487266.

[14] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt. "BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration". In: *ACM Transactions on Graphics 2017 (TOG)* (2017).

[15] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. "A Column Approximate Minimum Degree Ordering Algorithm". In: *ACM Trans. Math. Softw.* 30.3 (Sept. 2004), pp. 353–376. ISSN: 0098-3500. DOI: 10.1145/1024074.1024079. URL: http://doi.acm.org/10.1145/1024074.1024079.

[16] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1049.

[17] F. Dellaert and M. Kaess. "Factor Graphs for Robot Perception". In: *Foundations and Trends in Robotics* 6.1-2 (Aug. 2017). http://dx.doi.org/10.1561/2300000043, pp. 1–139.

[18] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks". In: *Conf. on Uncertainty in Artificial Intelligence (UAI)*. 2000, pp. 176–183.

[19] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. "An Evaluation of the RGB-D SLAM System". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. St. Paul, MA, USA, 2012.

[20] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *Eur. Conf. on Computer Vision (ECCV)*. 2014.

[21] J. Engel, J. Sturm, and D. Cremers. “Semi-Dense Visual Odometry for a Monocular Camera”. In: *Intl. Conf. on Computer Vision (ICCV)*. Sydney, Australia, 2013.

[22] C. Erdogan, M. Paluri, and F. Dellaert. “Planar Segmentation of RGBD Images Using Fast Linear Fitting and Markov Chain Monte Carlo”. In: *Proceedings of the Ninth Conference on Computer and Robot Vision*. CRV ’12. Washington, DC, USA, 2012, pp. 32–39. ISBN: 978-0-7695-4683-4.

[23] *FARO Focus 3D survey lidar scanner*. URL: `https://www.faro.com/en-gb/products/construction-bim-cim/faro-focus/`.

[24] M. A. Fischler and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782.

[25] A. Flint, C. Mei, I. Reid, and D. Murray. “Growing semantically meaningful models for visual SLAM”. In: *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*. 2010, pp. 467–474. DOI: `10.1109/CVPR.2010.5540176`.

[26] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry”. In: *IEEE Trans. Robotics* 33.1 (2017), pp. 1–21. ISSN: 1552-3098. DOI: `10.1109/TRO.2016.2597321`.

[27] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems”. In: *IEEE Trans. Robotics* 33.2 (2017), pp. 249–265.

[28] D. Fourie, J. Leonard, and M. Kaess. “A nonparametric belief solution to the Bayes tree”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 2189–2196. DOI: `10.1109/IROS.2016.7759343`.

[29] D. Galvez-Lopez and J. D. Tardos. “Bags of Binary Words for Fast Place Recognition in Image Sequences”. In: *IEEE Trans. Robotics* (2012).

[30] A. Geiger, J. Ziegler, and C. Stiller. “StereoScan: Dense 3D reconstruction in real-time”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 963–968. DOI: `10.1109/IVS.2011.5940405`.

[31] W. Givens. “Computation of plain unitary rotations transforming a general matrix to triangular form”. In: *Journal of the Society for Industrial and Applied Mathematics* 6.1 (1958), pp. 26–50.

[32] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez. "PL-SVO: Semi-direct Monocular Visual Odometry by combining points and line segments". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 4211–4216.

[33] G. Graber, T. Pock, and H. Bischof. In: *Intl. Conf. on Computer Vision Workshops (ICCV Workshops)*.

[34] A. Handa, T. Whelan, J. McDonald, and A. Davison. "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. Hong Kong, China, 2014.

[35] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. "Real-time plane segmentation using RGB-D cameras". In: *RoboCup 2011: Robot Soccer World Cup XV*. Springer, 2012, pp. 306–317.

[36] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34 (Apr. 2013). DOI: `10.1007/s10514-012-9321-0`.

[37] A. S. Householder. "Unitary triangularization of a nonsymmetric matrix". In: *Journal of the ACM (JACM)* 5.4 (1958), pp. 339–342.

[38] M. Hsiao and M. Kaess. "MH-iSAM2: Multi-hypothesis iSAM using Bayes Tree and Hypo-tree". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 1274–1280. DOI: `10.1109/ICRA.2019.8793854`.

[39] M. Hsiao, E. Westman, G. Zhang, and M. Kaess. "Keyframe-based dense planar SLAM". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2017, pp. 5110–5117. DOI: `10.1109/ICRA.2017.7989597`.

[40] A. S. Huang, E. Olson, and D. C. Moore. "LCM: Lightweight Communications and Marshalling". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2010, pp. 4057–4062. DOI: `10.1109/IROS.2010.5649358`.

[41] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. "Visual odometry and mapping for autonomous flight using an RGB-D camera". In: *International Symposium on Robotics Research (ISRR)*. Vol. 2. 2011.

[42] G. Huang, M. Kaess, J. J. Leonard, and S. I. Roumeliotis. "Analytically-selected multi-hypothesis incremental MAP estimation". In: 2013, pp. 6481–6485. DOI: `10.1109/ICASSP.2013.6638914`.

[43] V. Ila, J. M. Porta, and J. Andrade-Cetto. “Information-Based Compact Pose SLAM”. In: *IEEE Trans. Robotics* 26.1 (2010), pp. 78–93. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2034435.

[44] *Immersive 3D spaces for real-world applications | Matterport*. URL: https://matterport.com/.

[45] M. Jancosek and T. Pajdla. “Multi-view reconstruction preserving weakly-supported surfaces”. In: *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*. 2011, pp. 3121–3128. DOI: 10.1109/CVPR.2011.5995693.

[46] L. P. Kaelbling and T. Lozano-Pérez. “Unifying perception, estimation and action for mobile manipulation via belief space planning”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2012, pp. 2952–2959. DOI: 10.1109/ICRA.2012.6225237.

[47] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[48] M. Kaess. “Simultaneous localization and mapping with infinite planes”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2015, pp. 4605–4611. DOI: 10.1109/ICRA.2015.7139837.

[49] M. Kaess, A. Ranganathan, and F. Dellaert. “iSAM: Incremental Smoothing and Mapping”. In: *IEEE Trans. Robotics* 24.6 (Dec. 2008), pp. 1365–1378.

[50] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. “iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree”. In: *Intl. J. of Robotics Research, IJRR* 31.2 (Feb. 2012), pp. 216–235.

[51] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. “iSAM2: Incremental smoothing and mapping using the Bayes tree”. In: *Intl. J. of Robotics Research* 31.2 (2012), pp. 216–235.

[52] N. Keivan, A. Patron-Perez, and G. Sibley. “Asynchronous adaptive conditioning for visual-inertial SLAM”. In: *Experimental Robotics*. Springer. 2016, pp. 309–321.

[53] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. “Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion”. In: *Intl. Conf. on 3D Vision (3DV)*. 2013, pp. 1–8.

[54] C. Kerl, J. Sturm, and D. Cremers. “Dense Visual SLAM for RGB-D Cameras”. In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*. 2013.

[55] C. Kerl, J. Sturm, and D. Cremers. "Robust Odometry Estimation for RGB-D Cameras". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013.

[56] A. Kim and R. M. Eustice. "Perception-driven navigation: Active visual SLAM for robotic area coverage". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 3196–3203. DOI: 10.1109/ICRA.2013.6631022.

[57] Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun. "Multi-view image and ToF sensor fusion for dense 3D reconstruction". In: *Intl. Conf. on Computer Vision Workshops (ICCV Workshops)*. 2009, pp. 1542–1549. DOI: 10.1109/ICCVW.2009.5457430.

[58] G. Klein and D. Murray. "Improving the Agility of Keyframe-Based SLAM". In: *Eur. Conf. on Computer Vision (ECCV)*. 2008, pp. 802–815. ISBN: 978-3-540-88688-4.

[59] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*. 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[60] K. Konolige, M. Agrawal, and J. Sola. "Large-scale visual odometry for rough terrain". In: *Robotics research*. Springer, 2010, pp. 201–212.

[61] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A general framework for graph optimization". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[62] Y. Latif, C. Cadena, and J. Neira. "Robust loop closing over time for pose graph SLAM". In: *Intl. J. of Robotics Research* 32.14 (2013), pp. 1611–1626. DOI: 10.1177/0278364913498910.

[63] S. M. LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: *Tech. Rep.* (1998).

[64] S. M. LaValle and J. J. Kuffner Jr. "Randomized kinodynamic planning". In: *Intl. J. of Robotics Research* 20.5 (2001), pp. 378–400.

[65] F. Le Gall. "Powers of Tensors and Fast Matrix Multiplication". In: *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*. ISSAC '14. Kobe, Japan: ACM, 2014, pp. 296–303. ISBN: 978-1-4503-2501-1. DOI: 10.1145/2608628.2608664. URL: http://doi.acm.org/10.1145/2608628.2608664.

[66] G. H. Lee, F. Fraundorfer, and M. Pollefeys. "Robust pose-graph loop-closures with expectation-maximization". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2013, pp. 556–563.

[67] T. Lee, S. Lim, S. Lee, S. An, and S. Oh. “Indoor mapping using planes extracted from noisy RGB-D sensors”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 1727–1733. DOI: `10.1109/IROS.2012.6385909`.

[68] T. Lemaire and S. Lacroix. “Monocular-vision based SLAM using Line Segments”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2007, pp. 2791–2796. DOI: `10.1109/ROBOT.2007.363894`.

[69] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *Intl. J. of Robotics Research* 34.3 (2015), pp. 314–334.

[70] Y. Li, S. Li, Q. Song, H. Liu, and M. Q. . Meng. “Fast and Robust Data Association Using Posterior Based Approximate Joint Compatibility Test”. In: *IEEE Transactions on Industrial Informatics* 10.1 (2014), pp. 331–339. ISSN: 1551-3203. DOI: `10.1109/TII.2013.2271506`.

[71] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: `10.1023/B:VISI.0000029664.99615.94`. URL: `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

[72] W. L. D. Lui and R. Jarvis. “An active visual loop closure detection and validation system for topological SLAM”. In: *Australasian Conference on Robotics and Automation*. 2010.

[73] L. Ma, C. Kerl, J. Stueckler, and D. Cremers. “CPA-SLAM: Consistent Plane-Model Alignment for Direct RGB-D SLAM”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2016.

[74] L. Ma, J. M. Falquez, S. McGuire, and G. Sibley. “Large scale dense visual inertial SLAM”. In: *Field and Service Robotics*. Springer. 2016, pp. 141–155.

[75] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. “Fusion++: Volumetric Object-Level SLAM”. In: *Intl. Conf. on 3D Vision (3DV)*. 2018, pp. 32–41. DOI: `10.1109/3DV.2018.00015`.

[76] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. “SemanticFusion: Dense 3D semantic mapping with convolutional neural networks”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2017, pp. 4628–4635. DOI: `10.1109/ICRA.2017.7989538`.

[77] M. Meilland and A. I. Comport. “On unifying key-frame and voxel-based dense visual SLAM at large scales”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2013, pp. 3677–3683. DOI: `10.1109/IROS.2013.6696881`.

[78] M. Montemerlo and S. Thrun. “Simultaneous localization and mapping with unknown data association using FastSLAM”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. Vol. 2. 2003, pp. 1985–1991. DOI: 10.1109/ROBOT.2003.1241885.

[79] H. P. Moravec. “Sensor Fusion in Certainty Grids for Mobile Robots”. In: *AI Magazine* 9 (1988), pp. 61–74.

[80] B. Mu, S. Y. Liu, L. Paull, J. Leonard, and J. P. How. “SLAM with objects using a nonparametric pose graph”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 4602–4609. DOI: 10.1109/IROS.2016.7759677.

[81] R. Mur-Artal and J. D. Tardós. “Visual-Inertial Monocular SLAM With Map Reuse”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2653359.

[82] R. Mur-Artal, J. Montiel, and J. D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Trans. Robotics* 31.5 (2015), pp. 1147–1163.

[83] J. Neira and J. D. Tardos. “Data association in stochastic mapping using the joint compatibility test”. In: *IEEE Trans. Robotics and Automation* 17.6 (2001), pp. 890–897. ISSN: 1042-296X. DOI: 10.1109/70.976019.

[84] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *Intl. Conf. on Computer Vision (ICCV)*. 2011, pp. 2320–2327.

[85] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In: *IEEE ISMAR*. IEEE, 2011. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=155378.

[86] P. Newman and K. Ho. “SLAM-Loop Closing with Visually Salient Features”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2005, pp. 635–642. DOI: 10.1109/ROBOT.2005.1570189.

[87] V. Nguyen, A. Harati, and R. Siegwart. “A lightweight SLAM algorithm using Orthogonal planes for indoor mobile robotics”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2007, pp. 658–663. DOI: 10.1109/IROS.2007.4399512.

[88] V. Nguyen, A. Harati, A. Martinelli, R. Siegwart, and N. Tomatis. “Orthogonal SLAM: a Step toward Lightweight Indoor Autonomous Navigation”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2006, pp. 5007–5012. DOI: 10.1109/IROS.2006.282527.

[89] E. Olson and Y. Li. "IPJC: The Incremental Posterior Joint Compatibility Test for Fast Feature Cloud Matching". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, Oct. 2012, pp. 3467–3474.

[90] E. Olson and P. Agarwal. "Inference on networks of mixtures for robust robot mapping". In: *Intl. J. of Robotics Research* 32.7 (2013), pp. 826–840.

[91] *Open Source Computer Vision Library | OpenCV*. URL: `https://github.com/opencv/opencv`.

[92] *OpenMVS | open Multi-View Stereo reconstruction library*. URL: `https://github.com/cdcseacave/openMVS/wiki`.

[93] N. Palomeras, M. Carreras, and J. Andrade-Cetto. "Active SLAM for Autonomous Underwater Exploration". In: *Remote Sensing* 11 (Nov. 2019), p. 2827. DOI: `10.3390/rs11232827`.

[94] C. Papachristos, S. Khattak, and K. Alexis. "Uncertainty-aware receding horizon exploration and mapping using aerial robots". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2017, pp. 4568–4575. DOI: `10.1109/ICRA.2017.7989531`.

[95] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger, and A. Birk. "Fast 3D mapping by matching planes extracted from range sensor point-clouds". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2009, pp. 1150–1155. DOI: `10.1109/IROS.2009.5354061`.

[96] S. Pathak, A. Thomas, and V. Indelman. "Nonmyopic data association aware belief space planning for robust active perception". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2017, pp. 4487–4494. DOI: `10.1109/ICRA.2017.7989520`.

[97] S. Pathak, A. Thomas, and V. Indelman. "A unified framework for data association aware robust belief space planning and perception". In: *Intl. J. of Robotics Research* 37.2-3 (2018), pp. 287–315. DOI: `10.1177/0278364918759606`. eprint: `https://doi.org/10.1177/0278364918759606`. URL: `https://doi.org/10.1177/0278364918759606`.

[98] S. Pathak, A. Thomas, A. Feniger, and V. Indelman. "Robust active perception via data-association aware belief space planning". In: *Eur. Conf. on AI (ECAI)*. 2016.

[99] B. Peasley, S. Birchfield, A. Cunningham, and F. Dellaert. "Accurate on-line 3D occupancy grids using Manhattan world constraints". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 5283–5290. DOI: `10.1109/IROS.2012.6386157`.

[100] D. Perea Strm, I. Bogoslavskyi, and C. Stachniss. “Robust Exploration and Homing for Autonomous Robots”. In: *J. of Robotics and Autonomous Systems* 90.C (Apr. 2017), pp. 125–135. ISSN: 0921-8890. DOI: 10.1016/j.robot.2016.08.015. URL: https://doi.org/10.1016/j.robot.2016.08.015.

[101] T. Pfeifer and P. Protzel. “Robust Sensor Fusion with Self-tuning Mixture Models”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2018.

[102] M. Pfingsthorn and A. Birk. “Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints”. In: *Intl. J. of Robotics Research* 35.6 (2016), pp. 601–630. DOI: 10.1177/0278364915585395. eprint: https://doi.org/10.1177/0278364915585395.

[103] M. Pizzoli, C. Forster, and D. Scaramuzza. “REMODE: Probabilistic, monocular dense reconstruction in real time”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 2609–2616.

[104] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. “PL-SLAM: Real-time monocular visual SLAM with points and lines”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2017, pp. 4503–4508. DOI: 10.1109/ICRA.2017.7989522.

[105] E. Rosten, R. Porter, and T. Drummond. “Faster and Better: A Machine Learning Approach to Corner Detection”. In: *IEEE Trans. Pattern Anal. Machine Intell.* 32.1 (2010), pp. 105–119. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.275.

[106] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *Intl. Conf. on Computer Vision (ICCV)*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[107] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. “Towards 3D point cloud based object maps for household environments”. In: *Robotics and Autonomous Systems* 56.11 (2008), pp. 927–941.

[108] S. C. S. Agarwal A. Tamjidi. “Motion planning for active data association and localization in non-Gaussian belief spaces”. In: *Workshop on the Algorithmic Foundations of Robotics*. 2016.

[109] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal. “Normal Distributions Transform Occupancy Maps: Application to large-scale online 3D mapping”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 2233–2238. DOI: 10.1109/ICRA.2013.6630878.

[110] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects”. In: *Proc.*

IEEE Int. Conf. Computer Vision and Pattern Recognition. 2013, pp. 1352–1359. DOI: 10.1109/CVPR.2013.178.

[111] R. Salas-Moreno, B. Glocken, P. Kelly, and A. Davison. “Dense planar SLAM”. In: *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 157–164. DOI: 10.1109/ISMAR.2014.6948422.

[112] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. “A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms”. In: *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*. Vol. 1. 2006, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

[113] P. Smith, I. Reid, and A. Davison. “Real-Time Monocular SLAM with Straight Lines”. In: *Proc. British Machine Vision Conference*. Edinburgh, 2006, pp. 17–26.

[114] E. J. Sondik. “The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs”. In: *Oper. Res.* 26.2 (Apr. 1978), pp. 282–304. ISSN: 0030-364X. DOI: 10.1287/opre.26.2.282. URL: http://dx.doi.org/10.1287/opre.26.2.282.

[115] S. Song, M. Chandraker, and C. C. Guest. “Parallel, real-time monocular visual odometry”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 4698–4705. DOI: 10.1109/ICRA.2013.6631246.

[116] C. Stachniss, D. Hahnel, and W. Burgard. “Exploration with active loop-closing for FastSLAM”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 2. 2004, 1505–1510 vol.2. DOI: 10.1109/IROS.2004.1389609.

[117] F. Steinbrucker, J. Sturm, and D. Cremers. “Real-time visual odometry from dense RGB-D images”. In: *Intl. Conf. on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 719–722. DOI: 10.1109/ICCVW.2011.6130321.

[118] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. Oct. 2012, pp. 573–580.

[119] N. Sünderhauf and P. Protzel. “Switchable constraints for robust pose graph SLAM”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 1879–1884. DOI: 10.1109/IROS.2012.6385590.

[120] N. Sünderhauf and P. Protzel. “Towards a robust back-end for pose graph SLAM”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2012, pp. 1254–1261. DOI: 10.1109/ICRA.2012.6224709.

[121] S. Suresh, P. Sodhi, J. G. Mangelson, D. Wettergreen, and M. Kaess. “Active SLAM using 3D Submap Saliency for Underwater Volumetric Exploration”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2020.

[122] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng. “Point-plane SLAM for hand-held 3D sensors”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 5182–5189. DOI: 10.1109/ICRA.2013.6631318.

[123] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. “Robust monocular SLAM in dynamic environments”. In: *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*. 2013, pp. 209–218. DOI: 10.1109/ISMAR.2013.6671781.

[124] T. Tao, Y. Huang, F. Sun, and T. Wang. “Motion Planning for SLAM Based on Frontier Exploration”. In: *2007 International Conference on Mechatronics and Automation*. 2007, pp. 2120–2125. DOI: 10.1109/ICMA.2007.4303879.

[125] S. Thrun and M. Montemerlo. “The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures”. In: *Intl. J. of Robotics Research* 25 (May 2006), pp. 403–429. DOI: 10.1177/0278364906065387.

[126] A. Trevor, J. Rogers, and H. Christensen. “Planar surface SLAM with 3D and 2D sensors”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2012, pp. 3041–3048. DOI: 10.1109/ICRA.2012.6225287.

[127] S. Tully, G. Kantor, H. Choset, and F. Werner. “A multi-hypothesis topological SLAM approach for loop closing on edge-ordered graphs”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2009, pp. 4943–4948. DOI: 10.1109/IROS.2009.5354255.

[128] R. Valencia, J. Valls Miró, G. Dissanayake, and J. Andrade-Cetto. “Active Pose SLAM”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 1885–1891. DOI: 10.1109/IROS.2012.6385637.

[129] J. Wang and B. Englot. “Robust exploration with multiple hypothesis data association”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2018.

[130] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. “Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE. 2012, pp. 957–964.

[131] S. Wenhardt, B. Deutsch, E. Angelopoulou, and H. Niemann. “Active Visual Object Reconstruction using D-, E-, and T-Optimal Next Best Views”. In: *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*. 2007, pp. 1–7. DOI: 10.1109/CVPR.2007.383363.

[132] E. Westman, A. Hinduja, and M. Kaess. “Feature-Based SLAM for Imaging Sonar with Under-Constrained Landmarks”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2018, pp. 1–9. DOI: 10.1109/ICRA.2018.8461004.

[133] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald. “Real-time Large Scale Dense RGB-D SLAM with Volumetric Fusion”. In: *Intl. J. of Robotics Research* 34.4-5 (2015), pp. 598–626.

[134] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison. “ElasticFusion: Dense SLAM Without A Pose Graph”. In: *Proceedings of Robotics: Science and Systems*. Rome, Italy, 2015. DOI: 10.15607/RSS.2015.XI.001.

[135] B. Yamauchi. “A frontier-based approach for autonomous exploration”. In: *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. ’Towards New Computational Principles for Robotics and Automation’*. 1997, pp. 146–151. DOI: 10.1109/CIRA.1997.613851.

[136] Z. Zhang. “Microsoft Kinect Sensor and Its Effect”. In: *IEEE MultiMedia* 19.2 (2012), pp. 4–10. ISSN: 1070-986X. DOI: 10.1109/MMUL.2012.24.

[137] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu. “StructSLAM: Visual SLAM With Building Structure Lines”. In: *IEEE Transactions on Vehicular Technology* 64.4 (2015), pp. 1364–1375. ISSN: 0018-9545. DOI: 10.1109/TVT.2015.2388780.

[138] X. Zuo, X. Xie, Y. Liu, and G. Huang. “Robust visual SLAM with point and line features”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 1775–1782. DOI: 10.1109/IROS.2017.8205991.